Instituto de Engenharia de Sistemas e Computadores

Institute of Systems Engineering and Computers

INESC Coimbra

IRIS - Interactive Robustness analysis and parameters' Inference

for multicriteria Sorting problems (Version 2.0)

User Manual

Luís DIAS and Vincent MOUSSEAU

INESC Coimbra

LIMITED WARRANTY AND DISCLAIMERS

ORDERS

IRIS - Interactive Robustness analysis and parameters' Inference for multicriteria Sorting problems

Version 2.0

User Manual

Luís DIAS[1,2] and Vincent MOUSSEAU[3]

(1) INESC Coimbra
Rua Antero de Quental, 199
3000-033 Coimbra, PORTUGAL

(2) Faculdade de Economia, Universidade de
Coimbra,
Av. Dias da Silva, 165,
3004-512 Coimbra, PORTUGAL

(3) LAMSADE, Université Paris-Dauphine
Place du Maréchal De Lattre de Tassigny,
75775 Paris Cedex 16, FRANCE

Abstract: This document is the User Manual for the decision support software IRIS - Interactive Robustness analysis and parameters' Inference for multicriteria Sorting problems (Version 2). This tool has been built to support the assignment of actions (alternatives, projects, candidates) described by their evaluation (performance) at multiple dimensions (criteria) to a set of predefined ordered categories, using a pessimistic concordance-only variant of the ELECTRE TRI method. Rather than demanding precise values for the ELECTRE TRI parameters, IRIS allows to enter constraints on these values, namely assignment examples that it tries to restore. It adds a module to identify the source of inconsistency among the constraints when it is not possible to respect all of them at the same time. On the other hand, if the constraints are compatible with multiple assignments for the actions, IRIS allows to draw robust conclusions by indicating the range of assignments (for each action) that do not contradict any constraint.

Index

1. Getting started, Obtaining the IRIS Software

IRIS runs on Windows 95/98/Me/XP computers. The monitor should be at least VGA (640x480) with 16 colors. It occupies little space on disk and is not too demanding in terms of RAM (however, the more the better...). The program may run without a mouse, but becomes somewhat cumbersome to use. You should have a 2-button mouse to make the best use of this software.

To install IRIS unzip the contents of the file **iris2.zip** to a new folder with a location and a name of your choice.

IRIS may be purchased form INESC Coimbra, a non-profit Portuguese R&D institute owned by the University of Coimbra and INESC:

INESC Coimbra	Fax: +351 239 824692
Rua Antero de Quental, 199	Phone: +351 239 851040
3000-033 Coimbra	
Portugal	C/O Luis M.C. Dias
	LDias@inescc.pt

IRIS page in the Internet: **www4.fe.uc.pt/lmcdias/iris.htm**

2. A brief overview of IRIS

IRIS is a Decision Support Software designed to address the problem of assigning a set of actions to predefined ordered categories, according to their evaluations (performances) at multiple criteria. For instance, it may be used to sort funding requests according to merit categories (e.g. "Very good", "Good", "Fair", "Not eligible"), or to sort loan applicants into categories (e.g. "Accept", "Require more collateral", "Reject"), or to sort employees in a company into categories that define incentive packages, etc.

IRIS implements a methodology developed by Luis Dias, Vincent Mousseau, José Figueira and João Clímaco, presented in Dias et al. (2002) (see also Section 3), which is based on the ELECTRE TRI method (see Roy and Bouyssou (1993) and Yu (1992)). The inconsistency analysis method is presented in Mousseau et al. (2003) (see also Section 3).

The main characteristics or IRIS are:

- IRIS implements the pessimistic ELECTRE TRI, using a variant of the original function to compute discordance (veto effects), as proposed by Mousseau and Dias (2002).

- IRIS accepts imprecision concerning the criteria weights and the cutting level. The users may indicate intervals for each of these parameters, as well as linear constraints on the weights. Furthermore, the constraints may be defined indirectly, as indicated in the next item.

- IRIS accepts assignment examples, where the users indicate minimum and maximum categories for some of the actions, according to their holistic judgment. These assignment examples are translated into constraints on the parameter values, meaning that the assignments of ELECTRE TRI should restore these examples.

- When the constraints are inconsistent, IRIS infers a combination of parameter values that least violates the constraints, by minimizing the maximum deviation. Furthermore, a module becomes available to determine the alternative subsets of constraints that must be removed to restore the consistency.

- When the constraints are consistent, IRIS infers a "central" combination of parameter values by minimizing the maximum slack. For each action, it depicts the category corresponding to that combination, as well as the range of categories where the action might be assigned without violating any constraint (robustness analysis). For each category in the range IRIS may also determine a combination of parameter values that assigns the action to that category.

- Moreover, when the constraints are consistent, IRIS may compute some indicators concerning the precision of the inputs (by estimating the volume of the polyhedron of all feasible combinations of parameter values) and the precision of the outputs (by indicating the geometric mean of the number of possible assignments per action).

2.1. What is new in IRIS 2

IRIS 2.0 has benefited from improvements suggested by users of IRIS 1.0/1.1 and from a recent theoretical development (see *(ii)* below):

(i) New integer programming routines were incorporated in the software, thus allowing it to run the inconsistencies analysis module without resorting to an external solver (IRIS 1.0 depended on the presence of the LINGO solver from Lindo Systems, Inc., which is now no longer necessary). However, computations may be slower in this new version.

(ii) IRIS 2 can work with veto thresholds, thus allowing to model discordance in the construction of outranking relations. It uses the variant *S''* outranking relation proposed by Mousseau and Dias (2002), allowing the user to work with single (v_j) or double (v_j and u_j) discordance thresholds per criterion.

(iii) Action names and criteria names may be edited and saved. These names may be arbitrarily chosen (any string including alphanumerical characters 0-9 and a-z, spaces, and symbols such as ".", "!", "+". Etc.), whereas in IRIS 1 actions and criteria were identified by numbers. For instance, "Cost (10^6 €)" is a valid name for a criterion.

(iv) The input files (".tri" extension) have a new format, in order to contain information about the discordance-related thresholds and the action and criteria names. However, IRIS 2 is able to read files created by IRIS 1. Conversely, IRIS 1 is able to read files created by IRIS 2, but information about discordance-related thresholds and the action and criteria names will be lost.

(v) A progress bar appears when results are being computed, which can be noticed in problems of large size.

3. Methodology

3.1. The sorting problematic

Roy (1985) defines four "problematics" (types of problems) in multicriteria decision aid:

- **description problematic**: the purpose of the analysis is to describe the decision situation in a formal language, in terms of actions, criteria and evaluations;
- **choice problematic**: the purpose of the analysis is to select one action (or a limited number of actions);
- **ranking problematic**: the purpose of the analysis is to rank the actions by order of preference;
- **sorting problematic**: the purpose of the analysis is to sort the actions into categories defined *a priori*.

The sorting problematic evaluates each action according to its intrinsic absolute merit. Each action is assigned to a category independently from the remaining actions. If the categories are ordered according to the Decision Maker's preferences (e.g. the categories "high risk", "medium risk", "low risk", "very low risk" in the evaluation of applications for credit) the problematic may be called **ordinal sorting**. Otherwise, the problematic may be called **nominal sorting** (e.g. separating job applicants according to the categories "creative profile", "technical profile", "human relations profile", "leadership profile").

3.2. ELECTRE TRI

The ELECTRE family of methods has been created in the 1960's by Bernard Roy and his collaborators (e.g. see Roy, 1991; Roy and Bouyssou, 1993). It consists of several methods developed for the choice and ranking problematics, and a method to deal with the ordinal sorting problematic: the ELECTRE TRI (Yu, 1992; Roy and Bouyssou, 1993).

Let us introduce some **notation**:

- m — number of actions;
- n — number of criteria;
- t — number of categories;
- $A=\{a_1, ..., a_m\}$ — set of actions;
- $G=\{g_1(.), ..., g_n(.)\}$ — set of criteria (real valued functions on A);
- $C=\{C_1, ..., C_t\}$ — set of ordered categories (C_1 is the worst one, C_t is the best one);
- $B=\{b_0, ..., b_t\}$ — set of profiles (reference actions) that separate consecutive categories.

Each category C_i is limited by two reference actions (profiles): b_i is its upper limit and b_{i-1} is its lower limit:
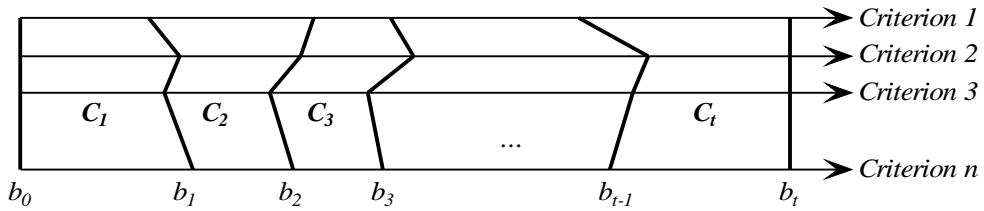
The assignment of actions to categories is based on the concept of **outranking relation** on AxB. An action $a_i \hat{I} A$ outranks a profile $b_h \hat{I} B$ (denoted $a_i \, S \, b_h$) if it can be considered at least as good as the latter (i.e. a_i is not worse than b_h), given the evaluations (performances) of a_i and b_h at the n criteria. If a_i not worse than b_h in every criterion, then it is obvious that $a_i \, S \, b_h$. However, if there are some criteria where a_i is worse than b_h, then a_i may outrank b_h or not, depending on the relative importance of those criteria and the differences in the evaluations (small differences might be ignored, whereas vary large differences may oppose a veto to the outranking). In the next subsections we present the variant of ELECTRE TRI implemented by IRIS (for other variants see Yu, 1992; Roy and Bouyssou, 1993).

3.2.1. DEFINITION OF THE OUTRANKING RELATION

We present here the definition of the outranking relation on AxB, as proposed by Mousseau and Dias (2002). Let us introduce some more notation:

- k_j is the importance coefficient (weight) of criterion $g_j(.)$, which is always a positive number;
- $q_j(b_h)$ is the indifference threshold associated with criterion $g_j(.)$ and profile b_h;
- $p_j(b_h)$ is the preference threshold associated with criterion $g_j(.)$ and profile b_h;
- $u_j(b_h)$ is the discordance threshold associated with criterion $g_j(.)$ and profile b_h;
- $v_j(b_h)$ is the veto threshold associated with criterion $g_j(.)$ and profile b_h;
- D_j is the advantage of a_i over b_h on criterion $g_j(.)$:

$$D_j = \begin{cases} g_j(a_i) - g_j(b_h) & \text{, if } g_j(.) \text{ is to be maximized (the more the better)} \\ g_j(b_h) - g_j(a_i) & \text{, if } g_j(.) \text{ is to be minimized} \end{cases}$$

- $c_j(a_i, b_h)$ is the concordance index for the assertion "$a_i \, S \, b_h$", considering criterion $g_j(.)$;
- $c(a_i, b_h)$ is the concordance index for the assertion "$a_i \, S \, b_h$", considering all the criteria;
- $d_j(a_i, b_h)$ is the discordance index for the assertion "$a_i \, S \, b_h$", considering criterion $g_j(.)$;
- $s(a_i, b_h)$ is the credibility index for the assertion "$a_i \, S \, b_h$", considering all the criteria;
- λ is the cutting level.

For each criterion ($j=1,...,n$), a concordance index indicates how much that criterion agrees with the hypothesis "$a_i \, S \, b_h$", which is computed as follows:

$$c_j(a_i,b_h)=\begin{cases} 0 & , \text{if } D_j < -p_j(b_h) \\ \left(p_j(b_h)+D_j\right)\big/\left(p_j(b_h)-q_j(b_h)\right), & \text{if } -p_j(b_h) \le D_j < -q_j(b_h) \\ 1 & , \text{if } D_j \ge -q_j(b_h) \end{cases}.$$

The concordance is maximum (1) when a_i is better than b_h or is worse but by a small difference (up to $q_j(b_h)$). When a_i is worse than b_h, the concordance starts to decrease when the difference in favor of b_h becomes larger than $q_j(b_h)$, and attains its minimum (0) when the difference in favor of b_h becomes equal to or greater than $p_j(b_h)$:
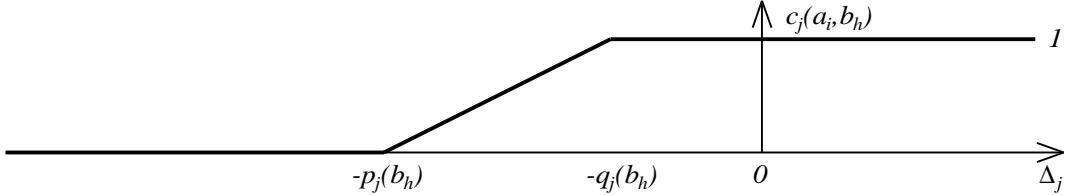
The n single-criterion concordance indices (one for each criterion) are then aggregated into a global (multicriteria) concordance index, considering the relative weight k_j of each criterion:

$$c(a_i,b_h)=\sum_{j=1}^{n}k_j\,c_j\left(a_i,b_h\right), \text{ where we assume that } \sum_{j=1}^{n}k_j =1.$$

On the other hand, for each criterion ($j=1,...,n$), a discordance index indicates how much that criterion disagrees with the hypothesis "$a_i\,S\,b_h$", which is computed as follows:

$$d_j(a_i,b_h)=\begin{cases} 1 & , \text{if } D_j < -v_j(b_h) \\ \left(-D_j - u_j(b_h)\right)\big/\left(v_j(b_h)-u_j(b_h)\right) & , \text{if } -v_j(b_h) \le D_j < -u_j(b_h) \\ 0 & , \text{if } D_j \ge -u_j(b_h) \end{cases}.$$

The discordance is minimum (0) when a_i is better than b_h or is worse but by a difference up to $u_j(b_h)$ (where $u_j(b_h) \ge p_j(b_h)$). When a_i is worse than b_h, the discordance starts to increase when the difference in favor of b_h becomes larger than $u_j(b_h)$, and attains its minimum (1) when the difference in favor of b_h becomes equal to or greater than $v_j(b_h)$:

The use of the parameters $u_j(b_h)$ is optional. If these parameters are not used, then IRIS considers the value by default $u_j(b_h) = 0.25\,p_j(b_h) + 0.75\,v_j(b_h)$, as advocated by Mousseau and Dias (2002).

The n single-criterion discordance indices (one for each criterion) are then aggregated into a global (multicriteria) discordance index, considering the maximum of these values:

$$d^{max}(a_i,b_h)=\max_{j\in\{1,...,n\}} d_j(a_i,b_h).$$

Finally, the credibility of the statement "$a_i S b_h$" is given by:

$$s(a_i,b_h) = c(a_i,b_h) [1 - d^{max}(a_i,b_h)].$$

The cutting level λ is a threshold that indicates whether the credibility is significant or not:

$$a_i \text{ outranks } b_h \ (a_i S b_h) \quad \Leftrightarrow \quad s(a_i,b_h) \geq \lambda.$$

3.2.2. ASSIGNMENT RULE

The pessimistic variant of ELECTRE TRI implemented in IRIS assigns each action a_i to the highest category C_h such that a_i outranks b_{h-1}. To use such a rule the following conditions have to be taken into account when defining the set of profiles *B*:

- $g_j(b_h)$ is better than $g_j(b_{h-1})$, $\forall j \in \{1,...,n\}$ (b_h dominates b_{h-1}), for $h = 1, ...,t$;
- $a_i S b_0$ (a_i outranks the worst profile b_0), $\forall a_i \in A$;
- $\sim(a_i S b_t)$ (a_i does not outrank the best profile b_t), $\forall a_i \in A$;
- if $a_i \in A$ is indifferent to a profile $b_h \in B$ (i.e. $a_i S b_h \wedge b_h S a_i$), then a_i will not be indifferent to any other profile.

Now, the assignment rule can be implemented as follows to place a_i in a category from *C*:

- if a_i does not outrank b_1 (i.e. $s(a_i,b_1) < \lambda$), then a_i belongs to category C_1; otherwise,
- if a_i does not outrank b_2 (but has outranked b_1), then a_i belongs to category C_2; otherwise,
- if a_i does not outrank b_3, then a_i belongs to category C_3; etc.

Formally, the rule may be written as:

$$a_i \text{ belongs to category } C_h \quad \Leftrightarrow \quad a_i S b_{h-1} \wedge \sim(a_i S b_h) \quad \Leftrightarrow \quad s(a_i,b_{h-1}) \geq \lambda \wedge s(a_i,b_h) < \lambda.$$

3.3. Inference of parameter values

IRIS does not require the user to indicate precise values for the criteria weights $k_1, ..., k_n$ and the cutting level λ. Rather, it allows to obtain such values through an inference procedure (Mousseau and Slowinski, 1998) that tries to restore assignment examples.

The user may indicate the following constraints on the parameter values:

- LB_j and UB_j denote, respectively, a lower and an upper bound for k_j;
- l_{min} and l_{max} denote, respectively, a lower and an upper bound for λ;
- $C_{worst}(a_i)$ denotes the worst envisaged category for a_i, and $C_{best}(a_i)$ its best envisaged category;
- $a_{0z}.1 + a_{1z} k_1 +...+ a_{nz} k_n \geq \beta_z$ ($z=1,...,n_{cons}$) denote a set of n_{cons} additional constraints.

These constraints define the following system of inequalities:

(1) $k_j \geq LB_j$ ($j=1,...n$)   (note: this lower bound should be greater than *0*)

(2) $-k_j \geq -UB_j$ ($j=1,...n$)   (note: this upper bound should be lower than *0.5*)

(3) $l \geq l_{min}$ (note: this lower bound should not be lower than *0.5*)

(4) $-l \geq -l_{max}$ (note: this upper bound should be lower than *1*)

(5) $c_1\left(a_i, b_{C_{worst}(a_i)-1}\right).k_1 + c_2\left(a_i, b_{C_{worst}(a_i)-1}\right).k_2 + ... + c_n\left(a_i, b_{C_{worst}(a_i)-1}\right).k_n \geq 1 / [1 - d^{max}(a_i, b_h)]$

$$(a_i \hat{I} A)$$

(6) $-c_1\left(a_i, b_{C_{best}(a_i)}\right).k_1 - c_2\left(a_i, b_{C_{best}(a_i)}\right).k_2 - ... - c_n\left(a_i, b_{C_{best}(a_i)}\right).k_n \geq -1 / [1 - d^{max}(a_i, b_h)]$ [1]

$$(a_i \hat{I} A)$$

(7) $a_{0z}.1 + a_{1z} k_1 + ... + a_{nz} k_n \geq \beta_z$ $(z=1,...,n_{cons})$,

to which we add

(8) $k_1 + k_2 + ... + k_n = 1$.

Let us write (1)-(7) in a more compact matrix notation as $Z \times (1,k_1,...,k_n)^T \geq 0$, where Z is an appropriate matrix with as many rows as the number of inequalities in (1)-(7) and $n+1$ columns.

Now, the following linear program may be used to infer the parameter values, if exist, that satisfy all the constraints (which implies restoring all the assignment examples) with greatest slack:

$min \{\alpha \in \Re: \alpha + Z \times (1,k_1,...,k_n)^T \geq 0, k_1+...+k_n=1\}$ (the variables are $\alpha, 1, k_1, ..., k_n$)

If the minimum α (its optimal value) is zero or less, then the system $Z \times (1,k_1,...,k_n)^T \geq 0$ is consistent and the optimal value for the variables $1, k_1, ..., k_n$ satisfies all the constraints. Otherwise, if the minimum α is positive, then there does not exist any combination of parameter values able to satisfy all the constraints in (1)-(8) simultaneously (see Section 3.6 on dealing with an inconsistent system of constraints).

3.4. Robust assignment ranges

Let us consider again the system $Zx(1,k_1,...,k_n)^T \geq 0, k_1+...+k_n=1$ introduced in the previous section, which represents all the assignment examples, together with any other bounds and additional constraints that the user wishes to insert. Besides inferring a combination of values for the parameters (see previous section), it is possible to determine the best and worst possible assignment for each action, given a consistent system of constraints, using linear programming (for a more general approach see Dias and Clímaco, 2000):

- To find $W(a_i)$, the worst assignment for an action a_i compatible with the constraints:

    1. $h \leftarrow 1$

    2. While $min\{s(a_i,b_h)-1: Zx(1,k_1,...,k_n)^T \geq 0, k_1+...+k_n=1\} \geq 0$ (variables are $1, k_1, ..., k_n$)

        do $h \leftarrow h + 1$

        end while

    3. $W(a_i) \leftarrow h$

---

[1] A very small positive constant $\varepsilon=0.0001$ is added by IRIS to this right-hand-side value to ensure that the inequality becomes strict. We did not add it here to simplify the presentation.

- To find $B(a_i)$, the best assignment for an action a_i compatible with the constraints:

    4. $h \neg t\text{-}1$

    5. While $max\{s(a_i,b_h)\text{-}1: Z\mathbf{x}(1,k_1,...,k_n)^T ℨ 0, k_1+...+k_n=1\} < 0$ (variables are $1, k_1, ..., k_n$)

    do $h \neg h\text{ - }1$

    end while

    3. $W(a_i) \neg h+1$

Except a few rare cases (see Section 3.4.1), an action a_i may be assigned to the range of categories that goes from $W(a_i)$ to $B(a_i)$ without violating any constraint. The following robust conclusions (valid for all the acceptable combinations of parameter values) may be drawn for every $a_i \hat{I} A$:

- a_i is not worse than $W(a_i)$;

- a_i is not better than $B(a_i)$.

Sometimes, $W(a_i) = B(a_i)$, which means that a precise robust assignment has been found for that action despite the lack of precise values for the parameter values.

3.4.1. IMPOSSIBLE ASSIGNMENTS WITHIN A RANGE

It may occur that some actions $a_i \hat{I} A$ cannot be assigned to categories which lay between $W(a_i)$ and $B(a_i)$. As an illustration, let us consider the assignment of an action a_i according to four criteria:
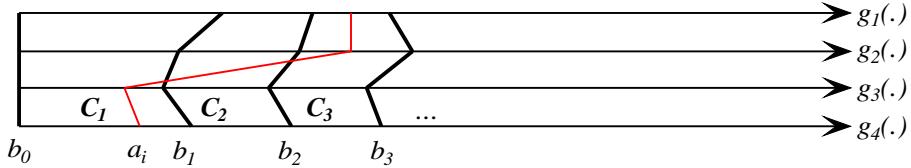
Action a_i has performances that are between b_2 and b_3 according to criteria $g_1(.)$ and $g_2(.)$. According to these criteria, it would belong to C_3. On the other hand, according to criteria $g_3(.)$ and $g_4(.)$ it has performances that are between b_0 and b_1, hence should belong to C_1. If $k_1 + k_2 ℨ 1$, then the first two criteria are important enough to make a_i fall into C_3; otherwise, a_1 falls into C_1: there is no intermediate possibility, i.e. whatever the values for k_1, k_2, k_3, k_4, and 1, it is not possible for a_i to be assigned to C_2.

In the version of ELECTRE TRI presented in sections 3.1 - 3.2, such impossible assignments may appear only when an action compares equivalently with two consecutive profiles, when discordance does not intervene:

$$c_j(a_i,b_{h\text{-}1}) = c_j(a_i,b_h), \text{ "}j\hat{I}\{1,...,n\} \quad \hat{U} \quad a_i \text{ cannot be assigned to } C_h, \text{ "}1,k_1,...,k_n.$$

3.5. Interaction process to build an ELECTRE TRI model

Dias et al. (2002) describe an interactive process to progressively build an ELECTRE TRI model (i.e. to define the values for the criteria weights and the cutting level), combining parameter inference (Section 3.3) and robustness analysis (Section 3.4).

At a given iteration, the input may consist of a system of constraints (1)-(8) on the criteria weights and the cutting level (recall Section 3.3.), besides fixed values for the actions performances, the profiles, and the indifference, preference, discordance and veto thresholds. The general idea is to start with few constraints of the parameter values, adding more inequalities as a product of an interactive learning process about the problem and the method. For instance, the user may start with loose bounds for the criteria weights (e.g. $0.1 \leq k_j \leq 0.49$) and the cutting level (e.g. $0.51 \leq \lambda \leq 0.99$) and no further constraints or assignment examples.

In any iteration, the system of constraints corresponding to the input information may be consistent or not. The analyses that may be performed will depend on the presence or absence of a consistent system.

If the system is consistent:

In this case, there will be a set of combinations of values for the variables λ, k_1, ..., k_n that satisfy the system (1)-(8), i.e. restore all the assignment examples and simultaneously conform to the additional constraints. The interaction should aim at reducing the set of accepted combinations of parameter values, either by modifying a constraint or adding a new one. To guide the user in this task, several results may be computed:

- a "central" combination of parameter values $(\lambda, k_1, ..., k_n)^*$ that is inferred from the current information (Section 3.3);
- for each action, the category were it belongs according to those inferred values;
- for each action, the range of categories where it might be assigned without violating any constraint (Section 3.4), which also allows to see which actions are more affected by the imprecision;
- for each action, a sample combination of parameter values compatible with each category in its range (e.g. if an action a_1 could be assigned to any category between C_2 and C_5 the user could analyze four combinations of parameter values, each one leading to a different category); this analysis is particularly useful for the worst and best categories in the range, since it may suggest new constraints on the corresponding "extreme" parameter values;
- the relative size (volume) of the set of parameters that satisfy all the constraints.
- the geometric mean of the number of categories where each action may be assigned.

If the system is inconsistent:

In this case, there will not exist any combination of values for the variables $1, k_1, ..., k_n$ that satisfies the system (1)-(8). The interaction should aim at restoring the system's consistency, by removing (or at least relaxing) one or more constraints. To guide the user in this task, several results may be computed:

- a "central" combination of parameter values $(1, k_1,..., k_n)^*$ that is inferred from the current information to minimize the maximum constraint violation (Section 3.3);

- the category where each action is assigned to, according to the inferred values $(1, k_1,..., k_n)^*$, highlighting the assignment examples that were not restored;

- for each constraint, an indication of whether it is violated and by how much;

- a list of sets of constraints that, if removed, yield a consistent system (see Section 3.6).

The proposed procedure is designed to be used interactively, i.e., the output at a given iteration is used to guide the revision of the input for the following iteration. The procedure can start with very little information. Each iteration will provide opportunity to add, delete or modify a specific supplementary constraint. Adding only a single piece of information at each allows to better understand its effect on the results. This process should aim at progressively reducing the set of accepted combinations of parameter values, until the end users (decision makers, problem owners) are satisfied with the results' precision, and yet comfortable with and confident about the constraints introduced.

The final outputs of the procedure are:

- a set of constraints and assignment examples defining a set of acceptable combinations of parameter values;

- an inferred combination of parameter values defining a model in a precise manner;

- a precise assignment or range of assignments for each action in *A* that is robust with respect to the constraints inserted.

However, the most important outcome may be that the end users will increase the insight on their view of the problem, learn about their preferences, and will possibly modify their opinions.

3.6. Dealing with inconsistencies

It may occur that, after introducing some constraints, the system of inequalities

(1) $k_j \ge LB_j$ $(j=1,...n)$
(2) $-k_j \ge -UB_j$ $(j=1,...n)$
(3) $\lambda \ge \lambda_{min}$
(4) $-\lambda \ge -\lambda_{max}$
(5) $c_1\!\left(a_i,b_{C_{worst}(a_i)-1}\right)\!.k_1 + c_2\!\left(a_i,b_{C_{worst}(a_i)-1}\right)\!.k_2 +...+c_n\!\left(a_i,b_{C_{worst}(a_i)-1}\right)\!.k_n \ge \lambda \;/[1 - d^{max}(a_i,b_h)]$

$(a_i \hat{I} A)$

(6) $-c_1\!\left(a_i,b_{C_{best}(a_i)}\right)\!.k_1 - c_2\!\left(a_i,b_{C_{best}(a_i)}\right)\!.k_2 -...-c_n\!\left(a_i,b_{C_{best}(a_i)}\right)\!.k_n \ge -\lambda \;/[1 - d^{max}(a_i,b_h)]$

$(a_i \hat{I} A)$

(7) $a_{0z}.\lambda + a_{1z}\, k_1 +...+ a_{nz}\, k_n \ge \beta_z$ $(z=1,...,n_{cons})$,
(8) $k_1 + k_2 + ... + k_n = 1$,

becomes inconsistent, i.e. there does not exist any combination of values for the parameters $\lambda,k_1,...,k_n$ able to satisfy all the constraints simultaneously. Besides the information referred to in the previous section, there are methods to compute alternative ways of restoring the consistency by removing some constraints (Mousseau et al., 2003). One of these methods uses mixed integer programming (continuous and 0-1 variables) to compute a succession of sets of constraints S_1, S_2, ..., S_p such that:

1. $\forall i \hat{I} \{1,...,p\}$, if the constraints in S_i are removed from system (1)-(8) then it becomes consistent;

2. $\forall i \hat{I} \{1,...,p\}$, $i \neq j$, $S_i \not\subset S_j$;

3. $\forall i \hat{I} \{1,...,p\}$, $i<j$, $|S_i| \le |S_j|$;

4. If removing a set of constraints S from the system (1)-(8) makes it become consistent, then either $S \not\subset S_i$ $(i \hat{I} \{1,...,p\})$ or $|S| \ge |S_p|$.

Each one of the sets S_1, S_2, ..., S_p presents an alternative manner to restore the consistency. The end user should choose one of these sets, which are presented by increasing order of cardinality, and remove (or at least relax) the constraints in that set. The interaction may then continue as explained in the previous section.

The method to compute these sets is the following one:

- Set $p \gets 1$

- Solve the 0-1 programming problem

$$min\; \{\textstyle\sum_{i=1}^{z} y_i :$$
$$Z \times (\lambda,k_1,...,k_n)^T + M.(y_1,...,y_z)^T \ge 0$$
$$k_1+k_2+ ...+k_n=1$$
$$\lambda,k_1,...,k_n \ge 0,\; y_1,...,y_z \in \{0,1\}$$
$$\}$$

In this mathematical program, *M* denotes a very large number and *z* denotes the number of constraints in the system (1)-(7). Each of these constraints is associated with a variable that can take only the values 0 or 1. Since these variables are multiplied by an arbitrarily large number, setting any of these to the value 1 amounts to ignore the corresponding original constraint. The objective is to minimize the sum of these variables, so that all would be zero if the system (1)-(8) was consistent.

- Since the system (1)-(8) is not consistent, the optimal solution to the above problem will contain several $y_i=1$ ($i \in \{1,...,z\}$). Let $S_1=\{ i \in \{1,...,z\}: y_i=1\}$. Then, removing the constraints indexed by S_1 from (1)-(8) would result in a consistent system.

- Set $p \neg 2$

- Solve the 0-1 programming problem

$$min \ \{ \sum_{i=1}^{z} y_i :$$
$$Z \times (1,k_1,...,k_n)^T + M.(y_1,...,y_z,0)^T \ni 0$$
$$k_1+k_2+ ...+k_n=1$$
$$\sum_{i \in S_1} y_i \leq \# S_1 - 1$$
$$1,k_1,...,k_n \geq 0, \ y_1,...,y_z \in \{0,1\}$$
$$\}$$

This mathematical program is equal to the former, except the introduction of a new constraint $\sum_{i \in S_1} y_i \leq \# S_1 - 1$. This constraint prohibits the former optimal solution (or a superset of that solution).

- A set S_2 is formed from the new optimal solution as explained for the case of S_1.

- Set $p \neg 3$, add the constraint $\sum_{i \in S_2} y_i \leq \# S_2 - 1$, etc.

- The process stops as soon as a pre-defined number of sets is reached or when the 0-1 programming problem becomes infeasible, which means that there are no more alternative ways to restore the consistency of (1)-(8).

4. Software presentation

4.1. General structure

IRIS is a SDI (Single Document Interface) program, like for instance Microsoft Explorer. This means that the user can work at a single document (problem) at a time. Of course, the user can work on several problems at the same time by running several instances of IRIS simultaneously.

The program runs on a single window and the user can change its position, change its size, minimize it, etc. This window is divided in two areas. The left area of the screen is for inputs, whereas the right area of the screen is for outputs. Each area is organized according to a multi-page notebook metaphor with tabs to change pages. The space occupied by each area may be changed by clicking on the dividing line and dragging it to the left or to the right.

When inputs change, outputs become invalid, which is shown by using a red font in the output pages. The outputs will reflect the changes in input only after the option **Robust Assignments** from the **Results** menu is chosen (or, as a shortcut, press **Alt+R**, then **R**).
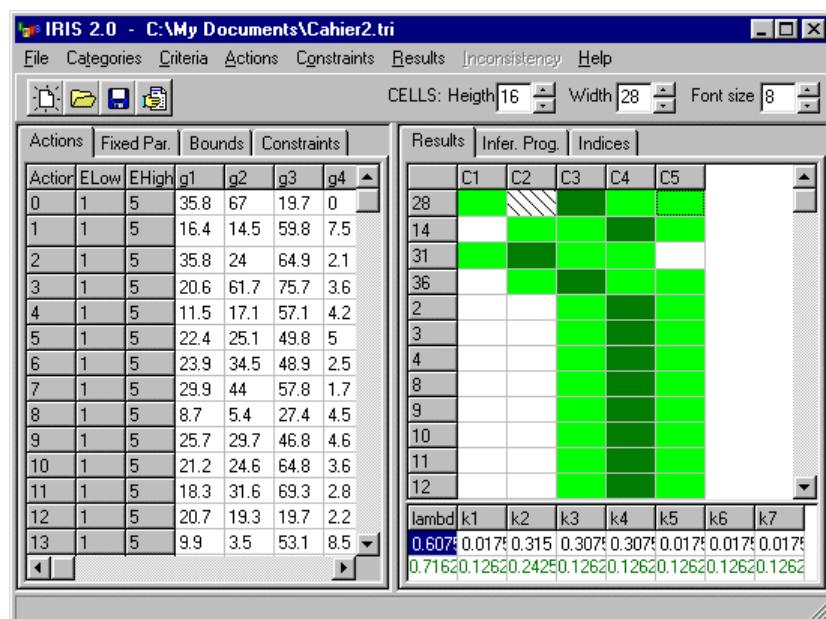
There are some grids associated with input and output pages. The user can edit the height, width and font size of the grid elements by setting their values at the right top of the screen:

It may happen that the contents of some grid cells cannot be displayed in its entirety. In such cases, if the user does not wish to enlarge the width of the cells, he/she may position the mouse over the cell so that its contents will be entirely displayed. All of the input or output pages have pull-down

menus associated with them. To access these menus, the user just has to click on the right button of the mouse or select the pull-down menu key that is available in some keyboards. The last option in each menu (**Help**) leads directly to the page of the on-line manual related to the current page.

4.2. Input

The inputs must be read from a file (and then may be changed) or typed in by the user. To open an existing file choose **File|Open** (or button) and locate the file. The default extension is ".tri". To create a new file choose **File|New** (or button) and insert the number of actions (alternatives), the number of criteria, and the number of categories for your problem. The program allows to add or delete criteria, actions or categories later. The caption of the window indicates the name of the current inputs file. In the present version of the program there is a limit of 20 criteria. The number of actions is limited only by the amount of memory. To save the current file choose **File|Save Data As** (or button), which allows the user to define the location and name of the file, or choose **File|Save Data** to save it under its current name and location.

An alternative to creating and editing the inputs file using IRIS, which is the most natural option, is to create or edit that file using a text processor, given the syntax presented in Appendix A. Appendix B shows how to import data from a spreadsheet like Excel.

The inputs area, which may be enlarged or reduced, contains four pages:

- **Actions**: To edit the performances of the actions on the multiple criteria and (optionally) to set some assignment examples.
- **Fixed Par.**: To edit the performances that define category bounds (profiles) and to edit the criteria thresholds associated with these bounds.
- **Bounds**: To edit the upper and lower bounds of the importance coefficients (weights) and the cutting level (lambda).
- **Constraints**: To edit the explicit constraints (other than bounds) on the parameter values. Note that the implicit constraints related to assignment examples are edited in the Actions page.

4.2.1. ACTIONS PAGE

When working in the Actions page, the user may edit the multicriteria performances of the actions to be sorted, and may insert assignment examples.

The performances of the actions may be directly input in the corresponding cells. The user may navigate between cells using the mouse or the keyboard arrow keys. All input must be numerical, either positive or negative values, either integer or not (the decimal point is ".", regardless of Window's settings). The performance cells cannot be blank: zero values must be explicitly inserted as a number.

The names of the actions may be edited by clicking on the corresponding cell ("Action" column". Any string of characters may be used to name an action, including alphanumerical characters 0-9 and a-z, spaces, and symbols such as ".", "!", "+". For instance, "Project 4*" is a valid action name. Criteria names may also be edited by clicking on the corresponding cell (title row of the grid) using the same rules. For instance, "Cost (10^6 €)" is a valid name for a criterion. Please note that IRIS does not prohibit setting different alternatives (or criteria) with equal names.

The user may change the number of criteria, either creating new ones or deleting some of them. The **Criteria** menu and the pop-up menu offer the commands to perform this. The user may also change the number of actions, either creating new ones or deleting some of them. The **Actions** menu and the pop-up menu offer the commands to perform this.

Each action has a lower and a higher category where it may be assigned (columns *ELow* and *EHigh*, respectively) that allow the user to indicate assignment examples. Typically, the column *ELow* contains the lowest category (which is always 1) and the column *EHigh* contains the highest category (which is 5 in the above figure). If the user changes these values then the action's assignment will be constrained: it becomes an assignment example. For instance, in the figure above one sees three assignment examples (which the program highlights): action 1 is assigned to category 5 (the highest one); action 10 is assigned to the interval of categories 3 to 4; and action 12 is assigned to category 4.

To change the values in the columns *ELow* and *EHigh*, the user must click (using the mouse) over the cell that he/she wants to change. The value in *ELow* cannot exceed the value in *EHigh*. Hence, if both values are equal to 2 and the user wishes to change both values to 3, he/she

must change *EHigh* first. The **Actions** menu contains a command **Erase Examples** to remove all the assignment examples (by setting *ELow* equal to 1 and *EHigh* equal to the number of categories).

4.2.2. **FIXED PARAMETERS** PAGE

When working in the Fixed Par. page, the user may edit the performances of the profiles (reference actions, often fictitious, that separate two consecutive categories), as well as the thresholds associated with the criteria (which may vary from profile to profile). When there are *t* categories, there will exist *t-1* profiles.

Actions	Fixed Par.	Bounds	Constraints
	g1	g2	g3	g4	g5	g6	g7
g(b1)	-10	-60	90	28	40	1	0
q1	1	4	1	1	0	0	0
p1	2	6	3	2	3	0	0
v1		600	300	1000	300		
g(b2)	0	-40	75	23	32	2	2
q2	1	4	1	1	0	0	0
p2	2	6	3	2	3	0	0
v2		600	300	1000	300		
g(b3)	8	-20	60	18	22	4	3
q3	1	4	1	1	0	0	0
p3	2	6	3	2	3	0	0
v3		600	300	1000	300		
g(b4)	25	30	35	10	14	5	4
q4	1	4	1	1	0	0	0
p4	2	6	3	2	3	0	0
v4		600	300	1000	300		
MAX/	1	1	-1	-1	-1	1	1

Use vj   Use uj

The performances of the profiles may be directly input in the corresponding cells. A row staring with *g(bi)* refers to the *i*-th profile. Profile *g(b1)* separates the two worst categories, denoted categories 1 and 2; profile *g(b2)* separates categories 2 and 3; and so forth. The rows starting with *qi* refer to the indifference thresholds associated with the *i*-th profile, and the rows starting with *pi* refer to the preference thresholds for the *i*-th profile. The preference threshold for a given criterion cannot be less than the corresponding indifference threshold.

By default IRIS is prepared to accept veto thresholds (see figure above). To disable veto for a particular criterion the user may place the value of zero (which is hidden) for the veto threshold of that criterion. The user may also disable/enable veto for all the criteria by selecting/unselecting the button "Use vj" (figure below, on the left). When "Use vj" is not selected, IRIS considers $s(a_i,b_h) = c(a_i,b_h)$.

The user may choose to work with discordance thresholds also, instead of letting them take the default value $u_j(b_h) = 0.25\, p_j(b_h) + 0.75\, v_j(b_h)$. To enable/disable the use of explicit discordance thresholds the user may select/unselect the button "Use uj" (figure below, on the right). This button is available only when the button "Use vj" is selected. As for the veto thresholds, discordance thresholds with value 0 are ignored and hidden.

Actions | Fixed Par. | Bounds | Constraints

	g1	g2	g3	g4	g5	g6	g7
g(b1)	-10	-60	90	28	40	1	0
q1	1	4	1	1	0	0	0
p1	2	6	3	2	3	0	0
g(b2)	0	-40	75	23	32	2	2
q2	1	4	1	1	0	0	0
p2	2	6	3	2	3	0	0
g(b3)	8	-20	60	18	22	4	3
q3	1	4	1	1	0	0	0
p3	2	6	3	2	3	0	0
g(b4)	25	30	35	10	14	5	4
q4	1	4	1	1	0	0	0
p4	2	6	3	2	3	0	0
MAX/	1	1	-1	-1	-1	1	1

Use vj   Use uj

Actions | **Fixed Par.** | Bounds | Constraints

	g1	g2	g3	g4	g5	g6	g7
g(b1)	-10	-60	90	28	40	1	0
q1	1	4	1	1	0	0	0
p1	2	6	3	2	3	0	0
u1							
v1		600	300	1000	300		
g(b2)	0	-40	75	23	32	2	2
q2	1	4	1	1	0	0	0
p2	2	6	3	2	3	0	0
u2							
v2		600	300	1000	300		
g(b3)	8	-20	60	18	22	4	3
q3	1	4	1	1	0	0	0
p3	2	6	3	2	3	0	0
u3							
v3		600	300	1000	300		
g(b4)	25	30	35	10	14	5	4
q4	1	4	1	1	0	0	0
p4	2	6	3	2	3	0	0
u4							
v4		600	300	1000	300		
MAX/	1	1	-1	-1	-1	1	1

Use vj   Use uj

The last row in the grid indicates if the preference increases or decreases with the performances. If the value is *1*, then the higher the performance, the better (the criteria is one to be maximized, such as "customer satisfaction"). If the value is *-1*, then the lower the performance, the better (the criteria is one to be minimized, such as "fuel consumption").

The user may navigate between cells using the mouse or the keyboard arrow keys. All input must be numerical, either positive or negative values, either integer or not (the decimal point is "."). The performance cells cannot be blank: zero values must be explicitly inserted as a number.

The user may change the number of categories, either creating new ones (by splitting existing categories) or deleting some of them (by merging consecutive categories). The **Categories** menu and the pop-up menu offer the commands to perform this.

The user may also change the number of criteria, either creating new ones or deleting some of them. The **Criteria** menu and the pop-up menu offer the commands to perform this.

4.2.3. **BOUNDS** PAGE

When working in the Bounds page, the user may edit the upper and lower bounds of the cutting level (*lambda*) and the weights (*ki* refers to the weight of the *i*-th criterion).

File   Categories   Criteria   Actions   Constraints   Results   Inconsistency   Help

CELLS: Heigh 16   Width 28   Font size 8

Actions | Fixed Par. | **Bounds** | Constraints

	lambd	k1	k2	k3	k4	k5	k6	k7
LB - L	0.6	0.01	0.01	0.01	0.01	0.01	0.01	0.01
UB - L	0.99	0.49	0.49	0.49	0.49	0.49	0.49	0.49

Results | Inferred constrain

	C1	C2	C3	C4
0				
1				

lambd	k1	k2	k3	k4

Bounds may be directly input in the corresponding cells. The user may navigate between cells using the mouse or the keyboard arrow keys. All input must be numerical, in the interval [0,1] (the decimal point is "."). The upper bounds should not, of course, be lower than the corresponding lower bounds. Zero values must be explicitly inserted as a number (a blank cell originates an error).

4.2.4. CONSTRAINTS PAGE

When working in the Constraints page, the user may edit the constraints (other than bounds and assignment examples) that the weights and cutting level should respect.

The constraints may be directly input in the corresponding cells. The first "normalization" equality (yellow color) is fixed. The user may navigate between the remaining cells using the mouse or the keyboard arrow keys. Zero-valued coefficients may be left blank (indeed, to improve readability, IRIS hides all the zero values except those in the *RHS* column). The right hand sides cannot be negative, but the remaining coefficients can. To enter the type of inequality type "<=" (or just "<", which is considered the same), "=", or ">=" (or just ">").

The user may change the number of constraints, either creating new ones or deleting some of them. The constraints menu and the corresponding pop-up menu offer the commands to perform this. The option of deleting asks the user which constraint is to be deleted, and the constraints identification labels change accordingly after the deletion.

In the above example, the constraints *ad1* to *ad6* force k_2 to have a value which is not less than any of the other weights:

- $-k_1 + 1k_2 \geq 0 \Leftrightarrow k_2 \geq k_1$ (*ad1*)
- $1k_2 - 1k_3 \geq 0 \Leftrightarrow k_2 \geq k_3$ (*ad2*)
- $1k_2 - 1k_4 \geq 0 \Leftrightarrow k_2 \geq k_4$ (*ad3*)
- $1k_2 - 1k_5 \geq 0 \Leftrightarrow k_2 \geq k_5$ (*ad4*)
- $1k_2 - 1k_6 \geq 0 \Leftrightarrow k_2 \geq k_6$ (*ad5*)
- $1k_2 - 1k_7 \geq 0 \Leftrightarrow k_2 \geq k_7$ (*ad7*)

4.3. Output

After having specified all of the inputs, the user may get some results. The **Results** menu offers the following options:

- The **Volume Computation** option estimates the volume of the polyhedron of combinations of parameter values that respect all the constraints (including bounds and assignment examples) using Monte-Carlo simulation. A window appears where user may change the number of significant digits (**Precision**) and press the button **Start simulation**. Then, IRIS determines the dimension of the polyhedron and estimates its total volume (**Absolute volume**), as well as the ratio between the total volume and the volume of the polyhedron defined by the bounds on the parameters (i.e. excluding the other explicit constraints and the assignment example constraints) **(Volume to bounds**). The window must be closed to continue using IRIS (button **Exit**).
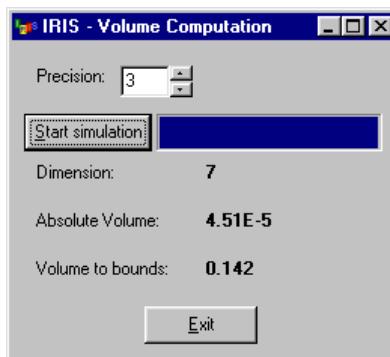
- The **Robust Assignments** option computes the inferred parameter values and assignments, as well as the range of possible assignments for each action, if the constraints are consistent. After choosing this option, the outputs area of the screen will reflect the inputs.
- The **by Input Order** option instructs IRIS to sort the actions by their input number.
- The **by Variability Order** option instructs IRIS to sort the actions by decreasing variability order, where variability refers to the difference between the best and worst possible assignments.

The outputs area of the screen, updated after choosing the option **Robust Assignments**, may be enlarged or reduced and contains the following pages:

4.3.1. RESULTS PAGE

This page displays a grid with the inferred parameter values and assignments, as well as the range of possible assignments for each action. Depending on the selection previously made in that menu (**by Input Order** or **by Variability Order**), the actions appear in the same order as the Actions page (e.g. first figure below) or appear by decreasing order of variability (e.g. second figure below), where variability here means the difference between the best and worst categories in the actions assignment range. The user may change the actions' order at any time, even after the results are computed.
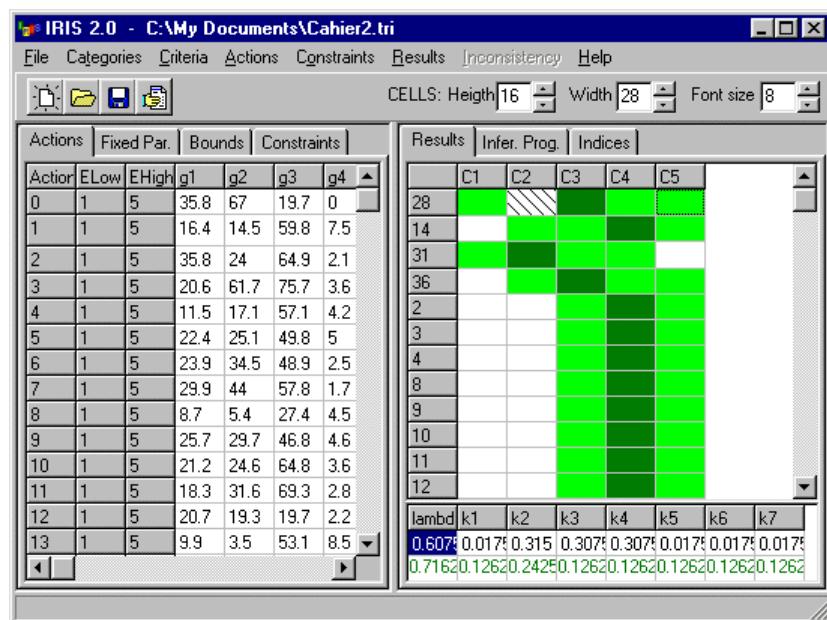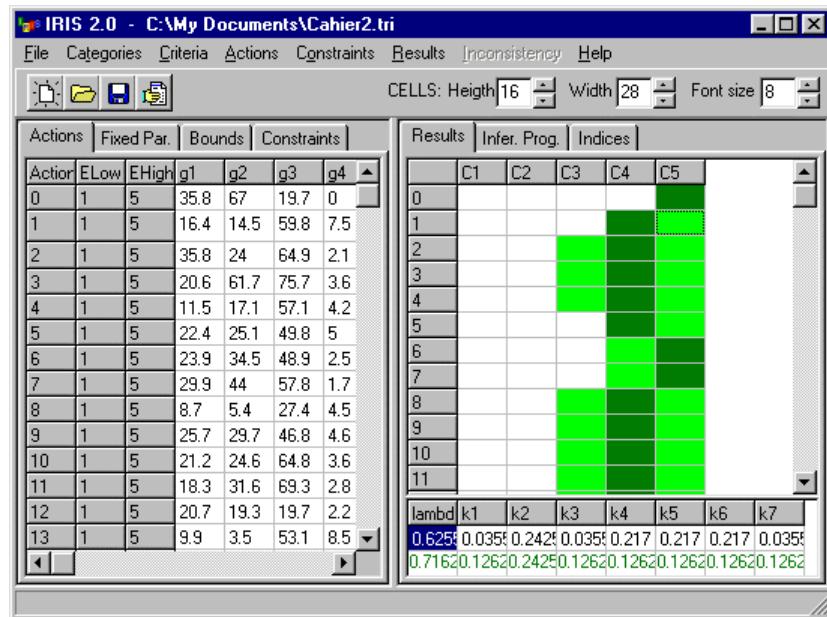
The **Results** page uses color to indicate the range of possible assignments for each action (robustness analysis), i.e. the categories where it may be assigned without violating the constraints, bounds and assignment examples. These ranges appear in green color. In some situations, there are some intermediate categories where an action cannot be assigned (recall Section 3.4.1), as for instance *a28* in the figure above: when *a28* is good enough to be better than *C1*, then it reaches *C3* without being assigned to *C2*. These situations are presented to the user as a "hole" in a range.

In each range, one of the cells has a darker shade of green, meaning it is the assignment recommended by IRIS, based on the inferred combination of parameter values. This combination is chosen to be relatively central to the set of combinations that respect all the bounds, constraints and examples. It is presented in the last row of the **Results** page, in green color. If the user selects any cell in a range, then the penultimate line in the **Results** page shows a combination of parameter

values that assigns the action in the cell's row to the category in the cell's column. For instance, the figure above shows a combination of values that assigns *a28* to *C5*.

The actions that are assignment examples can easily be identified by the blue color of their label, as in the figure below, which also shows a situation where the outputs are outdated because of a change in the inputs. This is shown by the use of a red font.
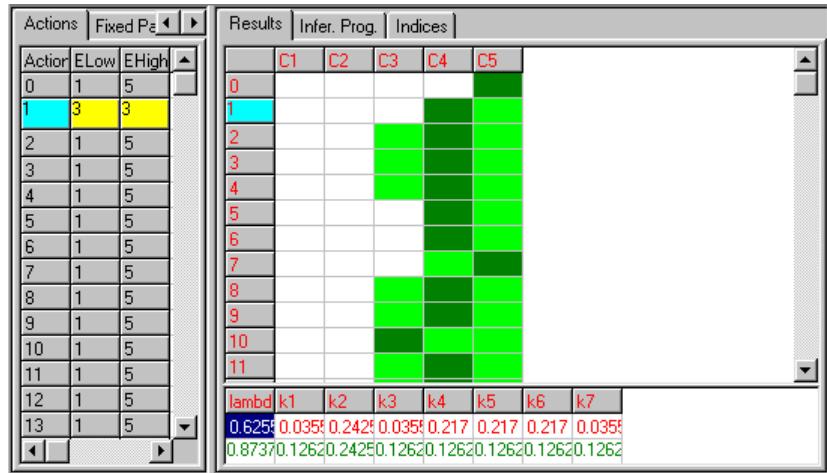
If there is no combination of parameter values that respects simultaneously all the bounds, constraints and assignment examples, then there will be no ranges to depict (inputs are inconsistent). In these cases, IRIS shows a proposal for assigning all the actions such that the maximum deviation is minimized (this can be seen in detail in the **Inference Program** page). The assignment examples that are not restored appear in red color, as in the figure below. The inconsistency analysis module may help the user in these cases (see section 4.5. below).
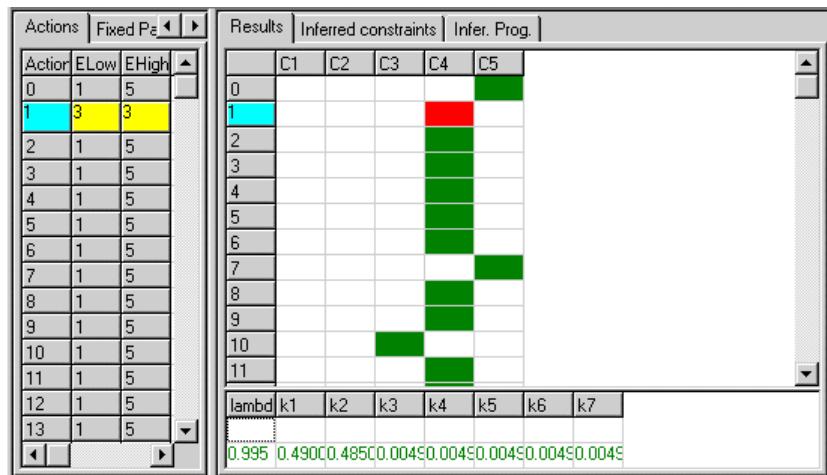
4.3.2. **INFERRED CONSTRAINTS** PAGE

This page only presents information: it displays the linear constraints corresponding to the assignment examples.

4.3.3. INFER.PROG. PAGE

This page only presents information: it displays the linear program corresponding to the inference problem, indicating which constraints are violated when they are inconsistent. The rightmost column indicates the deviation between the inferred solution and each of the constraints. If a value in that column is strictly greater than zero, then the constraint is being violated, and this is highlighted using the red color, as shown below. The inferred solution (shown in the **Results page**) is the one that minimizes the greatest of these values.

Descr	alpha	lambda	k1	k2	k3	k4	k5	k6	k7	<,=,>	RHS	Error
C(1)>=3	1	-1	1	1	1	1	1	1	1	>=	0	-0.004!
C(1)<=3	1	1	-1	-1	-1	-1	-1	-1	-1	>=	1E-5	0.0050
ad1	1		-1	1						>=	0	0.0050
ad2	1			1	-1					>=	0	-0.480(
ad3	1			1		-1				>=	0	-0.480(
ad4	1			1			-1			>=	0	-0.480(
ad5	1			1				-1		>=	0	-0.480(
ad6	1			1					-1	>=	0	-0.480(
LB lambda	1	1								>=	0.6	-0.395
UB lambda	-1	1								<=	0.99	0.0050
LB k1	1		1							>=	0.01	-0.480(
UB k1	-1		1							<=	0.49	1.5E-5
LB k2	1			1						>=	0.01	-0.475(
UB k2	-1			1						<=	0.49	-0.004!
LB k3	1				1					>=	0.01	0.0050

4.3.4. INDICES PAGE

This page only presents information: it indicates the geometric mean of the number of possible assignments per action (when the constraints are consistent), and its variation relative to the previous computation.

4.4. Results report

The user may produce a report (text file) on the outputs that have been computed by selecting **File|Report** (or button). It is not necessary to supply an extension, since the program will automatically append the extension .rpt. This file indicates (see also Appendix C):

- the inferred assignment, as well as its best and worst category for each action (if the input is consistent);
- the inference mathematical program; and
- the solution to the inference program, which corresponds to the inferred assignment.

This text file may then be formatted as in a text processor or may even be read by a spreadsheet program.

If the user chooses **File|Print form** he/she will obtain a bitmap file that mirrors the present contents of the IRIS window.

4.5. Inconsistency analysis

When the constraints are inconsistent, the menu option **Inconsistency** becomes available, leading to a new window that helps the user fix the inconsistency. The inconsistency analysis form is divided in two parts. On the left side, it shows the list of the constraints forming the inconsistent system, with a number identifying each of them. On the same side, the user may choose the maximum number of suggestions for fixing the inconsistency (by removing/relaxing some of the constraints) and initiate the computation using the **Suggest** button. On the right side, the results appear as a list of different manners to resolve the inconsistency, by removing an increasing number of constraints. First, there will appear, if exist, proposals to remove one constrain only; then, proposals involving the removal of two constraints, and so on, until there are no more alternative manners to resolve the inconsistency or the maximum number of suggestions is reached. For instance, in the picture below the fifth proposal ("7+8+12") refers to the removal of three constraints identified by the numbers 7, 8 and 12 (although a relaxation, rather than deletion of the constraints, is often sufficient to restore the consistency).
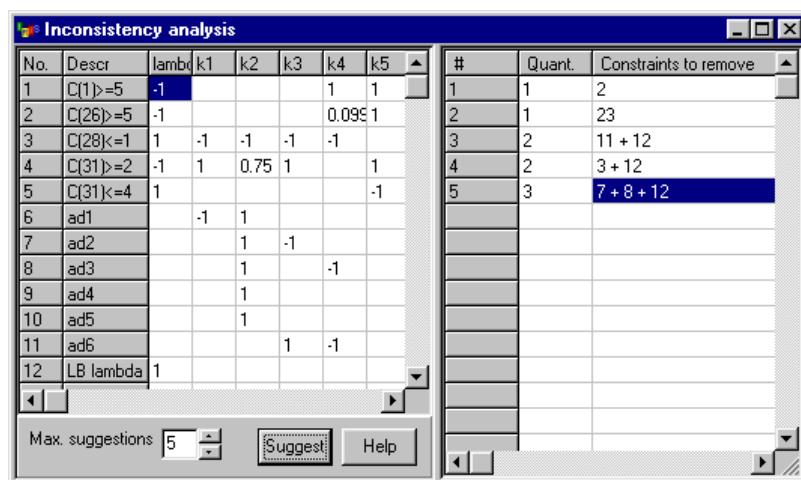
5. A step by step example

5.1. Opening a project

To open a file created previously, the user run IRIS choosing one of the two versions available: either **iris1.exe** or **iris1si.exe**. The latter version does not include the inconsistency analysis module, but does not require to have LINGO previously installed. From the menu **File**, the user must then choose the option **Open** and locate the file, which usually will have a **.tri** extension. Alternatively to using the menu, the user may click on the button .
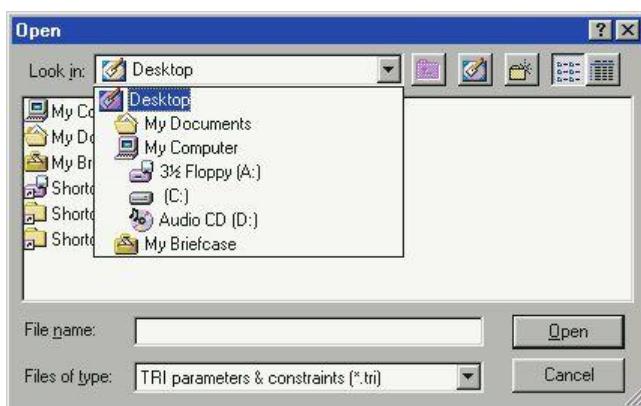
To follow this example, the user should locate the file **test1.tri** which comes with IRIS. The path and name of the file will appear in the caption of the IRIS window.
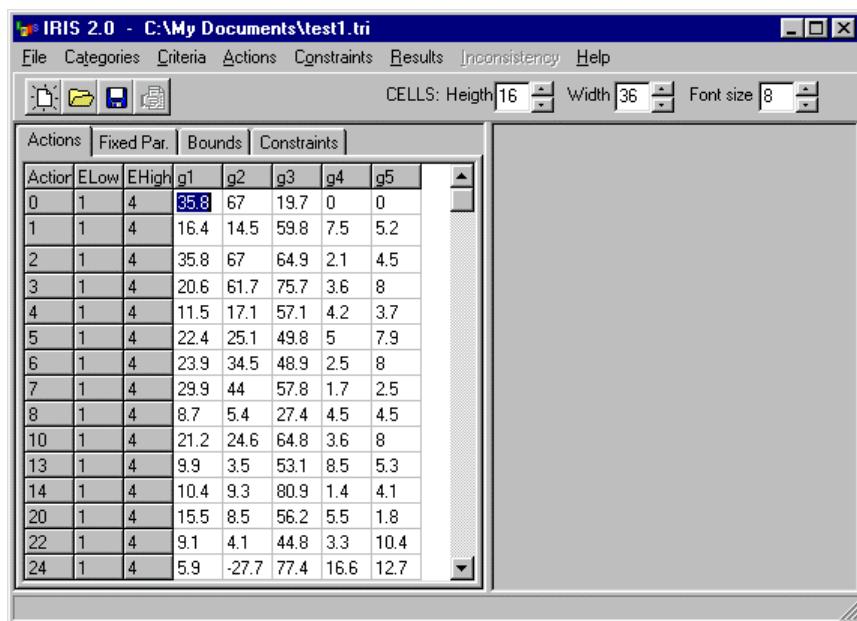
5.2. Obtaining results

After opening a project using **File|Open** or ⌂ (or after creating a new one using **File|New**), the user may start editing the file and obtaining results. Choosing **Results|Volume Computation** allows to compute the relative volume of parameter polyhedron that respects the constraints imposed so far (i.e. the constraints which state that k_2 is not lower than any other k_j (j ¹2), according to the data in the **Constraints** page, plus the bounds imposed in the **Bounds** page). The user may choose a precision (which is three digits by default), press the button **Start simulation**, and wait for the simulation to end:
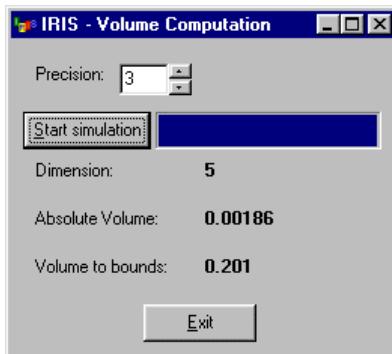
In this example, the 6-dimension polyhedron of acceptable values for the weights and cutting level has a dimension of *0.00186*.[2] Considering only the combinations that respect the bounds defined in the **Bounds** page, about *20%* of them respect the constraints in the **Constraints** page. The user may now press the **Exit** button to return to the main window.

Choosing the option **Results|Robust Assignments** IRIS will determine the range of categories where each category may be assigned to, given the polyhedron of acceptable values for the parameters, indicating the inferred "central" parameter values as well as the precise assignments corresponding to these. In this example, the user may notice that action a_2 cannot be assigned to category C_3. Recall Section 4.3 to know how to interpret these results and interactively calculate some other ones. For instance action a_3 was assigned to C_3, but might also have been assigned to C_2 or C_4 without violating any constraint. Selecting any of these cells will instruct IRIS to calculate a combination of parameter values leading to the selected assignment. The user may select the order of presentation of the actions in the **Results** menu, either choosing **by Input Order** or **by Variability Order**.

---

[2] Since this is the result of a Monte-Carlo simulation, the user may see a slightly different result.
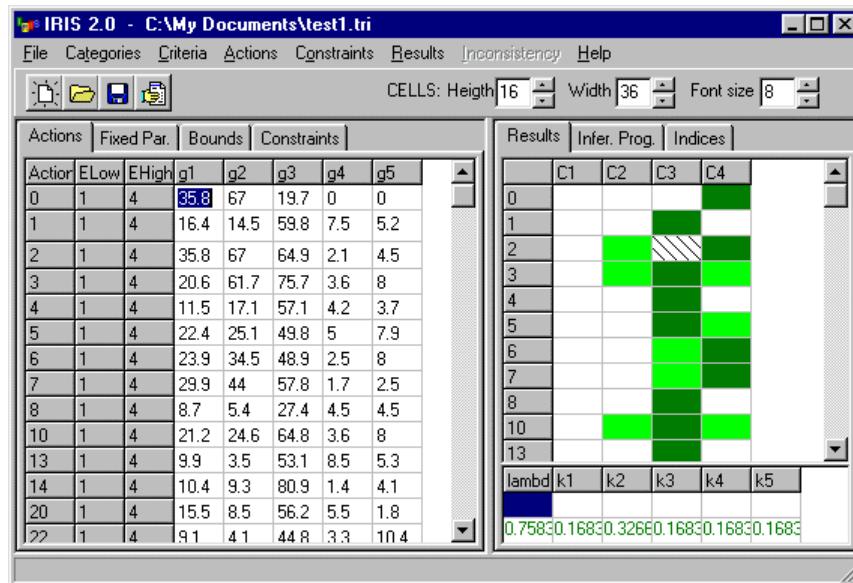
5.3. Editing the inputs

After opening a project or creating a new one, the inputs may be edited either before or after obtaining results. For convenience, the user may reduce the width of the cells using the control Width 20 . The user may change the performance values in the **Actions** page, which also allows to insert assignment examples (Section 4.2.1). For instance, let us suppose that the user wished that a_2 was assigned to C_4. To insert such an assignment example, it would be necessary to click on the cell in the column **ELow** of row "**2**", and place the value 4 as the lowest category (i.e. the category of a_2 cannot be lower than C_4). The user may also click does not need to click on the cell in the column **EHigh** in the same row, since C_4 is already the best category.
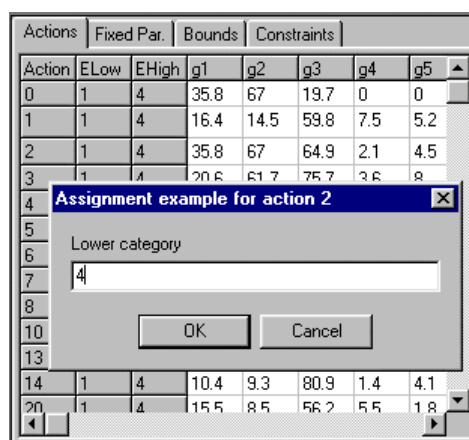
Using the mouse, the user may select the remaining pages and change the values corresponding to the fixed parameters (Section 4.2.2 — **Fixed Par.** page), the bounds on the criteria weights and the cutting level (Section 4.2.3 — **Bounds** page), and the constraints on those parameters (Section 4.2.4 — **Constraints** page). The outputs page becomes invalid after inputs are

changed, which is shown by the use of a red font. Results must be re-computed to reflect the changes in the inputs.

> An alternative to editing the inputs using IRIS is to edit the inputs file using a text processor or a spreadsheet, saving the file in text format. The syntax of the inputs file (usually with a **.tri** extension is described and exemplified in Appendix A)

5.4. Saving the data

To save the data, the user may choose between the options **File|Save Data** and **File|Save Data As**. The button ![save] corresponds to the latter option, which asks for the file's name and location and allows to save it under a different name, e.g. **test2** (IRIS automatically appends the extension **.tri**). The caption of the IRIS window will reflect the change.

5.5. Obtaining new results

The red font in the outputs page shows that the inputs have changed. To reflect these changes in the results the user must re-calculate the results by choosing **Results|Robust Assignments**. The assignment example that a_3 belongs to C_4 causes a reduction of the set of acceptable values for the parameters and hence leads to decrease in the ranges of possible assignments. This is visible in the **Results** page. An indicator for the reduction (the geometric mean of the number of categories in a range) is presented in the Indices page.
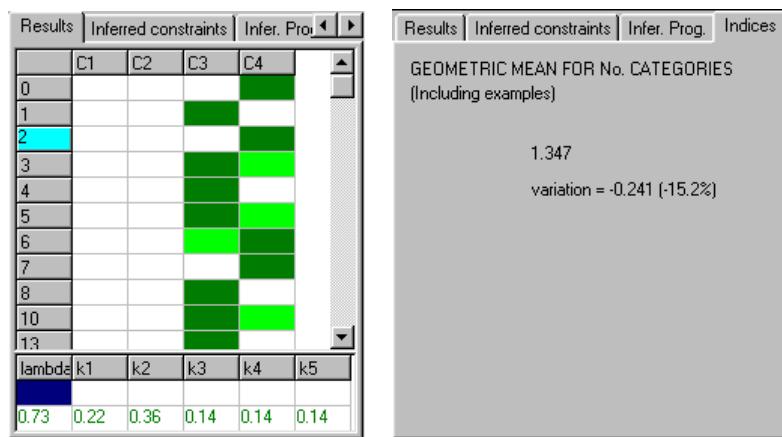
5.5. Analyzing inconsistencies

The results now show that action a_5 can be assigned either to C_3 or C_4 (the inferred suggestion). Let us suppose, however, that according to the user's experience, that action should be assigned to C_2 at the best. Obviously, that is inconsistent with the constraints imposed so far, as IRIS will state when

the user introduces this example (via the **Actions** page, as exemplified above) and re-calculates the results (**Results|Robust Assignments**). The assignments corresponding to the new inferred parameter values are presented in red color in the cases where the examples have not been restored (a_2 and a_5):
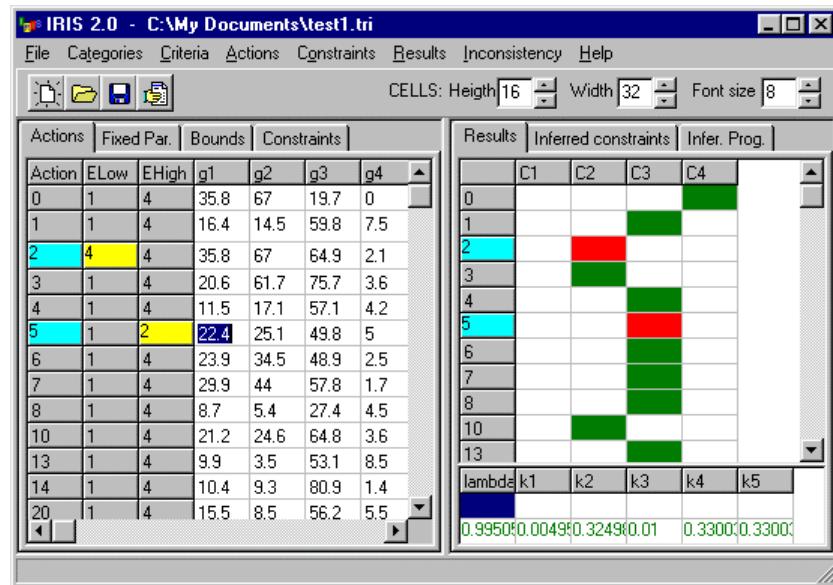
The **Inconsistency** option becomes available on the main menu, leading to a window where the user learns some suggestions on how to overcome the inconsistency (for details refer to Section 4.5). In that window, which may be moved and resized independently from the main window, the user just has to indicate the maximum number of suggestions and press the button **Suggest**. In this case there exist two possible ways to restore the consistency: either to remove constraint 2 (i.e. that a_5 belongs to category C_3 or lower), or to remove constraints 1 (i.e. that a_2 belongs to category C_4) and 8 (i.e. that the cutting level cannot exceed *0.99*). Closing this window returns the user to the main IRIS window.
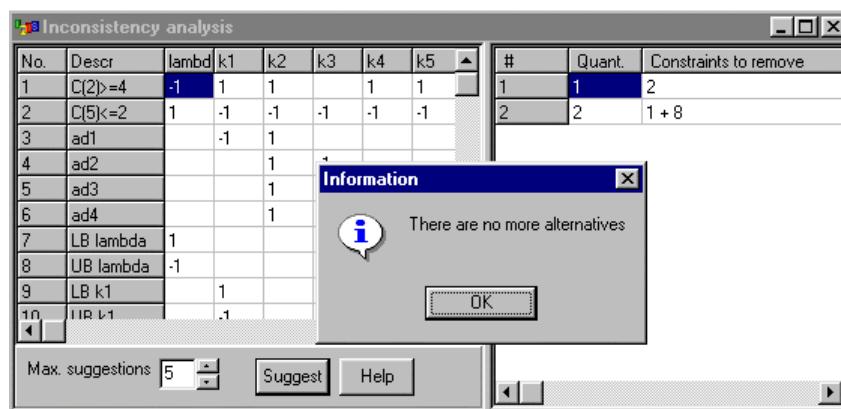
5.6. Producing a report

The user may choose **File|Report** after having computed any results. If issued at this point in the example, the report coincide with "example 2" in Appendix C. The user may select the name and location of the report file.

5.7. Further interactions

At this point (recall section 5.5), since constraint 8 cannot be relaxed, the user would remove constraint 1. To perform this, in the **Actions** page set the **EHigh** column cell corresponding to a_5 to the value of 4. Recomputing the results we are back to the previous situation (presented below with the results sorted by variability order:

The fact that action a_{28} is the one with highest variability of possible assignments invites the user to see if these possibilities may be reduced. Supposing that the user considered this action as a good example for category C_1, the following results would be reached:

At this point, the user might be so satisfied with the low variability of the results, that the process could stop. The user may choose **File|Report** after having computed any results. If issued At this

point in the example, a report (choose **File|Report**) would coincide with "example 1" in Appendix C.

5.8. Creating a new project

To create a new project, the user may choose the command **File|New**. A window appears where the user indicates the dimensions of the project:
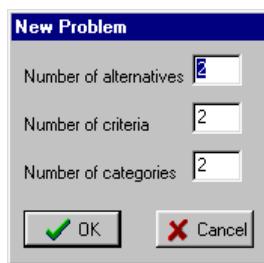
After setting these dimensions, the user may start editing the inputs. The number of criteria, a ctions (alternatives) and categories may be changed later using the **Criteria**, **Actions**, and **Categories** menus, respectively.

An alternative to creating the inputs using IRIS is to create the inputs file using a text processor or a spreadsheet, saving the file in text format. The syntax of the inputs file (usually with a **.tri** extension is described and exemplified in Appendix A).

Credits

- Wei Yu (1992) published the ELECTRE TRI method in his PhD thesis, under the supervision of Bernard Roy (LAMSADE - Univ. Paris-Dauphine, France).
- Vincent Mousseau and Roman Slowinski (1998) were the originators of the mathematical program to infer parameter values from assignment examples.
- Luis Dias and João Clímaco (2000) proposed the computation of the worst and best categories for the actions given a set of constraints on the parameter values, to derive robust conclusions.
- Luis Dias, Vincent Mousseau, José Figueira, João Clímaco, and Carlos Gomes da Silva (Dias et al., 2002; Mousseau et al., 2003) have developed the methodology that IRIS is based on, combining parameter inference with robustness analysis and inconsistency analysis (see Section 3). That research has been partially supported by Portuguese-French cooperation project 328J4 (ICCTI-French Embassy at Portugal).
- Luis Dias and Vincent Mousseau have defined the functionality and interface of IRIS.
- Luis Dias, Carlos Gomes da Silva and Rui Lourenço have performed the software engineering and programming work.

References

Dias, L.C., Clímaco, J.N. (2000), ELECTRE TRI for Groups with Imprecise Information on Parameter Values, Group Decision and Negotiation 9, 355-377.

Dias, L., V. Mousseau, J. Figueira, J. Clímaco (2002), An Aggregation/Disaggregation Approach to Obtain Robust Conclusions with ELECTRE TRI, European Journal of Operational Research 138, 332-348.

Mousseau V., L. Dias (2002), "Valued outranking relations in Electre providing manageable disaggregation procedures", Cahier du LAMSADE, No. 189, Jan 2002. (Forthcoming in European Journal of Operational Research).

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

Mousseau V., Slowinski R. (1998): "Inferring an ELECTRE TRI Model from Assignment Examples", Journal of Global Optimization 12, 157-174.

Roy, B. (1985), Méthodologie multicritère d'aide à la décision, Economica, Paris.

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

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.

Appendix A: Syntax of the input file (*.tri)

Although the easiest way of creating or updating input files is through IRIS, the syntax of such files allows to create or edit them using a text processor or even a spreadsheet capable of saving files in simple text (ASCII) format. The first character in each line determines the data it contains or determines that the line contains a comment, if the character is 'c'. This does not mean, however, that IRIS accepts data in arbitrary order. Rather, IRIS expects to read data in a sequence that cannot be changed:

- IRIS recognizes the input file as being from version 2 by a first line like the following one, otherwise it will treat it as being a version 1 file:

  ```
  c version 2
  ```

- The first non-comment line should start with a 't' to indicate the size of the problem. If n is the number of criteria, t is the number of categories, and m is the number of actions (alternatives), then the line should have the following format, with spaces or "tabs" separating the numbers:

  ```
  t      n      t      m
  ```

- The next non-comment line should start with a 'd' to indicate the direction of the criteria, followed by n_{crit} numbers separated by spaces or "tabs". Each of these numbers may take the value 1 when the corresponding criterion is to maximize (preference increases with the performance), or -1 when it is to minimize (preference decreases with the performance, e.g. a cost):

  ```
  d      1/-1      1/-1      ...      1/-1
  ```

- Then, IRIS expects t-1 lines starting with a 'p' to present the performances associated with the profiles that separate the categories. The first profile will be the one separating the lowest (worst) category from the second lowest. The last profile will be the one separating the second best category from the best one. Each line will present a profile that dominates the profile presented in the preceding line. If we denote by $g_j(b_r)$ the performance of the rth profile according to the jth criterion, then the successive lines should appear as follows (the id_number is ignored but must be present):

  ```
  p      id_number      g₁(b₁)  g₂(b₁)  ...      gₙ(b₁)
  p      id_number      g₁(b₂)  g₂(b₂)  ...      gₙ(b₂)
  ...
  p      id_number      g₁(b_{t-1})  g₂(b_{t-1})  ...  gₙ(b_{t-1})
  ```

- Afterwards, IRIS expects $n.(t$-$1)$ lines starting with an 's' to present the thresholds associated with the criteria/profiles. In each line, the first number after 's' identifies the profile (an integer between 1 and t-1), the second number identifies the criterion (an integer between 1 and n), the

third number indicates the indifference threshold, the fourth indicates the preference threshold, the fifth indicates the discordance threshold (0 if disabled), and the last indicates the veto threshold (0 if disabled). If we denote by $q_j(b_r)$, $p_j(b_r)$, $u_j(b_r)$, and $v_j(b_r)$, respectively, the indifference, preference, discordance, and veto thresholds associated to the jth criterion given the performance of the rth profile, then the successive lines should appear as follows ($r=1,..., t-1$; $j=1,..., n$):

$$\text{s} \quad r \quad j \quad q_j(b_r) \quad p_j(b_r) \quad u_j(b_r) \quad v_j(b_r)$$

- Next, IRIS expects m lines starting with an 'a' to present the performances of the actions and (possibly) assignment examples. In each line, a first number after 'a' (an integer) identifies the position of the action, followed by n numbers indicating the performances of the action at the multiple criteria. Then, there should appear two integer numbers (between *1* and *t*), the first one constraining the action's assignment from below (i.e. indicating its worst envisaged category), and the second one constraining the action's assignment from above (i.e. indicating its best envisaged category). If the first of these two numbers is higher than *1* or if the second of these numbers is lower than *t*, then these actions will belong to the set of assignment examples. Finally, all the characters until the end of the line will become the action's name If we denote by $g_j(a_i)$ the performance action a_i at the jth criterion, by $C_{worst}(a_i)$ its worst envisaged category, and by $C_{best}(a_i)$ its best envisaged category, then there should appear m lines as follows:

$$\text{a} \quad i \quad g_1(a_i) \quad g_2(a_i) \quad ... \quad g_n(a_i) \quad C_{worst}(a_i) \quad C_{best}(a_i) \quad name$$

- The next block IRIS looks for presents the bounds for the criteria weights (one line per bound). Each of these *2.n* lines will present a lower bound if it starts by 'K S', or an upper bound if it starts by 'K I'. Then, one number identifies the criterion (an integer between *1* and *n*) and the following one indicates the bound's value. If we denote by LB_j and UB_j, respectively, the lower and upper bound for k_j (the weight of the jth criterion), then the successive lines should appear as follows, for $j=1,..., n$ (the order is arbitrary):

$$\text{K S} \quad j \quad UB_j$$
$$\text{K I} \quad j \quad LB_j$$

- Then, IRIS expects one line starting with 'K N' followed by a number n_{cons} indicating the number of additional constraints on the criteria weights (not counting with the constraint that their sum is equal to one). This line is mandatory even if there are no additional constraints (then n_{cons} should be set to zero):

$$\text{K N} \quad n_{cons}$$

- If n_{cons} is greater than zero, IRIS expects n_{cons} lines starting with 'K g' or 'K e'.

A constraint $a_0.1 + a_1 k_1 + ... + a_n k_n \geq \beta$ should be coded as:

```
K g    a₀      a₁      ...      aₙ      β.
```

A constraint $a_0.1 + a_1 k_1 + ... + a_n k_n \leq \beta$ should be coded as:

```
K g    -a₀     -a₁     ...      -aₙ     -β.
```

A constraint $a_0.1 + a_1 k_1 + ... + a_n k_n = \beta$ should be coded as:

```
K e    a₀      a₁      ...      aₙ      β.
```

- Then, IRIS expects two lines, one starting with 'L m' followed by a lower bound for the cutting level lambda, and the other starting with 'L M' followed by an upper bound for the same parameter:

```
L m    lₘᵢₙ
L M    lₘₐₓ
```

- Finally, IRIS expects one line starting with 'n' followed by n lines containing the names of the n criteria.

Example:

```
c version 2
c this is the file for test1.tri
c Size of the problem:
t 5    4       20
c Directions of preference:
d     1        1      -1       -1       -1
c Profiles:
p 1    -10    -60      90      28       40
p 2    8      -20      60      18       22
p 3    25      30      35      10       14
c Thresholds:
s 1    1       1        2       0        0
s 1    2       4        6       9        12
s 1    3       1        3       0        0
s 1    4       1        2       0        0
s 1    5       0        3       0        0
s 2    1       1        2       0        0
s 2    2       4        6       9        12
s 2    3       1        3       0        0
s 2    4       1        2       0        0
s 2    5       0        3       0        0
s 3    1       1        2       0        0
s 3    2       4        6       9        12
s 3    3       1        3       0        0
s 3    4       1        2       0        0
s 3    5       0        3       0        0
c Actions:
a 0    35.8   67       19.7   0        0        1       4       0
a 1    16.4   14.5     59.8   7.5      5.2      1       4       1
a 2    35.8   67       64.9   2.1      4.5      1       4       2
a 3    20.6   61.7     75.7   3.6      8        1       4       3
a 4    11.5   17.1     57.1   4.2      3.7      1       4       4
a 5    22.4   25.1     49.8   5        7.9      1       4       5
a 6    23.9   34.5     48.9   2.5      8        1       4       6
a 7    29.9   44       57.8   1.7      2.5      1       4       7
a 8    8.7    5.4      27.4   4.5      4.5      1       4       8
a 9    21.2   24.6     64.8   3.6      8        1       4       10
a 10   9.9    3.5      53.1   8.5      5.3      1       4       13
a 11   10.4   9.3      80.9   1.4      4.1      1       4       14
a 12   15.5   8.5      56.2   5.5      1.8      1       4       20
a 13   9.1    4.1      44.8   3.3      10.4     1       4       22
a 14   5.9    -27.7    77.4   16.6     12.7     1       4       24
```

```
a 15   16.9   12.4   60.1   5.6    5.6    1      4      25
a 16   5.1    4.9    28.9   2.5    46     1      4      28
a 17   24.4   22.3   32.8   3.3    5      1      4      29
a 18   29.5   8.6    41.8   5.2    6.4    1      4      30
a 19   7.3    -64.5  67.5   30.1   8.7    1      4      31
c Upper bounds on weights:
K S 1 0.49
K S 2 0.49
K S 3 0.49
K S 4 0.49
K S 5 0.49
c Lower bounds on weights:
K I 1 0.01
K I 2 0.01
K I 3 0.01
K I 4 0.01
K I 5 0.01
c Additional constraints:
K N 4
K g    0      -1     1      0      0      0      0
K g    0      0      1      -1     0      0      0
K g    0      0      1      0      -1     0      0
K g    0      0      1      0      0      -1     0
c Bounds on lambda:
L M 0.99
L m 0.6
n name of criteria:
g1
g2
g3
g4
g5
c *** END OF FILE ***
```

Appendix B: Importing data from MS-Excel

Sometimes, the user may already have data in a spreadsheet like MS-Excel, namely a table that indicates the performances of the actions at the several criteria. This appendix shows how these data may be easily imported into IRIS. For instance, let us assume that the performance data of 10 actions at 5 criteria were in a file **example.xls**, as follows (if the actions were in columns instead of rows, we could use the TRANSPOSE function in Excel):

example.xls					
	A	**B**	**C**	**D**	**E**	**F**
1			Performance data			
2	Action	g1(.)	g2(.)	g3(.)	g4(.)	g5(.)
3	0	34	4	7	434	98
4	1	24	4	4	556	97
5	2	35	2	5	775	75
6	3	41	6	6	123	87
7	4	19	3	9	523	99
8	5	23	8	7	578	100
9	6	54	3	3	698	69
10	7	22	3	3	732	86
11	8	52	5	8	431	100
12	9	87	3	2	623	95

To import these data, the user may proceed as follows:

1. Start IRIS.
2. Create a new file, indicating 10 actions, 5 criteria, and the number of categories (Section 5.7).
3. Save the file, choosing its location and name, e.g. **example.tri** (Section 5.4).
4. In Excel, open the file **example.tri**, accepting the default choices proposed by the Text Import Wizard (Original Data Type = Delimited, Delimiters = Tab, Column Data Format = General):

The spreadsheet (Microsoft Excel - example.tri) contains:

	A	B	C	D	E	F	G	H	I	J
1	c version 2									
2	c Size of the problem:									
3	t 5	3	10							
4	c Directions of preference:									
5	d	1	1	1	1	1				
6	c Profiles:									
7	p 1	0	0	0	0	0				
8	p 2	0	0	0	0	0				
9	c Thresholds:									
10	s 1	1	0	0	0	0				
11	s 1	2	0	0	0	0				
12	s 1	3	0	0	0	0				
13	s 1	4	0	0	0	0				
14	s 1	5	0	0	0	0				
15	s 2	1	0	0	0	0				
16	s 2	2	0	0	0	0				
17	s 2	3	0	0	0	0				
18	s 2	4	0	0	0	0				
19	s 2	5	0	0	0	0				
20	c Actions:									
21	a 0	0	0	0	0	0	1	3	0	
22	a 1	0	0	0	0	0	1	3	1	
23	a 2	0	0	0	0	0	1	3	2	
24	a 3	0	0	0	0	0	1	3	3	
25	a 4	0	0	0	0	0	1	3	4	
26	a 5	0	0	0	0	0	1	3	5	
27	a 6	0	0	0	0	0	1	3	6	
28	a 7	0	0	0	0	0	1	3	7	
29	a 8	0	0	0	0	0	1	3	8	
30	a 9	0	0	0	0	0	1	3	9	
31	c Upper bounds on weights:									

5. Copy the data in cells B3:F12 from **example.xls** to the cells B21:F30 in the file **example.tri**.

6. Save **example.tri**, keeping the text (tab-delimited) format and close that file.

7. Re-open **example.tri** in IRIS: it now contains the copied data.

Appendix C: Syntax of the report file (*.rpt)

The first part of the results report indicates the name of the input file used to derive results. This gives the user the possibility of saving the inputs under a convenient name before producing the results report, so that the two files are congruent.

INPUT FILE
Full path of the file

The second part of the report indicates the worst category $W(a_i)$, the inferred category $I(a_i)$, and the best category $B(a_i)$ for each action a_i. These are separated by the "tab" character. When the inputs are inconsistent the worst category and the best category are blank.

RESULTS:

ACTION	Worst Cat	Inferred Cat	Best Cat
$name_i$	$W(a_i)$	$I(a_i)$	$B(a_i)$
...			

The third part of the report presents the constraints to the inference mathematical program that minimizes α. The last column (Error) appears only when the system of constraints is inconsistent, indicating by how much is the constraint violated (if Error is positive) or the existing slacks (if Error is negative). Let us denote each constraint as following the pattern

$$b_1 a + b_2 l + b_3 k_1 + \ldots + b_{n+2} k_n \{\leq, =, \geq\} b_{n+3}.$$

INFERENCE PROGRAM:

Descr	alpha	lambda	k1	...	k*n*	<,=,>	RHS	Error
...								
label	b_1	b_2	b_3	...	b_{n+2}	<=/=/>=	b_{n+3}	*error*
...								

The last part of the report presents the optimal solution $(l^*, k_1^*, \ldots, k_n^*)$ to the inference program.

INFERRED SOLUTION:

lambda	k1	...	k*n*
l^*	k_1^*		k_n^*

The format of this results file makes it convenient to be read by a text processor or a spreadsheet.

Example 1 (consistent system):

```
INPUT FILE:
C:\My Documents\test1.tri

RESULTS:
ACTION     Worst Cat      Inferred Cat       Best Cat
0          4      4       4
1          3      3       3
2          4      4       4
3          3      3       4
4          3      3       3
5          3      3       4
6          3      4       4
7          4      4       4
8          3      3       3
10         3      3       4
13         3      3       3
14         3      3       3
20         3      3       3
22         3      3       3
24         2      2       2
25         3      3       3
28         1      1       1
29         3      3       3
30         3      3       3
31         1      1       2

INFERENCE PROGRAM:
Descr       alpha lambda k1    k2    k3    k4    k5 <,=,> RHS     Error
C(2)>=4     1     -1     1     1     0     1     1  >=    0
C(28)<=1    1      1    -1    -1    -1    -1     0  >=    0.0001
ad1         1      0    -1     1     0     0     0  >=    0
ad2         1      0     0     1    -1     0     0  >=    0
ad3         1      0     0     1     0    -1     0  >=    0
ad4         1      0     0     1     0     0    -1  >=    0
LB lambda   1      1     0     0     0     0     0  >=    0.6
UB lambda  -1      1     0     0     0     0     0  <=    0.99
LB k1       1      0     1     0     0     0     0  >=    0.01
UB k1      -1      0     1     0     0     0     0  <=    0.49
LB k2       1      0     0     1     0     0     0  >=    0.01
UB k2      -1      0     0     1     0     0     0  <=    0.49
LB k3       1      0     0     0     1     0     0  >=    0.01
UB k3      -1      0     0     0     1     0     0  <=    0.49
LB k4       1      0     0     0     0     1     0  >=    0.01
UB k4      -1      0     0     0     0     1     0  <=    0.49
LB k5       1      0     0     0     0     0     1  >=    0.01
UB k5      -1      0     0     0     0     0     1  <=    0.49

INFERRED SOLUTION:
lambda    k1            k2            k3            k4         k5
0.80004   0.10498       0.39002       0.10498       0.10498    0.29504
```

Example 2 (inconsistent system):

```
INPUT FILE:
C:\My Documents\test1.tri

RESULTS:
ACTION   Worst Cat Inferred Cat    Best Cat
0                  4
1                  3
2                  2
3                  2
4                  3
5                  3
6                  3
7                  3
8                  3
10                 2
13                 3
14                 2
20                 3
22                 3
24                 2
25                 3
28                 1
29                 3
30                 3
31                 1


INFERENCE PROGRAM:
Descr       alpha     lambda k1   k2   k3   k4   k5    <,=,>   RHS     Error
C(2)>=4     1         -1     1    1    0    1    1     >=      0       0.00505
C(5)<=2     1         1      -1   -1   -1   -1   -1    >=      0.0001  0.00505
ad1         1         0      -1   1    0    0    0     >=      0       -0.32003
ad2         1         0      0    1    -1   0    0     >=      0       -0.31498
ad3         1         0      0    1    0    -1   0     >=      0       0.00505
ad4         1         0      0    1    0    0    -1    >=      0       0.00505
LB lambda 1           1      0    0    0    0    0     >=      0.6     -0.39505
UB lambda -1          1      0    0    0    0    0     <=      0.99    0.00505
LB k1       1         0      1    0    0    0    0     >=      0.01    0.00505
UB k1       -1        0      1    0    0    0    0     <=      0.49    -0.48505
LB k2       1         0      0    1    0    0    0     >=      0.01    -0.31498
UB k2       -1        0      0    1    0    0    0     <=      0.49    -0.16502
LB k3       1         0      0    0    1    0    0     >=      0.01    0
UB k3       -1        0      0    0    1    0    0     <=      0.49    -0.48
LB k4       1         0      0    0    0    1    0     >=      0.01    -0.3200
UB k4       -1        0      0    0    0    1    0     <=      0.49    -0.15997
LB k5       1         0      0    0    0    0    1     >=      0.01    -0.32003
UB k5       -1        0      0    0    0    0    1     <=      0.49    -0.15997


INFERRED SOLUTION:
lambda   k1        k2       k3       k4       k5
0.99505  0.00495   0.32498  0.01     0.33003  0.33003
```

Appendix D: Menu structure

Available menus in the bar

FILE Menu:

- New: Creates a new problem, asking for its dimensions.
- Open: Opens a problem, reading the inputs from disk.
- Report: Creates a file containing the computed results.
- Save Data: Saves the current inputs on disk, considering the current file's name and location.
- Save Data As: Saves the current inputs on disk, allowing the user to specify the file's name and location.
- Print Form: Prints a copy of the IRIS window.
- Print Setup: Allows the user to define the printer's settings.
- Exit: Terminates the program.

CATEGORIES Menu

- Split: Splits a category (prompting the user to choose which one) into two categories. The user has to specify suitable profiles for the two categories.
- Merge: Merges two consecutive categories (the user chooses the lower one) into a single one. The user does not need to edit the profiles, since the merged category inherits the lower bound of the lower category and the upper bound of the higher category.

CRITERIA Menu

- Insert: Adds a new criterion.
- Delete: Deletes a criterion (prompting the user to choose which one).

ACTIONS Menu

- Insert: Adds a new alternative (action).
- Delete: Deletes an alternative (action) (prompting the user to choose which one).
- Erase Examples: Removes all the assignment examples (the actions are not deleted, only the constraints imposed on them).

CONSTRAINTS Menu

- Insert: Adds a new constraint.

- Delete: Deletes a constraint (prompting the user to choose which one). Note that the constraint number "zero" ("norm") cannot be deleted.

RESULTS Menu

- Volume Computation: Provides an estimate of the volume of the polytope formed by the combinations of parameter values that respect all the constraints, bounds and assignment examples.

- Robust Assignments: Updates the outputs, solving the inference problem and determining the assignment ranges (robustness analysis).

- by Input Order: Sorts the actions by their input number.

- by Variability Order: Sorts the actions by decreasing variability order.

INCONSISTENCY Menu

This menu is available only when the constraints are inconsistent. It activates the inconsistency analysis form.

HELP Menu

- Online Manual: Opens the on-line manual. A default browser must be installed.

- How to Get Help: Briefly explains how to get help.

- About...: Provides information on the IRIS version.

Available pop-up menus

(These menus are accessible either using the right button of the mouse or

using a special key present in some keyboards)

- Actions page: A menu gives access to the options in the criteria and actions' menus (described just above) and an option for getting specific help.

- Fixed Par. page: A menu gives access to the options in the criteria and categories' menus (described just above) and an option for getting specific help.

- Bounds page: A menu gives access an option for getting specific help.

- Constraints page: A menu gives access to the options in the constraints menu (described just above) and an option for getting specific help.

- Results and Infer. Prog. pages: A menu gives access an option for getting specific help.