Yugoslav Journal of Operations Research
9 (1999), Number 1, 75-95

CLASSIFICATION OF OBJECTS BASED ON CASE-BASED REASONING

Sanja PETROVIĆ
University of Nottingham
Computer Sciences Department
University Park
NG7 2RD United Kingdom

Abstract: This paper presents a new approach to the classification of objects which are described by many features to one of the categories in a predetermined set of categories. The classification is based on case-based reasoning. Rather than classifying new objects using necessary and sufficient conditions, defined by a set of rules, the knowledge and experience memorised during past classifications are used. Each case comprises the description of the object and the category to which it is assigned. The memorised cases are organised in a case base. In the process of classifying a new object, a case which is the most similar to the new object is retrieved from the case base. The new object is assigned to the same category as the retrieved case. The described approach to classification was a basis for the design of an expert system which helps the decision maker classify a given multicriteria problem to an adequate method for its solving.

Keywords: Classification of objects, case-based reasoning, multicriteria analysis, selection of a multicriteria method.

1. INTRODUCTION

Various problems of operational research can be considered as classification tasks, where objects, described by many features, have to be assigned to one category from a predetermined set of categories. The need for classification of objects arises in a wide variety of domains, for example in machine repair diagnosis, medical diagnostics, loan applicant classification, and many other.

We focus our research on the classification of objects based on case-based reasoning (CBR). CBR is a new type of reasoning based on the premise that human reasoning processes are founded on specific experience rather than a set of general guidelines or first principles [8]. In order to utilise past experience, the ways the experts solve actual problems from a domain are memorised as cases. CBR is preferred for solving complex and ill-structured problems [3]. Many classification tasks are characterised by too much data available or by a lack of data with a large amount of complex or unknown interrelations among variables. Some data is hard to interpret or quantify. Therefore, many classification tasks belong to the class of complex and ill-structured problems [1]. Usually, there are no algorithmic procedures, repetitive and routine, which can produce one strict solution for such problems. In the case-based classification, instead of classifying new objects using necessary and sufficient conditions, which are not always possible to define, the knowledge and experience memorised during past classifications are applied. One case comprises the description of the object and the category to which it is assigned. The memorised cases are organised in a case base. In the process of classifying a new object, a case which is the most similar to the new object is retrieved from the case base. The new object is assigned to the same category as the case retrieved. The new object is integrated in the case-base. Therefore, the CBR classification system learns incrementally and improves its performance.

Some interesting approaches to the classification of objects in different domains based on CBR are described in the literature. The CBR system PROTOS performs classification in the domain of audiological disorders [14]. Given a description of the symptoms and test results of some patient, PROTOS determines which hearing disorder that patient has. PROTOS learns domain-specific knowledge under the guidance of an expert who provides the additional information needed to rectify problem solving failures.

The retrieval of relevant previous cases is crucial to the success of CBR system. The usefulness of previous cases is determined by assessing the similarity of a new object with the previous cases. Different measures of similarities between cases from the case-base and a new object are considered and tested using the diagnosis and repair tasks in an electromechanical domain [6].

A novel architecture for combining rule-based and case-based reasoning is presented in [5]. Rules present broad trends in the domain, while cases are good to capture the exceptions which violate the rules. The principal idea is to apply the rules to a target problem to get a first approximation of the answer. If the problem is judged to be compellingly similar to a known exception to the rules in any aspect of its behavior, then that aspect is modeled after the exception rather than the rules. The proposed approach is illustrated in the domain of auto insurance where the task was to assess the risk of insuring a new client.

Knowledge acquisition problems, systematic case-based construction and verification of expert knowledge are treated in [9] and tested in medical diagnosis problems.

In this paper a new approach to case-based classification is described. The proposed approach is applied in the domain of multicriteria analysis (MCA). The problem is to select an MCA method adequate for solving a given multicriteria problem. The problem is treated as a classification task, where the description of each multicriteria problem constitutes an object which has to be assigned to one category, i.e. to one among a number of multicriteria methods adequate for its solving. Section 2 presents the architecture and the basic elements of the developed CBR classification system. Section 3 describes the developed CBR system MAGIC (acronym for **M**ulticriteria **A**nalysis **G**uidance for Method Selection **I**nferred from **C**ases) which assists the decision maker (DM) in selecting an adequate MCA method. An illustrative example is given with the discussion about the validation of the system. The usefulness of the application of the CBR in classification problems is pointed out in the conclusion.

2. CBR SYSTEM FOR THE CLASSIFICATION OF OBJECTS

Classification problems which are considered in this research can be presented in the following formal way.

Given: Object C is represented as a conjunction of ordered pairs $(index, value)$, where indices are features of objects relevant for the selection of a category:

$$C = ((index, value)_i \mid i \in I_C) .$$

Task: Assign a given object C to one category from the set $\mathcal{K} = \{K_1, \ldots, K_M\}$.

An index can be single–valued or multivalued depending on the number of values that can be assigned to it.

2.1. System architecture

Developing a CBR system which should assist in assigning a category to a particular object starts with the identification of cases that can provide the basis for the classification of the new object. For each category an expert provides à certain number of typical objects – prototypes which belong to the category and objects which are exceptions to the category. They are treated as training cases. Given a new object to be classified, the system proposes a category, matching the new object against the cases in the case base. The object which is considered to be the most similar to the new object is retrieved. The category of the retrieved object is proposed for the new object. The new object with the proposed category is integrated in the case base if it contains knowledge and experience not present in the case base. In order to learn from past experience, the CBR system has to collect feedback from the real world and to the notice consequences of its reasoning. The category proposed may be evaluated as inadequate and then the CBR system continues searching the case base until some other category is proposed. In this way, facing similar problems, the system can avoid the same mistake in the future and can propose the adequate category immediately. Learning from failure, which is realised in the developed CBR classification system, is an important characteristic of CBR. Three original algorithms are developed: (1) an algorithm for system training which integrates training cases in the case base and forms the initial case base, (2) an algorithm for the classification of a new object and (3) an algorithm for learning from failure which continues the classification process after the failure of the proposed category is registered.

The CBR system developed consists of five main modules depicted in Fig. 1: the *input module* which accepts training cases and new objects, the *training module* and the *object classification module* which are used for system training and classification of a new object, the *category module* which contains a number of categories and is used to evaluate the proposed solution, and the *output module* which presents the classification results.

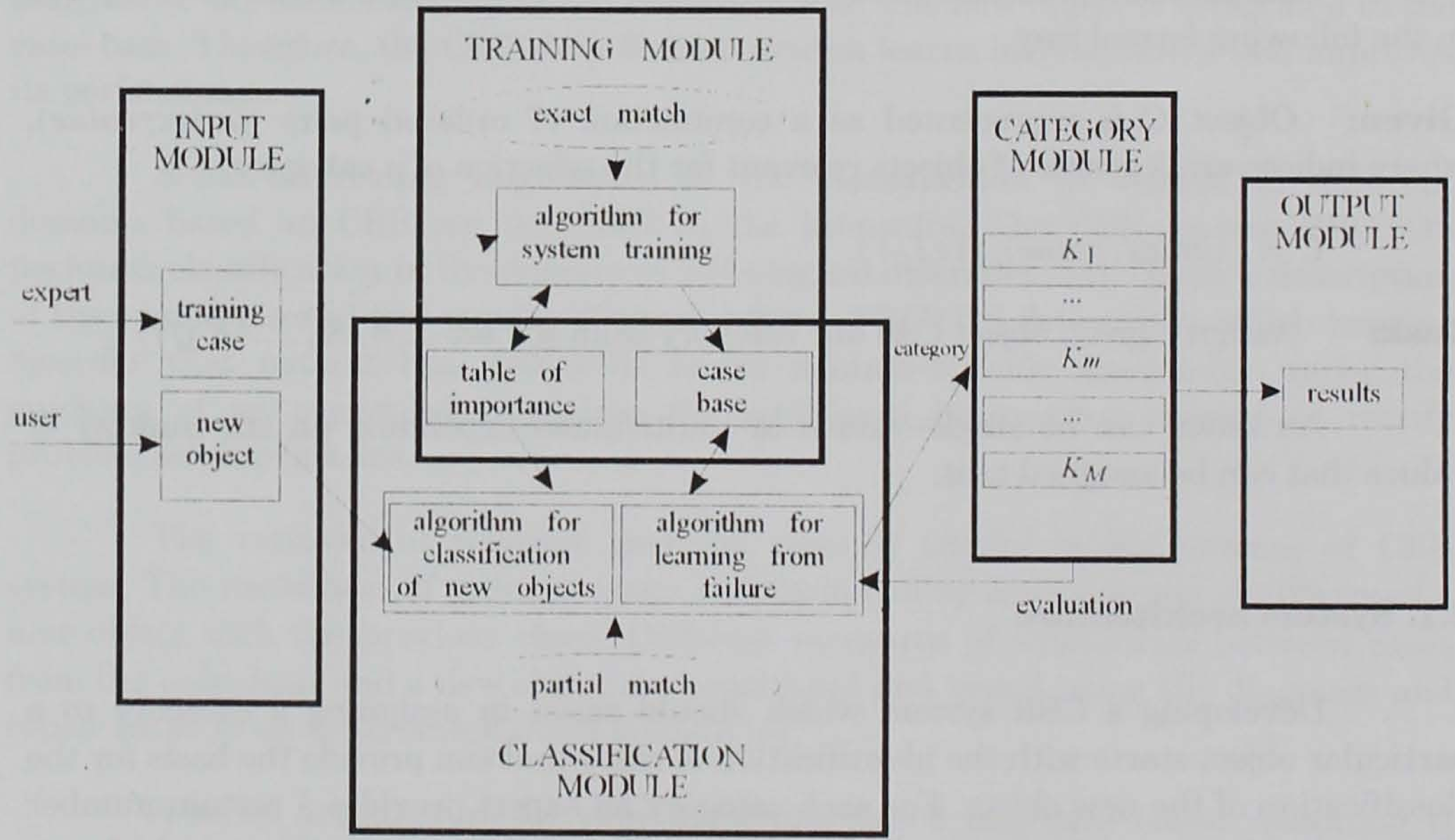

Figure 1: The architecture of the CBR classification system

The remainder of this section gives a more detailed look at the CBR system developed.

2.2. Organisation of the case–base

The number of cases in the case–base can become very large. In order to enable efficient search of the case–base the cases are organised hierarchically. During the classification of a new object only a small relevant subset of the cases, considered to be the most similar to the new object, is retrieved. The subset of cases obtained may then be compared to the new problem.

Cases are hierarchically organised in a shared–feature network, a structure which is known in the CBR literature [8]. The shared–feature network provides a means of clustering cases so that the cases which have some common index values are grouped together. We used the general induction principles offered by the UNIMEM method for incremental inductive learning [4] to construct the network and enriched them with some additional characteristics necessary for CBR.

Each node in the network is created by a generalisation process performed applying two cases. The generalisation of the index values depends on the index types. The generalisation extracts the values of single–valued indices which are equal in both cases, and produces the union of the values of multivalued indices. The created node stores the cases and has indices which are produced as the result of the generalisation process.

In the proposed CBR algorithms, each node is represented as a triple $N = \{\mathcal{P}, \mathcal{D}, \mathcal{C}\}$ where $\mathcal{P}$ is a subset of ordered pairs (*index*, *value*) which describe node N, $\mathcal{P} = \{ (index, value)_i \mid i \in I_N \}$, $\mathcal{D}$ is a set of node descendants, and $\mathcal{C}$ is a set of cases stored with the node. Descendants of a node, i.e., nodes on a lower hierarchical level, inherit all the features of the predecessor. Nodes on the higher levels of hierarchy represent generalised concepts, while descending down the hierarchy more specific concepts are reached.

The goal of the construction of the network is to infer the description of nodes from the description of a given set of cases. Cases are integrated in the present network, one at a time, without reprocessing previously encountered cases.

2.3. System training

An algorithm developed for the CBR system training creates an initial shared–feature network based on input training cases. Each training case is classified into a suitable category.

Let us define the concepts which form the basis for the CBR system training.

(a) *Importance of an ordered pair (index, value)*

One of the crucial questions for the classification of a new object is how much the value of an index influences the selection of the category. It is assumed that the set of ordered pairs (*index*, *value*) which characterises objects is finite. To each ordered pair (*index*, *value*) is associated to a coefficient which reflects the relative importance of the index value in the selection of the category. However, the nearest neighbour method, often used in many CBR systems [17] where a fixed weight is associated to each index and used in the similarity measure between a new object and cases in the case base is not available. It can be noted that the frequencies of the appearances of the ordered pairs *(index, value)* in the cases are different. Calculation of the importance of an ordered pair *(index, value)* is based on the idea that if some index value occurs in the cases which are classified into a large number of categories, or even into all of them, then that ordered pair *(index, value)* is not useful in the selection of the category. On the other hand, if an ordered pair *(index, value)* appears only in the cases which are classified into a small number of categories, preferably one, then that index value characterises the category substantially and is very important for the category selection process.

In order to calculate the importance of the ordered pairs *(index, value)* a *table of importance* is defined. The rows of the table present ordered pairs $(index, value)_i$, $i = 1, \dots, I$, used in the object descriptions, while columns present categories, $K_1, \dots, K_m, \dots, K_M$. To each field (i, m) a number n_{im} is associated which presents the number of cases in the case base which have the ordered pair $(index, value)_i$ and are classified into category K_m. The importance of each ordered pair $(index, value)_i$ is calculated as the square root of the sum of the relative frequencies:

$$w_i = \sqrt{\sum_{m=1}^{M} \left(\frac{n_{im}}{\sum_{m=1}^{M} n_{im}} \right)^2}, \qquad i = 1, \dots, I \tag{1}$$

Number $n_{im} / \sum_{m=1}^{M} n_{im}$ presents the conditional probability of choosing category K_m, if the object description has the ordered pair $(index, value)_i$. Importance w_i takes values from the interval [0, 1], so that higher values correspond to more important ordered pairs $(index, value)_i$. After the insertion of each case C in the case base the weights w_i of ordered pairs $(index, value)_i$, $i \in I_C$, contained in the case description are modified.

(b) *Importance of a node*

To each node is assigned a real value from the interval [0, 1] which expresses its importance as a whole. The importance of node N is defined as:

$$W(N) = 1 - \prod_{i \in I_N} (1 - w_i) \tag{2}$$

The more important the ordered pairs $(index, value)_i$, $i \in I_N$, node N has, the closer the value $W(N)$ is to 1. If node N contains an ordered pair $(index, value)_i$ with importance 1, then $W(N)$ is equal to 1.

(c) *Match relation*

The *match* relation involves indices and it is defined in Table 1. The first argument presents a value of the index in the network, the second one a value in a new case (a training case, or a new object). If the index is single–valued, then the *match* holds true if the arguments are equal. If the index is multivalued, then the *match* holds true if the first argument is a superset of the second one.

Table 1: *Match* relation

Index	*match(valueN, valueC)*
Multivalued	iff $valueN \supseteq valueC$
single–valued	iff $valueN = valueC$

(d) *Degree of the exact match between a node and a case*

There is an *exact match* between node *N* and case *C* if for each ordered pair (*index, valueN*) contained in node *N* there exists a pair (*index, valueC*) in case *C*, and the relation *match(valueN, valueC)* holds true.

The *degree of the exact match* between node *N* and case *C* is defined as:

$$ExactMatchDeg(N,C) = \begin{cases} \sum_{i \in I_N} w_i, & \text{if there is an } \textit{exact match} \text{ between} \\ & \text{node } N \text{ and case } C \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

If the *index* is multivalued then the importance w_i of the ordered pair $(index, value)_i$ in the node is defined as the maximum of the importance of the *index* and all values in the *value*, separately.

(e) *Match between a case and a new object*

A case from the case base matches a new object if all of its index values are in relation *match* with the corresponding values of the new object.

An algorithm developed for the CBR system training incrementally updates the shared–feature network as input training cases are inserted in the system. Updating means finding the place in the network where the input training case fits. The algorithm developed is iterative by nature. Searching the network is based on the defined concepts and heuristics [12], [13]. The node with the largest degree of exact match with the input training case is selected. Then, either a new node is derived by generalisation of the input case and one case of the selected node chosen using the developed heuristics, or the input case is stored with the selected node if the node contains no case that *matches* the input case.

2.4. Classification of a new object

The goal of the classification of a new object is to find the node whose description is the most similar to the new object and to match the new object against the cases of the selected node. The category for the new object is proposed on the basis of the selected case.

The concepts used in the algorithm developed for the classification of a new object are introduced below.

(a) *Similar relation*

A *similar* relation is defined for each index using the knowledge and experience of an expert from the domain of the classification problem. The first argument concerns a value in the network, the second one a value in the new object. If the object with value *valueN* of a particular index is classified into one category and if there are reasons to classify into the same category the object with *valueC* of the same index, then *similar*(*valueN*, *valueC*) holds.

The *degree of the similarity* between two index values can take three values: *FULL_MATCH* if the two values are equal, *PARTIAL_MATCH* if the value in the network is similar to the new object value, but not equal, and *NO_MATCH* otherwise. Parameters *FULL_MATCH*, *PARTIAL_MATCH* and *NO_MATCH* are set at 1, 0.5, and 0, respectively.

(b) *Degree of the partial match between a node and a new object*

The proper definition of the similarity measure between a network node and a new object is of great value for good functioning of the CBR system. The similarity between the node and the new object is expressed as the degree of the partial match between them. The partial match enables a new object with no match in the case base to be solved. The *degree of the partial match between node N and new object C* takes into consideration the importance of the index values of a node which is similar to the values in the new object and punishes non-similarity. It is defined in the following way:

$$PartialMatchDegNC(N,C) = \Big(\sum_{i \in I_{NC}} s_i \cdot w_i - P \cdot \sum_{i \in I_{\overline{NC}}} w_i\Big) W(N) \qquad (4)$$

where:

defined concepts and heuristics [12], [13]. The node with the largest degree of exact match with the input training case is selected. Then, either a new node is derived by generalisation of the input case and one case of the selected node chosen using the developed heuristics, or the input case is stored with the selected node if the node contains no case that *matches* the input case.

2.4. Classification of a new object

The goal of the classification of a new object is to find the node whose description is the most similar to the new object and to match the new object against the cases of the selected node. The category for the new object is proposed on the basis of the selected case.

The concepts used in the algorithm developed for the classification of a new object are introduced below.

(a) *Similar relation*

A *similar* relation is defined for each index using the knowledge and experience of an expert from the domain of the classification problem. The first argument concerns a value in the network, the second one a value in the new object. If the object with value *valueN* of a particular index is classified into one category and if there are reasons to classify into the same category the object with *valueC* of the same index, then *similar*(*valueN*, *valueC*) holds.

The d*egree of the similarity* between two index values can take three values: *FULL_MATCH* if the two values are equal, *PARTIAL_MATCH* if the value in the network is similar to the new object value, but not equal, and *NO_MATCH* otherwise. Parameters *FULL_MATCH*, *PARTIAL_MATCH* and *NO_MATCH* are set at 1, 0.5, and 0, respectively.

(b) *Degree of the partial match between a node and a new object*

The proper definition of the similarity measure between a network node and a new object is of great value for good functioning of the CBR system. The similarity between the node and the new object is expressed as the degree of the partial match between them. The partial match enables a new object with no match in the case base to be solved. The *degree of the partial match between node N and new object C* takes into consideration the importance of the index values of a node which is similar to the values in the new object and punishes non-similarity. It is defined in the following way:

$$PartialMatchDegNC(N,C) = \Big(\sum_{i \in I_{NC}} s_i \cdot w_i - P \cdot \sum_{i \in I_{\overline{NC}}} w_i \Big) W(N) \qquad (4)$$

where:

I_{NC} is a subset of the set I_N such that for $i \in I_{NC}$, *value* in the ordered pair $(index, value)_i$ is in the *similar* relation with the corresponding value of the new object C

$I_{\overline{NC}}$ is a subset of the set I_N such that for $i \in I_{NC}$, *value* in the ordered pair $(index, value)_i$ is not in the *similar* relation with the corresponding value of the new object C,

s_i is the degree of the similarity between the value of the *index* in the network and in the new object,

P is a parameter expressing penalty when there is a difference between the index value in the network and in the new object (the parameter is set at 2 in the CBR system developed).

In order to direct the searching process to more specific nodes which contain more indices and to nodes which contain pairs *(index, value)* of higher importance, the obtained difference is multiplied by the importance of the node.

If the *index* is multivalued, then the importance w_i of the ordered pair $(index, value)_i$ in a network node is defined as the maximum of the importance of the *index* and all values contained in the intersection of the *value* and the new object value, separately.

(c) *Degree of the partial match between a case and a new object*

The *degree of the partial match between a case C1 and a new object C2* is defined in an analogous way:

$$PartialMatchDegCC(C1, C2) = \sum_{i \in I_{C1C2}} s_i \cdot w_i - P \cdot \sum_{i \in I_{\overline{C1C2}}} w_i \tag{5}$$

where:

I_{C1C2} is a subset of the set I_{C1} such that for $i \in I_{C1C2}$, *value* in the ordered pair $(index, value)_i$ is in the *similar* relation with the corresponding value of the new object $C2$,

$I_{\overline{C1C2}}$ is a subset of the set I_{C1} such that for $i \in I_{\overline{C1C2}}$, *value* in the ordered pair $(index, value)_i$ is not in the *similar* relation with the corresponding value of the new object $C2$.

The algorithm developed for the classification of a new object propagates the index values of the new object to an existing network using defined concepts and heuristics [12], [13]. The node with the largest degree of partial match with the new object is selected. The case of the selected node with the highest degree of partial match with the new object is retrieved. The category of the retrieved case is proposed for the new object.

In order to control the growth of the case base, rules are defined to determine whether the new object has to be memorised or not [12]. Generally, a new object is integrated into the network if it contains knowledge or experience not already present in the case base.

Training cases and the order of their insertion in the case base influence the network structure and the descriptions of the nodes. An induction process has to identify all the indices which should not be in the node because of their irrelevance for the categories under the node. In order to accomplish this, fine tuning of the node descriptions is performed during the system training and new object classification. A certainty factor is assigned to each index in the node. In the algorithm for the system training, the certainty factors of all indices in the selected nodes are increased by 1. After the completion of the training phase the certainty factor of the index in the node is equal to the number of the cases stored under that node in the hierarchy. The certainty factors associated to indices in the nodes selected during the search are updated accordingly. If the value of the node index is similar to the value in the new object, with the degree of similarity *FULL_MATCH*, the certainty factor of the index is increased by 1. No change is made if the value is similar with the degree *PARTIAL_MATCH*. If the value of the node index is not similar to the value in the new object, the certainty factor of the index is decreased by 1. When the certainty factor of the index drops below a specified threshold then this index is not relevant for the categories under the node hierarchy. The index is therefore removed from the node. Removal of the index can cause two nodes in the same hierarchical level to have the same description. Such two nodes are joined and a new node which contains the union of the descendants and the cases stored with the two nodes is created.

3. CBR SYSTEM FOR SELECTION OF AN MCA METHOD

In many real-world problems there is a need to analyse a finite set of alternatives described by many criteria. Criteria are usually expressed in different units of measures and different scales and they are totally or partially conflicting and incommensurable. The aim of MCA is to help the DM explore the multicriteria problem at hand, express his/her preference structure, and eventually lead to a preferred course of action.

Over the past two decades many MCA methods have been developed [16]. It is estimated that there are more than 50 distinct MCA methods [11]. A large number of MCA methods have arisen not just because of different schools of thought that have different ideas for solving multicriteria problems, but also from diverse views of DMs and various preference information they enter into analysis. Consequently, these methods are not easy to classify, evaluate and compare. But, despite the development of a large number of methods, no single one can be considered a superior method, applicable in all decision making situations. Many analysts are not able to clearly justify their choice of one MCA method rather than another. Very often the choice is motivated by familiarity with a specific method. This means that the decision making situation adapts to the MCA method and it should be the opposite – within the set of

MCA methods an analyst should select the MCA method adequate for a given situation. Therefore, there is increased interest in the formalisation of a process whose aim is to assist the DM in the selection of an appropriate MCA, and in implementation of an appropriate software package.

Expert system technology offers various techniques to simulate the behaviour of an expert in MCA. Rules are very often used in expert systems for knowledge representation. However, the large number of methods, the diversity of multicriteria problems and the large number of problems that are exceptions to a typical application of an MCA method, require a large number of rules which are cumbersome and tedious to define, modify and maintain.

Instead, the prototype system MAGIC based on the described case-based classification was developed. MAGIC was developed using the object-oriented programming language C++. MAGIC introduces six rather popular MCA methods: ELECTRE II [15], PROMETHEE II [2], the Conjunctive method, Maximin, TOPSIS, and SAW - The Simple Additive Weighting Method [7]. The modular structure of MAGIC enables relatively easy incorporation of new MCA methods.

The introduced MCA methods require the following preference information from the DM: *weights* which present the relative importance of criteria, *acceptable levels* which present cutoff values acceptable for each criterion, *indifference thresholds* with respect to a single criterion - using this parameter the DM expresses his/her indifference toward the difference between the criterion values of two alternatives, and *preference thresholds* with respect to a single criterion - using these parameters the DM expresses his/her attitude toward the difference between the criterion values of two alternatives to consider that one alternative is strictly preferred over the other. Indices used to describe the cases of multicriteria problems which can be solved by the introduced MCA methods are given in Table 2. Index *cri type* is multivalued, while all other indices are single-valued.

Table 2: Indices used to describe MCA cases and their domains

Index	***Description***	***Domain***
probl type	type of the *multicriteria* problem	{*rank, class, best, several*}
num of alter	linguistic term used to express the problem dimension taking into account number of alternatives	{*small, moderate, large*}
num of att	linguistic term used to express the problem dimension taking into account number of criteria	{*small, moderate, large*}
cri type	type of criteria values	{*card, ling, bin*}
weights	existence of criteria weights	{*yes, no*}
acc level	existence of criteria cutoff values	{*yes, no*}
indiff	existence of indifference thresholds	{ *yes, no* }
pref	existence of preference thresholds	{ *yes, no* }

Index values are specified in the problem description or calculated using the values of problem features not used as indices. For example, the value of the index *num of alter* is calculated using the value of the feature *alternatives* which shows the exact number of alternatives, in the following way:

IF value of the *alternatives* ≤ 6	THEN *num of alter* is set to *small*
IF value of the *alternatives* [7, 20]	THEN *num of alter* is set to *moderate*
IF value of the *alternatives* ≥ 21	THEN *num of alter* is set to *large*

Similarly, the value of the index *num of cri* is calculated using the value of the feature *criteria*:

IF value of the *criteria* ≤ 5	THEN *num of cri* is set to *small*
IF value of the *criteria* [6, 10]	THEN *num of cri* is set to *moderate*
IF value of the *criteria* ≥ 11	THEN *num of cri* is set to *large*

The values used in these rules are elicited through an interview with an expert in the MCA.

The definition of the *similar* relation which contains domain-specific knowledge is given in Table 3.

Table 3: *Similar* relation

Index	*similar(valueN, valueC)*	*degree of similarity*
Probl type	iff *valueN = rank, valueC = best* iff *valueN = rank, valueC = several* iff *valueN = valueC*	*PARTIAL_MATCH* *PARTIAL_MATCH* *FULL_MATCH*
num of alter and *num of att*	iff *valueN = small, valueC = moderate* iff *valueN = large, valueC = moderate* iff *valueN = moderate, valueC = small* iff *valueN = moderate, valueC = large* iff *valueN = valueC*	*PARTIAL_MATCH* *PARTIAL_MATCH* *PARTIAL_MATCH* *PARTIAL_MATCH* *FULL_MATCH*
att type	iff *valueN* $\supset$ *valueC* iff *valueN = valueC*	*PARTIAL_MATCH* *FULL_MATCH*
all other indices	iff *valueN = valueC*	*FULL_MATCH*

For each MCA method introduced in the prototype system, three training cases are identified. In total 18 training cases are integrated in the case base and are presented in Table 4.

Table 4: Training cases in MAGIC

Conj1	*Conj2*	*Conj3*
problem type several	*problem type class*	*problem type several*
num of objects large	*num of objects large*	*num of objects large*
num of attributes small	*num of criteria small*	*num of criteria small*
criteria type card	*criteria type card*	*criteria type ling*
weights no	*weights no*	*weights no*
acc level no	*acc level yes*	*acc level no*
indifference no	*indifference no*	*indifference no*
preference no	*preference no*	*preference no*
method Conjunctive	*method Conjunctive*	*method Conjunctive*
ELECTRE1	*ELECTRE2*	*ELECTRE3*
problem type class	*problem type best*	*problem type class*
num of objects small	*num of objects small*	*num of objects small*
num of criteria small	*num of criteria small*	*num of criteria large*
criteria type card	*criteria type card+ling+bin*	*criteria type ling*
weights yes	*weights yes*	*weights yes*
acc level no	*acc level no*	*acc level no*
indifference no	*indifference no*	*indifference no*
preference no	*preference no*	*preference no*
method ELECTRE	*method ELECTRE*	*method ELECTRE*
PROM1	*PROM2*	*PROM3*
problem type rank	*problem type several*	*problem type best*
num of objects small	*num of objects small*	*num of objects large*
num of criteria small	*num of criteria small*	*num of criteria small*
criteria type card	*criteria type card*	*criteria type card*
weights yes	*weights yes*	*weights yes*
acc level no	*acc level no*	*acc level no*
indifference no	*indifference yes*	*indifference yes*
preference yes	*preference no*	*preference yes*
method PROMETHEE	*method PROMETHEE*	*method PROMETHEE*
Max1	*Max2*	*Max3*
problem type best	*problem type best*	*problem type best*
num of objects large	*num of objects large*	*num of objects small*
num of criteria large	*num of criteria small*	*num of criteria small*
criteria type card	*criteria type card+ling*	*criteria type card+ling*
weights no	*weights no*	*weights no*
acc level no	*acc level no*	*acc level no*
indifference no	*indifference no*	*indifference no*
preference no	*preference no*	*preference no*
method Maximin	*method Maximin*	*method Maximin*
TOPSIS1	*TOPSIS2*	*TOPSIS3*
problem type best	*problem type rank*	*problem type rank*
num of objects small	*num of objects large*	*num of objects small*
num of criteria small	*num of criteria small*	*num of criteria large*
criteria type card	*criteria type card*	*criteria type card*
weights yes	*weights yes*	*weights yes*
acc level no	*acc level no*	*acc level no*
indifference no	*indifference no*	*indifference no*
preference no	*preference no*	*preference no*
method TOPSIS	*method TOPSIS*	*method TOPSIS*
SAW1	*SAW2*	*SAW3*
problem type rank	*problem type best*	*problem type best*
num of objects large	*num of objects large*	*num of objects large*
num of criteria large	*num of criteria large*	*num of criteria large*
criteria type card	*criteria type ling*	*criteria type card*
weights yes	*weights yes*	*weights yes*
acc level no	*acc level no*	*acc level no*
indifference no	*indifference no*	*indifference no*
preference no	*preference no*	*preference no*
method SAW	*method SAW*	*method SAW*

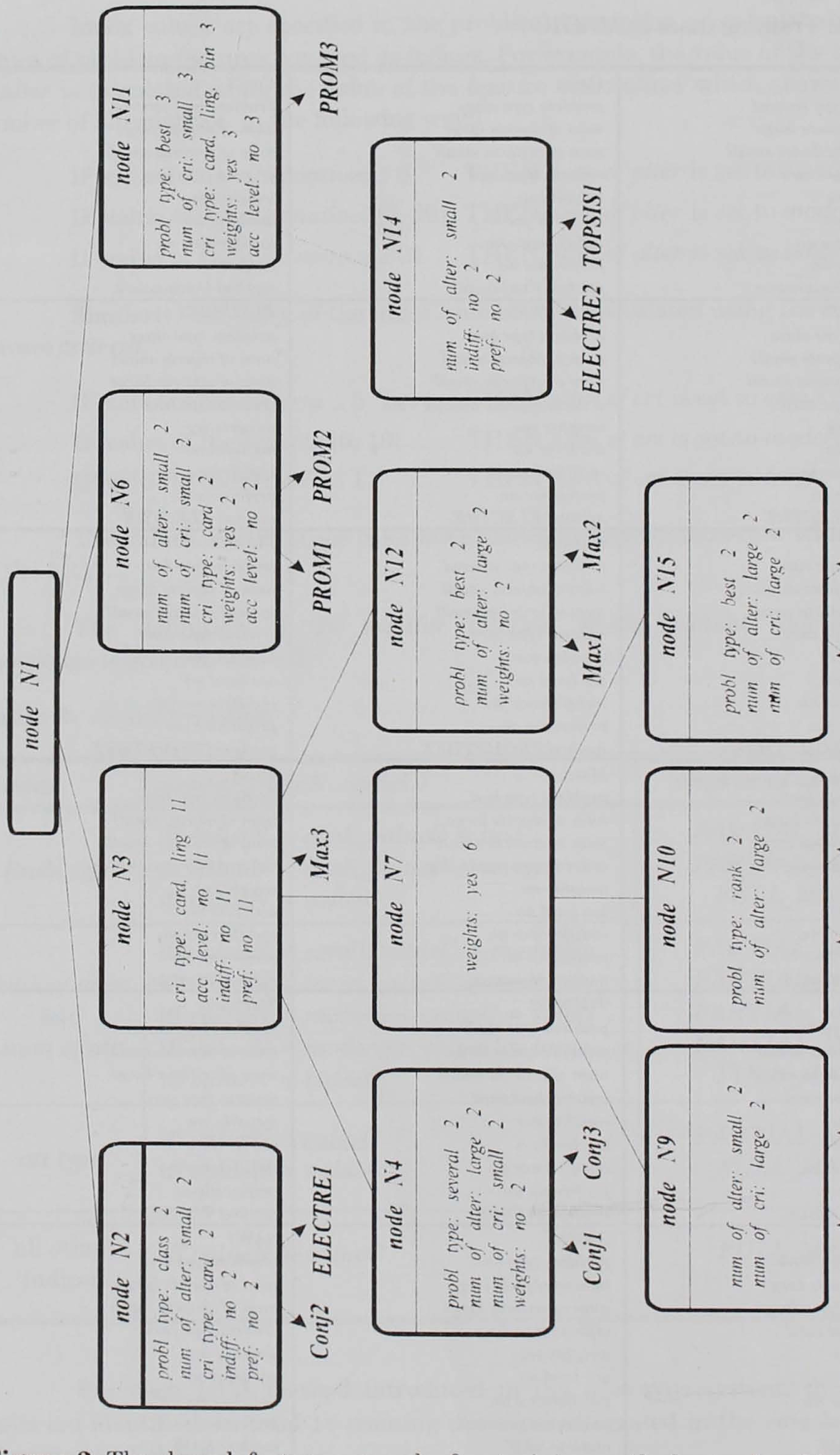

Figure 2: The shared–feature network after training on the basis of problem type

If cases are integrated in the case base on the basis of the problem type they solve, then the shared–feature network presented in Fig. 2 is obtained. The table of importance is presented in Table 5.

Table 5: Table of importance after insertion of the training cases

Index	*i*	*Conjun–ctive*	*ELECTRE*	*PROMET–HEE*	*Maximin*	*TOPSIS*	*SAW*	v_i
probl type								
clas	1	1	2	0	0	0	0	0.75
rank	2	0	0	1	0	2	1	0.61
best	3	0	1	1	3	1	2	0.5
several	4	2	0	1	0	0	0	0.75
num of alt								
small	5	0	3	2	1	2	0	0.53
moderate	6	0	0	0	0	0	0	–
large	7	3	0	1	2	1	3	0.49
num of att								
small	8	3	2	3	2	2	0	0.46
moderate	9	0	0	0	0	0	0	–
large	10	0	1	0	1	1	3	0.58
att type								
card	11	2	2	3	3	3	2	0.42
ling	12	1	2	0	2	0	1	0.53
bin	13	0	1	0	0	0	0	1
weights								
yes	14	0	3	3	0	3	3	0.5
no	15	3	0	0	3	0	0	0.71
acc level								
yes	16	1	0	0	0	0	0	1
no	17	2	3	3	3	3	3	0.41
indiff								
yes	18	0	0	2	0	0	0	1
no	19	3	3	1	3	3	3	0.42
pref								
yes	20	0	0	2	0	0	0	1
no	21	3	3	1	3	3	3	0.42

In order to enable learning and to successfully use previous experience in the new problem solving, the proposed MCA method is evaluated. The proposed method is evaluated as inadequate if:

1) Some of the input data required for the method applied are not given in the new problem. For example the DM did not specify the relative importance of criteria, but the method proposed requires them.

2) The proposed method does not include all input data. For example the DM specified the acceptance level for all criteria, but the method proposed does not use them.

3) The DM, on the basis of his subjective judgement, asks for some other method to solve the given problem.

The system continues searching the network until an adequate category is proposed. The algorithm for learning from failure is developed [12]. The new object with an adequate category is integrated in the network in the appropriate place.

3.1. Example

The proposed algorithm for classification of a new object is illustrated with an example. We have stated a very current problem of the selection of the best mobile phone station. There are now at least 10 different stations on the market. The criteria used are: price, weight, volume, and life of power supply. All of them are described by cardinal values. The problem $MS1$ which has to be solved is given as:

$$MS1 = \langle\, (\textit{probl type, best}),\ (\textit{num of alter, moderate}),\ (\textit{num of cri, small}),$$

$$(\textit{cri type, } \{\textit{card}\}),\ (\textit{weights, yes}),\ (\textit{acc level, no}),\ (\textit{indiff, no}),\ (\textit{pref, no})\,\rangle.$$

Let us suppose that the case base is presented by the shared-feature network depicted in Fig. 2.

In the first step of the algorithm the degrees of the partial match between the descendants of node $N1$ and the problem $MS1$ are calculated:

$$\textit{PartialMatchDegNC}(N2, MS1) = 0.22$$

$$\textit{PartialMatchDegNC}(N3, MS1) = 1.52$$

$$\textit{PartialMatchDegNC}(N6, MS1) = 1.96$$

$$\textit{PartialMatchDegNC}(N11, MS1) = 2.28$$

Node $N11$ is selected and searching the network continues. Further on, node *N14* is selected. Node $N14$ contains no descendants, but has two cases: $TOPSIS1$ such that $\textit{PartialMatchDegCC}(TOPSIS1, MS1) = 3.40$, and $ELECTRE2$ such that $\textit{PartialMatchDegCC}(ELECTRE2, MS1) = 3.40$. Both cases have the same degree of partial match with the new object and the system first proposes the TOPSIS method. New object $MS1$ is stored with node $N14$, because it does not match the cases stored with that node.

Let us suppose that some additional criteria are included in the alternatives' evaluation: design, reliability, audibility, existence of an interface to the human user (microphone, loudspeaker, display, etc.), and existence of an interface to other terminal equipment (PC, facsimile machine, etc.). The values of the first three criteria are linguistic, while the fourth and fifth are binary. The problem *MS*2 which has to be solved now is more complex:

*MS*2 = ((*probl type*, *best*), (*num of alter*, *moderate*),
(*num of cri*, *moderate*), (*cri type*, {*card*, *ling*, *bin*}), (*weights*, *yes*),
(*acc level*, *no*), (*indiff*, *no*), (*pref*, *no*)).

Nodes *N*11 and *N*14 are selected again. The degrees of the partial match between the cases of node *N*14 and the problem *MS*2 are calculated:

PartialMatchDegCC(*TOPSIS*1, *MS*2) = 1.91
PartialMatchDegCC(*ELECTRE*2, *MS*2) = 3.75
PartialMatchDegCC(*MS*1, *MS*2) = 2.64

Case *ELECTRE2* is selected as the most similar to the object *MS*2. The existence of linguistic and binary criteria in object *MS*2 influences the proposal of the method. Instead of the TOPSIS method, the ELECTRE method is suggested to solve the problem *MS*2. Problem *MS*2 is stored with the node *N*14. The part of the network modified after solving the problems *MS*1 and *MS*2 is shown in Fig. 3.

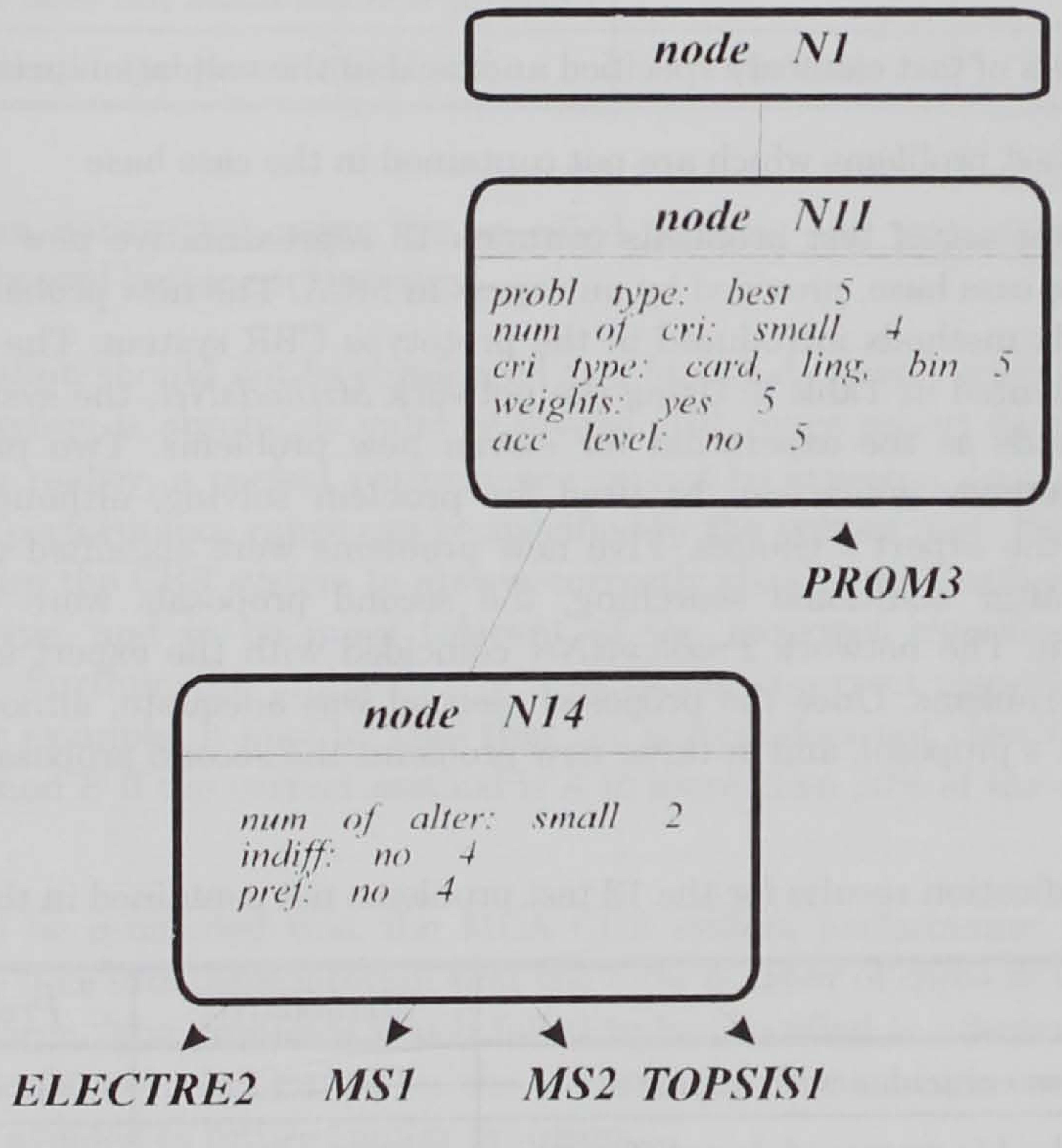

Figure 3: Part of the network modified after solving the problems *MS*1 and *MS*2

3.2. Validation

The validation of the CBR system is basically very similar to the expert systems validation with a specificity resulting from the dynamic character of CBR. The validation of an expert system generally involves running a set of representative problems through the system and comparing system outputs to the known results or human expert solutions.

The awkward structure of the MCA problems, particularly the fact that some *multicriteria* problems can be solved by more than one method, makes the validation delicate. In addition, the case base is continuously enriched with new cases causing dynamic changes of the CBR system behaviour and thus making the validation even more complex. As there is no firm theory used for the selection of an MCA method, the validation of the CBR system requires the validation of every single problem-case.

The prototype the CBR system was trained in two different ways using the 18 training cases presented in Table 4: (1) the training cases were inserted on the basis of the method used for the problem solving and (2) the training cases were inserted on the basis of the types of problems solved. In this way, two different shared-feature networks were created, *MethodsNet* and *ProblemsNet*, based on the same set of training cases.

Two sets of test cases are specified and used in the validation process.

a) The set of test problems which are not contained in the case base

The first set of test problems contains 18 representative new problems not contained in the case base, provided by an expert in MCA. The new problems should be solved by the six methods introduced in the prototype CBR system. The classification results are presented in Table 6. Using the network *MethodsNet*, the system proposed the same methods as the expert did for eleven new problems. Two problems were classified to methods which can be used for problem solving, although they were different from the expert' s choices. Five new problems were classified to inadequate methods, but after additional searching, the second proposals were equal to the expert' s opinion. The network *ProblemsNet* coincided with the expert' s proposals in fourteen new problems. Once the proposed method was adequate, although different from the expert' s proposal, and in three new problems the second proposals were equal to the expert' s.

Table 6: Classification results for the 18 test problems not contained in the case base

	MethodsNet	*ProblemsNet*
System solution coincides with expert's	11 (61.11%)	14 (77.78%)
Adequate method is proposed, but different from expert's proposal	2 (11.11%)	1 (5.55%)
Adequate method is proposed after failure	5 (27.78%)	3 (16.67%)

b) The set of test problems which are contained in the case base

The aim of the second set of test problems is to examine the behaviour of the CBR system in the classification of a new problem which matches a case already present in the case base. For that purpose, each training case is treated as a new problem. The order of their classification is taken randomly. It is tested whether the retrieval process will give a case that matches the new problem. The following results are obtained and presented in Table 7. Using the network *MethodsNet*, in 16 out of 18 new problems the system retrieved their corresponding cases from the case base. In one instance, the expected method was proposed although not by the corresponding case, and in another instance the adequate method was obtained in the second proposal. The network *ProblemsNet* showed even better performance. In all of them the correctness of reasoning was proved and the corresponding cases were selected.

Table 7: Classification results for 18 new problems contained in the case base

	MethodsNet	*ProblemsNet*
Retrieved case matches the new problem	16 (88.90%)	18 (100%)
Adequate method is proposed, although retrieved case does not mach the new problem	1 (5.55%)	0
Adequate method is proposed after failure	1 (5.55%)	0

We can notice that using the specified training and test cases the network *ProblemsNet* showed better performance.

Validation should not be considered as a binary decision variable which shows whether the system is absolutely valid or invalid [10]. Since expert systems represent or abstract the reality, a perfect performance cannot be expected. In some situations, the acceptable performance range can be specified by the system user. For example, the user may require the CBR system to always correctly classify multicriteria problems of a particular type, and to be more tolerant of the incorrect classification of other problem types. Further, the probability of a particular incorrect classification may be important. For example, it may be vital that the system does not classify the problem into MCA method B if the correct method is A in more than 20% of the cases – that is $P(B \mid A) \le 0.2$.

It can be concluded that the MCA CBR system performance is quite good, especially if we take into consideration that the total number of cases in the initial case base was not large. The problems which failed to be classified to adequate methods in the first proposals were integrated in the network in places which enable the same mistakes to be avoided in future similar problems.

4. CONCLUSION

The philosophy, concepts and techniques of CBR, originally appearing in cognitive science and computer science, are applicable in the OR domain. It is shown that CBR is useful in the classification of problems. The proposed case-based classification is illustrated in solving the important and attractive problem of the selection of an adequate MCA method for solving a given multicriteria problem. Reasoning in the CBR classification system relies on experience in the problem domain rather than on formal models and rules. The knowledge acquisition process, which is a bottleneck in the expert system development, consists mainly of memorising the objects classified. In the CBR classification system two processes are integrated: (1) the classification of a new object in which an adequate category is selected, and (2) the learning process which modifies the network structure and node descriptions as the consequences of knowledge generalisation and knowledge tuning. The system learns from both kinds of experience, success and failure, and thus avoids repeating mistakes made in solving previous problems.

Further CBR system developments will include the incorporation of additional MCA methods and consideration of a combination of methods in order to include all characteristics of new MCA problems.

REFERENCES

[1] Basadur, M., Ellspermann, S.J., and Evans, G.W., "A new methodology for formulating ill–structured problems", *Omega, Int. J. Mgmt Sci.*, 22 (6) (1994) 627–645.

[2] Brans, J.P., Vincke, Ph., and Mareschal, B., "How to select and how to rank projects: The Promethee method", *European Journal of Operational Research*, 24 (1986) 228–238.

[3] Deng, P.–S., "Using the case–based reasoning approach to the support of ill–structured decisions", *European Journal of Operational Research*, 93 (1996) 511–521.

[4] Gennari, J., Langley, P., and Fisher, D., "Models of incremental concept formation", *Artificial Intelligence, Special Volume on Machine Learning*, 40 (1–3) (1989) 11–61.

[5] Golding, A., and Rosenbloom, P., "Improving rule–based systems through case–based reasoning", in: B. Buchanan, and D. Wilkins (eds.), *Readings in Knowledge Acquisition and Learning, Automating the Construction and Improvement of Expert Systems*, Morgan Kaufmann Publishers, San Mateo, California, 1993, 759–764.

[6] Gupta, K.M., and Montazemi, A.R, "Empirical evaluation of retrieval in case–based reasoning systems using modified cosine matching function", *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 27 (5) (1997) 601–612.

[7] Hwang, C.–L., and Yoon, K., *Multiple Attribute Decision Making, Lecture Notes in Economics and Mathematical Systems*, Springer–Verlag, 1981.

[8] Kolodner, J., *Case-Based Reasoning*, Morgan–Kaufmann, 1993.

[9] Mechitov, A.I., Moshkovich, H.M., Olson, D.L., and Killingsworth, B., "Knowledge acquisition tool for case–based reasoning systems", *Expert Systems with Applications*, 9 (2) (1995) 201–212.

[10] O' Keefe, R., Balci, O., and Smith, E., "Validating expert system performance", *IEEE Expert*, Winter 2 (4) (1987) 81–90.

[11] Ozernoy, V., "Choosing the "Best" multiple criteria decision-making method", *INFOR*, 30 (2) (1992) 159–171.

[12] Petrovic, S., "Case-based reasoning in multicriteria analysis", PhD Disertation, University of Belgrade, Mathematical Faculty, 1997 (in Serbian).

[13] Petrovic, S., "Case-based reasoning in the DSS for multicriteria analysis", accepted for publication in *Journal of Decision Systems*, 7 (1998).

[14] Porter, B., Bareiss, R., and Holte, R., "Concept learning and heuristic classification in weak–theory domains", *Artificial Intelligence*, 45 (1–2) (1990) 229–263.

[15] Roy, B., "The outranking approach and the foundations of ELECTRE methods", in: C. Bana e Costa (ed.), *Readings in Multiple Criteria Decision Aid*, Springer–Verlag, 1990, 155–183.

[16] Stewart, T.J., "A critical survey on the status of multiple criteria decision making theory and practice", *Omega Int. J. of Mgmt. Sci.*, 5 (6) (1992) 569–586.

[17] Watson, I., and Marir, F., "Case-based reasoning: A review", *The Knowledge Engineering Review*, 9 (4) (1994) 327–354.