Interactive Computer System for Solving Problems with Multiple Criteria*

Ivo Marinchev[1] and Leoneed Kirilov[2]

Abstract – **A Computer System for solving Multiple Attribute Decision Making Problems (MADMP) is presented in the paper. It is assumed that the set of alternatives is explicitly known and finite one. The attributes are assumed to be given as numerical values. The system incorporates some of the well-known and classical methods for solving MADMP. These are methods ELECTRE, PROMETHEE. It also includes the RDM (Reference Direction Method).**

Keywords – **decision making, multiple attributes, multiple criteria, JAVA**

I. Decision Making with Multiple Objectives and Decision Support

Multiple Criteria Decision Making (MCDM) is a choice among a set of decisions/variants/alternatives made by an expert on given problem according to multiple criteria/objectives/goals. The expert is called Decision Maker (DM). The set of alternatives is generated from multicriteria model according to given rule. Usually it is an optimization procedure. The multiple criteria models are natural generalization of single criteria ones. The set of objectives is optimized in total. For more details, different models, and approaches for solving them the reader can refer to [1,6,7].

Decision Support System (DSS) is every interactive computer system designed to support the process of Decision Making (DM) [8]. Its basic purpose is to support but not to substitute the DM in the process of the decision making (DM). The DSS has three basic components:

1. Model;
2. Optimization module(s) (solver);
3. Man-machine interface or shortly interface.

The model which is usually a mathematically one, is hidden for the user. The analyst makes the choice of the model. The analyst also chooses a solving method, constructs a suitable model for given problem with the help of the DM. He makes the conception for the DSS.

The optimization module implements one or more methods for solving the model. For general purposes a sufficiently general model is selected/constructed. The latter is solved by appropriate method(s). But when trying to solve real problems for real users a modification of a model is done and possibly a modification of a method to solve it. All mentioned is a responsibility of an analyst in cooperation with the DM (an expert of a given problem).

An interface is an important element of a DSS. At first, it is viewed from the DM. Therefore it has to be sufficiently attractive. Sometimes one DSS could be chosen on the base of its interface. The interface has to be full of matter and convenient (user-friendly).

Usually an input/output/editing module is also available to the DSS. The following natural requirement follows from the said above.

If one DSS is designed for solving a very specific problem, then it is not easy to use it for another problem without modification. On the other hand, if it solves a general model, then sometimes it would be necessary to customize it for solving certain real problems.

Usually a sufficiently general model is realized that can solve a number of problems, for example – linear model, or nonlinear, or integer, etc. Such DSS we name *universal systems*. The other class DSS we name *specialized systems*. The user uses such DSS to solve his/her problem. The analyst constructs a model and an appropriate method. Maintenance in the future is provided.

Multi-Criteria DSS (MCDSS) is a DSS with multi-objective model(s). Multiple objective models are generalization of a single objective ones. But are they better alternative? What are their disadvantages?

One such disadvantage is related to multi-objectivity. This leads to non-uniqueness of the produced optimal solutions in the objective space. Indecision arises about the ”best” solution. The question is, what is better to the DM? To use single objective model with one optimal solution which could be very close to real solution or to use multi-objective model with a set of solutions. The DM has to choose one of them on the base of compromises.

The other question is about convergence of multi-objective methods. This subject is not investigated completely. This is compensated by the fact that one DM could intuitively find satisfactory solution for a small number of iterations. Most multi-objective DSS provide tools for avoiding cycling. Single-objective methods are well studied for convergence as a rule. But when trying to solve real problem the question is - to use complicated model without guarantee for finding optimal solution or to use simple model with optimal but not real solution.

*The work report in this paper has been partially supported by Project IIT-010051 “Advanced methods and tools for knowledge representation and processing” and Project IIT-010049 “MCDM methods”.

[1]Ivo Marinchev is with the Institute of Information Technologies, Bulgarian Academy of Sciences, Acad. G. Bonchev Str., Bl. 29A, 1113 Sofia, Bulgaria. E-mail: ivo_m@iinf.bas.bg

[2]Leoneed Kirilov is with the Institute of Information Technologies, Bulgarian Academy of Sciences, Acad. G. Bonchev Str., Bl. 29A, 1113 Sofia, Bulgaria. E-mail: lkirilov@iinf.bas.bg

Further, most real DMs are indecisive to use MCDSS in their activities. The available MCDSS are experimental. They can be used to prove the efficiency of a given method or they can be used for educational purposes. Some of them are called commercial according to their authors but not by the software companies [3].

The conclusion from the above is that MCDM approach still has not taken its place among other optimization approaches for solving real problems.

II. Multiple Attribute Decision Making

Multiple Attribute Decision Making (MADM) is one of the two major fields of Multiple Criteria Decision Making. The other one is Multiple Objective Mathematical Programming. In MADM it is assumed that the feasible set consists of a finite number explicitly known alternatives. A best one has to be chosen according to the set of k ($k \geq 2$) objectives [1].

Thus, the problem has simply the matrix formulation of dimension (n, k) – decision matrix, where the elements a_{ij}, $i = 1, 2, ..., n$ and $j = 1, 2, ..., k$ are the values of i-th alternative according to the j-th attribute (objective). We assume that the rows of a matrix form the alternatives and the columns form the objectives.

A number of approaches are available for solving this problem. They can be classified according to the type of information required by the DM and its basic features, and according to the strategy used to find solution(s). Some basic approaches can be found in [2].

What is the best method for solving MADM is a "nonsense". The choosing of a method depends on the nature of the problem solved, on the preferences of the Decision Maker (DM), and other factors.

A number of computer systems for solving MADM problems (MADMP) are available. Note that the users better know these problems and respective computer systems than the MCDM problems. Also commercial systems for this class dominate. But the accent is mainly on the method implementation without taking into account the user-friendliness of the interface. Example of such well-known systems are ELECTRE I-IV [9,10], PROMCALC and GAIA [11, 12], TRIMAP [13]. Also EXPERT CHOICE of Saaty [14] – an implementation of Analytic Hierarchy Process, proposed by the same author.

In this paper we present a Decision Support System (DSS) for solving basic problem of MADM. It incorporates a number of well-known, classical and effective methods for their solving. These are ELECTRE and PROMETHEE methods [3,4]. It also includes Reference Direction Method (RDM) [5]. As it is known, RDM is basically designed for solving Multiple Objective Mathematical Programming Problems. Here a version for MADM problems is realized.

The reason for choosing such method is that most methods for solving MADM are non-interactive. Also the input information, requested by the DM, has not clear interpretation for him/her. The proposed result (alternative(s)) has not explicit connection with the input information. On the other hand, most Interactive Methods (IMs) have clear and simple interpretable dialog. After that a series of solutions are generated. If the compromise solution is not among them, the process repeats.

The ELECTRE method for example uses the following strategy for searching best alternative. On the base of a set of weights (input parameters) for each attribute ELECTRE constructs "outranking relationship". This means that for two nondominated alternatives A and B (that are incomparable in general) the DM could accept for example A to be more acceptable than B. The result is that:

1. The dominated alternatives are simply eliminated;
2. The nondominated alternatives are outranked, i.e. a subset of nondominated alternatives is presented to the DM. In the ideal case this is one alternative.

ELECTRE with its simple logic, full using of information in the decision matrix is one of the best methods [2].

III. Motivation for Selecting Java Technologies for Implementation of Our System

We have implemented our system using Java programming language. The reason of making this decision is that Java is not only general purpose programming language but complete development platform that has enormous diversity of APIs (Application Programming Interfaces) supporting almost all contemporary programming technologies. Most of them are embedded in the standard programming libraries that are part of any standard compliant Java 2 virtual machine.

One technology that we consider very suitable for our system is *Java Web Start*. It allows launching applications through the network using recently introduced Java network launching protocol. Java Web Start applications are extensions of the Java applets technology that have some very useful advantages:

1. Applications can be launched through a web browser but they no longer depend on the Java virtual machine built in it, hence they are not restricted to Java 1.1 which is default JVM embedded in the Internet Explorer browser (i.e. when no Java 2 plug-in is used).
2. Applications are cached locally and on any subsequent activation they are started from the local hard drive. At the same time the system launcher checks their web site whether new version is available. If so it downloads the newer version and replaces the old one with it. This solves the issue with versioning (supporting many different versions of a certain application or its components) in a very graceful manner.
3. Like Java applets, Java web start applications are executed in the secure sandbox that isolates them from the local system resources. This restriction protects the user from executing malicious code. But Java web start technology allows restricted (user confirmation is required) access to local file system to store any persistent data on it. This solves the big issue with privacy because users

often prefer keeping their data locally and not giving access to it to anyone else.

Indeed, Java Web Start technology was one of the main motives in selecting Java technologies for the implementation of our system. Although the current version of it is developed as a stand-alone application in the future it will be very easy to separate it in two components according to the client-server computational model. The thin client component implementing user interface and user interactions code and server component performing all numerical computations. The client component will be Java web start application that will be executed on the user's local machine. The server component will solve optimization tasks sent by the client components and will return the final results.

And last but not least is Java's cross platform compatibility. The language completely justifies Sun's "Write once, run anywhere" paradigm. We develop our system mainly on the Windows workstations, but the final system can be deployed and used on any Java 2 enabled platform – Windows, Linux, FreeBSD, Solaris, Mac OS, etc. At the same time client part and server part can be executed on different operating system allowing any available system to be used as a client or server.

IV. System Architecture

Current version of the system is implemented as a stand-alone Java application. It uses Swing library for graphic user interface (GUI). We have selected this library because it supports (and enforces) the use of the model-view programming model in user interface programming. The later allows clean separation between the internal data structures (data model) and processing and their visual representation and user interactions with them.

Internally our implementation is organized according MV (model-view) architecture, which is supported by the GUI library. The basic idea behind this architecture is that every entity that the program processes is represented with some sort of internal model – a collection of interrelated data structures. This internal model has one or more views associated with it. Every view is a user interface component that renders the model on some sort of external device (usually monitor screen). The separation between data processing and data visualization has many advantages. The most important of them are:

1. One data model can have many views that represent its different aspects. For example in our program every table (from the Tabbed Pane control) is a different view to the same data model. Every view renders only the data needed for its corresponding optimization method and ignores all irrelevant information (parameters).
2. If some data in the data model is changed by the user (through any of the views associated with it) or by some processing algorithm, it becomes immediately visible in all associated views. In practice it is implemented with the subscription-notification mechanism (listeners in Java terms) built in the Swing library. By means of it every view that is registered to receive certain events (changes in the data model data in our case) is notified for the changes. Receiving this event the corresponding view repaints its canvas to reflect the new data model state.
3. Data model and view can be changed independently. The data exchange between the model and the view is channeled through a well-defined interface (TableModel interface) that is part of the Swing library. This architecture allows the model and the view internals (data structures and algorithms used) to be changed independently as far as the interface is properly implemented. So the system is easily extensible and different peoples can work on the different part of it simultaneously without the complications of implementation synchronization.

ELECTRE | PROMETEE I | PROMETHEE II

	F1	F2
Minimize	max	max
Weights	0.5	0.5
A1	0.0	20.0
A2	1.0	18.0
A3	3.0	15.0
A4	5.0	12.0
A5	7.0	10.0
A6	9.0	5.0
A7	10.0	0.0

ELECTRE | PROMETEE I | PROMETHEE II

	F1	F2
Minimize	max	max
Weights	0.5	0.5
q	0.5	0.5
p	1.0	1.0
s	0.5	0.5
TYPE (I,II,III,IV,V,VI)	TYPE I	TYPE I
A1	0.0	20.0
A2	1.0	18.0
A3	3.0	15.0
A4	5.0	12.0
A5	7.0	10.0
A6	9.0	5.0
A7	10.0	0.0

Fig. 1. Different views of a single data model

V. User Interface

We have created the user interface of our system that conforms to the following preliminary defined criteria:

1. The interface must organize a lot of information (decision matrixes) in the least possible space.

2. The interface must be easily extensible in order to be possible to add new optimization algorithms with little efforts.
3. All optimization algorithms (methods) must have uniform look and feel.
4. The interface must be easy to understand and use.

In order to comply with these criteria we have selected to organize the user interface with the use of the Tabbed Pane (Tab Strip) control. This control is broadly used when a lot of categorized information has to be confined to single screen (usually in options and/or preferences dialogs) and it is in perfect accordance with all specified criteria. Every tab pane is associated with a given optimization method and displays its related data in the form of generalized matrix (containing not only the alternatives and criteria but also all constraints or parameters used in the corresponding optimization algorithm.

VI. An Illustrative Example

We shall demonstrate the work of the system on the following example, described in [2]. A country has to buy a fleet of jet fighters from the U.S. The Pentagon officials offered the characteristic information about four models of fighters. The Air Force analyst team of the country agreed that six characteristics should be considered: F1 – maximum speed; F2 – ferry range, F3 – maximum payload, F4 – purchasing cost, F5 – reliability, F6 – maneuverability. The values of each attribute for each alternative are given in table 1.

Table 1. A problem for selection fighter aircraft

	F1 Maximum speed (Mach)	F2 Ferry range (NM)	F3 Maximum payload (pounds)	F4 Acquisition cost ($*10^6)	F5 Reliability High - low	F6 Maneuverability High – low
A1	2.0	1500	20000	5.5	Average	Very high
A2	2.5	2700	18000	6.5	Low	Average
A3	1.8	2000	21000	4.5	High	High
A4	2.2	1800	20000	5.0	Average	Average

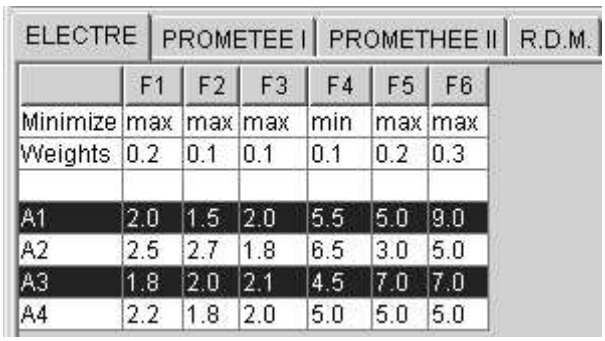

Fig. 2. The ELECTRE solution of the problem

As it is seen the 5th and 6th attributes are qualitative. They are converted to quantitative ones by using bipolar scale (interval scale). Using 10-point scale and setting 0 points to the minimum attribute value and 10 points to the maximum attribute value we receive the following relations (see for more details [2]) – Very low (1), Low (2), Average (5), High (7), Very high (9). Fig. 2 shows the ELECTRE solution of the above problem.

VII. Conclusion

In the present paper we present decision support system for solving multiple attribute analysis problems. The main features of the system are:

1. It implements several multiple criteria methods (ELECTRE, PROMETHEE, and RDM), allowing the decision-maker to compare the solutions obtained with the different of them.
2. It is implemented in JAVA hence it is completely portable and can be used on any Java enabled platform (including web browsers).
3. The unified interface allows the user to work with the different methods with ease.

References

[1] R. Steuer, *Multiple Criteria Optimization: Theory, Computation and Application*, John Wiley & Sons, New York, 1986.

[2] Ch. Hwang, K. Yoon, *Multiple Attribute Decision Making: Methods and Applications*, Springer-Verlag, Berlin, 1981.

[3] Ph. Vincke, *Multicriteria Decision Aid*, John Wiley & Sons, New York, 1992.

[4] C. Bana e Costa (Ed.), *Readings in Multiple Criteria Decision Aid*, Springer-Verlag, Berlin, 1990.

[5] S.C. Narula, L. Kirilov, V. Vassilev, "Reference Direction Approach for Solving Multiple Objective Nonlinear Programming Problems", *IEEE Transactions on Systems, Man, and Cybernetics*, vol.24, No5, pp.804-806, 1994.

[6] Sawaragi Y., Nakayama H., Tanino T., *Theory of multiobjective optimization*, Acad. Press, Inc., Orlando, Florida, 1985.

[7] Miettinen K., *Nonlinear multiobjective optimization*, Kluwer, Norwell. USA, 1999.

[8] Eom H., "The Current State of Multiple Criteria Decision Support Systems", *Human Systems Management* 8, 113-119, 1989.

[9] Roy B., Skalka J., "Electre IS – Aspects methodologiques et guide d'utilization". Document du Lamsade 30, Univ. Paris Dauphin, 1984.

[10] Skalka J., Bouyssou D., Bernabeu Y., "ELECTRE III et IV: aspects methodologiques et guide d'utilization". Document du Lamsade 25, Univ. Paris Dauphin, 1984.

[11] Mareschal B., "Weight Stability Intervals in Multicriteria Decision Aid", *European J. of Operational Research* 33, 54-64, 1988.

[12] Mareschal B., Brans J., "Geometrical Represenation for MCDA", *European J. of Operational Research* 34, 69-77, 1988.

[13] Climaco J., Henggeler Antunes C., "TRIMAP: an Interactive Tricriteria Linear Programming Package", *Foundations of Control Engineering* 12(3), 101-119, 1987.

[14] Saaty T., *The Analytic Hierarchy Process*, McGraw-Hill, New York, 1980.