CO_OP2.0 DISTRIBUTED DECISION SUPPORT SYSTEM FOR STRATEGIC PLANNING

by

Skindilias Christos

March 1986

Thesis Advisor: Tung X. Bui

Approved for public release; distribution is unlimited
This thesis focuses on the implementation and use of a multiple criteria, multiple-user Decision Support System capable of supporting distributed strategic decision making. An example of the use of such a distributed decision support system for selecting warships for the Hellenic Navy demonstrates the usefulness of the proposed group DSS.
Co_oP 2.0
Distributed Decision Support System for
Strategic Planning

by

Christos K. Skindilias
Lieutenant, Hellenic Navy
B.S., Naval Academy of Greece, 1975

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
March 1986

Author: Christos K. Skindilias

Approved by: Tung X. Bui
Tung X. Bui, Thesis Advisor

T.R Sivasankan, Second Reader

Willis R. Greer Jr., Chairman,
Department of Administrative Sciences

Kneale T Marshall,
Dean of Information and Policy Sciences
ABSTRACT

This thesis focuses on the implementation and use of a multiple criteria, multiple-user Decision Support System capable of supporting distributed strategic decision making. An example of the use of such a distributed decision support system for selecting warships for the Hellenic Navy demonstrates the usefulness of the proposed group DSS.
# TABLE OF CONTENTS

I. INTRODUCTION ................................................. 10
   A. DEFINITION OF THE PROBLEM .......................... 10
   B. SCOPE OF THE RESEARCH ............................... 11
   C. ORGANIZATION OF THE THESIS ......................... 11

II. A FRAMEWORK FOR IMPLEMENTING REMOTE MULTIPERSON
    GROUP DECISION SUPPORT SYSTEMS ..................... 13
   A. DEFINITIONS AND BASIC CONCEPTS ..................... 13
      1. Definitions of Group DSS .......................... 13
      2. Assumptions ....................................... 15
      3. Communications Issues in Distributed
         Decision Making ................................. 17
            a. Need for Format Transparent
               Information Exchange .......................... 17
            b. Limited versus Free Information
               Exchange ....................................... 17
            c. Evolving Pattern of Communication
               Requirements ................................. 18
      4. The Role of the Communications Component ......... 18
            a. The Coordinator Role .......................... 19
            b. The Detective Role: .......................... 20
            c. The Inventor Role: .......................... 20
   B. AN ARCHITECTURE FOR GROUP DSS ................... 21

III. THE MODEL COMPONENT ...................................... 23
   A. THE MODEL BASE .......................................... 23
      1. The Model Base for Individual Decision
         Making ............................................. 23
         a. The ELECTRE Method: Basic Concepts ........ 24
         b. The Analytic Hierarchy Process: Basic Concepts .. 25
      2. The Model Base for Group Decision Making ....... 26
   B. THE MODEL MANAGER .................................... 28
      1. Integration of Models .............................. 29
      2. Combined Use of MCDM and Techniques of
         Aggregation of Preferences ..................... 30
LIST OF TABLES

1. LOGICAL DATA BASE RECORDS FOR STORE THE DATA OF A PROBLEM ................. 38
2. LOGICAL DATA BASE RECORDS FOR STORE THE DATA OF A PROBLEM .................. 39
3. LOGICAL DATA BASE RECORDS FOR STORE THE DATA OF A PROBLEM .................. 41
4. LOGICAL DATA BASE RECORDS FOR STORE THE DATA OF A PROBLEM .................. 42
5. EFFORT DISTRIBUTION .................. 48
6. SPECIFICATIONS OF THE WAR SHIPS .................. 52
7. SPECIFICATIONS OF THE WAR SHIPS .................. 53
LIST OF FIGURES

1. Main Menu ............................................ 22
2. The Co_oP Decision Making Process .................. 32
3. The Help Menu ......................................... 36
4. Step 2 Group Norm Definition .......................... 188
5. Step 1 Group Problem Definition ....................... 189
6. User 1 / Problem Initiation ............................ 190
7. User 1 / Prioritization of Evaluation Criteria at the First Level .... 191
8. User 1 / Prioritization of Evaluation Criteria at Level 2 For Criteria 1.1 to 1.4 .......... 192
9. User 1 / Prioritization of Evaluation Criteria at Level 3 for Criteria 1.3.1 and 1.3.2 .......... 193
10. User 2 / Prioritization of Evaluation Criteria for Level 1 ..................... 194
11. User 2 / Prioritization of Evaluation Criteria at Level 2 for Criteria 2.1 and 2.2 ........ 195
12. User 2 / Prioritization of Evaluation Criteria at Level 3 for Criteria 2.1.1 to 2.1.3 .......... 196
13. User 3 / Prioritization of Evaluation Criteria at Level 1 ..................... 197
14. User 3 / Prioritization of Evaluation Criteria at Level 2 for Criteria 3.1 and 3.2 ........ 198
15. User 3 / Prioritization of Evaluation Criteria at Level 2 for Criteria 4.1 To 4.4 ........ 199
17. The Reduced Set of Criteria .......................... 201
19. User 1 / Evaluation of Alternatives According to Criterion Air-Crafts (AHP) ...... 203
20. User 1 / Evaluation of Alternatives According to Criteria Guns (AHP) .... 204
21. User 2 / Evaluation of Alternatives According to Criterion Sonar (DIRECT) .. 205
22. User 1 / Evaluation of Alternatives According to Criterion Maintenance (DIRECT) .......... 206
23. Final Group Solution of the Problem ............ 207
24. Group Norm Definition ............... 208
25. Group Problem Definition .......... 209
26. User 1 / Problem Initiation .............. 210
27. User 1 / Prioritization of Evaluation Criteria AHP. .............. 211
28. User 1 / Final Weights of Evaluation Criteria ............... 212
29. User 1 / The Reduced Set of Criteria .......... 213
30. User 1 / Individual Evaluation of Alternatives Using Direct mode .......... 214
31. Solution of User 1 (With Direct Mode) ............ 215
32. User 1 / Evaluation of Alternatives Using Electre .............. 216
33. User 1 / Concordance, Discordance, Outranking Matrix .......... 217
34. User 2 / Individual Evaluation of Alternatives Using Direct Mode .......... 218
35. Solution of User 2 (With Direct Mode) ............ 219
36. User 2 / Evaluation of Alternatives Using Electre .............. 220
37. User 2 / Concordance, Discordance, Outranking Matrix .......... 221
39. Solution of User 3 (With Direct Mode) ............ 223
40. User 3 / Evaluation of Alternatives Using Electre .............. 224
41. User 3 / Concordance, Discordance, Outranking Matrix .......... 225
42. Group Solution of the Problem (Solved with Direct Mode) .............. 226
43. Group Solution of the Problem (Solved with Electre Mode) .............. 227
ACKNOWLEDGEMENTS

At the completion of this research the author wishes to express his gratitude as well as his personal sincere appreciation to professors T.X. Bui and T.R. Sivasankan for their assistance.

Finally, the author dedicates this thesis to his wife Elpida, who has always encouraged and helped him during his efforts for education and continuous self-improvement.
I. INTRODUCTION

A. DEFINITION OF THE PROBLEM

It is often observed that most of strategic problems are analyzed, discussed, and solved by many decision makers. The existence of multiple users have created a number of problems. First, it is difficult to physically reunite all decision makers in a geographic location. It is even more problematic in finding an appropriate time for all the group members. Second, the success of a group decision making process relies on the skillfulness of the group leader. Unfortunately, the quality of the group leader varies from one negotiator to the other, and from one situation to the other. In a military decision-making context, this problem becomes even more complicated if one considers the increasing complexity of the technological aspect of warfare and uncertainty regarding political issues.

This research proposes a computer-based group decision support system that attempts to resolve, or at least reduce, the problems enumerated above. It designs and implements a microcomputer-based DSS that allows group members to remotely and sequentially participate to collective decision problems. In particular, the proposed DSS is an expansion of a DSS based on multiobjective decision methods, is implemented in a local area network using a bus architecture and the Carrier Sense Multiple Access with Collision Detection (CMSA/CD) protocol. The CMSA/CD protocol is known by its relatively good performance, simplicity of implementation, and inherent system reliability. Such a protocol allows control of collective information exchange and data routing among group decision members.

The use of such a group DSS distributed in time and in space, is expected to eliminate the physical presence of
group members and the need of scheduling meetings. More important, the proposed distributed DSS provides a objective and flexible framework to integrate organizational norms and constraint into the decision situation.

B. SCOPE OF THE RESEARCH

This research does not attempt to discuss the already large and interdisciplinary literature on group decision making. It attempts to expand some of the work in group decision support systems outlined by [Ref. 1 to 3] Two major expansions include the possibility for the user (i) to directly assess his preferences in cardinal terms, and (ii) to allow division of evaluation tasks according to individual expertise. In particular, this research primarily focuses on the software design and implementation of the networked micro-computer-based group DSS operating under a cooperative environment. However, the modular approach adopted for the proposed DSS would make it possible to expand the system to more complex form of group decision situations found, for example, in military strategic planning.

C. ORGANIZATION OF THE THESIS

Section II outlines basic definitions, concepts and architectures related to group decision making under computer-based settings. It emphasizes the communications aspects among group members via computerized media. Chapters III, IV, V and VI successively discusses the characteristics of the components of the group DSS. Two multiple criteria decision methods are presented in III.A. Four techniques of aggregation of preferences are defined in III.B. The multi-window interface has been adopted for the GDSS interface (section IV.A). Data definitions and dictionaries are described in section V. Section VI addresses special
applications of the communications modules. Some observations on the development process of the GDSS are reviewed in section VII. Two examples of remote multiperson decision-making in military strategic planning are analyzed in section VIII. They illustrate the use of the GDSS to the selection problem of navy ships.
II. A FRAMEWORK FOR IMPLEMENTING REMOTE MULTIPERSON GROUP DECISION SUPPORT SYSTEMS

A. DEFINITIONS AND BASIC CONCEPTS

1. Definitions of group DSS

A collective decision-making process can be viewed as a decision situation in which (i) there are two or more persons, each of them characterized by his or her own perceptions, attitudes, motivations, and personalities, (ii) who recognize the existence of a common problem, and (iii) attempt to reach a collective decision [Ref. 1]. Furthermore, the group can interact simultaneously (i.e., pooled-interdependent mode) or make individual decisions separately and then confront and discuss the results (i.e., sequential-interdependent).

One can observe three broad types of group decision making: a single decision maker acting in a collective decision environment, non-cooperative decision making, and cooperative decision making.

In the group decision-making situation with one person, a particular decision maker ultimately makes the decision and assumes responsibility for his line of action. However, the decision can be regarded as a collective one because of the existence of a dense network of influences that surrounds this single decision maker. In fact, other participants in the decision maker's organization can either support or act against the decision. Thus, the behavior and attitudes of other people who are indirectly involved in the decision-making process should be analyzed.

In the non-cooperative decision situation, the decision makers play the role of antagonists or disputants.
Conflict and competition are common forms of non-cooperative decision-making. While the former represents a situation in which disputants seek to hurt their opponents to pursue their own interests, the latter is characterized by the facts that each competitor is an action candidate, and is trying to outperform others.

In a cooperative environment, the decision makers attempt to reach a common decision in a friendly and trusting manner, and share the responsibility. Consensus, negotiation, voting schemes, and even the recourse to a third party to dissolve differences are examples of this type of group decision making.

Also, the literature in decision-making describes two types of decision situations involving more than one user: pooled interdependent and sequential interdependent. In a pooled decision-making situation, decision makers reunite together to form a more or less homogeneous group, and attempt to resolve a collective problem simultaneously. Elsewhere, in a sequential interdependent situation, members of the group can attack the collective problem at different periods in time, looking at different decision angles.

Another classification of group problem solving approach found in the literature is the distinction between content-oriented and process-oriented approaches. The first approach focuses on the content of the problem, attempting to find an optimal or satisfactory solution given certain social or group constraints, or objectives. By contrast, the second approach is based on the observation that the group goes through certain phases in the group decision-making process, and on the belief that there could be an arranged way to effectively deal with these phases.

When a collective decision fails, it becomes necessary for the participants in the group problem solving to start bargaining or negotiating until a consensus is
found. While bargaining involves discussion within a specific criterion or issues, negotiation includes many criteria or issues in the discussion and search for consensus.

2. Assumptions

Without loss of generality, the cooperative multiple criteria group decision support system implemented in this thesis, is a DSS that (i) contains MCDM and supporting models in the individual Model component, and (ii) is able to support multiple decision makers via a Group DSS to reach a consensus in a cooperative environment.

Under certain decision circumstances, MCDM can play a crucial role in supporting group decision-making:

(1) Due to interpersonal differences, the existence of multiple and conflicting objectives is substantially more dominant in group decision-making than in single person decision-making;

(2) Subjective and qualitative assessments seem to play a more crucial role in group than in single user decision-making. It has been observed that it is relatively easy for decision makers to agree upon problems that have objective, quantifiable and well-defined attributes. Conversely, decision makers tend to disagree upon attributes that require subjective and qualitative assessments. Furthermore, in group decision-making, in addition to the evaluation of the situational problem, decision makers invariably attempt to evaluate and the decision analyses of themselves and others.

(3) The simplicity of MCDM outputs makes it easier to communicate, coordinate and aggregate individual analyses in the group decision-making process.

(4) The process often plays a more decisive role than the content in group problem solving. MCDM provide a simple but structured framework for controlling the decision-making process, i.e., assessment of alternatives, assessment of evaluation criteria, selection of an appropriate algorithm for assessment of preferences, and search for a solution or compromise;

(5) The division of decision processes into four stages also allows alternate utilization of both objective optimization and subjective evaluation.
(6) The iterative use of the MCDM processes would permit integration of predecision and postdecision phases in the habitual decision phase.

Specifically, the Co-op DSS discussed in this research attempts to support the following decision situation:

(1) There are multiple users or decision makers. They may share an equal weight or have an unequal or 'hierarchically' distributed weight corresponding to a particular decision-making context.

(2) The group shares a common set of feasible decision alternatives. From this set of alternatives, the decision makers can either select one or more alternatives, or rank them according to a given set of criteria.

(3) Each decision maker may have personal objectives that reflect a priori values and aspirations levels. Objectives are concretely expressed by criteria or attributes that are discrete, and at least ordinally measurable. Due to personal differences, individual decision outcomes— as opposed to a collective decision outcome that the group is trying to reach an agreement on— often differ from one decision maker to the other.

(4) The decision makers can be geographically dispersed and not required to log into the system at the same time. Via a distributed computer network system, they can communicate to others either sequentially or in an on-line mode.

(5) The decision makers interact in a cooperative manner and in a trusting environment. The system does not handle attempts to cheat or to seek coalition within sub-groups.

(6) The decision makers can either work closely together by forming a homogeneous group that uses a single decision support system, or work independently and then proceed to a multilateral assessment of the problem.

(7) The decision makers can segment a group decision problem into (hierarchically) sequential single user decision problems according to individual expertise and responsibility.
3. **Communications Issues in Distributed Decision Making**

In the context of a distributed group decision making, the demands for information exchange are marked by certain characteristics that should be considered in the design of communications capabilities. These characteristics could be best expressed by the requirements of having information exchanges that are (i) format-transparent, (ii) either constrained or unconstrained, and (iii) evolving throughout the decision phases.

a. **Need for Format-Transparent Information Exchange**

The demand for and/or generation of information among decision makers can take a variety of formats, ranging from unstructured and written notes to structured and numerical tables [Ref. 4]. The most complex form of traffic is the situation in which decision makers simultaneously require information exchanges on different subjects from different members using complicated combinations of input/output formats. It would then be necessary to identify, classify and convert information characterized by various individual formats into standard message formats, including the creation and maintenance of information related to group problem solving techniques, such as aggregation of preferences which requires some standardized inputs from individual results.

b. **Limited versus Free Information Exchange**

In some group decision situations, it is conceivable that all shared information is 'public' in that every member of the decision group has the right to access any information that is sent by one member of the group to another, whereas in some other decision situations, individual-to-individual or private message transfers are authorized [Ref. 5]. Thus, the creation, (statistical) maintenance and storage of message routing activities remains crucial in enforcing group norms concerning the type
of information sharing (e.g., consensually predefined by the group prior to the group decision-making process, or monitored by the mediator.

c. Evolving Pattern of Communication Requirements

The requirements for information sharing evolves through various phases of the group decision-making process. For example, [Ref. 6] argues that a group problem solving phase that emphasizes search and innovation requires more spontaneity, and therefore an open communications pattern; whereas, bargaining activities that induce a preference for deliberate control of information exchange would be facilitated by using individual-to-individual communication channels.

Furthermore, empirical studies have shown that, under certain circumstances, communication channels can escalate conflict [Ref. 7]. While encouraging information exchange between group members is often recognized as an effective strategy to resolve individual differences, eliminating communication channels has shown its effectiveness in preventing deterioration of relationships. While the decision to encourage or discourage communication between decision makers depends on a number of unpredictable situation-dependent factors, the GDSS communications component should be designed in such a way that it can accommodate various communications needs and changes during the group decision-making process. In other words, the pattern of communications protocols should vary according to the dynamics of the group decision-making process.

4. The Role of the Communications Component

One of the roles of the communications component that emerges from the literature is that it makes it easier for each member of the group to electronically communicate without having to be concerned about detailed and complicated protocol procedures. This issue of user
transparency is particularly crucial given the diversity, and consequently the complexity, of the communication requirements and facilities.

However, the effort to obtain ease of communication access is not unique to the design of group DSS. Rather, it has always been one of the most important objectives of computer networks design. Yet, one can identify at least three roles that are specific to a communications system in group problem solving. At different phases of the distributed decision process, the communications system can play the role of a coordinator, a detective, or an inventor.

a. The Coordinator Role

Most problem solving activity begins with situation analysis and problem definition. Situation analysis is characterized by a (common) recognition that there exists an urgent and important problem to be solved. Once identified in the situation analysis, a problem is transformed in the problem definition phase in such a way that solutions can be generated, analyzed and selected. [Ref. 8] and [Ref. 9] emphasize that the success of information gathering and problem definition relies on the ability of the group to eliminate mistrust and threat that could cause group participants to withhold or distort information. Walton [Ref. 6] suggests that by installing a communication medium that follows some norms of fairness (e.g., equality of participation, preserving autonomy), information exchange can be more abundant and accurate. The communication component should thus coordinate various protocols to engender participants' confidence. Such protocols could include the ones that (i) assure each member can successively broadcast his/her ideas given a equal amount of time, or (ii) support teleconferencing to synchronize arguments.
b. The Detective Role:
A decision maker's analysis could be distorted by (i) the individual's attempt to 'spy' on others' activities, or (ii) the influence of some members who try to take over an individual's responsibility. The communications component should then play the role of detective to prevent unwanted data exchange or temporarily disable all links, or prevent malicious modification of public data. Concurrently, decision makers tend to delay sending their individual results. The communications component should press its users to submit opinions before a given due date.

From a general perspective, the detective role consists of enforcing communications protocols previously defined to drive the collective decision-making process.

c. The Inventor Role:
The inventor role is an extension of the coordinator role. Given the complex nature of a collective decision problem and the diverse and unpredictable decision approaches adopted by the participants, the communications component should be able to detect incompatible information exchange, and, if possible, propose alternate formats. The inventor role implies (i) potential for tolerance to uncertainty in requests and needs for data transfers, and (ii) continued search for communications operations that facilitate information exchange [Ref. 10]. Thus protocols for distributed GDSS should be able to analyze, evaluate and determine the content of transmissible information, rather than simply perform a transport task.

The functions of the communications component are at least twofold. First, it monitors a broad spectrum of data transports during a group problem solving process. This transport function ranges from information exchange to information hiding, from selective and personalized routing to collective diffusion of data from public to private...
information. Second, it coordinates various communications activities (i.e., initialization, operation during consensus search, negotiation and mediation) by making it transparent to the members of the decision group.

B. AN ARCHITECTURE FOR GROUP DSS

Co-oP is a network of microcomputer-based process-driven DSS for cooperative multiple criteria group decision making (Figure 1). Each participant of the group decision making process has his own individual DSS whose model base is based on multiple criteria decision methods (MCDM) and other personal decision support tools. The group DSS contains a set of aggregation of preferences techniques and consensus seeking algorithms that can be used in conjunction with individual MCDM.

The individual DSS are linked together by a microcomputer local area network. The latter support both locally and remotely (via modem) linked individual workstations.
MAIN MENU

1. MULTIPLE CRITERIA GROUP PROBLEM DEFINITION
2. GROUP NORM DEFINITION
3. PRIORITIZATION OF EVALUATION CRITERIA
4. INDIVIDUAL EVALUATION OF ALTERNATIVES
5. DIRECT INPUT OF THE DATA
6. COMPUTATION OF GROUP DECISION
7. IDENTIFICATION OF NEGOTIABLE ALTERNATIVES

Enter a number:

MAIN MENU
For HELP enter (ALT) R / (ESC) to quit Help

Figure 1. The Main Menu
III. THE MODEL COMPONENT

The Model Component of a DSS is expected to support the user perform the following problem-solving activities: projection, deduction, analysis, creation of alternatives, comparison of alternatives, optimization and simulation [Ref. 11]. The literature in DSS often identifies three modules in a DSS model component: the model base, the model base management, and the interface unit. This chapter describes the three components of the group DSS.

A. THE MODEL BASE

The Model Base of a DSS consists of a library of decision models that help the group members perform individual and group analyses.

1. The Model Base for Individual Decision Making

In addition of the possibility for the user to directly enter his preferences/assessments to the system and if needed, share them to other group members, the purpose of the Co-oP MCDM model base is to provide the decision makers with a set of decision models that can solve the most common types of decision problems. Co-oP contains two models that (i) cover three basic decision situations, i.e., selection, ranking, sorting, (ii) are not excessively difficult to use for the decision makers, and (iii) could interact with techniques of aggregation of preferences. The MCDM methods implemented in each of the individual DSS are the Analytic Hierarchy Process (AHP) [Ref. 12], and ELECTRE [Ref. 13].

ELECTRE and AHP have been selected for two reasons:
(1) The two MCDM are conceptually robust, and practically easy to learn and use. They have proven their usefulness in aiding a number of ill-defined
decision situations (for example, [Ref. 14] and [Ref. 15]).

Neither ELECTRE nor AHP require full information on the decision maker's preferences and assessment of alternatives, and hence, give more autonomy and control to the decision maker [Ref. 16]. This feature makes it easier to expand the algorithm to resolve group decision making. This section briefly outlines basic concepts of the ELECTRE and AHP methods.

a. The ELECTRE Method: Basic Concepts

There are a number of reasons that make it difficult for a decision maker to exhaustively compare all known alternatives. First, the decision maker often cannot compare some alternatives, due to uncertainty associated with the measurements and evaluation. Second, the decision maker may be unwilling to compare two alternatives because they are incomparable; e.g., option A is better than option B by some criteria, whereas B is better than A by some other criteria. The notion of indifference in utility theory does not reflect this incomparability [Ref. 17]. Last but not least, the ill-structuredness and occasional inconsistency of the decision maker's preferences are serious obstacles to enforcing the complete comparability of alternatives (see [Ref. 12]).

The concept of outranking relations seeks to compare decision alternatives only when the decision maker's preferences are well defined. In other words, a_i outranks a_j when the information obtained from the decision maker's preferences safely justifies the proposition that a_i is at least as good as a_j.

The outranking relation can be explained by two further concepts: the presence of concordance (i.e., for a sufficiently important subset of evaluation criteria, A is at least weakly preferred to B); and the absence of discordance (i.e., among the criteria for which B is
preferred to A, there is no significant discordant preference that would strongly oppose any form of preference of A over B).

These indexes are used in conjunction with concordance and discordance 'thresholds' chosen arbitrarily by the decision maker in the interval [0,1]. The concordance threshold, p, is more severe as it approaches 1; the discordance threshold, q, is more severe as it approaches 0. Then, the outranking relations can be summarized as follows:

<table>
<thead>
<tr>
<th>IF</th>
<th>THEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_A/B &gt; p and D_A/B ≤ q</td>
<td>A outranks B</td>
</tr>
<tr>
<td>A outranks B, and B outranks A</td>
<td>The alternatives are equivalent</td>
</tr>
<tr>
<td>Otherwise</td>
<td>The alternatives are incomparable</td>
</tr>
</tbody>
</table>

The decision maker can start with a less severe set of threshold values, and then sharpen them to reduce the number of outranking relations.

b. The Analytic Hierarchy Process: Basic Concepts

The Analytic Hierarchy Process (AHP) is a MCDM method that attempts to support complex decision problems by successively decomposing and synthesizing various elements of a decision situation [Ref. 12]. Like ELECTRE, AHP permits subjective and qualitative pairwise comparison of alternatives. Unlike ELECTRE whose concept is based on the notion of non-dominated alternatives, AHP has its foundation on the concept of priority. The latter can be defined as a 'level of strengths' of one alternative relative to another. Departing from a predefined priority scale, the decision maker is asked to build a positive reciprocal matrix of
pairwise comparison. A vector of priority can be derived by computing the eigenvector of the reciprocal matrix. The property of the eigenvector resides in the fact that it is a consistency indicator. Consistency is obtained when pairwise comparisons are transitively and proportionally consistent.

Additional algorithms are added to help measure the decision maker's consistency. These algorithms contrast the user's evaluation scores with (i) a randomly simulated score that represents the most irrational evaluation, and (ii) the eigenvalue that represents the most accurate consistency. The examination of the consistency values enables the user to eventually revise initial judgments, and, if appropriate, modify them to improve overall consistency.

2. The Model Base for Group Decision Making

Four techniques of aggregation of preferences are implemented in the GDSS. They are chosen because of their popularity. These include the additive function, the multiplicative function, the sums-of-the-ranks approach, and the sums-of-the-outranking-relations approach.

In conjunction with the techniques of aggregation of preferences, the weighed majority rule is also implemented to account for the distribution of decision power among decision makers. This rule allows the group members to differentiate their decisional power according to various degrees of expertise or organizational hierarchies.

(1) The Sums-of-the-Outranking-Relations Principle

This technique is derived from the sum-of-the-ranks technique found in the literature of aggregation of preferences. Formally, it can be expressed as follows:

$$\text{Max } \sum_{i=1}^{m} \sum_{k=1}^{m} o_{ik}$$

This technique should be used only with extreme care. Experience with this technique has shown that the idea of selecting the alternative that has the highest number of outranking relations works fine only when the number of alternatives are small. An
example with three decision makers and three alternatives, with $a_3$ as the elected alternative, is given below.

---

**Ordinal Ranking**

<table>
<thead>
<tr>
<th>Rank</th>
<th>DM_1</th>
<th>DM_2</th>
<th>DM_3</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>Sums of the Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$a_1$</td>
<td>$a_3$</td>
<td>$a_3$</td>
<td>$a_1$</td>
<td>-</td>
<td>2</td>
<td>1 3</td>
</tr>
<tr>
<td>2</td>
<td>$a_2$</td>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$a_2$</td>
<td>1</td>
<td>-</td>
<td>1 2</td>
</tr>
<tr>
<td>3</td>
<td>$a_3$</td>
<td>$a_2$</td>
<td>$a_1$</td>
<td>$a_3$</td>
<td>2</td>
<td>2</td>
<td>- 4 (&lt;-Max)</td>
</tr>
</tbody>
</table>

**Sums-of-the-Ranks Rule**

The sums-of-the-Ranks rule (Borda, 1781) can be defined as follows:

where $r_{d, ai}$ is the rank assigned by decision maker $d$ to alternative $a_i$. The example below illustrates this rule.

<table>
<thead>
<tr>
<th>Altern.</th>
<th>DM_1</th>
<th>DM_2</th>
<th>DM_3</th>
<th>Sums-of-the-Ranks</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>$a_2$</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>5 (&lt;- Min)</td>
</tr>
<tr>
<td>$a_3$</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>$a_4$</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

Due to its computational simplicity this technique is widely used to determine consensus ranking. Note that the averages-of-the-ranks rule yields the same results. However, when there are ties, the results are different.

**Additive Ranking**

In the additive ranking method, group results are obtained by computing the arithmetic mean of the individual rankings assigned to each alternative.
Due to its simplicity, this method remains one of the most popular aggregation of preferences techniques. The example below illustrates this rule.

<table>
<thead>
<tr>
<th>Altern.</th>
<th>DM₁</th>
<th>DM₂</th>
<th>DM₃</th>
<th>Additive Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>3.33</td>
</tr>
<tr>
<td>a₂</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1.66</td>
</tr>
<tr>
<td>a₃</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2.66 -- MAX</td>
</tr>
<tr>
<td>a₄</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2.33</td>
</tr>
</tbody>
</table>

(4) **Multiplicative Ranking**

The philosophy that underlies the multiplicative approach is to allow more voting power to each decision maker of the group. In effect, the multiplication of individual cardinal rankings amplifies the individual opinions. Specifically, it allows vetoes to take place. The example below illustrates this rule.

<table>
<thead>
<tr>
<th>Altern.</th>
<th>DM₁</th>
<th>DM₂</th>
<th>DM₃</th>
<th>Multiplicative Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>3.17480</td>
</tr>
<tr>
<td>a₂</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1.44224</td>
</tr>
<tr>
<td>a₃</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2.51984</td>
</tr>
<tr>
<td>a₄</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2.08008</td>
</tr>
</tbody>
</table>

B. THE MODEL MANAGER

The role of the model manager is to coordinate various modelling activities of the GDSS. In Co-op, the multiple criteria group decision making is decomposed into five decision processes (see Figure 2).

(1) **Definition of the Group Problem**

The group must agree upon a common problem and delegate a group member -- usually the group leader or the secretary -- to define a problem. In the Co-op context, the defined group problem consists of identifying the alternatives and evaluation criteria. Section VIII provides an example of this process.
(2) Group Norm Definition

The group has to identify its members and assign individual passwords. It also has to agree upon the way it handles data transfers, interactive conversation, utilization of electronic mail, and the type(s) of techniques of aggregation of preferences adopted. Division of evaluation tasks between group members can also be specified. The group can also request automatic selection and computation of appropriate decision technique.

(3) Individual Evaluation of Criteria

This process requires that each group member prioritize his/her evaluation criteria. This can be either accomplished by asking each decision maker to directly assign weights to the criteria or use the Analytic Hierarchy Process scheme to generate the weighed or priority vector. Co-op allows elimination of weak criteria.

(4) Individual Assessment of Alternatives

Given a chosen problem, this process allows the group members to individually express their preferences regarding the alternatives. This process can be either direct (i.e., the user enters cardinal weights to each alternative) or indirect (i.e., the group member uses one or two available MCDM techniques).

(5) Computation of Group Results

Guided by the instructions defined in the group norm (i.e., the second process), group results are automatically computed once all individual analyses are submitted.

1. Integration of Models

Unless otherwise specified by the group norm, the Co-op group module automatically searches for all aggregation techniques that are compatible with the individual MCDM used. If direct assessment of alternatives or AHP has been adopted by every group member for individual assessment of alternatives, all of the four implemented techniques will be computed, since the latter are compatible with the AHP in that they are based on cardinal preferences. However, the ELECTRE method can work only with the sums-of-the outranking-relations and, to a certain degree, the sums-of-the-ranks algorithms.
When both available MCDM are used concurrently by a group member, the Co-op model manager automatically searches for group decision techniques that can accept inputs from both AHP and ELECTRE. When a single user alternately uses both available MCDM, the Co-op model manager sequentially displays group results according to all possible combinations of individual methods.

Such a sensitivity analysis constitutes a point of departure for the group to start exchanging points of view and directions to reach agreement, and, if any, reducing tension. The group can then temporarily exit from ELECTRE, and use the electronic notepad to informally resolve these problems of control and of tension management. If some concessions can be obtained, the participants can return to ELECTRE and modify evaluation scores accordingly. By switching back and forth between the individual DSS and the group DSS, the participants can perform 'sequential concessions'.

2. Combined Use of MCDM and Techniques of Aggregation of Preferences

Bui [Ref. 2] argues for a unified MCDM framework. Such an attempt is necessary to (i) support a wide range of decision situations, (ii) enable economy of information search, (iii) allow division of evaluation tasks. In the Co-op version implemented for this thesis, there are three possible levels of interaction between ELECTRE and AHP. First, ELECTRE, when used alone, assumes that the decision has a defined vector of criterion weights. AHP can help the ELECTRE user perform prioritization of evaluation criteria prior to the pairwise evaluation of alternatives. Second, when the size of a decision problem is large, the number of inputs required to perform the AHP method can be excessive. The Co-op user can use ELECTRE as a sorting tool to reduce the problem size, and then utilize AHP. Third, since the two methods refer to the same decision space (defined in the
Co-oP first process), they can be concurrently used to verify the decision maker’s consistency.

C. THE LINKAGE MODULE

The purpose of the Co-oP Linkage Module is to feed input data to various models of the Model Base and to route output data to various files managed by the Data Base Component.
Figure 2. The Co_op Decision Making Process
IV. THE INTERFACE COMPONENT

A. SCREEN DESIGN

Despite the structured aspect of the multiple criteria group problem solving processes, it remains an eventual burden for the decision makers to memorize what he has done in the previous steps. Maintaining a high degree of coordination and cohesiveness of thoughts is particularly prevalent in complex decision problems [Ref. 17].

Screen Format: During the problem definition and the group norm definition processes, data entry in outline form is adopted. Such an entry form would not only facilitate the thinking process of the managers, but also help decompose objectives into hierarchical levels [Ref. 12]. Section VIII exhibits examples of the outline forms used for defining the collective decision problem and the definition of group norms.

For the multiple criteria group decision processes (i.e., processes 3 through 7), Co-oP proposes a screen format that displays simultaneously four different windows (see Section VIII). Whenever possible, Co-oP uses the same screen format throughout its usage. The purpose of such a design is to provide the user with a synoptic and familiar snapshot of the current state of the problem, throughout the entire decision-making process.

The Step Window located at the bottom screen keeps the decision maker up to date on the current decision making status. It consists of a two-line status text indicating alternatively the current step in the hierarchy of group problem processes, and any required prompts or diagnostic messages related to the DSS-user interaction.
The Dialogue Window provides a conversational medium between the decision maker and the DSS. It enables the Question/Answer mode of interaction to be accompanied by verbal and color/graphic explanation of various processing sequences and intermediate results.

To support the decision maker's orientation during the group decision-making process, the Working window at the upper left corner of the screen reminds the user of vital information from past dialogue or inputs. Also, it displays the results obtained by other participants if requested.

The Solution window is located at the upper right of the screen. It displays intermediate and final results including statistical indexes, and highlights optimal values. Tabular outputs and bar graphs are combined to provide alternate ways to represent outputs.

Throughout the entire Co-oP process, the windows can be recognized by their colors. However, they vary in size according to the required amount of information displayed (e.g., number of decision makers, number of decision alternatives, and number of evaluation criteria).

In addition to the above mentioned window, an electronic notepad window can be invoked at any time to make use of person-oriented and unstructured communications.

B. DIALOGUE STYLE

In addition to the window structure that governs the entire Co-oP group decision making process, Co-oP combines menus and questions to communicate with its users. The purpose of these dialogue styles is to provide the users with a structured, simple and controlled framework to interact with an integrated set of multiple criteria group decision methods. Whenever possible, concise queries and uniform terminology are used throughout the six processes of the Co-oP group decision making process.
The use of menus and queries also facilitates establishing error procedures. Although error control procedures are not unique to the design of multiple user interface, an eventual I/O error occurring in a group DSS can generate unexpected and severe consequences in a distributed DSS. Input control routines have been implemented at each entry level to minimize the likelihood of input errors, or to maximize the possibility of recovering from errors when the latter occur.

To handle errors made by the users, Co-oP provides two types of error control procedures. The first type of procedure detects syntax errors. For instance, entering a negative number of decision makers or typing an invalid filename would be gracefully rejected by the Co-oP dialogue manager. The second type of control routines attempts to prevent decision makers from violating basic assumptions or rules of the decision methods. For instance, the dialogue manager will refuse a concordance threshold higher than 100 percent when ELECTRE is used.

Co-oP also generates short explanation messages in the Step window to maintain the user confidence in the system, or at least make the multiple criteria group decision making less unnatural to the users.

C. THE HELP COMMANDS

Help facilities are implemented on a separate and resident program that can be concurrently invoked during the Co-oP decision-making process. Due to its relatively large amount of text, the help program is hierarchically broken down into eight section (see Figure 3).
HELP FOR Co_oP

MAIN MENU

(1) General Information
(2) Create a new Problem
(3) Prioritization of Evaluation Criteria
(5) Evaluation of Alternatives
(6) Direct Input of Weights
(7) Computation of group Decision
(8) N A I

SELECTION : _

HELP FOR Co_oP

SUBMENU 1 - GENERAL INFORMATION

(1) What Co_oP IS
(2) How to Use It
(3) AHP Method
(5) Electre Method

SELECTION : _

Figure 3. The Help Menu
V. THE DATA COMPONENT

A. THE DATA STRUCTURE

The current version of Co-op is a process-centered group DSS, as opposed to a data-centered DSS (for instance, see [Ref. 18]). As a consequence, the structure of the Co-op data component is minimal. Its objective is to (i) insure smooth and fast data transport from one MCDM step to the other, and (ii) facilitate data exchange between decision makers.

Data files are grouped according to each process. These include (i) a file containing the problem definition (Process 1), (ii) a norm file for each group norm, (iii) a solution file for each group members, and (iv) a group results file for each decision problem. Data dictionaries are given in Tables 1, 2, 3 and 4.

To minimize the time needed for data transfers between individual workstations, data files are physically centralized and stored in the server of the Local Area Network. However, they are functionally distributed in that they can be accessed only by authorized group members.

B. THE DATA MANAGER

In the current version of the GDSS, the Data Manager performs a double functions. It (i) assures that data are correctly transferred to their location, and (ii) checks the consistency transfer, i.e., validating the number of data modification.
<table>
<thead>
<tr>
<th>PROBLEM = RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>name1</strong></td>
</tr>
<tr>
<td><strong>levels</strong></td>
</tr>
<tr>
<td><strong>numofalternatives</strong></td>
</tr>
<tr>
<td><strong>level1</strong></td>
</tr>
<tr>
<td><strong>level2</strong></td>
</tr>
<tr>
<td><strong>level3</strong></td>
</tr>
<tr>
<td><strong>level4</strong></td>
</tr>
<tr>
<td><strong>level5</strong></td>
</tr>
<tr>
<td><strong>level6</strong></td>
</tr>
<tr>
<td><strong>level7</strong></td>
</tr>
<tr>
<td><strong>level11</strong></td>
</tr>
<tr>
<td><strong>sublevel1</strong></td>
</tr>
<tr>
<td><strong>sublevel2</strong></td>
</tr>
<tr>
<td><strong>alternatives</strong></td>
</tr>
</tbody>
</table>

END
<table>
<thead>
<tr>
<th><strong>TABLE 2</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGICAL DATA BASE RECORDS FOR STORE THE DATA OF A PROBLEM</td>
</tr>
</tbody>
</table>

**SOLUTION1 = RECORD**

- **pfactor, qfactor** integers, they hold the Concordance and Discordance Threshold
- **numofcriteria** integer, it holds the number of criteria
- **numofalternatives** integer, it holds the number of the alternatives
- **alternatives** array[1..9] of string, it holds the name of the alternatives
- **numofusers** integer, it indicates the number of the users
- **solved** array[1..3] of boolean, it indicates if a particular user has solve the problem
- **grading** array[1..3] of array[1..3], it contains weights of criteria 1 - 5 for each user
- **completed** boolean, it indicates if the evaluation of the criteria is completed of all the user
- **completedall** boolean, it indicates if the problem is solved
- **vector1** array[1..5] of reals, it contains the weights of the criteria of sublevel 1 - 5
- **vector2, vector3, vector4, vector5, vector6, vector7** array [1..5, 1.5] of reals, it holds the weights of all the rest criteria
- **normvector1** array [1..125] of strings, it holds the names of the final criteria (after the evaluation)
- **normvector2** array [1..125] of reals, it holds the weights of the final criteria (after the evaluation)
- **normindex** array[1..vectorg]
- **altmatrix** altrim[1..# alternatives, 1..# criteria] array[1..3] of boolean, it indicates if a specific user has compute the evaluation of the alternatives
| Table 2 (continued) |

| ahp : record |
| status       boolean, it indicates if the solution of a problem has been computed with the AHP |
| altvector1   array[1..9] of real, it contains the final weights of the alternatives |
| numoftries   integer, it indicates how many times the user has modify the solution of the problem |

| electre : record |
| status        boolean, it indicates if the solution of a problem has been computed with the ELECTRE |
| outranking    array[1..9, 1..9] of char, it contains the outranking matrix for the alternatives |
| numoftries    integer, it indicates how many times the user has modify the solution of the problem |

END ;
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usersnames array [1..3] of strings</td>
<td>it holds the names of the users</td>
</tr>
<tr>
<td>usersids array [1..3] of strings</td>
<td>it holds the users id</td>
</tr>
<tr>
<td>numofcriteria integer</td>
<td>it holds the number of criteria</td>
</tr>
<tr>
<td>numofalternatives integer</td>
<td>it holds the number of the alternatives</td>
</tr>
<tr>
<td>alternatives array[1..9] of strings</td>
<td>it holds the name of the alternatives</td>
</tr>
<tr>
<td>normvector1 array [1..125] of strings</td>
<td>it holds the names of the final criteria (after the evaluation)</td>
</tr>
<tr>
<td>normvector2 array [1..125] of reals</td>
<td>it holds the weights of the final criteria (after the evaluation)</td>
</tr>
<tr>
<td>ahp : record status boolean</td>
<td>it indicates if the solution of a problem has been computed with the AHP</td>
</tr>
<tr>
<td>altvector1 array[1..9] of real</td>
<td>it contains the final weights of the alternatives</td>
</tr>
<tr>
<td>numoftries integer</td>
<td>it indicates how many times the user has modify the solution of the problem</td>
</tr>
<tr>
<td>electre : record status boolean</td>
<td>it indicates if the solution of a problem has been computed with the ELECTRE</td>
</tr>
<tr>
<td>outranking array[1..9,1..9] of char</td>
<td>it contains the outranking matrix for the alternatives</td>
</tr>
<tr>
<td>numoftries integer</td>
<td>it indicates how many times the user has modify the solution of the problem</td>
</tr>
</tbody>
</table>

END ;
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>numofusers</td>
<td>integer, it holds the number of the users that are going to solve the problem</td>
</tr>
<tr>
<td>modifytimes</td>
<td>integer, it indicates how many times a user can modify the solution of the problem</td>
</tr>
<tr>
<td>lasttime</td>
<td>integer, it indicates the last date that a user must submit his solution</td>
</tr>
<tr>
<td>usernames</td>
<td>array [1..3] of strings, it holds the names of the users</td>
</tr>
<tr>
<td>specindex</td>
<td>array[1..5] of strings, it indicates the criteria that each user is going to solve in the division of tasks case</td>
</tr>
<tr>
<td>usersids</td>
<td>array [1..3] of strings, it holds the users id</td>
</tr>
<tr>
<td>weight</td>
<td>array[1..3] of real, it indicates the weight of the decision of each user</td>
</tr>
<tr>
<td>agregation</td>
<td>boolean, it indicates if we are going to use all the techniques of aggregation of preference</td>
</tr>
<tr>
<td>nai</td>
<td>boolean, it indicates if the program will use NAI automatically after the complication of the group result</td>
</tr>
<tr>
<td>specialized</td>
<td>boolean, it indicates if we are going to use division of tasks or not</td>
</tr>
<tr>
<td>broadcasting</td>
<td>boolean, it indicates if the users have the right to see the others users results</td>
</tr>
<tr>
<td>modify</td>
<td>boolean, it indicates if the user has the right to modify the solution of the problem</td>
</tr>
<tr>
<td>agregationname</td>
<td>array[1..4] of characters, it indicates the techniques of aggregation of preference that we are going to use</td>
</tr>
</tbody>
</table>
VI. THE COMMUNICATIONS COMPONENT

A. THE GROUP NORM CONSTRUCTOR

The Co-op Group Norm Constructor resides in the second Co-op multiple criteria decision making process. The group leader or secretary has to initiate the group decision making by (1) identifying the group members, (2) assigning respective decision weights, (3) determining the mode of group decision making (e.g., division of evaluation tasks or 'pooled' decision making), (4) selecting the techniques of aggregation of preferences, (5) setting the mode of information exchange (i.e., broadcast of individual results), and (6) defining the deadline for the group members to submit individual results.

B. THE GROUP NORM FILTER

The Co-op Group Norm Filter consists of a set of subroutines that enforce the norms set by the Group Norm Monitor.

C. THE FORMATTER

The main role of the Co-op formatter is to convert individual results computed by the ELECTRE and AHP methods to data formats that can be inputted into the modules containing the techniques of aggregation of preferences. For instance, individual cardinal rankings are converted into ordinal rankings for the sums-of-the-ranks algorithm.
VII. IMPLEMENTATION OF THE GDSS

A. SOFTWARE STRUCTURE

Turbo Pascal cannot handle program files whose size is larger than 62 kilobytes. To override such constraint, Co-op has been decomposed into 15 including files. The latter are described below. Also, filenames under IBM-PC-DOS cannot have more than eight letters, abbreviated filenames have been used.

DIRLIST1
PROCEDURE DIRLIST displays on the screen the existing files of previously defined problems (problem_name.def).

DIRLIST2
PROCEDURE DirListA The same as above but for the norms files (norm_name.gn).

PROCEDURE
FUNCTION STUPCASE turns a string to uppercase characters.
FUNCTION EXIST examines if the file requested by a user to access exists. If it exists it returns the value TRUE else returns the value FALSE.
PROCEDURE WAIT stops the execution of the program until the moment that the user will hit a key.
PROCEDURE CLEARSCREEN clears the screen for line 1 to line 10 to make space for new data.
PROCEDURE CONVERT converts a string to the corresponding numerical value.
PROCEDURE IDENTIFY reads the user input and accepts it only if it is Y or N.
PROCEDURE CHECKNUMBER reads a number that the user enters and accepts it only if it is within a predefined range.

PROCEDURE SORT1 sorts an array of numbers.

PROCEDURE WRITENORMFILE reads from the program the current norm data and writes them in a file (e.g., data of the current norm).

PROCEDURE WRITEPROBLEMFFile reads the norm data from the current norm file and passes them to the program.

PROCEDURE READPROBLEMFFile reads from the program the problem data and writes them in a file (e.g., data of the current problem).

PROCEDURE READNORMFILE reads the data for the corresponding norm file and passes them to the program.

PROCEDURE READSOLUTIONFILE reads the data from the user file and passes them to the program.

PROCEDURE WRITESOLUTIONFILE reads the current user data from the problem and writes them to the current user file.

FILES

PROCEDURE OPENFILE opens for the first time a file that it will keep the data of a new problem.

PROCEDURE OPENSOLUTIONFILE opens for the first time a file that it will keep the data of the solution of the problem (one for each user).

PROCEDURE OPENNORMFILE opens for the first time a file that it will keep the data of a new norm.

UTILITIES

PROCEDURE DISKDATA asks the user if he wants to see a predefined problem or norm.

PROCEDURE DISKSTATUS displays all the existing problems and norms of the current directory.

PROCEDURE READ1 asks the user the name of the problem that he wishes to solve.
PROCEDURE READ2 asks the user the name of the norm that he wants to use.
PROCEDURE READ3 asks the user's name.
PROCEDURE READ4 asks the user's password.
PROCEDURE READ5 asks the decision method that the user is going to use.
PROCEDURE DATA includes read1, read2, read3, read4.
PROCEDURE PRIORITYOFCRITERIA permits evaluation of evaluation criteria.

STEP 1
PROCEDURE CREATEPROBLEM reads the data of a new problem and writes them in a file.
PROCEDURE DISPLAY displays the data of a problem to the screen after the request of the user.
PROCEDURE CORRECTDATA corrects the data of the problem in case of an error occurs.

STEP 2
OVERLAY PROCEDURE NORMDEFINITION reads the data of a new norm and writes them in a file.

STEP 2-1
PROCEDURE NORMSELECTION asks the user to select one of the existing norms.
PROCEDURE DISPLAYNORM displays the data of a norm to the screen.

STEP 3
PRIORITYOFCRITERIA is the main program for the evaluation of the criteria.
STEP 3-1
OVERLAY PROCEDURE EVALUATE includes the evaluation of a set of criteria using AHP.
OVERLAY PROCEDURE DIRECT1 is similar to the previous procedure but using direct mode.

STEP3-2
PROCEDURE SELECTCRITERIA computes the final weights after the computation of all the sets of criteria.
PROCEDURE FINALCRITERIA gives the user the opportunity to reduce the number of the final criteria.

STEP4
PROCEDURE SOLVEWITHAHP controls the evaluation to the alternatives if the user select: AHP, direct mode, general direct mode, and displays the final weights for the alternatives.
PROCEDURE COMPUTEALTERNATIVES controls the computation of the alternatives according to the method that the user is going to use.

STEP4-1
OVERLAY PROCEDURE EVALUATE1 evaluates a set of alternatives using AHP.
OVERLAY PROCEDURE EVALUATE3, upon request, assigns weights in a set of alternatives directly (without grading previously the criteria).
OVERLAY PROCEDURE DIRECT2A evaluates a set of alternatives using the direct mode.

STEP4-2
OVERLAY PROCEDURE ELECTRE evaluates a set of alternatives using the ELECTRE method.

STEP6
OVERLAY PROCEDURE GDSS computes the group results.
B. EFFORT DISTRIBUTION AND MAINTENANCE PROBLEMS

1. Effort Distribution

The development of the software took approximately six man-months. The effort distribution is indicated below:

<table>
<thead>
<tr>
<th>TABLE 5 EFFORT DISTRIBUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement Analysis</td>
</tr>
<tr>
<td>Requirement Analysis</td>
</tr>
<tr>
<td>Initial Design</td>
</tr>
<tr>
<td>Detailed Design</td>
</tr>
<tr>
<td>Coding</td>
</tr>
<tr>
<td>Unit testing/Debugging</td>
</tr>
<tr>
<td>Testing Integration</td>
</tr>
<tr>
<td>% Of the Total time</td>
</tr>
</tbody>
</table>

It is worth noticing that the iterative design adopted for the development of Co-oP has helped in incrementing the functionalities of the software.

2. Implementation Problems and Maintenance Issues

(1) Design of Algorithms:
The understanding of algorithms, conversion of algorithm in structured pseudo-codes required elaborated design.

(2) Programming Language:
Mastering the language adopted for the software development has taken a substantial learning effort. Window scrolling, overlays, cursor handling--due to the limited capability of the programming language--took a non-negligible learning effort.

(3) Debugging logical errors:
Due to the complexity of the data structures, in particular, the manipulation of matrices in the AHP
techniques and the integration of multiple-user files, testing the correctness of data transfers represented an important part in the testing phase.
A. EXAMPLES OF POSSIBLE USE OF GDSS IN THE MILITARY CONTEXT

The proposed software is most appropriate for decision situations where there is distribution in space and in time. Such decision settings are often encountered at various high-level decision making in the armed forces as well as in the civil government. The example discussed below illustrates an decision example that deals with the selection of a naval warship.

B. A HYPOTHETICAL EXAMPLE

To exemplify the potential usefulness of the developed software, this section describes a hypothetical example. The latter consists of selecting a naval ship. Two scenarios are discussed below. The first one assumes a multiple-user decision situation where there is an exclusive division of tasks at upper-level decision. In other words, each group member is assumed to have special expertise and is assigned to evaluate the alternatives according to the decision criteria closely related to his knowledge. The second scenario illustrates a group decision situation where collective assessment at the staff level is performed. In other words, each group member has his/her opinions on the entire set of evaluation criteria.

SCENARIO 1: DIVISION OF TASKS AT UPPER-LEVEL DECISION

(1) Decision alternatives:

Naval ships can be bought from three countries: the United Kingdom, the Netherlands and West Germany. This example concentrates on a particular class of
warship, i.e., the Corvette. For the purposes of this scenario, the specifications of the three ships are given in Tables 6 and 7.

(2) Decision makers:
Decision makers include the chief of the weapon department, the chief of the engineering department and the chief of the electronics department. All of the above officers are under the command of the chief of department of new constructors, a Real Admiral. Each of the officers has specific expertise in the performance evaluation of the ship candidates. The chief of the weapon department, officer enjoys however the highest decision power. It is assumed that the decision makers operate under more or less complete information about the ships. Each decision maker has a technical staff of his own that performs detailed surveys of the characteristics of the ships.

(3) Decision making norms:
To get started, a member of the decision group has to define the decision norms. It is assumed that the chief of the weapon department takes this responsibility. As discussed in Chapter V, the group leader sets different distributed computer-based communications norms. Figure 4 is an actual display screen of the interactive norm definition process.

(4) Decision making procedures:
The evaluation process is broken down to group members. Each decision member has the exclusive right to assess the alternatives according to the criteria that are related to his expertise.

(5) Evaluation Criteria:
For the sake of simplicity, this example excludes political and economical issues that in real-life situations often play an important role in the selection process. The evaluation criteria are grouped in four sets: 'gun systems', 'electronics', 'engine' and 'cost'. The latter are respectively analyzed by officer chief of the weapon department, the chief of the engineering department and the chief of the electronics department. Such a division of evaluation task is motivated by the fact that each of the officers detains unique expertise their field. Figure 5 lists the criteria chosen for the ship selection problem.
<table>
<thead>
<tr>
<th></th>
<th>GERMANY, FEDERAL</th>
<th>NETHERLANDS</th>
<th>UNIT. KINGDOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>TYPE 122</td>
<td>TYPE 21</td>
<td></td>
</tr>
<tr>
<td>DISPL (tons)</td>
<td>3600 - Full load</td>
<td>3050 - Standard</td>
<td>3000 - Standard</td>
</tr>
<tr>
<td>DIMENSION (ft)</td>
<td>130 x 14.5 x 6.5</td>
<td>130.5 x 14.4 x 6.2</td>
<td>133 x 15 x 43</td>
</tr>
<tr>
<td>AIRCRAFTS</td>
<td>2 Lynx helicopter</td>
<td>2 AB 212 ASW</td>
<td>2 Lynx helicopter</td>
</tr>
<tr>
<td></td>
<td>with AGS 18 sonar</td>
<td>helicopters</td>
<td></td>
</tr>
<tr>
<td>MISSILES</td>
<td>SSM: 8 Harpoon</td>
<td>SSM: 4 Harpoon</td>
<td>SSM: 8 Harpoon</td>
</tr>
<tr>
<td></td>
<td>SAM: 1-8 Sea Spar</td>
<td>SAM: NATO Sea</td>
<td>SAM: Sea wolf</td>
</tr>
<tr>
<td></td>
<td>2 mult sting</td>
<td>Sparrow PDMS</td>
<td>VLS</td>
</tr>
<tr>
<td></td>
<td>launchers; 2 RAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ASDM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GUNS</td>
<td>1-76mm/62; Breda</td>
<td>2-76 mm/62</td>
<td>1-4.5 in 55 Mk8</td>
</tr>
<tr>
<td></td>
<td>105mm 20 tube</td>
<td>Compact</td>
<td>C/WS: 2-30 mm</td>
</tr>
<tr>
<td></td>
<td>rocket laun</td>
<td></td>
<td>Goalkeeper</td>
</tr>
<tr>
<td>A/S WEAPONS</td>
<td>4 Mk2 32 torpedo</td>
<td>4 MK2 torpedo</td>
<td>6 STWS torpedo</td>
</tr>
<tr>
<td></td>
<td>tubes</td>
<td>fot Mk 46 torp</td>
<td>tubes</td>
</tr>
<tr>
<td>MAIN ENGINES</td>
<td>2 GE-LM 2500</td>
<td>2 Rolls-Royce</td>
<td>2 Rolls-Royce</td>
</tr>
<tr>
<td></td>
<td>Gas Turbines</td>
<td>TM3B Gas Turb</td>
<td>SMIA Gas Turb.</td>
</tr>
<tr>
<td></td>
<td>2 MTU 20V 956</td>
<td>2 Rolls-Royce</td>
<td>4 Paxman Valenta</td>
</tr>
<tr>
<td></td>
<td>TB92 Diesels</td>
<td>RM1C Gas Turb.</td>
<td>Diesels</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 Gec Electr Mot</td>
</tr>
<tr>
<td>SPEED (KNOTS)</td>
<td>30 knots</td>
<td>30 knots</td>
<td>28 knots</td>
</tr>
<tr>
<td>RANGE (miles)</td>
<td>4000 at 18 knots</td>
<td>4700 at 16 knots</td>
<td>7800 at 15 knots</td>
</tr>
<tr>
<td>COMPLEMENT</td>
<td>204</td>
<td>176</td>
<td>143</td>
</tr>
</tbody>
</table>
TABLE 7 SPECIFICATIONS OF THE WAR SHIPS

**FEDERAL, GERMANY**

**RADAR**
- **1. SURVEILLANCE**: Type 996, Plessey AWS - 5 plus AWS - 6
- **2. SEA WOLF GUIDANCE**: Two Marcony Type 191
- **3. NAVIGATION**: One Kelvin Huges Type 1007

**SONAR**
- Type 2050 (Bow Mounted)
- Type 2031 (Towed array)

**UNITED, KINGDOM**

**RADAR**
- **1. SURVEILLANCE**: HSA DA 08
- **2. FIRE CONTROL**: HSA WM 25 and STIR
- **3. NAVIGATION**: SMA 3RM 20

**SONAR**
- Active Passive Atlas DSQS 21 BZ and BO

**NETHERLANDS**

**RADAR**
- **1. SURFACE SURCH**: One DA - 08
- **2. FIRE CONTROL**: One LW - 08
  - One WM 25 SYSTEM
  - One STIR
- **3. NAVIGATION**: One ZW - 06

**SONAR**
- SOS - 505 Bow Mounted
(6) Individual Prioritization of Evaluation Criteria:
As discussed earlier, Co-oP currently provides two modes for individual prioritization of evaluation criteria. Each group member can choose any combination of these two modes. For this example, the chief of the weapon department, the chief of the engineering department and the chief of the electronics department respectively chose the AHP, direct, and direct methods. Figures 7 to 16 successively display the outputs of the prioritization process of the three decision makers. In order to reduce the number of evaluation iteration, the criteria that score low values are eliminated (Figure 17).

(7) Individual Evaluation of Alternatives
To support the individual evaluation of alternatives, three methods are supported by Co-oP: direct assessment, AHP and ELECTRE. The results of this process are given in Figures 18 to 22.

(8) Group Result
The group result is displayed in Figure 23. It is a combination of the outcomes generated by three decision makers. Figure 23 suggests that TYPE3 is the best one, with an overall score of .34.

SCENARIO 2: COLLECTIVE ASSESSMENT AT THE STAFF LEVEL
To illustrate the Co-oP ability to handle group decision making situations where division of evaluation tasks does not apply, this scenario is identical to the first one with the exception in that there are only four evaluation criteria. Furthermore, these criteria are used by all decision members for evaluating alternatives.

(1) Decision alternatives: Same as in Scenario 1
(2) Decision makers: Same as in Scenario 1
(3) Decision making norms:
Figure 24 is an actual display screen of the interactive norm definition process.
(4) Decision making procedures:
Unlike in scenario 1, each decision member assesses the alternatives according to all of the criteria that are defined for the problem (See Figure 25).
(5) Evaluation Criteria:
For the sake of simplicity, this example retains only four principal criteria, i.e., 'gun systems', 'electronics', 'engine' and 'cost'.

(6) Individual Prioritization of Evaluation Criteria:
Figures 27 and 28 display the outputs of the prioritization process of the first decision maker. In order to reduce the number of evaluation iteration, the criteria that score low values are eliminated (Figure 29).

(7) Individual Evaluation of Alternatives
The results of this process are given in Figures 30 to 41.

(8) Group Result
The group results are displayed in Figure 24 (computed by the Direct Mode) and Figure 25 (computed by ELECTRE). With the direct mode, TYPE 2 is first in all aggregation of preferences techniques, including the sums-of-the-ranks, the additive ranking, the multiplicative ranking and the sum-of-outranking-relations. This result is confirmed by the ELECTRE mode. Note that in the latter mode, only the sums-of-the-ranks and sums-of-outrankings relations are computed.
IX. CONCLUSIONS

This thesis was concentrated on the extension of a the Co-oP decision support system for multiple criteria group decision making. The development focused in the creation of a computer-based communications framework for supporting decision making situations that are distributed in time and in space. The software is written in Pascal and is operational in a network of three personal computers.

A naval warship selection problem was discussed to illustrate the usefulness of the implemented group decision support system.

However, the proposed decision support system can only be applied to a certain class of decision situations. In effect, the decision makers are assumed to be cooperative, and knowledgeable about multiple criteria decision making.
APPENDIX A

THE PROGRAM LISTING

PROGRAM GDSS ( INPUT, OUTPUT );

($v-$, $r-$)

LABEL

normdef, back, solve1, solve2, solve3,
telos, create, gdss1, nai, 10, 20, 30 ;

CONST

size = 25 ;
position1 = 5 ;
position2 = 13 ;
position3 = 21 ;
maxcrit1 = 5 ;
maxcrit2 = 5 ;
maxcrit3 = 5 ;
windows = 3 ;
number : array[1..Windows,1..4] of Integer
= (( 2, 2, 78, 13 ),
( 2, 15, 78, 21 ),
( 2 23, 78, 24 ) ) ;

TYPE

name = string [size] ;
ask = string [5] ;
color = string [28] ;
um2 = array [1..Maxcrit1,1..Maxcrit1] of integer ;
level = array [1..Maxcrit1,1..Maxcrit1] of name ;
vectors = array [1..Maxcrit1,1..Maxcrit1] of real ;
matrix20 = array [1..20,1..20] of real ;
array1 = array [1..9,1..3] of real ;
Title = array [1..Maxcrit1] of name ;
um1 = array [1..Maxcrit1] of integer ;
vectors1 = array [1..Maxcrit1] of real ;
vectorg = array [1..125] of name ;
vectorn = array [1..125] of real ;
vectorf = array [1..20] of real ;
title1 = array [1..20] of name ;
name2 = array [1..6] of name ;
array9 = array [1..9,1..9] of char ;
elpida = array [1..4] of char ;

57
Case1 = record
  name1 : name ;
  numofproblem, levels, numofalternatives, numofusers : integer ;
  level2, level3, level4, level5, level6, level7 : level ;
  level1 : title ;
  sublevel1 : num1 ;
  sublevel2 : num2 ;
  alternatives : title1 ;
end (* record *) ;

Case2 = record
  pfactor, qfactor : real ;
  numofcriteria : integer ;
  numofalternatives : integer ;
  alternatives : title1 ;
  numofusers : integer ;
  solved : array [1..3] of boolean ;
  grading : array1 ;
  Completed : boolean ;
  completedall : boolean ;
  vector1 : vectors1 ;
  vector2, vector3, vector4, vector5, vector6, vector7 : vectors ;
  normvector1 : vectorg ;
  normvector2 : vectorn ;
  normindex : vectorg ;
  altmatrix : matrix20 ;
  finalindex : array[1..3] of boolean ;
  Finalindex1 : array[1..3] of boolean ;
  Ahp : record
    status : boolean ;
    altvector1 : vectorf ;
    numoftries : integer ;
  end ;
  electre : record
    status : boolean ;
    outranking : array9 ;
    numoftries : integer ;
  end ;
end ;(* record *)

Solution1 = record
  username : name ;

58
userid : name;
umofcriteria : integer;
normvector1 : vectorg;
normvector2 : vectorn;
numofalternatives : integer;
alternatives : title1;

ahp : record
        status : boolean;
umoftries : integer;
altpvector1 : vectorf;
end;
electre : record
        status : boolean;
umoftries : integer;
        outranking : array9;
end;
end; (* record *)

norm1 = record
        numofusers ,
        modifytimes,  
        lasttime : integer ;
        usernames : name2 ;
        specindex : title ;
        usersids : name2 ;
        weight : vectors1 ;
        currentname : name ;
        agregation,  
        nai,  
        specialized, 
        broadcasting,  
        modify : boolean ;
        agregationname : elpida ;
end; (* record *)

VAR

problemfile : file of case1 ;
problem ,problem : case1 ;
specfile : file of case2 ;
specfile2 ,specfile1 : case2 ;
solution ,solution : solution1 ;
solutionfile : file of solution1 ;
norm ,norma : norm1 ;
normfile : file of norm1 ;
BEGIN (* main program *)

back:
window ( 1,1,80,23 );
textbackground ( 14 );
 clrscr;
 window (1,24,80,25);
 textbackground ( white ) ;
 clrscr ;
 textcolor ( black ) ;
 gotoxy ( 2,1 ) ;
 write ( 'multiple criteria group dss - main menu ' ) ;
 window ( 1,1,80,23) ;
 textcolor ( blue ) ;
 textbackground ( 14 ) ;
 gotoxy ( 3 , 2 ) ;
 write ( ' main menu ' ) ;
 gotoxy ( 3,4 ) ;
 write ( '1. Multiple criteria group problem definition' ) ;
 gotoxy ( 3,6 ) ;
 write ( ' 2. Group norm definition ' ) ;
 gotoxy ( 3,8 ) ;
 write ( '3. Prioritization of evaluation criteria ' ) ;
 gotoxy ( 3,10 ) ;
 write ( '4. Individual evaluation of alternatives ' ) ;
 gotoxy ( 3,12 ) ;
 write ( ' 5. Direct input of the data ' ) ;
 gotoxy (3,14) ;
 write ( ' 6. Computation of group decision ' ) ;
 gotoxy (3,16) ;
 write ( '7. Identification of negotiable alternatives ' ) ;
 gotoxy ( 3,18) ;
 write ( ' 8. Help ' ) ;
 gotoxy(3,20) ;
 write ( ' 9. Exit ' ) ;
 textcolor ( black ) ;
 repeat
 gotoxy ( 3,22) ;
 clrcls ;
 write ( ' enter a number : ' ) ;
 read ( answer ) ;
 val ( answer,count,code ) ;
 until ( ( 0 ( count) and ( count ( 10 ) and ( code = 0 ) ) ) ;
 case count of
 1: goto create ;
 2: goto normdef ;
 3: goto solve1 ;
 4: goto solve2 ;
 5: goto solve3 ;
 6: goto gdss1 ;
 7: goto nai ;
 8: goto back ;
 9: goto telos ;
end ;
create:

string128 := 'step 1: problem definition' ;
diskstatus ;
createproblem ( problem ) ;

(* correct the data of the problem *)
window ( 1,1,80,17 ) ;
textbackground ( blue ) ;
clrscr ;
textcolor ( white ) ;
display ( problem ) ;

window ( 1,18,80,23 ) ;
textbackground ( 14 ) ;
clrscr ;
window ( 1,24,80,25 ) ;
textbackground ( white ) ;
clrscr ;
textcolor ( black ) ;
gotoxy ( 2,1 ) ;
write ('step 1: multiple criteria group problem definition' ) ;
gotoxy ( 2,2 ) ;
write ( 'correct the data of the problem' ) ;

window ( 1,18,80,23 ) ;
textbackground ( 14 ) ;
clrscr ;
textcolor ( black ) ;
correctdata ( problem ) ;
clrscr ;
window ( 1,1,80,17 ) ;
textbackground ( blue ) ;
clrscr ;
textcolor ( white ) ;
display1 ( problem ) ;

window ( 1,18,80,23 ) ;
textbackground ( 14 ) ;
clrscr ;
textcolor ( black ) ;
correctdata1 ( problem ) ;
openfile ( prname ) ;
writeproblemfile ;
goto back ;

(* norm definition *)
normdef:
string128 := 'step 2: normdefinition' ;
diskstatus ;
normdefinition ;
opennormfile ( normname ) ;
writtenormfile ;
goto back ;

(* priority of criteria *)
solve1:
priorityofcriteria ;
goto back ;

(* evaluation of alternatives *)
solve2:
computealternatives ;
goto back ;

(* direct input of the data *)
solve3:
string128 := 'step 5: direct input of the weights';
diskstatus ;
clrscr ;
window ( 1,24,80,25 ) ;
textcolor ( black ) ;
textbackground ( white ) ;
gotoxy ( 2,2 ) ;
clreol ;
write ( 'identification of the problem' ) ;
window ( 1,13,80,23 );
textbackground ( 14 );
clrscr ;
read1 ;
readproblemfile ;
read2 ;
readnormfile ;
read3 ;
if ( not exist ( pruser ) ) then begin
    solution.ahp.status := false ;
solution.electre.status := false ;
solution.ahp.numoftries := 0 ;
solution.electre.numoftries := 0 ;
open solutionfile ( pruser )
end ;
readsolutionfile ;
numofcriteria := solution.Numofcriteria ;
normvector1 := solution.Normvector1 ;
normvector2 := solution.Normvector2 ;
read4 ;
writenormfile ;
counttimes := solution.Ahp.Numoftries ;
if norm.Modify then begin
    if counttimes < norm.Modifytimes then begin
        begin
            counttimes := counttimes + 1 ;
            index := true ;
solvewithahp ;
        end
    else
    begin
        clrscr ;
gotoxy ( 5,3 ) ;
write ( 'you can modify your output ' ) ;
gotoxy ( 5,10 ) ;
write ( 'hit any key to continue ' ) ;
read ( kbd,ch ) ;
64
goto back;
end;
end else
begin
if countimes = 0 then
begin
countimes := countimes + 1;
index := true;
solvewithahp;
clrscr;
end
else
begin
clrscr;
gotoxy(5,9);
write('you cant modify your output');
gotoxy(5,10);
write('hit any key to continue');
read(kbd,ch);
goto back;
end;
end;
goto back;

(* gdss *)
gdssl:
string128 := 'step 5 : computation of group result';
string129 := '';
diskstatus;
data;
readproblemfile;
if norm.specialized Then
begin
if (not exist(specialname)) then
begin
clrscr;
gotoxy(2,2);
write('the problem is not yet solved');
gotoxy(2,4);
write('hit any key to continue');
read(kbd,ch);
goto back;
end;
readspecfile;

if ((specfile2.Completedall) and

(specfile2.Electre.Status)) then
begin
    window (1,1,80,23);
    textbackground (blue);
clrscr;

    window (1,24,80,25);
    textbackground (white);
clrscr;
textcolor (black);
goxy (2,1);
write ('step 6: computation of group decision');

    gotoxy (2,2);
write ('final result (electre) - specialized mode');
    window (1,1,80,23);

    textbackground (blue);
clrscr;
textcolor (white);

for a := 1 to specfile2.Numofalternatives do
begin
    answer := specfile2.Alternatives[a];
delete (answer,4,length(answer));
goxy (2,a+3);
write (answer:4);
end;

for a := 1 to specfile2.Numofalternatives do
begin
    answer := specfile2.Alternatives[a];
delete (answer,4,length(answer));
goxy (5 +(a*5),3);
write (answer:3);
end;

for a := 1 to specfile2.Numofalternatives do
begin
    for b := 1 to specfile2.Numofalternatives do
begin
    gotoxy (5 +(b*5),a+3);
write (specfile2.Electre.Outranking[a,b]);
end;
end;
textcolor (green);
goxy (5,10);
write ('** an outranking relation * is the');
gotoxy (5,11);
write('one that satisfies both concordance ');
gotoxy (5,12);
write('and discordance requirements. ');
gotoxy (5,13);
write('** an - indicates that there is ');
gotoxy (5,14);
write('no outranking relations. ');
gotoxy (5,16);
write('hit any key to continue ');
read (kbd,ch);
end;

if ((specfile2.Completedall) and (specfile2.Ahp.Status))
then
begin
    altvector1 := specfile2.Ahp.Altvector1;

    window (1,1,80,23);
textbackground (blue);
clrscr;
window (1,24,80,25);
textbackground (white);
clrscr;
textcolor (black);
gotoxy (2,1);
write('step 6: computation of group decision ');
gotoxy (2,2);
write('final result (ahp) - specialized mode ');

    window (1,1,80,23);
textbackground (blue);
clrscr;
textcolor (white);

gotoxy (2,3);
write('final solution ');

for a1 := 1 to problem.Numofalternatives do begin
    textcolor (white);
gotoxy ((5*a1),19);
write(copy(problem.Alternatives[a1],1,3));
gotoxy ((5*a1),20);
textcolor (red);
write(altvector1[a1]:3:2);
end;

    textbackground (red);
for a1 := 1 to problem.Numofalternatives do begin
  gotoxy( (5 + (5 * a1)),17);
  for b1 := 1 to round(altvector1[a1]*10) do begin
    gotoxy( ((5 * a1)),(17-b1));
    write('');
  end;
end;

textbackground(blue);
gotoxy(2,22);
write('hit any key to continue');
read(kbd,ch);
end
else
  clrscr;
gotoxy(2,2);
  write('the problem is not yet solved');
gotoxy(4,2);
end
else
gdss;
goto back;
(* nai *)

nai:

(* not available yet *)
goto back;
telos:
END (* MAIN PROGRAM *)

#include file step1

procedure createproblem(var problem : case1);

label
  10, 20 ,30 ;
var
  axz : integer ;
  str1,str2 : name ;
  code1 : integer ;
textmode (c80);
c1rscr;
gotoxy(1,2);
clear1(problem);
problem.Levels := 0;
for c := 1 to 5 do
problem.Sublevel[c] := 0;
for c := 1 to 5 do
begin
for b := 1 to 5 do
problem.Sublevel2[c,b] := 0;
end;
textbackground (black);
textcolor (white);

window (1,1,80,23);
textbackground (14);
c1rscr;

window (1,24,80,25);
textbackground (white);

c1rscr;
textcolor (black);
gotoxy(2,1);
write ('step 1: multiple criteria group problem definition');
gotoxy(2,2);
write ('definition of alternatives * hit q to stop');
window (1,1,80,23);
textbackground (14);
c1rscr;
gotoxy(2,2);
textcolor (black);
write ('enter the name of the problem: ');
read (answer);
prname := answer;
delete(answer,8,length(answer));
prname := concat(answer,'.Def');

problem.Name1 := answer;

gotoxy (1,2);
c1reol;
gotoxy (3,2);
write ('name of problem: ', answer);
line := 4;
a := 0;
b := 0;
w := 1;
position := 1;
c := 0;
gotoxy (1, line);
clearscreen ( line);

gotoxy (3, 4);
write ('enter the alternatives :');
a5 := 0;
while ((answer <> 'q') and (a5 < 19)) do
begin
  gotoxy (42, (4 + a5));
a5 := a5 + 1;
  write ('a', a5, '.');
  Read (answer);
  answer := stiupcase(answer);
  problem. Alternatives [a5] := answer;
end;
problem. Numofalternatives := a5 - 1;
for a5 := 1 to 10 do
begin
  gotoxy (1, a5 + 2);
cleareol;
end;
window (1, 24, 80, 25);
textbackground (white);
clear;
textcolor (black);
gotoxy (2, 1);
write ('step 1 - multiple criteria group problem definition ');
gotoxy (2, 2);
write ('definition of criteria * 1)st level 2)nd level 3)rd level
        q)uit');

window (1, 1, 80, 23);
textbackground (14);
textcolor (black);
repeat
  gotoxy (3, line);
  write ('enter the number of the level : ');
  read (answer);
  answer := stiupcase(answer);
line := 4;
deline;
until (answer = '1') or (answer = '2') or (answer = '3')
while answer <> 'q' do
begin
  clearscreen (line);

if answer = '1' then begin
  gotoxy(5, line);
  lreol;
  10:
  position := position1;
  clearscreen(line);
  textcolor(blue);
  gotoxy(position, line);
  textcolor(blue);
  write(a + 1, '.');
  Gotoxy(position + 3, line);
  read(answer);
  answer := stupidcase(answer);
  if (answer > '2') and (answer > '3') and (answer > 'q') and (answer > '1') then begin
    a := a + 1;
    problem.Level1[a] := answer;
    line := line + 1;
    b := 0;
    goto 10
  end;
end;
problem.levels := a;

if answer = '2' then begin
  gotoxy(5, line);
  clreol;
  textcolor(14);
  position := position2;
  20:
  clearscreen(line);
  textcolor(red);
  gotoxy(position, line);
  write(a, '.', b + 1);
  Gotoxy(position + 5, line);
  read(answer);
  answer := stupidcase(answer);
  if (answer > '1') and (answer > '3') and (answer > 'q') and (answer > '2') then begin
    b := b + 1;
    problem.Level2[a, b] := answer;
    line := line + 1;
    c := 0;
    goto 20
  end;
end;
problem.Sublevel1[a] := b;
if answer = '3' then
begin
  gotoxy(5,line) ;
  clreol ;
  textcolor ( yellow ) ;
  position := position3 ;
30:  clearscreer ( line ) ;
  textcolor ( yellow ) ;
  gotoxy ( position,line ) ;
  write (a,'.',b,'.',c+ ) ;
  Gotoxy (position +7, line ) ;
  read ( answer ) ;
  answer := stupcase(answer) ;
  while answer = '3' do
  begin
    gotoxy ( position +7, line ) ;
    clreol ;
    read ( answer ) ;
    answer := stupcase(answer) ;
  end;
  if ( answer () '2' ) and ( answer () '1' )
    and ( answer () 'q' ) then
  begin
    c := c +1 ;
  case a of
    1 : problem.Level3[b,c] := answer ;
    2 : problem.Level4[b,c] := answer ;
    3 : problem.Level5[b,c] := answer ;
    4 : problem.Level6[b,c] := answer ;
    5 : problem.Level7[b,c] := answer ;
  end;
  line := line + 1 ;
  goto 30
  end ;
end ;
window (1,1,80,25 ) ;
clrscr ;

END ;

PROCEDURE DISPLAY ( problem : case1 ) ;

VAR
  line , a , b , c ,col1,col2,col3 : integer ;
  change : boolean ;
BEGIN

gotoxy(3,1);
textcolor(white);
line := 2;
coll := 2;
change := false;
for a:=1 to problem.levels Do
begin
  if (length(problem.level1[a]) > 1) Then
  begin
    textcolor(white);
    begin
      if (line > 14) then
        gotoxy(coll + 30, line);
      line := 2;
      change := true;
    end;
    begin
      if ((line > 14) and (change)) then
        gotoxy(coll + 60, line);
    end;
    writeln(a, '.', Problem.Level1[a]);
    line := line + 1;
  end;
for b:=1 to problem.Sublevel1[a] do
begin
  textcolor(red);
  if (length(problem.Level2[a,b]) > 1) then
  begin
    if (line > 14) then
      gotoxy(coll + 30, line);
    line := 2;
    change := true;
  end;
  begin
    if ((line > 14) and (change)) then
      gotoxy(coll + 60, line);
  end;
  writeln(a, ', ', B, ', ', problem.Level2[a,b]);
  line := line + 1;
for c:=1 to problem.Sublevel2[a,b] do
begin
  textcolor(yellow);
  writeln(c, ' ', problem.Sublevel2[a,b]);
end;
case a of
  1: begin
    if (length(problem.Level3[b,c])>1) then
      begin
        if (line>15) then
          begin
            coll := coll + 30;
            line := 2;
            change := true;
          end;
        if ((line>14) and (change)) then
          begin
            coll := coll + 60;
            line := 2;
          end;
        gotoxy (col+2,line);
        write(a,'.',b,'.',c,'',
              problem.Level3[b,c]);
        line := line + 1;
      end;
    end;
  2: begin
    if (length(problem.Level4[b,c])>1) then
      begin
        if (line>15) then
          begin
            coll := coll + 30;
            line := 2;
            change := true;
          end;
        if ((line>14) and (change)) then
          begin
            coll := coll + 60;
            line := 2;
          end;
        gotoxy (col+2,line);
        write(a,'.',b,'.',c,'',
              problem.Level4[b,c]);
        line := line + 1;
      end;
    end;
  3: begin
    if (length(problem.Level5[b,c])>1) then
      begin
        if (line>15) then
          begin
            coll := coll + 30;
            line := 2;
            change := true;
          end;
if ((line > 14) and (change)) then begin
    coll := coll + 60;
    line := 2;
end;
gotoxy (coll+2, line);
write(a,'.',B,'.',C,' ',
    problem.Level5[b,c]) ;
line := line + 1;
end;
end;

4: begin
    if (length(problem.Level6[b,c]) > 1) then begin
        if (line > 15) then begin
            coll := coll + 30;
            line := 2;
            change := true;
        end;
        if ((line > 14) and (change)) then begin
            coll := coll + 60;
            line := 2;
        end;
gotoxy (coll+2, line);
write(a,'.',B,'.',C,' ',
    problem.Level6[b,c]) ;
line := line + 1;
end;
end;

5: begin
    if (length(problem.Level7[b,c]) > 1) then begin
        if (line > 15) then begin
            coll := coll + 30;
            line := 2;
            change := true;
        end;
        if ((line > 14) and (change)) then begin
            coll := coll + 60;
            line := 2;
        end;
gotoxy (coll+2, line);
write(a,'.',B,'.',C,' ',
    problem.Level7[b,c]) ;
line := line + 1;
PROCEDURE DISPLAY1 (problem : case1);

VAR
  line : integer;
BEGIN
  gotoxy(3,2);
  textcolor(white);
  write ('alternatives: ');
  for line := 1 to problem.numofalternatives Do
  begin
    gotoxy(4, line + 3);
    write (line, '.', problem.alternatives[line]);
  end;
END;

PROCEDURE CORRECTDATA (var problem : case1);
BEGIN
  repeat
    gotoxy(1,2);
    write ('do you want to modify the criteria (y/n)?');
    Repeat
gotoxy(47,2);
    clrscr;
    read (answer);
    answer := stupcase(answer);
  until ((answer = 'y') or (answer = 'n'));
  if answer = 'y' then begin
    gotoxy(1,4);
    write ('enter the tree level (e.g., 2.1.3)?');
    gotoxy(47,4);
    read (answer2);
  end;
END;
answer2 := stupcase(answer2);
gotoxy ( 1,6 ) ;
write ( ' name of criteria ', answer2 ) ;
gotoxy ( 43,6 ) ;
write ( '?' ) ;
Gotoxy( 47,6 ) ;
read( answer1 ) ;
w := 1 ;
convert ( answer2, w, d1 ) ;
a1 := d1 ;
w := 3 ;
convert ( answer2, w, d1 ) ;
b1 := d1 ;
w := 5 ;
convert ( answer2, w, d1 ) ;
c1 := d1 ;
if ( c1 = 0 ) and ( b1 = 0 ) and ( a1 = 0 ) then
  problem.Level[a1] := stupcase( answer1 ) ;
if ( a1 ) problem.Levels ) Then
  problem.Levels := problem.Levels + 1 ;
if c1 = 0 then
begin
  problem.Level2[a1, b1] := stupcase( answer1 ) ;
  if ( b1 ) problem.Sublevel1[a1] ) then
end
else
begin
case a1 of
  1: problem.Level3[b1, c1] := stupcase( answer1 ) ;
  2: problem.Level4[b1, c1] := stupcase( answer1 ) ;
  3: problem.Level5[b1, c1] := stupcase( answer1 ) ;
  4: problem.Level6[b1, c1] := stupcase( answer1 ) ;
  5: problem.Level7[b1, c1] := stupcase( answer1 ) ;
end ;
if ( c1 ) problem.Sublevel2[a1, b1] ) then
problem.Sublevel2[a1, b1] := problem.Sublevel2[a1, b1]+1 ;
end ;
gotoxy ( 47,2 ) ;
cleol ;
gotoxy ( 47,4 ) ;
cleol ;
gotoxy ( 47,6 ) ;
cleol ;
a1 := 0 ; b1 := 0 ; c1 := 0 ;
end;
until answer = 'n' ;
END ;
PROCEDURE CORRECTDATA1 ( var problem : case1 ) ;

BEGIN
   repeat
      gotoxy (1,2) ;
      write ('do you want to modify the alternatives (y/n)?') ;
      Repeat
         gotoxy(52,2) ;
         clreol ;
         read (answer) ;
         answer := stucase(answer) ;
      until ( ( answer = 'y') or ( answer = 'n') ) ;
   if answer = 'y' then
      begin
         gotoxy(1,4) ;
         write ('enter the number of the alternative (e.g., 3)')
         gotoxy(53,4) ;
         read (answer2) ;
         answer2 := stucase(answer2) ;
         gotoxy (1,6) ;
         write ('name of alternative ', answer2) ;
         gotoxy (33,6) ;
         write ('?')
         gotoxy(37,6) ;
         read (answer1) ;
         val(answer2, a, code) ;
         problem.alternatives[a] := Stupcase(answer1) ;
         gotoxy (47,2) ;
         clreol ;
         gotoxy (47,4) ;
         clreol ;
         gotoxy (47,6) ;
         clreol ;
      end ;
      until answer = 'n' ;
   END ;

   

INCLUDE FILE STEP2

OVERLAY PROCEDURE NORMDEFINITION ;
VAR
   x1,y1,limit : integer ;
   count3 : real ;
   lasthour : string[22] ;
   problemname1 : name ;
BEGIN

window ( 1,1,80,22 );
textbackground ( 14 ); clrscr ;

window ( 1,24,80,25 );
textbackground ( white ); clrscr ;
textcolor ( black );
gotoxy ( 2,1 );
write ( ' step 2 : group norm definition ' );

window ( 1,1,80,22 );
textbackground ( 14 ); clrscr ;
textcolor ( black );
gotoxy ( 2,2 );
write ( ' name of the group norm ' );
gotoxy ( 25,2 );
write ( '?' );
Read ( answer );
norm.Currentname := stupcase(answer); delete ( answer,8,length(answer) );
normname := concat ( answer , '.Gn' );
textcolor ( blue );
gotoxy ( 2,4 );
write ( ' 1. Identification of group members ' );
textcolor ( black );
gotoxy ( 5,6 );
write ( ' 1.1 Number of group members ( max 3 ) ' );
gotoxy ( 52,6 );
write ( '?' );
Count3 ::= 0 ;
x1 ::= 55 ; y1 ::= 6 ; limit ::= 4 ;
checknumber ( answer , x1,y1,limit,count3 );
norm.Numofusers ::= trunc( count3 );
for a ::= 1 to trunc( count3 ) do
begin

gotoxy ( 9,6+a );
write ( ' - name of member # ', a );
gotoxy ( 52,6+a );
write ( '?' );
Gotoxy ( 55,6+a );
read ( answer );
norm.Usersnames[a] ::= stupcase(answer); end;
gotoxy ( 5,a+7 );
write ('1.2 Id of member ', norm. Usersnames[1])
gotoxy (52, 7 + a)
write ('?
')
Gotoxy (55, a + 7)
read (answer)

gotoxy (2, 12)
textcolor (blue)
write ('1.2. Group decision techniques')
textcolor (black)
gotoxy (5, 14)
write ('2.1 Weighted majority rule
')
gotoxy (9, 15)
write ('= equal weights (y/n)
')
gotoxy (52, 15)
write ('?
')
x1 := 55; y1 := 15
identify (answer, x1, y1)

if answer = 'y' then
begin
for a := 1 to norm. Numofusers do
    norm. Weight[a] := 1
end
else
begin
for a := 1 to norm. Numofusers do
begin
    gotoxy (12, 15 + a)
    write ('- weight for ', norm. Usersnames[a])
gotoxy (52, 15 + a)
    write ('?
')
    Count3 := 0
    x1 := 55; y1 := (15 + a); limit := 100
    checknumber (answer, x1, y1, limit, count3)
    norm. Weight[a] := count3
end
end
clrscr

gotoxy (5, 2)
textcolor (blue)
write ('2.2 Collective evaluation mode
')
gotoxy (8, 4)
textcolor (black)
write ('choose one of the following modes :
')
gotoxy (10, 6)
write ('(1) each group member will evaluate
alternatives')
gotoxy ( 10, 7 );
write ( ' according to all criteria.' );
gotoxy ( 10, 8 );
write ( ' (2) Each group member will evaluate only
alternatives' );
gotoxy ( 10, 9 );
write ( ' according to his exclusive area of
expertise.' );
gotoxy ( 8, 11 );
write ( ' Enter a number ? ' );

Count3 := 0;
x1 := 31; y1 := 11; limit := 2;
checknumber ( answer, x1, y1, limit, count3 );

if answer = '1' then
  norm.Specialized := false
else
begin
  norm.Specialized := true;
a := 0;
repeat
gotoxy ( 8, 13 );
clreol;
write ( ' the name of the problem ? ' );
Read ( answer );
delete ( answer, 8, length ( answer ) );
prname := concat ( answer, '.Def' );
norm.normnamex := prname;
Problemname1 := answer;
until exist ( prname );
readproblemfile;
for a := 1 to problem.Levels do
begin
gotoxy ( 16, 14 + a );
write ( ' name of user for critiria
problem.Leveli[a], ? ' );
error := false;
repeat
gotoxy ( 54, 14 + a );
clreol;
read ( answer );
answer := stupcase ( answer );
for b := 1 to norm.numofusers do
begin
  if answer = norm.usersnames[b] then
    error := true;
end;
until error;
Norm.Specindex[a] := answer;
end;
end;
clrscr;

gotoxy ( 5,2 );
write ( '2.3 Automatic selection of techniques of' );
gotoxy ( 5,3 );
write ( 'aggregation of preference (y/n)' );
gotoxy ( 52,3 );
write ( '?' );
X1 := 55 ; y1 := 3 ;
identify ( answer ,x1,y1 ) ;

if answer = 'y' then
  norm.Agregation := true
else
begin
  norm.Agregation := false;
  a := 0 ;
gotoxy ( 9,5 );
write ( 'r1 : sum of ranks (y/n)' );
gotoxy ( 52,5 );
write ( '?' );
X1 := 55 ; y1 := 5 ;
identify ( answer ,x1,y1 ) ;
if answer = 'y' then
begin
  a := a + 1 ;
  norm.Agregationname[a] := 'l' ;
end
else
begin
  a := a + 1 ;
  norm.Agregationname[a] := 'e' ;
end;
gotoxy ( 9,6 );
write ( 'r2 : sum of outranking relations ' );
gotoxy ( 52,6 );
write ( '?' );
X1 := 55 ; y1 := 6 ;
identify ( answer ,x1,y1 ) ;
if answer = 'y' then
begin
  a := a + 1 ;
  norm.Agregationname[a] := 'z' ;
end
else
begin
  a := a + 1 ;

82
norm.Aggregationname[a] := 'e' ;
end ;
gotoxy ( 9,7 ) ;
write ('- r3 : additive ranking (y/n)') ;
gotoxy ( 52,7 ) ;
write (' ? ' ) ;
X1 := 55 ; y1 := 7 ;
identify ( answer , x1,y1 ) ;
if answer = 'y' then
begin
  a := a + 1 ;
  norm.Aggregationname[a] := '3' ;
end
else
begin
  a := a + 1 ;
  norm.Aggregationname[a] := '3' ;
end ;
gotoxy ( 3,3 ) ;
write ('- r4 : multiplicative ranking (y/n)') ;
gotoxy ( 52,9 ) ;
write (' ? ' ) ;
X1 := 55 ; y1 := 8 ;
if answer = 'y' then
begin
  a := a + 1 ;
  norm.Aggregationname[a] := '4' ;
end
else
begin
  a := a + 1 ;
  norm.Aggregationname[a] := 'e' ;
end ;
gotoxy ( 5,9 ) ;
write ('2.4 Automatic computation of nai (y/n) ') ;
gotoxy ( 52,9 ) ;
write (' ? ' ) ;
X1 := 55 ; y1 := 9 ;
identify ( answer , x1,y1 ) ;
if answer = 'y' then
norm.Nai := true
else
  norm.Nai := false ;
clrscr ;
gotoxy ( 2,2 ) ;
textcolor ( blue ) ;
write ('3. Information exchange ' ) ;
textcolor ( black ) ;
gotoxy ( 5,4 ) ;
write ( '3.1 Broadcasting of individual outputs (y/n)' );
gotoxy ( 52,4);
write ( '?' );
X1 := 55 ; y1 := 4 ;
identify ( answer , x1,y1 );
if answer = 'y' then
  norm.Broadcasting := true
else
begin
  norm.Broadcasting := false ;
end ;
gotoxy ( 5,5 );
write ( '3.2 Permission to modify individual analyses ' );
gotoxy ( 5,6 );
write ( '?' );
identify ( answer , x1,y1 );
if answer = 'n' then
  norm.Modify := false
else
begin
  norm.Modify := true ;
gotoxy ( 9,7 );
write ( '3.2.1 How many times (max 10 )' );
gotoxy ( 52,7);
write ( '?' );
Count3 := 0 ;
checknumber ( answer , x1,y1,limit,count3 ) ;
norm.Modifytimes := trunc( count3 ) ;
if norm.Specialized then
end ;
gotoxy ( 5,8 );
write ( '3.3 Time limit to submit individuals results : ' );
gotoxy ( 9,9 );
write ( '3.3.1 How many days (max 14 )' );
gotoxy ( 52,9);
write ( '?' );
Count3 := 0 ;
checknumber ( answer , x1,y1,limit,count3 ) ;
norm.Lasttime := trunc( count3 ) ;
gotoxy ( 9,10 );
write ( '3.3.2 Hours ( 1:00 to 24:00 )' );
gotoxy ( 52,10);
write ( '?' );
Gotoxy ( 55,10 );
read ( lasthour ) ;
for a := 2 to norm.Numofusers do
  norm.Usersids[a] := 'X' ;
END ;
INCLUDE FILE STEP2-1

PROCEDURE NORMSELECTION ( VAR X1,Y1 : INTEGER ) ;
BEGIN

gotoxy ( x1,y1 ) ;
write ( ' name of the norm ? ' ) ;
Repeat
gotoxy ( 23,y1 ) ;
criseol ;
read ( answer ) ;
norm.Currentname := stupcase( answer ) ;
normname := concat ( answer , '.Gn' ) ;
until ( exist ( normname ) ) ;
END ;

PROCEDURE DISPLAYNORM ;

VAR
message1,message2 : string[80] ;

BEGIN

window ( 1,1,80,25 ) ;
textbackground ( blue ) ;
crisr ;
textcolor ( white ) ;
readnormfile ;
crisr ;
gotoxy ( 2,2 ) ;
write ( ' name of the group norm : ',norm.Currentname ) ;
gotoxy ( 2,3 ) ;
write ( ' 1. Identification of group members ' ) ;
gotoxy ( 5,4 ) ;
write ( ' 1.1 Number of group members : ',norm.Numofusers ) ;
for a := 1 to norm.Numofusers do
begin
  gotoxy ( 9,4+a ) ;
  write ( ' - name of member # ',a :' ' ,norm.Usersnames[a] ) ;
end ;
gotoxy ( 2,9 ) ;
write ( ' 2. Group decision techniques ' ) ;
gotoxy ( 5,10 ) ;
write ( ' 2.1 Weighted majority rule ' ) ;
gotoxy (9,11);
write ('- weights of members: ');
for a := 1 to norm.Numofusers do
begin
  gotoxy (12, 11 + a);
  write (a, ' ', Norm.Usersnames[a],',
  norm.weight[a]:4:2);
end;

if norm.specialized then
begin
  message1 := 'Each group member will evaluate only
  alternatives';
  message2 := 'according to his exclusive area of
  expertise';
end
else
begin
  message1 := 'each group member will evaluate
  alternatives';
  message2 := 'according to all criteria';
end;

gotoxy (5,15);
write ('2.2 Collective evaluation mode: ');
gotoxy (9,16);
write (message1);
gotoxy (9,17);
write (message2);

if norm.specialized then
begin
  prname := norm.normnamex;
  readproblemfile;
  gotoxy(11,18); write ( 'Criteria' );
  gotoxy(25,18); write ( 'user name' );
  for a := 1 to problem.levels do
  begin
    gotoxy (11,18 + a);
    write ( problem.level1[a] );
    gotoxy (35,18 + a);
    write ( norm.specindex[a] );
  end;
end;
TextoCOLOR (red);
gotoxy (2,25);
write ('hit any key to continue');
read (kbd, ch);
crlscr;
textcolor (white);
gotoxy (5,3);
write ( '2.3 Selection of techniques of aggregation of 
preference : ' ) ;
if norm.Aggregation then 
begin
  gotoxy ( 65,3 ) ;
  write ( ' r1 r2 r3 r4 ' ) ;
end
else
begin
  gotoxy ( 60,3 ) ;
  for a := 1 to 4 do 
  begin
    case norm.Aggregationname[a] of 
'1': write ( 'r1 ' ) ;
'2': write ( 'r2 ' ) ;
'3': write ( 'r3 ' ) ;
'4': write ( 'r4 ' ) ;
end;
end;
end;

go to xy ( 5,5 ) ;
write ( '2.4 Automatic computation of nai : ' ) ;
if norm.Nai then 
  write ( 'yes ' )
else 
  write ( 'no ' ) ;
go toxy ( 2,8 ) ;
write ( '3. Information exchange ' ) ;
go toxy ( 5,9 ) ;
write ( '3.1 Broadcasting of individual outputs : ' ) ;
if norm.Broadcasting then 
  write ( 'yes ' )
else 
  write ( 'no ' ) ;
go toxy ( 5,10 ) ;
write ( '3.2 Permission to modify individual 

group analysis : ' ) ;
if norm.Modify then 
begin
  writeln ( 'yes ' ) ;
  write ( 'you can modify the input ' 
  norm.Modifytimes,' times ' ) ;
end
else 
  write ( 'no ' ) ;
go toxy ( 5,12 ) ;
write ( '3.3 Time limit to submit individual: ' ) ;
go toxy ( 9,13 ) ;
write ( 'date : ', norm.Lasttime ) ;
go toxy ( 9,14 ) ;
write ('hour: ', '22:30');
textcolor (red);
gotoxy (2,25);
write ('hit any key to continue');
read (kbd, ch);

END;

INCLUDE FILE STEP3

PROCEDURE PRIORITYOFCRITERIA;

LABEL

telosx,telosx1,telosx2;

VAR

pruser1,filename2: name;
error5: boolean;

PROCEDURE FINALWEIGHTS;

BEGIN

for a := 1 to problem.Levels do begin
  for b := 1 to problem.Sublevel1[a] do begin
    vector2[a,b] := vector2[a,b] * vector1[a];
    for c := 1 to problem.Sublevel2[a,b] do begin
      case a of
        1: vector3[b,c] := vector3[b,c] * vector3[a,b];
        2: vector4[b,c] := vector4[b,c] * vector3[a,b];
        3: vector5[b,c] := vector5[b,c] * vector3[a,b];
        4: vector6[b,c] := vector6[b,c] * vector3[a,b];
        5: vector7[b,c] := vector7[b,c] * vector3[a,b];
      end;
    end;
    b := 0;
  end;
  END;
END.
PROCEDURE FINALCRITERIA1;
BEGIN
  numofcriteria := 1;
  selectcriteria ( problem, vector1, vector2, vector3,
                  vector4, vector5, vector6, vector7,
                  normvector1, normvector2, numofcriteria );
  numofcriteria := ( numofcriteria - 1 );
  sort1 ( normvector1, normvector2, numofcriteria );
  finalcriteria ( normvector1, normvector2, numofcriteria );
  solution.Numofcriteria := numofcriteria;
  solution.Normvector2 := normvector2;
  solution.Normvector1 := normvector1;
  solution.Username := name;
  if ( not norm.specialized ) Then
    begin
      if ( not exist ( pruser ) ) then
        begin
          solution.Ahp.Status := false;
          solution.Electre.Status := false;
          solution.Ahp.Numoftries := 0;
          solution.Electre.Numoftries := 0;
          opensolutionfile ( pruser );
        end;
        writesolutionfile;
      end;
    end;
END;

BEGIN (* main *)

string128 := 'step 3:prioritization of evaluation criteria';
diskstatus;
string129 := 'identification of the problem methods: ahp or direct';
data;
writelnormfile;
read5;
readproblemfile;
cirscr;
if norm.Specialized then
  begin
    if ( not exist ( pruser3 ) ) then
      begin
        error5 := false;
      end;
    end;
END.
for a := 1 to 3 do
begin
  specfile2.solved[A] := false;
  specfile2.Finalindex[a] := false;
end;
specfile2.completed := False;
specfile2.Completedall := false;
openspecfile ( pruser3 );
writespecfile;
end
else
  error5 := true;
readspecfile;
vector1 := specfile2.vector1;
vector2 := specfile2.vector2;
vector3 := specfile2.vector3;
vector4 := specfile2.vector4;
vector5 := specfile2.vector5;
vector6 := specfile2.vector6;
vector7 := specfile2.vector7;

b := 0;
repeat
  b := b + 1;
  until ( namex = norm.usersnames[b] );
specfile2.solved[b] := true;
end;

If methodx = 'ahp' then
  evaluate ( problem.Level1,problem.Levels,vector1 )
else
  direct1 ( problem.Level1,problem.Levels,vector1 ) ;
clrsclr ;
if ( ( norm.specialized ) And ( error5 ) ) then
begin
  for a := 1 to problem.levels Do
    vector1[a] := ( vector1[a] + specfile2.vector1[a ] )/2;
end;
for mal1 := 1 to problem.Levels do
begin
  if norm.Specialized then
  begin
    if norm.Specindex[mal1] <> namex then
goto telosx ;
  end;
  for mal2 := 1 to problem.Sublevel1[mal1] do
begin
  vectortan[mal2] := vector2[mal1,mal2];
  array2[mal2] := problem.Level2[mal1,mal2];
end;
if methodX = 'ahp' then
evaluate (array2, problem.Sublevel1[ma1], vectortan)
else
direct1 (array2, problem.Sublevel1[ma1], vectortan )
ma12 := 0;
for ma12 := 1 to problem.Sublevel1[ma1] do
begin
  if norm.Specialized then
  begin
    if norm.Specindex[ma1] () nameX then
      goto telosX;
  end;
  vectortan[ma1, ma12] := vectortan[ma12 ];
end;
ma12 := 0;
telosX:
end;
c1rsc1;
for si := 1 to problem.Levels do
begin
  if norm.Specialized then
  begin
    if norm.Specindex[si] () nameX then
      goto telosXi;
  end;
  for s2 := 1 to problem.Sublevel1[si] do
begin
  for s3 := 1 to problem.Sublevel2[si, s2] do
begin
  case s1 of
  1: begin
        array2[s3] := problem.Level3[si, s3 ];
        vectortan[s3] := vector3[si, s3 ];
      end;
  2: begin
        array2[s3] := problem.Level4[si, s3 ];
        vectortan[s3] := vector4[si, s3 ];
      end;
  3: begin
        array2[s3] := problem.Level5[si, s3 ];
        vectortan[s3] := vector5[si, s3 ];
      end;
  4: begin
        array2[s3] := problem.Level6[si, s3 ];
        vectortan[s3] := vector6[si, s3 ];
      end;
  5: begin

array2[s3] := problem.Level7[s2,s3] ;
vector tan[s3] := vector7[s2,s3] ;
end ;
end ;
s3 := \emptyset ;
if methodx = 'ahn' then
    evaluate(array2, problem.Sublevel2[s1,s2], vector tan)
else
direct1(array2, problem.Sublevel2[s1,s2], vector tan);
clsscr ;
for s3 := 1 to problem.Sublevel2[s1,s2] do
begin
    if norm.Specialized then
        begin
            if norm Specindex[s1] <> namex then
                goto te losx1 ;
        end ;
    case s1 of
        1 : vector3[s2,s3] := vector tan[s3] ;
        2 : vector4[s2,s3] := vector tan[s3] ;
        3 : vector5[s2,s3] := vector tan[s3] ;
        4 : vector6[s2,s3] := vector tan[s3] ;
        5 : vector7[s2,s3] := vector tan[s3] ;
    end ;
end ;
s2 := \emptyset ;
telo sx1:
end ;
if ( not norm.Specialized ) then
begin
    finalweights ;
    finalcriterial ;
end
else
    begin
    specfile2.vector1 := vector1 ;
    specfile2.vector2 := vector2 ;
    specfile2.vector3 := vector3 ;
    specfile2.vector4 := vector4 ;
    specfile2.vector5 := vector5 ;
    specfile2.vector6 := vector6 ;
    specfile2.vector7 := vector7 ;
    write specfile ;
end ;
if norm.specialized then
begin
readspecfile ;
\text{mall} := 0 ;
for a:=1 to 3 do
begin
if specfile2.solved[a] then
\text{mall} := \text{mall} + 1 ;
end ;
if \text{mall} = norm.numofusers then
begin
vector1 := specfile2.vector1 ;
vector2 := specfile2.vector2 ;
vector3 := specfile2.vector3 ;
vector4 := specfile2.vector4 ;
vector5 := specfile2.vector5 ;
vector5 := specfile2.vector6 ;
vector7 := specfile2.vector7 ;
finalweights ;
finalcriteria1 ;
specfile2.completed := true ;
specfile2.normvector1 := normvector1 ;
specfile2.normvector2 := normvector2 ;
for a := 1 to numofcriteria do
begin
\text{for mall} := 1 to problem.levels do
begin
if problem.level1[mall] = normvector1[a] then
begin
specfile2.normindex[a] := norm.specindex[mall] ;
goto telosx2 ;
end ;
end ;
end ;
end ;
end ;

\text{for s1} := 1 to problem.Levels do
begin
\text{for s2} := 1 to problem.Sublevel1[s1] do
begin
\text{end} ;
for s3 := 1 to problem.Sublevel2[s1, s2] do 
begin 
  case s1 of 
    1: begin 
      if problem.level13[s2, s3] = normvector1[a] then begin 
        specfile2.normindex[a] := norm.specindex[S1]; 
        goto telosx2; 
      end; 
    end; 
    2: begin 
      if problem.level4[s2, s3] = normvector1[a] then begin 
        specfile2.normindex[a] := norm.specindex[S1]; 
        goto telosx2; 
      end; 
    end; 
    3: begin 
      if problem.level5[s3] = normvector1[a] then begin 
        specfile2.normindex[a] := norm.specindex[S1]; 
        goto telosx2; 
      end; 
    end; 
    4: begin 
      if problem.level6[s2, s3] = normvector1[a] then begin 
        specfile2.normindex[a] := norm.specindex[S1]; 
        goto telosx2; 
      end; 
    end; 
    5: begin 
      if problem.level7[s2, s3] = normvector1[a] then begin 
        specfile2.normindex[a] := norm.specindex[S1]; 
        goto telosx2; 
      end; 
    end; 
  end; 
end; 
telosx2: 
end; 
specfile2.numofcriteria := numofcriteria; 
specfile2.numofalternatives := problem.numofalternatives; 
specfile2.alternatives := problem.alternatives; 
writespecfile; 
end; 
end; 
END;
INCLUDE FILE STEP3-1

OVERLAY PROCEDURE EVALUATE (var array2:title; var w :integer;
var vector:tan; vectors1);

LABEL

ert1, ert3;

CONST

count = 3;

VAR

a3, b3, c3, d3, h3, k3, f3, p3, : a1, b1,
levels1, i, count1, histogram : integer;
row, row1, lambda, c1, r1, cr,
score, answer3, integer1 : real;
array5, vectorbase, exchange3 : vectors1;
st : string [9];
ch : char;
lamda1, vector2 : array [1..50] of real;
exchange4 : array[1..20] of name;
matrix3, result, matrix4 : array [1..20, 1..20] of real;
answer4 : name;

PROCEDURE INFO ;

BEGIN

window (1, 13, 90, 23);
textbackground(14);
textcolor( red);
gotoxy(1, 6);
writein (' note: be as accurate as possible ... ... ... greater than 1 e.g, 2.45 ');
Writein (' a possible scale for inexact 13: ');
writein (' 3 = weakly important than ');
more important than ');
writein (' 7 = very strongly more imp. than ');
absolutely more imp. than');

END ;

BEGIN (* main *)

window (1, 1, 50, 12);
textbackground(blue);
c1rscr;
window (51,1,80,12);
textbackground ( white );
crscr;
window (1,13,80,23 );
textbackground(14 );
crscr ;
window (1,24,80,25 );
textbackground (white );
crscr;
textcolor ( black );
gotoxy ( 2,1 );
write ('step 3 : prioritization of evaluation criteria ');
gotoxy ( 2,2 );
write ('method : ahp ');
levels1 := w ;
if levels1 () 0 then
begin
    window ( 1,1,50,12 ) ;
textbackground (blue);
crscr;

    window (51,1,80,21);
textbackground ( white );
crscr ;

    window (1,13,80,23 );
textbackground(14 );
crscr ;

    if w = 1 then
    vectortan[w] := 1 ;
    if w = 2 then
    begin
        window ( 1,1,50,12 ) ;
textbackground (blue);
textcolor ( white );
vectorbase := vectortan ;
levels1 := w ;
gotoxy ( 1,1 );
write (' pairwise comparison ');
gotoxy ( 10,3 ) ;
for al := 1 to levels1 do
    write ( copy( array2[al],1,5) , ' ' ) ;
for al := 1 to levels1 do
begin
    gotoxy ( 2 ,3+a1 ) ;
    write ( copy( array2[al],1,5) ) ;
end ;
window (51,1,80,21);
textbackground ( white ) ;
textcolor( Ø ) ;
gotoxy (1,1) ;
write ( ' priority vector ' ) ;
for al := 1 to levels1 do
begin

gotoxy ( 2 ,3 +a1 ) ;
write ( copy( array2[al],1,16) ) ;
end ;
for a := 1 to levels1 do
begin

gotoxy(20,3 + a ) ;
write ( chr( 179 ) ) ;
end ;
for a := 1 to levels1 do
begin

gotoxy(27,3 + a ) ;
write ( chr( 179 ) ) ;
end ;

window (1,13,80,23) ;
textbackground(14) ;
clrscr ;
textcolor( black ) ;
gotoxy( 2,2) ;
write ( ' is ',array2[1],', more important than ',
Repeat

gotoxy ( 75,2 ) ;
cleol ;
read ( answer ) ;
answer := stlupcase ( answer ) ;
until ( ( answer = 'y') or ( answer = 'n') ) ;
if answer = 'y' then
begin

gotoxy(2,3) ;
write ( ' how many times is ',array2[1],', more important than ',
array2[2], ' ? ' ) ;
Gotoxy ( 2,4) ;
write ( ' ( see note below ) ' ) ;
info ;
textcolor ( black ) ;
repeat

gotoxy(75,3) ;
cleol ;
read ( answer ) ;
val ( answer,answer3,code ) ;
until ( ( code = Ø ) and ( answer3 ) Ø ) ) ;
vectortan[1] := ((10 / (answer3 + 1)) * answer3)/10;
vectortan[2] := ( 10 / ( answer3 + 1 ) ) / 10;
window (1,1,50,12);
textbackground(blue);
textcolor ( white );
gotoxy (18, 4);
write ( answer3:4:2 );
gotoxy(10,5);
write ((1 / answer3):4:2 );
end
else
if answer = 'n' then
begin
gotoxy(2,3);
write ('how many times is ',array2[2],', more
important than ',array2[1],'? ');
gotoxy(2,4);
write ('( See note below )');

info ;
textcolor ( black );
repeat
    gotoxy(75,3);
    clrscr;
    read ( answer );
    val ( answer,answer3,code );
until code = 0;
vectortan[2] := (((10 / (answer3 + 1)) * answer3 )/10;
vectortan[1] := ( 10 / ( answer3 +1 ) ) / 10;
window (1,1,50,12);
textbackground(blue);
textcolor ( white );
gotoxy(10,5);
write ((1/ answer3):4:2 );
gotoxy(18,4);
write ( answer3:4:2 );
end;
window (51,1,80,21);
textbackground ( white );
textcolor( black );
for al := 1 to levels1 do
begin
    gotoxy(22,al+3);
    write ( ( vectortan[al]):5:3 );
end;

window (1,13,80,23);
textbackground(14);
crscr;
textcolor ( black );
for a1 := 1 to 2 do
begin
  gotoxy ( ( 5 * a1 ), 9 );
  write ( copy( array2[a1], 1, 3 ) );
end;

for a1 := 1 to 2 do
begin
  gotoxy ( ( 5 * a1 ), 10 );
  write ( ( vectortan[a1] ) );
end;
textbackground ( green );
for a1 := 1 to 2 do
begin
  gotoxy ( ( 5 * a1 ), 8 );
  write ( ' ' );
end;

for a1 := 1 to 2 do
begin
  gotoxy ( ( 5 + ( 5 * a1 ) ), 9 );
  if ( round( vectortan[a1] * 10 ) ) > 7 then
    istogram := 7
  else
    istogram := ( round( vectortan[a1] * 10 ) );
  for bl := 1 to istogram do
begin
    gotoxy ( ( 5 * a1 ), ( 9 - bl ) );
    write ( ' ' );
end;
end;
textbackground ( 14 );
textcolor ( black );
gotoxy ( 2, 11 );
write ( ' hit any key to continue ' );
read ( kbd, ch );
end;

if ( w ) > 2 then
begin
  for a3 := 1 to 5 do
    matrix2[a3, a3] := 1;
  window ( 1, 1, 150, 12 );
textbackground ( blue );
textcolor ( white );
vectorbase := vectortan;
levels1 := w;
gotoxy ( 1, 1 );
write ( ' pairwise comparison ' );
gotoxy ( 10, 3 );

99
for a[1] := 1 to levels1 do
    write ( copy( array2[a[1],1,5] , ' ' ) );
for a[1] := 1 to levels1 do
begin
    gotoxy (2,3+a[1] );
    write ( copy( array2[a[1],1,5] ) );
end;

window (51,1,80,21);
textbackground ( white ) ;
textcolor ( 0 ) ;
gotoxy (1,1) ;
write ( ' priority vector ' ) ;
for a[1] := 1 to levels1 do
begin
    gotoxy ( 2,3+a[1] );
    write ( copy( array2[a[1],1,18] ) );
end;
for a := 1 to levels1 do
begin
    gotoxy (20,3 + a ) ;
    write ( chr ( 179 ) ) ;
end;

for a := 1 to levels1 do
begin
    gotoxy (27,3 + a ) ;
    write ( chr ( 179 ) ) ;
end;

info ;
textcolor ( 0 ) ;
for a := 1 to ( levels1 - 1 ) do
begin
    criterial := array2[a] ;
    for b := 1 to ( levels1 - a ) do
begin
    criteria2 := array2[a+b] ;
    repeat
        gotoxy ( 1,2 ) ;
        write ('is ',criterial,'more important than ',
        criteria2 ,'(y/n) ? ' ) ;
        Gotoxy ( 77,2 ) ;
        clrscr ;
        read ( answer ) ;
        answer := stupcase ( answer ) ;
        clrscr ;
        until ((answer = 'y') or ( answer = 'n' ) ) ;
    if answer = 'y' then
begin


gotoxy(1,3);
write('how many times is ',criteria1:5,' more
important than ',criteria2:5,'? ');
gotoxy(1,4);
write(' ( See note below ) '); repeat
gotoxy(77,3);
cleared;
read(answer);
val(answer,answer3,code);
until((code=0) and (answer3=0));
matrix2[a,a+b]:=answer3;
matrix2[a+b,a]:=(1/answer3);
matrix2[a,a]:=1;
window(1,1,50,12);
textbackground(blue);
textcolor(white);
gotoxy((2+((a+b)*8)),3+a);
write((answer3:4):2);
gotoxy(2+(a*8),(3+(a+b)));
write(((1/answer3):4):2);
end;
if answer='n' then begin
gotoxy(1,3);
write('how many times is ',criteria2,' more
important than ',criteria1:5,'? ');
gotoxy(1,4);
write(' ( See note below ) '); repeat
gotoxy(75,3);
cleared;
read(answer);
val(answer,answer3,code);
until((code=0) and (answer3=0));
matrix2[a,a+b]:=(1/answer3);
matrix2[a+b,a]:=answer3;
matrix2[a,a]:=1;
window(1,1,50,12);
textbackground(blue);
textcolor(white);
gotoxy((2+((a+b)*8)),3+a);
write(((1/answer3):4):2);
gotoxy(2+(a*8),(3+(a+b)));
write((answer3:4):2);
end;
window(1,13,80,23);
textbackground(14); 
textcolor( black );
gotoxy(1,2);
clreol;
gotoxy(1,3);
clreol;
gotoxy(1,4);
clreol;
end;
end;

matrix2[levels1,levels1] := 1;

err1:

matrix3 := matrix2;
for a3 := 1 to count do
begin
for b3 := 1 to levels1 do
begin
for c3 := 1 to levels1 do
array5[c3] := matrix2[b3,a3];

for h3 := 1 to levels1 do
begin
score := 0;
for k3 := 1 to levels1 do
begin
integer1 := array5[k3] * matrix2[k3,h3];
score := score + integer1;
end;
result[b3,h3] := score;
end;
end;
matrix2 := result;
end;
result := matrix2;

for p3 := 1 to levels1 do
begin
row := 0;
for f3 := 1 to levels1 do
row := row + result[p3,f3];
vectorbase[p3] := row;
end;
row1 := 0;
for p3 := 1 to levels1 do
row1 := row1 + vectorbase[p3];
for p3 := 1 to levels1 do
window(51,1,80,21);
textbackground (white);
for al := 1 to levels1 do
begin
    gotoxy(22, a1 + 3);
    write (vectorbase[a1]; 5:3);
end;

window (1, 13, 80, 24);
textbackground(14);

integer1 := 0;
for al := 1 to levels1 do
begin
    score := 0;
    for bl := 1 to levels1 do
begin
        integer1 := matrix3[a1, bl] * vectorbase[bl];
        score := score + integer1;
    end;
    lamda[a1] := score;
end;
integer1 := 0;
for al := 1 to levels1 do
begin
    vector2[a1] := lamda[a1] / vectorbase[a1];
    integer1 := integer1 + vector2[a1];
end;
lamda := (integer1 / levels1);
if levels1 = 1 then
    levels1 := 2;

ci := ((lamda - levels1) / (levels1 - 1));
case levels1 of
1: ri := 0.000000000001;
2: R1 := 0.000000000001;
3: R1 := 0.58;
4: R1 := 0.90;
5: R1 := 1.12;
6: R1 := 1.24;
7: R1 := 1.32;
8: R1 := 1.41;
9: R1 := 1.45;
10: R1 := 1.49;
11: R1 := 1.51;
12: R1 := 1.48;
13: R1 := 1.56;
14: R1 := 1.57;
15: R1 := 1.59;
end;
cr := ci / ri;

window (1,13,80,23);
textbackground(14);
cls
vectortan := vectorbase;

repeat
count1 := 0;
for a := 1 to (levels1 - 1) do
begin
if vectorbase[a] < vectorbase[a+1] then
begin
exchange3[a] := vectorbase[a];
vectorbase[a] := vectorbase[a+1];
vectorbase[a+1] := exchange3[a];
exchange4[a] := array2[a];
array2[a] := array2[a+1];
array2[a+1] := exchange4[a];
count1 := count1 + 1;
end;
end;
until count1 = 0;

for a1 := 1 to levels1 do
begin
  gotoxy ((5 * a1), 9);
  write (copy(array2[a1], 1, 3));
end;
for a1 := 1 to levels1 do
begin
  gotoxy ((5 * a1), 10);
  write (vectorbase[a1]:3:2);
end;

textbackground (green);
for a1 := 1 to levels1 do
begin
  gotoxy ((5 * a1), 8);
  write ('   ');
end;

for a1 := 1 to problem.Levels do
begin
  gotoxy ((5 + (5 * a1)), 9);
  if (round(vectortan[a1] * 10) 7) then
  istogram := 7
  else
  istogram := (round(vectortan[a1] * 10));
for b1 := 1 to istogram do
    begin
        gotoxy((5*a1),(9-b1));
        write("");
        end;
    end;

textbackground(14);
textcolor(blue);
gotoxy(36,1);
write(">** lambda max =", lamda:4:2);
gotoxy(36,2);
write("> consistency index =", ci:4:2);
gotoxy(36,3);
write("> randomized index =", ri:4:2);
gotoxy(36,4);
write("> consistency ratio =", cr:4:2);

gotoxy(36,6);
write("> there is some statistical'');
gotoxy(36,7);
write("> inconsistency in your evaluation.'');
Gotoxy(36,8);
write("> (study highlighted values for '));
gotoxy(36,9);
write("> probable inconsistent evaluation');
textcolor(black);

col:
for p3 := 1 to levels1 do
    begin
        for f3 := 1 to levels1 do
            begin
                result[p3,f3] := 0;
                matrix2[p3,f3] := 0;
            end;
    end;

gotoxy(2,11);
textcolor(blue);
write("do you want to modify the evaluation of the
   criteria (y/n)? ")
Textcolor(black);
repeat
    gotoxy(65,11);
cleared;
    read (answer);
    answer := stucase (answer);
until ((answer = 'y') or (answer = 'n'));
window (1,13,80,23);
textbackground(14);
if answer = 'y' then
begin
  clrscr;
  error := false;
  repeat
    gotoxy (2,2);
    clrrel;
    write ('name of the first criteria ? ');
    Read (answer);
    answer := copy(answer,1,4);
    answer := tocase(answer);
    for ai := 1 to levels1 do
      begin
        answer4 := copy(array2[ai],1,4);
        if (answer4 = answer) then
          error := true;
      end;
  until error;
  a := 1;
  repeat
    a := a + 1;
    answer4 := copy(array2[a],1,4);
  until (answer = answer4);
  criteria1 := array2[a];
end

ert3:
  error := false;
  repeat
    gotoxy (2,3);
    clrrel;
    write ('name of the second criteria ? ');
    Read (answer);
    answer := tocase(answer);
    answer := copy(answer,1,4);
    for bi := 1 to levels1 do
      begin
        answer4 := copy(array2[bi],1,4);
        if (answer4 = answer) then
          error := true;
      end;
  until error;
  answer4 := copy(criteria1,1,4);
  if (answer = answer4) then
    goto ert3;

b := 0;
repeat
  b := b + 1;
  answer4 := copy(array2[b],1,4);
until (answer4 = answer);
criteria2 := array2[b];
window (1,1,50,12);
textbackground(blue);
matrix2 := matrix3;
textcolor(red + 16);
gotoxy((2 + (b * 8)), 3 + a);
write(matrix2[a,b]:4:2);
textcolor(black);
window(1,13,80,23);
textbackground(14);
clrscr;
repeat
  gotoxy(1,2);
  write('is', criteria1, ' more important than ', criteria2, ' (y/n)?');
Gotoxy(75,2);
c1reol;
  read(answer);
  answer := stupcase(answer);
c1reol;
until((answer = 'y') or (answer = 'n'));
if answer = 'y' then
  begin
    gotoxy(1,3);
    write('how many times is ', criteria1, ' more important than ', criteria2, '?');
    gotoxy(1,4);
    write(' (See note below)');
    info;
    repeat
      gotoxy(75,3);
      c1reol;
      read(answer);
      val(answer, answer3, code);
      until((code = 0) and (answer3 < 0))
      matrix2[a,b] := answer3;
      matrix2[b,a] := (1 / answer3);
      matrix2[a,a] := 1;
      window(1,1,50,12);
textbackground(blue);
textcolor(white);
gotoxy((2 + (b * 8)), 3 + a);
write(answer3:4:2);
gotoxy(2 + (a * 8), (3 + b));
write((1 / answer3):4:2);
end;
window(1,13,80,23);
textbackground(14);
textcolor(black)
if answer = 'n' then
begin
  gotoxy (1,3);
  write ('how many times is ', criteria2,
       ' more important than ', criteria1,'? ');
  gotoxy (1,4);
  write (' (See note below ) ');
  info;
repeat
  gotoxy(75,3);
  clreol;
  read (answer);
  val (answer, answer3, code);
  until ((code = 0) and (answer3 <> 0));
  matrix2[a,b] := (1/answer3);
  matrix2[b,a] := answer3;
  matrix2[a,a] := 1;

  window (1,1,50,12);
  textbackground(blue);
  gotoxy ((2+(b*3)), 3+a);
  write ((1/answer3):4:2);
  gotoxy(2+(a*8), (3+b));
  write (answer3:4:2);
end;

window (1,1,80,23);

GET.RET.
end;

window (1,1,80,25);

END.

OVERLAY PROCEDURE DIRECT1 (var array2:title; var w:integer;
var vectortan: vectors1);

LABEL
  ert1, ert3;
CONST

    count = 3 ;

VAR

    a3, b3, c3, d3, h3, k3, f3, p3, l, a1, b1,
    levels1, i, count1,istogram : integer ;
    row, row1, lambda, ci, ri, cr,
    score, answer3, integer1 : real ;
    array5, vectorbase, exchange3 : vectors1 ;
    st : string [9] ;
    ch : char ;
    lambda1, vector2 : array [1..50] of real ;
    exchange4 : array[1..20] of name ;
    matrix2, result, matrix3 : array [1..20, 1..20] of real ;
    answer4 : name ;

PROCEDURE INFO ;
BEGIN

    window (1,13,80,23 ) ;
    textbackground(14) ;
    clrscr ;
    textcolor( red ) ;
    gotoxy(1,10) ;
    writeln ( "note : be as accurate as possible
             -- any # between 0 and 10 e.g., 2.45 ' ) ;
    writeln ( ' a possible scale for inexact is : ' ) ;
    writeln ( ' 3 = weakly important than , 5 = strongly more
             important than ',
             ' 7 = very strongly more imp. than
             9 = absolutely more imp. than' );

END ;

BEGIN (* main *)

    window ( 1,1,50,12 ) ;
    textbackground(blue);
    clrscr ;

    window (51,1,80,12);
    textbackground ( white ) ;
    clrscr ;
    window (1,13,80,23 ) ;
    textbackground(14) ;
    clrscr ;
window (1,24,80,25);
textbackground(white);
cIrsr;
textcolor ( black ) ;
gotoxy(2,1); 
write ( 'step 3 : prioritization of evaluation criteria' );
gotoxy(2,2); 
write ( 'direct input of criteria weights' );
levelsl := w ;
if levelsl <> 0 then
begin
  window (1,1,50,12);
textbackground(blue);
cIrsr ;
  window (51,1,80,21);
textbackground ( white ) ;
cIrsr ;
  window (1,13,80,23);
textbackground(14); 
cIrsr ;
  if w = 1 then
    vectortan[w] := 1
  else
  begin
    for a3 := 1 to 5 do
      matrix2[a3,a3] := 1 ;
    window (1,1,50,12);
textbackground(blue);
textcolor ( white ) ;
  vectorbase := vectortan ;
  levels1 := w ;
gotoxy (1,1) ;
  window (51,1,80,21);
textbackground ( white ) ;
textcolor ( 0 ) ;
gotoxy (1,1) ;
write ( ' priority vector ' ) ;
for a1 := 1 to levels1 do
begin
  gotoxy (2,3+a1);
  write (copy (array2[a1],1,18)) ;
end ;
for a := 1 to levels1 do
begin
  gotoxy (20,3 + a) ;
  write ( chr (179) ) ;
end ;

110
for a := 1 to levels1 do
begin
  gotoxy(27,3 + a );
  write ( chr ( 179 ) );
end;

ert1:
  info;
  textcolor ( 0 );
  gotoxy ( 2,2 );
  write ( 'enter the weights of the criteria : ' );
  for a := 1 to levels1 do
  begin
    gotoxy ( 2 , 2+a );
    write ( array2[a] , ' : ' );
  end;

  for a := 1 to levels1 do
  begin
    repeat
      gotoxy ( length(array2[a])+5 , 2+a );
      clrscr;
      read ( answer ) ;
      val ( answer , answer3 , code ) ;
      until (( code = 0 ) and ( answer3 ) -1 ) and
      ( answer3 ( 11 ) ) ;
    vectorbase[a] := answer3 ;
  end;

row1 := 0 ;
for p3 := 1 to levels1 do
  row1 := row1 + vectorbase[p3] ;
for p3 := 1 to levels1 do

window (51,1,80,21);
textbackground ( white ) ;
for a1 := 1 to levels1 do
begin
  gotoxy(22,a1+3);
  write ( vectorbase[a1]:5:3 ) ;
end;

window (1,13,80,23 );
textbackground(14) ;
clrscr ;
vevortan := vectorbase ;
repeat
  count1 := 0 ;
  for a := 1 to ( levels1 - 1 ) do

begin
if vectorbase[a] < vectorbase[a+1] then
begin
  exchange3[a] := vectorbase[a];
  vectorbase[a] := vectorbase[a+1];
  vectorbase[a+1] := exchange3[a];
  exchange4[a] := array2[a];
  array2[a] := array2[a+1];
  array2[a+1] := exchange4[a];
count1 := count1 + 1;
end;
end;
until count1 = 0;

for a := 1 to levels1 do
begin
  gotoxy ( ( ( 5 * a ) ) , 9 ) ;
  write ( copy ( array2[a], 1, 3 ) ) ;
end;

for a := 1 to levels1 do
begin
  gotoxy ( ( ( 5 * a ) ) , 10 ) ;
  write ( vectorbase[a]:3:2) ;
end;
textbackground ( green ) ;
for a := 1 to levels1 do
begin
  gotoxy ( ( 5 * a ) ) , 8 ) ;
  write ( ' ' ) ;
end;

for a := 1 to levels1 do
begin
  gotoxy ( (5 + ( 5 * a1 ) ) , 9 ) ;
  for b := 1 to round ( vectorbase[a1] * 10 ) do
begin
  gotoxy ( (5 * a1 ) , (9 -b1 ) ) ;
  write ( ' ' ) ;
  end;
end;
gotoxy (2,11);
textbackground ( 14 ) ;
textcolor ( blue ) ;
write(' do you want to modify the evaluation of the criteria (y/n) ? ');
Textcolor ( black);
repeat
  gotoxy( 65,11 ) ;

112
cireol;
read ( answer ) ;
answer := strtoupper ( answer ) ;
until ((answer = 'y') or (answer = 'n')) ;
window (1,13,30,23) ;
textbackground (14) ;
if answer = 'y' then
begin
   clrscr ;
goto ert1 ;
end ;
end ;
END ;

INCLUSE FILE STEP3-2

PROCEDURE SELECTCRITERIA (var problem : case1 ;
var vector1 : vectors1 ;
var vector2, vector3,
vector4, vector5,
vector6, vector7 : vectors ;
var normvector1 : vectorg ;
var normvector2 : vectorn ;
var numofcriterial : integer ) ;

VAR

f, a, b, c        : integer ;
normvector5      : vectorg ;
normvector6      : vectorn ;
numofcriterial   : integer ;

BEGIN

for a := 1 to 125 do
begin
   normvector2[a] := 0 ;
   normvector1[a] := ' ' ;
end ;
f := numofcriterial ;
for a := 1 to problem.Levels do
begin
   if problem.Sublevel1[a] = 0 then
   begin
      normvector1[f] := problem.Level1[a] ;
      normvector2[f] := vector1[a] ;
   end ;
end ;
\[ f := f + 1; \]
end
else
begin
for \( b := 1 \) to problem.Sublevel1[a] do
begin
if problem.Sublevel2[a, b] = 0 then
begin
normvector1[f] := problem.Level2[a, b];
normvector2[f] := vector2[a, b];
f := f + 1;
end
else
begin
for \( c := 1 \) to problem.Sublevel2[a, b] do
begin
case \( a \) of
1:
begin
normvector1[f] := problem.Level3[b, c];
normvector2[f] := vector3[b, c];
f := f + 1;
end;
2:
begin
normvector1[f] := problem.Level4[b, c];
normvector2[f] := vector4[b, c];
f := f + 1;
end;
3:
begin
normvector1[f] := problem.Level5[b, c];
normvector2[f] := vector5[b, c];
f := f + 1;
end;
4:
begin
normvector1[f] := problem.Level6[b, c];
normvector2[f] := vector6[b, c];
f := f + 1;
end;
5:
begin
normvector1[f] := problem.Level7[b, c];
normvector2[f] := vector7[b, c];
f := f + 1;
end;
end;
end;
end;
end;
end;
umofcriteria := f;
END;
PROCEDURE FINALCRITERIA ( var normvector1 : vectorg;
   var normvector2 : vectorg;
   var numofcriteria : integer );

VAR

   a, b, number, startpoint : integer;
   sum, percent : real;
   answer : char;
   numofcriterial : integer;
   normvector5 : vectorg;
   normvector6 : vectorg;

PROCEDURE WRITECRITERIA;

VAR

   linex, rowx : integer;

BEGIN

   window ( 1,1,80,15 );
   textbackground ( blue );
   textcolor ( white );
   clrscr;
   gotoxy ( 3,2 );
   write ( 'the final criteria ( ', numofcriterial , ') 
        and their weights are : ' );
   gotoxy ( 1,4 );
   sum := 0;
   numofcriterial := numofcriterial;
   normvector5 := normvector1;
   normvector6 := normvector2;
   for a := 1 to numofcriterial do
      sum := sum + normvector2[a];
   for a := 1 to numofcriterial do
      begin
         normvector2[a] := normvector2[a] / sum;
         end;
   linex := 3; rowx := 2;
   for a := 1 to numofcriterial do
      begin
         if ( linex ) 13 ) then
            begin
               linex := 3;
               rowx := 45;
            end;
         gotoxy ( rowx,linex );
         write (a,'. ',Normvector1[a], normvector2[a]:4:2 );
         linex := linex + 1;
      end;
END;
BEGIN (* main *)
    window ( 1,1,80,15 );
    textbackground ( blue );
    clrscr ;

    window ( 1,16,80,23 );
    textbackground ( 14 );
    clrscr ;

    window ( 1,24,80,25 );
    textbackground ( white );
    clrscr ;
    textcolor ( black );
    gotoxy ( 2,1 );
    write ('step 3 : prioritization of evaluation criteria ');
    gotoxy (2,2);
    write ('determine the number of the criteria ');

    startpoint := numofcriteria ;
    writecriteria ;

    window ( 1,16,80,23 );
    textbackground ( 14 );
    clrscr ;
    textcolor ( black );
    gotoxy ( 2,2 );
    write ('do you want to reduce the number of the criteria (y/n) ? ');
    Repeat
        gotoxy (63,2);
        clrreol;
        read ( answer );
        answer := stupcase ( answer );
        until ( ( answer = 'y') or ( answer = 'n') );

repeat
    if answer = 'y' then
    begin
        gotoxy( 2,2);
        clrreol;
        write ('you have two methods : ');
        gotoxy (4,4);
        write (' 1. Define the number of the criteria that you want to use ');
        gotoxy (4,5);
        write (' 2. Define the sum (%) that you wish ');
        gotoxy (2,7);
        write ('method that you wish (1 or 2) ? ');
        Repeat
            gotoxy( 50,7);
            clrreol;

3

116
read ( answer ) ;
until ( ( answer = '1' ) or ( answer = '2' ) ) ;
if answer = '1' then
begin
  gotoxy ( 2,2 ) ;
  write ('the number of the criteria that you wish
(up to',startpoint , ')?') ;
Repeat
  gotoxy ( 60 ,2 ) ;
  clreol ;
  read ( number ) ;
until ( number (= startpoint ) ) ;
numofcriteria := number ;
end ;
if answer = '2' then
begin
  clrscr ;
  gotoxy ( 2,2 ) ;
  write ('enter the value (%) that you wish:') ;
  read ( precent ) ;
  sum := 0 ;
a := 0 ;
repeat
  a := a + 1 ;
  sum := sum + ( normvector2[a] * 100 ) ;
b := a ;
until ( sum > precent ) ;
umofcriteria := ( b - 1) ;
end ;
end ;
writecriteria ;
window ( 1,16,80,23 ) ;
clrscr ;
textcolor ( black ) ;
gotoxy ( 2,2 ) ;
write ('do you want to change the number of the
criteria (y/n) ?') ;
Repeat
  gotoxy ( 70,2 ) ;
  read ( answer ) ;
  answer := stucase ( answer ) ;
until (( answer = 'y' ) or ( answer = 'n' ) ) ;
if answer = 'y' then
begin
  numofcriteria := numofcriteria1 ;
normvector1 := normvector5 ;
normvector2 := normvector6 ;
clrscr ;
end ;
until ( answer = 'n' ) ;
END ;

117
INCLUDE FILE STEP4

PROCEDURE SOLVEWITHAHP;

LABEL
telos3x;

PROCEDURE DISPLAYFINALS;
BEGIN

  window (1,1,80,23);
  textbackground ( blue );
  clrscr;

  window (1,24,80,25);
  textbackground ( white );
  clrscr;
  textcolor ( black );
gotoxy ( 2,1 );

  if index then
    write ( 'step 5 : direct input of the weights' );
  else
    write ( 'step 4 : individual evaluation of alternatives' );

gotoxy ( 2,2);
write ( 'final result' );

  window (1,1,80,23);
  textbackground ( blue );
  clrscr;
  textcolor ( white );
gotoxy ( 2,3);
write ( 'final solution' );

  for al := 1 to problem.Numofalternatives do begin
    textcolor ( white );
gotoxy ( ( ( 5 * al ) ), 19 );
    write ( copy( problem.Alternatives[al],1,3 ) );
gotoxy ( ( ( 5 * al ) ), 20 );
textcolor ( red );
    write ( altvector1[al]:3:2 );
  end;

  textbackground ( red );
  for al := 1 to problem.Numofalternatives do

BEGIN (* main *)

if index then
begin
alternatives1 := problem.alternatives;
Evaluate3 ( alternatives1, altvector, problem.Numofalternatives, normvector1, ax );

alternativesx := problem.alternatives;
altvectorx := altvector;
altvectorl := altvector;

End
else
begin
if norm.Specialized then
altmatrix := specfile2.Altmatrix;
if indexE then
begin
for ax := 1 to numofcriteria do
begin
if norm.specialized Then
begin
if specfile2.normindex(ax) <> namex Then
goto telos3x;
end;
alternatives1 := problem.Alternatives;
evaluate1 ( alternatives1, altvector, problem.Numofalternatives, normvector1, ax );
for b := 1 to problem.Numofalternatives do
altmatrix [ b,ax ] := altvector[b];
end;
end;
else
begin
specfile2.Electre.Numoftries :=
solution.Electre.Numoftries;
specfile2.Altmatrix := altmatrix;
writespecfile;
end;
end;
end end
else
begin
alternatives1 := problem.Alternatives ;
direct2a ( alternatives1 , altvector ,
        problem.Numofalternatives, normvector1, ax ) ;
if norm.Specialized then
begin
specfile2.Electre.Numoftrries :=
solution.Electre.Numoftrries ;
specfile2.Altmatrix := altmatrix ;
end ;
end : if not norm.Specialized then
sortresult else
begin
b := 0 ;
for a := 1 to 3 do begin
if specfile2.Finalindex[a] then
b := b + 1 ;
end ;
if b = norm.Numofusers then
begin
sortresult ;
specfile2.Completedall := true ;
end ;
end ;
end if not norm.Specialized then
displayfinals else begin
if specfile2.Completedall then
begin
displayfinals ;
specfile2.Amp.Status := true ;
specfile2.Numofcriteria := numofcriteria ;
specfile2.Normvector2 := normvector2 ;
specfile2.Normvector1 := normvector1 ;
specfile2.Numofalternatives :=
problem.Numofalternatives ;
writespecfile;
answer := concat ( ".", Inte );
probrname := concat ( probname, answer );
end;
end;
END;

PROCEDURE COMPUTEALTERNATIVES;
LABEL
telos6;
PROCEDURES ALLUSERS;
LABEL
telos6x:
BEGIN
if methodx = 'electre' then
begin
countimes := solution.Electre.Numoftries;
if norm.Modify then
begin
if countimes = norm.Modifytimes then
begin
countimes := countimes + 1;
clrscr;
solution.Electre.Status := true;
solution.Electre.Numoftries := countimes;
electre;
end
else
begin
clrscr;
gotoxy(5,3);
write('you cant modify your output');
gotoxy(5,10);
write('hit any key to continue');
read(kbd,ch);
goto telos6x;
end;
end
else
begin
if countimes = 0 then
begin
countimes := countimes + 1;
end
end
end
end
end
end
 clrscr ;  
solution.Electre.Status := true ;  
solution.Electre.Numoftries := countimes ;  
electre ;  
end  
else  
begin  
 clrscr ;  
gotoxy ( 5,9 ) ;  
write ( 'you cant modify your output ' ) ;  
gotoxy ( 5,10 ) ;  
write ( 'hit any key to continue ' ) ;  
read ( kbd,ch ) ;  
goto telos6x ;  
end ;  
end ;  
end  
else  
begin  
countimes := solution.Ahp.Numoftries ;  
if norm.Modify then  
begin  
 if countimes < norm.Modifytimes then  
 begin  
 countimes := countimes + 1 ;  
 index := false ;  
 if methodx = 'ahp' then  
 index2 := true  
 else  
 index2 := false ;  
 solvewithahp ;  
 end  
 else  
 begin  
 clrscr ;  
gotoxy ( 5,9 ) ;  
write ( 'you cant modify your output ' ) ;  
gotoxy ( 5,10 ) ;  
write ( 'hit any key to continue ' ) ;  
read ( kbd,ch ) ;  
goto telos6x ;  
end ;  
end  
else  
begin  
 if countimes = 0 then  
 begin  
 countimes := countimes + 1 ;  
 index := false ;  
 if methodx = 'ahp' then  

index2 := true
else
    index2 := false;
solvewithahp;
c1rscr
end
else
begin
c1rscr;
gotoxy ( 5,9 );
write ( 'you cant modify your output ' ) ;
gotoxy ( 5,10 );
write ( 'hit any key to continue ' );
read ( kbd,ch );
goto telos6x;
end;
end;
telos6x:
END;

BEGIN (* main *)

string128 := 'step 4 : individual evaluation of
    alternatives';
diskstatus ;
c1rscr;
window ( 1,24,80,25 );
textcolor ( black ) ;
textbackground ( white ) ;
gotoxy ( 2,2 );
c1reol;
write ( 'identification of the problem
    methods : ahp,electre,direct') ;
window ( 1,13,80,23 );
textbackground ( 14 );
c1rscr;
read1;
readproblemfile ;
read2 ;
readnormfile ;
read3 ;
if not norm.Specialized then
begin
    if ( not exist(pruser) ) then
        begin
            clrscr ;
            write ( ' you must compute first the criteria ' ) ;
            wait ;
        end;
end;

goto telos6;
end;
else
begin
readspecfile;
if ( not specfile2.Completed ) then
begin
clrscr;
write ('the evaluation of the criteria is not yet completed');
wait;
goto telos6;
end;
end;
if ( not norm.Specialized ) then
begin
readsolutionfile;
numofcriteria := solution.Numofcriteria;
normvector1 := solution.Normvector1;
normvector2 := solution.Normvector2;
end
else
begin
numofcriteria := specfile2.Numofcriteria;
normvector1 := specfile2.Normvector1;
normvector2 := specfile2.Normvector2;
end;
read4;
read5;
writtenormfile;
if ( norm.Specialized ) then
begin
a := 0;
repeat
a := a + 1;
until ( nameex = norm.Usersnames[a] )
specfile2.Finalindex[a] := true;
end
allusers;
telos6;
END;
INCLUDE FILE STEP4-1

OVERLAY PROCEDURE EVALUATE1 (VAR ALTERNATIVES : TITLE1;
VAR ALTVECTOR3 : VECTORF;
VAR W : INTEGER;
VAR NORMVECTOR1 : VECTORG;
VAR AX : INTEGER);
LABEL
ert, ert4;

CONST
count = 3;

VAR

matrix2, result, matrix3 : array [1..20, 1..20] of real;
array5, altvector5, exchange3, altvector6 : vectorf;
lambdai, vector2 : vectorf;
exchange4, alternativesk : title1;
umofalternatives, I, count1, xi, yi, ai, bi,
a3, b3, c3, d3, h3, k3, f3, p3, l : integer;
score, answer3, integer1, row, row1,
lamda, ci, ri, cr : real;
st : string [9];
ch : char;

PROCEDURE INFO1
begin
window (1, y1 + 1, 80, 23);
textbackground (14);
textcolor (red);
gotoxy1, 5);
writeln ("Note: be as accurate as possible
   -- any # greater than 1");
writeln ("e.g., 2.45 or 15.");
writeln ("A possible scale for inexact is:");
writeln ("3 = weakly important than, 5 = strongly more
   important than");
writeln ("7 = very strongly more imp. than
   9 = absolutely more imp. than");
Textcolor (black);
end;

BEGIN

numofalternatives := w;
case numofalternatives of
   1, 2, 3, 4, 5, 6, 7 : begin
      x1 := 50; y1 := 12;
   end;

125
end;

8,9,10,11 : begin
    x1 := 55 ; y1 := 13 ;
end;

12,13,14,15 : begin
    x1 := 70 ; y1 := 15 ;
end;
end;

numofalternatives := w ;
if numofalternatives <> 0 then
begin
    window (1,1,x1,y1);
    textbackground(blue);
    clrscr;

    window (x1+1,1,30,y1);
    textbackground(white);
    clrscr;

    window (1,y1+1,80,23);
    textbackground(14);
    clrscr;

    window (1,24,80,25);
    textbackground(white);
    clrscr;
    textcolor(black);
    gotoxy(2,1);
    write('step 4: individual evaluation of alternatives ');
    gotoxy(2,2);
    write('evaluation of alternatives according to criterion ',
         'normvector1[ax]', ' methodx' ) ;
    window (1,1,x1,y1);
    textbackground(blue);
    textcolor(white);
    altvector5 := altvector3;
    numofalternatives := w ;
    gotoxy(1,1);
    write(' pairwise comparison ');
    gotoxy(10,3);
    for a1 := 1 to numofalternatives do
        write(copy(alternatives[a1],1,5));
    for a1 := 1 to numofalternatives do
        begin
            gotoxy(2,3+a1);
            write(copy(alternatives[a1],1,5));
        end;
window (xl+1,1,80,y1);
textbackground ( white ) ;
textcolor ( 0 ) ;
gotoxy ( 1,1 ) ;
write ('  priority vector ' ) ;

for a1 := 1 to numofalternatives do
begin
   gotoxy ( 2 ,3+a1 ) ;
   write (copy( alternatives[a1],1,18) ) ;
end ;
for a := 1 to numofalternatives do
begin
   gotoxy(20,3+a ) ;
   write ( chr ( 179 ) ) ;
end ;
for a := 1 to numofalternatives do
begin
   gotoxy(27,3+a ) ;
   write ( chr ( 179 ) ) ;
end ;

window (1,y1+1,80,23);
textbackground(14) ;
clrscr ;

info1 ;
for a := 1 to numofalternatives do
   matrix2[a,a] := 1 ;

for a := 1 to ( numofalternatives - 1 ) do
begin
   criteria1 := alternatives[a] ;
   for b := 1 to ( numofalternatives - a ) do
begin
      criteria2 := alternatives[a+b] ;
      repeat
         textcolor ( 0 ) ;
         gotoxy( 1,2 ) ;
         write ( '  is ',criteria1, ' better than ',
               criteria2 ,', (y/n) ' ? ' ) ;
         gotoxy ( 64,2 ) ;
clear1 ;
read ( answer ) ;
answer := strtoupper ( answer ) ;
clear1 ;
until ( ( answer = 'y') or ( answer = 'n' ) ) ;
if answer = 'y' then
begin
   textcolor ( black ) ;

127
gotoxy (1,3);
write (' how many times is ', criteria1, ' better than ', criteria2, '? ');
Gotoxy (1,4);
write (' ( see note below ) ');
repeat
gotoxy(64,3);
cireol;
read ( answer );
answer := stupcase ( answer );
val ( answer, answer3, code );
until (( code = 0 ) and ( answer3 > 0 ));
matrix2[a,a+b] := answer3;
matrix2[a+b,a] := (1 / answer3); matrix2[a,a] := 1;

window (1,1,xl,yl);
textbackground ( blue );
gotoxy ( ((3 + ((a + b) * 8)), 3 + a));
textcolor ( white );
write ( answer3:4:2);
gotoxy ( 2 + (a * 8), (3 + (a + b)));
write ( (1 / answer3):4:2);
end;

window (1,y1+1,80,23);
textbackground (14);
textcolor ( black );
if answer = 'n' then
begin
gotoxy (1,3);
write (' how many times is ', criteria2, ' better than ', criteria1, '? ');
Gotoxy (1,4);
write (' ( see note below ) ');
repeat
gotoxy(64,3);
cireol;
read ( answer, answer3, code);
until (( code = 0 ) and ( answer3 > 0 ));
matrix2[a,a+b] := (1 / answer3);
matrix2[a+b,a] := answer3;
matrix2[a,a] := 1;

window (1,1,x1,y1);
textbackground ( blue );
textcolor ( white );
ggotoxy (((2 + ((a + b) * 8)) , 3 + a) ;
 write (('/ answer3':4:2) ;
gotoxy( 2 + (a * 8) , ( 3 + (a + b) )) ;
write (answer3:4:2) ;
end :

window (1,y1+1,80,23) ;
textbackground (14) ;
gotoxy (1,2) ;
c1reol ;
gotoxy (1,3) ;
c1reol ;
gotoxy (1,4) ;
c1reol ;
end ;
end ;

matrix2[numofalternatives,numofalternatives] := . : ;

err:

(* matrix multiplication *)

matrix3 := matrix2 ;
for a3 := 1 to count do
begin
 for b3 := 1 to numofalternatives do
 begin
 for c3 := 1 to numofalternatives do
 begin
 array5[c3] := matrix2[b3,a3] ;
 for h3 := 1 to numofalternatives do
 begin
 score := 0 ;
 for k3 := 1 to numofalternatives do
 begin
 integer1 := array5[k3] * matrix2[k3,h3] ;
 score := score + integer1 ;
 end ;
 result [b3,h3] := score ;
 end ;
 end ;
 matrix2 := result ;
 end ;
 result := matrix2 ;

(* normalise vector *)

for p3 := 1 to numofalternatives do
begin
row := 0;
for f3 := 1 to numofalternatives do
row := row + result[p3,f3];
altvector5[p3] := row;
end;
row1 := 0;
for p3 := 1 to numofalternatives do
row1 := row1 + altvector5[p3];
for p3 := 1 to numofalternatives do
begin
end;
window ( x1 + 1 ,1,80,y1);
textbackground ( white );
textcolor ( black );
for a1 := 1 to numofalternatives do
begin
gotoxy(22,a1+3);
write ( altvector5[a1]:5:3 );
end;
window ( 1,y1+1,80,23 );
textbackground ( 14 );
c1rsr;
(* compute lmax and the other data *)
integer1 := 0;
for a1:= 1 to numofalternatives do
begin
score := 0;
for b1 := 1 to numofalternatives do
begin
integer1 := matrix3[a1,b1] * altvector5[b1];
score := score + integer1;
end;
lambda[a1] := score;
end;
integer1 := 0;
for a1 := 1 to numofalternatives do
begin
vector2[a1] := lambda[a1] / altvector5[a1];
integer1 := integer1 + vector2[a1];
end;
lambda := ( integer1 / numofalternatives );
if numofalternatives = 1 then
numofalternatives :=2 ;
ci := ((lambda - numofalternatives) / (numofalternatives - 1));
case numofalternatives of
  1: ri := 0.000001 ;
  2: Ri := 0.000001 ;
  3: Ri := 0.58 ;
  4: Ri := 0.90 ;
  5: Ri := 1.12 ;
  6: Ri := 1.24 ;
  7: Ri := 1.32 ;
  8: Ri := 1.41 ;
  9: Ri := 1.45 ;
 10: Ri := 1.49 ;
 11: Ri := 1.51 ;
 12: Ri := 1.48 ;
 13: Ri := 1.56 ;
 14: Ri := 1.57 ;
 15: Ri := 1.39 ;
End ;

cr := c1 / ri ;
window ( 1, y1+1, 80, 23 ) ;
altvector6 := altvector5 ;
alternativesk := alternatives ;
repeat
  count1 := 0 ;
  for a := 1 to ( numofalternatives - 1 ) do
    begin
      if altvector5[a] < altvector5[a+1] then
        begin
          exchange3[a] := altvector5[a] ;
          altvector5[a] := altvector5[a+1] ;
          altvector5[a+1] := exchange3[a] ;
          exchange4[a] := alternatives[a] ;
          alternatives[a] := alternatives[a+1] ;
          alternatives[a+1] := exchange4[a] ;
          count1 := count1 + 1 ;
        end ;
    end ;
  until count1 = 0 ;

for al := 1 to numofalternatives do
begin
  gotoxy ( ( ( 5 * al ) ) , 10 ) ;
  write ( copy( alternatives[al],1,3) ) ;
end ;

for al := 1 to numofalternatives do
begin
  gotoxy ( ( ( 5 * al ) ) , 11 ) ;
write ( altvector5[ai]\;3:2 \;); 
end;

textbackground ( green ) :
for ai := 1 to numofalternatives do 
begin 
gotoxy ( (5 + (5 * ai)) ,9 ) ; 
for b1 := 1 to round ( altvector5[ai] * 10 ) do begin 
gotoxy ( ((5 * ai)) ,(9-b1)) ; 
write ( ' ' ) ; 
end ; 
end ;

textbackground ( 14 ) ;
textcolor ( blue ) ;
gotoxy ( 36,1) ;
write ( '** lambda max = ', lambda:4:2 ) ;
gotoxy ( 36,2) ;
write ( ' consistency index = ', ci:4:2 ) ;
gotoxy (36,3) ;
write ( ' randomized index = ', ri:4:2 ) ;
gotoxy (36,4) ;
write ( ' consistency ratio = ', cr:4:2 ) ;
gotoxy (36,6) ;
write ( '** there is some statistical') ;
gotoxy( 36,7) ;
write ( ' inconsistency in your evaluation.');
Gotoxy(36,8) ;
write ( ' (study highlighted values for ') ;
gotoxy(36,9) ;
write ( ' probable inconsistent evaluation)');
textcolor ( black ) ;

altvector3 := altvector6 ;
alternatives := alternativesk ;
for p3 := 1 to numofalternatives do begin 
for f3 := 1 to numofalternatives do begin 
result [p3,f3] := 0 ;
matrix2[p3,f3] := 0 ;
end ;
end ;
gotoxy ( 36,11 ) ;
write ( 'do you want to modify the data (y/n) ? ' ) ;
Repeat
gotoxy( 75,11 ) ;
cleol ;
read ( answer ) ;
answer := stucase ( answer ) ;
until ( ( answer = 'y' ) or ( answer = 'n' ) ) ;

textbackground ( 14 ) ;
if answer = 'y' then
begin
  clrscr ;
  error := false ;
  repeat
    gotoxy ( 2,2 ) ;
cleol ;
    write ( 'name of the first alternative ? ' ) ;
    Read ( answer ) ;
    answer := stucase ( answer ) ;
    for ai := 1 to numofalternatives do
      begin
        if answer = alternatives[ai] then
          error := true ;
        end ;
  until error ;
  a := 0 ;
  repeat
    a := a +1 ;
    until answer = alternatives[a] ;
  criterial := answer ;
end ;
  error := false ;
ert4:
  gotoxy ( 2,3 ) ;
cleol ;
  write ( 'name of the second alternative ? ' ) ;
  Read ( answer ) ;
  answer := stucase ( answer ) ;
  for bi := 1 to numofalternatives do
    begin
      if answer = alternatives[bi] then
        error := true ;
    end ;
  until error ;
  if answer = criterial then
    goto ert4 ;
  b := 0 ;
  repeat
    b := b +1 ;
    until answer = alternatives[b] ;
criteria2 := answer;
window (1,1,x1,y1);
textbackground ( blue );
matrix2 := matrix3;
textcolor ( red + 16 );
gotoxy ( ( 2 + ( b * 8 ) ), 3 + a );
write ( matrix2[a,b]:4:2 );
textcolor ( black );

window ( 1 , y1+1 , 80,23 );
textbackground ( 14 );
clrscr;
repeat
    gotoxy( 1,2 );
    write ( 'is ',criteria1, ' better than ', criteria2,'? ' );
    Gotoxy ( 64,2 );
    clrreol;
    read ( answer );
    answer := stepcase ( answer );
    clrreol;
    until ( ( answer = 'y' ) or ( answer = 'n' ) );
if answer = 'y' then begin
    gotoxy ( 1,3 );
    write ( 'how many times is ',criteria1, ' better than ', criteria2 , ' ? ' );
    Gotoxy ( 1,4 );
    write ( ' ( see note below ) ' );
infol;
    repeat
        gotoxy( 64 ,3 );
        clrreol;
        read ( answer );
        val ( answer , answer3 , code );
    until ( ( code = 0 ) and ( answer3 <> 0 ) ) ;
    matrix2[a,b] := answer3;
    matrix2[b,a] := ( 1 / answer3 );
    matrix2[a,a] := 1 ;

    window (1,1,x1,y1);
textbackground ( blue );
gotoxy ( ( 2 + ( b * 8 ) ), 3 + a );
textcolor ( white );
write ( answer3:4:2 );
gotoxy( 2 + (a * 8 ),( 3 + b ) );
write ((1 / answer3):4:2 );
end;
window (1, y1+1, 80, 23);
textbackground (14);
textcolor (black);
if answer = 'n' then
begin
  gotoxy (1,3);
  write (' how many times is ', criteria2, ' better than ', criteria1, '? ');
  Gotoxy (1,4);
  write (' ( see note below ) ');
  info1;
repeat
  gotoxy (64, 3);
  clreol;
  read ( answer );
  val ( answer, answer3, code );
  until ( ( code = 0 ) and ( answer3 <> 0 ) );
matrix2[a,b] := (1/ answer3);
matrix2[c,a] := answer3;
matrix2[a,a] := 1;
window (1,1,x1,y1);
textcolor (white);
textbackground (blue);
  gotoxy ((2 + (b * 8)), 3 + a);
  write ((1/ answer3):4:2);
  gotoxy( (2 + (a * 8)), (3 + b ));
  write (answer3:4:2);
end;
window (1, y1+1, 80, 23);
textbackground (14);
gotoxy (1,2);
clreol;
gotoxy (1,3);
clreol;
gotoxy (1,4);
clreol;
gotoxy (1,24);
end;

OVERLAY PROCEDURE EVALUATE3 ( VAR ALTERNATIVES : TITLE1 ;
VAR ALTVECTOR3 :VECTORF ;
VAR W : INTEGER ;
VAR NORMVECTOR1 : VECTORG ;
var ax : integer );
LABEL
  ert, ert4 ;

CONST
  count = 3 ;

VAR
  matrix2, result, matrix3 : array [1..20,1..20] of real ;
  array5, altvector5, exchange3, altvector6 : vectorf ;
  lamda1, vector2 : vectorf ;
  exchange4, alternativesk : title1 ;
  numofalternatives, I, count1, x1, y1, a1, b1,
  a3, b3, c3, d3, h3, k3, f3, p3, l : integer ;
  score, answer3, integer1, row, row1,
  lamda, c1, ri, cr : real ;
  st : string [9] ;
  ch : char ;

PROCEDURE INFO1 ;
begin
  window (1,y1+1,80,23) ;
  textbackground (14) ;
  textcolor ( red ) ;
  gotoxy (1,8) ;
  writeln ('"note : be as accurate as possible
  any # between 0 and 10 e.g , 2.45 or 9.34 ' ) ;
  writeln (' a possible scale for inexact is : ' ) ;
  writeln ( '" 3 = weakly important than , 5 = strongly more
  important than ' ) ;
  writeln ( '" 7 = very strongly more imp. than 9 =
  absolutely more imp. than' );
  textcolor ( black ) ;
  end ;

BEGIN

  numofalternatives := w ;

  case numofalternatives of
    1,2,3,4,5,6,7 : begin
      x1 := 50 ; y1 := 12 ;
    end ;
    8,9,10,11 : begin
      x1 := 55 ; y1 := 13 ;
    end ;
    12,13,14,15 : begin
      x1 := 70 ; y1 := 15 ;
    end ;

  136
end;

numofalternatives := w;
if numofalternatives = 0 then
begin
    window (1,1,x1,y1);
    textbackground(blue);
c1rscr;

    window (x1+1,1,80,y1);
    textbackground(white);
c1rscr;

    window (1,y1+1,80,23);
    textbackground(14);
c1rscr;

    window (1,24,80,25);
    textbackground (white);
c1rscr;
    textcolor ( black );
gotoxy (2,1);
write ('step 5:direct input of alternatives weights');
altvector5 := altvector3;

    window (x1+1,1,80,y1);
    textbackground ( white );
c1rscr;
    textcolor ( 0 );
gotoxy (1,1);
write ( ' priority vector ' );
for a1 := 1 to numofalternatives do
begin
    gotoxy ( 2 ,3 +a1 );
    write (copy( alternatives[a1],1,18) );
end;
for a := 1 to numofalternatives do
begin
    gotoxy(20,3 + a );
    write ( chr ( 179 ) );
end;
for a := 1 to numofalternatives do
begin
    gotoxy(27,3 + a );
    write ( chr ( 179 ) );
end;

window (1,y1+1,80,23);
window (1,1,x1,y1);
textbackground(white);
c1rscr;

end;
ert;
info1;
textcolor ( 0 ) ;
gotoxy ( 2,2 ) ;
write ( 'enter the weights of the alternatives : ' ) ;
for a := 1 to numofalternatives do
begin
  gotoxy ( 2 , 2+a ) ;
  write ( alternatives[a], ', ' ) ;
end;
for a := 1 to numofalternatives do
begin
  repeat
    gotoxy ( length(alternatives[a])+5 , 2+a ) ;
    clrscr ;
    read ( answer ) ;
    val ( answer , answer3 , code ) ;
    until((code = 0)and(answer3 =1) and (answer3 = 1 )) ;
  altvector5[a] := answer3 ;
end ;
rowl := 0 ;
for p3 := 1 to numofalternatives do
  rowl := rowl + altvector5[p3] ;
for p3 := 1 to numofalternatives do
begin
end ;
window (51,1,80,21) ;
textbackground ( white ) ;
for a := 1 to numofalternatives do
begin
  gotoxy(22,a+3) ;
  write ( altvector5[a]:5:3 ) ;
end ;
window (1,13,80,23) ;
textbackground(14) ;
clrscr ;
altvector3 := altvector5 ;
alternativesk := alternatives ;
repeat
count1 := 0 ;
for a := 1 to ( numofalternatives-1 ) do
begin
  if altvector5[a] ( altvector5[a+1] then
begin
exchage3[a] := altvector5[a];
altvector5[a] := altvector5[a+1];
altvector5[a+1] := exchange3[a];
exchage4[a] := alternatives[a];
alternatives[a] := alternatives[a+1];
alternatives[a+1] := exchange4[a];
count1 := count1 + 1;
end;
end;
until count1 = 0;
for a := 1 to (numofalternatives) do
begin
  gotoxy((5*a),9);
  write(copy(alternatives[a],1,3));
end;
for a := 1 to (numofalternatives) do
begin
  gotoxy((5*a),10);
  write(altvector5[a]:3:2);
end;
textbackground(green);
for a := 1 to (numofalternatives) do
begin
  gotoxy((5*a),8);
  write(' ');
end;
for a := 1 to (numofalternatives) do
begin
  gotoxy((5 + (5*a)),9);
  for b1 := 1 to round(altvector5[a]*10) do
begin
    gotoxy((5*a),(9-b1));
    write(' ');
  end;
end;
gotoxy(2,11);
textbackground(14);
textcolor(blue);
write('do you want to modify the evaluation of the alternatives (y/n) ? ');
Repeat
  gotoxy(65,11);
  clrscr;
  read(answer);
  answer := tolower(answer);
until (answer = 'y' or answer = 'n');

139
window (1,13,80,23);
textbackground(14);
if answer = 'y' then
begin
  clrscr;
goto err;
end;
end;

alternatives := alternativesk;
END;

OVERLAY PROCEDURE DIRECT2A ( VAR ALTERNATIVES : TITLE:
  VAR ALTVECTORS : VECTORF ;
  VAR W : INTEGER ;
  VAR NORMVECTORS : VECTORF ;
  VAR AX : INTEGER ) ;
LABEL
  ert3,telos9x ;
CONST
  count = 3 ;
VAR
  matrix2,result,matrix3 : array [1..20,1..20] of real ;
  array5,altvector5,exchange3,altvectors : vectorf ;
  lambda1,vector2 : vectorf ;
  exchange4,alternativesk : title1 ;
  tempgrade,tempgrade1 : vectorf ;
  numofalternatives,1,count1,x1,y1,a1,b1,limit,x2,y2,
  a3,b3,c3,d3,h3,k3,f3,p3,1 : integer ;
  score,answer3,integer1,row,
  row1,lambda,c1,r1,cr,count3 : real ;
  ch : char ;
BEGIN
  numofalternatives := w ;
  case numofalternatives of
  1,2,3,4,5,6,7 : begin
    x1 := 50 ; y1 :=12 ;
  end ;
  8,9,10,11 : begin
    x1 := 55 ; y1 :=13 ;
end;
12,13,14,15 begin
    x1 := 70; y1 := 15;
end;
end;

numofalternatives := w;
if numofalternatives <> 0 then begin
    window (1,1,x1,y1);
    textbackground (blue);
    clrscr;

    window (x1+1,1,80,y1);
    textbackground (white);
    clrscr;

    window (1,y1+1,80,23);
    textbackground (14);
    clrscr;

    window (1,24,80,25);
    textbackground (white);
    clrscr;
    textcolor (black);
    gotoxy (2,1);
    write ('step 4: individual evaluation of alternatives);
    gotoxy (2,2);
    write ('method used: direct inout');

    window (1,1,x1,y1);
    textbackground (blue);
    textcolor (white);
    gotoxy (2,1);
    write ('altern. Evaluation: working area');
    for a := 1 to numofcriteria do begin
        answer := normvector1[a];
        delete ( answer, 4, length (answer) ) ;
        gotoxy (2,a+3);
        write (answer:4);
    end;
    for a := 1 to numofalternatives do begin
        answer := alternatives[a];
        delete (answer, 4, length (answer)) ;
        gotoxy (9 + (5*(a-1)),3);
        write (answer);
    end;
end;

window (x1+1,1,80,y1); 141
textbackground ( white ) ;
textcolor( 0 ) ;
gofoxy (1,1) ;

write ( ' priority vector ' ) ;
for a1 := 1 to numofalternatives do begin
gofoxy ( 2 ,3 +a1 ) ;
write ( copy( alternatives[a1],1,18) ) ;
end ;
for a := 1 to numofalternatives do begin
gofoxy(20,3 + a ) ;
write ( chr ( 179 ) ) ;
end ;
for a := 1 to numofalternatives do begin
gofoxy(27,3 + a ) ;
write ( chr ( 179 ) ) ;
end ;

window (1,y1+1,80,23);
textbackground(14) ;
clrscr ;
for a := 1 to numofcriteria do begin
if norm.specialized Then begin
if specfile2.normindex[a] () namex Then
go to telos9x ;
end ;
gofoxy ( 2,2 ) ;
write ( '*** evaluate alternative according to criteria',normvector[a],', ' ) ;
for b := 1 to numofalternatives do begin
window (1,y1+1,80,23);
textbackground(14) ;
textcolor ( black ) ;
gofoxy ( 5,b+3 ) ;
write ( b ,'- for alternative ', alternatives[b] ,'
 any value between 0 and 10 ? ' ) ;
x2 := 76 ; y2 := b + 3 ; count3 := 0 ; limit := 10 :
checknumber ( answer,x1,y1,limit,count3 ) ;
tempgrade1[b] := count3 ;
window (1,1,x1,y1) ;
textbackground ( blue ) ;
textcolor ( white ) ;
gofoxy ( 9 + ( 5*(b-1)),3+a ) ;
write ( tempgrade1[b]:3:2 ) ;
end ;
tempgrade := tempgrade1;
(* normalize vector *)
row := 0;
for p3 := 1 to numofalternatives do
  row := row + tempgrade[p3];
for p3 := 1 to numofalternatives do
window (x1 + 1, 1, 80, y1);
textbackground (white);
textcolor (black);
for al := 1 to numofalternatives do
  begin
    gotoxy (22, a1 + 3);
    write (tempgrade[al]:3:2);
  end;
window (1, y1+1, 80, 23);
textbackground (14);
clear ;
for al := 1 to numofalternatives do
  begin
    gotoxy ((5 * al), 10);
    write (copy(alternatives[al],1,3));
  end;
for al := 1 to numofalternatives do
  begin
    gotoxy ((5 * al), 11);
    write (tempgrade[al]:3:2);
  end;
textbackground (green);
for al := 1 to numofalternatives do
  begin
    gotoxy ((5 + (5 * al)), 9);
    for bl := 1 to round (tempgrade[al] * 10) do
      begin
        gotoxy ((5 * al), (9 - bl));
        write ('');
      end;
  end;
textbackground (14);
textcolor (black);
gotoxy (36,11);
write ("do you want to modify the weights (y/n)? ");
repeat
  gotoxy (78,11);
clear ;
  read (answer);
  answer := stupcase(answer);
until ((answer = 'Y') or (answer = 'n'));
if answer = 'y' then
begin
  clrscr;
  error := false;
  repeat
    gotoxy (2,2);
    clrscr;
    write ('name of the alternative ? ');
    read (answer);

    answer := stupcase (answer);
    for al := 1 to numofalternatives do
begin
  if answer = alternatives[al] then
    error := true;
end;
until error;
al := 0;
repeat
  al := al + 1
until answer = alternatives[al];
clrscr;

Window (1,1,x1,y1);
textbackground (blue);
textcolor (red);
gotoxy (9 + (5*(al-1)),3+a);
write (tempgrade[al]:3:2);

window (1,y1+1,80,23);
textbackground (14); textcolor (black);
gotoxy (2,2);
write ('for alternative ', alternatives[al],
' any value between 0 and 10 ?');
x2 := 76; y2 := b + 3; count3 := 0; limit := 10;

checknumber (answer,x1,y1,limit,count3);
tempgrade[al] := count3;

window (1,1,x1,yi);
textbackground (blue);
textcolor (white);
gotoxy (9 + (5*(al-1)),3+a);
write (tempgrade[al]:3:2);

goto err3;
end;

for al := 1 to numofalternatives do
  altmatrix[a,al] := tempgrade[al];
window (1,y1+1,80,23);
textbackground ( 14 );
textcolor ( black );
clrscr ;
telos9x:
   end ;
   end ;
END ;

INCLUDE FILE 4-2

OVERLAY PROCEDURE ELECTRE ;

TYPE
   scale  = array[1..4] Of name :;
   ind    = array[1..9] Of integer :;
   aray2  = array[1..5,1..3] Of real ;

VAR
   f1,f2,a,b,c,limit : integer ;
   sum,result,pfactor,
   qfactor,max,min,count3 : real ;
   discordance,concordance,
   matrixdance,matrixcon : matrix20 ;
   outranking : array9 ;
   criteria : vector ;
   critvalue : vectorn ;
   alter : title1 ;
   grading : aray1 ;
   gradingweight : aray2 ;
   index,cindex,dindex : ind ;
   st1 : name ;

PROCEDURE WRITEWORKSHEET ;
BEGIN
   window ( 1,1,41,13) ;
   textbackground ( blue );
   textcolor ( white );
   gotoxy ( 2,1 );
   write ( ' altern.Evaluation : working area ' ) ;
   for a := 1 to numofcriteria do
      begin
         answer := criteria[a] ;
         delete ( answer,4,length(answer) ) ;
         gotoxy ( 5 + ( 4*(a-1)),3 ) ;
         write ( answer ) ;
      end 
   end ;

145
write ( answer:4 ) ;
end ;

for a := 1 to numofalternatives do
begin
gotoxy ( 2,a+3 ) ;
answer := alter[a] ;
delete ( answer,4,length( answer)) ;
write ( answer:3 ) ;
end ;
END ;

PROCEDURE GRADES ;
VAR
base, step : real ;

BEGIN
for a := 1 to numofcriteria do
begin
base := critvalue[a] * 100 ;
step := base / 4 ;
for b := 1 to 5 do
gradingweight[b,a] := base - ( step * ( b -1) ) ;
end ;
END ;

PROCEDURE WRITEGRADING ;
BEGIN
window (41,1,80,12) ;
textbackground ( white );
textcolor ( black ) ;
gotoxy ( 2,1 ) ;
write ( 'grading scale ' ) ;

for a := 1 to numofcriteria do
begin
gotoxy ( 7 + ( 4*(a-1)),3 ) ;
answer := criteria[a] ;
delete ( answer,4,length( answer) ) ;
write ( answer:4 ) ;
end ;

146
gotoxy(4 + (4*a),4);
write(round(critvalue[a]*100));
end;
textcolor(black);
gotoxy(1,5);
write('excellent');
gotoxy(1,7);
write('good');
gotoxy(1,8);
write('average');
gotoxy(1,9);
write('fair');
gotoxy(1,10);
write('weak');
grades;
for a := 1 to numofcriteria do
begin
  for b := 1 to 3 do
  begin
    gotoxy(4 + (4*a), b + 5);
    write(round(gradingweight[b,a]));
  end;
end;
END;

PROCEDURE GRADEALTERNATIVES;
LABEL
jmp3;
BEGIN
  window (1,13,30,24);
textbackground (14);
textcolor(black);
if norm.specialized Then
  grading := specfile2.grading;
for a := 1 to numofalternatives do
begin
  gotoxy(2,2);
  write('** evaluate alternative ', alter[a], ' : ');
  for b := 1 to numofcriteria do
  begin
    if norm.specialized Then
    begin
      if specfile2.normindex[b] = name Then
        goto jmp3;
      end;
    write(b, ' - for criterion ', criteria[b], ' any value
  end;
147
between 0 and round(gradingweight[1, b]),'??'

X1 := 76;
y1 := b + 3;
count3 := 0;
limit := round(gradingweight[1, b]);
checknumber (answer, x1, y1, limit, count3);
grading[a, b] := count3;
window (1, 1, 40, 12);
textbackground (blue);
gotoxy(2 + (4*b), a+3);
write (round(grading[a, b]));
window (1, 13, 80, 23);
textbackground (14);
jmp3:
   end;
clrscr;
end;
END;

PROCEDURE FACTORS;
BEGIN
window (1, 13, 80, 23);
textbackground (14);
clrscr;
textcolor (black);
gotoxy (2, 4);
write ('** concordance threshold (p) [0 - 100] : ');
gotoxy (2, 5);
write ('nb .. becomes severe as it approaches 100? '):
X1 := 70; y1 := 5; count3 := 0; limit := 100;
checknumber (answer, x1, y1, limit, count3);
pfactor := count3;
gotoxy (2, 6);
write (** discordance threshold (q) [0 - 100] : ');
gotoxy (2, 7);
write ('nb .. becomes severe as it approaches 100? '):
X1 := 70; y1 := 8;
count3 := 0; limit := 100;
checknumber (answer, x1, y1, limit, count3);
qfactor := count3;
if ((specfile2.pfactor) = 0) And
   (specfile2.Pfactor (pfactor)) then
   pfactor := specfile2.pfactor;
if ((specfile2.qfactor) = 0) And
   (specfile2.Qfactor qfactor) then
   qfactor := specfile2.qfactor;
Window (41, 1, 80, 12);
textbackground (white);
textcolor (black);
gotoxy (2, 11);

148
write ('p = ', pfactor:3:2,' % q = ', qfactor:3:2,' % ');
if norm.specialized Then
begin
  specfile2.pfactor := pfactor;
  specfile2.qfactor := qfactor;
  specfile2.grading := grading;
  A := \emptyset;
repeat
  a := a + 1;
until ( namex = norm.Usersnames[a] ) ;
  specfile2.Finalindex1[a] := true;
  writespecfile;
end;
END;

PROCEDURE COMPUTE1 :
BEGIN
  for c := 1 to numofcriteria do
  begin
    if ( grading[a,c] = grading[b,c] ) then
      sum := sum + critvalue[c] ;
    end;
    if ( a <> b ) then
      concordance[a,b] := sum * 100;
    else
      concordance[a,b] := 1;
  END;

PROCEDURE COMPUTE2 :
BEGIN
  for c := 1 to numofcriteria do
  begin
    if ( grading[b,c] > grading[a,c] ) then
      sum := grading[b,c] - grading[a,c] ;
    if ( result <= sum ) then
      result := sum;
    end;
    if ( a <> b ) then
      discordance[a,b] := result / critvalue[1];
    else
      discordance[a,b] := 1;
  END;

PROCEDURE COMPUTECONC :
BEGIN
  for a := 1 to numofalternatives do
begin
  for b := 1 to numofalternatives do
  begin
    sum := 0;
    compute1;
  end;
end;
END;

PROCEDURE FINTINDEX1;
BEGIN
  for a := 1 to numofalternatives do
  begin
    c := 0;
    for b := 1 to numofalternatives do
    begin
      if concordance[a][b] >= pfactor then
        c := c + 1;
      cindex[a] := c;
    end;
    for a := 1 to numofalternatives do
      f1 := f1 + cindex[a];
  END;

PROCEDURE COMPUTEDISCONC;
BEGIN
  for a := 1 to numofalternatives do
  begin
    for b := 1 to numofalternatives do
    begin
      sum := 0;
      result := 0;
      compute2;
    end;
  END;

PROCEDURE FINTINDEX2;
BEGIN
  for a := 1 to numofalternatives do
  begin
    c := 0;
    for b := 1 to numofalternatives do
begin
  if discordance[a, b] (= qfactor then
    c := c + 1;
    dindex[a] := c - 1;
  end;
end;
f2 := 0;
for a := 1 to numofalternatives do
  f2 := f2 + dindex[a];
END;

PROCEDURE WRITEALT (var st1: name; var matrixcon : matrix20; var index : ind);
BEGIN
  for a := 1 to numofalternatives do
  begin
    answer := alter[a];
    delete ( answer, 4, length( answer) ) ;
    gotoxy( 2, a + 3 );
    write ( answer: 4 ) ;
  end;
  for a := 1 to numofalternatives do
  begin
    answer := alter[a];
    delete ( answer, 4, length( answer) ) ;
    gotoxy ( 5 + ( a * 5 ), 3 ) ;
    write ( answer: 3 ) ;
    textcolor ( red ) ;
    gotoxy ( 5 + (( a+1) * 5 ), 3 ) ;
    write ( st1 ) ;
    textcolor ( black ) ;
    for a := 1 to numofalternatives do
      begin
        for b := 1 to numofalternatives do
          begin
            gotoxy ( 5 + ( b * 5 ), a + 3 );
            if ( a = b ) then
              write ( ' - ' )
            else
              write ( round(matrixcon[a, b]) ) ;
          end;
      end;
    textcolor ( red ) ;
    for c := 1 to numofalternatives do
begin
gotoxy ( 5 + ( (b+1) * 5 ) ,c + 3 ) ;
write ( index[c] ) ;
end ;
textcolor ( black ) ;
END ;

PROCEDURE CONFINT ;
BEGIN

window ( 1,13,80,23 ) ;
textbackground ( 14 ) ;
textcolor ( blue ) ;
c1rscr ;
gotoxy ( 2,2 ) ;
write ('concordance matrix ' ) ;
computeconc :
'sintindex1 ;
textcolor ( black ) ;
st1 := '#ci' ;
matrixcon := concordance ;
index := cindex ;
writealt ( st1 ,matrixcon,index ) ;
textcolor ( blue ) ;
gotoxy ( 38,2 ) ;
write ('** a concordance index indicates to ' ) ;
gotoxy ( 38,3 ) ;
write (' what extent an option is better than ' ) ;
gotoxy ( 38,4 ) ;
write (' another in terms of criteria weights ' ) ;
gotoxy ( 38,5 ) ;
write ('** the index varies between [ 0 - 100 ] ' ) ;
gotoxy ( 38,6 ) ;
write (' the higher the better . ' ) ;
Gotoxy ( 38,7 ) ;
write (' ','fi,' indexes are ) = ',round(pfactor)) ;
gotoxy ( 38,8 ) ;
write ('** column #ci indicates the # of indexes ' ) ;
gotoxy ( 38,9 ) ;
write (' satisfying p for each option ' ) ;
gotoxy ( 2,10 ) ;
write ('hit any key to continue ' ) ;
read ( kbd,ch ) ;
END ;
PROCEDURE DISFINT;
BEGIN
  window (1,13,80,23);
textbackground (14);
crscr;
textcolor (blue);
gotoxy(2,2);
write('discordance matrix');
computedisconc;

fintindex2;
textcolor (black);

stl := '#di';
matrixcon := discordance;
index := dindex;
writealt (stl,matrixcon,index);
textcolor (blue);
gotoxy(40,2);
write('** a discordance index indicates to ');
gotoxy(40,3);
write('what extent an option contains a bad ');
gotoxy(40,4);
write('element that makes it un-satisfactory ');
gotoxy(40,5);
write('** the index varies between [0-100] ');
gotoxy(40,6);
write('the lower the better. ');
Gotoxy(40,7);
write(',f2,', indexes are (= ',qfactor:3:2 );
gotoxy(40,8);
write('** column #ci indicates the # of indexes ');
gotoxy(40,9);
write('satisfying q for each option ');
gotoxy(2,10);
write('hit any key to continue ');
read(kbd,ch);
END;

PROCEDURE COMPUTEOUTRANKING;
BEGIN
  for a := 1 to numofalternatives do
  begin
    for b := 1 to numofalternatives do
    begin
      if ((concordance[a,b])= pfactor) and
          (discordance[a,b]< qfactor) then
        outranking[a,b] := '*'
    else

    end
  end
END;
outranking[\, a, b] := \, '-' \; ;
end;
end;
for a := 1 to numofalternatives do
  outranking[a, a] := \, '-' \; ;
end;

PROCEDURE OUTFINT ;
VAR
  ans : name ;
BEGIN
  window ( 1, 13, 80, 23 ) ;
  textbackground ( 14 ) ;
  textcolor ( blue ) ;
crscr ;
gotoxy ( 2, 2 ) ;
write (' outranking matrix ' ) ;
compute outranking ;
textcolor ( black ) ;
for a := 1 to numofalternatives do
  begin
    ans := alter[a] ;
    delete ( ans, 4, length( ans ) ) ;
gotoxy ( 2, a + 3 ) ;
write ( ans:4 ) ;
  end;
for a := 1 to numofalternatives do
  begin
    ans := alter[a] ;
    delete ( ans, 4, length( ans ) ) ;
gotoxy ( 5 + ( a * 5 ), 3 ) ;
write ( ans:3 ) ;
  end;
for a := 1 to numofalternatives do
  begin
    for b := 1 to numofalternatives do
      begin
        gotoxy ( 5 + ( b * 5 ), a + 3 ) ;
write ( outranking[\, a, b] ) ;
      end;
  end;
textcolor ( blue ) ;
gotoxy ( 38, 2 ) ;
write ( '=' an outranking relation * is the ' ) ;
gotoxy ( 38, 3 ) ;
write ( ' one that satisfies both concordance ' ) ;
gotoxy ( 38, 4 ) ;
write ( ' and discordance requirements. ' ) ;
Gotoxy ( 38, 5 ) ;
write ('** an - indicates that there is ');
gotoxy (38,6);
write ( ' no outranking relations. ');
gotoxy (2,10);
write ( 'hit any key to continue' );
read ( kbd, ch );

BEGIN (* main *)

window (1,1,40,12);
textbackground ( blue );
clrscr ;

window (41,1,80,12);
textbackground ( white );
clrscr ;

window (1,1,30,23);
textbackground ( 16 );
clrscr ;

window (1,24,80,25);
textbackground ( white );
clrscr;
textcolor ( black );
gotoxy (2,1);
write ( 'step 4 : evaluation of alternatives' );
gotoxy (2,2);
write ( 'method used : electre' );

if ( not norm.Specialized ) then
begin
alter := problem.alternatives;
numofalternatives := problem.numofalternatives;
criteria := solution.normvector1;
critvalue := solution.normvector2;
end
else
begin
alter := problem.alternatives;
numofalternatives := problem.numofalternatives;
criteria := Specfile2.normvector1;
critvalue := Specfile2.normvector2;
end;

write wowsheet ;
write grading ;
gradealternatives;
factors;
if norm.specialized Then
begin
  b := 0;
  for a := 1 to 3 do
    begin
      if specfile2.finalindex1[a] Then
        b := b + 1;
    end;
  if ( b = norm.Numofusers ) then
    specfile2.Electre.Status := true
  else
    specfile2.Electre.Status := false;
end:

if ( not norm.Specialized ) or
  ( norm.Specialized and specfile2.Electre.Status )
then begin
  repeat
    window (1,13,80,23);
    textbackground ( 14 );
    clrscr;
    textcolor( black );
    gotoxy( 2,2 );
    write ( 'menu ' );
    gotoxy(2,4);
    write ( '1. Concordance matrix ' );
    gotoxy(2,5);
    write ( '2. Discordance matrix ' );
    gotoxy(2,6);
    write ( '3. Outranking matrix ' );
    gotoxy(2,7);
    write ( '4. Modify thresholds ' );
    gotoxy(2,8);
    write ( '5. Exit electre ' );
    gotoxy( 2,10 );
    write ( 'selection (1-5)? ' );
  Repeat
    gotoxy( 30,10);
    clrscr;
    read ( answer );
  until ((answer = '1') or (answer = '2') or (answer = '3') or (answer = '4') or (answer = '5'));
  if answer = '1' then
    confint;
  if answer = '2' then
isfint;
if answer = '3' then
begin
  computeconc;
  computedisconc;
  outfint;
end;

if answer = '4' then
  factors;
until (answer = '5');

if (not norm.Specialized) then
begin
  solution.Electre.Outranking := outranking;
  writesolutionfile;
end
else
begin
  specfile2.Electre.Outranking := outranking;
  writespecfile;
end;
end;
END;

INCLUDE FILE STEP6.

OVERLAY PROCEDURE GDSS;

LABEL
telos1;

TYPE

names = name;
altnames1 = array[1..20] of names;
altvector5 = array[1..6, 1..20] of real;
ordinal2 = array [1..20] of integer;
ordinal3 = array [1..6] of ordinal2;

VAR

a, b, c, numofalternatives, resultx, numofusers, xxx, countahp, countelectre, suma, countl2, f1 : integer;
PROCEDURE COMPUTE1;
BEGIN
  for b := 1 to numofalternatives do
  begin
    suma := 0;
    for c := 1 to numofusers do
    begin
      ordinal1 := ordinal[c];
      suma := suma + ordinal1[b];
    end;
    ar1[b] := suma;
  end;
  for a := 1 to numofalternatives do
  begin
    gotoxy ( 6, a+5 );
    write ( ar1[a] );
  end;
END;

PROCEDURE COMPUTE2;
BEGIN
  for b := 1 to numofalternatives do
  begin
    resultx1 := 0;
    for c := 1 to numofusers do
    begin
      altvector8 := altvector6[c];
      resultx1 := resultx1 + altvector8[b];
    end;
    ar2[b] := ( resultx1 / numofusers );
  end;
  for a := 1 to numofalternatives do
PROCEDURE COMPUTE3;
BEGIN
  for b := 1 to numofalternatives do
    begin
      resultx1 := 1;
      for c := 1 to numofusers do
        begin
          altvector8 := altvector6[c];
          resultx1 := resultx1 * altvector8[b];
        end;
      resultx2 := ln ( resultx1 );
      ar3[b] := exp ( 1 / numofusers ) * resultx2;
    end;
  for a := 1 to numofalternatives do
    begin
      gotoxy ( 18, a+5 );
      write ( ar3[a]:3:2 );
    end;
END;

PROCEDURE COMPUTE4;
BEGIN
  for b := 1 to numofalternatives do
    begin
      suma := 0;
      for c := 1 to numofusers do
        begin
          ordinal1 := ordinal[c];
          suma := suma + ( numofalternatives - ordinal1[b] );
        end;
      ar4[b] := suma;
    end;
  for a := 1 to numofalternatives do
    begin
      gotoxy ( 24, a+5 );
      write ( ar4[a] );
    end;
END;

PROCEDURE COMPUTEORDINAL;
BEGIN
  for b := 1 to numofalternatives do
    begin
      suma := 1;
      for c := 1 to ( numofalternatives ) do
begin

if altvector8[b] < altvector8[c] then
    suma := suma + 1;
end;
ordinali[b] := suma;
end;
END;

PROCEDURE COMPUTEINDIVIDUALVECTOR;
BEGIN
for a1 := 1 to numofalternatives do
begin
    suma := 0;
for b1 := 1 to numofalternatives do
begin
    if (solution.Electre.Outranking[a1,b1] = '+' ) then
        suma := suma + 1;
    end;
    individualvector1[a1] := suma;
end;
END;

PROCEDURE COMPUTE1;
BEGIN
for a := 1 to numofalternatives do
begin
    suma := 0;
for b := 1 to numofusers do
begin
    individualvector := individualvector[b] :=
    suma := suma + individualvector[a];
end;
go toxy (9,5+a);
write (suma);
end;
END;

PROCEDURE COMPUTE2;
BEGIN
for a := 1 to numofalternatives do
begin
    suma := 0;
for b := 1 to numofusers do
begin
    individualordinal := individualordinal[b] :=
    suma := suma + individualordinal[a];
end;
go toxy (15,5+a);
write (suma);
end;
END;
PROCEDURE WIN1 ;
BEGIN
if norm.Broadcasting then
begin
for a := 1 to numofusers do
begin
  gotoxy ( 6 * a, 4 ) ;
  write ( usersnames[a]:4 ) ;
end ;
gofoxy ( 2,5 ) ;
textcolor ( red ) ;
write ( 'weight:' ) ;
For a := 1 to numofusers do
begin
  gotoxy ( (6*a)+2,5 ) ;
  write ( weight[a]:3:2 ) ;
end ;
textcolor ( blue ) ;
for a := 1 to numofalternatives do
begin
  gotoxy ( 2,a+5 ) ;
  write ( copy(altnames6[a],1,3) ) ;
end ;
for a := 1 to numofusers do
begin
  altvector8 := altvector6[a] ;
  for b := 1 to numofalternatives do
  begin
    gotoxy ( (6 * a)+2 , b + 5 ) ;
    write ( altvector8[b]:3:2 ) ;
  end ;
end ;
else
begin
for a := 1 to numofusers do
begin
  if ( pruser1 = usersnames[a] ) then
  begin
    gotoxy ( 6 , 4 ) ;
    write ( usersnames[a]:4 ) ;
    xxx := a ;
  end ;
for a := 1 to numofalternatives do
begin
  gotoxy ( 2,a+5 ) ;
  write ( copy( altnames6[a],1,3) ) ;
end ;
altvector8 := altvector6[xxx] ;
for b := 1 to numofalternatives do
begin
  gotoxy (8, b + 5);
  write (altvector8[b]:3:2);
end;
end;
END;

PROCEDURE WIN2;
BEGIN
  window (26,1,50,16);
  textbackground (14);
  clrscr;
  textcolor (black);
  gotoxy (2,2);
  write (' ordinal ranking ');
  if norm.Broadcasting then
    begin
      for a := 1 to numofusers do
        begin
          gotoxy ((6*a)-2,+);
          write (usersnames[a]:4);
        end;
    end;
  else
    begin
      gotoxy (6,4);
      write (usersnames[xxx]);
    end;
    for a := 1 to numofusers do
      begin
        altvector8 := altvector6[a];
        for b := 1 to numofalternatives do
          begin
            suma := 1;
            for c := 1 to (numofalternatives) do
              begin
                if altvector8[b] (altvector8[c]) then
                  suma := suma + 1;
              end;
            ordinal[b] := suma;
          end;
        ordinal[a] := ordinal;
      end;
    if norm.Broadcasting then
      begin
        for a := 1 to numofusers do
          begin
            ordinal := ordinal[a];
            for b := 1 to numofalternatives do
              begin
                
              end;
          end;
      end;
END;
gotoxy ( 6 * a , b + 5 ) ;
write ( ordinal1[b] ) ;
end ;
end;

else
begin
    ordinal1 := ordinal[xxx] ;
    for b := 1 to numofalternatives do
    begin
        gotoxy ( 6 , b + 5 ) ;
        write ( ordinal1[b] ) ;
        end ;
    end;
END;

PROCEDURE WIN3 : BEGIN
    window (51,1,30,15) ;
textbackground ( white ) ;
cls cr ;
textcolor ( black ) ;
gotoxy ( 1,2 ) ;
write ( ' group results ' ) ;
for a := 1 to 4 do
begin
    gotoxy ( ( 6*a),4 ) ;
    write ( 'r',a ) ;
end ;
for a := 1 to numofalternatives do
begin
    gotoxy ( 2,a+5 ) ;
    write ( copy (alternames6[a],1,3) ) ;
end ;
if norm.Aggregation then
begin
    computer1 ;
    computer2 ;
    computer3 ;
    computer4 ;
end ;
else
begin
    f1 := 0 ;
    repeat
    f1 := f1 + 1 ;
case norm.Aggregationname[f1] of
    '2' : computer2 ;
    '1' : computer1 ;
    '4' : computer4 ;
end ;
163
PROCEDURE WIN4;
BEGIN
  if norm.Broadcasting then
  begin
    for a := 1 to numofusers do
      begin
        gotoxy (6 * a, 4);
        write (usersnames[a]:4);
      end;
    for a := 1 to numofalternatives do
      begin
        gotoxy (2, a+5);
        delete(altnames6[a], 4, length(altnames6[a]));
        write(altnames6[a]);
      end;
    for a := 1 to numofusers do
      begin
        individualvector1 := individualvector[a];
        for b := 1 to numofalternatives do
          begin
            gotoxy((6 * a)+2, b+5);
            write(individualvector1[b]);
          end;
      end;
  end;
else
  begin
    for a := 1 to numofusers do
      begin
        if (pruser1 = usersnames[a]) then
          begin
            gotoxy(6, 4);
            write(usersnames[a]:4);
            xxx := a;
          end;
      end;
    for a := 1 to numofalternatives do
      begin
        gotoxy(2, a+5);
        write(altnames6[a]);
      end;
    individualvector1 := individualvector[xxx];
for b := 1 to numofalternatives do
  begin
    gotoxy ( 6 , b + 5 ) ;
    write ( individualvector1[b] ) ;
  end ;
end ;
END ;

PROCEDURE WIN5 ;
BEGIN
  window ( 26 , 1 , 50 , 16 ) ;
textbackground ( 14 ) ;
c1rscr ;
textcolor ( black ) ;
gotoxy ( 2 , 2 ) ;
write ( ' ordinal rankink ' ) ;
if norm. Broadcasting then
  begin
    for a := 1 to numofusers do
      begin
        gotoxy ( 6 * a , 4 ) ;
        delete ( usersnames[a] , 4 , length ( usersnames[a] ) ) ;
        write ( usersnames[a] ) ;
      end ;
  end
else
  begin
    gotoxy ( 6 , 4 ) ;
    delete ( usersnames[xxx] , 4 , length ( usersnames[xxx] ) ) ;
    write ( usersnames[xxx] ) ;
  end
for a := 1 to numofusers do
  begin
    individualvector1 := individualvector[a] ;
    for b := 1 to numofalternatives do
      begin
        suma := 1 ;
        for c := 1 to ( numofalternatives ) do
          begin
            if individualvector1[b] ( individualvector1[c] then
              suma := suma + 1 ;
            end ;
            individualordinal1[b] := suma ;
          end ;
        individualordinal1[a] := individualordinal1 ;
        end
        if norm. Broadcasting then

165
begin 
for a := 1 to numofusers do 
begin 
individualordinal1 := individualordinal[a] ; 
for b := 1 to numofalternatives do 
begin 
gotoxy ( 6 * a , b + 5 ) ; 
write ( individualordinal1[b] ) ; 
end ; 
end ; 
end 
else 
begin 
individualordinal1 := individualordinal[xxx] ; 
for b := 1 to numofalternatives do 
begin 
gotoxy ( 6 * a , b + 5 ) ; 
write ( individualordinal1[b] ) ; 
end ; 
end ;

PROCEDURE WIN6 ;
BEGIN 
window (51,1,80,16 ) ;
textbackground ( white ) ;
c1rsr ;
textcolor ( black ) ;
gotoxy ( 1,2 ) ;
write ( ' group results ' ) ;
gotoxy ( 9,4) ;
write ( ''r4'' ) ;
gotoxy ( 15,4) ;
write ( ''r1'' ) ;
for a := 1 to numofalternatives do 
begin 
gotoxy ( 2,a+5 ) ;
write ( copy ( altnames6[a],1,3 ) ) ;
end ;
if norm.Aggregation then 
begin 
computeX1 ;
computeX2 ;
end 
else 
begin 
for fl := 1 to 4 do 
begin 
case norm.Aggregationname[fl] of
'1': compute1;
'4': compute2;
end;
end;
end:
END;

PROCEDURE WIN7;
BEGIN
  gotoxy (2,2);
  write (' ordinal ranking ');
  if norm.Broadcasting then
    begin
      for a := 1 to numofusers do
        begin
          gotoxy ((5*a)+2,4);
          write (copy(usersnames[a],1,3));
        end;
      for a := 1 to numofalternatives do
        begin
          gotoxy (2,a+5);
          write (copy(altnames6[a],1,3));
        end;
    end
  else
    begin
      gotoxy (6,4);
      write (usersnames[xxx]);
    end;
  if norm.Broadcasting then
    begin
      for a := 1 to numofusers do
        begin
          ordinal := ordinal[a];
          for b := 1 to numofalternatives do
            begin
              gotoxy ((6*a)+2,b+5);
              write (ordinal[b]);
            end;
        end;
    end
  else
    begin
      ordinal := ordinal[xxx];
      for b := 1 to numofalternatives do
        begin
          gotoxy (6,b+5);
        end;
    end;
END;
PROCEDURE WIN8;
BEGIN
  window (51,1,80,16);
textbackground (white);
cirscr;
textcolor (black);
gotoxy (1,2);
write ('group results');
for a := 1 to 4 do
begin
  gotoxy (6*a,4);
  write ('r',a);
end;
for a := 1 to numofalternatives do
begin
  gotoxy (2,a+5);
  write (copy{alternames}@a,1,3);
end;
if norm.Aggregation then
begin
  computer1;
  computer4;
end
else
begin
  for f1 := 1 to 4 do
  begin
  case norm.Aggregation@f1 of
    '1': computer1;
    '4': computer4;
    end;
  end;
end;
END;

PROCEDURE WINDOW1;
BEGIN
  window (1,17,80,23);
textbackground (blue);
cirscr;
textcolor (white);
gotoxy (2,2);
write ('r1 : sum of ranks');
gotoxy (2,3);
write ('r2 : additive ranking');
gotoxy (40,2);
write ('r3 : multiplicative ranking');
gotoxy (40,3);
write ('r4 : sum of outranking relations');
gotoxy (2,5);
BEGIN (* MAIN *)
pruser1 := namesx;
umofalternatives := problemA.Numofalternatives;
umofusers := norm.Numofusers;
usersnames := norm.Usersnames;
weight := norm.Weight;
altvector6[a] := solution.Ahp.Altvector1;
altnames6 := solution.Alternatives;
b := Ø;
for a := 1 to numofusers do
begin
userson := norm.Usersnames[a];
filnamel := concat ( problame, '.', Userx );

if exist ( filnamel ) then
b := b + 1;
end;
countahp := Ø;
countelectre := Ø;

if b < numofusers then
begin
cirscr;
writeln ('the solutions are not completed' )
end;
if b = numofusers then
begin
for a := 1 to numofusers do
begin
userson := usersnames[a];
filnamel := concat ( problame, '.', Userx );
pruser := filnamel;
if exist ( pruser ) then
begin
readsolutionfile;
if ( solution.Ahp.Status ) then
begin
countahp := countahp + 1;
altpvector6[a] := solution.Ahp.Altvector1;
altnames6 := solution.alternatives;
Altvector8 := altvector6[a];
end;
if ( solution.Electre.Status ) then
begin

169
c.rntelectre := countelectre + 1;
computeindirecvector;
individualvector[a] := individualvector1;
altnames6 := solution.Alternatives;
end;
end;
end;

if countahp = numofusers then
begin
for a := 1 to numofusers do
begin
altvector8 := altvector6[a];
for b := 1 to numofalternatives do
altvector8[b] := altvector8[b] * weight[a];
altvector6[a] := altvector8;
end;
window (1,1,25,15);
textbackground ( 14);
crscre;
window (26,1,50,16);
textbackground ( 14);
crscr;
window (51,1,80,16);
textbackground ( white);
crscr;
window (1,17,80,23);
textbackground ( blue);
crscr;
window (1,24,80,25);
textbackground ( white);
crscr;
textcolor ( black );
gotoxy ( 2,1);
write ( ' step 5 : computation of group decision ' );
gotoxy ( 2,2);
write (' all the solutions have computed with ahp ' );
window (1,1,25,16);
textbackground ( 14);
textcolor ( blue );
crscr;
gotoxy ( 1,2);
write (' alt. Cardinal rankings ' );
win1;
win2;
win3;
window1;
end;
if countelectre = numofusers then
begin
for a := 1 to numofusers do
begin
altvector8 := altvector6[a] ;
for b := 1 to numofalternatives do
   altvector8[b] := altvector8[b] * weight[a] ;
altvector6[a] := altvector8 ;
end ;
window (1,1,25,16) ;
textbackground (14) ;
c1rscr ;
window (26,1,50,16) ;
textbackground (14) ;
c1rscr ;
window (51,1,90,16) ;
textbackground (white) ;
c1rscr ;
window (1,17,80,23) ;
textbackground (blue) ;
c1rscr ;
window (1,24,80,25) ;
textbackground (white) ;
c1rscr ;
textcolor (black) ;
gotoxy (2,1) ;
write ('step 5 : computation of group decision') ;
gotoxy (2,2) ;
write ('all the solutions have computed with electre') ;
window (1,1,25,16) ;
textbackground (14) ;
textcolor (blue) ;
c1rscr ;
gotoxy (1,2) ;
write ('alt. Individual rankings') ;
win4 ;
win5 ;
win6 ;
window1 ;
end ;
count12 := 0 ;
for a := 1 to numofusers do
begin
userx := usersnames[a] ;
filename1 := concat ( problname,'.Userx
' ) ;
pruser := filename1 ;
read solutionfile ;
if solution.Ahp.Status then
begin
    indexm[a] := 'a'' ;
count12 := count12 + 1 ;
end
else
begin
    if solution.Electre.Status then
    begin
        indexm[a] := 'e'' ;
count12 := count12 + 1 ;
    end
else
    indexm[a] := 'n'' ;
end
end
if count12 < numofusers then
begin
    window ( 1,1,80,25 ) ;
textcolor ( 14 ) ;
clrscr ;
gotoxy ( 2,3 ) ;
textcolor ( blue ) ;
write ('the solutions are not completed ');

gotoxy ( 2,4 ) ;
write ('hit any key to return to main menu ')
read ( kbd,ch ) ;
goto telos1 ;
end
else
begin
    for a:= 1 to numofusers do 
    begin
        if indexm[a] = 'a' then
        begin
            altvector8 := altvector8[a] ;
            for b := 1 to numofalternatives do
            begin
                suma := 1 ;
                for c := 1 to ( numofalternatives ) do
                begin
                    if altvector8[b] < altvector8[c] then
                    suma := suma + 1 ;
                end
                ordinal1[b] := suma ;
            end
            ordinal[a] := ordinal1 ;
        end
end
else
begin
  individualvector1 := individualvector[a] ;
  for b := 1 to numofalternatives do
  begin
    suma := 1 ;
    for c := 1 to ( numofalternatives ) do
    begin
      if individualvector1[b] <
      individualvector1[c] then
        suma := suma + 1 ;
    end ;
    individualordinal[b] := suma ;
  end ;
  ordinal[a] := individualordinal1 ;
end ;

window (1,1,25,16) ;
textbackground (14) ;
clrscr ;
window (26,1,50,16) ;
textbackground (14) ;
clrscr ;
window (51,1,80,16) ;
textbackground (white) ;
clrscr ;
window (1,17,80,23) ;
textbackground (blue) ;
clrscr ;
window (1,24,80,25) ;
textbackground (white) ;
clrscr ;
textcolor (black) ;
gotoxy (2,1) ;
write ('step 5 : computation of group decision ' ) ;
gotoxy (2,2) ;
write ('the solutions have computed with electre or ahp ' ) ;
window (1,1,25,16) ;
textbackground (14) ;
textcolor (blue) ;
clrscr ;
win7 ;
win8 ;
window1 ;
end ;
window (1,1,80,25) ;
telos1:
    end;
END;

INCLUDE FILE DIARLIST1

PROCEDURE DIARLIST;

TYPE
Char12arr = array [ 1..12 ] of Char;
String20 = string[ 20 ];
RegRec = record
    AX, BX, CX, DX, BP, SI,
    DI, DS, ES, Flags : Integer;
end;

VAR
Regs : RegRec;
DTA : array [ 1..43 ] of Byte;
Mask : Char12arr;
NamR : String20;
Error, I : Integer;

BEGIN
FillChar(DTA, SizeOf(DTA), 0);   { Initialize the DTA buffer }
FillChar(Mask, SizeOf(Mask), 0);  { Initialize the mask }
FillChar(NamR, SizeOf(NamR), 0);  { Initialize the file name }
WriteLn;
WRITELN;
Regs.AX := $1A00;                 { Function used to set the DTA }
Regs.DS := Seg(DTA);             { store the parameter segment in DS }
Regs.DX := Ofs(DTA);             { offset in DX }
MSDos(Regs);                     { Set DTA location }
Error := 0;
Mask := '????????.GN?';          { Use global search }
Regs.AX := $4E00;                { Get first directory entry }
Regs.DS := Seg(Mask);            { Point to the file Mask }
Regs.DX := Ofs(Mask);
Regs.CX := 22;                   { Store the option }
MSDos(Regs);
Error := Regs.AX and $FF;        { Get Error return }

174
I := 1;  { initialize 'I' to the first element }
if (Error = 0) then
repeat
NamR[I] := Chr(Mem[Seg(DTA):Ofs(DTA)+29+I]);
I := I + 1;
until not (NamR[I-1] in [' '..'~']) or (I>20);

NamR[0] := Chr(I-1);  { set string length because assigning }
{ by element does not set length }

while (Error = 0) do
begin
Error := 0;
Regs.AX := $4F00;  { Function used to get the next }  { directory entry }
Regs.CX := 22;  { Set the file option }
MSDos(Reg);  { Call MSDos }
Error := Regs.AX and $FF;  { get the Error return }
I := 1;
repeat
NamR[I] := Chr(Mem[Seg(DTA):Ofs(DTA)+29+I]);
I := I + 1;
until not (NamR[I-1] in [' '..'~']) or (I>20);
NamR[0] := Chr(I-1);
if (Error = 0) THEN
WriteLn( ' ', NamR);  
end
END;

INCLUDE FILE FILES

PROCEDURE OPENFILE ( var prname : name );
BEGIN
assign ( problemfile , prname );
rewrite ( problemfile );
with problema do
begin
name1 := prname ;
numof alternatives := 0 ;
numof users := 0 ;
levels := 0 ;


fillchar (level1, sizeof(level1), 0);
fillchar (level2, sizeof(level2), 0);
fillchar (level3, sizeof(level3), 0);
fillchar (level4, sizeof(level4), 0);
fillchar (level5, sizeof(level5), 0);
fillchar (level6, sizeof(level6), 0);
fillchar (level7, sizeof(level7), 0);
fillchar (sublevel1, sizeof(sublevel1), 0);
fillchar (sublevel2, sizeof(sublevel2), 0);
fillchar (alternatives, sizeof(alternatives), 0);
end;
write (problemfile, problema);
close (problemfile);
END;

PROCEDURE OPENSOLUTIONFILE ( var pruser : name );
BEGIN
assign (solutionfile, pruser);
rewrite (solutionfile);
with solutiona do
begin
  ahp.Numoftries := 0;
electre.Numoftries := 0;
numofalternatives := 0;
numofcriteria := 0;
fillchar(ahp.Altvect1, sizeof(ahp.Altvect1), 0);
username := ' ';
userid := ' ';
fillchar(alternatives, sizeof(alternatives), 0);
fillchar(normvect1, sizeof(normvect1), 0);
electre.Status := false;
fillchar(electre.Outranking, sizeof(electre.Outranking), 0);
end;
write (solutionfile, solutiona);
close (solutionfile);
END;

PROCEDURE OPENNORMFILE ( var normname : name );
BEGIN
assign (normfile, normname);
rewrite (normfile);
with norma do
begin
  numofusers := 0;
  modifytimes := 0;
  lasttime := 0;
  agregation := false;
end;
nai := false;
specialized := false;
broadcasting := false;
modify := false;
fillchar (usersnames, sizeof(usersnames), 0);
fillchar (specindex, sizeof(specindex), 0);
fillchar (usersids, sizeof(usersids), 0);
fillchar (weight, sizeof(weight), 0);
fillchar (currentname, sizeof(currentname), 0);
fillchar (agregationname, sizeof(agregationname), 0);
end;
write (normfile, norma);
close (normfile);
END:

PROCEDURE OPENSPECFILE ( var pruser3 : name ) : BEGIN
assign ( specfile, pruser3 );
rewrite ( specfile );
with specfile do begin
    numofusers := 0;
pfactor := 0;
qfactor := 0;
fillchar (vector1, sizeof(vector1), 0);
fillchar (vector2, sizeof(vector2), 0);
fillchar (vector3, sizeof(vector3), 0);
fillchar (vector4, sizeof(vector4), 0);
fillchar (vector5, sizeof(vector5), 0);
fillchar (vector6, sizeof(vector6), 0);
fillchar (vector7, sizeof(vector7), 0);
fillchar (normvector2, sizeof(normvector2), 0);
fillchar (normvector1, sizeof(normvector1), 0);
fillchar (normindex, sizeof(normindex), 0);
fillchar (alternatives, sizeof(alternatives), 0);
fillchar (altnatrix, sizeof(altnatrix), 0);
fillchar (grading, sizeof(grading), 0);
umofcriteria := 0;
umofalternatives := 0;
for a := 1 to 3 do begin
    solved[a] := false;
    finalindex[a] := false;
    finalindex2[a] := false;
end;
completed := false;
completedall := false;
ahp.status := false;
fillchar (ahp.altvector1, sizeof(ahp.altvector1), 0);
ahl.numoftries := 0;
electre.status := false;
electre.numoftries := 0;

Fillchar(electre.outranking, sizeof(electre.outranking), 0);
End;
write (specfile, specfile1);
close (specfile);
END;

INCLUDE FILE UTILITIES

PROCEDURE DISKDATA;
BEGIN
repeat
window (1,1,40,12);
textbackground (blue);
clrscr;
window (41,1,80,12);
textbackground (blue);
clrscr;
window (1,13,80,23);
textbackground (14);
clrscr;
window (1,24,80,25);
textbackground (white);
clrscr;
textcolor (black);
gotoxy (2,1);
write (string128);
gotoxy(2,2);
write (' files related to the problem ');
window (1,1,40,12);
textbackground (blue);
textcolor (white);
gotoxy (2,2);
write (' names of problems : ');
dirlista;
window (41,1,80,12);
textbackground (blue);
textcolor (white);
gotoxy (2,2);
write (' names of norm : ');

178
dirlist;

window (1,13,80,23);
textbackground (14);
textcolor (black);
gotoxy (3,2);
write('do you want to see a predefined norm (y or n) ?
X1 := 56 ; y1 := 2;
identify (answer,x1,y1);
if answer = 'y' then
begin
   x1 := 3 ; y1 := 3;
normselection (x1,y1);
displaynorm;
end;
until answer = 'n';
c1rcscr;
repeat
   window (1,1,80,23):
textbackground (blue):
c1rcscr;

   window (41,1,80,23):
textbackground (blue):
c1rcscr;

   window (1,13,80,23):
textbackground (14):
c1rcscr;

   window (1,34,80,47):
textbackground (white):
c1rcscr;
textcolor (black):
gotoxy (3,2):
write ('choose:

gotoxy (3,1):
write ('from:

   window (1,1,80,23):
textbackground (14):
textcolor (black):
gotoxy (3,2):
write ('dirlist:

   window (41,1,80,23):
textbackground (blue):
textcolor (black):
gotoxy (3,2):
write ('end'):
dirlist;
window (1,13,80,23);
textbackground (14);
textcolor (black);
gotoxy (3,2);
write('do you want to see a predefined problem (y/n) ?');
x1 := 56; y1 := 2;
identify (answer,x1,y1);
if answer = 'y' then
begin
  gotoxy (3,3);
  repeat
    clrscr;
    gotoxy (2,1);
    write ('name of problem?');
    Read (answer);
    answer := stpcase (answer);
    answer := concat (answer,'.','Def');
  until (exist (answer));
  pname := answer;
  readproblemfile;
window (1,1,80,7);
textbackground (blue);
clrscr;
textcolor (white);
gotoxy (2,1);
write ('name of problem: ',problem.Name);
gotoxy (2,2);
write ('alternatives:');
for a := 1 to problem.Numofalternatives do
begin
  gotoxy (2,a+2);
  write (problem.Alternatives[a]);
end;
window (1,8,80,25);
textbackground (blue);
clrscr;
gotoxy (2,1);
write ('criteria:');
display (problem);
gotoxy (2,17);
textcolor (red);
write ('hit any key to continue');
read (kbd,ch);
textcolor (white);
end;
until answer = 'n';
END;
PROCEDURE DISKSTATUS;
BEGIN
diskdata;
END;

PROCEDURE READ1;
BEGIN
  repeat
    gotoxy ( 2,2 );
    clear;
    write (' the name of the problem ? ' );
    Read ( answer );
    specname := concat ( answer,'.Spc' );
    delete ( answer ,8,length( answer) );
    bname := concat ( answer,'.Def' );
    probname := answer;
    until exist (bname )
END;

PROCEDURE READ2;
BEGIN
  repeat
    gotoxy ( 2,4 );
    clear;
    write (' the name of the norm ? ' );
    Read ( answer );
    delete ( answer ,8,length( answer) );
    answer := stupcase ( answer );
    normname := concat ( answer,'.Gn' );
    until exist (normname )
END;

PROCEDURE READ3;
BEGIN
  gotoxy ( 2,6);
  write (' your name ? ' );
  Error := false;
  repeat
    gotoxy ( 16,6 );
    clear;
    read ( namex );
    namex := stupcase ( namex );
    for a := 1 to norm.Numofusers do
      begin
        if namex = norm.Usersnames[a] then
          error := true;
        end;

  Error ;
until ( error ) ;
b := 0 ;

repeat
    b := b + 1 ;
until ( name = norm.Usersnames[b] ) ;
str ( b, inte ) ;
anwer := concat ( ',', Name ) ;
prs := concat ( problname, answer ) ;
prs3 := concat ( problname,'.spc') ;
END ;

PROCEDURE READ4
BEGIN
    gotoxy ( 2,8 ) ;
    write (' your id ? ') ;
    if ( norm.Usersids[b] = 'x' ) then
        begin
            gotoxy ( 15,3 ) ;
            read ( idx ) ;
            idx := stupcase ( idx ) ;
            norm.Usersids[b] := idx ;
        end
    else
        repeat
            gotoxy ( 16,8 ) ;
            clreol;
            read ( idx ) ;
            idx := stupcase ( idx ) ;
            until ( idx = norm.Usersids[b] ) ;
        END ;
END ;

PROCEDURE READS
BEGIN
    gotoxy ( 2,10 ) ;
    write (' the method that you want to use ? ') ;
    Repeat
        gotoxy ( 49,10 ) ;
        clreol;
        read ( methodx ) ;
        methodx := stupcase ( methodx ) ;
        until (( methodx = 'ahp') or ( methodx = 'electre')
            or ( methodx = 'direct' ) ) ;
END ;
PROCEDURE DATA;
BEGIN
  window(1,24,80,25);
  textbackground(white);
  textcolor(black);
  gotoxy(2,1);
  clreol;
  write(string128);
  gotoxy(2,2);
  write(string129);
  window(1,13,80,23);
  textbackground(white);
  clrscr;
  read1;
  read2;
  readnormfile;
  read3;
  read4;
END;

INCLUDE FILE PROCED

FUNCTION STUPCASE (st : name) : name;
VAR
  I : integer;
BEGIN
  for I := 1 to length(st) do
    st[i] := upcase(st[i]);
  stupcase := st;
END;

FUNCTION EXIST (filename : name) : boolean;
VAR
  fil : file;
BEGIN
  assign (fil, filename);
  {$i-}
  reset (fil);
  {$i+}
  exist := (ioreult = 0);
END;

PROCEDURE WAIT;
BEGIN
  gotoxy(50,24); write('hit any key to continue');
  read(kbd, ch)
END;
PROCEDURE CLEAR1 ( var problem : casel ) ;
BEGIN
  for a := 1 to 5 do
    problem.Level1[a] := ' ';
END;

PROCEDURE CLEARSCREEN (var line : integer ) ;
BEGIN
  if line = 19 then
    begin
      gotoxy(1,4) ;
      for e := 1 to 10 do
data ;
      line := 9 ;
    end;
END;

PROCEDURE CLEAR ( var matrix2 : level ) ;
BEGIN
  for line := 1 to 5 do
    begin
      for a := 1 to 5 do
        matrix2[line, a] := ' ';
    end;
END;

PROCEDURE CONVERT (var answer2 : ask ; var w, d1 : integer ) ;
BEGIN
  if ( answer2[w] = '1' ) and ( answer2[w] = '2' )
    and ( answer2[w] = '3' ) and ( answer2[w] = '4' )
    and ( answer2[w] = '5' )
then
  d1 := 0 ;
  case answer2[w] of
    '1' : d1 := 1 ;
    '2' : d1 := 2 ;
    '3' : d1 := 3 ;
    '4' : d1 := 4 ;
    '5' : d1 := 5 ;
  end;
END;

PROCEDURE IDENTIFY ( var answer : name ; var x1, y1 : integer ) ;
BEGIN
  repeat
    gotoxy(x1, y1) ;
    clrscr ;
    read ( answer ) ;
    answer := toupper ( answer ) ;
    until ( ( answer = 'y' ) or ( answer = 'n' ) ) ;
END ;
PROCEDURE CHECKNUMBER ( var answer : name ;
                        var x1,y1,limit : integer ;
                        var count3 : real ) ;
BEGIN
  repeat
    gotoxy ( x1,y1 ) ;
    clrscr ;
    read ( answer ) ;
    val ( answer,count3,code1 ) ;
    until ((code1 = 0) and (count3 (= limit) and (count3 = 0)) ) ;
END ;

PROCEDURE SORT1 ( var normvector1 : vector ;
                   var normvector2 : vector ;
                   var numofcriteria : integer ) ;
BEGIN
  repeat
    count := 0 ;
    for a := 1 to numofcriteria do
      begin
        if normvector2[a] < normvector2[a+1] then
          begin
            exchange2[a] := normvector2[a] ;
            normvector2[a] := normvector2[a+1] ;
            normvector2[a+1] := exchange2[a] ;
            exchange1[a] := normvector1[a] ;
            normvector1[a] := normvector1[a+1] ;
            normvector1[a+1] := exchange1[a] ;
            count := count + 1 ;
          end ;
      end ;
    until count = 0 ;
END ;

PROCEDURE WRITENORMFILE ;
BEGIN
  assign ( normfile , normname ) ;
  reset ( normfile ) ;
  norm := norm ;
  write ( normfile , norm ) ;
  close ( normfile ) ;
END ;

PROCEDURE WRITEPROBLEMFILE ;
BEGIN
  assign ( problemfile , prname ) ;
  reset ( problemfile ) ;
PROCEDURE READPROBLEMFILE;
BEGIN
assign ( problemfile, prname );
reset ( problemfile );
read ( problemfile , problema );
problem := problema ;
close ( problemfile );
END ;

PROCEDURE READNORMFILE;
BEGIN
assign ( normfile , normname );
reset ( normfile );
read ( normfile , norma );
norm := norma ;
close ( normfile );
END ;

PROCEDURE READSOLUTIONFILE;
BEGIN
assign ( solutionfile , pruser );
reset ( solutionfile );
read ( solutionfile , solutiona );
solution := solutiona ;
close ( solutionfile );
END ;

PROCEDURE READSPECFILE;
BEGIN
assign ( specfile , pruser3 );
reset ( specfile );
read ( specfile , specfile1 );
specfile2 := specfile1 ;
close ( specfile );
END ;
PROCEDURE WRITESOLUTIONFILE;
BEGIN
  assign (solutionfile, pruser);
  reset (solutionfile);
  
  solutiona := solution;
  write (solutionfile, solutiona);
  close (solutionfile);
END;

PROCEDURE WRITESPECFILE;
BEGIN
  assign (specfile, pruser3);
  reset (specfile);
  specfile1 := specfile2;
  write (specfile, specfile1);
  close (specfile);
END;
APPENDIX B

FIGURES OF SCENARIO 1

NAME OF THE GROUP NORM ? select frigates

1. IDENTIFICATION OF GROUP MEMBERS
   1.1 Number of Group Members (MAX 3) ? 3
      - Name of Member # 1 ? user1
      - Name of Member # 2 ? user2
      - Name of Member # 3 ? user3
   1.2 ID of Member A1 ? x1

2. GROUP DECISION TECHNIQUES
   2.1 Weighted Majority Rule :
      - EQUAL Weights (Y/N) ? y
   2.2 Collective Evaluation Mode
      Choose one of the two modes :
      (1) Each group member will evaluate alternatives
          according to all criteria
      (2) Each group member will evaluate only alternatives
          according to his exclusive area of expertise
      Enter selection ? 2
      The name of the problem ? ships
      - Name of user for criteria WEAPONS : user1
      - Name of user for criteria ELECTRONICS : user2
      - Name of user for criteria ENGINE : user3
      - Name of user for criteria ECONOMICAL : user3
   2.3 Automatic Selection of Techniques of Aggregation of Preference (Y/N) ? y
   2.3 Automatic Computation of NAI (Y/N) ? y

3. INFORMATION EXCHANGE
   3.1 Broadcasting of individual outputs (Y/N) ? y
   3.2 Permission to Modify Individual Analyses
      AFTER Group analyses (Y/N) ? y
      3.2.1 How Many Times (MAX 10) ? 9
   3.3 Time Limit to Submit Individuals Results :
      3.3.1 How Many Days (MAX 14) ? 7
      3.3.2 Hours (11:00 to 24:00) ? 13:00

STEP 2 : GROUP NORM DEFINITION

Figure 4. Group Norm Definition
NAME OF PROBLEM : SHIPS

ENTER THE ALTERNATIVES : 1. type 1
2. type 2
3. type 3
4. q

ENTER THE CRITERIA : 1. weapons
   1.1 air-crafts
   1.2 guns
   1.3 missiles
       1.3.1 ssm
       1.3.2 sam
   1.4 a/s weapons
2. electronics
   2.1 radar
       2.1.1 surveillance
       2.1.2 fire control
       2.1.3 navigation
   2.2 sonar
3. engine
   3.1 performance
   3.2 maintenance
4. cost related
   4.1 technical support
   4.2 life cycle
   4.3 cost of operation
   4.4 cost
   4.5 q

STEP 1 : MULTIPLE CRITERIA GROUP PROBLEM DEFINITION
Definition of criteria * 1st level 2nd level 3rd level 4th level 5th level

Figure 5. Step 1 Group Problem Definition
The Name of the Problem? ships
The Name of the Norm? normsh
Your Name? user1
Your ID? x1
The method that you want to use? ahp

Step 3: Prioritization of Evaluation Criteria
Identification of the problem Methods: AHP or DIRECT

Figure 6 User 1 / Problem Initiation
**PAIRWISE COMPARISON**

<table>
<thead>
<tr>
<th>WEAPON</th>
<th>ELECT</th>
<th>ENGIN</th>
<th>ECONOMICAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEAPON</td>
<td>1.20</td>
<td>2.00</td>
<td>2.50</td>
</tr>
<tr>
<td>ELECT</td>
<td>0.83</td>
<td>1.95</td>
<td>2.40</td>
</tr>
<tr>
<td>ENGIN</td>
<td>0.50</td>
<td>0.51</td>
<td>1.90</td>
</tr>
<tr>
<td>ECONOMICAL</td>
<td>0.40</td>
<td>0.42</td>
<td>0.53</td>
</tr>
</tbody>
</table>

**PRIORITY VECTOR**

<table>
<thead>
<tr>
<th>WEAPON</th>
<th>ELECTRONICS</th>
<th>ENGINE</th>
<th>ECONOMICAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.366</td>
<td>0.385</td>
<td>0.183</td>
<td>0.146</td>
</tr>
</tbody>
</table>

**Consistency Analysis**

- **λ MAX** = 4.28
- **Consistency Index** = 0.00
- **Randomized Index** = 0.30
- **Consistency Ratio** = 0.00

**LMDA X: 3.37 0.30 0.18 0.14**

**DO YOU WANT TO MODIFY THE EVALUATION OF THE CRITERIA (Y/N)?**

**STEP 3: PRIORITYIZATION OF EVALUATION CRITERIA**

**Method:** AHP

---

*Figure 7. User 1 / Prioritization of Eval. Crit. at the First Level.*
PAIRWISE COMPARISON

<table>
<thead>
<tr>
<th></th>
<th>AIR-C</th>
<th>GUNS</th>
<th>MISSI</th>
<th>A/S W</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIR-C</td>
<td>1.70</td>
<td>1.20</td>
<td>1.20</td>
<td>1.20</td>
</tr>
<tr>
<td>GUNS</td>
<td>0.59</td>
<td>0.56</td>
<td>0.59</td>
<td></td>
</tr>
<tr>
<td>MISSI</td>
<td>0.83</td>
<td>1.00</td>
<td>1.10</td>
<td></td>
</tr>
<tr>
<td>A/S W</td>
<td>0.83</td>
<td>1.70</td>
<td>0.91</td>
<td></td>
</tr>
</tbody>
</table>

PRIORITY VECTOR

<table>
<thead>
<tr>
<th></th>
<th>AIR-CRAFTS</th>
<th>0.307</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUNS</td>
<td>0.181</td>
<td></td>
</tr>
<tr>
<td>MISSILES</td>
<td>0.256</td>
<td></td>
</tr>
<tr>
<td>A/S WEAPONS</td>
<td>0.256</td>
<td></td>
</tr>
</tbody>
</table>

$\lambda_{\text{max}} = 4.81$

CONSISTENCY INDEX = 0.30
RANDOMIZED INDEX = 0.30
CONSISTENCY RATIO = 0.30

DO YOU WANT TO MODIFY THE EVALUATION OF THE CRITERIA (Y/N) ?

STEP 3: PRIORITIZATION OF EVALUATION CRITERIA
Method: AHP

Figure 8. User 1 / Prioritization of Evaluation Criteria at Level 2 For Criteria 1.1 to 1.4
### STEP 3: PRIORITIZATION OF EVALUATION CRITERIA

Direct input of criteria weights

<table>
<thead>
<tr>
<th>PRIORISE COMPARISON</th>
<th>PRIORITY VECTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSM</td>
<td>SAM</td>
</tr>
<tr>
<td>SSM</td>
<td>1.30</td>
</tr>
<tr>
<td>SAM</td>
<td>0.77</td>
</tr>
</tbody>
</table>

SSM  | SAM  | 0.43  | 0.56 |

HIT ANY KEY TO CONTINUE

**Figure 9.** User 1 / Prioritization of Evaluation Criteria at Level 3 for Criteria 1.3.1 and 1.3.2
Figure 10. User 2 / Prioritization of Evaluation Criteria for Level 1
DO YOU WANT TO MODIFY THE EVALUATION OF THE CRITERIA (Y/N)?

STEP 3: PRIORITIZATION OF EVALUATION CRITERIA
Direct input of criteria weights

Figure 11. User 2 / Prioritization of Evaluation Criteria at Level 2 for Criteria 2.1 and 2.2
Figure 12. User 2 / Prioritization of Evaluation Criteria at Level 3 for Criteria 2.1.1 to 2.1.3
Figure 13. User 3 / Prioritization of Evaluation Criteria at Level 1
Figure 14. User 3 / Prioritization of Evaluation Criteria at Level 2 for Criteria 3.1 and 3.2
Figure 15. User 3 / Prioritization of Evaluation Criteria at Level 2 for Criteria 4.1 To 4.4
The final criteria (15) and their weights are:

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. SONAR</td>
<td>0.19</td>
</tr>
<tr>
<td>2. AIR-CRAFTS</td>
<td>0.13</td>
</tr>
<tr>
<td>3. A/S WEAPONS</td>
<td>0.11</td>
</tr>
<tr>
<td>4. MAINTANCE</td>
<td>0.08</td>
</tr>
<tr>
<td>5. GUNS</td>
<td>0.06</td>
</tr>
<tr>
<td>6. SURVEILLANCE</td>
<td>0.08</td>
</tr>
<tr>
<td>7. FIRE CONTROL</td>
<td>0.08</td>
</tr>
<tr>
<td>8. SAM</td>
<td>0.05</td>
</tr>
<tr>
<td>9. PERFORMANCE</td>
<td>0.05</td>
</tr>
<tr>
<td>10. SSM</td>
<td>0.05</td>
</tr>
<tr>
<td>11. NAVIGATION</td>
<td>0.34</td>
</tr>
<tr>
<td>12. TECHNICAL SUPPORT</td>
<td>0.03</td>
</tr>
<tr>
<td>13. COST OF OPERATION</td>
<td>0.03</td>
</tr>
<tr>
<td>14. LIFE CYCLE</td>
<td>0.01</td>
</tr>
<tr>
<td>15. COST</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Do you want to reduce the number of the criteria (Y/N)? Y

Figure 16. Final Weights of Evaluation Criteria
**THE FINAL CRITERIA (5) AND THEIR WEIGHTS ARE:**

1. SONAR : 0.32
2. AIR-CRAFTS : 0.22
3. A/S WEAPONS : 0.19
4. MAINTANCE : 0.13
5. GUNS : 0.13

---

**DO YOU WANT TO CHANGE THE NUMBER OF THE CRITERIA (Y/N) ?**  
N

---

**STEP 3: PRIORITIZATION OF EVALUATION CRITERIA**

*Figure 17. The Reduced Set of Criteria*
<table>
<thead>
<tr>
<th>PAIRWISE COMPARISON</th>
<th>PRIORITY VECTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TYP1</strong></td>
<td><strong>TYPE2</strong></td>
</tr>
<tr>
<td>TYP1</td>
<td>1.10</td>
</tr>
<tr>
<td>TYPE2</td>
<td>0.91</td>
</tr>
<tr>
<td>TYPE3</td>
<td>0.83</td>
</tr>
</tbody>
</table>

**LAMBDA MAX** = 3.30
**CONSISTENCY INDEX** = 0.00
**RANDOMIZED INDEX** = 0.58
**CONSISTENCY RATIO** = 0.00

**DO YOU WANT TO MODIFY THE DATA (Y/N) ? N**

**STEP 4: INDIVIDUAL EVALUATION OF ALTERNATIVES**
Evaluation of alternatives According to Criterion SONAR (AHP)

---

**Figure 18. User 2 / Evaluation of the Alternatives According to Criteria Sonar (AHP)**
### Pairwise Comparison

<table>
<thead>
<tr>
<th></th>
<th>TYP1</th>
<th>TYP2</th>
<th>TYP3</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYP1</td>
<td>1.20</td>
<td>0.83</td>
<td>1.20</td>
</tr>
<tr>
<td>TYP2</td>
<td>0.83</td>
<td>0.77</td>
<td>1.20</td>
</tr>
</tbody>
</table>

### Priority Vector

<table>
<thead>
<tr>
<th></th>
<th>TYP1</th>
<th>TYP2</th>
<th>TYP3</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYP1</td>
<td>0.338</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYP2</td>
<td>0.275</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYP3</td>
<td>0.396</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**λmax = 3.30**

- Consistency Index = 0.00
- Randomized Index = 0.58
- Consistency Ratio = 0.00

**DO YOU WANT TO MODIFY THE DATA (Y/N)?**

**Figure 13. User 1 / Evaluation of Alternatives According to Criterion Air-Crafts (AHP)**
<table>
<thead>
<tr>
<th>Pairwise Comparison</th>
<th>Priority Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE    TYPE    TYPE</td>
<td>TYPE TYPE TYPE</td>
</tr>
<tr>
<td>TYPE    0.91    0.91</td>
<td>TYPE 123 0.313</td>
</tr>
<tr>
<td>TYPE    1.10    1.00</td>
<td>TYPE 234 0.344</td>
</tr>
<tr>
<td>TYPE    1.10    1.00</td>
<td>TYPE 1.2.3 0.344</td>
</tr>
</tbody>
</table>

- **LAMDA MAX** = 3.30
- Consistency Index = -0.00
- Randomized Index = 0.58
- Consistency Ratio = -0.00

Do you want to modify the data (Y/N)?

**Step 4: Individual Evaluation of Alternatives**
Evaluation of Alternatives According to criterion GUNS (AHP)

Figure 20. User 1 / Evaluation of Alternatives According to Criteria Guns (AHP)
Figure 21. User 2 / Evaluation of alternatives According to Criterion Sonar (DIRECT)
### Evaluation of Working Area

<table>
<thead>
<tr>
<th>TYP</th>
<th>SDN</th>
<th>AIR</th>
<th>A/S</th>
<th>MAI</th>
<th>GUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYP1</td>
<td>8.00</td>
<td>7.00</td>
<td>7.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYP2</td>
<td>8.32</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYP3</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Priority Vector

<table>
<thead>
<tr>
<th>TYP</th>
<th>TYP1</th>
<th>TYP2</th>
<th>TYP3</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYP1</td>
<td>0.36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYP2</td>
<td>0.32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYP3</td>
<td>0.32</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Figure 22.** User 1 / Evaluation of alternatives According to Criterion Maintenance (DIRECT)
Figure 23. Final Group Solution of the Problem
APPENDIX C
FIGURES FOR SCENARIO 2

NAME OF THE GROUP NORM ? select one

1. IDENTIFICATION OF GROUP MEMBERS
1.1 Number of Group Members (MAX 3) ? 3
   - Name of Member #1 ? user1
   - Name of Member #2 ? user2
   - Name of Member #3 ? user3
1.2 ID of Member USER1 ? x1

2. GROUP DECISION TECHNIQUES
2.1 Weighted Majority Rule :
   - EQUAL Weights (Y/N) ? n
   - WEIGHT for USER1 ? 3
   - WEIGHT for USER2 ? 3
   - WEIGHT for USER3 ? 4

2.2 Collective Evaluation Mode
   - Choose one of the following modes :
     (1) Each group member will evaluate alternatives according to all criteria
     (2) Each group member will evaluate only criteria according to his exclusive area of his expertise
   Enter a number ? 1

2.3 Automatic Selection of Techniques of Aggregation of Preference (Y/N) ? y
2.4 Automatic Computation of NRI (Y/N) ? y

3. INFORMATION EXCHANGE
3.1 Broadcasting of individual outputs (Y/N) ? y
3.2 Permission to Modify Individual Analyses AFTER Group analyses (Y/N) ? y
   3.2.1 How Many Times (MAX 10) ? 4
3.3 Time Limit to Submit Individuals Results :
   3.3.1 How Many Days (MAX 14) ? 7
   3.3.2 Hours (1:00 to 24:00) ? 12:00

STEP 2 : GROUP NORM DEFINITION

Figure 24. Group Norm Definition
<table>
<thead>
<tr>
<th>NAME OF PROBLEM : SHIPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTER THE ALTERNATIVES :</td>
</tr>
<tr>
<td>1. type 1</td>
</tr>
<tr>
<td>2. type 2</td>
</tr>
<tr>
<td>3. type 3</td>
</tr>
<tr>
<td>4. q</td>
</tr>
<tr>
<td>ENTER THE CRITERIA      :</td>
</tr>
<tr>
<td>1. weapons</td>
</tr>
<tr>
<td>2. electronics</td>
</tr>
<tr>
<td>3. engine</td>
</tr>
<tr>
<td>4. cost related</td>
</tr>
<tr>
<td>5. q</td>
</tr>
</tbody>
</table>

Figure 25. 1 Group Problem Definition
<table>
<thead>
<tr>
<th>USER 1</th>
<th>PROBLEM INITIATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>select</td>
</tr>
<tr>
<td>NORM</td>
<td>select1</td>
</tr>
<tr>
<td>USER</td>
<td>user1</td>
</tr>
<tr>
<td>ID</td>
<td>x1</td>
</tr>
<tr>
<td>METHOD</td>
<td>ahp</td>
</tr>
</tbody>
</table>

**STEP 3: PRIORITIZATION OF EVALUATION CRITERIA**
Identification of the problem Methods: AHP or DIRECT

Figure 26. User 1 / Problem Initiation
**Figure 27. User 1 / Prioritization of Evaluation Criteria (AHP)**

<table>
<thead>
<tr>
<th>PAIRWISE COMPARISON</th>
<th>PRIORITY VECTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEAP</td>
<td>ELECT</td>
</tr>
<tr>
<td>WEAP</td>
<td>1.00</td>
</tr>
<tr>
<td>ELECT</td>
<td>1.00</td>
</tr>
<tr>
<td>ENGIN</td>
<td>0.63</td>
</tr>
<tr>
<td>COST</td>
<td>0.53</td>
</tr>
</tbody>
</table>

- **Lambda Max** = 4.21
- **Consistency Index** = 0.30
- **Randomized Index** = 0.30
- **Consistency Ratio** = 0.30

**STEP 3: PRIORITIZATION OF EVALUATION CRITERIA**
**Method**: AHP

Do you want to modify the evaluation of the criteria (Y/N)?

Weights: 0.32 0.32 0.20 0.17
THE FINAL CRITERIA (4) AND THEIR WEIGHTS ARE:
1. WEAPONS: 0.32
2. ELECTRONICS: 0.32
3. ENGINE: 0.20
4. COST RELATED: 0.17

YOU HAVE TWO METHODS:

1. DEFINE THE NUMBER OF THE CRITERIA THAT YOU WANT TO USE
2. DEFINE THE SUM (%) THAT YOU WISH

METHOD THAT YOU WISH (1 OR 2)?

STEP 3: PRIORITIZATION OF EVALUATION CRITERIA
Determine the number of the criteria

Figure 28. User 1 / Final Weights of Evaluation Criteria
THE FINAL CRITERIA (3) AND THEIR WEIGHTS ARE:

1. WEAPONS : 0.38
2. ELECTRONICS : 0.38
3. ENGINE : 0.24

DO YOU WANT TO CHANGE THE NUMBER OF THE CRITERIA (Y/N) ?  n

STEP 3: PRIORITIZATION OF EVALUATION CRITERIA
Determine the number of the criteria

FIGURE 29. USER 1 / THE REDUCED SET OF CRITERIA
**Figure 30. User 1 / Individual Evaluation of Alternatives Using Direct mode**

<table>
<thead>
<tr>
<th>Working Area</th>
<th>Priority Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYP TYP TYP</td>
<td>TYPE 1 0.32</td>
</tr>
<tr>
<td>MEA 9.00 7.00 9.00</td>
<td></td>
</tr>
<tr>
<td>ELE 9.00 9.00 7.00</td>
<td>TYPE 2 0.32</td>
</tr>
<tr>
<td>ENG 8.00 8.00 9.00</td>
<td>TYPE 3 0.36</td>
</tr>
<tr>
<td>TYP TYP TYP</td>
<td>0.32 0.32 0.36</td>
</tr>
</tbody>
</table>

**Method used:** Direct Input

Do you want to modify the weights (Y/N) ?
Figure 31. Solution of User 1 (With Direct Mode)
Figure 32. User 1 / Evaluation of Alternatives Using Electre
**CONCORDANCE MATRIX**

<table>
<thead>
<tr>
<th>TYP</th>
<th>TYP</th>
<th>TYP</th>
<th>OCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYP</td>
<td>76</td>
<td>62</td>
<td>2</td>
</tr>
<tr>
<td>TYP</td>
<td>62</td>
<td>24</td>
<td>1</td>
</tr>
<tr>
<td>TYP</td>
<td>38</td>
<td>76</td>
<td>1</td>
</tr>
</tbody>
</table>

**Discordance index**

- A Concordance index indicates to what extent an option is better than another in terms of criteria weights.
- The index varies between \([0 - 100]\) the higher the better.

**DISCORDANCE MATRIX**

<table>
<thead>
<tr>
<th>TYP</th>
<th>TYP</th>
<th>TYP</th>
<th>IDI</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYP</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>TYP</td>
<td>11</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>TYP</td>
<td>5</td>
<td>13</td>
<td>2</td>
</tr>
</tbody>
</table>

**Outranking relation**

- An Outranking relation » is the one that satisfies both concordance and discordance requirements.
- An » indicates that there is no outranking relations.

**OUTRANKING MATRIX**

<table>
<thead>
<tr>
<th>TYP</th>
<th>TYP</th>
<th>TYP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYP</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>TYP</td>
<td>*</td>
<td>-</td>
</tr>
<tr>
<td>TYP</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**STEP 4 : EVALUATION OF ALTERNATIVES**

Method used : ELECTRE

Figure 33. User 1 / Concordance, Discordance, Outranking Matrix
Figure 34. User2 / Individual Evaluation of Alternatives Using Direct Mode
Figure 35. Solution of User 2 (With Direct Mode)
### ALTERNATIVE EVALUATION: WORKING AREA GRADING SCALE

<table>
<thead>
<tr>
<th>WEA ELE ENG</th>
<th>VAL ELE ENG</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYP 31 29 16</td>
<td>Weig.: 38 38 24</td>
</tr>
<tr>
<td>TYP 27 32 22</td>
<td>Exce 38 38 24</td>
</tr>
<tr>
<td>TYP 30 25 17</td>
<td>Good 29 29 18</td>
</tr>
</tbody>
</table>

- **Excellent**: 38 - 38 - 24
- **Good**: 29 - 29 - 18
- **AVERAGE**: 19 - 19 - 12
- **FAIR**: 10 - 10 - 6
- **WEAK**: 0 - 0 - 0

\[ P = 55.00 \%, Q = 55.00 \% \]

---

**MENU**

1. CONCORDANCE MATRIX
2. DISCORDANCE MATRIX
3. OUTRANKING MATRIX
4. MODIFY THRESHOLDS
5. EXIT ELECTRE

**SELECTION (1-5) ?**

---

**Figure 36. User 2 / Evaluation of Alternatives Using Electre**
**CONCORDANCE MATRIX**

<table>
<thead>
<tr>
<th>TYP</th>
<th>TYP</th>
<th>TYP</th>
<th>#CI</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYP</td>
<td>-</td>
<td>38</td>
<td>76</td>
</tr>
<tr>
<td>TYP</td>
<td>62</td>
<td>-</td>
<td>62</td>
</tr>
<tr>
<td>TYP</td>
<td>24</td>
<td>38</td>
<td>-</td>
</tr>
</tbody>
</table>

**DISCORDANCE MATRIX**

<table>
<thead>
<tr>
<th>TYP</th>
<th>TYP</th>
<th>TYP</th>
<th>#DI</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYP</td>
<td>-</td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td>TYP</td>
<td>11</td>
<td>-</td>
<td>8</td>
</tr>
<tr>
<td>TYP</td>
<td>11</td>
<td>18</td>
<td>-</td>
</tr>
</tbody>
</table>

**OUTRANKING MATRIX**

<table>
<thead>
<tr>
<th>TYP</th>
<th>TYP</th>
<th>TYP</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYP</td>
<td>-</td>
<td>-</td>
<td>*</td>
</tr>
<tr>
<td>TYP</td>
<td>*</td>
<td>-</td>
<td>*</td>
</tr>
<tr>
<td>TYP</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

**HIT ANY KEY TO CONTINUE**

**STEP 4: EVALUATION OF ALTERNATIVES**

Method used: ELECTRE

Figure 37. User 2 / Concordance, Discordance, Outranking Matrix
Figure 38. User 3 / Individual Evaluation of Alternatives Using Direct Mode
Figure 39. Solution of User 3 (With Direct Mode)
### ALTERNATIVE EVALUATION: WORKING AREA

#### GRADING SCALE

<table>
<thead>
<tr>
<th>WEA ELE ENG</th>
<th>WEA ELE ENG</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYP 31 26 15</td>
<td>Weig.: 38 38 24</td>
</tr>
<tr>
<td>TYP 28 28 23</td>
<td>Exce 38 38 24</td>
</tr>
<tr>
<td>TYP 32 25 17</td>
<td>Good 29 29 18</td>
</tr>
<tr>
<td></td>
<td>Aver 19 19 12</td>
</tr>
<tr>
<td></td>
<td>Fair 10 10 6</td>
</tr>
<tr>
<td></td>
<td>Weak 0 0 0</td>
</tr>
<tr>
<td></td>
<td>P = 55.00 % 0 = 55.00 %</td>
</tr>
</tbody>
</table>

#### Menu

1. CONCORDANCE MATRIX
2. DISCORDANCE MATRIX
3. OUTRANKING MATRIX
4. MODIFY THRESHOLDS
5. EXIT ELECTRE

SELECTION (1-5) ?

#### Step 4: Evaluation of Alternatives

Method Used: ELECTRE

---

**Figure 40. User 3 / Evaluation of Alternatives Using Electre**
A Concordance index indicates to what extent an option is better than another in terms of criteria weights.

The index varies between [0 - 100] the higher the better.

3 indexes are \( = 55 \)

Column CCI indicates the 0 of indexes satisfying \( P \) for each option.

A Discordance index indicates to what extent an option contains a bad element that makes it un-satisfactory.

The index varies between [0 - 100] the lower the better.

6 indexes are \( \leq 55.00 \)

Column CCI indicates the 0 of indexes satisfying \( Q \) for each option.

An Outranking relation * is the one that satisfies both concordance and discordance requirements.

An - indicates that there is no outranking relations.

**Figure 41. User 3 / Concordance, Discordance, Outranking Matrix**
### ALT. CARDINAL RANKINGS | ORDINAL RANKING | GROUP RESULTS
---|---|---
USER1 USER2 USER3 | USER1 USER2 USER3 | R1 | R2 | R3 | R4

<table>
<thead>
<tr>
<th>Weig.</th>
<th>3.00</th>
<th>3.00</th>
<th>4.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYP 1.01</td>
<td>1.01</td>
<td>1.35</td>
<td></td>
</tr>
<tr>
<td>TYP 1.06</td>
<td>1.06</td>
<td>1.41</td>
<td></td>
</tr>
<tr>
<td>TYP 1.01</td>
<td>1.01</td>
<td>1.34</td>
<td></td>
</tr>
</tbody>
</table>

**R1:** SUM OF RANKS
**R2:** ADDITIVE RANKING
**R3:** MULTIPLICATIVE RANKING
**R4:** SUM OF OUTRANKING RELATIONS

HIT ANY KEY TO CONTINUE

**STEP 5: COMPUTATION OF GROUP DECISION**
All the solutions have computed with AHP

---

**Figure 42. Group Solution of the Problem**
(Solved with Direct Mode)
<table>
<thead>
<tr>
<th>USER1</th>
<th>USER2</th>
<th>USER3</th>
<th>USE</th>
<th>USE</th>
<th>USE</th>
<th>R4</th>
<th>R1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYP 0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>TYP 0</td>
<td>9</td>
</tr>
<tr>
<td>TYP 2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>TYP 6</td>
<td>3</td>
</tr>
<tr>
<td>TYP 1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>TYP 3</td>
<td>6</td>
</tr>
</tbody>
</table>

R1 : SUM OF RANKS
R2 : ADDITIVE RANKING
R3 : MULTIPLICATIVE RANKING
R4 : SUM OF OUTRANKING RELATIONS

HIT ANY KEY TO CONTINUE

STEP 5 : COMPUTATION OF GROUP DECISION
All the solutions have computed with ELECTRE

Figure 43. Group Solution of the Problem
(Solved with Electre Mode)
LIST OF REFERENCES


<table>
<thead>
<tr>
<th>No.</th>
<th>Copies</th>
<th>Initial Distribution List</th>
</tr>
</thead>
</table>
| 1.  | 2      | Defense Technical Information Center  
Cameron Station,  
Alexandria, Virginia 22304-6145 |
| 2.  | 2      | Library, Code 0142  
Naval Postgraduate School  
Monterey, California 93943-5002 |
| 3.  | 2      | Dr. Tung X. Bui, Code 54BD  
Department of Administrative Sciences  
Naval Postgraduate School  
Monterey, California 93943-5000 |
| 4.  | 1      | Dr. T.R Sivasankan, Code 54SV  
Department of Administrative Sciences  
Naval Postgraduate School  
Monterey, California 93943-5000 |
| 5.  | 7      | LT C. K. Skindellas  
Strofiliou 1b  
Kifissia  
Athens, Greece |
| 6.  | 1      | Computer Technology Programs, Code 37  
Naval Postgraduate School  
Monterey, California 93943-5000 |