Processes and Tools for Decision Support
H.G. Sol (editor)
North-Holland Publishing Company

AN EXPERT SYSTEM FOR DECISION MAKING

M. Bohanec(1), I. Bratko(1,2), V. Rajkovic(1,3)

(1) "J. Stefan" Institute, Jamova 39, Ljubljana Yugoslavia
(2) Faculty of Electrical Engineering, "E. Kardelj University, Ljubljana, Yugoslavia
(3) School of Organisational Sciences, Kranj University of Maribor, Yugoslavia

A new decision support system, developed as an expert system, is presented. The method is formally described and discussed. Its distinguishing feature is its human orientation which is mainly reflected in the system's ability to *explain* utility calculation. A corresponding computer implementation is presented together with a practical application in decision making.

1. INTRODUCTION

Expert systems are intelligent information systems that behave, in a certain sense, as a human expert in the application domain (e.g. Michie (1979)). A major new feature introduced by the methodology of expert systems is the system's ability to *explain* its decision in user understandable terms.

Expert systems are typically composed of two modules:

(1) a knowledge–base,
(2) an inference machine.

The knowledge–base contains the knowledge about a particular problem domain. The inference machine (a) solves problems stated by the user by using the knowledge–base, and (b) generates user–oriented explanations of the solutions.

The decision making process (DMP) can be treated as the selection of a particular alternative from a given set of alternatives so as to best satisfy some given aims or goals. The problem to be solved is to evaluate alternatives, e.g. to calculate their utilities. This can be done on the basis of the decision, or utility, knowledge which a decision maker or a decision system has (White (1978)). An expert system for DM has to establish an appropriate knowledge base and use it for utility calculation. In addition to this, it has to explain the way the utility was calculated.

The explanation of utility calculation is especially interesting because DM knowledge is subjectively defined, it offers different interpretations and has some degree of uncertainty. This kind of knowledge is usually referred to as "soft knowledge" (e.g. Expert Systems (1980)). Typically, soft knowledge is also poorly formalised, nonsystemised and often changes with time. When we are dealing with soft knowledge the explanation of results seems to be the only effective way of verifying them.

We believe that the main shortcoming of the existing DM techniques is their black–box behaviour. Usually this is accompanied by a complicated and inadequate aggregation of partial utilities and numerical coding of DM information (Alter (1980)). This leads to minimal possibilities for discussing the credibility of the final results which is fundamental not only for the verification of decisions but also for negotiation among different DM groups. As a consequence, it is usually difficult to handle changes in the DMP and there are no means for dealing with uncertain or incomplete information.

It seems that these problems can be resolved by a better fit between the decision maker and the DM method, using the approach of expert systems. In this paper a formal model of a novel DM method, DECMAK, is presented. The main emphasis is on the human factor. A computer implementation of the method is presented together with an analysis of its use in the DMP and a discussion of its practical applications.

2. A FORMAL MODEL OF THE "DECMAK" METHOD

The DECMAK method was gradually developed and tested on several practical decision making situations (Efstathiou, Rajkovic (1979) ; Rajkovic, Bohanec (1980)). The method is based on the following formal model. A *semantic tree* is a triple

$$(X, F, E)$$

where:

X is a set of *performance variables* $x_1, x_2, ..., x_n$ whose domains are $D_1, ..., D_n$;

F is a set of functions $f_1, ..., f_n$, $(m < n)$ from tuples of performance variables into performance variables:

E is a set of equations $e_1, ..., e_m$ of the form:

$$x_i = f_i (x_{i1}, x_{i2}, \ldots)$$

The set E satisfies the following conditions:

(1) there is exactly one variable which never appears as an argument of any of the functions; this variable is called the *root–variable* (or "overall utility"), all other variables are non–root variables;

(2) each non–root variable appears as an argument of the functions in the equations exactly once;

(3) each variable appears in the left–hand side of the equations at most once; the variables that never appear in the left–hand side are called the *leaves*.

The leaves are also callad *basic variables*. All other variables are *aggregate variables*.

Note that the above constraints ensure that the equation set E can be representad as a tree as illustrated in Fig. 1.

The domains $D_1, ..., D_n$ of performance variables are discrete and finite. They typically consist of a few $(2 - 5)$ values. The values can be numerical or "descriptive", e.g.:

(poor, satisfactory, good)

$X = \{x_1, \ldots, x_{10}\}$
$x_1 = f_1(x_2, x_3)$
$x_2 = f_2(x_5, x_6)$
$x_3 = f_3(x_4, x_7)$
$x_4 = f_4(x_8, x_9, x_{10})$

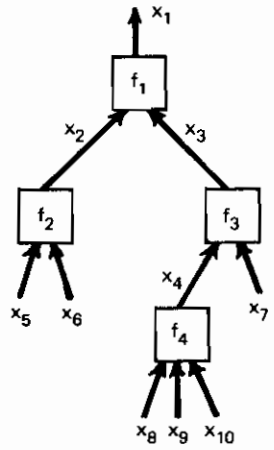


Figure 1.
An example of semantic tree. x_1 is the root variable (overall utility); $x_5,\ldots,x_{10}$ are basic variables

All the domains are assumed to be ordered. Descriptive values are introduced primarily to facilitate a more direct representation of the user's understanding of the problem. So the user is not forced to transform his usual terms into numerical values. There is no explicit intention to use numerical values as absolute measures and descriptive values as relative measures.

The constraints imposed upon the set of equations E facilitate the evaluation of variables. Any variable x can be easily evaluated by implementing the following rule. If x is an aggregate variable then

$$x_i = f_i(x_{i1}, x_{i2}, \ldots)$$

If x_i is a basic variable then its value has to be supplied by the user.

In terms of DM this model is interpreted as follows: a decision alternative is specified by an instantiation of the basic variables. This is sufficient to compute the values of all other variables, including the root–variable which represents the "overall utility" of this alternative. Alternatives are compared through their overall utilities. Thus the root variable is the overall criterion for the final ranking of alternatives. The set of functions F and equations E define how the basic descriptive features of alternatives are combined into a single utility measure. The semantic tree has to be designed by the user–expert. We call this tree "semantic" because it defines the concepts of the problem–domain and the relations among them. The tree is in fact a representation of the expert's decision knowledge about the given decision problem.

The above formal model was motivated by the assumption that such a semantic tree is a natural form for representing decision knowledge and provides a suitable framework for the expert for systematically formalising his or her decision expertise. This assumption was confirmed by the applications of the method outlined in the sequel. Advantageous features of the model are: the variables are entirely defined by the user, and the user may use descriptive rather than numerical values. The tree structure facilitates a *gradual* aggregation of the basic–

variable values through aggregate variables. As the expert defines each function in F independently of other functions, he can focus his attention entirely on this local problem. In this way the whole problem of formalising the decision knowledge is decomposed into a set of subproblems (i.e. defining particular functions in F). Each of these can be kept reasonably small by limiting the number of arguments of the functions to a few variables only. In practical applications the number of arguments was typically between two and four.

The expert defines the functions in F by specifying the function value for some chosen argument values. This is done through interaction with the program which supports this process, as described in the following section. For the undefined points in the argument–domain, the system computes the function values by a simple interpolation method.

The semantic tree model has been extended for handling unreliable and/or incomplete information, for the case that the user does not have a complete specification of alternatives. Instead of dealing with precise (single) variable–values, the model can thereby deal with distributions of values. Distributions are defined by a list of values, each of them associated with its corresponding "certainty factor". Thus for example, if some basic variable describes the documentation of a technical system and the user is not sure about the quality of the documentation of some available alternative system, we can specify a "fuzzy" estimate: good with certainty factor 0.4, satisfactory with factor 0.2, poor with factor 0.1. The certainty factors can be normalised so that their sum becomes 1, and can then be treated as probabilities.

If the values of variables $x_{i1}, x_{i2}, \ldots$ are specified as distributions and

$$x_i = f_i(x_{i1}, x_{i2}, \ldots)$$

this results in a value distribution for x_i. In DECMAK, the certainty factors propagate through the semantic tree by two alternative rules for mapping the distributions of the values of the arguments $x_{i1}, x_{i2}, \ldots$ into the distribution for x_i. The first rule is borrowed from the theory of fuzzy sets (Zadeh (1965)), and the second from probability theory.

Let V_u be the set of all vectors $\vec{v} = (v_1, v_2, \ldots)$ such that if $x_{i1} = v_1, x_{i2} = v_2, \ldots$, then

$$f(x_{i1}, x_{i2}, \ldots) = u$$

Thus for any combination $\vec{v} \in V_u$ of argument values, the variable $x_i = u$. Let us denote the certainty factors of $v_1, v_2, \ldots$ by $c(v_1)$, $c(v_2)$, ... and the certainty factor of $\vec{v} = (v_1, v_2, \ldots)$ by $c(\vec{v})$. Then by the first rule (borrowed from fuzzy sets):

$$c(\vec{v}) = \min_i(c(v_i))$$

By the probabilistic rule:

$$c(\vec{v}) = \prod_i c(v_i)$$

The certainty factor of $x_i = u$ is then by the "fuzzy rule":

$$c(u) = \max_{\vec{v} \in V} (c(\vec{v}))$$

By the probabilistic rule:

$$c(u) = \sum_{\vec{v} \in V_u} c(\vec{v})$$

In the present implementation of DECMAK the user can choose between both principles for certainty–factor propagation in the semantic trees. For an example see graphical illustrations to a practical example in section 4 where the min/max rules were used.

As for comparison of the two types of rules, the probabilistic rules are mathematically better justified. However, the fuzzy rules turned out to have practical advantages for the following reason. If the number of values is high the user may find it difficult to specify the probability distributions which would properly reflect his intuitive image, specially when the values are not independent.

3. THE COMPUTER IMPLEMENTATION OF THE MODEL

The main goal of the computer implementation of the DECMAK method was an appropriate man–machine communication, which makes the model alive in the sense of a creative partnership between man and computer. This is a distinguishing new quality in comparison with traditional decision support systems, where the role of the computer is mainly in transferring the documentation and calculation burdens from decision maker to computer.

In this section the achievement of the goal mentioned by the DECMAK program (Bohanec (1980)) will be explained.

3.1. Decision knowledge–base construction

Every decision subproblem, i.e. every function in F, can be treated separately. The decision maker (user) starts with a subproblem which he or she wants to solve. There is no prescribed order of dealing with decision subproblems. Let us take function f_3 (Fig. 1) as the first subproblem. Fig. 1 can represent a car selection decision situation. The overall utility of a car is x_1, x_2 can be the price and x_3 technical characteristics.

After user identification, the names of performance variables x_3, x_4, x_7 and corresponding domains D_3, D_4, D_7 have to be entered. In our case

x_3 = TECHNICAL–CHARACTERISTICS; D_3 = (poor, satisfactory, good, very good)
x_4 = COMFORT; D_4 = (bad, acceptable, good, very good)
x_7 = SECURITY; D_7 = (low, medium, high)

If values are descriptive, i.e. words, the corresponding compatibility functions can be entered (Zadeh (1975)); Efstathiou et al. (1979)). In the present system, compatibility functions are only used for graphical representation of results.

After this, a decision knowledge construction for our decision subproblem "TECHNICAL–CHARACTERISTICS" can start. There are three possibilities:

(1) The user states the values of arguments and the function value. This can be interpreted as formulating logical statements, or *rules*, such as:

> if SECURITY is high and COMFORT is very good
> then TECHNICAL–CHARACTERISTICS are very good.

(2) The program generates combinations of arguments and the user states corresponding values. This can be interpreted as questions:

> What is value of TECHNICAL–CHARACTERISTICS
> if SECURITY is low and COMFORT is very good?

(3) The user asks questions of the above type and the program calculates the answers using the linear multidimensional interpolation formula.

This method of knowledge–base construction can take place when some knowledge already exists in the base. In any case the user has to confirm a calculated value before it is entered as a new piece of knowledge. This is so not only because the simple interpolation formula may be inappropriate, but mainly because of handling possible discontinuities in the functions from F.

When knowledge for all the decision subproblems is so defined, the program builds up the whole semantic tree. The tree can be simply reviewed and revised whenever desired.

3.2. Evaluation of alternatives

Once a decision knowledge–base has been constructed, it can be used for the evaluation of alternatives. First the user enters the name of an alternative. Then the program asks for the values of all leaves – basic variables. In our case these are x_5, x_6, x_7, x_8, x_9 and x_{10}.

If the user is not certain about values of variables with respect to the alternative being evaluated he can put several values together with appropriate certainty or probability factors. For example:

SECURITY: high/0.8, medium/0.4

When all the leaf variables have been defined, evaluation can start. The program calculates the overall utilities as a single value. For example:

x_1 : OVERALLCARUTILITY : acceptable

If the data was of a fuzzy or a probabilistic nature, the overall utility is expressed as a distribution of values.

3.3. Explanation of evaluation

Once the final overall utility has been calculated, the usual question is: how and why was the utility obtained? The user can follow the utility calculation along the semantic tree. He follows the decomposition of overall utility into partial utilities. Every partial utility can be examined separately. This is done by displaying rules (points of a function f) which were taken into account during the utility calculation. Whether a rule already existed or it was calculated during the evaluation is also displayed.

Such an explanation is especially useful in group decision making where negotiation among different interests takes place. In this case the negotiation moves from overall utilities to distinguishing features of alternatives in corresponding nodes of the semantic tree.

3.4. Some technical data about the DECMAK program

The program is implemented on PDP–11 (under RT–11 or RSX–11 operating systems) and DEC–10 (under TOPS–10) computers. It is written in Pascal and can be easily transferred to other machines. Its size is 3500 lines (about 100 subprograms).

A special point of the implementation is its user orientation. It has a HELP system, and was characterised as a "friendly system" by its users.

4. A PRACTICAL EXAMPLE

4.1. A decision problem

One real decision problem, where the DECMAK method was used, was:

In a factory with about 2000 employees, a purchase of a computer system was planned to be used in their administration and research work. The change analysis showed that they needed a computer with about 140 interactive terminals, 10 printers and 700 Mbytes of disk storage. The decision problem was to choose a computer among alternative offers.

This problem was solved in the following steps:

(1) establishing a decision making group,
(2) change analysis,
(3) identifying alternatives,
(4) identification of performance variables and semantic tree construction.
(5) definition of the decision–knowledge functions,
(6) analysis of the alternatives and evaluation,
(7) explaining the results of evaluation,
(8) implementation of the chosen alternative.

Steps 1, 2, 3 and 8 are highly dependent on the problem environment. As they are not directly related to the DECMAK method itself they will not be discussed here. Steps 4 to 7 using the DECMAK method will be further discussed in more detail.

4.2. Performance variable identification and semantic tree construction

The performance variable SYSTEM (x_1) represents the quality of the computer (overall utility). Its domain is:

$$D_1 = \text{(unacceptable, acceptable, good, very good)}$$

The quality of a computer depends on economic conditions, technical features and personnel. So three new performance variables are introduced:

x_2 = ECONOM; D_2 = (unacceptable, acceptable, good, very good)
x_3 = TECHNICAL; D_3 = (bad, acceptable, good, very good)
x_4 = PERSONNEL; D_4 = (bad, acceptable, good, excellent)

and

$$\text{SYSTEM} = f_1(\text{ECONOM, TECHNICAL, PERSONNEL})$$

Each of these variables depends on other characteristics, e.g. TECHNICAL depends on hardware and software.

By this top–down approach a semantic tree with 9 nodes and 20 leaf variables was eventually designed.

A semantic tree is the final result of the analysis of the decision problem (steps 4 and 5). If our knowledge about the problem is good enough, we can construct semantic trees of this size quickly – typically in 2 or 3 hours.

4.3. Definition of decision–knowledge functions

In our case we defined all the 9 decision–knowledge functions interactively using the DECMAK program. The usual heuristic for defining a function is:

(1) Enter rules which seem to be "realistic", that means that we expect the situation the rule fits. For example:

if ECONOM = acceptable
and TECHNICAL = good
and PERSONNEL = acceptable
then SYSTEM = acceptable.

(2) Let the program ask some questions which are close to the rule we entered.
(3) Test the quality of the current knowledge–base with some questions which we expect to occur during evaluation. Bad answers mean that we have to refine our knowledge–base by iterating this heuristic once more.

For the definition of all 9 functions in our case we spent 2 times 3 hours of interactive work with the DECMAK system. The function f_1, one of the 9 functions in our semantic tree, was thus specified as shown in Table 1.

Table 1.
An excerpt from the rule–defined function f, where SYSTEM = f (ECONOM, TECHNICAL, PERSONNEL).

ECONOM	TECHNICAL	PERSONNEL	SYSTEM
unacceptable	bad	bad	unacceptable
unacceptable	bad	acceptable	unacceptable
...	...	...	...
acceptable	good	good	acceptable
acceptable	good	excellent	good
acceptable	very good	bad	acceptable
acceptable	very good	good	good
acceptable	very good	very good	good
...	...	...	...
very good	very good	acceptable	good
very good	very good	excellent	very good

4.4. Analysis of the alternatives and evaluation

The next step in solving our decision problem was to determine the values of the leaf variables for all alternatives (5 computer systems in our case) and to evaluate them. When all the data are present, this is simple and straight–forward. But in some of the offers we could not find all the data that were needed for precise and certain evaluation. In such cases we had to define the value of a variable as a distribution of values that, in our judgement, best fitted the real situation.

An example: for one of the computers the capacity of the disk storage per unit was not explicitly specified. As we know nothing about its disk storage, initially we let the values for this parameter be:

DISKS : 0 – 100 Mb, 100 – 300 Mb, 300 – 600 Mb, more than 600 Mb

But on second thoughts we decided that the first and the last values were surely not possible, and that the most certain value was 100 – 300 Mb, but the size was possibly in the range 300 – 600 Mb, too. So we stated for the parameter DISKS the more precise value:

DISKS: 100 – 300 Mb/0.9, 300 – 600 Mb/0.4

The analysis of the alternatives and their evaluation (which was repeated with different values for the same alternative in order to check the sensitivity of the evaluation) took 2 times 2 hours of discussion and interactive work with the system.

4.5. Explaining the results of the evaluation

The evaluation of the computer systems gave us the following (note that the "fuzzy" rule for certainty–factor propagation was used):

COMPUTER	VALUE	CERTAINTY FACTOR
A	acceptable	1.0
B	unacceptable	1.0
C	acceptable	0.3
	good	0.7
D	acceptable	0.6
	good	0.15
E	good	0.5

Which of the systems is the best and why?

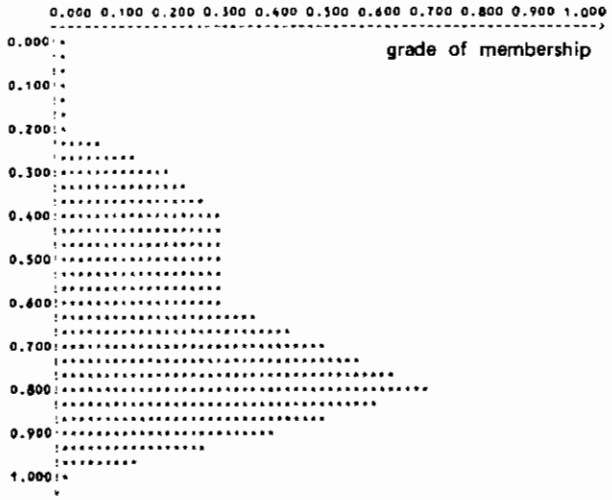
The first step is to answer the question: do the above results agree with our intuitive expectations? If they do not, we can go through the tree and the rules and check if the knowledge functions work properly. If not, we have to modify a function in question and to repeat the evaluation. In our case the results agreed with our expectations.

In the results we find that computer E got the highest single value (although with the relatively small certainty factor 0.5). Alternative C was also evaluated as *good* with an even greater certainty factor of 0.7, but it also got the value *acceptable*, which involves a greater degree of risk in choosing the alternative C. Graphical illustration of the situation is in Fig. 2. The vertical axis corresponds to SYSTEM overall utility and the horizontal to the grade of membership. The left diagram shows the fuzzy utility of computer E where "good" is defined by a fuzzy distribution of overall system utility values in the interval between 0 and 1. The certainty factor is taken into account by using the "min" rule (Zadeh (1975), Efstathiou et al. (1979)). The corresponding diagram for computer C is on the right side of Fig. 2.

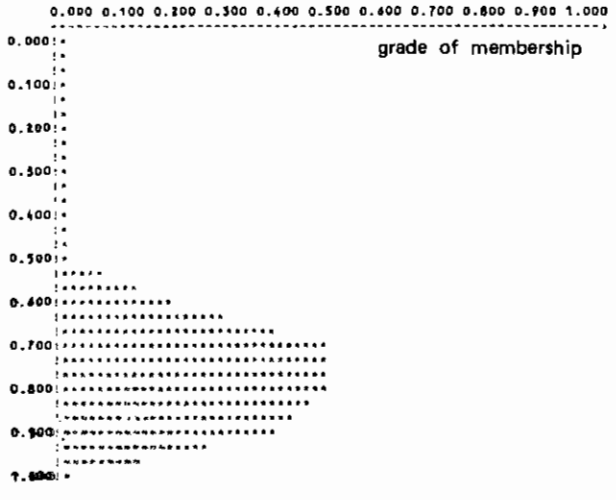
As this situation does not look clear enough, further analysis is required.

We must look at the other utility values derived for the alternatives:

NODE	COMPUTER	VALUE	CERTAINTY FACTOR
ECONOM	C	good	1.0
	E	acceptable	1.0
TECHNICAL	C	good	0.7
	E	very good	0.6
PERSONNEL	C	acceptable	0.3
		good	0.7
	E	good	0.5
		very good	0.5

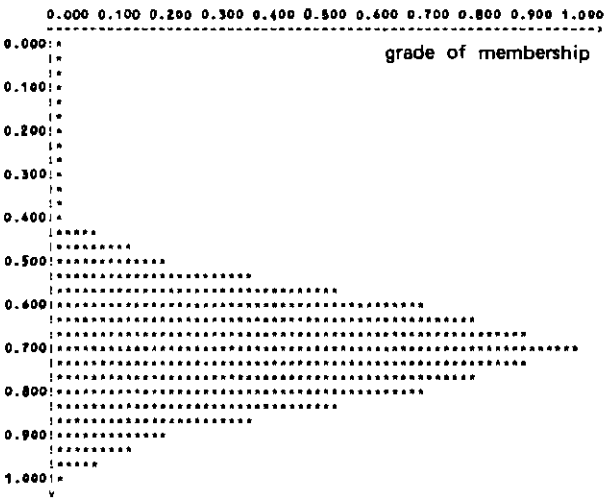


SYSTEM (C)



SYSTEM (E)

Fig. 2. Graphical illustration of overall utilities for systems C and E



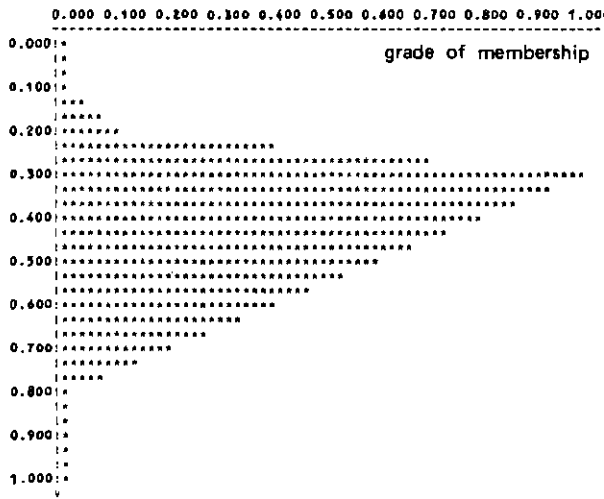


Fig. 3. Graphical illustration of economic conditions for systems C and E

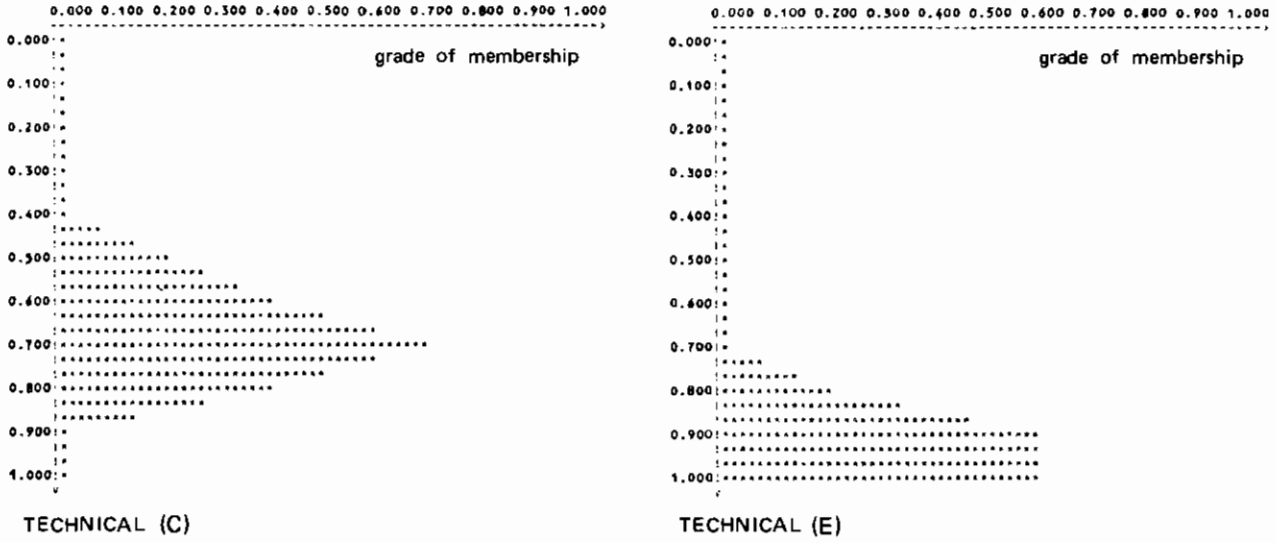
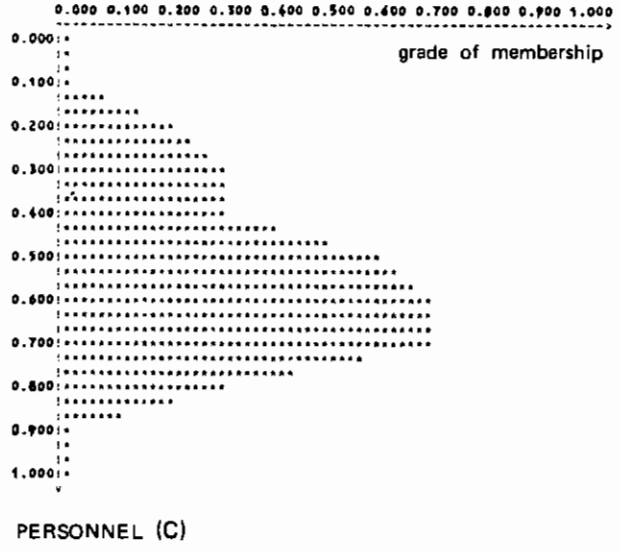


Fig. 4. Graphical illustration of technical characteristics of systems C and E



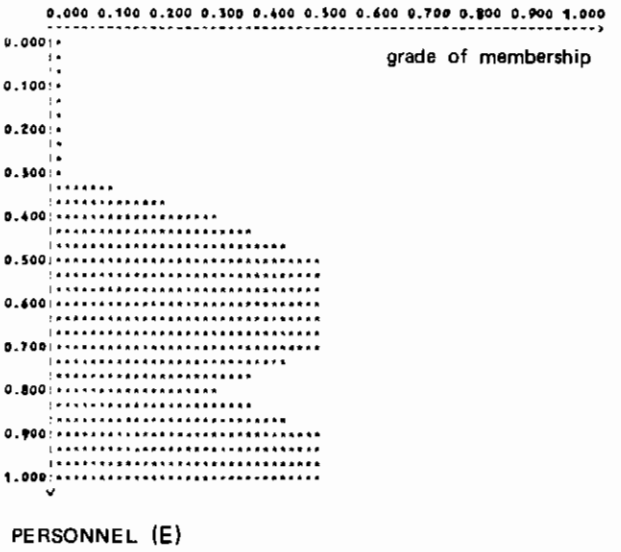


Fig. 5. Graphical illustration of personnel requirements for systems C and E

So, in technical characteristics and personnel, E is better and in economic criteria C is better. The same conclusion can be drawn from Figs. 3, 4 and 5 which show the fuzzy distributions of values of ECONOM, TECHNICAL, PERSONNEL for both computers. If we agree that technical characteristics and personnel of the above values are more important than economic criteria, then E should be chosen, otherwise C. If we still cannot decide, further similar top–down comparison can be done.

5. CONCLUSION

The main advantage of the DECMAK system, working as an expert system, is in its user (decision maker) orientation. The method is fully transparent for ordinary users. This can be explained by the following features of the method:

- direct aggregation of utilities,
- possible usage of descriptive variables and values,
- interactive use of the DECMAK program,
- utilities can be expressed as distributions of values,
- easy change handling,
- participation of different interest groups,
- explanation of utility calculations.

For implementation of expert systems a new way of thinking is needed. This should be different from the traditional black–box decision making where a computer is needed primarily for number crunching and represents a barrier behind which a user's opinion may be manipulated.

The DECMAK system was used in several decision making situations such as computer hardware and software selection. The above mentioned features are particularly important in group decision making. In these cases, decision problems should be well structured and documented. Explanation of the decisions is extremely useful in negotiation among different decision interests. The possibility of forgeting important things is reduced. Crucial points are recognised which are essential in the implementation phase of the alternative chosen.

REFERENCES:

[1] Alter, S., Decision Support Systems: Current Practice and Continuing Challenges (Addison Wesley, 1980).

[2] Bohanec, M., DECMAK – Program Documentation and User's Guide, IJS Report DP-2266, J. Stefan Institute, Ljubljana (1980).

[3] Efstathiou, J., Hawgood, J., Rajkovic, V., Verbal Measures and Inseparable Multidimensional Utility in System Evaluation, in: Schneider, H.–J. (ed.), Formal Models and Practical Tools for Information Systems Design, Holland, 1979)

[4] Efstathiou, J., Rajkovic, V., Multiattribute Decisionmaking Using a Fuzzy Heuristic Approach, IEEE Trans. on Systems, Man and Cybernetics, 9 (1979) 326–333.

[5] Expert Systems, Proc. 69th Infotech State of the Art Conference (London, 1980)

[6] Michie, D. (Ed.), Expert Systems in the Microelectronic Age (Edinburgh, University Press, 1979)

[7] Rajkovic, V., Bohanec, M., A Cybernetic Model of the Computer Aided Decision Making Process, Proc. of 9. Int. Congress on Cybernetics, Namur (1980), 185–189.

[8] White, J.D., Decision Methodology (J.Wiley & Sons, 1978)

[9] Zadeh, L.A., Fuzzy Sets, Information and Control, 8 (1965) 338–353.

[10] Zadeh, L.A., The Concept of a Linguistic Variable and Its Application to Approximate Reasoning – I, II, III, Information Sciences, 8 and 9 (1975), 199–251, 301–357, 43–80.