IRIS: A DSS for Multiple Criteria Sorting Problems

Luís C. Dias

Faculdade de Economia, Universidade de Coimbra, Av. Dias da Silva, 165, 3004-512 Coimbra, Portugal
and INESC Coimbra, Rua Antero de Quental, 199, 3000-033 Coimbra, Portugal
Tel.: +351 239 851040; Fax: +351 239 824692
LDias@inescc.pt

Vincent Mousseau

LAMSADE, Université Paris-Dauphine, Place du Maréchal De Lattre de Tassigny, 75775 Paris Cedex 16, France
mousseau@lamsade.dauphine.fr

Abstract

This paper presents IRIS (Interactive Robustness analysis and parameters' Inference for multicriteria Sorting problems), a Decision Support System (DSS) designed to sort actions (projects, candidates, alternatives, clients, ...) described by their performance on multiple criteria into an ordered set of categories defined a priori. It is based on the ELECTRE TRI sorting method, but does not require the Decision Maker (DM) to indicate precise values for all of the method's parameters. More realistically, the software expects the DM to indicate some constraints these parameters should respect, including sorting examples that the program should reproduce. If the constraints indicated by the DM do not contradict each other (i.e. form a consistent system), then IRIS infers a combination of parameter values that reproduces all the sorting examples, indicating also the range of possible assignments of actions to categories that would be possible without violating any of the stated constraints. If the constraints are contradictory (i.e. form an inconsistent system), then IRIS suggests a combination of parameter values that minimizes an error function and identifies alternative ways to restore the system's consistency by removing some constraints.

Key words: Decision Support Systems, Sorting, Classification, ELECTRE TRI, Software

1 Introduction

This paper presents the IRIS (Interactive Robustness analysis and parameters' Inference for multicriteria Sorting problems) software, a DSS for multicriteria ordinal classification (sorting) problems. In classification problems, a set of "objects" (which we call *actions*) is to be classified into different categories. These actions (projects, candidates, alternatives, clients, students, …) are described by a vector evaluating their performance on multiple criteria. For instance, criteria such as "publications", "advanced formation", or "links with industry" can be used to classify R&D institutions into categories such as "Poor", "Fair", "Good" and "Excellent". Other examples can be the classification of loan requests into categories "Reject", "Accept with high interest rate", "Accept with low interest rate", or to sort a company's employees into categories associated with incentive packages, or to sort automobiles into categories related with environmental friendliness. In all these examples, categories can be ordered by increasing preference for the DM, thus are often called *sorting* problems (e.g, Greco et al. 2001; Zopounidis and Doumpos 2002). Other important aspects are that categories are defined a priori, and each action is compared to the definitions of the categories independently from the other actions. Hence, absolute evaluations are at stake, rather than the relative evaluations that occur in choice or ranking problem statements, where actions are compared one to each other (Roy, 1996).

Multicriteria classification methods can be distinguished from others (e.g. methods from statistics) in that classification does not automatically result from the vectors describing the actions, but does depend on the judgment of a DM. The DM defines the "*boundaries*" of the categories, the importance of each criterion, etc. The DM's judgment is represented in the method through values assigned to preference parameters. However, it is usually difficult for a DM to assign precise quantitative values to these parameters. Moreover, the parameters reflect preferences that are often vague and that may change with time. In some situations, there will not exist a single isolated DM, but a set of DMs conjointly responsible for the decision whose preferences do not perfectly match. For all these reasons, in many cases it is wise to avoid questioning the DM(s) directly in a quest for precise parameter values. As an alternative, the DM can be aided by procedures that infer parameter values from sorting examples (Mousseau and Slowinski 1998) and/or by procedures that work with imprecise information and perform robustness analysis (Dias and Clímaco 2000).

The IRIS software intends to become a reference implementation of the ELECTRE TRI method, succeeding the implementations of (Yu 1992) and (Mousseau *et al.* 1999). It supports

the methodology proposed by (Dias *et al.* 2002), (Mousseau and Dias 2004), (Mousseau *et al.* 2003), based on the method ELECTRE TRI (Yu 1992), (Roy and Bouyssou 1993), combining parameter inference with robustness analysis. The main feature of IRIS is that it does not require precise values for the criteria importance parameters nor the method's outranking cutting level. The DM may instead indicate constraints (intervals or relations) that the parameters should respect. In particular, IRIS allows entering constraints in the form of sorting examples (for some particular actions chosen by the DM, he/she indicates a category or an interval of categories where the actions should be classified according to his/her judgment). When the constraints define a consistent system, IRIS computes a "central" combination of parameter values that respects all the constraints and the corresponding sorting, additionally indicating the range of categories where each action could have been classified without violating any constraint. When the constraints contradict each other (i.e. form an inconsistent system), IRIS suggests alternative ways of removing subsets of constraints in a way that restores consistency (Mousseau *et al.* 2003).

The next section briefly overviews the ELECTRE TRI method. Section 3 summarizes the methodology implemented in IRIS on how to construct an ELECTRE TRI sorting model in an interactive way. Section 4 presents the structure of the IRIS user interface, which is illustrated by a "guided tour" in Section 5. Section 6 briefly presents two real-world applications of IRIS. A conclusions section ends the paper.

2 Brief overview of ELECTRE TRI

The ELECTRE family of methods for multicriteria decision aiding has been developed by Bernard Roy and his collaborators for the last three decades (Roy 1991), (Roy and Bouyssou 1993). Among these methods, ELECTRE TRI (Yu 1992), (Roy and Bouyssou 1993) has been specifically designed for multicriteria sorting problems.

Let $A = \{a_1, ..., a_m\}$ denote a set of m actions evaluated according to n criteria (functions) $g_1,...,g_n$. We denote $C = \{C^1, ..., C^k\}$ a set of k categories by preference order, being C^1 the least preferred (worst category) and C^k the most preferred (best category). Each category C^h ($h=1,...,k$) is defined by two *profiles*: b^h is its upper-bound profile, whereas b^{h-1} is its lower-bound profile. Thus, it is necessary to define $k+1$ profiles b^0, ..., b^k such that, except the first one and the last one, each profile is simultaneously the upper bound of a category and the lower bound of the category above it (Fig. 1). (Yu, 1992) has established the conditions these profiles should respect, namely: each profile b^h must be preferred to profile b^{h-1} according to all the

criteria, and the profile b^0 (b^k) must be worse (better, respectively) on all criteria than the actions to sort.

{Figure 1 – Definition of categories using profiles.}

The assignment of actions to categories is based on the concept of *outranking relation* (a binary relation meaning "not worse than"). An action a_i is said to outrank a profile b^h (denoted by a_i S b^h) when there are sufficient arguments to state that a_i is at least as good as b^h. Formally, a *credibility degree* $s(a_i,b^h)$ is computed first (a valued outranking relation), considering the "weight" of the criteria that agree with the statement a_i S b^h and the discordance opposed by the remaining criteria (see Section 2.1). Then, this index is compared to a *cutting level* λ:

$$a_i \, S \, b^h \quad \text{iff} \quad s(a_i,b^h) \geq \lambda. \tag{1}$$

The category of each $a_i \in A$ is found by comparing it with the successive profiles. According to the pessimistic version (the most used one), each action $a_i \in A$ is assigned to the highest category C^h such that a_i outranks its lower-bound profile (b^{h-1}). This is equivalent to:

$$a_i \to C^h \quad \text{iff} \quad s(a_i,b^{h-1}) \geq \lambda \quad \text{and} \quad s(a_i,b^h) < \lambda, \tag{2}$$

since in ELECTRE TRI the profiles verify the condition that

$$s(a_i,b^h) < \lambda \Rightarrow s(a_i,b^{h+1}) < \lambda, \; \forall \, a_i \in A, \; k=0,\ldots,k-1.$$

2.1 Computation of credibility indices *s(a,b)*

a) Computation of single-criterion concordance

Let us use D_{jab} to denote the *advantage* of an action a over another action b on criterion g_j:

$$\Delta_{jab} = \begin{cases} g_j(a) - g_j(b), & \text{if criterion } g_j \text{ is to be maximized (the more the better)} \\ -g_j(a) + g_j(b), & \text{if criterion } g_j \text{ is to be minimized.} \end{cases}$$

The concordance index of criterion g_j regarding the assertion $a \, S \, b$ is computed as follows:

$$c_j(a,b) = \begin{cases} 0, & \text{if } \Delta_{jab} < -p_j \\ (\Delta_j + p_j)/(p_j - q_j), & \text{if } -p_j \leq \Delta_{jab} < -q_j \\ 1, & \text{if } \Delta_{jab} \geq -q_j \end{cases}$$

The criterion fully agrees with $a \, S \, b$ whenever the advantage D_{jab} is positive or, if negative, when the disadvantage (i.e., $-D_{jab}$) does not exceed the criterion's *indifference threshold* q_j. Concordance is null if the disadvantage reaches or exceeds the *preference threshold* p_j. This single-criterion concordance index varies linearly in between these two thresholds.

b) Computation of global discordance

The single-criterion concordance indices are aggregated into an overall *(multicriteria) concordance index* by the operation

$$c(a,b) = \sum_{j=1}^{n} k_j . c_j(a,b)$$

where k_j (a non-negative value) represents the *importance coefficient* ("weight") of the criterion g_j (j=1, ..., n). For a matter of normalization, and without loss of generality, the sum of the weights $k_1,...,k_n$ should be 1.

c) Computation of single-criterion discordance

The discordance index of criterion g_j regarding the assertion $a\ S\ b$ is computed as follows:

$$d_j(a,b) = \begin{cases} 0, & \text{if } -\Delta_{jab} \le u_j \\ \left(-\Delta_j - u_j\right)/\left(v_j - u_j\right), & \text{if } u_j < -\Delta_{jab} \le v_j \\ 1, & \text{if } -\Delta_{jab} > v_j \end{cases}$$

Discordance is null whenever the advantage D_{jab} is positive or, in case it is negative, when the disadvantage (i.e., $-D_{jab}$) does not exceed the criterion's *discordance threshold* u_j. The criterion fully disagrees with $a\ S\ b$ (it "vetoes" that conclusion) whenever the disadvantage exceeds its *veto threshold* v_j. This single-criterion discordance index varies linearly in between these two thresholds. This variant described here was proposed by (Mousseau and Dias, 2004), who suggest to consider $u_j = 0.25\ p_j + 0.75\ v_j$ in the cases where the DM does not want to deal explicitly with parameter u_j.

d) Computation of credibility

The *credibility index* for the assertion $a\ S\ b$ is computed by the following expression (Mousseau and Dias, 2004):

$$s(a,b) = c(a,b).\left[1 - d^{max}(a,b)\right], \quad \text{with } d^{max}(a,b) = \max_{j \in \{1,...,n\}} d_j(a,b).$$

If there exists any criterion such that $D_{jab} \le -v_j$ (i.e. if there exists a veto), $s(a,b)$ becomes null.

3 Interactive construction of a sorting model

(Dias *et al.* 2002) have proposed an interactive process to help a DM in the progressive construction of an ELECTRE TRI sorting model. It aims at supporting the DM in setting values for the weights $k_1,...,k_n$ and the cutting level λ, assuming the DM has already fixed the value of

the remaining parameters. Indeed, the weights and cutting level are often considered the most difficult to fix, since they cannot be set independently of each other. The process combined the ideas of parameter inference (Mousseau and Slowinski 1998) and robustness analysis (Dias and Clímaco 2000), considering there existed no discordance ($v_j = +\infty$). Later, (Mousseau and Dias 2004) have proposed a variant of the outranking relation that allowed incorporating the concepts of discordance and veto, as presented in Section 2.

At a given iteration of the interactive process let R denote the set of constraints on $(\lambda,k_1,...,k_n)$ as indicated by the DM, defining a set T of combinations of parameter values deemed acceptable in that iteration. These constraints can be any linear equalities or inequalities, including for instance:

- an interval of values for the cutting level λ;
- an interval of values for each weight k_j;
- comparisons involving the weight of criteria coalitions, e.g., $k_1 \geq k_3$, $k_1 \geq k_2+k_4$;
- limits to the assignment of actions to categories (sorting examples), e.g.: stating that a_1 belongs to category C^3 originates (from (2)) the constraints $s(a_1,b^2)-\lambda \geq 0$ and $\lambda-s(a_1,b^3) \geq \varepsilon$ (ε is an arbitrary very small positive value that is necessary because the inequality in (2) is strict); stating that a_2 belongs to categories C^3 or C^4 originates the constraints $s(a_2,b^2)-\lambda \geq 0$ and $\lambda-s(a_2,b^4) \geq \varepsilon$; stating that a_3 belongs to category C^1 originates only one constraint $\lambda-s(a_3,b^1) \geq \varepsilon$; etc. Each of these constraints is linear, since only the weights and the cutting level are variables (Mousseau and Dias 2004).

The interactive process may start with an empty set of constraints, adding one or two constraints at the time as the DM increases his/her insight on the problem and becomes increasingly confident about his/her judgment. By proceeding in this manner, the constraints will usually define a consistent system admitting at least one solution. However, the possibility of reaching an inconsistent system of constraints is also foreseen. The results presented and the analysis allowed will vary according to the (in)consistency of the constraints.

3.1 When the constraints define a consistent system

Whenever the set of constraints R defines a consistent system, there will exist at least one combination of values for $(\lambda,k_1,...,k_n)$ that respects all of R's constraints. Let T denote the set of such combinations. IRIS is then able to compute (infer) a combination from T and determine the sorting of the actions corresponding to it. Furthermore, it tells the DM how different could

the sorting be without violating any constraint. Hence, the DM may either feel confident to accept the sorting suggested by IRIS, or may choose to proceed aiming at reducing set T by adding new constraints. In the latter case, IRIS is also able to help. The complete set of outputs from IRIS when R is a consistent system is listed next:

- Using linear programming (for details see (Mousseau and Dias 2004)), IRIS computes a combination of parameter values form T that maximizes the minimum slack concerning R's constraints.
- For each action, IRIS indicates the category where it is sorted according to the inferred parameter values.
- For each action, IRIS shows the range of categories where the action could have been sorted without violating any constraint. This allows observing which of the proposed sorting results are more affected by the existing imprecision. It also allows to help the DM choose sorting examples that do not contradict the constraints previously stated by him/her. Finally, it also allows drawing robust conclusions (i.e. conclusions that hold true for all the acceptable combinations of parameter values), e.g., "a_1 belongs to category C^3 or higher", or "a_1 reaches, at most, category C^4".
- For each action→category assignment, IRIS may compute (infer) a combination of parameter values, if exists, that leads to that assignment. The chosen combination maximizes the minimum slack associated with the assignment-related constraints. The DM may then provide new explicit constraints on the parameter values, namely when he/she wishes to exclude values leading to extreme assignments (first or last from a range) that he/she considers inadequate.
- Using Monte-Carlo simulation, IRIS estimates the relative volume of polyhedron T, which can be seen as an indicator of the input's precision (it indicates the proportion of combinations that is considered acceptable given the constraints). IRIS also computes the geometric average of the number of categories where each action may be assigned to given the constraints, which can be seen as an indicator of the output's precision. The main interest of these "proxy" indicators (others might as well have been included) is to observe how they change from one iteration to another, rather than their absolute value.

3.2 When the constraints define an inconsistent system

Whenever the set of constraints R defines an inconsistent system, there will exist no combination of values for $(1,k_1,...,k_n)$ that respects all of R's constraints (i.e., $T=Æ$). IRIS will

still infer a combination of parameter values and will show the sorting corresponding to it, although some of the constraints are violated. The analysis should focus on how to remove the inconsistency among the set of constraints. The complete set of outputs from IRIS when R is an inconsistent system is listed next:

- Using linear programming, IRIS computes a combination of parameter values form T that minimizes the maximum deviation concerning R's violated constraints.
- For each action, IRIS indicates the category where it is sorted according to the inferred parameter values, highlighting the sorting examples that were not reproduced.
- For each constraint, IRIS indicates whether it is respected or violated and, in the latter case, computes the deviation.
- IRIS includes an inconsistency analysis module to find subsets of constraints from R that, if removed, would render the system consistent. These suggestions are presented by cardinality order: first subsets containing only one constraint, then subsets containing two constraints, and so on. This involves solving a series of 0-1 programs (see (Mousseau *et al.* 2003)), but these details are hidden from the user. Among the subsets of constraints suggested by IRIS, the DM should abdicate from one, knowing that after removing the constraints in this subset from R the system will become consistent again. By not limiting itself to present a single proposal that would minimize the number of constraints to remove, IRIS takes into account the possibility that the DM attaches different priorities to the constraints.

3.3 Interactive process

IRIS proposes a process to construct an ELECTRE TRI sorting model, which is interactive to the extent that the results from a given iteration can be used to guide the DM in revising the inputs for the following iteration. This process may start with scarce information, i.e. wide intervals for all the variables $(1,k_1,...,k_n)$, and none or few additional constraints (including sorting examples). At each iteration, the DM ought to change the information provided minimally, by adding, modifying, or deleting one or a few constraints at a time. The fast response times by IRIS allow the DM to observe immediately the effects of the changes, which will be better understood by making such small incremental steps. This will allow the DM to learn continuously about the problem at hand and how the method works, hence allowing him/her to progress from iteration to iteration.

The purpose of the interactive process is to progressively reduce the set T of acceptable combinations for the parameters $(1,k_1,...,k_n)$, although one cannot expect any kind of

convergence. Indeed, the DM may proceed by trial and error, placing constraints that will later be removed. The DM may stop the process when he/she feels that the precision of the outputs (as perceived from the sorting ranges) is satisfactory to his/her purpose, and he/she feels confident and comfortable with the constraints imposed on *T*.

The tangible products of this interactive process are:

- a set of sorting examples and other constraints defining the acceptable values for the parameters $(l,k_1,...,k_n)$;
- a precise combination of values $(l^*,k_1^*,...,k_n^*)$, resulting from the inference program, that defines an ELECTRE TRI sorting model;
- a precise category or a range of categories for each action to sort, which is robust given the information provided (i.e., given the acceptable parameter values, assignments outside these ranges are not possible).

However, the most important result from the interactive process may well be an increased insight of the DM on the decision problem faced by him/her, and an increased knowledge about his/her preferences, that may even have changed during the process.

4 IRIS user interface

The IRIS 2.0 software (see (Dias and Mousseau, 2003)) runs under the Microsoft Windows operating system (version 95 or later). A demonstration version is downloadable at: `http://www4.fe.uc.pt/lmcdias/iris.htm`. The left part of the screen is associated with inputs, whereas the right part of the screen is associated with outputs (Fig. 2), and the user may drag the line dividing these two areas. Each of these areas is organized according to a "notebook with multiple tabs" metaphor.

The area on the left is used to edit the inputs, namely:

- the performances of the actions to sort and the indication of sorting examples (*Actions* page);
- the values for the fixed parameters, which are categories' profiles and the indifference, preference, discordance and veto criteria thresholds (*Fixed Par.* page);
- the upper and lower bounds for the variables $(l,k_1,...,k_n)$ (*Bounds* page); and
- additional constraints to the value of these variables (*Constraints* page).

The results will reflect changes in the inputs after the user instructs IRIS to compute them. The right area shows the multiple results, namely:

- sorting ranges, inferred sorting, and inferred parameter values (*Results* page);
- list of constraints for the inference program and optimal deviations (*Infer. Prog.* page); and
- geometric mean of the number of possible categories for each action (*Indices* page).

The menu options (pull-down menu at the top and context-sensitive "pop-up" menus) include the usual commands for data editing, file management and user help. Of particular interest are the commands to split a category in two, to merge consecutive categories, to enable the explicit use of discordance thresholds u_j (if this option is off, IRIS considers $u_j = 0.25\ p_j + 0.75\ v_j$), and to compute the volume of the polytope formed by the combinations of parameter values that respect all the constraints. Data can be stored and retrieved as text files, allowing simple processes for the transference of data tables from word processors or spreadsheets

{Figure 2 – Fixed parameters and initial results sorted by variability.}

5 An illustrative example

In order to illustrate the use of IRIS, we will follow in this section the steps of an hypothetical DM, considering data from a problem presented in (Dias *et al.* 2000), which was adapted from an application of ELECTRE TRI to the banking sector presented in (Dimitras *et al.* 1995). The data differ from that presented in (Dias *et al.* 2000) only in the use of a veto threshold in one of the criteria (left part of Fig. 2), whereas the cited paper considered no veto.

The decision problem consists in sorting 40 actions (each one representing a company) into categories of bankruptcy risk: very high (C^1), high (C^2), medium (C^3), low (C^4), or very low (C^5). These five categories are ordered from worst (C^1) to best (C^5). The left part of Fig. 2 presents the profiles that divide the categories (their performances and fixed thresholds on seven criteria). Each of the actions to be sorted was evaluated in seven criteria (left part of Fig. 3).

First iteration:

The variables in this problem are the parameters the DM needs to set: the cutting level and weights $(1,k_1,...,k_7)$. Initially, the DM placed wide intervals in the *Bounds* page: she placed $k_j \in$ [0.01, 0.49] ($j=1,...,7$) (according to theory, these weights must be positive and no criterion should weigh more than 0.5), and she placed $\lambda \in$ [0.6, 0.99] (according to theory, the cutting level may vary between 0.5 and 1.0). Additionally, the DM informed that the second criterion was the one with highest weight. Therefore, the constraints $k_2 \geq k_1$, $k_2 \geq k_3$, $k_2 \geq k_4$, $k_2 \geq k_5$, $k_2 \geq k_6$ and $k_2 \geq k_7$ were inserted in page *Constraints*. No sorting example was considered at this stage.

The right part of Fig. 2 illustrates the results corresponding to this (relatively scarce) information. Since there are many combinations of values for $(1,k_1,...,k_7)$ that satisfy the few constraints that were introduced, IRIS shows for each action a range of categories where it may be sorted without violating any constraint. This example (which shows the ranges sorted by variability), presents the curiosity of showing that action a_{28} cannot be assigned to category C^2, although it could be assigned to categories C^1 or C^3 (for a characterization of such situations see (Dias *et al.* 2000)). Among the categories in each range, a darker color is used to identify the sorting proposed by IRIS based on the inferred values for $(1,k_1,...,k_n)$, which are shown in the last line of the *Results* page. In the line immediately above it IRIS presents the combination of values for these parameters that corresponds to the selected cell (Fig. 2 shows a situation where the selected cell corresponds to the assignment of a_{28} into C^4). The latter combination of parameter values changes immediately when the user selects a different cell.

Given the current information, the *Indices* page would state that the (geometric) average number of possible categories per action was 2.249, which is an indicator of the precision from the perspective of outputs. The proportion of combinations respecting all the bounds that also respect the additional constraints (given by the menu command *Volume computation*) is 14.3%, which is an indicator of the precision from the perspective of inputs. The DM may monitor how these figures change from one iteration to the next one.

Second and third iterations:

Let us now imagine that the DM felt confident that action a_{28} was not worse than category C^4. Hence, she inserts in page *Actions* this assignment example, which is an interval-type one since it admits two categories: C^4 and C^5. The results are depicted in Fig. 3. The *Indices* would state that the (geometric) average number of possible categories per action is now 1.518, whereas the result of the *Volume computation* command drops to 0.9%.

Next, in a third iteration, the DM decides to indicate that action a_{31} is a good representative for the worst category (C^1), and that action a_3 should only be allowed into categories C^3 or C^4. The results are depicted in Fig. 4. The result of the *Volume computation* command decreased to 0.6%, indicating a further reduction of the set of combinations of parameter values respecting all the constraints. A visible consequence of this reduction is, for instance, the fact that actions a_{10} and a_{11} can no longer be assigned to category C^5.

{Figure 3 – Results after introducing one sorting example (iteration 2)}

{Figure 4 – Results after introducing two additional sorting examples (iteration 3).}

Fourth and fifth iteration:

So far, the constraints introduced by the DM did not contradict the constraints previously introduced: the system of constraints remained consistent. However, sometimes inconsistency is introduced inadvertently (e.g. when placing several constraints at a time) or to question previously introduced constraints. For instance, the DM may wonder why action a_6 cannot be sorted into category C^3 or lower. By indicating this sorting example the system becomes inconsistent, since there is no combination of values for $(1,k_1,...,k_n)$ capable of satisfying all the constraints. In this situation IRIS shows an inferred combination of parameter values that violates one or more constraints by the least possible deviation, and indicates the sorting corresponding to it, highlighting the sorting examples that were not reproduced (Fig. 5).

{ Figure 5 – A new constraint makes the system inconsistent (fourth iteration).}

{Figure 6 – Inconsistency analysis module (fourth iteration).}

In these circumstances, the DM must relax or remove constraints. The inconsistency analysis module tells her which possibilities exist to render the system consistent (Fig. 6): either the constraint number 3 is removed (this is the constraint just inserted, which caused the inconsistency), or the constraints number 1, 4 and 13 are removed. Constraints 1 and 4 correspond to sorting examples introduced by the DM, but constraint 13 corresponds to the cutting level's upper bound, currently set to 0.99. Since this bound is already rather high, the DM must conform to remove the constraint of sorting a_6 in a category lower than C^4, returning to the situation depicted in Fig. 4.

In a subsequent iteration (the fifth), the DM wonders why action a_9 cannot be sorted into category C^3 or lower. By indicating this sorting example the system becomes inconsistent again, and the inconsistency analysis module helps the DM decide how to restore its consistency. In this case there are again two possibilities: either remove the constraint concerning a_9 or remove the constraint stating a_{28} belongs to C^4 or higher. The DM feels stronger about the constraint concerning a_9, hence decides to drop the other constraint. As a matter of fact, she relaxes the constraint stating that a_{28} belongs to C^3 or higher, to see if this adjustment suffices. The system actually becomes consistent and the results corresponding to it are depicted in Fig. 7.

{Figure 7 – Results after relaxing the sorting example concerning a_{28}.}

We will stop here our illustration of the use of IRIS. At this stage, 27 actions are sorted into a single category, whereas each of the remaining 13 actions can be assigned at most into two categories. The DM may be satisfied with these results, or she might want to continue either adding information or revising her previous judgment.

6 Applications of IRIS in real world decision problems

In this section, we provide two illustrations of real world decision problems in which the IRIS software has been playing a significant role in the modeling process. These are ongoing applications whose detailed description lays outside the scope of this paper. Both applications involve the definition of a multiple criteria sorting model using the ELECTRE TRI method.

6.1 Deciding whether to answer to a call for tender (CfT) at EADS.

EADS Launch Vehicles (EADS-LV) developed and uses since 1996 an ELECTRE TRI model to support the decision concerning whether to answer to a CfT (see (Collette and Siarry 2002), chpt. 12). For EADS-LV, answering to a CfT implies spending some time conceiving a commercial offer; this generates a cost that will be integrated in the contract in case of success, or be considered as a loss in unsuccessful cases. Choosing not to answer to a CfT will neither generate gains neither losses.

The model developed aims at sorting CfT into 4 ordered categories ("*Yes*", "*Undecided yes*", "*Undecided no*", "*No*"). The "*Yes*" category (the "*No*" category, respectively) groups CfT for which EADS-LV is (is not, respectively) in position of success. The "*Undecided Yes*" and "*Undecided No*" categories correspond to weaker statements. It is used weekly by a committee in order to make a decision for the current CfT. Since 1996, a large database of more than 400 CfT has been collected including, for each CfT, the evaluation vector, the assignment provided by ELECTRE TRI, the decision made by the committee (either answer or not to the CfT) and the actual output of the CfT (Success or Loss).

An important issue for EADS is to check whether the ELECTRE TRI model has a good predictive ability concerning the result of the CfT (success or not). Therefore, the model is to be regularly updated in order that the output of the ELECTRE TRI model becomes as close as possible to the actual output of the CfT on the past data. More precisely, CfT sorted in the categories "*Yes*" or "*Undecided yes*" should lead to a success while CfT sorted in the categories "*No*" or "*Undecided no*" should lead to a Loss.

In order to update the ELECTRE TRI model concerning the weights of criteria, IRIS is a efficient tool to account for past data. Each past CfT can induce an assignment example in a way that successful CfT should be assigned at least to the category "*Undecided Yes*" and unsuccessful CfT should be assigned at most to the category "*Undecided No*". The following fictitious example illustrates the way these assignment examples are designed. The assignment examples a_1, a_4, a_5 do not contradict the original assignment model, but a_2 and a_3 provide situation that should be restored by the inferred model.

{Table 1 : assignment examples induced by 5 past CfT}

6.2. Integrating multiple expert opinions into a single sorting model.

This application concerns the public transport pricing in the Paris region (Ile de France). STIF is the organizing authority, in charge of operating public transport networks and deciding on ticket prices and price structures.

STIF is willing to modify the pricing of public transport in Ile de France. With the support of LAMSADE, STIF has been developing a methodology (see (Mousseau *et al.* 2001)) aiming at defining alternative pricing strategies, in cooperation with the involved stakeholders. Strategies are grounded on the partition of Ile de France into zones, the price being defined from a zone to another on the basis of the distance and the quality of the public transport in the origin and destination zones. Therefore, a model specifying the quality of the public transport offer is required.

A multiple criteria sorting model based on the ELECTRE TRI method has been build to assign a qualitative level for the quality of the public transport offer (4 qualitative levels C^1:"very low offer", C^2:"fairly low offer", C^3: " fairly high offer" and C^4: "very high offer"). The 11 criteria considered correspond to the different aspects (number of stations, frequency, accessibility …) of the public transport network. Experts could easily define the limits of categories (and corresponding thresholds), but weighting directly the criteria has been considered as a difficult task, therefore an indirect procedure has been proposed.

Based on their knowledge, 7 experts have rated the level of quality of 58 zones according to their evaluations on the criteria. Obviously, the rating (C^1, C^2, C^3 or C^4) provided by an expert for a specific zone corresponds to an assignment example. Within this context, IRIS has proved to be a very efficient tool to analyze the information contained in the 406 (58x7) assignment examples provided by the experts.

First, IRIS made it possible to check for inconsistencies in the rating of the 58 zones for each expert individually. Most experts did not provide ratings perfectly compatible with ELECTRE TRI; hence it was possible to determine the minimal subset of assignment examples that when removed lead to an information compatible with ELECTRE TRI. This analysis led each expert to define "reasonable" values for his/her criteria weights.

Second, an important issue to integrate multiple expert judgments into a single sorting model is to analyze the compatibility among judgments provided by coalitions of experts. To proceed to such analysis, IRIS was used considering the assignment examples provided by subsets of *n*

experts (*i.e., n**58 examples), and computing the minimal subset of assignment examples that when removed lead to a consistent information. The smaller this subset is, the more compatible the expert judgments are. Moreover, for each coalition of experts, a weight vector was computed (using the inference procedure available in IRIS).

Third, a Delphi-like iterative procedure was used to facilitate "convergence" of assignments examples provided by the 7 experts. Three iterations were performed in which experts were asked to reconsider their opinion on the zones for which no unanimity arose at the preceding iteration (a feedback on the preceding iteration was provided). Within this iterative context, IRIS proved to be very powerful to analyze dynamically the evolution of coalitions of experts and how a consensus among experts arises.

7 Conclusions

IRIS was built to support DMs facing sorting problems. It does not apply to situations where the DM wishes to automatically sort the actions based on their characteristics exclusively. Rather, it applies to situations where the preferences of the DM — as well as the characteristics of the actions — will yield a partition of the set of actions into a set of ordered categories defined beforehand. From another perspective, it applies to situations where the DM wishes to attribute grades to actions, defining the grades beforehand and grading the actions independently of each other. The important issue to stress is that the subjective values of the DM will influence the outcome of the sorting decisions.

The main advantage of IRIS is the support it may offer to DMs that do not entirely know their preferences, or that do not know how to quantify these preferences taking into account the meaning of ELECTRE TRI's parameters and using precise numbers. By accepting imprecise information (i.e., intervals or other constraints, including sorting examples) IRIS integrates the inference of parameter values with the search for conclusions that are valid despite the imprecision (the robust conclusions). It facilitates an interactive process that fosters self-learning and the progressive delimitation of the inputs and outputs' variability.

The main shortcoming of IRIS is not to consider all the parameters as variables, leaving the DM unsupported to set the value of the category profiles and criteria thresholds (indifference, preference, discordance, and veto). However, the parameters it considers as variable (weights and the cutting level) are arguably the most difficult to set, given their interdependence across the criteria (whereas the remaining parameters require only single-criterion judgments).

One of the paths to pursue in future research is precisely to study how to solve efficiently inference problems where more parameters are allowed to vary. Another ongoing path is to improve the inconsistency analysis module so that it may propose to the DM different forms of relaxing (rather than removing) constraints, possibly taking into account different priority levels attached to constraints. Finally, real world applications such as the ones described in the previous section will surely cause new challenges to be addressed in future versions of IRIS.

As a final remark, we wish to emphasize the importance of interactivity is DSS. Interactivity, as noted by Courbon (Courbon *et al.* 1994), should not be seen as a last minute concern, but as a conception tool for DSSs. In a decision process supported by IRIS, the model is successively shaped in an interactive manner. In a way, as suggested by Courbon in the cited paper, a DSS may help a DM (since, in our case, it explores the model's imprecision and extracts conclusions), but it may also be helped by the DM (to the extent that, in our case, he/she continuously monitors and delimits the model's imprecision).

Acknowledgments

The methodology IRIS is based on was developed in the framework of two Portuguese-French cooperation projects financed by ICCTI and the French Embassy in Portugal (ref. 328J4 and 500B4). The software specification was performed by Luís Dias and Vincent Mousseau, and the software implementation was conducted by Luis Dias, Carlos Gomes da Silva and Rui Lourenço.

The authors thank the referees for their suggestions to amend the paper.

References

Courbon JC, Dubois D, Roy B, Pomerol JC. 1994. Autour de l'aide a la décision et de l'intelligence artificielle, *Cahiers du LAFORIA* 94/01, Institut Blaise Pascal, Paris.

Collette Y, Siarry P. 2002. *Optimisation multiobjectif* , Eyrolles.

Dias LC, Clímaco JN. 2000. ELECTRE TRI for groups with imprecise information on parameter values, *Group Decision and Negotiation* 9, 355-377.

Dias L, Mousseau V, Figueira J, Clímaco J. 2000. An Aggregation/Disaggregation Approach to obtain Robust Conclusions with ELECTRE TRI, *Cahier du LAMSADE*, No. 174, Université Paris-Dauphine.

Dias L, Mousseau V. 2003. IRIS – **I**nteracive **R**obustness analysis and parameter's **I**nference for multiple criteria **S**orting problems (version 2.0) – User Manual, Document of INESC Coimbra, 1/2003, available at http://www4.fe.uc.pt/lmcdias/IrisMan2.pdf.

Dias L, Mousseau V, Figueira J, Clímaco J. 2003. An aggregation/disaggregation approach to obtain robust conclusions with ELECTRE TRI, *European Journal of Operational Research* 138, 332-348.

Dimitras A, Zopounidis C, Hurson C. 1995. Multicriteria decision aid method for the assessment of business failure risk, *Foundations of Computing and Decision Sciences*, 20(2), 99-112.

Greco S, B Matarazzo, R Slowinski. 2001. Rough sets theory for multicriteria decision analysis. *European Journal of Operational Research* 129, 1-47.

Mousseau V, Dias L. 2004. Valued outranking relations in ELECTRE providing manageable disaggregation procedures, *European Journal of Operational Research*, 156 (2), 467-482.

Mousseau V, Figueira J, Dias L, Clímaco J, Gomes da Silva C. 2003. Resolving inconsistencies among constraints on the parameters of an MCDA model, *European Journal of Operational Research* 143, 332-348.

Mousseau V, Roy B, Sommerlatt I. 2001. Development of a decision aiding tool for the evolution of public transport ticket pricing in the Paris region. In M. Paruccini A. Colorni and B. Roy, editors, *A-MCD-A Aide Multicritère à la Décision - Multiple Criteria Decision Aiding*. Joint Research Center, European Commision, Luxembourg, 2001, 213-230.

Mousseau V, Slowinski R. 1998. Inferring an ELECTRE TRI model from assignment examples, *Journal of Global Optimization*, 12, 157-174.

Mousseau V, Slowinski R, Zielniewicz P. 1999. A user-oriented implementation of the ELECTRE-TRI method integrating preference elicitation support, *Computers & Operations Research*, 27, 757-777.

Roy B. 1991. The outranking approach and the foundations of ELECTRE methods, *Theory and Decision* 31, 49-73.

Roy B. 1996. *Multicriteria Methodology for Decision Analysis*, Kluwer.

Roy B, Bouyssou D. 1993. *Aide multicritère à la décision: méthodes et cas*, Economica, Paris.

Yu W. 1992. ELECTRE TRI. Aspects méthodologiques et guide d'utilisation, *Document du LAMSADE*, No. 74, Université Paris-Dauphine.

Zopounidis C, Doumpos D. 2002, Multicriteria classification and sorting problems: a literature review, *European Journal of Operational Research* 138, 229-246.

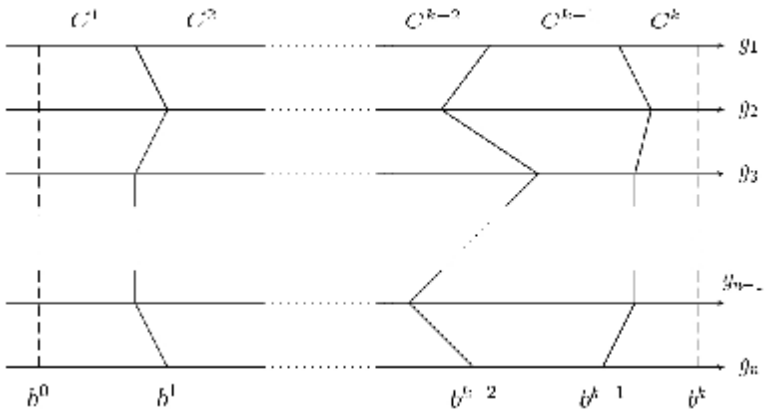

Figure 1 – Definition of categories using profiles.

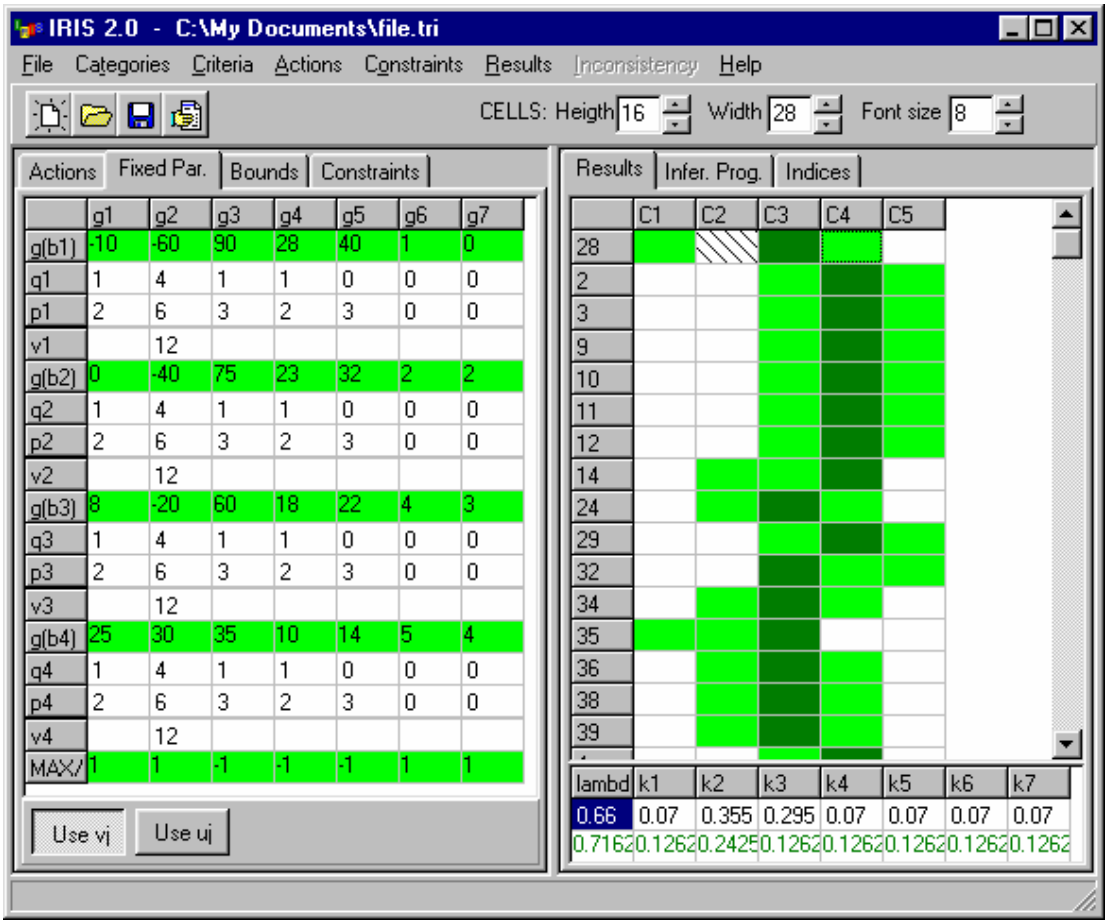

Figure 2 – Fixed parameters and initial results sorted by variability.

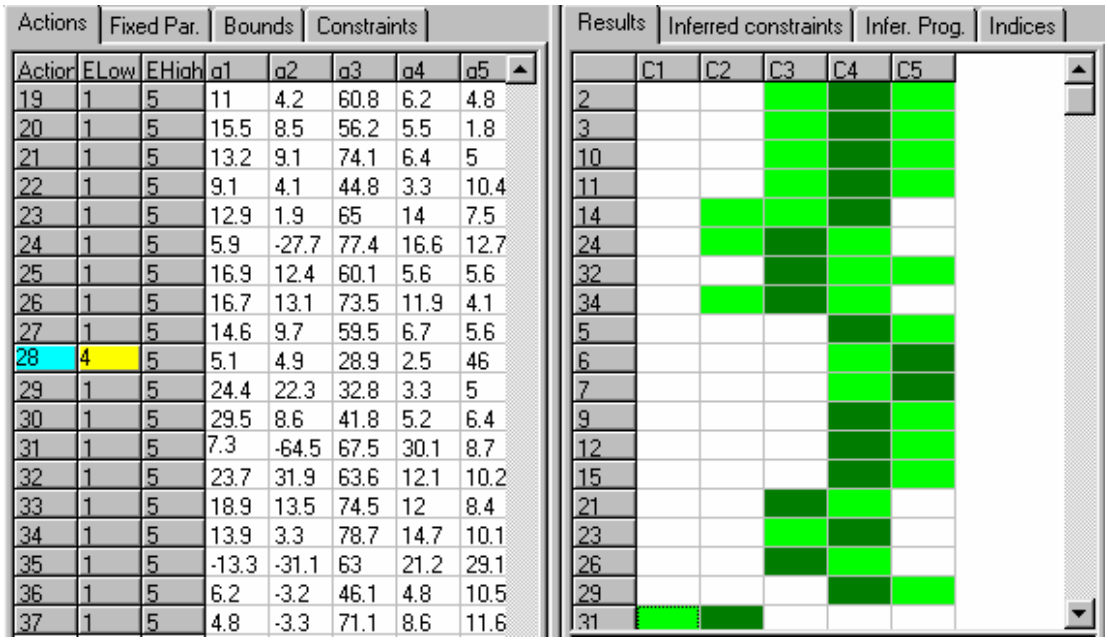

Figure 3 – Results after introducing one sorting example (iteration 2).

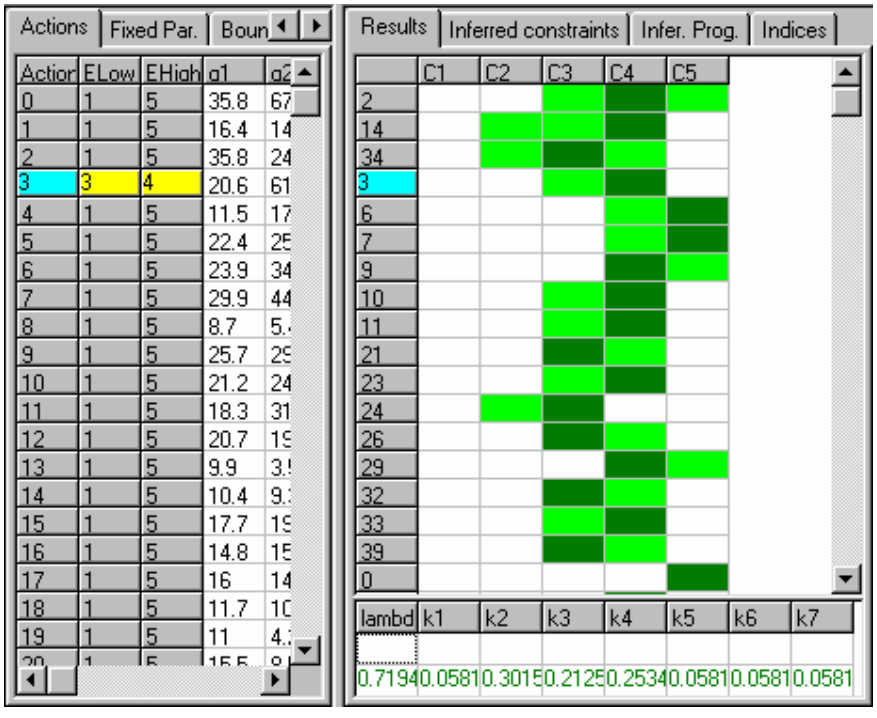

Figure 4 – Results after introducing two additional sorting examples (iteration 3).

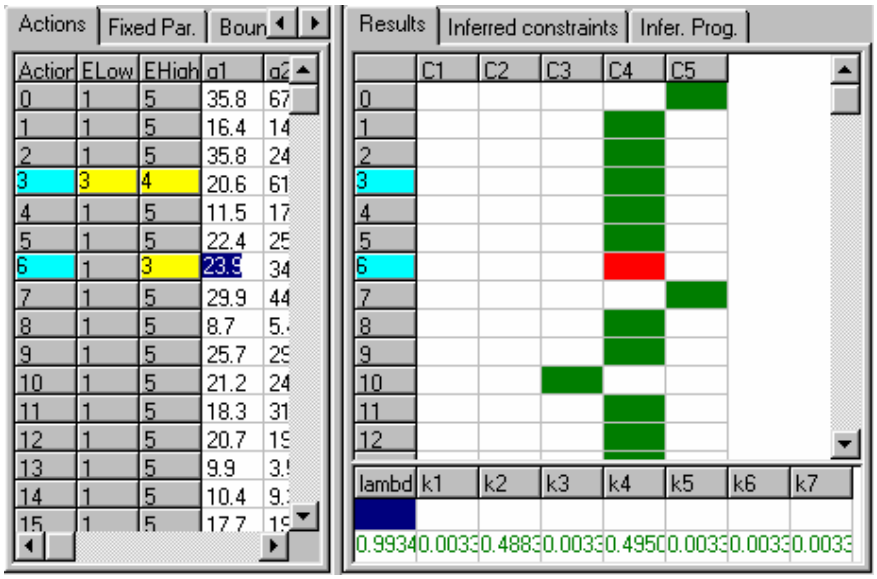

Figure 5 – A new constraint makes the system inconsistent (fourth iteration).

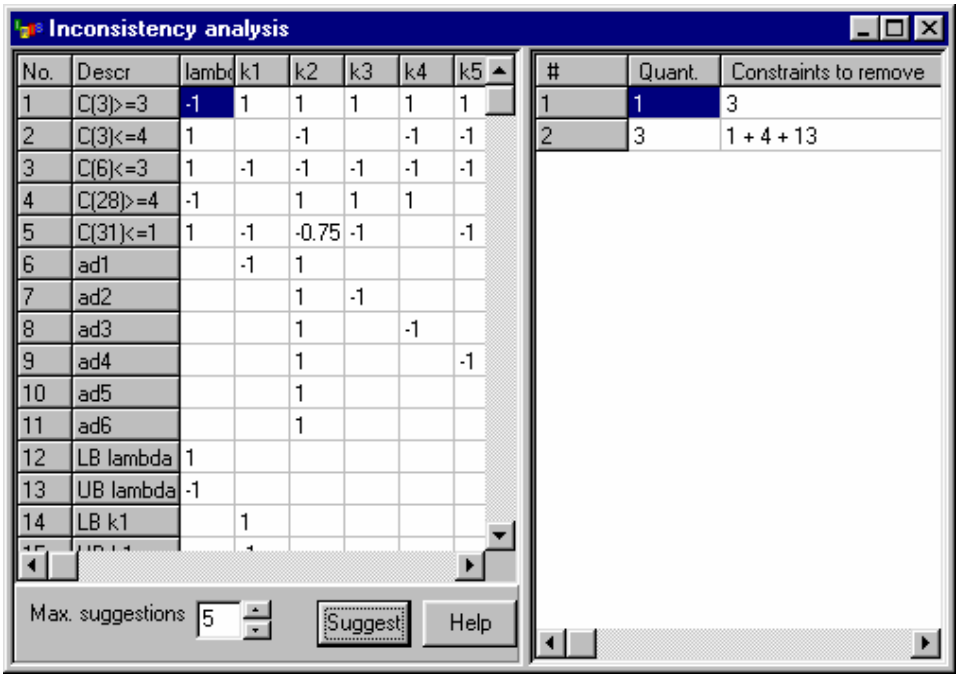

Figure 6 – Inconsistency analysis module (fourth iteration).

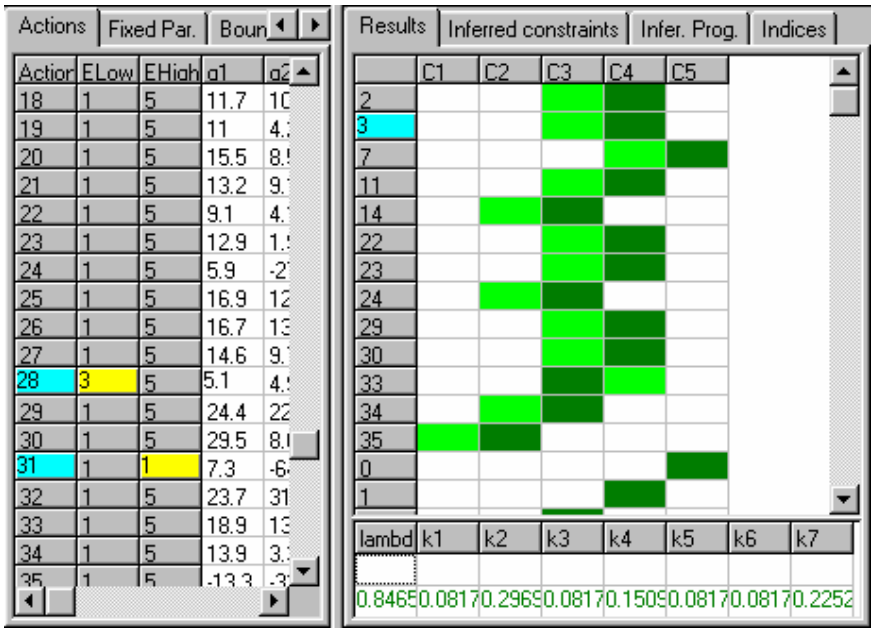

Figure 7 – Results after relaxing the sorting example concerning a_{28}.

Name	Original model assignment	Actual Result	Minimum category	Maximum category
$\mathbf{a_1}$	"*Undecided Yes*"	Successful	"*Undecided yes*"	"*Yes*"
$\mathbf{a_2}$	"*Undecided Yes*"	Unsuccessful	"*No*"	"*Undecided No*"
$\mathbf{a_3}$	"*Undecided No*"	Successful	"*Undecided yes*"	"*Yes*"
$\mathbf{a_4}$	"*Yes*"	Successful	"*Undecided yes*"	"*Yes*"
$\mathbf{a_5}$	"*No*"	Unsuccessful	"*No*"	"*Undecided No*"

Table 1 : assignment examples induced by 5 past CfT