Dealing with inconsistent judgments in multiple criteria sorting models

Vincent Mousseau[*], Luís C. Dias[†‡], José Figueira[†‡§]

August 31, 2004

[*]LAMSADE, Université Paris-Dauphine, Place du Maréchal De Lattre de Tassigny, 75775 Paris Cedex 16, France, e-mail: mousseau@lamsade.dauphine.fr

[†]INESC Coimbra, R. Antero de Quental 199, 3000-033 Coimbra, Portugal, e-mail: ldias@inescc.pt

[‡]Faculty of Economics, University of Coimbra, Av. Dias da Silva 165, 3004-512 Coimbra, Portugal, email: figueira@fe.uc.pt

[§]DIMACS, Rutgers University, CoRE Building, 96 Frelinghuysen Road Piscataway NJ 08854-8018 USA

Contents

Abstract

Some decision problems can be formulated as sorting models which consist in assigning alternatives evaluated on several criteria to ordered categories. The implementation of a multiple criteria sorting model requires to set the values of the preference parameters used in the model. Rather than fixing directly the values of these parameters, an usual approach is to infer these values from assignment examples provided by the decision maker (DM), *i.e.*, alternatives for which he/she specifies a required category or interval of acceptable categories.

However, the judgments expressed by DMs through assignment examples can be inconsistent, *i.e.*, may not match the sorting model. In such situations, it is necessary to support the DMs in the resolution of this inconsistency. In this paper, we propose algorithms that calculate different ways to modify the set of assignment examples so that the information can be represented in the sorting model. The modifications considered are either the removal of examples or the relaxation of existing assignments. These algorithms incorporate information about the confidence attached by the DMs to each assignment example. These algorithms aim at finding and ranking the solutions to solve inconsistency that the DMs are most likely to accept.

Keywords: *Multicriteria Decision Aiding, Sorting problem, Inconsistent judgment, Assignment examples, Inconsistency analysis for LP*

Introduction

Many real-world decision problems can be represented by a model stating explicitly the multiple points of view from which alternatives under consideration should be evaluated, through the definition of n_{crit} criterion functions $g_1, g_2, \ldots, g_j, \ldots, g_{n_{crit}}$. Given a set $A = \{a_1, a_2, \ldots, a_i, \ldots, a_{n_{alt}}\}$ of potential alternatives evaluated on the criteria, the analyst conducting the decision aiding study may formulate the problem in different terms. B. Roy [14] distinguishes among three problem statements, *i.e.*, problem formulations (choosing, sorting and ranking) that may guide the analyst in structuring the decision problem (see also [1]). Among these problem statements, a major distinction concerns *relative vs absolute judgments* of alternatives. This distinction refers to the way alternatives are considered and to the type of result expected from the analysis.

In the first case, alternatives are directly compared one to each other and the results are expressed using the comparative notion of "*better*" vs. "*worse*". Choosing (selecting a subset of the best alternatives) or ranking (defining a preference order on A) are typical examples of comparative judgments. The presence (or absence) of an alternative a_i in the set of the best alternatives results from the comparison of a_i to the other alternatives. Similarly, the position of an alternative in the preference order depends on its comparison to the others.

In the second case, each alternative is considered independently from the others in order to determine its intrinsic value by means of comparisons to norms or references; it consists of assigning each alternative to one of the pre-defined categories $C_1, C_2, ..., C_k, ..., C_{n_{cat}}$. The assignment of an alternative a_i results from its intrinsic evaluation on all criteria with respect to the norm defining the categories. Several methods have been proposed to handle multiple criteria sorting problems (MCSP), *e.g.*, Trichotomic Segmentation [9], N-TOMIC [8], ORCLASS [6], ELECTRE TRI [15], PROAFTN [2], UTADIS [16] and a general class of filtering methods [12].

One of the main difficulties that an analyst must face when interacting with a decision maker (DM) in order to build a sorting model is the elicitation of various preference parameters used by the method. Even when these parameters can be interpreted, it is difficult to fix directly their values and to have a clear understanding of the implications of these values in terms of the output of the model. In order to avoid direct elicitation of the parameters, several authors have designed disaggregation procedures which allow to infer parameter's values from holistic judgments (such a disaggregation approach was first introduced in the UTA method [4]). Such procedures have been defined for MCSP *e.g.*, [16] for UTADIS and [11] for ELECTRE TRI.

The holistic judgments required to infer sorting models are called assignment examples and correspond to alternatives (actual or fictitious) for which the DM can express a desired assignment, *e.g.*, "a_i should be assigned to C_3" ($a_i \rightarrow C_3$), or "a_i should be assigned to C_1 or C_2" ($a_i \rightarrow [C_1, C_2]$, *i.e.*, imprecise assignment examples can be considered). In some sorting methods (namely UTADIS and ELECTRE TRI when only the weights of criteria are inferred) such assignment examples define linear constraints on the model parameters.

In order to minimize the differences between the assignments made by the method and the assignments made by the DM, a mathematical program infers the values for these parameters that best restore the DM's judgments. Such a methodology requires from the DM much less cognitive effort than a direct elicitation of parameters (the elicitation of parameters is done indirectly using holistic information given by the DM) and provides a factual justification for the values assigned to the parameters.

Inference procedures are usually not designed as a problem to be solved only once, but rather several times in an interactive learning process, where the DM continuously revises the information they provide as they learn from the results of the inference programs (see [3]). At each iteration, the DM has the opportunity to revise assignment examples. This interactive process stops when the DM is satisfied with the values of the parameters and when the results of the model (*i.e.*, assignment of alternatives to categories) match their view of the decision problem.

During this interactive process, the DM might provide inconsistent judgments, *i.e.*, a set of assignment examples that cannot be satisfied simultaneously by the sorting model. Such inconsistencies can arise for several reasons (cognitive limitations, evolution of preferences during the process, ...). In such a situation, it is not always easy for the DM to identify the reasons for inconsistencies. Moreover, there usually exists more than one way to restore consistency. Hence, the DM need support in inconsistency analysis.

Consider a problem in which a DM has interactively specified assignment examples inducing linear inequalities on the preference parameters. This is namely the case with UTADIS [16] and ELECTRE TRI [3] when only the weights of criteria are inferred. Let $x_1, x_2, \ldots, x_j, \ldots, x_n$ denote the n parameters of the considered sorting model. The assignment examples define a polyhedron of possible values for the parameters, $T = \{x \in \mathbb{R}^n : \sum_{j=1}^{n} \alpha_{ij} x_j \geq \beta_i,\ i = 1, \ldots, m\}$; when an inconsistent set of assignment examples is provided by the DM, this polyhedron is empty. There exist various ways by which the set of assignment examples can be modified so that the polyhedron T becomes non empty.

The problem is then to identify all the "minimal" subsets (minimal in the sense of the inclusion) that resolve inconsistency, *i.e.*, subsets among which the DM must choose in order to make his/her information consistent. In [10] two algorithms are proposed to identify all the minimal subsets S_q, $q = 1, \ldots, Q$ to be deleted (sorted by cardinality) that resolve inconsistency and whose cardinality is lower than (or equal to) `maxcount` (`maxcount` is an input to the algorithms that states the maximum number of solutions to be computed).

In this paper we propose alternative ways to resolve inconsistencies stemming from a set of assignment examples; namely, instead of deleting assignment examples, we consider relaxing them, *i.e.*, enlarging the interval of the possible assignments for an alternative. Moreover, we consider that the DM may provide confidence levels associated with the assignment examples; such information can be exploited to find a way to solve inconsistency that best them.

The paper is organized as follows. Section 1 defines inconsistency relaxation and shows that the algorithms proposed by [10] still apply when considering constraints relaxation rather than constraints deletion. The section 2 considers the case where the DM is able to provide confidence levels associated to the assignment examples. We provide two ways to account for such information in order to rank the solutions according to the confidence levels provided by the DM. Section 3 provides an illustrative example within the context of the ELECTRE TRI.

1 Inconsistency resolution via constraints relaxation

Resolving the inconsistencies can be performed by deleting a subset of constraints. Let us denote $I = \{1, 2, ..., m\}$ the set of indices of the constraints and $T_\emptyset = \{x \in \mathbb{R}^n : \sum_{j=1}^{n} \alpha_{ij} x_j \geq \beta_i, \forall i \in I\}$ the initial empty polyhedron, *i.e.*, with all the constraints. Let $S \subseteq I$ denote a subset of indices of constraints. We will say that S resolves the inconsistency if and only if the polyhedron $T_S = \{\sum_{j=1}^{n} \alpha_{ij} x_j \geq \beta_i, \forall i \in I \setminus S\}$ is not empty.

In [10] two algorithms are proposed to compute alternative ways to restore consistency by constraints deletion. We consider here the case where consistency can be solved by relaxing constraints rather than deleting them.

Considering an infeasible system of linear inequalities (that can correspond to assignment examples), relaxing constraints rather than deleting them (in order to restore feasibility) has already been studied in the general case by (*e.g.*, [13] and [7]). The relaxations considered by these authors are continuous and deal with the right-hand-side of the constraints only. In our case, we will define the relaxations differently:

- the relaxations will be performed by changing the technical coefficients of the constraints rather than the right-hand-side;
- a discrete set of relaxations will be considered which have a meaning in the sorting model, namely increasing the interval of categories to which an alternative can be assigned.

Suppose the DM has specified a set of assignment examples, *i.e.*, a subset of alternatives $A^* \subseteq A$ such that each $a_i \in A^*$ is associated with $max(a_i)$ ($min(a_i)$, respectively) the index of the maximum (minimum, respectively) category to which a_i should be assigned according to their holistic preferences ($a_i \rightarrow [min(a_i), max(a_i)]$, $a_i \in A^*$). From the DM's perspective, $max(a_i)$ represents the statement "a_i *should be assigned at most to category* $C_{max(a_i)}$" and, $min(a_i)$ express that "a_i *should be assigned at least to category* $C_{min(a_i)}$". For each $a_i \in A^*$, $min(a_i)$ and $max(a_i)$ induce (when considering UTADIS and ELECTRE TRI) two linear constraints. Note that trivial constraints such as $min(a_i) = C_{min}$ and/or $max(a_i) = C_{max}$ do not need to be taken into account.

Let us consider the assignment example $a_i \rightarrow [min(a_i), max(a_i)]$, $a_i \in A^*$. A relaxation of this assignment example consists of assigning a_i to a wider interval $[C_k, C_{k'}]$ such that $k \leq min(a_i)$ or $k' \geq max(a_i)$, with at least one strict inequality. Let us consider the system of inequalities containing the constraints corresponding to all the possible relaxations of the assignment example $a_i \rightarrow [min(a_i), max(a_i)]$ (it also contains the constraints corresponding to the original assignment example). It should be noticed that all the constraints corresponding to a relaxation of one of the two initial constraints are redundant. Consider S the set of all indices of constraints induced from a set of assignment examples corresponding to a relaxation of the initial assignment examples. Therefore we can note that S contains many redundancies.

If we apply the algorithms proposed by [10] (*i.e.*, inconsistency resolution via constraints deletion) to the set S, the solutions correspond to constraints relaxation and/or deletion. It follows from the preceding remark that it is possible to use the algorithm proposed by [10] to solve inconsistencies by relaxation (rather than deletion) of assignment examples. Hence, in the rest of the paper, we will talk in terms of constraints deletion, knowing that it embraces the case of constraints relaxation.

2 Attributing confidence levels to assignment examples

In the course of the interactive process that aims at inferring the parameters of a sorting model, the DM provides assignment examples. For each assignment example, DMs might be more or less confident in their statements. Let us suppose that they are able to express confidence judgments during the interactive process. Such confidence judgments can be taken into account when an inconsistency arises. More precisely, algorithms that identify alternative ways for solving inconsistencies may use such information. Intuitively, these algorithms should provide solutions in an order such that the least confident constraints are relaxed/deleted with a higher priority than solutions relaxing more confident statements.

2.1 Defining confidence levels

Let us consider a confidence scale on the assignment examples $\Psi = \{\psi_0, \psi_1, \ldots, \psi_p, \ldots, \psi_\tau\}$, where ψ_0 (ψ_τ, respectively) corresponding to the minimum (maximum, respectively) confidence level, and $\prec$ denote an order on Ψ ($\psi_0 \prec \psi_1 \prec \ldots \prec \psi_p \prec \ldots \prec \psi_\tau$). The semantic of this qualitative scale is such that, when facing an inconsistency situation, an assignment example is less likely to be relaxed/deleted when its confidence level is high.

From the DM's perspective, $a_i \rightarrow max(a_i)$ represents the statements "a_i *should be assigned at most to category* $C_{max(a_i)}$" and, $min(a_i)$ express that "a_i *should be assigned at least to category* $C_{min(a_i)}$". These two statements induce two constraints. The DM can attach a confidence level to each of the above mentioned statements. This information will be interpreted as confidence levels attached to the corresponding constraints (for example, $a_1 \rightarrow C_2$ implies "a_1 should be assigned at least to C_2" and "a_1 should be assigned at most to C_2" and the DM may have different confidence levels concerning these two statements, *e.g.*, they may say that if a_1 is not assigned to C_2, then it is more likely to be assigned to a higher category than to a lower one). For each relaxed constraint, the attached confidence level corresponds to the confidence level of the original constraint from which it was derived (unless the DM provides specific information).

2.2 A lexicographic ranking procedure

Let us consider an inconsistent set of assignment examples provided by the DM and the set of linear constraints associated with these examples. Any relaxation of these assignment examples (see §1) will be also considered here. Following the notation introduced previously, m denotes the total number of constraints and $I = \{1, 2, \ldots, i, \ldots, m\}$ denotes the set of indices of these constraints. The resulting polyhedron is, $T_\emptyset = \{x \in \mathbb{R}^n : \sum_{j=1}^n \alpha_{ij} x_j \geq \beta_i, \forall i \in I\} = \emptyset$

Let I^p denote the subset of constraints whose confidence level is equal to ψ_p. Hence, $I^0, I^1, ..., I^p$, $..., I^\tau$ define a partition of I. Furthermore, we will denote $I^{\leq p} = \bigcup_{l=0}^{p} I^l$ the set of constraints whose confidence level is lower than or equal to ψ_p. Now, consider $S^l \subseteq I^{\leq l}$ a subset of indices of constraints whose confidence level is lower than or equal to ψ_l. We will say that S^l resolves the inconsistency at a confidence level ψ_l if and only if $T_{S^l} = \{x \in \mathbb{R}^n : \sum_{j=1}^n \alpha_{ij} x_j \geq \beta_i, \forall i \in I \setminus S^l\} \neq \emptyset$.

A simple way to account for the confidence level attached to each constraint is to proceed as follows:

1. Identify (by increasing order of cardinality) all minimal subsets $S_1^0, S_2^0, ..., S_{q_0}^0$ that resolve the inconsistency at level ψ_0 (*i.e.*, relaxations whose confidence level is equal to ψ_0 that make the original system of inequalities feasible).

2. Then, identify (by increasing order of cardinality) all minimal subsets $S_1^1, S_2^1, ..., S_{q_1}^1$ that resolve the inconsistency at level ψ_1.

3. Proceed in the same way until finding minimal subsets $S_1^\tau, S_2^\tau, ..., S_{q_\tau}^\tau$ that resolve the inconsistency at level ψ_τ or finding a total number of subsets equal to `maxcount`.

The program PM_1^0 identifies the smallest set of constraints S_1^0 whose confidence level is equal to ψ_0 so that $T_{S_1^0} = \{x \in \mathbb{R}^n : \sum_{j=1}^n \alpha_{ij} x_j \geq \beta_i, \forall i \in I \setminus S_1^0\} \neq \emptyset$

$$PM_1^0 \begin{cases} Min & \sum_{I^{\leq 0}} y_i \\ \text{s.t.} & \sum_{j=1}^{n} \alpha_{ij} x_j + M y_i \geq \beta_i, \ \ \forall i \in I^{\leq 0} \\ & \sum_{j=1}^{n} \alpha_{ij} x_j \geq \beta_i, \ \ \forall i \in I \setminus I^{\leq 0} \\ & x_j \geq 0, \ \ j = 1, \ldots, n \\ & y_i \in \{0,1\}, \ \ i \in I^{\leq 0} \end{cases} \quad (1)$$

where, M is a positive large number; the variables $y_i, i \in I^{\leq 0}$, are the binary variables assigned to each constraint index in S whose confidence index is lower than or equal to ψ_0. The indices of constraints for which $y_i^* = 1$ (at the optimum of PM_1^0) constitute the subset S_1^0.

PM_2^0 is defined in order to compute S_2^0. This new program is derived from PM_1^0 by adding the single constraint $\sum_{i \in S_1^0} y_i \leq |S_1^0| - 1$. This constraint makes it impossible to find S_1^0 (the optimal solution of PM_1^0) or any solution that includes this set. A third program PM_3^0 is then defined by adding the constraint $\sum_{i \in S_2^0} y_i \leq |S_2^0| - 1$, and so on until we reach an infeasible program, meaning that there are no more solutions in $I^{\leq 0}$.

When all the solutions in $I^{\leq 0}$ are found, the algorithm starts to search for solutions in $I^{\leq 1}$. The first solution S_1^1 is found by solving PM_1^1 which is derived from the previous program by replacing the constraints $\sum_{j=1}^{n} \alpha_{ij} x_j + M y_i \geq \beta_i, \ \ \forall i \in I^{\leq 0}$ and $\sum_{j=1}^{n} \alpha_{ij} x_j \geq \beta_i, \ \ \forall i \in I \setminus I^{\leq 0}$ by $\sum_{j=1}^{n} \alpha_{ij} x_j + M y_i \geq \beta_i, \ \ \forall i \in I^{\leq 1}$ and $\sum_{j=1}^{n} \alpha_{ij} x_j \geq \beta_i, \ \ \forall i \in I \setminus I^{\leq 1}$. The algorithm continues until finding `maxcount` solutions (or no more solution exists).

```
Begin
     p ← 0
     count ← 0
     While (p ≤ τ) and (count ≤ maxcount)
          q ← 1
          moresol ← true
          While moresol
               Solve PM_q^p
               If (PM_q^p has no solution) or (count>maxcount)
               Then
                    moresol ← false
               Else
                    S_q^p ← {i ∈ I : y_i^* = 1}
                    Add constraint Σ_{i∈S_q^p} y_i ≤ |S_k^l| − 1 to PM_q^p so as to define PM_{q+1}^p
                    q ←q+1
                    count ←count+1
               End if
          End while
          p ←p+1
     End while
End
```

This algorithm requires to solve several 0-1 programs. Note that it is possible to design an algorithm for the same purpose using only linear programming (see the second algorithm presented in [10]).

2.3 A penalty based ranking procedure

2.3.1 Defining a penalty function

Another approach to our problem consists of defining a penalty function $\pi(S)$ associated to each subset of constraints indices $S \subseteq I$, and to rank the subsets that resolve the inconsistency by decreasing penalty order: the larger $\pi(S)$, the greater the insatisfaction of the DM in removing S from I. This approach generalizes the lexicographic ranking as it is possible to define the penalty function π in a way that the penalty ranking coincides with the lexicographic ranking.

Given a subset $S \subseteq I$, $S \cap I^p$ denotes the subset of S corresponding to constraints indices whose confidence level is equal to ψ_p, $p = 0, ..., \tau$. Let $|S \cap I^p|$ denote the cardinality of S whose confidence level is equal to ψ_p.

In order to define the semantic of the penalty function π, we impose a few suitable conditions on π:

Condition 2.1. *(non-negativity)*
$\forall S \subseteq I,\ \pi(S) \geq \pi(\emptyset) = 0.$

Condition 2.2. *(anonymity)*
$\forall S, S' \subseteq I,$ *if* $|S \cap I^p| = |S' \cap I^p|,\ \forall p = 0, ..., \tau$ *then* $\pi(S) = \pi(S').$

Condition 2.3. *(confidence monotonicity)*
$\forall S, S' \subseteq I$ *such that* $|S \cap I^p| = |S' \cap I^p|,\ p = 1, \ldots, \tau, p \neq u, p \neq v,$ *it holds:*

$$\left.\begin{array}{l} |S' \cap I^u| = |S \cap I^u| + 1 \\ |S' \cap I^v| = |S \cap I^v| - 1 \\ u < v \end{array}\right\} \Rightarrow\ \pi(S) > \pi(S')$$

Condition 2.4. *(cardinality monotonicity)*
$\forall S, S' \subseteq I,$ *if* $\forall p = 0, ..., \tau,\ |S \cap I^p| \geq |S' \cap I^p|$ *then* $\pi(S) \geq \pi(S').$

These conditions express natural properties for a function π to define a consistent penalty function:

- Condition 2.1 (non-negativity) states that π has a lower bound (arbitrarily set to 0). Although it is not necessary, it would be possible to impose also an upper bound on the penalties, for instance $\pi(I) = 1$. In such a case, the penalty function π could be understood as a "disutility" function related to a utility function $u(.) = 1 - \pi(.)$. This would be of interest in that the questioning techniques used to elicit multi-attribute utility functions [5] from the DM can be used in this context.

- Condition 2.2 (anonymity) states that the penalty of a set S only depends on the number of constraints of each confidence level contained in S regardless of the constraints "label".

- Condition 2.3 (confidence monotonicity) states that considering a solution S, if the confidence level of one constraint in S decreases, then $\pi(S)$ should also decrease.

- Condition 2.4 (cardinality monotonicity) states that if a solution S contains less (or equal) constraints than another solution S', for each confidence level, the penalty should be lower (or equal) for S than for S'.

Among the possible penalty functions, one of the simplest can be defined considering that each constraint in S of a given confidence level ψ_p contributes to increase $\pi(S)$ by an amount Δ_p (the values Δ_p are to be defined by the DM):

$$\pi(S) = \sum_{p=0}^{\tau} \Delta_p \, |S \cap I^p| \tag{2}$$

Considering condition 2.3, the amounts for $\Delta_0, ..., \Delta_\tau$ should be such that $u < v \Rightarrow \Delta_u < \Delta_v$. This simple model can be generalized by considering that the penalty does not increase linearly with respect to the number of constraints:

$$\pi(S) = \sum_{p=0}^{\tau} \pi_p(|S \cap I^p|) \tag{3}$$

where $\pi_p(n)$ is a function (to be defined by the DM) denoting the penalty of removing n constraints of confidence level ψ_p (given the assumed conditions, π_p must be an increasing function).

More sophisticated non-additive models may be envisaged, namely those taking into account preference dependencies among different confidence levels.

2.3.2 Ranking solutions according to the penalty function

Given a penalty function π, it is necessary to define an algorithm to rank by increasing penalty the subsets of I that, if removed, yield a consistent system. In order to design an algorithm to identify the `maxcount` subsets of constraints that solve inconsistency and rank them according to the penalty function π, we will adapt the algorithms presented by [10] which rank by increasing cardinality all minimal subsets that resolve inconsistency : $S_1, S_2, \ldots, S_q, \ldots$ ($|S_q| \leq |S_{q+1}|$).

The algorithms presented in [10] provide the set of the solutions ordered by cardinality without any consideration about confidence levels. In our case, we want to provide the set of (at most) `maxcount` solutions ordered by increasing penalty. It should be noticed that the smallest cardinality solutions might not correspond to those of the smallest penalty. Therefore, we can proceed by computing the solutions by increasing cardinality and stop when we are sure that the solutions of a higher cardinality have a greater penalty than the ones we already obtained.

Let $S^{x,p}$ denote an arbitrary set of x constraints, all of confidence levels equal to ψ_p.

Proposition 2.1. $\forall S \subseteq I,\ S' \subseteq I :\ |S| \geq |S'|$, *it holds* $\pi(S) \geq \pi(S^{|S'|,0})$ *i.e., the penalty of any solution after the q-th is not lower than the penalty that would be awarded to the q-th solution if all the constraints indexed by* S_q *were of the lowest confidence.*

Proof. From repeatedly using Condition 2.3, $\pi(S) \geq \pi(S^{|S|,0})$. From Condition 2.4, since $|S| \geq |S'|$, it holds that $\pi(S^{|S|,0}) \geq \pi(S^{|S'|,0})$. □

In the following algorithm, `TOP-N` denotes a list of solutions of at most `maxcount` elements, ordered by increasing penalty, and S_{tail} denotes the last solution (*i.e.*, the solution with the highest penalty) of list `TOP-N`. At the end of the algorithm, `TOP-N` contains the `maxcount` first solutions ordered by increasing penalty.

```
Begin
     q  ← 0
     TOP-N  ← empty list
     S_tail  ← first solution
     repeat
          S_q ← q^th solution provided by the algorithm in [10]
          If (q ≤ maxcount) or (π(S_q) < π(S_tail))
               Then S_q enters TOP-N
          End If
     Until S_q = ∅ or (q ≥ maxcount and π(S_tail) ≤ π(S^{|S_q|,0}))
End
```

In this algorithm, when S_q enters `TOP-N` it does so respecting the penalty ranking. If the list is full (when it contains `maxcount` elements), this implies removing the highest penalty element (S_{tail}), and the variable S_{tail} will be updated. Proposition 2.1 allows us to define the stopping condition $\pi(S_{tail}) \leq \pi(S^{|S_q|,0}))$.

3 Illustrative example

Let us consider a situation in which a set of 40 alternatives has to be assigned to 5 categories using the ELECTRE TRI pessimistic method. Each alternative is evaluated on the basis of a set of 7 criteria (see Appendix A). The limits of categories are known but the criteria importance coefficients are to be defined (see Appendix B). Suppose the DM provides assignment examples with associated level of confidence on a scale (absolutely confident $\succ$ quite confident $\succ$ not so confident), where $a_i \rightarrow [C_k, C_{k'}]$ means that alternative a_i must be assigned to a category between C_k and $C_{k'}$, $(k \leq k')$:

- $a_1 \rightarrow C_5$, not so confident
- $a_{18} \rightarrow C_4$, quite confident
- $a_{23} \rightarrow [C_2, C_3]$, not so confident
- $a_{24} \rightarrow [C_2, C_3]$, quite confident
- $a_{26} \rightarrow C_5$, quite confident
- $a_{30} \rightarrow C_1$, not so confident
- $a_{31} \rightarrow C_5$, not so confident
- $a_{35} \rightarrow [C_1, C_2]$, absolutely confident
- $a_{36} \rightarrow C_4$, quite confident
- $a_{38} \rightarrow C_4$, not so confident
- $a_{39} \rightarrow C_3$, not so confident

From these assignment examples, it is possible to define relaxations as defined in Section 1. In this example, we will suppose that the relaxations have the same confidence level than their corresponding assignment examples. For instance, the relaxations corresponding to the assignment example $a_1 \rightarrow C_5$ are $a_1 \rightarrow [C_4, C_5]$, $a_1 \rightarrow [C_3, C_5]$ and $a_1 \rightarrow [C_2, C_5]$ (note that the relaxation $a_1 \rightarrow [C_1, C_5]$ amounts at removing the assignment example $a_1 \rightarrow C_5$).

These assignment examples and their relaxations generate a set of 41 constraints on the criteria weights w_j, $j = 1, ..., 7$ and cutting level λ which are presented in Appendix C. The first 17 constraints correspond to the original assignment examples and the remaining ones correspond to the

relaxations. The linear system associated with the assignment examples is infeasible, which means that the information provided by the DM is inconsistent, *i.e.* there is no way to represent this information in the ELECTRE TRI sorting model.

Considering this infeasible linear system, there exist 11 minimal subsets of constraints that resolve the inconsistency, where $I = \{1, 2, ..., 41\}$. These 11 subsets (ordered by cardinality) are listed below. Let us remark that due to the limited size of this example, we have computed all the solutions. Such a way of proceeding is time consuming when dealing with real world problems of large size .

- $S_1 = \{5, 8, 9, 10, 11, 25, 28, 29\}$
- $S_2 = \{5, 8, 9, 10, 11, 17, 28, 29\}$
- $S_3 = \{1, 5, 8, 9, 10, 14, 17, 28, 29\}$
- $S_4 = \{1, 5, 8, 9, 10, 14, 25, 28, 29\}$
- $S_5 = \{1, 8, 9, 10, 14, 25, 28, 29, 31\}$
- $S_6 = \{1, 8, 9, 10, 11, 25, 28, 29, 31\}$
- $S_7 = \{5, 7, 9, 10, 11, 13, 17, 28, 29, 30\}$
- $S_8 = \{1, 8, 9, 10, 12, 14, 16, 25, 28, 31, 38\}$
- $S_9 = \{5, 8, 9, 10, 11, 12, 14, 16, 25, 28, 31, 38\}$
- $S_{10} = \{3, 5, 7, 9, 11, 13, 15, 17, 23, 24, 28, 29, 30, 41\}$
- $S_{11} = \{1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 19, 20, 21, 22, 25, 26, 27, 31, 32, 33, 36, 37, 38, 39, 40\}$

To solve the inconsistency, the DM should choose one among these 11 alternative solutions. In order to be presented to the DM, these solutions should be formulated in terms of relaxation of the assignment examples. For instance, S_1 corresponds to (the formulation of the other solutions are provided in Appendix D):

$$\begin{cases} \text{relax } a_{23} \to [C_2, C_3] & \text{to } a_{23} \to [C_2, C_4] \\ \text{relax } a_{26} \to C_5 & \text{to } a_{26} \to [C_3, C_5] \\ \text{relax } a_{30} \to C_1 & \text{to } a_{30} \to [C_1, C_4] \\ \text{relax } a_{31} \to C_5 & \text{to } a_{31} \to [C_4, C_5] \\ \text{relax } a_{35} \to [C_1, C_2] & \text{to } a_{35} \to [C_1, C_3] \end{cases} \tag{4}$$

We can observe that some of these relaxations correspond to the deletion of one constraint from I (relax $a_{23} \to [C_2, C_3]$ to $a_{23} \to [C_2, C_4]$, constraint 5), while others require the deletion of several constraints from I (relax $a_{26} \to C_5$ to $a_{26} \to [C_3, C_5]$, constraints 8 and 25). Moreover, S_1 suggests to relax assignment examples whose associated confidence levels are "*not so confident*" (a_{23}, a_{30} and a_{31}), "*quite confident*" (a_{26}) or "*absolutely confident*" (a_{35}).

When interacting with the DM to solve an inconsistency, it is not reasonable to propose him/her a large number of alternative solutions. It is convenient to propose a limited number of solutions that might be interesting for him/her. In our case, we wish to propose approximately 5 solutions to the DM (`maxcount`=5).

If we consider the confidence judgments, the lexicographic ranking algorithm presented in §2.2 computes the solutions, considering first the solutions that removes constraints which are "*not so confident*" (*i.e.*, $I^{\leq 0}$), then the solutions removing constraints that are "*not so confident*" or "*quite confident*" (*i.e.*, $I^{\leq 1}$) and finally the remaining ones. In each group, the solutions are computed by increasing order of cardinality. In our example, no solution exists removing only "*not so confident*" constraints; the first five solutions computed only involve the deletion of constraints that are "*not so confident*" or "*quite confident*":

- $S_3 = \{1, 5, 8, 9, 10, 14, 17, 28, 29\}$
- $S_4 = \{1, 5, 8, 9, 10, 14, 25, 28, 29\}$
- $S_5 = \{1, 8, 9, 10, 14, 25, 28, 29, 31\}$
- $S_7 = \{5, 7, 9, 10, 11, 13, 17, 28, 29, 30\}$
- $S_8 = \{1, 8, 9, 10, 12, 14, 16, 25, 28, 31, 38\}$

If the DM provides a penalty function $\pi(.)$, then the penalty based procedure presented in §2.3 may also be used. For instance, let us consider a penalty function as in (2), with $\Delta_0 = 1$, $\Delta_1 = 2$ and $\Delta_2 = 3$. The five best solutions are:

- $S_5 = \{1, 8, 9, 10, 14, 25, 28, 29, 31\}$, $\pi(S_5) = 10$
- $S_1 = \{5, 8, 9, 10, 11, 25, 28, 29\}$, $\pi(S_1) = 11$
- $S_3 = \{1, 5, 8, 9, 10, 14, 17, 28, 29\}$, $\pi(S_3) = 11$
- $S_4 = \{1, 5, 8, 9, 10, 14, 25, 28, 29\}$, $\pi(S_4) = 11$
- $S_7 = \{5, 7, 9, 10, 11, 13, 17, 28, 29, 30\}$, $\pi(S_7) = 11$

Using either of these rankings of solutions, the DM is to chose among the "most promising" solutions to solve inconsistency. Hence, this largely reduces the cognitive effort of the DM to resolve inconsistency.

Conclusion

In this paper, we considered the problem of supporting the DM in the resolution of inconsistent judgments expressed in the form of assignment examples in multiple criteria sorting model. We have proposed the concept of relaxation of an assignment example, which is helpful in this context. To resolve the inconsistency, it is useful to obtain from the DM confidence statements associated with the assignment examples. We have proposed procedures that account for this information to assist the DM in finding the most relevant ways to restore consistency.

An illustrative example has been provided to show how the proposed procedures can be used within the context of the ELECTRE TRI sorting method. However, our procedures are general and apply to any sorting method for which assignment examples generate linear constraints on the preference-related parameters.

An interesting extension of this work consists in considering the possibility of associating different confidence levels to the original assignment examples constraints and their corresponding relaxation. This extension amounts at considering that the various relaxations of an assignment example are not judged as equivalent as regards their confidence levels.

Acknowledgements: This work has benefited from the luso-french grant n^{o}500B4 (ICCTI / Ambassade de France au Portugal) and program PESSOA 2004 (GRICES/EGIDE). The third author was also supported by the grant SFRH/BDP/6800/2001.

References

[1] C.A. Bana e Costa. Les problematiques de l'aide a la decision: vers l'enrichissement de la trilogie choix-tri-rangement. *RAIRO/ Recherche Operationnelle*, 30(2):191–216, 1996.

[2] N. Belacel. Multicriteria assignment method PROAFTN: methodology and medical application. *European Journal of Operational Research*, 125(1):175–183, August 2000.

[3] L.C. Dias, V. Mousseau, J. Figueira, and J.N. Clímaco. An aggregation/disaggregation approach to obtain robust conclusions with ELECTRE TRI. *European Journal of Operational Research*, 138(2):332–348, April 2002.

[4] E. Jacquet-Lagrèze and Y. Siskos. Assessing a set of additive utility functions for multicriteria decision making: the UTA method. *European Journal of Operational Research*, 10:151–164, 1982.

[5] R.L. Keeney and H. Raiffa. *Decisions with multiple objectives: Preferences and value tradeoffs*. J. Wiley, New York, 1976.

[6] O.I. Larichev and H.M. Moskovich. An approach to ordinal classification problems. *International Transactions in Operational Research*, 1(3):375–385, 1994.

[7] T. León and V. Liern. A fuzzy method to repair infeasibility in linearly constrained problems. *Fuzzy Sets and Systems*, 122(2):237–243, September 2001.

[8] R. Massaglia and A. Ostanello. N-tomic: a support system for multicriteria segmentation problems. In P. Korhonen, A. Lewandowski, and J. Wallenius, editors, *Multiple Criteria Decision Support*, pages 167–174. Springer Verlag, LNEMS 356, Berlin, 1991.

[9] J. Moscarola and B. Roy. Procédure automatique d'examen de dossiers fondée sur une segmentation trichotomique en présence de critères multiples. *RAIRO Recherche Opérationnelle*, 11(2):145–173, 1977.

[10] V. Mousseau, L.C. Dias, J. Figueira, C. Gomes, and J.N. Clímaco. Resolving inconsistencies among constraints on the parameters of an MCDA model. *European Journal of Operational Research*, 147(1):72–93, May 2003.

[11] V. Mousseau and R. Slowinski. Inferring an ELECTRE TRI model from assignment examples. *Journal of Global Optimization*, 12(2):157–174, 1998.

[12] P. Perny. Multicriteria filtering methods based on concordance/non-discordance principles. *Annals of Operations Research*, 80:137–167, 1998.

[13] G.M. Roodman. Post-infeasibility analysis in linear programming. *Management Science*, 25(9):916–922, September 1979.

[14] B. Roy. *Multicriteria Methodology for Decision Aiding*. Kluwer Academic, Dordrecht, 1996.

[15] B. Roy and D. Bouyssou. *Aide Multicritère à la Décision: Méthodes et Cas*. Economica, Paris, 1993.

[16] C. Zopounidis and M. Doumpos. Multicriteria preference disaggregation for classification problems with an application to global investing risk. *Decision Sciences*, 32(2):333–385, 2001.

Appendices

Appendix A: Evaluation matrix

	$g_1(a_i)$	$g_2(a_i)$	$g_3(a_i)$	$g_4(a_i)$	$g_5(a_i)$	$g_6(a_i)$	$g_7(a_i)$
a_0	35.8	67	19.7	0	0	5	4
a_1	16.4	14.5	59.8	7.5	5.2	5	3
a_2	35.8	24	64.9	2.1	4.5	5	4
a_3	20.6	61.7	75.7	3.6	8	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	7.9	5	3
a_6	23.9	34.5	48.9	2.5	8	5	3
a_7	29.9	44	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	4	2
a_{11}	18.3	31.6	69.3	2.8	3	4	3
a_{12}	20.7	19.3	19.7	2.2	4	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	14.7	53.5	6.8	3.8	4	4
a_{18}	11.7	10	42.1	12.2	4.3	5	2
a_{19}	11	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	2	2
a_{22}	9.1	4.1	44.8	3.3	10.4	3	4
a_{23}	12.9	1.9	65	14	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	2	2
a_{29}	24.4	22.3	32.8	3.3	5	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	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	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	17.1	10.3	1	1

Appendix B: Fixed parameters

	g_1	g_2	g_3	g_4	g_5	g_6	g_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	0.0	0.0	0.0
$p_j(b_1)$	2.0	6.0	3.0	2.0	3.0	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	0.0	0.0	0.0
$p_j(b_2)$	2.0	6.0	3.0	2.0	3.0	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	0.0	0.0	0.0
$p_j(b_3)$	2.0	6.0	3.0	2.0	3.0	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	0.0	0.0	0.0
$p_j(b_4)$	2.0	6.0	3.0	2.0	3.0	0.0	0.0

Appendix C: Constraints stemming from the assignment examples

1. $C(a_1) \geq 5 \Leftrightarrow -\lambda + w_4 + w_5 + w_6 \geq 0$
2. $C(a_{18}) \geq 4 \Leftrightarrow -\lambda + w_1 + w_2 + w_3 + w_4 + w_5 + w_6 \geq 0$
3. $C(a_{18}) \leq 4 \Leftrightarrow \lambda - w_5 - w_6 \geq \varepsilon$
4. $C(a_{23}) \geq 2 \Leftrightarrow -\lambda + w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 \geq 0$
5. $C(a_{23}) \leq 3 \Leftrightarrow \lambda - w_1 - w_2 - w_4 - w_5 - w_6 - w_7 \geq \varepsilon$
6. $C(a_{24}) \geq 2 \Leftrightarrow -\lambda + w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 \geq 0$
7. $C(a_{24}) \leq 3 \Leftrightarrow \lambda - w_4 - w_5 \geq \varepsilon$
8. $C(a_{26}) \geq 5 \Leftrightarrow -\lambda + 0.1w_4 + w_5 \geq 0$
9. $C(a_{30}) \leq 1 \Leftrightarrow \lambda - w_1 - w_2 - w_3 - w_4 - w_5 - w_6 - w_7 \geq \varepsilon$
10. $C(a_{31}) \geq 5 \Leftrightarrow -\lambda + w_5 \geq 0$
11. $C(a_{35}) \leq 2 \Leftrightarrow \lambda - w_2 - w_3 - w_4 - w_5 - w_6 \geq \varepsilon$
12. $C(a_{36}) \geq 4 \Leftrightarrow -\lambda + 0.2w_1 + w_2 + w_3 + w_4 + w_5 \geq 0$
13. $C(a_{36}) \leq 4 \Leftrightarrow \lambda - w_4 - w_5 \geq \varepsilon$
14. $C(a_{38}) \geq 4 \Leftrightarrow -\lambda + w_2 + w_3 + w_4 + w_5 \geq 0$
15. $C(a_{38}) \leq 4 \Leftrightarrow \lambda - w_5 \geq \varepsilon$
16. $C(a_{39}) \geq 3 \Leftrightarrow -\lambda + w_1 + w_2 + w_3 + w_4 + w_5 \geq 0$
17. $C(a_{39}) \leq 3 \Leftrightarrow \lambda - w_1 - w_2 - w_4 - w_5 \geq \varepsilon$
18. $C(a_1) \geq 4 \Leftrightarrow -\lambda + w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 \geq 0$
19. $C(a_1) \geq 3 \Leftrightarrow -\lambda + w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 \geq 0$
20. $C(a_1) \geq 2 \Leftrightarrow -\lambda + w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 \geq 0$
21. $C(a_{18}) \geq 3 \Leftrightarrow -\lambda + w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 \geq 0$
22. $C(a_{18}) \geq 2 \Leftrightarrow -\lambda + w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 \geq 0$
23. $C(a_{23}) \leq 4 \Leftrightarrow \lambda - w_5 \geq \varepsilon$
24. $C(a_{24}) \leq 4 \Leftrightarrow \lambda - w_5 \geq \varepsilon$
25. $C(a_{26}) \geq 4 \Leftrightarrow -\lambda + w_1 + w_2 + w_4 + w_5 \geq 0$
26. $C(a_{26}) \geq 3 \Leftrightarrow -\lambda + w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 \geq 0$
27. $C(a_{26}) \geq 2 \Leftrightarrow -\lambda + w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 \geq 0$
28. $C(a_{30}) \leq 2 \Leftrightarrow \lambda - w_1 - w_2 - w_3 - w_4 - w_5 - w_6 - w_7 \geq \varepsilon$
29. $C(a_{30}) \leq 3 \Leftrightarrow \lambda - w_1 - w_2 - w_3 - w_4 - w_5 - w_7 \geq \varepsilon$
30. $C(a_{30}) \leq 4 \Leftrightarrow \lambda - w_1 - w_4 - w_5 \geq \varepsilon$
31. $C(a_{31}) \geq 4 \Leftrightarrow -\lambda + w_1 + w_5 + w_7 \geq 0$
32. $C(a_{31}) \geq 3 \Leftrightarrow -\lambda + w_1 + w_3 + w_5 + w_6 + w_7 \geq 0$
33. $C(a_{31}) \geq 2 \Leftrightarrow -\lambda + w_1 + 10.75w_2 + w_3 + w_5 + w_6 + w_7 \geq 0$
34. $C(a_{35}) \leq 3 \Leftrightarrow \lambda \geq \varepsilon$
35. $C(a_{35}) \leq 4 \Leftrightarrow \lambda \geq \varepsilon$
36. $C(a_{36}) \geq 3 \Leftrightarrow -\lambda + w_1 + w_2 + w_3 + w_4 + w_5 + w_6 \geq 0$
37. $C(a_{36}) \geq 2 \Leftrightarrow -\lambda + w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 \geq 0$
38. $C(a_{38}) \geq 3 \Leftrightarrow -\lambda + w_1 + w_2 + w_3 + w_4 + w_5 \geq 0$
39. $C(a_{38}) \geq 2 \Leftrightarrow -\lambda + w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 \geq 0$
40. $C(a_{39}) \geq 2 \Leftrightarrow -\lambda + w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 \geq 0$
41. $C(a_{39}) \leq 4 \Leftrightarrow \lambda - w_5 \geq \varepsilon$

Appendix D: Solutions of the inconsistency problem

In the example provided in section 3, there exist 11 solutions to solve the inconsistency, *i.e.*, *

- $S_1 = \{5, 8, 9, 10, 11, 25, 28, 29\}$
- $S_2 = \{5, 8, 9, 10, 11, 17, 28, 29\}$
- $S_3 = \{1, 5, 8, 9, 10, 14, 17, 28, 29\}$
- $S_4 = \{1, 5, 8, 9, 10, 14, 25, 28, 29\}$
- $S_5 = \{1, 8, 9, 10, 14, 25, 28, 29, 31\}$
- $S_6 = \{1, 8, 9, 10, 11, 25, 28, 29, 31\}$
- $S_7 = \{5, 7, 9, 10, 11, 13, 17, 28, 29, 30\}$
- $S_8 = \{1, 8, 9, 10, 12, 14, 16, 25, 28, 31, 38\}$
- $S_9 = \{5, 8, 9, 10, 11, 12, 14, 16, 25, 28, 31, 38\}$
- $S_{10} = \{3, 5, 7, 9, 11, 13, 15, 17, 23, 24, 28, 29, 30, 41\}$
- $S_{11} = \{1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 19, 20, 21, 22, 25, 26, 27, 31, 32, 33, 36, 37, 38, 39, 40\}$

These solutions correspond to changes in the original assignment examples. These modifications of the original assignment examples are presented bellow:

S_1	original assignment	relaxed assignment
a_{23}	$[C_2, C_3]$	$[C_2, C_4]$
a_{26}	C_5	$[C_3, C_5]$
a_{30}	C_1	$[C_1, C_4]$
a_{31}	C_5	$[C_4, C_5]$
a_{35}	$[C_1, C_2]$	$[C_1, C_3]$

S_2	original assignment	relaxed assignment
a_{23}	$[C_2, C_3]$	$[C_2, C_4]$
a_{26}	C_5	$[C_4, C_5]$
a_{30}	C_1	$[C_1, C_4]$
a_{31}	C_5	$[C_4, C_5]$
a_{35}	$[C_1, C_2]$	$[C_1, C_3]$
a_{39}	C_3	$[C_3, C_4]$

S_3	original assignment	relaxed assignment
a_1	C_5	$[C_4, C_5]$
a_{23}	$[C_2, C_3]$	$[C_2, C_4]$
a_{26}	C_5	$[C_4, C_5]$
a_{30}	C_1	$[C_1, C_4]$
a_{31}	C_5	$[C_4, C_5]$
a_{38}	C_4	$[C_3, C_4]$
a_{39}	C_3	$[C_3, C_4]$

S_4	original assignment	relaxed assignment
a_1	C_5	$[C_4, C_5]$
a_{23}	$[C_2, C_3]$	$[C_2, C_4]$
a_{26}	C_5	$[C_3, C_5]$
a_{30}	C_1	$[C_1, C_4]$
a_{31}	C_5	$[C_4, C_5]$
a_{38}	C_4	$[C_3, C_4]$

S_5	original assignment	relaxed assignment
a_1	C_5	$[C_4, C_5]$
a_{26}	C_5	$[C_3, C_5]$
a_{30}	C_1	$[C_1, C_4]$
a_{31}	C_5	$[C_3, C_5]$
a_{38}	C_4	$[C_3, C_4]$

S_6	original assignment	relaxed assignment
a_1	C_5	$[C_4, C_5]$
a_{26}	C_5	$[C_3, C_5]$
a_{30}	C_1	$[C_1, C_4]$
a_{31}	C_5	$[C_3, C_5]$
a_{35}	$[C_1, C_2]$	$[C_1, C_3]$

S_7	original assignment	relaxed assignment
a_{23}	$[C_2, C_3]$	$[C_2, C_4]$
a_{24}	$[C_2, C_3]$	$[C_2, C_4]$
a_{30}	C_1	$[C_1, C_5]$
a_{31}	C_5	$[C_4, C_5]$
a_{35}	$[C_1, C_2]$	$[C_1, C_3]$
a_{36}	C_4	$[C_4, C_5]$
a_{39}	C_3	$[C_3, C_4]$

S_8	original assignment	relaxed assignment
a_1	C_5	$[C_4, C_5]$
a_{26}	C_5	$[C_3, C_5]$
a_{30}	C_1	$[C_1, C_3]$
a_{31}	C_5	$[C_3, C_5]$
a_{36}	C_4	$[C_3, C_4]$
a_{38}	C_4	$[C_2, C_4]$
a_{39}	C_3	$[C_2, C_3]$

S_9	original assignment	relaxed assignment
a_{23}	$[C_2, C_3]$	$[C_2, C_4]$
a_{26}	C_5	$[C_3, C_5]$
a_{30}	C_1	$[C_1, C_3]$
a_{31}	C_5	$[C_3, C_5]$
a_{35}	$[C_1, C_2]$	$[C_1, C_3]$
a_{36}	C_4	$[C_3, C_4]$
a_{38}	C_4	$[C_2, C_4]$
a_{39}	$C3$	$[C_2, C_3]$

S_{10}	original assignment	relaxed assignment
a_{18}	C_4	$[C_4, C_5]$
a_{23}	$[C_2, C_3]$	$[C_2, C_5]$
a_{24}	$[C_2, C_3]$	$[C_2, C_5]$
a_{30}	C_1	$[C_1, C_5]$
a_{35}	$[C_1, C_2]$	$[C_1, C_3]$
a_{36}	C_4	$[C_4, C_5]$
a_{38}	C_4	$[C_4, C_5]$
a_{39}	C_3	$[C_3, C_5]$

S_{11}	original assignment	relaxed assignment
a_1	C_5	$[C_1, C_5]$
a_{18}	C_4	$[C_1, C_4]$
a_{23}	$[C_2, C_3]$	$[C_1, C_3]$
a_{24}	$[C_2, C_3]$	$[C_1, C_3]$
a_{26}	C_5	$[C_1, C_5]$
a_{31}	C_5	$[C_1, C_5]$
a_{36}	C_4	$[C_1, C_4]$
a_{38}	C_4	$[C_1, C_4]$
a_{39}	C_3	$[C_1, C_3]$