15

Multi-dimensional Information Spaces as Memory Structures for Intelligent Data Processing in GMES

15.1 Memory management

Memory management is a complex field of computer science. Over the years, many techniques have been developed to make it more efficient [Ravenbrook, 2010]. Memory management is usually divided into three areas: hardware, operating system, and application, although the distinctions are a little fuzzy. In most computer systems, all three are present to some extent, forming layers between the user's program and the actual memory hardware:

- **Memory management at the hardware level** is concerned with the electronic devices that actually store data. This includes things like RAM and memory caches;
- **Memory in the operating system** must be allocated to user programs, and reused by other programs when it is no longer required. The operating system can pretend that the computer has more memory than it actually does, and that each program has the machine's memory to itself. Both of these are features of virtual memory systems;
- **Application memory management** involves supplying the memory needed for a program's objects and data structures from the limited resources available, and recycling that memory for reuse when it is no longer required. Because in general, application programs cannot predict in advance how much memory they are going to require, they need additional code to handle their changing memory requirements.

Application memory management combines two related tasks:

- **Allocation**: when the program requests a block of memory, the memory manager must allocate that block out of the larger blocks it has received from the operating system. The part of the memory manager that does this is known as the allocator;
- **Recycling**: when memory blocks have been allocated, but the data they contain is no longer required by the program, the blocks can be recycled for reuse. There are two approaches to recycling memory: either the programmer must decide when memory can be reused (known as manual memory management); or the memory manager must be able to work it out (known as automatic memory management).

The progress in memory management gives the possibility to allocate and recycle not directly blocks of the memory but structured regions or fields corresponding to some types of data. In such case we talk about corresponded "***access methods***".

15.2 Access methods

The Access Methods (AM) have been available from the beginning of the development of computer peripheral devices. As many devices so many possibilities for developing different AM there exist. In the beginning, the AM were functions of the Operational Systems Core or so called Supervisor, and were executed via corresponding macro-commands in the assembler languages [Stably, 1970] or via corresponding input/output operators in the high level programming languages like FORTRAN, COBOL, PL/I, etc.

The establishment of the first databases in the sixties of the previous century caused gradually accepting the concepts "physical" as well as "logical" organization of the data [CODASYL, 1971], [Martin, 1975]. In 1975, the concepts "access method", "physical organization" and "logical organization" became clearly separated.

In the same time, Christopher Date specially remarked:

"The Database Management System (DBMS) does not know anything about:

a) how physical records (blocks) are disposed;
b) how the stored fields are integrated in the records (nevertheless that in many cases it is obvious because of their physical disposition);
c) how the sorting is realized (for instance it may be realized on the base of physical sequence, using an index or by a chain of pointers);
d) how the direct access is realized (i.e. by index, sequential scanning or hash addressing).

This information is a part of the structures for data storing but it is used by the access method but not by the DBMS" [Date, 1975].

Every access method presumes an exact organization of the file, which it is operating with and is not related to the interconnections between the files, respectively – between the records of one file and that in the others files. These interconnections are controlled by the physical organization of the DBMS.

Therefore, in the DBMS we may distinguish four levels:

1. Access methods at the core (supervisor) of the operation system.
2. Specialized access methods which upgrade these at the core of the operating system.
3. Physical organization of the DBMS.
4. Logical organization of the DBMS.

During the eighties, the total growing of the research and developments in the computers' field, especially in image processing, data mining and mobile support cause impetuous progress of establishing convenient "spatial information structures" and "spatial-temporal information structures" and corresponding access methods. From different points of view, this period has been presented in [Ooi et al, 1993], [Gaede and Günther, 1998], [Arge, 2002], [Mokbel et al, 2003], [Moënne-Loccoz, 2005]. Usually the "one-dimensional" (linear) AM are used in the classical applications, based on the alphanumerical information, whereas the "multi-dimensional" (spatial) methods are aimed to serve the work with graphical, visual, multimedia information.

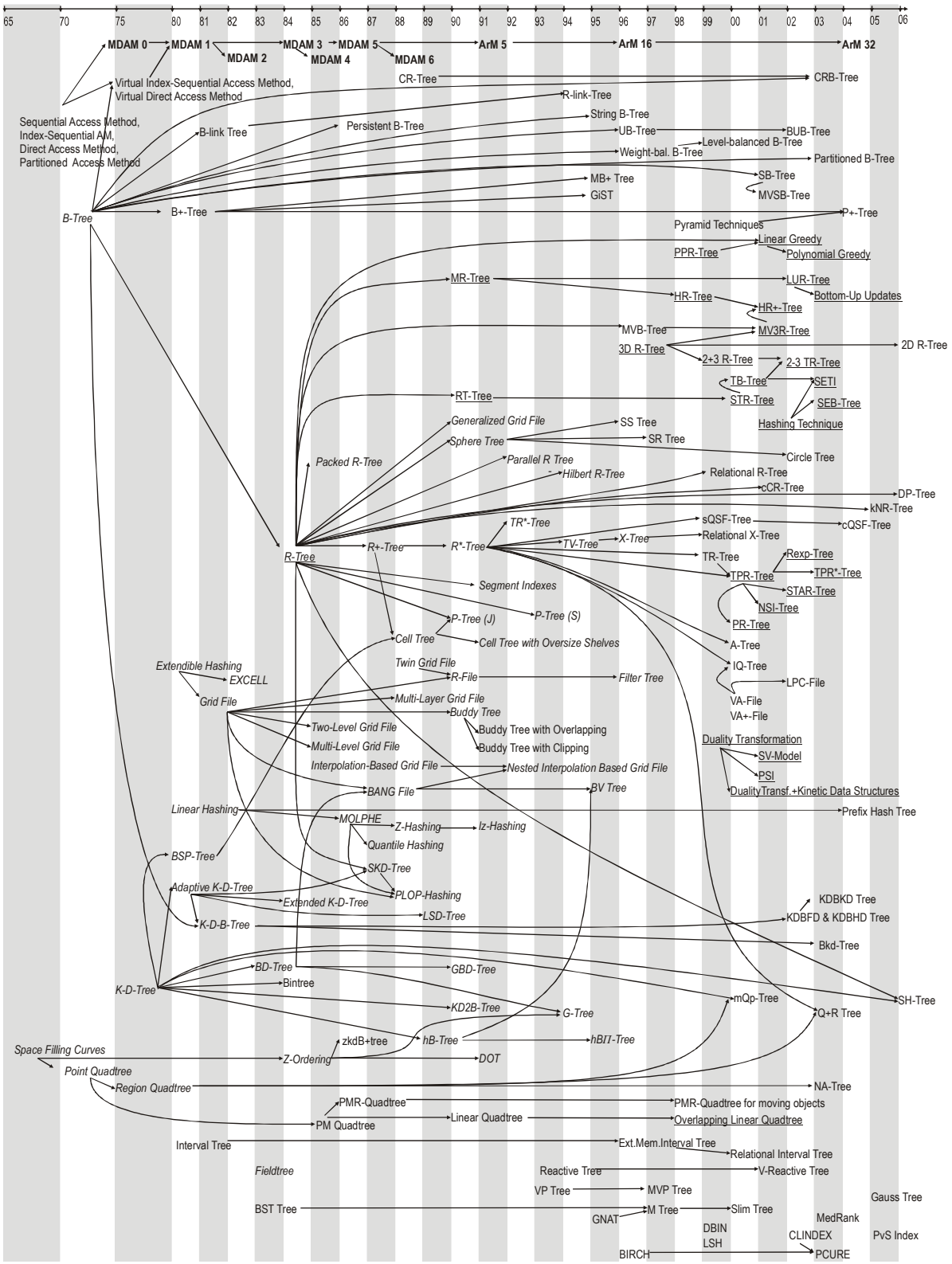

Figure 175. Genesis of the Access Methods and their modifications
extended variant of [Gaede and Günther, 1998] and [Mokbel et al, 2003] presented in [Markov et al, 2008]

15.2.1 Interconnections between raised access methods

Maybe one of the most popular analyses is given in [Gaede and Günther, 1998]. The authors presented a scheme of the genesis of the basic multi-dimensional AM and theirs modifications. This scheme firstly was proposed in [Ooi et al, 1993] and it was expanded in [Gaede and Günther, 1998]. An extension in direction to the multi-dimensional spatio-temporal access methods was given in [Mokbel et al, 2003].

The survey [Markov et al, 2008] presents a new variant of this scheme (Figure 175), where the new access methods, created after 1998, are added. A comprehensive bibliography of corresponded articles, where the methods are firstly presented is given.

15.2.2 The taxonomy of the access methods

From the point of view of the served area, the access methods, presented on Figure 175, may be classified as follows (Figure 176): One-dimensional AM; Multidimensional Spatial AM; Metric Access Methods; High Dimensional Access Methods; and Spatio-Temporal Access Methods.

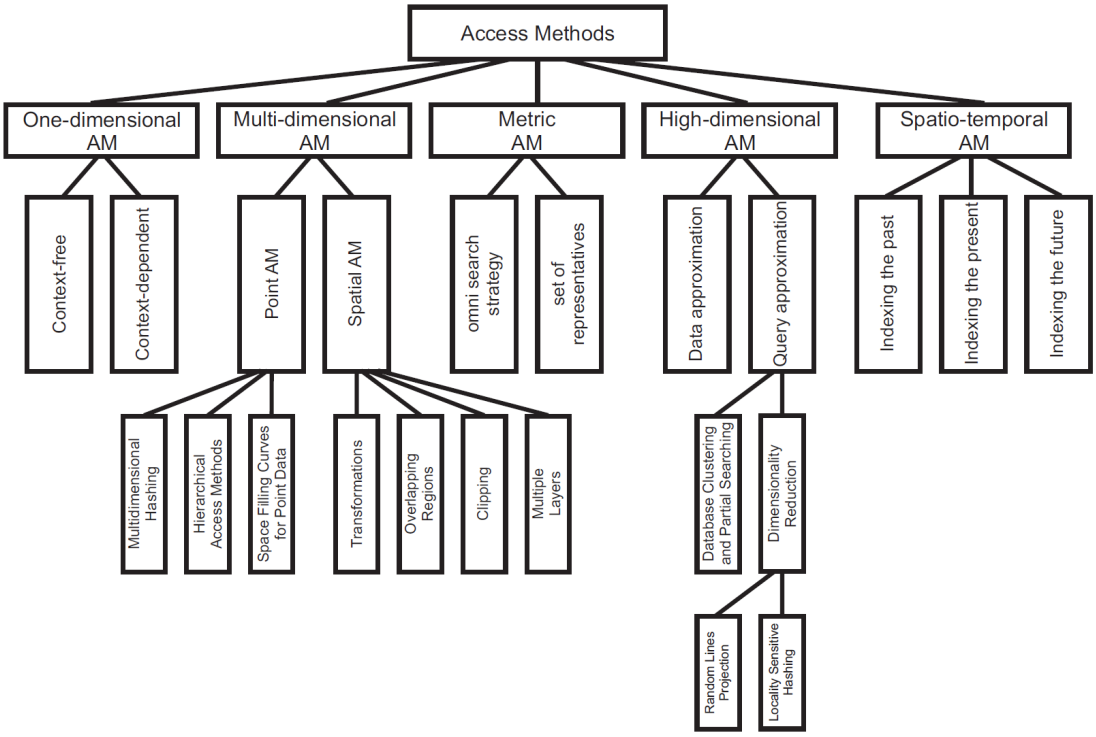

Figure 176. Taxonomy of the Access Methods

✓ One-dimensional access methods

One-dimensional AM are based on the concept "record". The "record" is a logical sequence of fields, which contain data eventually connected to unique identifier (a "key"). The identifier (key) is aimed to distinguish one sequence from another [Stably, 1970]. The records are united in the sets, called "files". There exist three basic formats of the records – with fixed, variable, and undefined length.

In the ***context-free methods,*** the storing of the records is not connected to their content and depends only on external factors – the sequence, disk address, or position in the file. The necessity of stable file systems in the operating systems does not allow a great variety of the context-free AM.

There are three main types well known from sixties and seventies: *Sequential Access Method (SAM); Direct Access Method (DAM) and Partitioned Access Method (PAM)* [IBM, 1965-68].

The main idea of the ***context-depended AM*** is that a part of the record is selected as a key, which is used for making decision where to store the record and how to search it. This way the content of the record influences the access to the record.

Historically, from the sixties of the previous century on, the attention is directed mainly to this type of AM. Modern DBMS are built using context-depended AM such as: unsorted sequential files with records with keys; sorted files with fixed record length; static or dynamic hash files; index file and files with data; clustered indexed tables [Connolly and Begg, 2002].

✓ Multidimensional spatial access methods

Multidimensional Spatial Access Methods are developed to serve information about spatial objects, approximated with points, segments, polygons, polyhedrons, etc. The implementations are numerous and include traditional multi-attributive indexing, geographical and/or GMES information systems, and spatial databases, content indexing in multimedia databases, etc.

From the point of view of the spatial databases, the access methods can be split into two main classes of access methods – Point Access Methods and Spatial Access Methods [Gaede and Günther, 1998].

Point Access Methods are used for organizing multidimensional point objects. Typical instances are traditional records, where every attribute of the relation corresponds to one dimension. These methods can be separated in three basic groups:

- Multidimensional Hashing (for instance Grid File and its varieties, EXCELL, Twin Grid File, MOLPHE, Quantile Hashing, PLOP-Hashing, Z-Hashing, etc);
- Hierarchical Access Methods (includes such methods as KDB-Tree, LSD-Tree, Buddy Tree, BANG File, G-Tree, hB-Tree, BV-Tree, etc.);
- Space Filling Curves for Point Data (like Peano curve, N-trees, Z-Ordering, etc).

Spatial Access Methods are used for working with objects, which have an arbitrary form. The main idea of the spatial indexing of non-point objects is to use an approximation of the geometry of the examined objects as more simple forms. The most used approximation is Minimum Bounding Rectangle (MBR), i.e. minimal rectangle, which sides are parallel of the coordinate axes and completely include the object. There exist approaches for approximation with Minimum Bounding Spheres (SS Tree) or other polytopes (Cell Tree), as well as their combinations (SR-Tree).

The usual problem when one operates with spatial objects is their overlapping. There are different techniques to avoid this problem. From the point of view of the techniques for the organization of the spatial objects, Spatial Access Methods can be split in four main groups:

- Transformation – this technique uses transformation of spatial objects to points in the space with more or less dimensions. Most of them spread out the space using space filling curves (Peano Curves, z-ordering, Hilbert curves, Gray ordering, etc.) and then use some point access method upon the transformed data set;
- Overlapping Regions – here the data sets are separated in groups; different groups can occupy the same part of the space, but every space object associates with only one of the groups. The access methods of this category operate with data in their primary space (without any transformations) eventually in overlapping segments. Methods which use this technique includes R-Tree, R-link-

Tree, Hilbert R-Tree, R*-Tree, Sphere Tree, SS-Tree, SR-Tree, TV-Tree, X-Tree, P-Tree of Schiwietz, SKD-Tree, GBD-Tree, Buddy Tree with overlapping, PLOP-Hashing, etc.;

- Clipping – this technique uses the clipping of one object to several sub-objects, which will be stored. The main goal is to escape overlapping regions. However, this advantage can lead to the tearing of the objects, extending the resource expenses, and decreasing the productivity of the method. Representatives of this technique are R+-Tree, Cell-Tree, Extended KD-Tree, Quad-Tree, etc.;
- Multiple Layers – this technique can be considered as a variant of the techniques of Overlapping Regions, because the regions from different layers can overlap. Nevertheless, there exist some important differences: first – the layers are organized hierarchically; second – every layer splits the primary space in a different way; third – the regions of one layer never overlaps; fourth – the data regions are separated from the space extensions of the objects. Instances for these methods are Multi-Layer Grid File, R-File, etc.

✓ Metric access methods

Metric Access Methods deal with relative distances of data points to chosen points, named anchor points, vantage points or pivots [Moënne-Loccoz, 2005]. These methods are designed to limit the number of distance computation, calculating first distances to anchors, and then finding the searched point in a narrowed region. These methods are preferred when the distance is highly computational, as e.g. for the dynamic time warping distance between time series. Representatives of these methods are: Vantage Point Tree (VP Tree), Bisector Tree (BST-Tree), Geometric Near-Neighbor Access Tree (GNNAT), as well as the most effective from this group – Metric Tree (M-Tree) [Chavez et al, 2001].

✓ High dimensional access methods

Increasing the dimensionality strongly aggravates the qualities of the multidimensional access methods. Usually these methods exhaust their possibilities at dimensions around 15. Only X-Tree reaches the boundary of 25 dimensions, after which this method gives worse results then sequential scanning [Chakrabarti, 2001].

The exit of this situation is based on the data approximation and query approximation in sequential scan. These methods form a new group of access methods – High Dimensional Access Methods.

Data approximation is used in VA-File, VA+-File, LPC-File, IQ-Tree, A-Tree, P+-Tree, etc.

For query approximation, two strategies can be used:

- Examine only a part of the database, which is more probably to contain the resulting set – as a rule these methods are based on the clustering of the database. Some of these methods are: DBIN, CLINDEX, PCURE;
- Splitting the database to several spaces with fewer dimensions and searching in each of them. Here two main methods are used:
  1. Random Lines Projection. Representatives of this approach are MedRank, which uses B+-Tree for indexing every arbitrary projection of the database, and PvS Index, which consist of combination of iterative projections and clustering.

2. Locality Sensitive Hashing, which is based on the set of local-sensitive hashing functions [Moënne-Loccoz, 2005].

✓ Spatio-temporal access methods

The Spatio-Temporal Access Methods have additional defined time dimensioning [Mokbel et al, 2003]. They operate with objects, which change their form and/or position during the time. According to position of time interval in relation to present moment, the Spatio-Temporal Access Methods are divided as follow:

- ***indexing the past***. These methods operate with historical spatio-temporal data. The problem here is the continuous increase of the information over time. To overcome the overflow of the data space two approaches are used – sampling the stream data at certain time position or updating the information only when data is changed. Spatio-temporal indexing schemes for historical data can be split in three categories:
  - the first category includes methods that manage spatial and temporal aspects into already existing spatial methods;
  - the second category can be explained as snapshots of the spatial information in each time instance;
  - the third category focuses on trajectory-oriented queries, while spatial dimension lag on second priority.

  Representatives of this group are: RT-Tree, 3DR-Tree, STR-Tree, MR-Tree, HR-Tree, HR+-Tree, MV3R-Tree, PPR-Tree, TB-Tree, SETI, SEB-Tree;
- ***indexing the present***. In contrast to previous methods, where all movements are known, here the current positions are neither stored nor queried. Some of the methods, which answer the questions of the current position of the objects are 2+3R-Tree, 2-3TR-Tree, LUR-Tree, Bottom-Up Updates, etc.;
- ***indexing the future***. These methods have to answer the questions about the current and future position of a moving object – here are embraced the methods like PMR-Quadtree for moving objects, Duality Transformation, SV-Model, PSI, PR-Tree, TPR-Tree, TPR*-tree, NSI, VCIR-Tree, STAR-Tree, R^{EXP}-Tree.

The survey of the access methods suggests that the context-free multi-dimensional access methods practically are not available. A step in developing such methods is the Multi-domain Information Model and corresponding Multi-domain Access Method introduced in [Markov, 1984] [Markov, 2004]. It will be outlined later in this chapter.

15.3 Multi-dimensional numbered information spaces

A simple idea is coming from this reasoning: "Why do we have no possibility to use "external memory arrays" with one, two, three, and more dimensions?" [Markov, 2004]. Additionally, by analogy with the Relation Model [Codd, 1970], we may want to have multidimensional relations. Such theory still does not exist, but it is possible to be realized.

The proposed external memory structure is based on the numbering as a main approach. The idea consists in replacing the (symbol or real; point or interval) values of the objects' attributes with integer numbers of the elements of corresponding ordered sets. This way, each object will be

described by a vector of integer values, which may be used as the co-ordinate address in the multi-dimensional information space.

In other words, the process of replacing the names by numbers permits the use of mathematical functions and address vectors for accessing the information instead of search engines.

This type of memory organization is called "Multi-dimensional numbered information spaces". Its advantages have been demonstrated in many practical realizations during more than twenty-five years [Markov, 1984], [Markov, 2004], [Markov, 2005]. In recent years, this kind of memory organization has been implemented in the area of intelligent systems memory structuring for several data mining tasks and especially in the area of association rules mining.

15.3.1 Multi-Domain Information Model (MDIM)

The independence of dimensionality limitations is very important for developing new intelligent systems aimed to process high-dimensional data. To achieve this we need information models and corresponding access method to cross the boundary of the dimensional limitations and to obtain the possibility to work with information spaces with practically unlimited number of dimensions. The first step is to establish context free multi-dimensional models and based on it to develop high-level context depended applications. Examining the state of the art in this area shows that the context-free multi-dimensional information models and access methods practically are not available. One attempt in this direction is establishing the **Multi-Domain Information Model** (**MDIM**) [Markov, 2004] and the corresponding Multi-domain Access Method. Their possibilities for operating with context-free multidimensional data structures will be presented below.

15.3.2 Basic structures of MDIM

Basic structures of MDIM are basic information elements, information spaces, indexes and metaindexes, and aggregates. The definitions of these structures are given below:

✓ Basic information element

The basic information element (BIE) of MDIM is an arbitrary long string of machine codes (bytes). When it is necessary, the string may be parceled out by lines. The length of the lines may be variable.

Let the universal set ***UBIE*** be the set of all BIE.

Let $\boldsymbol{E}_1$ be a set of basic information elements:

$$\boldsymbol{E}_1 = \{e_i \mid e_i \in \boldsymbol{UBIE}, i=1,\ldots, m_1\}.$$

Let $\boldsymbol{\mu}_1$ be a function, which defines a biunique correspondence between elements of the set $\boldsymbol{E}_1$ and elements of the set $\boldsymbol{C}_1$ of positive integer numbers:

$$\boldsymbol{C}_1 = \{c_i \mid c_i \in \boldsymbol{N}, i:=1,\ldots, m_1\},$$

$$\text{i.e. } \boldsymbol{\mu}_1 : \boldsymbol{E}_1 \leftrightarrow \boldsymbol{C}_1.$$

The elements of $\boldsymbol{C}_1$ are said to be numbers (co-ordinates) of the elements of $\boldsymbol{E}_1$.

✓ Information spaces

The triple $\mathbf{S_1} = (\mathbf{E_1}, \boldsymbol{\mu}_1, \mathbf{C_1})$ is said to be a ***numbered information space of range 1*** *(one-dimensional or one-domain information space).*

The triple $\mathbf{S_2} = (\mathbf{E_2}, \mu_2, \mathbf{C_2})$ is said to be a ***numbered information space of range 2*** iff $\mathbf{E_2}$ is a set whose elements are numbered information spaces of range 1 and μ_2 is a function which defines a biunique correspondence between elements of $\mathbf{E_2}$ and elements of the set $\mathbf{C_2}$ of positive integer numbers:

$$\mathbf{C_2} = \{c_j \mid c_j \in \mathbf{N}, j:=1,\ldots,m_2\},$$

$$\text{i.e. } \mu_2 : \mathbf{E_2} \leftrightarrow \mathbf{C_2}.$$

The triple $\mathbf{S_n} = (\mathbf{E_n}, \mu_n, \mathbf{C_n})$ is said to be a ***numbered information space of range n*** *(n-dimensional or multi-domain information space)* iff $\mathbf{E_n}$ is a set whose elements are information spaces of range **n-1** and μ_n is a function which defines a biunique correspondence between elements of $\mathbf{E_n}$ and elements of the set $\mathbf{C_n}$ of positive integer numbers:

$$\mathbf{C_n} = \{c_k \mid c_j \in \mathbf{N}, k:=1,\ldots,m_n\}, \text{ i.e. } \mu_n : \mathbf{E_n} \leftrightarrow \mathbf{C_n}.$$

Every basic information element "e" is considered as an ***information space*** $\mathbf{S_0}$ of range 0. It is clear that the information space $\mathbf{S_0} = (\mathbf{E_0}, \mu_0, \mathbf{C_0})$, is constructed in the same manner as all others:

- the machine codes (bytes) b_i, i=1,…,m_0 are considered as elements of $\mathbf{E_0}$,
- the position p_i (natural number) of b_i in the string e is considered as co-ordinate of b_i, i.e. $\mathbf{C_0} = \{p_k \mid p_k \in \mathbf{N}, k:=1,\ldots,m_0\}$,
- function μ_0 is defined by the physical order of b_i in e and we have:

$$\mu_0 : \mathbf{E_0} \leftrightarrow \mathbf{C_0}$$

In this way, the string $\mathbf{S_0}$ may be considered as a set of ***sub-elements (sub-strings)***. The number and length of the sub-elements may be variable. This option is very helpful but it closely depends on the concrete realizations and it is not considered as a standard characteristic of MDIM.

The information space $\mathbf{S_n}$, which contains all information spaces of a given application is called the ***information base*** of range **n**. Usually, the concept information base without indication of the range is used as generalized concept to denote all available information spaces.

✓ Indexes and metaindexes

The sequence $\boldsymbol{A} = (c_n, c_{n-1}, \ldots, c_1)$ where $c_i \in \boldsymbol{C_i}$, i=1,…,n is called the ***multidimensional space address*** of range **n** of a basic information element. Every space address of range **m**, **m<n**, may be extended to space address of range **n** by adding leading **n-m** zero codes. Every sequence of space addresses $\boldsymbol{A_1}, \boldsymbol{A_2}, \ldots, \boldsymbol{A_k}$, where **k** is arbitrary positive number, is said to be a ***space index***.

A special kind of space index is the ***projection***, which is the analytically given space index. There are two types of projections:

- ***hierarchical projection*** – where the top part of coordinates is fixed and the low part vary for all possible values of coordinates, where non-empty elements exist;
- ***arbitrary projection*** – in this case, it is possible to fix coordinates in arbitrary positions and the other coordinates vary for all possible values of coordinates, where non-empty elements exist.

Every index may be considered as a basic information element, i.e. as a string, and may be stored in a point of any information space. In such case, it will have a multidimensional space address, which may be pointed in the other indexes, and, this way, we may build a hierarchy of indexes. Therefore, every index, which points only to indexes, is called ***meta-index***.

The approach of representing the interconnections between elements of the information spaces using (hierarchies) of meta-indexes is called ***polyindexation***.

✓ **Aggregates**

Let **G** = {$\mathbf{S}_i$ | i=1,..,m} be a set of numbered information spaces.

Let τ={ν_{ij} : $\boldsymbol{S}_i \rightarrow \boldsymbol{S}_j$ | i=*const*, j=1,…m} be a set of mappings of one "main" numbered information space $\mathbf{S}_i \subset \mathbf{G}$, i=const, into the others $\mathbf{S}_j \subset \mathbf{G}$, j=1,…m, and, in particular, into itself.

The couple: **Đ** = (**G**, τ) is said to be an "***aggregate***".

It is clear we can build **m** aggregates using the set **G** because every information space $\mathbf{S}_i \subset \mathbf{G}$, j=1,…,m, may be chosen to be a main information space.

15.3.3 Operations in the MDIM

After defining the information structures, we need to present the operations, which are admissible in the model.

It is clear that the operations are closely connected to the defined structures.

In MDIM, we assume that **all** information elements of **all** information spaces **exist**. If for any $\mathbf{S}_i$: $\mathbf{E}_i=\varnothing \wedge \mathbf{C}_i=\varnothing$, than it is called ***empty.*** Usually, most of the information elements and spaces are empty. This is very important for practical realizations.

✓ **Operation with basic information elements**

Because of the rule that all the structures given above must exist, we only need two operations: (1) updating and (2) getting the value of BIE.

For both types of operations, we need two service operations: (1) getting the length and (2) the positioning in the BIE.

Updating, or simply – writing the element, has several modifications with obvious meaning: writing a BIE as a whole, appending/inserting in a BIE, cutting/replacing a part of a BIE and deleting a BIE.

There is only one operation for getting the value of a BIE, i.e. **Read** a portion from a BIE starting from given position. We may receive the whole BIE if the starting position is the beginning of BIE and the length of the portion is equal to the BIE length.

✓ **Operation with spaces**

With a **single space,** we may do only one operation – clearing (deleting) the space, i.e. replacing all BIE of the space with empty BIE – ∅. After this operation, all BIE of the space will have zero length. Really, the space is cleared via replacing it with empty space.

With **two spaces**, we may provide two operations with two modifications both: (1) copying and (2) moving the first space in the second.

The modifications concern how the BIE in the recipient space are processed. We may have: copy/move with clear and copy/move with merge.

The "clear" modifications first clear the recipient space and after that provide a copy or move operation.

The merge modifications may have two types of processing: destructive or constructive. The ***destructive merging*** may be "conservative" or "alternative". In the conservative approach, the recipient space BIE remain in the result if it is with none zero length. In the other approach – the

donor space BIE remain in the result. In the ***constructive merging*** the result is any composition of the corresponding BIE of the two spaces.

Of course, the move operation deletes the donor space after the operation.

✓ Operation with indexes and metaindexes

The indexes are the main approach for describing the interconnections between the structures.

We may receive the space address of the **next** or **previous, empty** or **non-empty** elements of the space starting from any given co-ordinate. This corresponds to the processing of given hierarchical projections.

Analogically, we may receive the space address of the **nextproj** or **previousproj non-empty** elements of the space for the current address in operation with a given arbitrary projection.

The possibility to count the number of non-empty elements of a given projection is useful for practical realizations.

The operations with indexes are based on usual logical operations between sets. The difference from usual sets is that the information spaces are built by the interconnection between two main sets: the set of co-ordinates and the set of information elements.

This way the operations with indexes may be classified in two main types: context free and context depended operations.

The context free operations defined in the MDIM are based on the classical logical operations – intersection, union, and supplement, but these operations are notso trivial. Because of the complexity of the structure of the information spaces, these operations have at least two principally different realizations based on:

- co-ordinates;
- information elements.

The operations based on co-ordinates are determined by the existence of the corresponding space information elements. Therefore, the values of the co-ordinates of the existing information elements determine the operations.

In the other case, the existing BIE values determine the logical operations.

In both cases, the result of the logical operations is an index.

The context depended operations need special realizations for concrete purposes.

The main information operation is creating the indexes and meta-indexes. This may be very complicated processes and could not be given in advance. The main purpose of the MDIM is to provide the possibility for access to the practically unlimited information space and easy approach for building interconnection between its elements. The goal of the concrete applications is to build tools for creating and operating with the indexes and meta-indexes and to implement these tools in the realization of user requested systems.

For instance, such tools may realize the transfer from one structure to another, information search, sorting, making reports, more complicated information processing, etc. The information operations can be grouped into four sets corresponding to the main information structures: basic information elements, information spaces and index or meta-index structures.

✓ Operations with aggregates

Theory of aggregates may be assumed as an extension of the Relation theory because the relation in the sense of the model of Codd [Codd, 1970] may be represented by the aggregate. It is easy to see that if the aggregation mappings are one-one mappings it will be relation in the sense of the model of Codd. Therefore, we may say that the aggregate is a more universal structure than the relation and the operations with aggregates include those of relation theory. The relation algebra is a very good starting point to understand the algebra of aggregates. The new element is that the mappings of different aggregates may be not one-one mappings. This field is not investigated until now.

✓ Multi-domain access method ArM 32

The program realization of MDIM is called Multi-Domain Access Method. For a long period, it has been used as a basis for the organization of various information bases. There exist several realizations of MDIM for different hardware and/or software platforms. The most resent one is the FOI Archive Manager – ArM. One of the first goals of the development of ArM was to represent the digitalized military defense situation, which is characterized by a variety of complex objects and events, which occur in the space and time and have a long period of variable existence. The great number of layers, aspects, and interconnections of the real situation may be represented only by information space hierarchy. In addition, the different types of users with individual access rights and needs insist on the realization of a special tool for organizing such information base. Over the years, the efficiency of ArM is proved in wide areas of information service of enterprise managements and accounting. Organizing the datum in appropriate multi-dimensional information space model permits omitting the heavy work of creating of OLAP structures [Markov, 2005].

The newest ArM Version No.9, called ArM32, is developed for MS Windows, and realizes the proposed algorithms.

The ArM32 elements are organized in numbered information spaces with variable ranges. There is no limit for the ranges of the spaces. Every element may be accessed by a corresponding multidimensional space address (coordinates) given via coordinate array of type cardinal. At the first place of this array, the space range needs to be given. Therefore, we have two main constructs of the physical organizations of ArM32 – numbered information spaces and elements.

In ArM32, the length of the string may vary from 0 up to 1G bytes. There is no limit for the number of strings in an archive but their total length plus internal indexes could not exceed 4G bytes in a single file.

The main ArM32 operations with basic information elements are: ArmRead (reading a part or a whole element); ArmWrite (writing a part or a whole element); ArmAppend (appending a string to an element); ArmInsert (inserting a string into an element); ArmCut (removing a part of an element); ArmReplace (replacing a part of an element); ArmDelete (deleting an element); ArmLength (returns the length of the element in bytes).

The operations over the spaces are: DelSpace (deleting the space); CopySpace and MoveSpace (copying/moving the first space in the second in the frame of one file); ExportSpace (copying one space from one file the other space, which is located in other file).

The operations, aimed to serve the hierarchical projections are: ArmNextPresent, ArmPrevPresent, ArmNextEmpty, ArmPrevEmpty. For arbitrary projections the operations are: ArmNextProj and ArmPrevProj.

The operations, which create indexes, are: ArmSpaceIndex (returns the space index of the non-empty structures in the given information space; ArmProjIndex (gives the space index of basic information elements of a given hierarchical or arbitrary projection).

The service operations for counting non-empty elements or subspaces are correspondingly: ArmSpaceCount (returns the number of the non-empty structures in given information space); ArmProjCount (gives the number of elements of given (hierarchical or arbitrary) projection).

The ArM32 logical operations defined in the multi-domain information model are based on the classical logical operations – intersection, union, and supplement, but these operations are not so trivial. Because of complexity of the structure of the spaces these operations have at least two principally different realizations based on codes of the information spaces' elements and on contents of those elements.

The ArM32 information operations can be grouped into four sets corresponding to the main information structures: elements, spaces, aggregates, and indexes. Information operations are context depended and need special realizations for concrete purposes. Such well-known operations are for instance transferring from one structure to another, information search, sorting, making reports, etc.

At the end, there exist several operations, which serve information exchange between ArM32 archives (files) such as copying and moving spaces from one to another archive.

15.4 Discussion

We need to discuss shortly the main concept we use – the information space. Its main structure is an ordered set of numbered information elements. These elements may be information spaces or terminal elements. Of course, the hierarchical structures are well known. The new aspect of this model is the possibility to connect elements from different spaces and levels of the hierarchy using poly-indexation and in this way to create very large and complex networks with a co-ordinate hierarchical basis.

The variety of interconnections is the characteristic, which permits us to call the ordered set of numbered information elements "Information Space". In the information space, different information structures may exist at the same time in the same set of elements. In addition, the creation and destruction of the link's structures do not change the basic set of elements. The elements and spaces always exist but, in any cases, they may be "empty". At the end, the possibility to use coordinates is good for well-structured models where it is possible to replace search with addressing.

Summarizing, the advantages of the numbered information spaces, are:

- the possibility to build growing space hierarchies of information elements;
- the great power for building interconnections between information elements stored in the information base;
- the practically unlimited number of dimensions (this is the main advantage of the numbered information spaces for well-structured tasks where it is possible "to address, not to search");
- the possibility to create effective and useful tools, in particular for association rule mining.

Below, we show the advantages of using such memory structuring in the field of association rule mining in GMES.

15.5 Data mining analysis environment "PaGaNe"

The authors of this chapter form an international joint research group that work out a design of a data mining analysis environment called "PaGaNe". It contains variety of data mining algorithms, such as association rule miners, class association rule (CAR) algorithms, etc. [Mitov et al, 2009a/b].

The main specificity of PaGaNe is using of the advantages of multi-dimensional numbered information spaces [Markov, 2004], given by the access method ArM 32, such as:

- the possibility to build growing space hierarchies of information elements;
- the great power for building interconnections between information elements stored in the information base;
- the possibility to change searching with direct addressing in well-structured tasks.

The PaGaNe approach is a successor of the main ideas of GPN, such as hierarchical structuring of memory that allows reflecting the structure of composing instances and gender-species connections naturally, convenience for performing different operations of associative search. The recognition is based on reduced search in the multi-dimensional information space hierarchies.

An important idea of the approaches, used in PaGaNe, is replacing the symbol values of the objects' features with integer numbers of the elements of corresponding ordered sets. This way each instance or pattern can be represented by a vector of integer values, which may be used as co-ordinate address in corresponded multi-dimensional information space.

Here we will stop our attention on MPGN algorithm (abbreviation from "**M**ulti-layer **P**yramidal **G**rowing **N**etworks of information spaces"), which is kind of CAR algorithm that use advantages of numbered information spaces in order to overcome bottlenecks of exponential growth of combinations between patterns in the training stage, as well as quickly finding the potential answer in the recognition stage. The main goal is to extend the possibilities of network structures by using a special kind of multi-layer memory structures called "pyramids", which permits defining and realizing of new opportunities.

15.6 CAR algorithm MPGN

15.6.1 Coding convention

Usually in classification tasks rectangular data sets are used, every one of which is a set of instances $\mathbf{R} = \{R^i, i \in 1,...,r\}$. Each instance represent a set of attribute-value pairs $R = \{C = c, A_1 = a_1, ..., A_n = a_n\}$. Because in the rectangular data sets the positions of class and attributes are fixed, the instances are written as vectors, which contains only values of attributes: $R = (c, a_1, ..., a_n)$.

Every instance has the same quantity of attributes, but some of the values may be omitted. First attribute is the class attribute denoted c; other attributes are input attributes, denoted a_k.

Attribute positions of a given instance, which can take arbitrary values from the attribute domain, are denoted as "-".

Thus each instance (record) is presented as: $R = (c, a_1, ..., a_n)$; where n is the number of attributes (feature space dimension), $c \in \mathrm{N}$; $a_k \in \mathrm{N}$ or $a_k =$"–", $k \in [1,...,n]$.

More precisely, the class values and attribute values receive values, which are natural numbers from 1 to some maximal value (specific for each attribute), i.e. $c \in [1,...,M_c]$, $a_k \in [1,...,M_{a_k}]$.

Pattern is denoted P and has similar structure as instances. A pattern is a subset of an instance.

$$\left.\begin{array}{l} P = (c, a_1, ..., a_n) \\ R = (c, b_1, ..., b_n) \\ a_i = b_i \text{ or } a_i = "-" \end{array}\right\} P \subseteq R .$$

For example $P = (2,3,2,-,2) \subseteq R^1 = (2,3,2,1,2)$.

Every instance is a pattern but not every pattern is an instance, because the attributes with arbitrary values in the pattern may be more than in the instance.

In $P = (c, a_1, ..., a_n)$ usually (c) is called head of a pattern and $(a_1, ..., a_n)$ is called its body.

The cardinality of one pattern is defined as number of "non-arbitrary" attribute values:

$|P| = \text{number of } a_k \neq "-"$; $|P| \leq n$.

For the set of patterns $\mathbf{P} = \{P^i, i \in 1,...,m\}$ we can define maximal cardinality as maximum of cardinalities of patterns in the set.

$$MaxCard(\mathbf{P}) = \max_{i \in 1,...,m} |P^i| .$$

The intersection between P^i and P^j is the result of matching of these patterns.

$$P^i \cap P^j = (c^l, a_1^l, ..., a_n^l) : \; c^l = \begin{cases} c^i : c^i = c^j \\ "-" : c^i \neq c^j \end{cases} \text{ and } a_k^l = \begin{cases} a_k^i : a_k^i = a_k^j \\ "-" : a_k^i \neq a_k^j \end{cases} .$$

If $|P^i \cap P^j| > 0$ and $c^i = c^j$, then $P^i \cap P^j$ is a pattern, called generalized pattern of P^i and P^j.

For example, the generalized pattern of $R^1 = (2,3,2,1,2)$ and $R^3 = (2,3,2,2,2)$ is $P = (2,3,2,-,2)$.

The contradiction between two patterns means that they have equal attributes (equal bodies) but belong to different classes (different heads).

P^i contradicts to P^j if $P^i = (c^i, a_1, ..., a_n)$ and $P^j = (c^j, a_1, ..., a_n)$, but $c^i \neq c^j$.

The query instance Q (or only query) is similar to pattern, but the class value is unknown. It is denoted $Q = (?, b_1, ..., b_n)$.

The intersection percentage between pattern P and query Q is calculated as:

$$IntersectionPercentage(P,Q) = 100 * \frac{|P \cap Q|}{|P|} .$$

The intersection percentage is 100% in the case when P became a subset of pattern Q', which has the head of the pattern P and the body of the query Q.

$$\left.\begin{array}{l} P = (c, a_1, ..., a_n) \\ Q = (?, b_1, ..., b_n) \end{array}\right\} \text{ if } P \subseteq Q' = (c, b_1, ..., b_n) \text{ then } IntersectionPercentage(P,Q) = 100\% .$$

The support of a pattern P in a data set $\mathbf{R} = \{R^i, i \in 1,...,r\}$ is the number of instances for which P became their subset.

$Supp(P, \mathbf{R}) = \text{number of } R^i : P \subseteq R^i,\ R^i \in \mathbf{R}, i \in 1,...,r$.

15.6.2 Training process

The training process in MPGN consists of:

- preprocessing phase;
- generalization phase;
- pruning phase.

✓ Preprocessing phase

MPGN deals with instances and patterns separately for each class. This allows MPGN algorithm to be realized for using on parallel computers.

The preprocessing phase is aimed to convert the learning set in a standard form for further steps. It consists of:

- discretization of numerical attributes [Mitov et al, 2009b];
- numbering the values of attributes.

After discretization and the juxtaposing positive integers to nominal values, the instances are converted to numerical vectors.

✓ Generalization phase

The process of generalization is a chain of creating the patterns of upper layer as intersection between patterns from lower layer until new patterns are generated. For each class, the process starts from the layer 1 that contains the instances of training set. Patterns, generated as intersections between instances of the training set are stored in layer 2. Layer N is formed by patterns generated as intersections between patterns of the layer N-1. This process continues until no intersections are possible.

During generalization, for every class a separate pyramidal network structure is built. The process of generalization creates "vertical" interconnections between patterns from neighborhood layers. These interconnections for every pattern are represented by a set of "predecessors" and a set of "successors".

The predecessors' set of a concrete pattern contains all patterns from lower layer, which were participated in the process of its generalization. This means that if different intersections generate one and the same pattern than all patterns, which participate in the intersection are united as predecessors of resulting pattern.

The predecessors sets for instances of layer one are empty.

The successors' set of a concrete pattern contains the patterns from upper layer, which are created from it.

The successors' sets of patterns on the top of the pyramid are empty. These patterns are called "vertexes" of the corresponded pyramids.

One pattern may be included in more than one pyramid, but the vertex pattern belongs only to one pyramid.

It is possible any pyramid to contain only one instance.

✓ **Pruning phase**

Pruning phase is the process of iterative analysis of vertex patterns of all pyramids from different classes and removing all contradictory vertex patterns. The pruning algorithm is displayed on diagram, showed in Figure 177.

As a result, some of the most general patterns are deleted, because the vertexes with the same bodies were available in other classes (and they are deleted). The primary pyramids are decomposed to several sub-pyramids with lower number of layers.

Finally, the received vertexes of the pyramids not contradict to vertexes of pyramids of other classes.

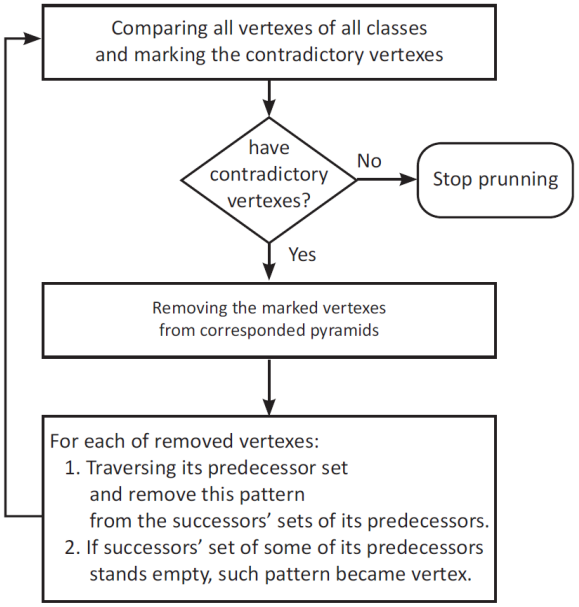

Figure 177. The MPGN pruning algorithm

15.6.3 Recognition process

The instance to be recognized is given by the values of its attributes $Q = (?, b_1, ..., b_n)$. Some of the values may be omitted. If some attributes are numerical, the values of these attributes are replaced with the number of corresponded discretized interval, where the value belongs. The categorical attributes also is recoded with the corresponded number values.

The recognition process consists of two main stages:

- creating recognition set for every class separately;
- analyzing resulting recognition sets from all classes and making decision which class to be given as answer.

✓ **Creating recognition set for every class**

In this stage each class is processed separately.

The goal is for each class to create the recognition set, which contains all patterns with maximal cardinality that have 100% intersection percentage with the query.

The process starts from the vertexes of all pyramids that belong to examined class. Using the predecessor sets of the patterns in the recognition set each pattern is replaced with the set of their predecessor that have 100% intersection percentage with the query, if such set is not empty. After lighting the recognition set keeping only patterns with maximal coverage, the process is iteratively repeated down to the layers until no new patterns became in the recognition set.

Figure 178 shows the block-scheme of this process.

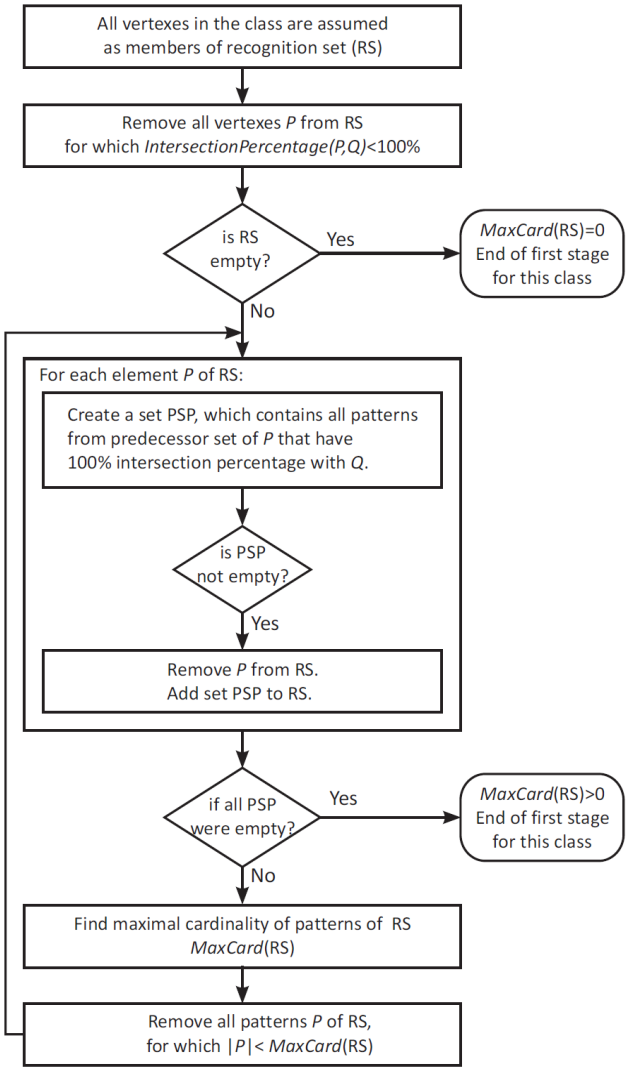

Figure 178. MPGN - Creating recognition set for one class

The process of creation of recognition set for each class also can be realized for using on parallel computers.

- ✓ **Analyzing results and make final decision**

The result of the first stage of the processing are the recognition sets for all classes $\mathrm{RS}c$, $c \in [1,...,M_c]$, which contain the patterns with maximal cardinality for this class $MaxCard(\mathrm{RS}c)$, $c \in [1,...,M_c]$ that have 100% intersection percentage with Q.

The goal is to find the class, which contains the patterns in its recognition set with highest cardinality.

For this purposes, first the maximum of all maximal cardinalities of the recognition sets of all classes is discovered.

$$MaxC = \max_{c \in [1,...,M_c]} MaxCard(\mathrm{RS}c)$$

The best case is when only one class has such maximal cardinality $MaxC$. Then, this class is given as answer.

The worst case is when $MaxC = 0$, which means that no recognition sets exists. In this case, if there is no instance, which body is equal to the query, the answer is that the request is not recognized and additional processing needs to be provided on the base of some other algorithms.

In the case when several classes have maximal cardinality $MaxC$ the number of instances with 100% intersection percentage with the query is find and the ratio between this number and all instances in the class is calculated. The class with maximal ratio is given as answer. The reason is that bigger ratio is received because the rule is more inherent to this class.

The full description of algorithm is shown in Figure 179.

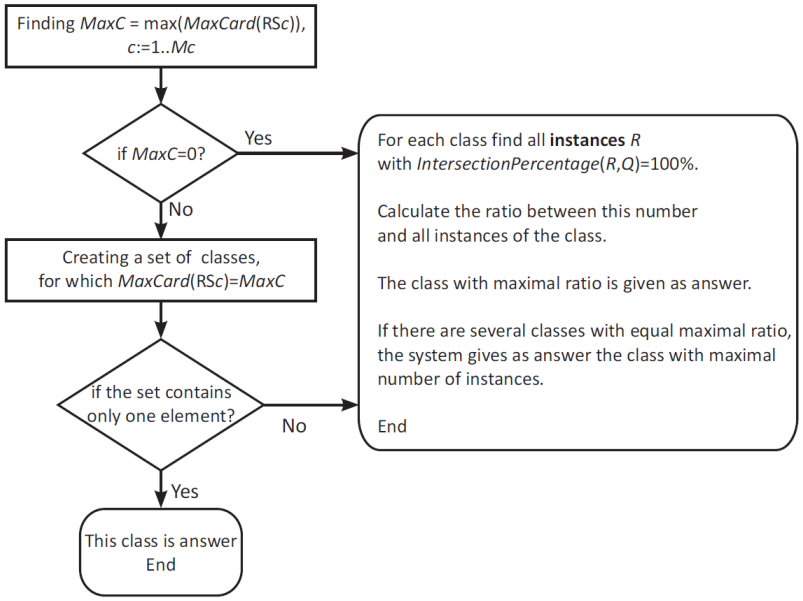

Figure 179. MPGN - Analyzing resulting recognition sets and making decision which class to be given as answer

This stage makes analysis of the results from previous step of all classes simultaneously. The parallelization of this process is not explicitly shown (it is not inherent to algorithm). The parallelization can be made in software realization during the process of comparing the results from classes between each other.

15.6.4 Experiments with Forestfires data set

Forest fires are major environmental issue, creating economical and ecological damage while endangering human lives. Fast detection is a key element for controlling such phenomenon. To achieve this, one alternative is to use automatic tools based on local sensors, such as provided by meteorological stations. In effect, meteorological conditions (e.g. temperature, wind) are known to influence forest fires and several fire indexes, such as the forest Fire Weather Index (FWI), use such data. In [Cortez and Morais, 2007] authors explore a data mining approach to predict the burned area of forest fires. Five different data mining techniques, e.g. Support Vector Machines (SVM) and Random Forests, and four distinct feature selection setups (using spatial, temporal, FWI components and weather attributes), were tested on real-world data collected from the northeast region of Portugal. The best configuration uses a SVM and four meteorological inputs (i.e. temperature, relative humidity, rain and wind) and it is capable of predicting the burned area of small fires, which

are more frequent. Such knowledge is particularly useful for improving firefighting resource management (e.g. prioritizing targets for air tankers and ground crews).

The *Forestfires data set* is available in the UC Irvine Machine Learning Repository [UCI MLR, 2011]: http://archive.ics.uci.edu/ml/datasets/Forest+Fires. The data can be used to test regression, feature selection or outlier detection methods.

The goal of the task is to predict the burned area of forest fires, in the northeast region of Portugal, by using meteorological and other data.

The dataset contains 517 instances. There are used 12 real attributes without missing values. Class values are two types – "no" (48% of instances) or "fire" (52% of instances).

Here is given explanation of each attribute [Cortez and Morais, 2007]:

1. X: x-axis spatial coordinate within the Montesinho park map: 1 to 9.
2. Y: y-axis spatial co-ordinate within the Montesinho park map: 2 to 9.
3. Month: month of the year: 'jan' to 'dec'.
4. Day: day of the week: 'mon' to 'sun'.
5. FFMC: FFMC index from the FWI system: 18.7 to 96.20.
6. DMC: DMC index from the FWI system: 1.1 to 291.3.
7. DC: DC index from the FWI system: 7.9 to 860.6.
8. ISI: ISI index from the FWI system: 0.0 to 56.10.
9. Temp: temperature in Celsius degrees: 2.2 to 33.30.
10. RH: relative humidity in %: 15.0 to 100.
11. Wind: wind speed in km/h: 0.40 to 9.40.
12. Rain: outside rain in mm/m2 : 0.0 to 6.4.

For the experiments, we randomize Forestfires data set and split in ratio 3:1. Thus, the learning set contains 388 instances and examining set contains the rest 129 instances. We chose to use experiments with supplied examining set in order to ensure comparableness of the results between classifiers in different environments (PaGaNe and Weka).

✓ Results of training process of MPGN with Forestfires

The result from training process of MPGN for chosen learning data set is given in Table 11.

Table 11. Results from training process of MPGN for Forestfires data set

	Class "No"	Class "Fire"
Number of layers for class	5	5
Number of vertexes for class	1	1
Number of non-contradictory vertexes after pruning	8	515

Graphical representations of the pyramids for each class (before pruning) are given in Figure 180 (class="No") and Figure 181 (class="Fire"). It is clearly seen that one pyramid for each class is formed.

After pruning (the process of clearing the contradictory vertexes), the pyramid of class "No" is decomposed to 8 pyramids, and respectively the pyramid of class "Fire" is divided to 515 pyramids.

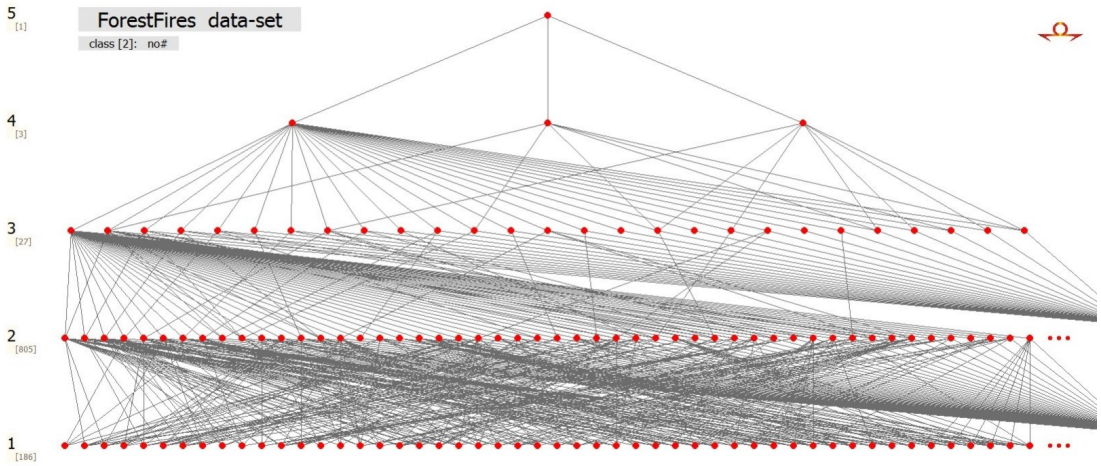

Figure 180. Pyramid for Forestfires data set, class="No"

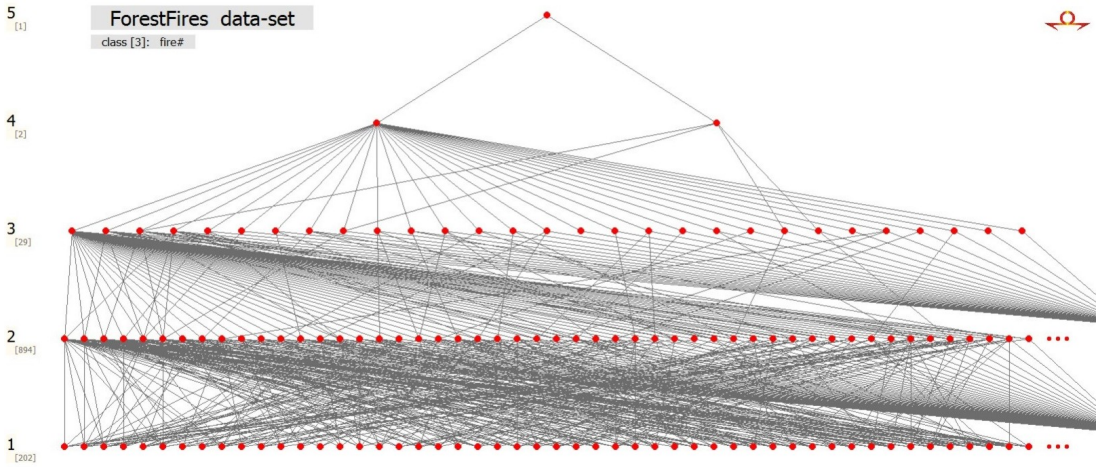

Figure 181. Pyramid for Forestfires data set, class="Fire"

✓ Results of recognition and comparison with other classifiers

We have made the comparison between classification accuracy of MPGN and other classifiers, realized in WEKA [Witten and Frank, 2005] for chosen data set.

For this comparison we have use best representatives from different groups of classifiers in WEKA (the classifiers that have showed best results within the group).

The chosen classifiers from groups, sorted in decreasing order of accuracy in the group, are as follows:

- Bayes – AODE (Aggregating One-Dependence Estimators), HNB (Hidden Naive Bayes), NaiveBayes;
- Functions – SMO (Sequential Minimal Optimization), SimpleLogistic, MultilayerPerceptron;
- Lazy – IB1, IBk, KStar;
- Meta – AdaBoostM1, LogitBoost, MultiScheme;
- Rules – JRip, ConjunctiveRule, OneR;
- Trees – ADTree, LADTree, J48.

The total number of the instances in the examining set was 129 instances.

In Table 12 the results of classification accuracy of MPGN and chosen representatives of WEKA classifiers for Forestfires data set are given.

Here is shown not only the global accuracy, received by classifiers, but also how they recognize corresponded classes, which is also important. For instance, ConjunctiveRule and MultiScheme have overall classification accuracy 52.71%, but this is as result that these classifiers give as answer only class="Yes". Other classifiers as classifiers in the Bayes group give not good accuracy (they are at the end of the ranking), but recognize relatively uniformly the instances of two classes.

Table 12. Results of recognition for Forestfires data set with MPGN and WEKA classifiers

Classifiers classifier model (supplied test set)	***Correctly classified instances***	**Accuracy (%)**	**Correctly classified "class=No" (%)**	**Correctly classified "class=Fire" (%)**
MPGN				
MPGN	*74*	57.36	57.38	57.35
Bayes				
AODE	*66*	51.16	44.26	57.35
HNB	*66*	51.16	52.46	50.00
NaiveBayes	*60*	46.51	44.26	48.53
Functions				
SMO	*71*	55.04	49.18	60.29
SimpleLogistic	*70*	54.26	65.57	44.12
MultilayerPerceptron	*67*	51.94	19.67	80.88
Lazy				
IB1	*69*	53.49	67.21	41.18
IBk	*68*	52.71	62.30	44.12
KStar	*66*	51.16	50.82	51.47
Meta				
AdaBoostM1	*72*	55.81	14.75	92.65
LogitBoost	*70*	54.26	65.57	44.12
MultiScheme	*68*	52.71	0.00	100.00
Rules				
JRip	*77*	59.69	26.23	89.71
ConjunctiveRule	*68*	52.71	0.00	100.00
OneR	*65*	50.38	47.54	52.94
Trees				
ADTree	*83*	64.34	36.07	89.71
LADTree	*78*	60.47	24.59	92.65
J48	*67*	51.94	19.67	80.88

Figure 182 shows overall classification accuracy of Forestfires data set for examined classifiers. Different colors of bars correspond to the groups, in which classifiers belongs. As we can see best results are given from ADTree and LADTree. In contrary, classifiers from Bayes group have not so good overall accuracy. MPGN showed relatively good results and it is on the fourth position.

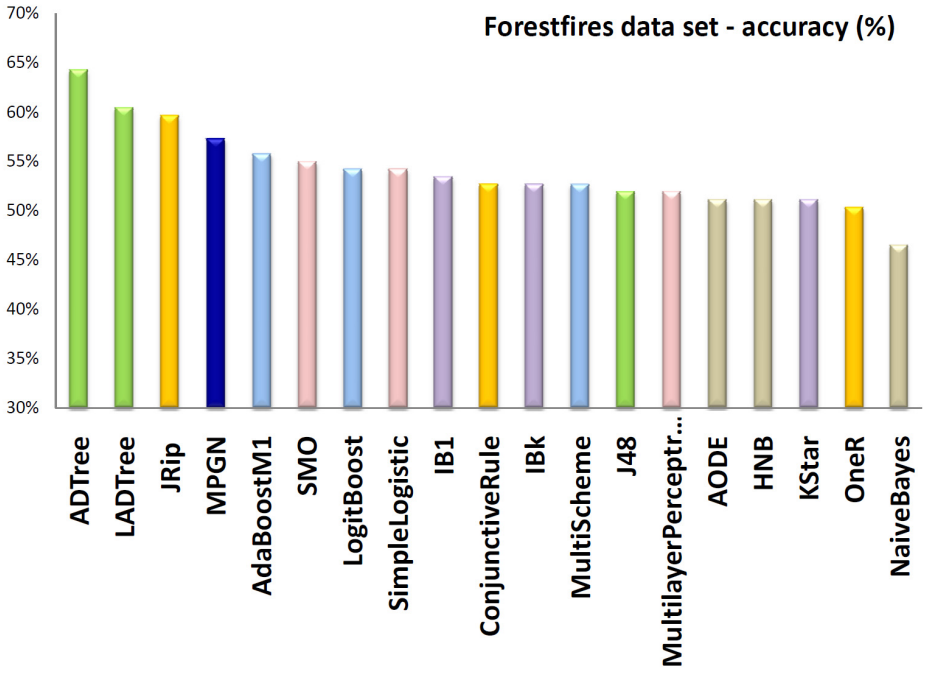

Figure 182. Classification accuracy of Forestfires data set given by MPGN and selected Weka classifiers

As we can see on Figure 183 it is not enough one classifier to have good overall accuracy. Keeping awareness commands in permanent readiness is also not so good situation. From first five classifiers only MPGN gives relatively good ratio on recognition two classes. In the other four classifiers prevailing answers are class="Fire".

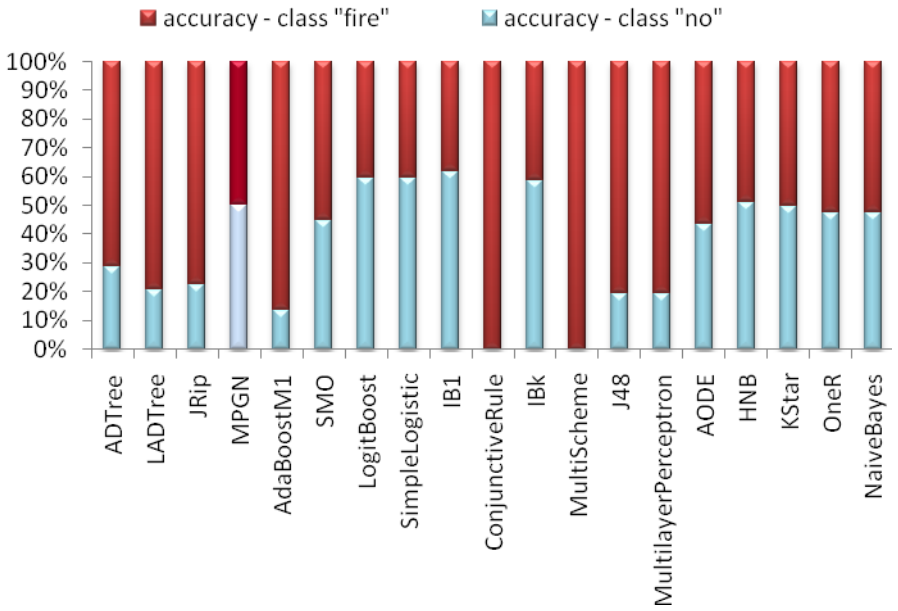

Figure 183. The ratio between recognized instances from class "No" versus class "Fire"

15.7 Discussion

The Memory Data Structures (MDS) and corresponded Access Methods (AM) had been available from the beginning of the developing the computer devices. As many devices there exists so many possibilities for developing different MDS and AM we have.

The main goal of creation of data mining environment PaGaNe was to show the advantages of using the multi-dimensional numbered information spaces as a model for memory structuring in the intelligent systems.

The main idea consists in replacing the (symbol or real; point or interval) values of the objects' attributes with integer numbers of the elements of corresponding ordered sets. This way each object is described by a vector of integer values, which is used as co-ordinate address in the multi-dimensional information space.

In other words, replacing the names by numbers permits the using of mathematical functions and address vectors for accessing the information.

ArM32 engine supports multithreaded parallel access to the information base in real time. Very important characteristic of ArM32 is possibility not to occupy space for empty structures (elements or spaces).

The Data Mining Environment PaGaNe is the first realization based on the Multi-dimensional numbered information spaces. The good results received by its systems show a new way for building intelligent systems.

The "class association rules" (CAR) algorithms have their important place in the family of classification algorithms. The advantages of associative classifiers can be highlighted in several very important directions, such as: very efficient training; possibility to deal with high dimensionality; no assumptions for the independence of attributes; very fast classification and the result is easily understandable by humans. The latter two advantages make CAR algorithms an irreplaceable assistant in the processes of disaster risk management, where fast reaction and reliability of the systems are crucial. The realization of discussed MPGN classifier, based on such structures, and conducted experiments with it show promising results in the area of using such paradigm in the field of disaster risk management.