Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 147 (2003) 72–93
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Decision Aiding

Resolving inconsistencies among constraints on the parameters of an MCDA model

Vincent Mousseau [a,b,*], José Figueira [a,c,d], Luís Dias [c,d], Carlos Gomes da Silva [e], João Clímaco [c,d]

[a] *LAMSADE, Université Paris Dauphine, Place du Maréchal De Lattre de Tassigny, 75775 Paris Cedex 16, France*
[b] *DIMACS, Rutgers University, CoRE Building, 96 Frelinghuysen Road, Piscataway, NJ 08854-8018, USA*
[c] *Faculdade de Economia, Universidade de Coimbra, Av. Dias da Silva 165, 3004-512 Coimbra, Portugal*
[d] *INESC – Coimbra, R. Antero de Quental 199, 3000-033 Coimbra, Portugal*
[e] *Escola Superior de Tecnologia e Gestão, Instituto Politécnico de Leiria, Morro do Lena, Alto Vieiro, 2401-951 Leiria, Portugal*

Received 24 January 2001; accepted 12 February 2002

Abstract

We consider a framework where decision makers (DMs) interactively define a multicriteria evaluation model by providing imprecise information (i.e., a linear system of constraints to the model's parameters) and by analyzing the consequences of the information provided. DMs may introduce new constraints explicitly or implicitly (results that the model should yield). If a new constraint is incompatible with the previous ones, then the system becomes inconsistent and the DMs must choose between removing the new constraint or removing some of the older ones. We address the problem of identifying subsets of constraints which, when removed, lead to a consistent system. Identifying such subsets would indicate the reason for the inconsistent information given by DMs. There may exist several possibilities for the DMs to resolve the inconsistency. We present two algorithms to identify such possibilities, one using {0,1} mixed integer linear programming and the other one using linear programming. Both approaches are based on the knowledge that the system was consistent prior to introducing the last constraint. The output of these algorithms helps the DM to identify the conflicting pieces of information in a set of statements he/she asserted. The relevance of these algorithms for MCDA is illustrated by an application to an aggregation/disaggregation procedure for the Electre Tri method.

Keywords: Multiple criteria analysis; Inconsistent linear systems; Electre Tri; Aggregation/disaggregation approach; Imprecise information

1. Introduction

Multicriteria decision aiding models usually have many preference parameters that the decision makers (DMs) must set. These parameters influence the manner in which differences in performances are evaluated,

* Corresponding author. Tel.: +33-1-44-05-41-84; fax: +33-1-44-05-40-91.
E-mail addresses: mousseau@lamsade.dauphine.fr (V. Mousseau), figueira@fe.uc.pt, figueira@lamsade.dauphine.fr (J. Figueira), ldias@inescc.pt (L. Dias), cgsilva@estg.iplei.pt (C. Gomes da Silva), jclimaco@inescc.pt (J. Clímaco).

0377-2217/03/$ - see front matter
PII: S0377-2217(02)00233-3

the role of each criterion in the aggregation of the performances. Providing precise figures for all parameter values is often difficult, to the extent that there may exist some imprecision, contradiction, arbitrariness, and/or lack of consensus concerning the value of the parameters (see [16]).

We consider an imprecise information context (see, e.g., [4,22]), where the DMs may indicate some constraints on the acceptable combinations of parameter values. Such information may be provided in an explicit manner (e.g., parameter t_1 belongs to $[0.2, 0.3]$, or parameter t_1 is larger than parameter t_2), or in an implicit manner (indicating a result that the model should restore, e.g., alternative a_1 should be better ranked than a_2). Methods that accept the latter type of constraints to infer parameter values are often called aggregation/disaggregation procedures (see [7,10,13]).

In the course of an interactive process, DMs may progressively add constraints on the parameter values. Let T_k denote the set of parameter values that are acceptable to the DMs (according to the constraints they provided) at the kth iteration. Given this set, it is possible to provide some output to support the DMs in revising T_k:

- *robust conclusions* – the results that are valid for all the combinations $t \in T_k$ (see [17,21]); for instance, "a_1 is never contained in the choice set" in a selection problem; "a_1 is always better ranked than a_2" in a ranking problem; or "a_1 can only be assigned to category *good* or *very good*" in a sorting problem;
- *variability information* – the results that vary more, according to the combination chosen; for instance, "the position of a_1 in the ranking is very unstable: for some input values it may be the best, whereas for other combinations it is one of the worst" in a ranking problem (see [8]);
- *inferred parameter values procedures* (see [7,13]) – a "central" combination $t \in T_k$ that satisfies all the constraints, hence able to restore the results that were demanded; for instance, "w is the best weight vector in order to account for the ranking $a_2 \succ a_4 \succ a_1 \succ a_9 \succ a_7$ (stated by the DMs)".

We consider interactive processes in which DMs start the first iteration with very little information. Each iteration will provide an opportunity to add, delete or modify a specific supplementary constraint. Adding a single piece of information at each iteration facilitates the control of the information supplied by the DMs. This interactive process stops when DMs are satisfied with the set T_k and when the results of the model match their view of the decision problem.

We will consider that all the constraints are linear (T_k is a polyhedron) and that the polyhedron T_{k+1} that corresponds to the next iteration is obtained by adding a single constraint, i.e., by intersecting T_k with a half-space or a hyperplane. A difficulty occurs when T_{k+1} becomes empty, meaning that the new constraint contradicts some of the previous ones. To resolve the inconsistency that appeared in the linear system of constraints, one must either drop the new constraint, or some of the older ones. The choice should belong to the DMs, after they learn which are the sets of constraints that lead to a non-empty T_{k+1} if removed. Note that when we refer to the removal of one or more constraints, the DMs may choose to relax these constraints instead (e.g. increasing the right-hand side of an $Ax \leqslant b$ system).

Dealing with inconsistent information provided by DMs in the context of an interactive preference elicitation process requires the development of specific algorithms aiming at solving the infeasibility problem induced by the MCDA context. Many authors have previously addressed the subject of infeasibility analysis in linear programming (see [3] for a complete summary of the state of the art in infeasibility analysis algorithms) according to different perspectives, namely:

1. Some authors (see [1,11,20]) are interested in determining an irreducibly inconsistent system (IIS). An IIS is a minimal subset of constraints that corresponds to an inconsistent system, in the sense that any proper subset of an IIS is a consistent system. Let us remark that the inconsistency in an IIS can be removed by deleting any constraint, but if there are several IISs, then the initial system of constraints can remain inconsistent.

2. A different problem is to determine the minimum number of constraints that has to be removed to restore the consistency in the initial system, which is equivalent to solving the minimum-cardinality IIS set-covering problem (see [2,14]).
3. Finally, we can mention the problem of determining the minimum weight alternative to restore the consistency in a system, which is equivalent to determining a minimum-weight IIS set-cover (see [2,14]).

The perspective we are interested in is close to problem 2, with the following specificities:

- we are also interested in sets of constraints that restore the consistency if removed that are not of minimum cardinality, since the DMs may rather drop two constraints they consider unimportant than drop a single important one;
- we know that one of the constraints caused the inconsistency, hence removing that constraint is a trivial manner to resolve the inconsistency; the question here is what other alternatives exist.

Hence, we may formulate the problem we are addressing as: to determine all "minimal" subsets of constraints S_i (in terms of cardinality) verifying: (a) the cardinality of each S_i does not exceed a given value (Ω) and (b) the deletion of S_i restores the consistency to the initial system.

In the context of the interactive processes we are considering, solving such problems will allow us to propose alternative ways to resolve an inconsistency that appeared at a given iteration. This helps the DMs to understand how their inputs are conflicting and to question previously expressed judgments. Analyzing and confronting the alternative solutions of such problems provide opportunities for the DMs to learn about their preferences as the interactive process evolves. Searching for the smallest subsets of constraints is consistent with the idea according to which the DMs will first consider the "less complex" ways to solve inconsistency. [1]

In the next section we define our problem formally and propose two techniques to solve it. One of the techniques consists of solving a succession of $\{0,1\}$ mixed integer linear programs, while the second one uses only linear programming. Section 3 presents an application to the aggregation/disaggregation approach for Electre Tri, reviewing this approach and including a numerical example. We finally indicate some extensions and conclude the paper.

2. Two different methods to cope with inconsistent systems

Consider a problem in which the DMs has interactively specified constraints on the preference parameters of an MCDA model by defining a polyhedron of acceptable values denoted by T_{k-1} (at iteration $k-1$). This polyhedron is defined by the following (general) consistent system of $m-1$ linear constraints on n variables $x_1, x_2, \ldots, x_n$:

$$\text{System (1):} \quad \begin{cases} \sum_{j=1}^{n} \alpha_{1j} x_j & \geqslant \beta_1, \\ & \vdots \\ \sum_{j=1}^{n} \alpha_{(m-1)j} x_j & \geqslant \beta_{m-1}, \end{cases} \qquad \alpha_{ij}, \beta_i \in \mathbb{R}, \ \ i = 1, \ldots, m-1, \ \ j = 1, \ldots, n.$$

Let us consider a new constraint $\sum_{j=1}^{n} \alpha_{mj} x_j \geqslant \beta_m$ that, when added to the System (1), leads to System (2) containing m linear constraints, which is now inconsistent:

[1] It is always possible to increase the value of Ω if DMs are not satisfied with any of the proposed sets of constraints.

$$\text{System (2):} \quad \begin{cases} \sum_{j=1}^{n} \alpha_{1j} x_j & \geqslant \beta_1, \\ & \vdots \\ \sum_{j=1}^{n} \alpha_{(m-1)j} x_j & \geqslant \beta_{m-1}, \\ \sum_{j=1}^{m} \alpha_{mj} x_j & \geqslant \beta_m, \end{cases} \qquad \alpha_{ij}, \beta_i \in \mathbb{R}, \ \ i = 1, \ldots, m-1, \ j = 1, \ldots, n.$$

Let $I = \{1, 2, \ldots, m\}$ be the set of indices of the constraints defining T_k at iteration k (i.e., with the new constraint that makes T_k empty). Hence, $T_k = \{x \in \mathbb{R}^n : \sum_{j=1}^{n} \alpha_{ij} x_j \geqslant \beta_i \ \forall i \in I\} = \emptyset$. Let $S \subseteq I$ denote a subset of indices of constraints. We will say that S resolves System (2) if and only if the system $\sum_{j=1}^{n} \alpha_{ij} x_j \geqslant \beta_i \ \forall i \in I \setminus S$ is consistent. Let $|S|$ denote the cardinality of the set S. Our problem is to identify all minimal subsets S_i that resolve System (2) and whose cardinality is lower (or equal to) than Ω (Ω being considered as an input to the algorithms). Formally, the problem we are addressing is to determine p sets $S_1, \ldots, S_p$ (if they exist) such that:

(i) S_i resolves System (2), $i \in \{1, 2, \ldots, p\}$.
(ii) $S_i \not\subseteq S_j, \ i, j \in \{1, \ldots, p\}, \ i \neq j$.
(iii) $|S_i| \leqslant |S_j|, \ i, j \in \{1, 2, \ldots, p\}, \ i < j$.
(iv) If there exists a set S that resolves System (2) such that $S \not\subseteq S_i \ \forall i = 1, 2, \ldots, p$, then $|S| > |S_p|$.
(v) $|S_i| \leqslant \Omega \ \forall i = 1, 2, \ldots, p$.

Since we already know that the System (1) is consistent and the System (2) is inconsistent, we can obviously set $S_1 = \{m\}$. From condition (ii), this implies that the remaining sets $S_2, \ldots, S_p$ will not include $\{m\}$. Two alternative methods to solve this problem are proposed hereafter.

It should be noted that both algorithms provide the same set of solutions (only the order of appearance of the solutions S_i having the same cardinality can vary).

The first approach makes use of $\{0, 1\}$ mixed integer linear programming while the second is based on linear programming techniques only. It is obvious that mixed integer linear programming methods are computationally more demanding than standard linear programming methods. Therefore, as the size of the problems increases, the first approach will probably become less efficient than the second one. However, the size of the problems considered for solving inconsistencies in MCDA models is not very large and both methods provide results within a computation time that is compatible with interactive decision support systems. Hence both algorithms are relevant for our problem.

2.1. An algorithm based on $\{0, 1\}$ mixed integer linear programming techniques

This first method is based on $\{0, 1\}$ mixed integer linear programming techniques. It allows the identification of p subsets of constraints that, when removed, make the polyhedron T_k feasible; this is done through $p - 1$ successive optimizations $(PM_2, PM_3, \ldots, PM_p)$. A similar approach can be found in [9].

The program PM_2 minimizes the number of constraints to be removed in order to make T_k feasible. The subset $S_1 = \{m\}$ is obviously the smallest subset verifying (i)–(v). The first problem PM_2 to be solved is the following:

$$PM_2 \begin{cases} \text{Min} & \sum_{i=1}^{m-1} y_i \\ \text{s.t.} & \sum_{j=1}^{n} \alpha_{ij} x_j + M y_i \geqslant \beta_i \quad \forall i \in I \setminus \{m\}, \\ & \sum_{j=1}^{n} \alpha_{mj} x_j \geqslant \beta_m, \\ & x_j \geqslant 0, \quad j = 1, \ldots, n, \\ & y_i \in \{0, 1\} \quad \forall i \in I \setminus \{m\}, \end{cases}$$

where M is a positive large number. The variables y_i, $\forall i \in I \setminus \{m\}$ are binary variables assigned to each constraint. The indices of constraints for which $y_i^* = 1$ (at the optimum of PM_2) constitute the subset S_2.

PM_3 is defined in order to compute S_3. This new program is derived from PM_2 by adding the single constraint $\sum_{i \in S_2} y_i \leqslant |S_2| - 1$. This constraint prevents PM_3 from finding an optimal solution that corresponds to (or includes) S_2. The indices of constraints for which $y_i^* = 1$ (at the optimum of PM_3) constitute the subset S_3. In order to compute $S_4, \ldots, S_p$, we proceed similarly: each new program is formed by adding one constraint to the previous program. It should be noted that, when multiple optimum solutions exist (i.e., sets S_i and S_j having the same cardinality), all optimal solutions are computed (only the order by which these solutions are obtained can change, depending on the implementation of the solver used to solve PM_k).

The outline of the algorithm is the following:

```
Begin
  k ← 2
  moresol ← true
  While moresol
    Solve PM_k
    If (PM_k has no solution) or (PM_k has an optimal value > Ω)
    Then
        moresol ← false
    Else
        S_k ← {i ∈ I − {m} : y_i^* = 1}
        Add constraint Σ_{i∈S_k} y_i ⩽ |S_k| − 1 to PM_k so as to define PM_{k+1}
        k ← k+1
    End if
  End while
End
```

Solving programs such as PM_k is a well-known problem in operations research (e.g. see [15]) which we do not address here. This algorithm is an adaptation to our particular type of problem (conditions (i)–(v), p. 4) of a well-known idea based on {0, 1} mixed integer programming (see for instance [9] in a MCDA context).

As an illustrative example, let us consider the example (1)–(8) (see Fig. 1). The system (1)–(7) is consistent (cf. shaded area Fig. 1), but when (8) is added (see dashed constraint in Fig. 1), the system (1)–(8) becomes inconsistent:

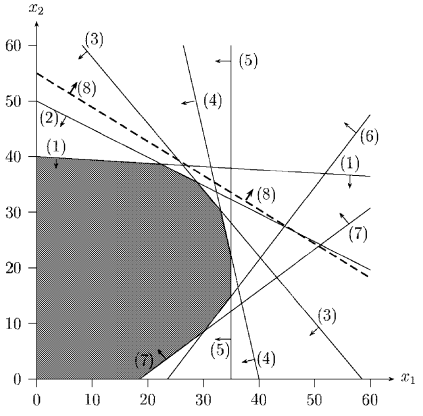

Fig. 1. Feasible set.

$$-0.05x_1 - x_2 \geqslant -40, \tag{1}$$

$$-0.5x_1 - x_2 \geqslant -50, \tag{2}$$

$$-1.2x_1 - x_2 \geqslant -70, \tag{3}$$

$$-4.5x_1 - x_2 \geqslant -179, \tag{4}$$

$$-x_1 \geqslant -35, \tag{5}$$

$$-1.3x_1 + x_2 \geqslant -60, \tag{6}$$

$$-0.75x_1 + x_2 \geqslant -14, \tag{7}$$

$$0.6x_1 + x_2 \geqslant 55. \tag{8}$$

We then build the following *PM* model where $y_i \in \{0,1\},\ i = 1,\ldots,7$:

$$\text{Min} \quad \sum_{i=1}^{7} y_i \tag{9}$$

$$\text{s.t.} \quad -0.05x_1 - x_2 \geqslant -40 - My_1, \tag{10}$$

$$-0.5x_1 - x_2 \geqslant -50 - My_2, \tag{11}$$

$$-1.2x_1 - x_2 \geqslant -70 - My_3, \tag{12}$$

$$-4.5x_1 - x_2 \geqslant -179 - My_4 \tag{13}$$

$$-x_1 \geqslant -35 - My_5, \tag{14}$$

$$-1.3x_1 + x_2 \geqslant -60 - My_6, \tag{15}$$

$$-0.75x_1 + x_2 \geqslant -14 - My_7, \tag{16}$$

$$0.6x_1 + x_2 \geqslant 55. \tag{17}$$

Let us compute, step by step, all the feasible solutions for the above example as follows (Appendix A presents for each solution the feasible sets obtained after solving inconsistency):

1. $S_1 = \{8\}$ (trivial solution).
2. In the first stage, we obtain $S_2 = \{1,2\}$, i.e., $y_1^* = y_2^* = 1$. The constraint $y_1 + y_2 \leqslant 1$ is then added to the model.
3. In the second stage, after optimizing PM the solution is $S_3 = \{2,3\}$, i.e., $y_2^* = y_3^* = 1$. We add the constraint $y_2 + y_3 \leqslant 1$ to the model.
4. In the third stage, we obtain $S_4 = \{3,4,5,6\}$, i.e., $y_3^* = y_4^* = y_5^* = y_6^* = 1$. We add the constraint $y_3 + y_4 + y_5 + y_6 \leqslant 3$.
5. The problem becomes infeasible, meaning that there are no more alternatives to solve our problem.

2.2. An algorithm based on linear programming techniques

In this section we propose a second algorithm to solve the problem we are addressing, i.e., to find the sets $S_2,\ldots,S_p$ (since we are considering $S_1 = \{m\}$), based on the results presented in Appendix B. The algorithm starts by considering the polyhedron defined by System (1), a consistent system of $m-1$ linear constraints, and maximizes the quantity $\sum_{j=1}^{n} \alpha_{mj}x_j$, over that polyedron by solving a linear program (LP). Since the system becomes inconsistent after adding the constraint $\sum_{j=1}^{n} \alpha_{mj}x_j \geqslant \beta_m$, it is obvious that the LP yields an

optimal solution (strictly) lower than β_m. From Proposition 2 (Appendix B), if we want System (2) to become feasible and maintain the constraint $\sum_{j=1}^{n} \alpha_{mj} x_j \geqslant \beta_m$ at the same time, we need to remove at least one of the constraints that is binding at the optimal solution. For each binding constraint, the algorithm will consider a new system, which is equal to System (1) without that constraint. Each of these systems originates a new LP with the same objective function. The maximum value of each LP will decide whether the respective constraint that was removed resolves System (2). For those LPs where $\sum_{j=1}^{n} \alpha_{mj} x_j < \beta_m$ at the optimum, a new system is considered for each of the binding constraints, and so on. A key aspect in this algorithm is that all systems that result from removing i constraints from System (1) are examined before any system that results from removing $i+1$ constraints from System (1).

Let C be a first-in-first-out (FIFO) queue structure whose elements $C_1, C_2, \ldots$ are sets contained in $I \setminus \{m\} = \{1, \ldots, m-1\}$ ("candidates"). Each of these sets, C_k, is a candidate to resolve System (2), corresponding to a system $\sum_{j=1}^{n} \alpha_{ij} x_j \geqslant \beta_i \ \forall i \in I \setminus (\{m\} \cup C_k)$. Let S be a FIFO queue structure whose elements are sets contained in $I \setminus \{m\}$ that resolve System (2) ("solutions").

For any set $F \subseteq \{1, \ldots, m-1\}$, let LP($F$) denote the linear program that maximizes $\sum_{j=1}^{n} \alpha_{mj} x_j$, subject to the constraints of System (1), except the constraints in F, i.e., $\mathrm{LP}(F) : \max \sum_{j=1}^{n} \alpha_{mj} x_j : \sum_{j=1}^{n} \alpha_{ij} x_j \geqslant \beta_i \ \forall i \in I \setminus (F \cup \{m\})$. Let $x^*(F)$ denote an optimal solution for LP(F). Let $B(F)$ be the set of indices of the constraints that are active (binding) at the solution $x^*(F)$, i.e., $B(F) = \{i \in I \setminus (F \cup \{m\})$ s.t. $\sum_{j=1}^{n} \alpha_{ij} x_j^*(F) = \beta_i\}$. Using this notation, an algorithm to propose p solutions for inconsistency by non-decreasing order of cardinality, using LP, is the following:

```
Begin
  C ← ∅
  S ← ∅
  card ← 1
  Solve LP(∅)
  For each constraint index i ∈ B(∅) do
    Add set {i} to queue C
  While C ≠ ∅ and card ⩽ Ω do
    F ← first (oldest) element from C
    card ← |F|
    If card ⩽ Ω Then
        Solve LP(F)
        If LP(F) is unbounded or has an optimal value ⩾ β_m Then
            Add set F to queue S
        Else
            For each constraint index i ∈ B(F) do
                If ( ∄X ∈ S ∪ C s. t. F ∪ {i} ⊇ X ) Then Add F ∪ {i} to queue C
                End if
        End if
    End if
  End while
End
```

Justification for the algorithm:

1. This algorithm presents the solutions S_2 to S_p by non-decreasing order of cardinality. Before the while loop, the algorithm considers as candidates sets of cardinality equal to 1. In the while loop, when the set F at the head of the candidates queue C is tested (by solving LP(F)) and fails, the algorithm places

at the tail of that queue other potential solutions whose cardinality equals $|F|+1$. All candidates of cardinality $|F|$ are tested before those of higher cardinality.

2. If a set R resolves System (2) and no subset of R resolves System (2), then the algorithm will find R for a sufficiently large value of Ω. We will show that this is true by induction. Suppose there exists a set $R=\{s_1,s_2,\ldots,s_{|R|}\}$ that resolves System (2). Before the while loop, the elements in $B(\emptyset)$ are considered as candidates. From Proposition 2 (see Appendix B), $B(\emptyset)\cap R\neq\emptyset$, i.e., there exists an element $s_{(1)}\in R$ that enters the candidates queue C. During the while loop, at a given moment, the candidates of cardinality k $(1\leqslant k<|R|)$ will start to appear at the head of the candidates queue C. If one of these elements F is such that $F\subset R$, then solving LP(F) yields an optimal value which is less than β_m. From Proposition 2, $B(F)\cap R\setminus F\neq\emptyset$, i.e., there exists an element $S_{(k)}$ that belongs to R and does not belong to F. This element is appended to F to constitute a set $\{S_{(1)},\ldots,S_{(k)}\}\in R$ that enters queue C. Some iterations later, the candidates of cardinality $|R|$ will start to appear at the head of C. One of these candidates is R, which will be declared a solution since LP(R) will either be unbounded, or have an optimal value that is not less than β_m.

One way to implement this algorithm is to solve the linear program before the while loop, and then another linear program for each iteration. An alternative way is to solve the initial LP($\emptyset$) and to save the simplex tableau corresponding to the removal of each constraint put in the candidates queue C. In the while loop, solving LP(F) will amount to performing a single simplex iteration from the corresponding saved tableau. Then, for each set inserted in C, a new tableau must be saved. This would be faster, but requires more memory.

Let us solve the previous example (1)–(8) (see Fig. 1) with this algorithm. The results will be the same obtained with the algorithm presented in the previous subsection.

1. Initially, C and S are empty queues. LP($\emptyset$) amounts to maximize $0.6x_1+x_2$, subject to constraints (1)–(7). Solving LP($\emptyset$) yields an optimum value of 52.857, meaning that the constraint $0.6x_1+x_2\geqslant 55$ (8) cannot be satisfied. The set of indices of the constraints that are active (binding) at the optimal solution is $B(\emptyset)=\{2,3\}$. This implies that at least one of these two constraints must be dropped to satisfy the constraint (8). The sets $\{2\}$ and $\{3\}$ are added to C. Hence:

   $C=\{\{2\},\{3\}\};\quad S=\emptyset.$

2. The set $\{2\}$ is removed from C. A new problem LP($\{2\}$) is formed with the same objective and considering all the constraints except (2), i.e., maximize $0.6x_1+x_2$, subject to constraints (1), (3)–(7). The solution of LP($\{2\}$) yields an optimum value of 54.348, which is still less than 55. Hence, removing only the constraint (2) is not enough to allow $0.6x_1+x_2\geqslant 55$ (8). The set of binding constraints is now $B(\{2\})=\{1,3\}$. The sets $\{1,2\}$ and $\{2,3\}$ are added to C. Hence:

   $C=\{\{3\},\{1,2\},\{2,3\}\};\quad S=\emptyset.$

3. The set $\{3\}$ is removed from C. A new problem LP($\{3\}$) is formed with the same objective and considering the constraints (1), (2), (4)–(7). Its solution yields an optimum value of 53.225. Hence, $0.6x_1+x_2\geqslant 55$ (8) cannot hold. The set of binding constraints is now $B(\{3\})=\{2,4\}$. Only the set $\{3,4\}$ is added to C, since the other set $\{2,3\}$ has already been inserted. Hence:

   $C=\{\{1,2\},\{2,3\},\{3,4\}\};\quad S=\emptyset.$

4. The set $\{1,2\}$ is removed from C. A new problem LP($\{1,2\}$) if formed, to maximize $0.6x_1+x_2$, subject to (3)–(7), yielding an optimum value of $70\geqslant 55$. This means that the system (3)–(8) is consistent (see Fig. 4 in Appendix A). Therefore, $\{1,2\}$ enters the queue S and no element is added to C. Hence:

   $C=\{\{2,3\},\{3,4\}\};\quad S=\{\{1,2\}\}.$

5. The set $\{2,3\}$ is removed from C. The solution of LP($\{2,3\}$) (i.e. maximize $0.6x_1 + x_2$, subject to (1), (4)–(7)) yields an optimum value of $57.180 \geqslant 55$. Therefore, the system (1), (4)–(8) is consistent (see Fig. 5 in Appendix A) and $\{2,3\}$ enters S. Hence:

   $C = \{\{3,4\}\}; \quad S = \{\{1,2\},\{2,3\}\}.$

6. The set $\{3,4\}$ is removed from C. The solution of LP($\{3,4\}$) yields an optimum value of $53.5 < 55$. $B(\{3,4\}) = \{2,5\}$. The set $\{3,4,5\}$ is added to C (the other set $\{2,3,4\}$ contains the set $\{2,3\}$ already in S). Hence:

   $C = \{\{3,4,5\}\}; \quad S = \{\{1,2\},\{2,3\}\}.$

7. The set $\{3,4,5\}$ is removed from C. The solution of LP($\{3,4,5\}$) yields an optimum value of $54.444 < 55$. $B(\{3,4,5\}) = \{2,6\}$. The set $\{3,4,5,6\}$ is added to C (the other set $\{2,3,4,5\}$ contains the set $\{2,3\}$ already in S). Hence:

   $C = \{\{3,4,5,6\}\}; \quad S = \{\{1,2\},\{2,3\}\}.$

8. The set $\{3,4,5,6\}$ is removed from C. The solution of LP($\{3,4,5,6\}$) yields an optimum value of $55.12 \geqslant 55$. Therefore, the system (1), (2), (7), (8) is consistent (see Fig. 6 in Appendix A) and $\{3,4,5,6\}$ is inserted in S. Hence:

   $C = \emptyset; \quad S = \{\{1,2\},\{2,3\},\{3,4,5,6\}\}.$

The algorithm stops since there are no more candidate solutions.

3. An application in the context of the inference of Electre Tri weights

We have proposed two algorithms for solving inconsistencies among constraints on the parameters of an MCDA model. The purpose of these algorithms, as far as MCDA is concerned, is to provide support to DMs in the preference elicitation process when they are confronted to inconsistent information. Such inconsistencies stem from information, which can be either holistic preferences or explicit constraints on the parameter values. These algorithms can be used in the context of different aggregation models. In this section we consider the multiple criteria sorting method Electre Tri and we illustrate the support provided by the algorithms during the preference elicitation based on real-world data.

3.1. Interactive inference of weights in Electre Tri

Electre Tri is a multiple criteria sorting method (see [12,18]) that assigns each alternative a_i from a set A to one of the n_{cat} pre-defined ordered categories $C_1, C_2, \ldots, C_{n_{\text{cat}}}$ (defined by multi-criteria limits $\Delta = \{b_h, h = 1, \ldots, n_{\text{cat}} - 1\}$) on the basis of its evaluation on n_{crit} criteria. In order to assign each alternative to a category, Electre Tri defines an outranking relation $\succeq$ ($\succeq \subset A \times \Delta \cup \Delta \times A$) and grounds the assignment of an alternative a_i on the way a_i compares to the limits of categories (a description of the Electre Tri method is given in Appendix C).

Mousseau and Slowinski [13] proposed a methodology to infer the values of the parameters from assignment examples through a certain form of regression (hence avoiding direct elicitation of the model parameters). This methodology proceeds in a way that requires from the DM much less cognitive effort: the elicitation of parameters is done indirectly using holistic information given by the DM through assignment examples, i.e, alternatives assigned by the DM to categories according to his/her comprehensive preferences.

We consider the inference process aiming at determining the criteria weights and the cutting level λ. We assume that the other preference parameters (category limits, discrimination thresholds) are fixed and that no veto phenomenon occurs (see [5,13]). In order to minimize the differences between the assignments made

by Electre Tri and the assignments made by the DM, a mathematical program infers the values for these parameters that best restore the assignment examples (see Section 3.2). The DM can tune up the model in the course of an interactive procedure that aims at finding weights and cutting level that are as compatible as possible with the assignment examples given by the DM (see Fig. 2). During this process, the DM may either (1) revise the assignment examples or (2) fix values (or intervals of variation) for the weights w_j or λ. In the first case, the DM may add (or delete) some assignment examples, or change the assignment of some alternatives. In the second case, the DM can change the additional information on the range of variation of w_j or λ according to his/her own intuition.

At some point, the DM may provide inconsistent information. The information is said to be inconsistent when no combinations of values for w_j and λ comply with the assignment examples (and explicit constraints on w_j and λ) given by the DM. In such situation the DM is asked to reconsider his/her judgments, i.e., remove or relax some previous statements. However, determining the reason for the inconsistency in order to reconsider his/her judgment is cognitively very demanding for the DM. The algorithms presented in the preceding section provide a substantial help to the DM in order to solve the inconsistency by providing him/her with alternative ways to restore inconsistency. More precisely, the algorithms propose to the DM different subsets of pieces of information (assignment examples or explicit constraints) whose deletion solve inconsistency.

3.2. Mathematical program to be solved at each iteration

At each iteration of the interactive weight elicitation process, a mathematical program is solved in order to infer values for weights $(w_j,\ j = 1, \ldots, n_{\text{crit}})$ and cutting level (λ) that best match the information provided by the DM.

As an input to this inference program, the DM specifies assignment examples, i.e., alternatives for which he/she provides a desired assignment (*alternative* a_i, *should be assigned to category* C_k, $a_i \rightarrow C_k$). In Electre Tri

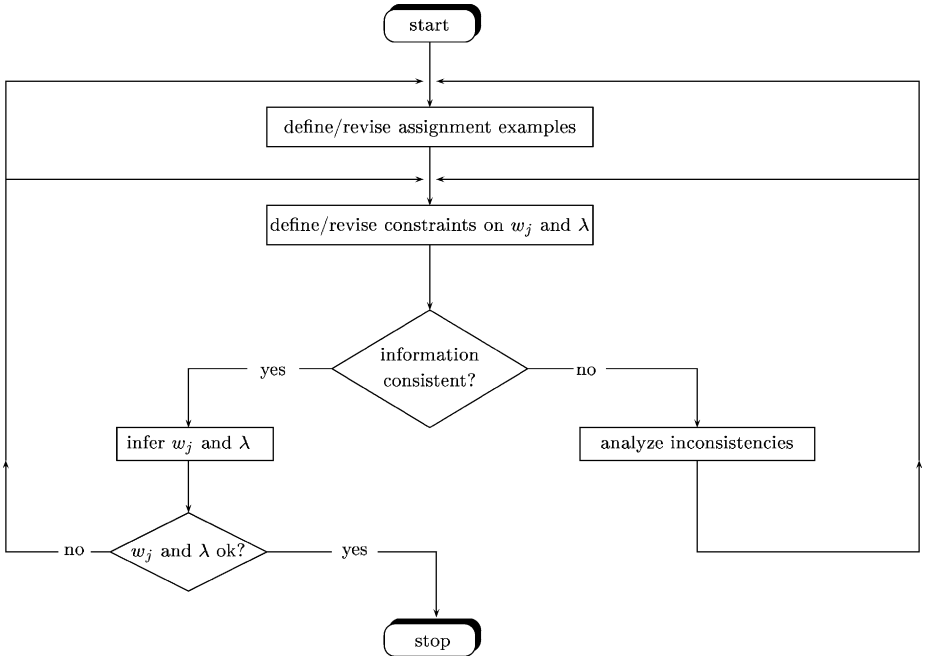

Fig. 2. Interactive inference process for determining ω_j and λ.

(pessimistic assignment rule), $a_i \to C_k$ holds iff a_i is at least as good as b_{k-i} but a_i is not at least as good as b_k (b_{k-1} and b_k being the lower and upper limits of C_k). Hence, according to the Electre Tri pessimistic assignment rule (see Appendix C), each assignment example induces two linear constraints on the weights w_j and cutting level λ where $C_j(a_i, b_{k-1})$ and $C_j(a_i, b_k)$ are partial concordance indices whose values are known:

$$\sum_{j=1}^{n_{\text{crit}}} c_j(a_i, b_{k-1}) w_j \geqslant \lambda, \tag{18}$$

$$\sum_{j=1}^{n_{\text{crit}}} c_j(a_i, b_k) w_j < \lambda. \tag{19}$$

The DM can also directly provide constraints on the weights that express his/her intuitive views concerning the relative importance of criteria. Each of these constraints represents a pair-wise comparison of coalitions of criteria (e.g., "$\{g_2\}$ *is at least as important as* $\{g_3, g_5\}$, which is interpreted in Electre Tri by the constraint $w_2 \geqslant w_3 + w_5$). Hence, each constraint (both for an assignment example or a pair-wise comparison of coalitions of criteria) has the following general form:

$$\sum_{j=1}^{n_{\text{crit}}} \alpha_j w_j \geqslant 0, \quad \text{where } \alpha_j \in \{1, 0, -1\}. \tag{20}$$

The DM can also directly provide bounds for the cutting level λ (if the DM does not provide such bounds, λ is theoretically bounded in the interval $[0.5, 1]$, see Appendix C):

$$\lambda \in [\lambda_{\min}, \lambda^{\max}], \quad \text{where } \lambda_{\min} \geqslant 0.5 \quad \text{and} \quad \lambda^{\max} \leqslant 1. \tag{21}$$

Let us suppose that the DM has specified, at the kth iteration of the interactive elicitation process:

- a set of assignment examples $A_k^* \subset A$ ($a_i \to C_{h_i}\ \forall a_i \in A_k^*$, defining a polyhedron T_k^a of acceptable values for w_j and λ),
- a list of r constraints on the weights of the form: $\sum_{j=1}^{n_{\text{crit}}} \alpha_{lj} w_j \geqslant 0,\ l = 1, \ldots, r,$ where $\alpha_{lj} \in \{1, 0, -1\}$ (defining a polyhedron T_k^w of acceptable values for w_j),
- a lower bound $\lambda_{\min} \geqslant 0.5$ and upper bound $\lambda^{\max} \leqslant 1$ for λ defining a polyhedron T_k^λ of acceptable values for λ).

The following linear program infers values for $w_j,\ j = 1, \ldots, n_{\text{crit}}$ and λ that "best" match the provided information (ε being a small positive constant):

$$\text{Max} \quad \sigma \tag{22}$$

$$\text{s.t.} \quad \sum_{j=1}^{n_{\text{crit}}} w_j c_j(a_i, b_{h_i - 1}) - \lambda \geqslant \sigma \quad \forall a_i \in A^*, \tag{23}$$

$$\lambda - \sum_{j=1}^{n_{\text{crit}}} w_j c_j(a_i, b_{h_i}) - \varepsilon \geqslant \sigma \quad \forall a_i \in A^*, \tag{24}$$

$$\sum_{j=1}^{n_{\text{crit}}} \alpha_{lj} w_j \geqslant \sigma, \quad l = 1, \ldots, r, \text{ where } \alpha_{lj} \in \{1, 0, -1\}, \tag{25}$$

$$\lambda - \lambda_{\min} \geqslant \sigma, \tag{26}$$

$$\lambda^{\max} - \lambda \geqslant \sigma, \tag{27}$$

$$\sum_{j=1}^{n_{\text{crit}}} w_j = 1, \quad \varepsilon \leqslant w_j \leqslant 0.5\ \forall_j = 1, \ldots, n_{\text{crit}},\ \sigma \text{ free}. \tag{28}$$

In this linear program, (23), (24) account for the constraints (18), (19), (25) account for the constraints (20) and (26), (27) account for the constraints (21). If the optimum value of this LP is positive, then all constraints (18)–(21) are fulfilled (w_j and λ fulfill the explicit constraints and all alternatives from the reference set A^* are "correctly" assigned for all $\lambda' \in [\lambda - \sigma, \lambda + \sigma]$. If $\sigma < 0$, then no combination of values for w_j and λ satisfies the information provided by the DM.

Suppose that $\sigma < 0$ (i.e., the information is inconsistent) and that it was not the case at the preceding iteration: the DM just added a constraint, either yielded by an assignment example or a pair-wise comparison of coalitions of criteria, that makes the polyhedron of acceptable values for w_j and λ empty. Let us remark that although an assignment example usually corresponds to a pair of constraints, only one will be the cause for the inconsistency, due to the assignment rules of Electre Tri (see Appendix C).

Identifying the reasons for inconsistency in the preference information can obviously be supported by the algorithms proposed in the previous section. Resolving the inconsistencies can be performed by deleting a subset of constraints yielding a non-empty polyhedron. The algorithms presented in this paper provide several such subsets among which the DM must choose in order to retrieve a consistent information concerning the weights and the cutting level. In such interaction with a DM, the results of the algorithm are not presented as subsets of constraints but rather in terms of the assignment examples and/or pair-wise comparison of coalitions of criteria.

3.3. Illustration of inconsistency resolution

In order to illustrate the use of the algorithms proposed in the previous section within the context of weights inference in Electre Tri, let us consider a decision problem considered in [6]. The data are adapted from a real-world application in the banking sector and the preference elicitation process was designed a posteriori and is fictitious (but realistic): 39 firms evaluated on seven criteria (see Appendix D) are to be assigned to five categories according to their risk of failure (C_1: high risk,...,C_5: low risk). As this section focuses on the use of the algorithms for MCDA purposes, we do not describe how the algorithms arrive to the solution. Section 2 provides the reader with step-by-step illustrations of the algorithms.

At a given stage of the interaction, let us suppose that the DM has provided the following consistent information $\{w_2 \geqslant w_1, w_2 \geqslant w_3, w_2 \geqslant w_4, w_2 \geqslant w_6, w_2 \geqslant w_7, w_3 \geqslant w_4, a_1 \to C_5, a_{28} \to C_1\}$ and he/she wants to add the assignment example $\{a_{31} \to C_2\}$. The resulting set of constraints on w_j and λ is inconsistent:

$$-w_1 + w_2 \geqslant 0, \tag{29}$$

$$w_2 - w_3 \geqslant 0, \tag{30}$$

$$w_2 - w_4 \geqslant 0, \tag{31}$$

$$w_2 - w_6 \geqslant 0, \tag{32}$$

$$w_2 - w_7 \geqslant 0, \tag{33}$$

$$w_3 - w_4 \geqslant 0, \tag{34}$$

$$w_4 + w_5 + w_6 - \lambda \geqslant 0, \tag{35}$$

$$-w_1 - w_2 - w_3 - w_4 - w_6 - w_7 + \lambda \geqslant \varepsilon, \tag{36}$$

$$w_1 + 0.75w_2 + w_3 + w_5 + w_6 + w_7 - \lambda \geqslant 0, \tag{37}$$

$$-w_1 - w_3 - w_5 - w_6 - w_7 + \lambda \geqslant \varepsilon, \tag{38}$$

$$\lambda \geqslant 0.6, \tag{39}$$

$$-\lambda \geqslant -0.99, \tag{40}$$

where $\sum_{j=1}^{7} w_{j=1}$ (i.e., weights normalization) and $w_j \in [0.01, 0.49]\ \forall j = 1, \ldots, 7$ (these bounds make it impossible for any criterion to "weigh" more than all others, i.e., to be a dictator) are additional constraints that cannot be removed. The constraints (29)–(34) correspond to explicit constraints on the weights w_j. The constraints (35)–(38) correspond to necessary and sufficient conditions for the assignment examples ($a_1 \to C_5,\ a_{28} \to C_1$ and $a_{31} \to C_2$) to be respected (the interested reader will find in [12] explanations on how these constraints are derived from the assignment examples). The constraints (39), (40) correspond to explicit constraints on the cutting level λ, stated as *I feel that the cutting level should be at least 0.6 and lower than 1* (when the DM does not specify any bounds on the cutting level, λ is theoretically constrained in the interval $[0.5, 1]$, see Appendix C.2).

The algorithms provide the information (and the corresponding constraints) to remove in order to retrieve consistency. Both algorithms provide as an output $S_1 = \{(38)\},\ S_2 = \{(34)\}$ and $S_3 = \{(35)\}$. In this case, the DM should be told that the information he/she provided is inconsistent and that solving this inconsistency requires him/her to reconsider one of the following pieces of information:

- $w_3 \geqslant w_4$ (i.e., constraint (34)),
- $a_1 \to C_5$ (i.e., constraint (35), a_1 should be assigned to a lower category than C_5) or,
- $a_{31} \to C_2$ (i.e., constraint (38), a_{31} should be assigned to a higher category than C_2). [2]

It should be emphasized that, in the absence of any support, the DM would have to check himself/herself the reason for inconsistency. Let us suppose that the DM would choose to keep the older constraints $w_3 \geqslant w_4$ and $a_1 \to C_5$, and relax the new one instead, which becomes $a_{31} \to \{C_2, C_3\}$. This change would make the system consistent and this fact allows to compute a combination of parameter values that respects all the constraints (including the ones corresponding to assignment examples), as well as to determine a range of possible assignments for each alternative without violating any constraint (see example in Fig. 3). The DM can then stop the interactive process if he/she is satisfied with the output, or pursue the process by integrating additional assignment examples and/or preference information.

In such an interactive process (see Fig. 2), it is possible to compute (each time w_j and λ are inferred) the range of categories to which an alternative can be assigned according to a polyhedron of possible values for the weights w_j and cutting level λ (see [4]). Doing so, the DM observes that alternative a_{39} cannot be assigned to category C_3 (a_{39} can be assigned only to C_4, see Fig. 3) and wonders why. It is easy to answer such question using the presented algorithms.

To answer this question, the information $a_{39} \to C_3$ would be added to the (consistent) system $\{w_2 \geqslant w_1,\ w_2 \geqslant w_3,\ w_2 \geqslant w_4,\ w_2 \geqslant w_4,\ w_2 \geqslant w_6,\ w_2 \geqslant w_7,\ w_3 \geqslant w_4,\ a_1 \to C_5,\ a_{28} \to C_1,\ a_{31} \to C_2\}$. The resulting (inconsistent) system is

$$-w_1 + w_2 \geqslant \varepsilon, \tag{41}$$

$$w_2 - w_3 \geqslant \varepsilon, \tag{42}$$

$$w_2 - w_4 \geqslant \varepsilon, \tag{43}$$

[2] $a_{31} \to C_2$ causes two constraints, (37) and (38), but it is easy to check that (38) is the one that is violated by the inferred values in the previous iteration; that is the reason why $S_1 = \{(38)\}$.

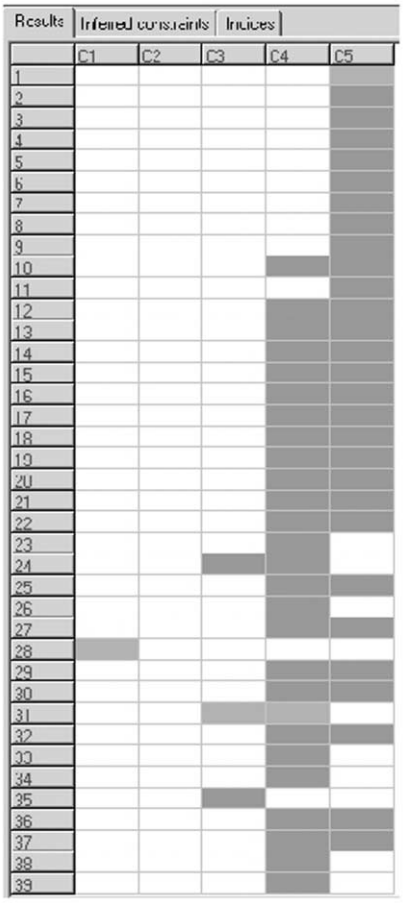

Fig. 3. Possible assignments of alternatives considering the preference information $\{\omega_2 \geqslant \omega_1, \omega_2 \geqslant \omega_3, \omega_2 \geqslant \omega_4, \omega_2 \geqslant \omega_6, \omega_2 \geqslant \omega_7, \omega_3 \geqslant \omega_4, a_1 \to C_5, a_{28} \to C_1, a_{31} \to C_3$ or $C_4, \lambda \in [0.6, 0.99]\}$.

$$w_2 - w_6 \geqslant \varepsilon, \tag{44}$$

$$w_2 - w_7 \geqslant \varepsilon, \tag{45}$$

$$w_3 - w_4 \geqslant \varepsilon, \tag{46}$$

$$w_4 + w_5 + w_6 - \lambda \geqslant 0, \tag{47}$$

$$-w_1 - w_2 - w_3 - w_4 - w_6 - w_7 + \lambda \geqslant \varepsilon, \tag{48}$$

$$w_1 + 0.75w_2 + w_3 + w_5 + w_6 + w_7 - \lambda \geqslant 0, \tag{49}$$

$$-w_1 - w_5 - w_7 + \lambda \geqslant \varepsilon, \tag{50}$$

$$w_1 + w_2 + w_3 + w_4 + w_5 - \lambda \geqslant 0, \tag{51}$$

$$-w_1 - w_2 - w_5 + \lambda \geqslant \varepsilon, \tag{52}$$

$$\lambda \geqslant 0.6, \tag{53}$$

$$-\lambda \geqslant -0.99, \tag{54}$$

where $\sum_{j=1}^{7} w_j = 1$ and $w_j \in [0.01, 0.49]$ $\forall j = 1, \ldots, 7$. The constraint (49) prevents a_{31} from being lower than C_2, whereas the constraint (50) prevents a_{31} from being higher than C_3. The constraints (51) and (52) prevent a_{39} from being lower or higher (respectively) than C_3.

The algorithms provide the information (and the corresponding constraints) to remove in order to retrieve consistency. Both algorithms provide as an output $S_1 = \{(44)\}$, $S_2 = \{(47)\}$ and $S_3 = \{(52)\}$. Hence, a_{39} cannot be assigned to C_3 without reconsidering either $w_2 > w_6$ (44) or $a_1 \to C_5$ (47). The DM might note that imposing $a_1 \to C_5$ was already one of the constraints causing the infeasibility in the prior iteration, which might influence his/her reaction to these results.

4. Conclusion and further research

In this paper, we have proposed two alternative algorithms to deal with inconsistencies among constraints on the parameters of an MCDA model. The inconsistencies considered here correspond to situations in which the DM specifies a list of linear constraints on preference parameter values that originate an empty polyhedron. More specifically, these algorithms allow one to compute subsets of constraints that, when removed, yield a non-empty polyhedron of acceptable values for preference parameters.

When eliciting preferences using an aggregation/disaggregation process, the algorithms presented in Section 2 are particularly useful for resolving inconsistent information provided by the DM. Section 3 describes and illustrates how these algorithms can be used to solve inconsistency when inferring the weights in the Electre Tri method from assignment examples.

The results presented in this paper suggest further research. First, it is obvious that the proposed algorithms can be applied to solve inconsistencies on preference parameters in various aggregation models besides Electre Tri. For example, these algorithms could be integrated in UTA-like methods [7] in order to provide support to the DMs when they provide "inconsistent" judgments.

Second, if some ordinal confidence index is attached to each constraint provided by the DM, it would be interesting to find the "smallest" subsets of constraints in which the DM has the least confidence.

Lastly, in this paper we have considered the reduction of inconsistencies through the deletion of subsets of constraints. It would be interesting to try to restore consistency by relaxing, rather than deleting subsets of constraints.

Acknowledgements

This work has benefited from the luso-french grant no. 500B4 (ICCTI/Ambassade de France au Portugal). We are grateful to the three anonymous referees whose remarks helped us to improve earlier versions of this paper.

Appendix A. Feasible sets obtained after solving inconsistency

See Figs. 4–6.

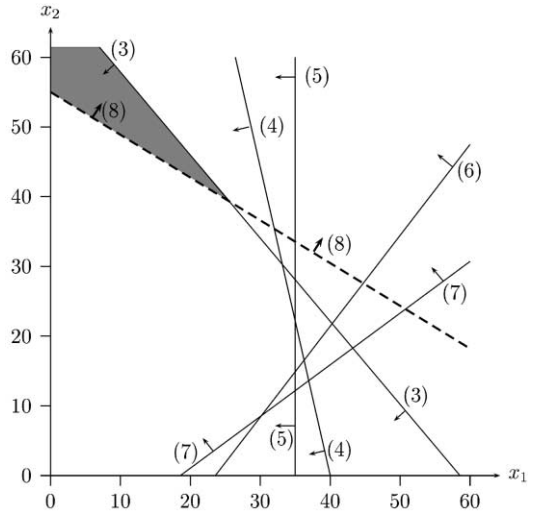

Fig. 4. Feasible set of System (2) when constraints (1) and (2) are suppressed.

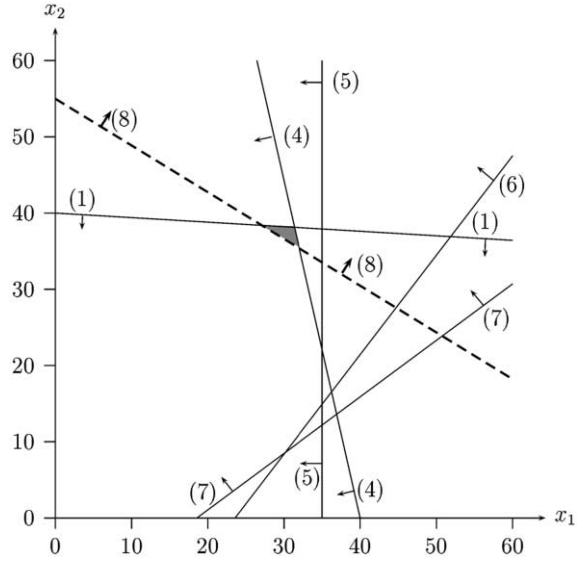

Fig. 5. Feasible set of System (2) when constraints (2) and (3) are suppressed.

Appendix B. Results concerning the second algorithm

Proposition 1. *Let* $F \subseteq C\{1,\ldots,m-1\}$ *be a set of indices of constraints. Let* $LP(F)$ *denote the linear program that maximizes* $\sum_{j=1}^{n} \alpha_{mj}x_j$, *subject to constraints of System* (1), *except the constraints in F, i.e.,* $LP(F) : \sum_{j=1}^{n} \alpha_{mj}x_j$ *s.t.* $\sum_{j=1}^{n} \alpha_{ij}x_j \geqslant \beta_i \ \forall_i \in I \setminus (F \cup \{m\})$. *It holds*:

(a) $LP(F)$ *is always feasible and,*

(b) *F resolves System* (2) $\iff$ LP(F) *is unbounded or its optimal value is not less than* β_m.

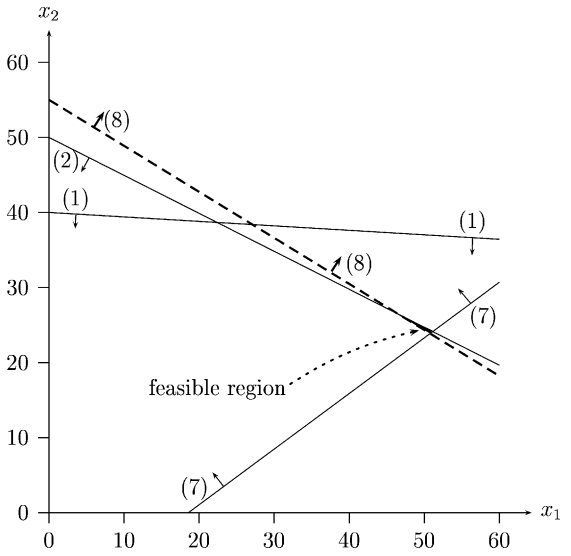

Fig. 6. Feasible set of System (2) when constraints (3)–(6) are suppressed.

Proof

(a) Since System (1) is consistent, it remains consistent after removing some of its constraints; hence, the linear program is feasible.

(b) When LP(F) is unbounded or when the optimal value of LP(F) is greater than, or equal to β_m, then all the constraints except the ones in F are satisfied, including the last constraint in System (2), i.e., S resolves System (2). Otherwise, the last constraint in System (2) is violated and F does not resolve System (2). □

Let

$$x^*(F) = \arg\max \left\{ \sum_{j=1}^{n} \alpha_{mj} x_j : \sum_{j=1}^{n} \alpha_{ij} x_j \geqslant \beta_i \ \forall i \in I \setminus (F \cup \{m\}) \right\}$$

denote an optimal solution for LP(F). Let $B(F)$ be the set of indices of the constraints that are active (binding) at the solution $x^*(F)$, i.e.,

$$B(F) = \left\{ i \in \setminus (F \cup \{m\}) : \sum_{j=1}^{n} \alpha_{ij} x_j^*(F) = \beta_i \right\}.$$

Proposition 2. *Let us consider R and F such that $F \subset R \subset I \setminus \{m\}$. If F does not resolve System* (2) *and R resolves System* (2), *then*, $B(F) \cap (R \setminus F) \neq 0$.

Proof. From Proposition 1, R resolves System (2) $\Rightarrow$ LP(R) is unbounded or its optimal value is not less than β_m, and F does not resolve System (2) $\Rightarrow$ the optimal value of LP(F) is less than β_m. Let $x^*(F)$ denote the optimal solution of the linear program LP(F). Note that $x^*(F)$ is also an optimal solution of the linear program that would be formed by deleting all the constraints that are not binding at $x^*(F)$, $\max \sum_{j=1}^{n} \alpha_{mj} x_j$ s.t. $\sum_{j=1}^{n} \alpha_{ij} x_j \geqslant \beta_i \ \forall_i \in B(F)$, otherwise it would not be optimal to LP(F). Since the polyhedron

$\{x \in \mathbb{R}^n : \sum_{j=1}^{n} \alpha_{ij} x_j \geqslant \beta_i \ \forall_i \in I \setminus (F \cup \{m\})\}$ is contained in the cone $\{x \in \mathbb{R}^n : \sum_{j=1}^{n} \alpha_{ij} x_j \geqslant \beta_i \ \forall_i \in B(F)$, then the value of the objective function of $LP(F)$ can increase only if (at least) one of the constraints in $B(F)$ is removed. Thus, the fact that the objective function does increase when F is replaced by R (i.e., when $I \setminus (F \subset \{m\})$ is replaced by $I \setminus (R \subset \{m\})$ by removing constraints in $R \setminus F$) implies that at least one of the constraints in $R \setminus F$ belongs to $B(F)$. □

Appendix C. Electre Tri, a brief reminder

Electre Tri is a multiple criteria sorting method, i.e, a method that assigns alternatives to pre-defined ordered categories. The assignment of an alternative a results from the comparison of a with the profiles defining the limits of the categories. We consider n_{crit} criteria $(g_1, g_2, \ldots, g_{n_{\text{crit}}})$ and Δ the set of the profiles defining n_{cat} categories ($\Delta = \{b_1, b_2, \ldots, b_{n_{\text{cat}}} - 1\}$) being the upper limit of category C_h and the lower limit of category C_{h+1}, $h = 1, 2, \ldots, n_{\text{cat}} - 1$ (see Fig. 7, where the profiles $b_{n_{\text{cat}}}$ and b_0 correspond to the ideal and anti-ideal alternatives, respectively). In what follows, we will assume, without any loss of generality, that preferences increase with the value on each criterion.

Schematically, Electre Tri assigns alternatives to categories following two consecutive steps:

- construction of an outranking relation $\succeq$ that characterizes how alternatives compare to the limits of categories,
- exploitation of the relation $\succeq$ in order to assign each alternative to a specific category.

C.1. Construction of the outranking relation

Electre Tri builds an outranking relation $\succeq$, i.e, validates or invalidates the assertion $a \succeq b_h$ (and $b_h \succeq a$), whose meaning is *a is at least as good as* b_h. Preferences restricted to the significance axis of each criterion are defined through pseudo-criteria (see [19] for details on this double-threshold preference representation). The indifference and preference thresholds ($q_j(b_h)$ and $p_j(b_h)$) constitute the intra-criterion preferential information. They account for the imprecise nature of the evaluations $g_j(a)$ (see [16]): $q_j(b_h)$ specifies the largest difference $g_j(a) - g_j(b_h)$ that preserves indifference between a and b_h on criterion g_j while $p_j(b_h)$ represents the smallest difference $g_j(a) - g_j(b_h)$ compatible with a preference in favor of a on criterion g_j.

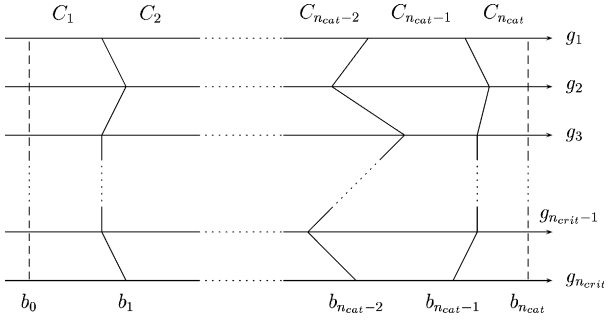

Fig. 7. Definition of categories using limit profiles.

Two types of inter-criteria preference parameters intervene in the construction of $\succeq$:

- the set of weight-importance coefficients $(w_1, w_2, \ldots, w_{n_{\text{crit}}})$ is used in the concordance test when computing the relative importance of the coalitions of criteria being in favor of the assertion $a \succeq b_h$,
- the set of veto thresholds $(v_1(b_h), v_2(b_h), \ldots, v_{n_{\text{crit}}}(b_h))$ is used in the discordance test; $v_j(b_h)$ represents the smallest difference $g_j(b_h) - g_j(a)$ incompatible with the assertion $a \succeq b_h$.

Electre Tri builds an outranking relation $\succeq$ using an index $\sigma(a, b_h) \in [0, 1]$ ($\sigma(b_h, a)$ resp.) that represents the degree of credibility of the assertion $a \succeq b_h$ ($b_h \succeq a$, resp.) $\forall a \in A,\ \forall b_h \in \Delta$. Formally, $\sigma(a, b_h) \in [0, 1]$ is defined by (C.1)–(C.5) ($\sigma(a, b_h)$ is defined similarly):

$$\sigma(a, b_h) = C(a, b_h) \cdot ND(a, b_h), \quad a \in A,\ b_h \in \Delta, \tag{C.1}$$

$$C(a, b_h) = \frac{1}{\sum_{j=1}^{n_{\text{crit}}} w_j} \cdot \sum_{j=1}^{n_{\text{crit}}} w_j c_j(a, b_h), \quad a \in A,\ b_h \in \Delta, \tag{C.2}$$

$$c_j(a, b_h) = \frac{p_j(b_h) - \min\{g_j(b_h) - g_j(a), p_j(b_h)\}}{p_j(b_h) - \min\{g_j(b_h) - g_j(a), q_j(b_h)\}}, \quad a \in A,\ b_h \in \Delta, \tag{C.3}$$

$$ND(a, b_h) = \prod_{j \in \overline{F}} \frac{1 - d_j(a, b_h)}{1 - C(a, b_h)}, \quad a \in A,\ b_h \in \Delta, \tag{C.4}$$

where $\overline{F} = \{j \in \{1, \ldots, n_{\text{crit}}\}$ such that $d_j(a, b_h) > C(a, b_h)\}$,

$$d_j(a, b_h) = 1 - \frac{v_j(b_h) - \min\{g_j(b_h) - g_j(a), v_j(b_h)\}}{v_j(b_h) - \min\{g_j(b_h) - g_j(a), p_j(b_h)\}}, \quad a \in A,\ b_h \in \Delta. \tag{C.5}$$

It should be noted that, in the absence of veto situation, $d_j(a, b_h) = 0\ \forall_j = 1, \ldots, n_{\text{crit}},\ \forall a \in A,\ \forall b_h \in \Delta$, therefore $ND(a, b_h) = 1$, and consequently $\sigma(a, b_h) = C(a, b_h)$.

C.2. Exploitation procedure

As the assignment of alternatives to categories does not result directly from the relation $\succeq$, an exploitation phase is necessary; it requires the relation $\succeq$ to be "defuzzyfied" using a so-called λ-cut: the assertion $a \succeq b_h$ ($b_h \succeq a$, resp.) is considered to be valid if $\sigma(ab_h) \geqslant \lambda$ ($\sigma(b_h a) \geqslant \lambda$, resp.), λ being a "cutting level" such that $\lambda \in [0.5, 1]$. This λ-cut determines the preference situation between a and b_h:

- $\sigma(a, b_h) \geqslant \lambda$ and $\sigma(b_h, a) \geqslant \lambda \Rightarrow a \succeq b_h$ and $b_h \succeq a \Rightarrow aIb_h$, i.e, a is indifferent to b_h,
- $\sigma(a, b_h) \geqslant \lambda$ and $\sigma(b_h, a) < \lambda \Rightarrow a \succeq b_h$ and not $b_h \succeq a \Rightarrow a \succ b_h$, i.e, a is preferred to b_h (weakly or strongly),
- $\sigma(a, b_h) < \lambda$ and $\sigma(b_h, a) \geqslant \lambda \Rightarrow$ not $a \succeq b_h$ and $b_h \succeq a \Rightarrow b_h \succ a$, i.e, b_h is preferred to a (weakly or strongly),
- $\sigma(a, b_h) < \lambda$ and $\sigma(b_h, a) < \lambda \Rightarrow$ not $a \succeq b_h$ and not $b_h \succeq a \Rightarrow aRb_h$, i.e, a is incomparable to b_h.

Remark that b_0 and b_{p+1} are denned such that $b_{p+1} \succ a$ and $a \succeq b_0\ \forall a \in A$. The role of the exploitation procedure is to analyze the way in which an alternative a compares to the profiles so as to determine the category to which a should be assigned. Two assignment procedures are available (pessimistic and optimistic), however we present the pessimistic procedure only, as it is the only one used in the example.

Pessimistic (or conjunctive) procedure:

(a) compare a successively to b_i, for $i = p, p-l, \ldots, 0$,
(b) b_h being the first profile such that $a \succeq b_h$, assign a to category $C_{h+1}(a \to C_{h+1})$.

If b_{h-1} and b_h denote the lower and upper profiles of the category C_h, the pessimistic (or conjunctive) procedure assigns alternative a to the highest category C_h such that a outranks b_{h-1}, i.e, $a \succeq b_{h-1}$. When using this procedure with $\lambda = 1$, an alternative a can be assigned to category C_h only if $g_j(a)$ equals (up to a threshold) or exceeds $g_j(b_h)$ for each criterion (conjunctive rule). When λ decreases, the conjunctive character of this procedure is weakened.

Appendix D. Dataset considered in the example

The dataset considered in Section 3.3 is adapted from [6]. The criteria are:

g_1: (Financial ratio) Earning before interest and taxes/total assets [increasing preferences];
g_2: (Financial ratio) Net income/net worth [increasing preferences];
g_3: (Financial ratio) Total liabilities/total assets [decreasing preferences];
g_4: (Financial ratio) Interest expenses/sales [decreasing preferences];
g_5: (Financial ratio) General and administrative expenses/sales [decreasing preferences];
g_6: (Qualitative criterion) Managers work experience [increasing preferences];
g_7: (Qualitative criterion) Market niche/position [increasing preferences].

The categories C_h are:

C_1: very high risk (worst category);
C_2: high risk;
C_3: medium risk;
C_4: low risk;
C_5: very low risk (best category).

The parameter values are set as shown in Tables 1 and 2 represents the list of alternatives.

Table 1
Parameter values in the dataset

	1	2	3	4	5	6	7
$g_j(b_1)$	−10.0	−60.0	90.0	28.0	40.0	1.0	0.0
$q_j(b_1)$	1.0	4.0	1.0	1.0	3.0	0.0	0.0
$p_j(b_1)$	2.0	6.0	3.0	2.0	0.0	0.0	0.0
$g_j(b_2)$	0.0	−40.0	75.0	23.0	32.0	2.0	2.0
$q_j(b_2)$	1.0	4.0	1.0	1.0	3.0	0.0	0.0
$p_j(b_2)$	2.0	6.0	3.0	2.0	0.0	0.0	0.0
$g_j(b_3)$	8.0	−20.0	60.0	18.0	22.0	4.0	3.0
$q_j(b_3)$	1.0	4.0	1.0	1.0	3.0	0.0	0.0
$p_j(b_3)$	2.0	6.0	3.0	2.0	0.0	0.0	0.0
$g_j(b_4)$	25.0	30.0	35.0	10.0	14.0	5.0	4.0
$q_j(b_4)$	1.0	4.0	1.0	1.0	3.0	0.0	0.0
$p_j(b_4)$	2.0	6.0	3.0	2.0	0.0	0.0	0.0

Table 2
List of alternatives

	g_1	g_2	g_3	g_4	g_5	g_6	g_7
a_1	16.4	14.5	59.8	7.5	5.2	5	3
a_2	35.8	67.0	64.9	2.1	4.5	5	4
a_3	20.6	61.7	75.7	3.6	8.0	5	3
a_4	11.5	17.1	57.1	4.2	3.7	5	2
a_5	22.4	25.1	49.8	5.0	7.9	5	3
a_6	23.9	34.5	48.9	2.5	8.0	5	3
a_7	29.9	44.0	57.8	1.7	2.5	5	4
a_8	8.7	5.4	27.4	4.5	4.5	5	2
a_9	25.7	29.7	46.8	4.6	3.7	4	2
a_{10}	21.2	24.6	64.8	3.6	8.0	4	2
a_{11}	18.3	31.6	69.3	2.8	3.0	4	3
a_{12}	20.7	19.3	19.7	2.2	4.0	4	2
a_{13}	9.9	3.5	53.1	8.5	5.3	4	2
a_{14}	10.4	9.3	80.9	1.4	4.1	4	2
a_{15}	17.7	19.8	52.8	7.9	6.1	4	4
a_{16}	14.8	15.9	27.9	5.4	1.8	4	2
a_{17}	16.0	14.7	53.5	6.8	3.8	4	4
a_{18}	11.7	10.0	42.1	12.2	4.3	5	2
a_{19}	11.0	4.2	60.8	6.2	4.8	4	2
a_{20}	15.5	8.5	56.2	5.5	1.8	4	2
a_{21}	13.2	9.1	74.1	6.4	5.0	2	2
a_{22}	9.1	4.1	44.8	3.3	10.4	3	4
a_{23}	12.9	1.9	65.0	14.0	7.5	4	3
a_{24}	5.9	−27.7	77.4	16.6	12.7	3	2
a_{25}	16.9	12.4	60.1	5.6	5.6	3	2
a_{26}	16.7	13.1	73.5	11.9	4.1	2	2
a_{27}	14.6	9.7	59.5	6.7	5.6	2	2
a_{28}	5.1	4.9	28.9	2.5	46.0	2	2
a_{29}	24.4	22.3	32.8	3.3	5.0	3	4
a_{30}	29.5	8.6	41.8	5.2	6.4	2	3
a_{31}	7.3	−64.5	67.5	30.1	8.7	3	3
a_{32}	23.7	31.9	63.6	12.1	10.2	3	2
a_{33}	18.9	13.5	74.5	12.0	8.4	3	3
a_{34}	13.9	3.3	78.7	14.7	10.1	2	2
a_{35}	−13.3	−31.1	63.0	21.2	29.1	2	1
a_{36}	6.2	−3.2	46.1	4.8	10.5	2	1
a_{37}	4.8	−3.3	71.1	8.6	11.6	2	2
a_{38}	0.1	−9.6	42.5	12.9	12.4	1	1
a_{39}	13.6	9.1	76.0	17.1	10.3	1	1

References

[1] J.W. Chinneck, MINOS (IIS): Infeasibility using MINOS, Computers and Operations Research 21 (1) (1994) 1–9.
[2] J.W. Chinneck, An effective polynomial-time heuristic for the minimum-cardinality IIS set-covering problem, Annals of Mathematics and Artificial Intelligence 17 (1996) 127–144.
[3] J.W. Chinneck, Feasibility and viability, in: T. Gal, H.J. Greenberg (Eds.), Advances in Sensitivity Analysis and Parametric Programming, Kluwer Academic Publishers, Dordrecht, 1997, pp. 14–22.
[4] L.C. Dias, J.N. Clímaco, ELECTRE TRI for groups with imprecise information on parameter values, Group Decision and Negotiation 9 (5) (2000) 355–377.
[5] L.C. Dias, V. Mousseau, J. Figueira, J. Clímaco, An aggregation/disaggregation approach to obtain robust conclusions with ELECTRE TRI, European Journal of Operational Research 138 (2) (2002) 332–348.

[6] M. Doumpos, C. Hurson, A multicriteria decision aid method for the assessment of business failure risk, Foundations of Computing and Decision Sciences 20 (2) (1995) 99–112.
[7] E. Jacquet-Lagrèze, J. Siskos, Assessing a set of additive utility functions for multicriteria decision-making, the UTA method, European Journal of Operational Research 10 (2) (1982) 151–164.
[8] R.D. Kampke, Sensitivity analysis for assessing preferentially independent order relations, Computers and Operations Research 23 (12) (1996) 1119–1130.
[9] S.H. Kim, B.S. Ahn, Interactive group decision making procedure under incomplete information, European Journal of Operational Research 116 (3) (1999) 498–507.
[10] L.N. Kiss, J.-M. Martel, R. Nadeau, ELECCALC – An interactive software for modelling the decision maker's preferences, Decision Support Systems 12 (4–5) (1994) 757–777.
[11] J. Loon, Irreducibly inconsistent systems of linear inequalities, European Journal of Operational Research 8 (1981) 283–288.
[12] V. Mousseau, J. Figueira, J.-Ph. Naux, Using assignment examples to infer weights for ELECTRE TRI method: Some experimental results, European Journal of Operational Research 130 (2) (2001) 263–275.
[13] V. Mousseau, R. Slowinski, Inferring an ELECTRE TRI model from assignment examples, Journal of Global Optimization 12 (2) (1998) 157–174.
[14] K. Murty, S. Kabadi, R. Chandrasekaran, Infeasibility analysis for linear systems, a survey, Working Paper, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, 2000, 21p.
[15] K.G. Murty, Operations Research: Deterministic Optimization Models, Prentice-Hall, Englewood Cliffs, NJ, 1995.
[16] B. Roy, Main sources of inaccurate determination, uncertainty and imprecision in decision models, Mathematical and Computer Modelling 12 (10–11) (1989) 1245–1254.
[17] B. Roy, A missing link in OR-DA: Robustness analysis, Foundations of Computing and Decision Sciences 23 (3) (1998) 141–160.
[18] B. Roy, D. Bouyssou, Aide multicritére á la décision: Methodes et cas, Economica, Paris, 1993.
[19] B. Roy, Ph. Vincke, Relational systems of preferences with one or more pseudo-criteria: Some new concepts and results, Management Science 30 (11) (1984) 1323–1334.
[20] M. Tamiz, S. Mardle, D. Jones, Detecting IIS in infeasible linear programmes using techniques from goal programming, Computers and Operations Research 23 (2) (1996) 113–119.
[21] P. Vincke, Robust solutions and methods in decision aid, Journal of Multi-Criteria Decision Analysis 8 (3) (1999) 181–187.
[22] M. Weber, Decision making with incomplete information, European Journal of Operational Research 28 (1987) 44–57.