Multiple criteria sorting with a set of additive value functions

Salvatore Greco, Vincent Mousseau, Roman Slowinski

UNIVERSITE PARIS DAUPHINE

Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision

CNRS UMR 7024

CAHIER DU LAMSADE

282

Décembre 2008

Multiple criteria sorting with a set of additive value functions

S. Greco, V. Mousseau, R. Slowinski

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Multiple criteria sorting with a set of additive value functions

Salvatore Greco[1], Vincent Mousseau[2], Roman Słowiński[3]

Abstract

We present a new multiple criteria sorting method that aims at assigning actions evaluated on multiple criteria to p pre-defined ordered classes. The preference information supplied by the decision maker (DM) is a set of assignment examples concerning reference actions. Each assignment example specifies a desired assignment for a reference action to one or several contiguous classes. The assignment examples used to build a preference model based on a set of general additive compatible value functions. For each compatible value function one can associate a set of $p-1$ thresholds on the value function separating consecutive classes. For each action a, the method computes two kinds of assignments to classes, concordant with the DM's preference model. The necessary assignment specifies the range of classes to which the action can be assigned considering all compatible value functions simultaneously. The possible assignment specifies, in turn, the range of classes to which the action can be assigned considering any compatible value functions individually. The compatible value functions and the necessary and possible assignments are computed through the resolution of linear programs. Based on the concepts of necessary and possible assignments, an interactive methodology is proposed, and illustrated on a real world example

Keywords: *Multiple criteria sorting, Additive value function, Disaggregation-aggregation procedure.*

[1]Faculty of Economics, University of Catania, Corso Italia, 55, 95129 Catania, Italy, e-mail: salgreco@mbox.unict.it

[2]LGI, Ecole Centrale Paris, Grande Voie des Vignes, 92 295 Châtenay Malabry France, e-mail: vincent.mousseau@ecp.fr

[3]Institute of Computing Science, Poznań University of Technology, 60-965 Poznań, and Systems Research Institute, Polish Academy of Sciences, 01-447 Warsaw, Poland, e-mail: roman.slowinski@cs.put.poznan.pl

Tri multicritère fondé sur un ensemble de fonctions de valeurs additives

Résumé

Nous présentons une nouvelle méthode de tri multicritère visant à affecter des actions évaluées sur différents critères à p classes pre-définies et ordonnées. L'information préférentielle fournie par le décideur (DM) est un ensemble d'exemple d'affectation correspondant à un sous-ensemble d'actions (appelées actions de référence) relativement bien connues du DM. Chaque exemple d'affectation spécifie l'affectation souhaitée (une classe ou un intervalle de classes) pour l'action de référence. Les exemples d'affectations sont utilisés pour bâtir un modèle de préférence du DM. Dans notre cas, le modèle de préférence sous-jacent est l'ensemble de fonctions de valeur additives compatible avec les exemples d'affectation. pour chaque fonction de valeur compatible, il est possible d'associer $p-1$ seuils séparant les classes consécutives sur l'échelle de cette fonction de valeur. Pour chaque action a, la méthode calcule une affectation possible et une affectation nécessaire. L'affectation nécessaire spécifie l'intervalle de classes auxquelles l'action peut être affectée pour toute fonction de valeur compatible tandis que l'affectation possible spécifie l'intervalle de classes auxquelles l'action peut être affectée pour au moins une fonction de valeur compatible. Le calcul de ces affections est effectué par résolution de programmes linéaires.

Cette méthode est conçue pour être utilisées interactivement, c'est-à-dire que le décideur peut fournir progressivement des exemple d'affectation. La méthode permet également d'aider le DM lorsqu'il fournit un ensemble d'exemples d'affectation qu'il n'est pas possible de représenter par une fonction de valeur additive. De plus, le DM peut spécifier des niveaux de confiance qualitatifs ("certain", "sûr", "assez sûr") pour les exemples d'affectation. Dans ce cas, les affectations possibles et nécessaires s'expriment comme des intervalles de classes correspondant aux niveaux de confiance.

Mots-clés: *Tri multicritère, Fonction de valeur additive value, Procédure d'agrégation désagrégation.*

1 Introduction

Real-world decision problems can be represented by multiple criteria models in which each point of view from which actions under consideration should be analyzed, is formulated by a criterion function. When using such multicriteria approach to decision modeling, several problematiques (or problem formulations) can be considered. [Roy96] distinguishes three fundamental problematiques for decision aiding: choice, ranking and sorting. Given a set A of actions, the choice problematique consists in a choice of a subset (as small as possible) composed of actions being judged as the most satisfying. The ranking problematic consists in establishing a preference pre-order (either partial or complete) on the set of actions. These two problematiques are said to be *comparative*, as they require to compare actions one to another to elaborate the choice set and the preference pre-order. The sorting problematic consists in formulating the decision problem in terms of a classification so as to assign each action to one of the predefined classes. The assignment of an action to the appropriate class relies on the intrinsic value of a (and not on the comparison to others).

In this paper, we are interested in the multiple criteria sorting problematique and, more precisely, in an multicriteria sorting methods based on value functions. Multiple attribute value theory (see [KR93]) is a well established compensatory preference model to represent how a Decision Maker (DM) account for tradeoffs among criteria. Several value based sorting methods have been proposed in the literature, namely [DGJL80], [Jac95], [ZD00a], [ZD00b], [KU03] [KBO09] (see also [ZD02] for a review).

However, a major difficulty when implementing value based sorting models comes from the fact that the value function must be elicited from the DM. Therefore, several authors (see e.g. [DGJL80], [ZD00b]) proposed methodologies that avoids direct elicitation of the value function, but elicit the value function indirectly from assignment examples (typical actions that should be assign to classes) supplied by the DM. The value function is then inferred through a certain form of regression on assignment examples. Such indirect elicitation is usually called disaggregation. Namely, [KBO09] method involve indirect specification of preferences through assignment of reference actions and provide possible assignments for non-reference actions based on a piecewise linear additive value function.

In this paper, we present a new multiple criteria value based sorting method called UTADIS$^{\text{GMS}}$. This method requires that the DM expresses his/her preferences providing a set of assignment examples on a subset of actions (s)he knows relatively well. Each assignment example specifies a desired assignment of a corresponding reference action to one or several contiguous classes. The set of assignment examples is a preference information used to build a preference model. Here, the underlying preference model is a set of general additive value functions compatible with the assignment examples. For each action a, the method computes two kinds of assignments to classes, concordant with the DM's preference model: necessary and possible. The necessary assignment specifies range of classes to which the action can be assigned considering all compatible value functions simultaneously. The possible assignment specifies, in turn, the range of classes to which the action can be assigned considering any compatible value functions individually. The compatible value functions and the necessary and possible assignments are computed through the resolution of linear programs.

The paper is organized as follows. Notation and definitions are introduced in the next Section. Section 3 considers sorting procedures grounded on a single value function and studies the relation between two particular procedures: "threshold based" and "example based" sorting procedures. Considering simultaneously all value functions compatible with assignment examples requires to study sorting procedures based on a set of value function. Such analysis is provided in section 4. Section 5 is devoted to the presentation of the new sorting method called UTADIS$^{\text{GMS}}$. In section 6, we study the case in which the DM provides assignment example that can not be represented by

a value function. Section 7 propose an extension of the method that makes it possible to account for confidence judgments attached to the assignment examples. Section 8 presents an illustrative example of the proposed method on a real world case study. The last section groups conclusions and presents some proposals for future research.

2 Notation and definitions

We shall use the following notation:

- $A = \{a_1, a_2, \ldots, a_j, \ldots, a_m\}$ - a finite set of m actions to be assigned to classes,
- $g_1, g_2, \ldots, g_i, \ldots, g_n$ - n evaluation criteria, $g_i : A \to \mathbb{R}$ for all $i \in G = \{1, 2, \ldots, n\}$,
- $C_1, C_2, \ldots, C_p$ - p predefined preference ordered classes, where $C_{h+1} \gg C_h$ ($\gg$ a complete order on the set of classes), $h = 1, \ldots, p-1$, moreover, $H = \{1, \ldots, p\}$,
- $X_j = \{x_j \in \mathbb{R} : g_j(a_i) = x_j, a_i \in A\}$ - the set of all different evaluations on g_j, $j \in G$,
- $x_j^0, x_j^1, \ldots, x_j^{m_j}$ - the ordered values of X_i, $x_i^k < x_i^{k+1}, k = 0, 1, \ldots, m_i - 1$, $m_i \leq m$ ($m_i < m$ when at least two actions have the same evaluation on criterion g_i).

In order to represent DM's preferences, we shall use a value function U such that:

$$U(a) = \sum_{j=1}^{n} u_j(g_j(a)) \tag{1}$$

where the marginal value functions u_j are defined by $u_j(x_j^k)$, $k = 0, 1, \ldots, m_j$, such that:

$$u_j(x_j^k) \leq u_j(x_j^{k+1}),\ k = 0, 1, \ldots, m_j - 1, j \in G. \tag{2}$$

To normalize the value function so that $U(a) \in [0, 1], \forall a \in A$, we set:

$$\left.\begin{array}{l} u_j(x_j^0) = 0, \forall j \in G \\ \sum_{j=1}^{n} u_j(x_j^{m_j}) = 1 \end{array}\right\} \tag{3}$$

In this paper, we are interested in sorting procedures that aim at assigning each action to one class or to a set of contiguous classes. The procedures we consider are *value driven sorting procedures*, that is, they use a single value function U (or a set of value functions) to decide the assignments in such a way that if $U(a) > U(b)$ then a is assigned to a class not worse than b.

3 Sorting with a single additive value function

Given a value function U defined as (1), two different sorting procedures can be considered:

- *threshold-based* value driven sorting procedure, in which the limits between consecutive classes are defined by thresholds on the utility scale,
- *example-based* value driven sorting procedure, in which classes are implicitly delimited by some assignment examples.

Definition 3.1. *A threshold-based value driven sorting procedure is driven by a value function U and its associated thresholds b_h^U, $h \in H$, in the following way:*

$$a \in A \text{ is assigned to class } C_h, \text{ denoted as } a \to C_h, \text{ iff } U(a) \in [b_{h-1}^U, b_h^U[,$$

where b_{h-1}^U is the minimum value for an action to be assigned to class C_h, and b_h^U is the supremum value for an action to be assigned to class C_h. Obviously, we set $b_0^U = 0$ and, moreover, we impose $b_{h-1}^U < b_h^U$, $\forall h \in H$.

Such a threshold-based value-driven sorting procedure has been used in the UTADIS method [DGJL80], [ZD00b].

In order to define the example-based value driven sorting procedure, let us consider preference information provided by the DM in terms of assignment examples on a subset of actions $A^* \subseteq A$, as defined bellow.

Definition 3.2. *An assignment example consists of an action $a^* \in A^*$ for which the DM defined a desired assignment $a^* \to [C_{L^{DM}(a^*)}, C_{R^{DM}(a^*)}]$, where $[C_{L^{DM}(a^*)}, C_{R^{DM}(a^*)}]$ is an interval of contiguous classes $C_{L^{DM}(a^*)}$, $C_{L^{DM}(a^*)+1}$, ..., $C_{R^{DM}(a^*)}$, $L^{DM}(a^*) \leq R^{DM}(a^*)$. Each such action is called a reference action. $A^* \subseteq A$ is called the set of reference actions. An assignment example is said to be precise if $L^{DM}(a^*) = R^{DM}(a^*)$, and imprecise, otherwise.*

Given a value function U, a set of assignment examples is said to be *consistent with U* iff

$$\forall a^*, b^* \in A^*, U(a^*) \geq U(b^*) \;\Rightarrow\; R^{DM}(a^*) \geq L^{DM}(b^*) \tag{4}$$

In this section, we suppose that the DM provides a set of assignment examples consistent with U.

Definition 3.3. *An example-based value driven sorting procedure is driven by a value function U and its associated assignment examples $a \in A^*$. It assigns an action $a \in A$ to an interval of classes $[C_{L^U(a)}, C_{R^U(a)}]$, in the following way:*

$$L^U(a) = Max_{a^* \in A^*} \left\{ L^{DM}(a^*) : U(a^*) \leq U(a) \right\} \tag{5}$$

$$R^U(a) = Min_{a^* \in A^*} \left\{ R^{DM}(a^*) : U(a^*) \geq U(a) \right\} \tag{6}$$

Let us remark that such example-based sorting procedure has been used in [KU03] along with a linear value function. The following lemma states that the interval $[C_{L^U(a)}, C_{R^U(a)}]$ is not empty.

Proposition 3.1. *For any action $a \in A \setminus A^*$, the example-based sorting procedure provides an assignment $a \to [C_{L^U(a)}, C_{R^U(a)}]$, such that $L^U(a) \leq R^U(a)$.*

Proof. See Appendix A.1 □

Proposition 3.2. *The example-based sorting procedure assigns each reference action $a^* \in A^*$ to an interval of classes $[C_{L^U(a^*)}, C_{R^U(a^*)}]$, such that:*

$$L^U(a^*) \geq L^{DM}(a^*) \tag{7}$$

$$R^U(a^*) \leq R^{DM}(a^*) \tag{8}$$

Proof. See Appendix A.2 □

Now, let us study the relationship between the threshold-based (Definition 3.1) and example-based (Definition 3.3) sorting procedures.

Proposition 3.3. *Consider the case where the DM provides precise assignment examples only, i.e., $L^{DM}(a^*) = R^{DM}(a^*)$, for each $a^* \in A^*$. Assuming the use of a single value function U in the example-based sorting procedure, if we choose, for each $h = 1, ..., p-1$, the threshold b_h^U in the interval $]Max_{a^* \to C_h}\{U(a^*)\}, Min_{a^* \to C_{h+1}}\{U(a^*)\}]$, we obtain a threshold-based sorting procedure that restores the assignment examples and assigns each non-reference action $a \in A \setminus A^*$ to a single class in the interval $[C_{L^U(a)}, C_{R^U(a)}]$ stemming from the example-based sorting procedure.*

Proof. See Appendix A.3 □

Figure 1 illustrates Proposition 3.3 in a sorting example involving 3 classes and 18 reference actions. In this case, assignment examples are such that $a_1^*, a_2^*, a_3^*, a_4^*, a_5^*$ and a_6^* are assigned to class C_1; $a_7^*, a_8^*, a_9^*, a_{10}^*, a_{11}^*$ and a_{12}^* are assigned to C_2, and the remaining actions are assigned to C_3. Consider a non-reference action $a' \in A \setminus A^*$. According to the value of $U(a')$, the example-based sorting procedure assigns a' as follows:

- If $U(a') \leq U(a_6^*)$, then $a' \to C_1$
- If $U(a') \in]U(a_6^*), U(a_7^*)[$, then $a' \to [C_1, C_2]$
- If $U(a') \in [U(a_7^*), U(a_{12}^*)]$, then $a' \to C_2$
- If $U(a') \in]U(a_{12}^*), U(a_{13}^*)[$, then $a' \to [C_2, C_3]$
- If $U(a') \geq U(a_{13}^*)$, then $a' \to C_3$

Consistently with Proposition 3.3, it appears that if we fix the thresholds such that $b_1 \in]U(a_6^*, U(a_7^*)]$ and $b_2 \in]U(a_{12}^*, U(a_{13}^*)]$, the threshold-based sorting procedure will assign a new actions a' consistently with the example-based sorting procedure, i.e.:

- If $U(a') \leq U(a_6^*)$, then $a' \to C_1$,
- If $U(a') \in]U(a_6^*), U(a_7^*)[$, then $a' \to C_1$ or C_2, depending whether $U(a') < b_1^U$ or not,
- If $U(a') \in [U(a_7^*), U(a_{12}^*)]$, then $a' \to C_2$,
- If $U(a') \in]U(a_{12}^*), U(a_{13}^*)[$, then $a' \to C_2$ or C_3, depending whether $U(a') < b_2^U$ or not,
- If $U(a') \geq U(a_{13}^*)$, then $a' \to C_3$.

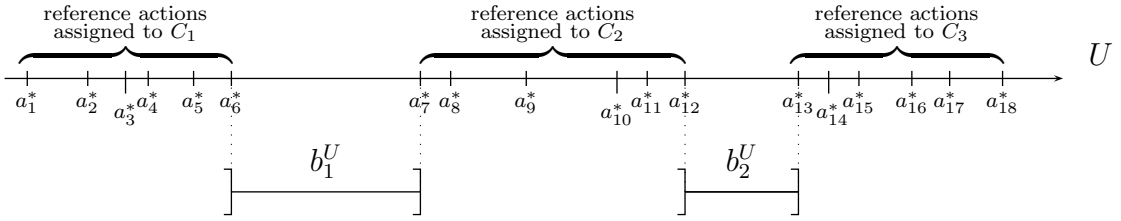

Figure 1: Intervals for the thresholds separating classes (precise assignments)

Proposition 3.4. *Consider the case where the DM provides possibly imprecise assignment examples, i.e., $L^{DM}(a^*) \leq R^{DM}(a^*)$ for each $a^* \in A^*$. Assuming the use of a single value function U in the example-based sorting procedure, if we choose, for each $h = 1, ..., p-1$, the threshold b_h^U in the interval $I(b_h^U) =]Max_{a^*: R^{DM}(a^*) \leq h}\{U(a^*)\}, Min_{a^*: L^{DM}(a^*) > h}\{U(a^*)\}[$, with $b_h^U < b_{h+1}^U$ we obtain a threshold-based sorting procedure that assigns each reference action $a^* \in A^*$ to a single class in the interval $[C_{L^{DM}(a^*)}, C_{R^{DM}(a^*)}]$, and assigns each non-reference action $a \in A \setminus A^*$ to a single class in the interval $[C_{L^U(a)}, C_{R^U(a)}]$ stemming from the example-based sorting procedure.*

Proof. See Appendix A.4 □

Figure 2 illustrates Proposition 3.4 in an example involving 5 classes and 11 reference actions. In this case, assignment examples are the following:

- a_1^* and $a_3^* \to [C_1, C_2]$
- $a_2^* \to [C_1, C_3]$
- a_4^* and $a_6^* \to [C_2, C_3]$
- $a_5^* \to [C_2, C_5]$
- a_7^* and $a_9^* \to [C_3, C_4]$
- $a_8^* \to [C_3, C_5]$
- a_{10}^* and $a_{11}^* \to [C_4, C_5]$

Consider a non-reference action $a' \in A \setminus A^*$. According to the value of $U(a')$, the example-based sorting procedure assigns a' as follows:

- If $U(a') < U(a_3^*)$, then $a' \to [C_1, C_2]$
- If $U(a') \in [U(a_3^*), U(a_4^*)[$, then $a' \to [C_1, C_3]$
- If $U(a') \in [U(a_4^*), U(a_6^*)[$, then $a' \to [C_2, C_3]$
- If $U(a') \in [U(a_6^*), U(a_7^*)[$, then $a' \to [C_2, C_4]$
- If $U(a') \in [U(a_7^*), U(a_9^*)[$, then $a' \to [C_3, C_4]$
- If $U(a') \in [U(a_9^*), U(a_{10}^*)[$, then $a' \to [C_3, C_5]$
- If $U(a') \geq U(a_{10}^*)$, then $a' \to [C_4, C_5]$

Let us study now the value of thresholds b_1, b_2, b_3, and b_4 (in the threshold-based sorting procedure) which are compatible with the assignment examples.

- b_1 is necessarily smaller or equal than $U(a_4^*)$ as a_4^* is assigned to at least C_2.
- b_2 is necessarily smaller or equal than $U(a_7^*)$ as a_7^* is assigned to at least C_3.
- b_3 is necessarily greater than $U(a_6^*)$ as a_6^* is assigned to at most C_3, and lower or equal than $U(a_{10}^*)$ as a_{10}^* is assigned to at least C_4.
- b_4 is necessarily greater than $U(a_9^*)$ as a_9^* is assigned to at most C_4.

Obviously, in order to build a consistent thresholds-based sorting procedure, these thresholds should also be such that $b_i < b_{i+1}$, $i = 1, ..., 4$. Consistently with Proposition 3.3, it appears that if we fix the thresholds as above, the threshold-based sorting procedure will assign a new action a' consistently with the example-based sorting procedure, i.e.:

- If $U(a') < u(a_3^*)$, then $a' \to C_1$ or C_2, depending whether $U(a') < b_1^U$ or not,
- If $U(a') \in [u(a_3^*), u(a_4^*)[$, then $a' \to C_1$, C_2 or C_3, depending whether $U(a') < b_1^U$, $U(a') \in [b_1, b_2[$, or $U(a') \geq b_2$,
- If $U(a') \in [u(a_4^*), u(a_6^*)[$, then $a' \to C_2$ or C_3, depending whether $U(a') < b_2^U$ or not,
- If $U(a') \in [u(a_6^*), u(a_7^*)[$, then $a' \to C_2$, C_3 or C_4, depending whether $U(a') < b_2^U$, $U(a') \in [b_2, b_3[$, or $U(a') \geq b_3$,
- If $U(a') \in [u(a_7^*), u(a_9^*)[$, then $a' \to C_3$ or C_4, depending whether $U(a') < b_3^U$ or not,
- If $U(a') \in [u(a_9^*), u(a_{10}^*)[$, then $a' \to C_3$, C_4 or C_5, depending whether $U(a') < b_3^U$, $U(a') \in [b_3, b_4[$, or $U(a') \geq b_4$,
- If $U(a') \geq u(a_{10}^*)$, then $a' \to C_4$ or C_5, depending whether $U(a') < b_4^U$ or not,

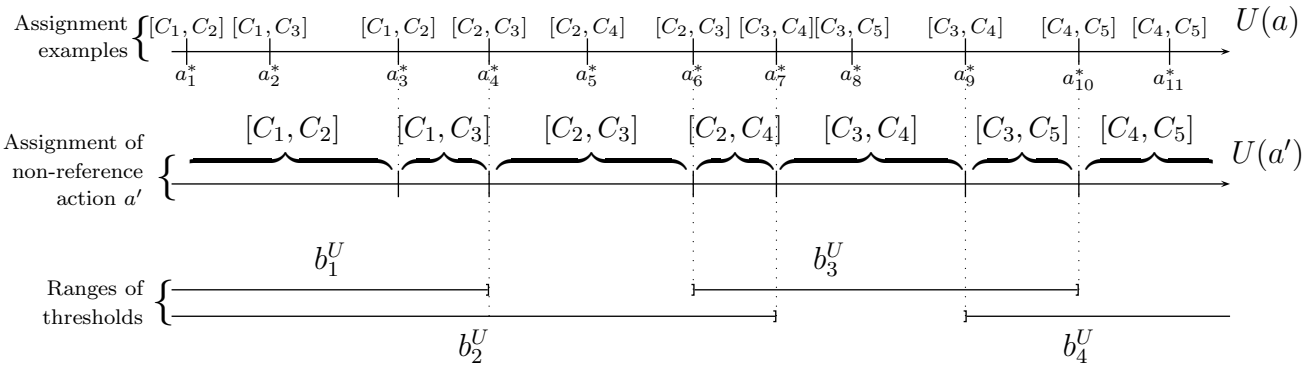

Figure 2: Intervals for the thresholds separating classes (imprecise assignments)

Hence, Propositions 3.3 and 3.4 show that the example-based and the threshold-based sorting procedures provide consistent results, the former being more general than the latter, as it implicitly considers intervals of possible thresholds instead of single values for the thresholds. Therefore, in the rest of the paper, we will consider the example-based sorting model only.

4 Sorting with a set of value functions

In this section we consider value driven sorting procedures based on a set $\mathcal{U}$ of value functions, and we will restrict to the example-based sorting procedure.

Definition 4.1. *Considering a set A^* of assignment examples, the set $\mathcal{U}_{A^*}$ of compatible value functions is defined by all value functions U, satisfying $U(a^*) > U(b^*)$ for all $a^*, b^* \in A^*$ such that $R^{DM}(b^*) < L^{DM}(a^*)$.*

Definition 4.2. *Given a set A^* of assignment examples and a corresponding set $\mathcal{U}_{A^*}$ of compatible value functions, for each $a \in A$, we define the possible assignment $C_P(a)$ as the set of indices of classes C_h for which there exists at least one value function $U \in \mathcal{U}_{A^*}$ assigning a to C_h, and the necessary assignment $C_N(a)$ as set of indices of classes C_h for which all value functions $U \in \mathcal{U}_{A^*}$ assign a to C_h, that is:*

$$C_P(a) = \{h \in H : \exists U \in \mathcal{U}_{A^*} \text{ for which } h \in [L^U(a), R^U(a)]\} \quad (9)$$

$$C_N(a) = \{h \in H : \forall U \in \mathcal{U}_{A^*} \text{ it holds } h \in [L^U(a), R^U(a)]\} \quad (10)$$

It is obvious to observe that it holds for each $a \in A$:

$$\begin{cases} C_P(a) = \bigcup_{U \in \mathcal{U}_{A^*}} [L^U(a), R^U(a)], \\ C_N(a) = \bigcap_{U \in \mathcal{U}_{A^*}} [L^U(a), R^U(a)], \quad \forall a \in A \\ C_N(a) \subseteq C_P(a) \end{cases} \quad (11)$$

Proposition 4.1. *For any $a \in A$, there exists $U \in \mathcal{U}_{A^*}$ and values of the thresholds b_h, $h \in H$ such that for any $h \in C_P(a)$ the threshold-based sorting model assigns a to class C_h, for each $h \in [L^U(a), R^U(a)]$.*

Proof. Since $C_P(a) = \{h \in H$ such that $\exists U \in \mathcal{U}_{A^*}$ for which $h \in [L^U(a), R^U(a)]\}$, considering that according to proposition 3.4 we have that for all $U \in \mathcal{U}_{A^*}$, there exist values for the thresholds b_h, $h \in H$ such that for any $h \in C_P(a)$ the threshold-based sorting model assigns a to class C_h, for each $h \in [L^U(a), R^U(a)]$, we get the thesis. □

Definition 4.3. *Given a set A^* of assignment examples and a corresponding set $\mathcal{U}_{A^*}$ of compatible value functions, for each $a \in A$, we define the following indices of classes:*

- *Minimum possible class:* $L_P^{\mathcal{U}}(a) = Max_{a \in A^*} \left\{ L^{DM}(a^*) : \forall U \in \mathcal{U}_{A^*}, U(a^*) \leq U(a) \right\}$
- *Minimum necessary class:* $L_N^{\mathcal{U}}(a) = Max_{a \in A^*} \left\{ L^{DM}(a^*) : \exists U \in \mathcal{U}_{A^*} \text{ for which } U(a^*) \leq U(a) \right\}$
- *Maximum necessary class:* $R_N^{\mathcal{U}}(a) = Min_{a \in A^*} \left\{ R^{DM}(a^*) : \exists U \in \mathcal{U}_{A^*} \text{ for which } U(a) \leq U(a^*) \right\}$
- *Maximum possible class:* $R_P^{\mathcal{U}}(a) = Min_{a \in A^*} \left\{ R^{DM}(a^*) : \forall U \in \mathcal{U}_{A^*}, U(a) \leq U(a^*) \right\}$

An analysis of these indices is provided by the following Proposition, namely, in what concerns how these indices relates to $C_P(a)$ and $C_N(a)$.

Proposition 4.2. *Given a set A^* of assignment examples and a corresponding set $\mathcal{U}_{A^*}$ of compatible value functions, it holds, for each $a \in A$,*

- $L_P^{\mathcal{U}}(a) \leq L_N^{\mathcal{U}}(a)$,
- $R_N^{\mathcal{U}}(a) \leq R_P^{\mathcal{U}}(a)$,
- $L_P^{\mathcal{U}}(a) \leq R_P^{\mathcal{U}}(a)$.

Moreover, for any $a \in A$, we have:

- $C_P(a) \subseteq [L_P^{\mathcal{U}}(a), R_P^{\mathcal{U}}(a)]$,
- *if* $L_N^{\mathcal{U}}(a) \leq R_N^{\mathcal{U}}(a)$, *then* $C_N(a) = [L_N^{\mathcal{U}}(a), R_N^{\mathcal{U}}(a)]$,
- *if* $L_N^{\mathcal{U}}(a) > R_N^{\mathcal{U}}(a)$, *then* $C_N(a) = \emptyset$.

Proof. See Appendix B.1 □

In this context, two important questions arise:

- under which condition on value functions $U \in \mathcal{U}_{A^*}$ we have $C_N(a) \neq \emptyset$?
- under which condition on value functions $U \in \mathcal{U}_{A^*}$ the following "*no jump property property*" holds? If $h, h' \in C_P(a)$ with $h \leq h'$, then $h'' \in C_P(a), \forall h'' \in [h, h']$ (12)

Let us observe that the first question is related to the robustness of the assignment, in the sense that it concerns classes to which a is assigned by all compatible value functions $U \in \mathcal{U}_{A^*}$. The second question instead regards the presence of a "jump" in the classes of the possible assignment. In fact, the "no jump property" (12) is equivalent to $C_P(a) = [L_P^U, R_P^U]$, which means that no class between the worst, $C_{L_P^U}$, and the best, $C_{L_P^U}$, is absent in the possible assignment. These two questions are studied in Propositions (4.3)-(4.6)

Proposition 4.3. *$C_N(a) \neq \emptyset$ if and only if the following strict continuity condition is satisfied: for all $a^*, b^* \in A^*$ such that $L^{DM}(a^*) > R^{DM}(b^*)$ there are no two compatible value functions $U, U' \in \mathcal{U}_{A^*}$ for which $U(a) \geq U(a^*)$ and $U'(b^*) \geq U'(a)$.*

Proof. See Appendix B.2 □

The following condition is important with respect to the absence of jumps in in the classes of the possible assignment.

Definition 4.4. ***(Weak continuity)*** *For all $a^*, b^* \in A^*$ such that $L^{DM}(a^*) > R^{DM}(b^*)$, if there exist $U', U'' \in \mathcal{U}_{A^*}$ for which $U'(a) \geq U'(a^*)$ and $U''(b^*) \geq U''(a)$, then there exists $U''' \in \mathcal{U}_{A^*}$ for which $U'''(a^*) \geq U'''(a) \geq U'''(b^*)$.*

Proposition 4.4. *The no jump property (12) holds, and therefore $C_P(a) = [L_P^U, R_P^U]$, if and only if the weak continuity (4.4) holds.*

Proof. See Appendix B.3 □

Corollary 4.1. *If the set $\mathcal{U}_{A^*}$ of compatible value functions satisfies the weak continuity (4.4), the example-based sorting procedure assigns each reference action $a^* \in A^*$ to an interval of classes $[L_P^U(a^*), R_P^U(a^*)] \subseteq [L^{DM}(a^*), R^{DM}(a^*)]$.*

Proof. See Appendix B.4 □

Definition 4.5. *A set $\mathcal{U}$ of value functions is convex if for all $U, U' \in \mathcal{U}$ and for all $\alpha \in [0,1]$, $\alpha U + (1-\alpha)U' \in \mathcal{U}$.*

Proposition 4.5. *If the set $\mathcal{U}_{A^*}$ of compatible value functions is convex, the $C_P(a) = [L_P^{\mathcal{U}}(a), R_P^{\mathcal{U}}(a)]$, i.e. the no jump property (12) holds.*

Proof. See Appendix B.5 □

Proposition 4.6. *If $\mathcal{U}_{A^*}$ is the set of all additive compatible value functions, then $C_P(a) = [L_P^{\mathcal{U}}(a), R_P^{\mathcal{U}}(a)]$, i.e. the no jump property (12) holds.*

Proof. See Appendix B.6 □

5 A new sorting method using a set of additive value functions

The new UTADIS$^{\text{GMS}}$ method (GMS stands for Greco Mousseau Slowinski) is an ordinal regression method for sorting problems using a set of additive value functions $U(a) = \sum_{i=1}^{n} u_i(g_i(a))$ as a preference model. One of its characteristic features is that it takes into account the set of all value functions compatible with the preference information provided by the DM. Moreover, it considers general non-decreasing marginal value functions instead of piecewise linear only.

The DM is supposed to provide preference information in form of possibly imprecise assignment examples on a reference set A^*, i.e., for all $a^* \in A^*$ the DM defines a desired assignment $a^* \rightarrow [C_{L^{DM}(a^*)}, C_{R^{DM}(a^*)}]$. According to definition (4) a value function is *compatible* if $\forall a^*, b^* \in A^*, U(a^*) \geq U(b^*) \Rightarrow R^{DM}(a^*) \geq L^{DM}(b^*)$. Definition (4) is equivalent to

$$\forall a^*, b^* \in A^*,\ L^{DM}(a^*) > R^{DM}(b^*) \Rightarrow U(a^*) > U(b^*) \tag{13}$$

On the basis of (13), we can state that, formally, a general additive compatible value function is an additive value function $U(a) = \sum_{i=1}^{n} u_i(a)$ satisfying the following set of constraints:

$$\left.\begin{array}{l} U(a) > U(b) \;\Leftrightarrow\; L^{DM}(a^*) > R^{DM}(b^*),\ \forall a^*, b^* \in A^* \\ u_i(g_i(a_{\tau_i(j)})) - u_i(g_i(a_{\tau_i(j-1)})) \geq 0,\ i = 1, ..., n,\ j = 2, ..., m \\ u_i(g_i(a_{\tau_i(1)})) \geq 0, u_i(g_i(a_{\tau_i(m)})) \leq u_i(\beta_i),\ i = 1, ..., n, \\ u_i(\alpha_i) = 0,\ i = 1, ..., n \\ \sum_{i=1}^{n} u_i(\beta_i) = 1 \end{array}\right\} (E^{A^*})$$

where α_i and β_i are the the minimum and maximum possible evaluation on the scale X_i of criterion g_i and where τ_i is the permutation on the set of indices of actions from A^* that reorders them according to the increasing evaluation on criterion g_i, i.e.

$$g_i(a_{\tau_i(1)}) \leq g_i(a_{\tau_i(2)}) \leq \ldots \leq g_i(a_{\tau_i(m-1)}) \leq g_i(a_{\tau_i(m)})$$

Let us observe that the set of constraints (E^{A^*}) is equivalent to

$$\left.\begin{array}{l} U(a^*) \geq U(b^*) + \varepsilon \;\Leftrightarrow\; L^{DM}(a^*) > R^{DM}(b^*),\ \forall a^*, b^* \in A^* \\ u_i(g_i(a_{\tau_i(j)})) - u_i(g_i(a_{\tau_i(j-1)})) \geq 0,\ i = 1, ..., n,\ j = 2, ..., m \\ u_i(g_i(a_{\tau_i(1)})) \geq 0, u_i(g_i(a_{\tau_i(m)})) \leq u_i(\beta_i),\ i = 1, ..., n, \\ u_i(\alpha_i) = 0,\ i = 1, ..., n \\ \sum_{i=1}^{n} u_i(\beta_i) = 1 \end{array}\right\} (E^{A^*})'$$

with $\varepsilon > 0$. Thus, to check if the set of all compatible value functions $\mathcal{U}_{A^*}$ is not empty it is sufficient to verify if $\varepsilon^* > 0$, where $\varepsilon^* = max\ \varepsilon$, subject to set of constraints $(E^{A^*})'$.

Remark that, due to this formulation of the ordinal regression problem, no linear interpolation is required to express the marginal value of any reference action. Thus, one cannot expect that increasing the number of characteristic points will bring some "new" compatible additive value functions. In consequence, UTADIS$^{\text{GMS}}$ considers all compatible additive value functions while classical UTADIS deals with a subset of the whole set of compatible additive value functions, more precisely the subset of piecewise linear additive value functions relative to the considered characteristic points.

Note that considering all criteria values as characteristics points in the modeling of the marginal value functions gives the preference model a great descritive ability, but this has however a conterpart with respect to an increased size of the mathematical program to be solved (all mathematical programs corresponds however to linear programming, such issue is manageable from an operational point of view). Note also that this increased description ability can require the DM to express additional preference information in order to obtain rich recommandation about the alternative outside the reference set.

On the basis of the set of all compatible value functions $\mathcal{U}_{A^*}$, we can define two binary relations on the set of all actions A:

- *necessary* weak preference relation $\succsim^N$, in case $U(a) \geq U(b)$ for all compatible value functions,
- *possible* weak preference relation $\succsim^P$, in case $U(a) \geq U(b)$ for at least one compatible value function.

Using necessary weak preference relation $\succsim^N$ and possible weak preference relation $\succsim^P$, taking into account definition 4.3 we can redefine indices $L_P^{\mathcal{U}}(a)$, $L_N^{\mathcal{U}}(a)$, $R_N^{\mathcal{U}}(a)$ and $R_P^{\mathcal{U}}(a)$ as follows:

- minimum possible class: $L_P^{\mathcal{U}}(a) = Max\left\{L^{DM}(a^*) : a \succsim^N a^*, a^* \in A^*\right\}$ (14)

- Minimum necessary class: $L_N^{\mathcal{U}}(a) = Max\left\{L^{DM}(a^*) : a \succsim^P a^*, a^* \in A^*\right\}$ (15)

- Maximum necessary class: $R_N^{\mathcal{U}}(a) = Min\left\{R^{DM}(a^*) : a^* \succsim^P a, a^* \in A^*\right\}$ (16)

- maximum possible class: $R_P^{\mathcal{U}}(a) = Min\left\{R^{DM}(a^*) : a^* \succsim^N a, a^* \in A^*\right\}$ (17)

Thus, using necessary weak preference relation $\succsim^N$ and possible weak preference relation $\succsim^P$ it is possible to deal quite simply with the sorting problem.

Necessary weak preference relation $\succsim^N$ and possible weak preference relation $\succsim^P$ can be calculated as follows [GMS08, FGS09]. For all actions $a, b \in A$, let π_i be a permutation of the indices of actions from set $A^* \cup \{a, b\}$ that reorders them according to increasing evaluation on criterion g_i, i.e.

$$g_i(a_{\pi_i(1)}) \leq g_i(a_{\pi_i(2)}) \leq ... \leq g_i(a_{\pi_i(\omega-1)}) \leq g_i(a_{\pi_i(\omega)})$$

where

- if $A^* \cap \{a, b\} = \emptyset$, then $\omega = m + 2$
- if $A^* \cap \{a, b\} = \{a\}$ or $A^* \cap \{a, b\} = \{b\}$, then $\omega = m + 1$
- if $A^* \cap \{a, b\} = \{a, b\}$, then $\omega = m$.

Then, we can fix the characteristic points of $u_i(g_i)$, $i = 1, ..., n$, in $g_i^0 = \alpha_i$, $g_i^j = g_i(a_{\pi_i(j)})$ for $j = 1, ..., \omega$, $g_i^{\omega+1} = \beta_i$

Let us consider the following sets $E(a, b)$ and $E(b, a)$ of ordinal regression constraints, with $i = 1, ..., n$, $j = 1, ..., \omega + 1$, as variables:

$$\left.\begin{array}{l} U(a) \geq U(b) \\ U(a^*) \geq U(b^*) + \varepsilon \;\Leftrightarrow\; L^{DM}(a^*) > R^{DM}(b^*),\ \forall a^*, b^* \in A^* \\ u_i(g_i^j) - u_i(g_i^{j-1}) \geq 0,\ i = 1, ..., n,\ j = 1, ..., \omega + 1 \\ u_i(g_i^0) = 0,\ i = 1, ..., n \\ \sum_{i=1}^n u_i(g_i^{\omega+1}) = 1. \end{array}\right\} (E(a, b))$$

and

$$\left.\begin{array}{l} U(b) \geq U(a) \\ U(a^*) \geq U(b^*) + \varepsilon \;\Leftrightarrow\; L^{DM}(a^*) > R^{DM}(b^*),\ \forall a^*, b^* \in A^* \\ u_i(g_i^j) - u_i(g_i^{j-1}) \geq 0,\ i = 1, ..., n,\ j = 1, ..., \omega + 1 \\ u_i(g_i^0) = 0,\ i = 1, ..., n \\ \sum_{i=1}^n u_i(g_i^{\omega+1}) = 1. \end{array}\right\} (E(b, a))$$

The above sets of constraints depend on the pair of actions $a, b \in A$ because their evaluations $g_i(a)$ and $g_i(b)$ give coordinates for two of $(\omega + 1)$ characteristic points of marginal value function $u_i(g_i)$, for each $i = 1, \ldots, n$.

Let us suppose that the polyhedron defined by $E(a,b)$ as well as the one defined by $E(b,a)$ are not empty. In this case we have that:

$$a \succsim^N b \Leftrightarrow \left\{ \begin{array}{l} min\ \varepsilon < 0 \\ \text{s.t. set } E(b,a) \text{ of constraints} \end{array} \right. \tag{18}$$

and

$$a \succsim^P b \Leftrightarrow \left\{ \begin{array}{l} max\ \varepsilon > 0 \\ \text{s.t. set } E(a,b) \text{ of constraints} \end{array} \right. \tag{19}$$

Therefore, on the basis of the above results, the following example-based sorting procedure can be proposed:

1. Ask the DM for sorting examples.
2. Verify that the set of compatible value functions $\mathcal{U}_{A^*}$ is not empty.
3. Calculate the necessary and the possible weak preference relations $a \succsim^N a^*$, $a \succsim^P a^*$, $a^* \succsim^N a$ and $a^* \succsim^N a$, for $a^* \in A^*$ and $a \in A$.
4. Calculate for each $a \in A$ the indices $L_P^{\mathcal{U}}(a)$, $L_N^{\mathcal{U}}(a)$, $R_N^{\mathcal{U}}(a)$ and $R_P^{\mathcal{U}}(a)$ using (14), (15),(16) and (17).
5. Specify for each $a \in A$ its possible assignment $C_P(a) = [L_P^{\mathcal{U}}(a), R_P^{\mathcal{U}}(a)]$.
6. Specify for each $a \in A$ its necessary assignment which is $C_N(a) = [L_N^{\mathcal{U}}(a), R_N^{\mathcal{U}}(a)]$ in case $L_N^{\mathcal{U}}(a) \leq R_N^{\mathcal{U}}(a)$, and $C_N(a) = \emptyset$, otherwise.

Observe that the above procedure is an example based sorting procedure rather than a threshold based procedure. It is also possible to define a threshold based sorting procedure as follows. In a threshold based sorting procedure, a pair $(U, \mathbf{b})$, where U is a value function and $\mathbf{b} = [b_1, \ldots, b_{p-1}]$, is a vector of thresholds $b_h, h = 1, \ldots, p-1$ such that $0 < b_1 < b_2 < ... < b_{p-1} < 1$, is consistent, if for all $a^* \in A^*$

$$b_{R^{DM}(a^*)} - 1 > U(a^*) \geq b_{L^{DM}(a^*)}. \tag{20}$$

On the basis of (20), we can state that, formally, a pair $(U, \mathbf{b})$, where U is an additive value function $U(a) = \sum_{i=1}^n u_i(a)$ and $\mathbf{b} = [b_1, \ldots, b_{p-1}]$ is a vector of thresholds, is compatible, if it satisfies the following set of constraints:

$$\left. \begin{array}{l} \left. \begin{array}{l} U(a^*) \geq b_{L^{DM}(a^*)-1} \\ U(a^*) < b_{R^{DM}(a^*)} \end{array} \right\} \forall a^* \in A^* \\ u_i(g_i(a_{\tau_i(j)})) - u_i(g_i(a_{\tau_i(j-1)})) \geq 0,\ i = 1, ..., n,\ j = 2, ..., m \\ u_i(g_i(a_{\tau_i(1)})) \geq 0, u_i(g_i(a_{\tau_i(m)})) \leq u_i(\beta_i),\ i = 1, ..., n, \\ u_i(\alpha_i) = 0,\ i = 1, ..., n \\ \sum_{i=1}^n u_i(\beta_i) = 1, \\ b_1 > 0, b_{p-1} < 1, \\ b_h > b_{h-1},\ h = 2, \ldots, p-1, \end{array} \right\} (E_T^{A^*})$$

where τ_i is the permutation on the set of indices of actions from A^* that reorders them according to the increasing evaluation on criterion g_i, i.e.

$$g_i(a_{\tau_i(1)}) \leq g_i(a_{\tau_i(2)}) \leq \ldots \leq g_i(a_{\tau_i(m-1)}) \leq g_i(a_{\tau_i(m)})$$

Let us also consider the following set of constraints $(E_T^{A^*})'$:

$$\left.\begin{array}{l} \left.\begin{array}{l} U(a^*) \geq b_{L^{DM}(a^*)-1} \\ b_{R^{DM}(a^*)} - U(a^*) \geq \varepsilon \end{array}\right\} \forall a^* \in A^* \\ u_i(g_i(a_{\tau_i(j)})) - u_i(g_i(a_{\tau_i(j-1)})) \geq 0,\ i = 1, ..., n,\ j = 2, ..., m \\ u_i(g_i(a_{\tau_i(1)})) \geq 0, u_i(g_i(a_{\tau_i(m)})) \leq u_i(\beta_i),\ i = 1, ..., n, \\ u_i(\alpha_i) = 0,\ i = 1, ..., n \\ \sum_{i=1}^{n} u_i(\beta_i) = 1, \\ b_1 \geq \varepsilon, b_{p-1} < 1, \\ b_h - b_{h-1} \geq \varepsilon,\ h = 2, \ldots, p-1, \end{array}\right\} (E_T^{A^*})'$$

Set of constraint $(E_T^{A^*})$ is equivalent to set $(E_T^{A^*})'$ if $\varepsilon > 0$. Thus, to check if the set of all compatible pairs $(U, \mathbf{b})$ is not empty it is sufficient to verify if $\varepsilon^* > 0$, where $\varepsilon^* = max\ \varepsilon$, subject to set of constraints $(E_T^{A^*})'$.

A compatible pair $(U, \mathbf{b})$ assigns action $a \in A$ to a class C_k with $k \geq h$, $h \in H$, if it satisfies set of constraints $(E_T^{A^*}(a \to C_{\geq h}))$ obtained by adding to set of constraints $(E_T^{A^*})$ the constraint $U(a) \geq b_{h-1}$. Analogously, a compatible pair $(U, \mathbf{b})$ assigns action $a \in A$ to a class C_k with $k \leq h$, $h \in H$, if it satisfies set of constraints $(E_T^{A^*}(a \to C_{\leq h}))$ obtained by adding to set of constraints $(E_T^{A^*})$ the constraint $U(a) < b_h$.

From a computational point of view, it is more convenient to express set of constraints $E_T^{A^*}(a \to C_{\geq h})$ as the set of constraints obtained by adding to $(E_T^{A^*})'$ the two further constraints, $\varepsilon > 0$ and $U(a) \geq b_{h-1}$. Consequently, to verify if there is a compatible pair $(U, \mathbf{b})$ assigning action $a \in A$ to a class C_k with $k \geq h$, $h \in H$, it is enough to verify if $\varepsilon^*(C_{\geq h}, a) > 0$, where $\varepsilon^*(C_{\geq h}, a) = max\ \varepsilon$, subject to set of constraints $(E_T^{A^*})'$ plus the constraint $U(a) \geq b_{h-1}$. Instead, all compatible pairs $(U, \mathbf{b})$ assign action $a \in A$ to a class C_k with $k \geq h$, $h \in H$, if $\varepsilon_*(C_{\geq h}, a) > 0$ where $\varepsilon_*(C_{\geq h}, a) = min\ \varepsilon$ subject to set of constraints $(E_T^{A^*})'$ plus the constraint $U(a) \geq b_{h-1}$.

Analogously, from a computational point of view, it is more convenient to express set of constraints $(E_T^{A^*}(a \to C_{\leq h}))$ as the set of constraints obtained by adding to $(E_T^{A^*})'$ the two further constraints $\varepsilon > 0$ and $b_h - U(a) \geq \varepsilon$. Consequently, to verify if there is a compatible pair $(U, \mathbf{b})$ assigning action $a \in A$ to some class C_k with $k \leq h$, $h \in H$, it is enough to verify that $\varepsilon^*(C_{\leq h}, a) > 0$, where $\varepsilon^*(C_{\leq h}, a) = max\ \varepsilon$ subject to set of constraints $(E_T^{A^*})'$ plus the constraint $b_h - U(a) \geq \varepsilon$. Instead, all compatible pairs $(U, \mathbf{b})$ assign action $a \in A$ to some class C_k with $k \leq h$, $h \in H$, if $\varepsilon_*(C_{\leq h}, a) > 0$ where $\varepsilon_*(C_{\leq h}, a) = min\ \varepsilon$, subject to set of constraints $(E_T^{A^*})'$ plus the constraint $b_h - U(a) \geq \varepsilon$.

On the basis of the above considerations, indices $L_P^{\mathcal{U}}(a)$, $L_N^{\mathcal{U}}(a)$, $R_N^{\mathcal{U}}(a)$ and $R_P^{\mathcal{U}}(a)$ can be redefined as follows:

- minimum possible class: $L_P^{\mathcal{U}}(a) = Max\left\{h \in H : \varepsilon_*(C_{\geq h}, a) > 0\right\}$, (21)
- Minimum necessary class: $L_N^{\mathcal{U}}(a) = Max\left\{h \in H : \varepsilon^*(C_{\geq h}, a) > 0\right\}$, (22)

- Maximum necessary class: $R_N^{\mathcal{U}}(a) = Min\left\{h \in H : \varepsilon^*(C_{\leq h}, a) > 0\right\}$, (23)
- maximum possible class: $R_P^{\mathcal{U}}(a) = Min\left\{h \in H : \varepsilon_*(C_{\leq h}, a) > 0\right\}$. (24)

Therefore, on the basis of the above observations, the following threshold-based sorting procedure can be proposed:

1. Ask the DM for sorting examples.
2. Verify if the set of compatible pairs $(U, \mathbf{b})$ is not empty.
3. Calculate for each $a \in A$ the indices $L_P^{\mathcal{U}}(a)$, $L_N^{\mathcal{U}}(a)$, $R_N^{\mathcal{U}}(a)$ and $R_P^{\mathcal{U}}(a)$ using (21), (22),(23) and (24).
4. Specify for each $a \in A$ its possible assignment $C_P(a) = [L_P^{\mathcal{U}}(a), R_P^{\mathcal{U}}(a)]$.
5. Specify for each $a \in A$ its necessary assignment, which is $C_N(a) = [L_N^{\mathcal{U}}(a), R_N^{\mathcal{U}}(a)]$ in case $L_N^{\mathcal{U}}(a) \leq R_N^{\mathcal{U}}(a)$, and $C_N(a) = \emptyset$, otherwise.

6 Management of inconsistencies

6.1 Analysis of incompatibility

Let us consider now the case where there is no value function U in an example-based sorting procedure, or equivalently, no pair $(U, \mathbf{b})$ in a threshold-based sorting procedure, compatible with the assignment examples . We say, this is the case of incompatibility. In such a case, the polyhedron generated by constraints E^{A^*} is empty or, equivalently, the maximal value of ε satisfying set of constraints $(E^{A^*})'$ is not positive. Such a case may occur in one of the following situations:

- the exemplary assignments of the DM do not match the additive model,
- the DM may have made an error in his/her statements; for example stating that $L^{DM}(a) > R^{DM}(b)$ while b dominates a (that is $g_i(b) \geq g_i(a)$),
- the statements provided by the DM are contradictory because his/her exemplary assignments are unstable, some hidden criteria are taken into account, ...

In such a case, the DM may want either to pursue the analysis with such an incompatibility or to identify its reasons in order to remove it and, therefore, to define a new set of exemplary assignments whose corresponding constraints E^{A^*ext} generate a non empty polyhedron. Let us consider below the two possible solutions.

6.1.1 Situation where incompatibility is accepted

If the DM wants to pursue the analysis with the incompatibility, he/she has to accept that some of his/her exemplary assignments of reference actions will not be reproduced by any value function. Note that, from a formal viewpoint, if the polyhedron generated by E^{A^*} is empty, then possible and necessary assignment $C_P(a)$ and $C_N(a)$ are meaningless for all $a \in A$. Thus, the acceptance of the inconsistency means that the DM does not change the exemplary assignments and computes $C_P(a)$ and $C_N(a)$ on a new set of constraints $(E^{A^*})'_{ext}$ differing from the original set $(E^{A^*})'$ by an additional constraint on the acceptable total error:

$$\left.\begin{array}{l} U(a^*) \geq U(b^*) + \varepsilon \;\Leftrightarrow\; L^{DM}(a^*) > R^{DM}(b^*),\ \forall a^*, b^* \in A^* \\ u_i(g_i(a_{\tau_i(j)})) - u_i(g_i(a_{\tau_i(j-1)})) \geq 0,\ i = 1, ..., n,\ j = 2, ..., m \\ u_i(g_i(a_{\tau_i(1)})) \geq 0, u_i(g_i(a_{\tau_i(m)})) \leq u_i(\beta_i),\ i = 1, ..., n, \\ u_i(\alpha_i) = 0,\ i = 1, ..., n \\ \sum_{i=1}^{n} u_i(\beta_i) = 1, \\ \varepsilon \geq \varepsilon^{ext} \end{array}\right\} (E^{A^*})'_{ext}$$

where $\varepsilon^{ext} < \varepsilon^*$, such that, remembering that $\varepsilon^* = max\ \varepsilon$ subject to set of constraints $(E^{A^*})'_{ext}$, we have that the resulting new set of constraints $(E^{A^*})'_{ext}$ is not empty.

On the basis of $(E^{A^R})'_{ext}$, for any pair $(a, b) \in A$, the set of constraints $E(a, b)_{ext}$ can be built as the union of the constraints $(E^{A^R})'_{ext}$ and the constraints relative to the breakpoints introduced by those actions a, b that possibly do not belong to A^R. Then preference relations $\succsim'^N$ and $\succsim'^P$ can be computed by minimizing and maximizing ε subject to $E'(a, b)$, rather than to $E(a, b)$, respectively.

Obviously, the necessary and possible rankings resulting from these computations will not fully respect the exemplary assignment provided by the DM, i.e. there is at least one couple $a^*, b^* \in A^*$ such that $L^{DM}(a^*) > R^{DM}(b^*)$ and nevertheless $U(a) \leq U(b)$.

Of course, also in this context indices $L_P^{\mathcal{U}}(a^*)$, $L_N^{\mathcal{U}}(a^*)$, $R_N^{\mathcal{U}}(a^*)$ and $R_P^{\mathcal{U}}(a^*)$ maintain all their main properties stated in proposition 4.2.

6.1.2 Situation where incompatibility is not accepted

If the DM does not want to pursue the analysis with the incompatibility, it is necessary to identify the troublesome exemplary assignments responsible for this incompatibility, so as to remove some of them. Remark that there may exist several sets of pairwise comparisons which, once removed, make set (E^{A^*}) of constraints non-empty. Hereafter, we outline the main steps of a procedure which identifies these sets.

Recall that the exemplary assignments of reference actions are represented in the ordinal regression constraints E^{A^*} by linear constraints. Hence, identifying the troublesome pairwise comparisons of reference actions amounts at finding a minimal subset of constraints that, once removed from E^{A^*}, leads to a set of constraints generating a non-empty polyhedron of compatible value functions. The identification procedure is to be performed iteratively since there may exist several minimal subsets of this kind.

Let associate with each couple of actions $a, b \in A^*$ such that $L^{DM}(a) > R^{DM}(b)$ a new binary variable $v_{a,b}$, i.e. $v_{a,b} = 0$ or $v_{a,b} = 1$. Using these binary variables, we rewrite the first constraint of set E^{A^*} as follows:

$$L^{DM}(a) > R^{DM}(b) \Leftrightarrow U(a) - U(b) + Mv_{a,b} > 0 \tag{25}$$

where $M > 1$. Remark that if $v_{a,b} = 1$, then the corresponding constraint is satisfied whatever the value function is, which is equivalent to elimination of this constraint. Therefore, identifying a minimal subset of troublesome pairwise comparisons can be performed by solving the following mixed 0-1 linear program:

$$
\begin{array}{rl}
Min \rightarrow & f = \sum_{a,b \in A^*:\, L^{DM}(a) > R^{DM}(b)} v_{a,b} \\
\text{s. t.} & \\
& \left\{ \begin{array}{l}
L^{DM}(a) > R^{DM}(b) \Leftrightarrow U(a) - U(b) + M v_{a,b} \geq \varepsilon,\ \forall a, b \in A^* \\
u_i(g_i(a_{\tau_i(j)})) - u_i(g_i(a_{\tau_i(j-1)})) \geq 0,\ i = 1, ..., n,\ j = 2, ..., m \\
u_i(g_i(a_{\tau_i(1)})) \geq 0, u_i(g_i(a_{\tau_i(m)})) \leq u_i(\beta_i),\ i = 1, ..., n, \\
u_i(\alpha_i) = 0,\ i = 1, ..., n \\
\sum_{i=1}^{n} u_i(\beta_i) = 1, \\
\varepsilon \geq \varepsilon'
\end{array} \right.
\end{array}
\tag{26}
$$

where ε' is a small positive value.

The optimal solution of (26) indicates one of the subsets of smallest cardinality being the cause of incompatibility. Alternative subsets of this kind can be found by solving (26) with additional constraint that forbids finding again the same solution. Let f^* be the optimal value of the objective function of (26) and $v^*_{a,b}$ the values of the binary variables at the optimum. Let also $S_1 = \{(a, b) \in A^* \times A^* : L^{DM}(a) > R^{DM}(b)$ and $v^*_{a,b} = 1\}$. The additional constraint has then the form

$$
\sum_{(a,b) \in S_1} v_{a,b} \leq f^* - 1 \tag{27}
$$

Continuing in this way, we can identify other subsets, possibly all of them. These subsets of pairwise comparisons are to be presented to the DM as alternative solutions for removing incompatibility. Such procedure has been described in [MDF+03]. Note however that when the number of alternatives in the reference set increases, solving 26 can become computationally difficult.

7 Specifying assignments with gradual confidence levels

The UTADIS$^{\text{GMS}}$ method presented in the previous section is intended to support the DM in an interactive process. Indeed, defining a large set of exemplary assignments of reference actions can be difficult for the DM. Therefore, one way to reduce the difficulty of this task would be to permit the DM an incremental specification of exemplary assignments. This way of proceeding allows the DM to control the evolution of the necessary and possible assignments.

Another way of reducing the difficulty of the task is to extend the UTADIS$^{\text{GMS}}$ method so as to account for different confidence levels given to exemplary assignments. Let $[L^{DM}_1(a^*), R^{DM}_1(a^*)] \supseteq [L^{DM}_2(a^*), R^{DM}_2(a^*)] \supseteq ... \supseteq [L^{DM}_s(a^*), R^{DM}_s(a^*)]$ be embedded sets of DM's exemplary assignments of reference action $a \in A^*$. To each set of exemplary assignments $[L^{DM}_t(a^*), R^{DM}_t(a^*)]$, $t = 1, \ldots s$, $a^* \in A^*$, corresponds a set of constraints $E^{A^*}_t$ generating a polyhedron of compatible value functions $P^{A^*}_t$. Polyhedrons $P^{A^*}_t$, $t = 1, \ldots s$, are embedded in the order of the exemplary assignments $[L^{DM}_t(a^*), R^{DM}_t(a^*)]$, $t = 1, \ldots s$, $a^* \in A^*$, i.e. $P^{A^*}_1 \supseteq P^{A^*}_2 \supseteq ... \supseteq P^{A^*}_s$.

We suppose that $P^{A^*}_s \neq \emptyset$ and, therefore, as $P^{A^*}_1 \supseteq P^{A^*}_2 \supseteq ... \supseteq P^{A^*}_s$, we have $P^{A^R}_t \neq \emptyset$, for all $t = 1, \ldots, s$. If $P^{A^R}_s = \emptyset$ we consider only exemplary assignments until $[L^{DM}_p(a^*), R^{DM}_p(a^*)], a^* \in A^*$ with $p = max\left\{t : P^{A^*}_t \neq \emptyset\right\}$.

Definition 7.1. *Given a set of exemplary assignments $[L^{DM}_t(a^*), R^{DM}_t(a^*)], t = 1, \ldots s, a^* \in A^*$, and corresponding sets $\mathcal{U}^t_{A^*}$ of compatible value functions of level t, for each $a \in A$, the following indices have to be considered:*

- *minimum possible class:* $L^{t,\mathcal{U}}_P(a) = Max\left\{L^{DM}(a^*) : \forall U \in \mathcal{U}^t_{A^*}, U(a^*) \leq U(a),\ a^* \in A^*\right\}$, (28)

- *minimum necessary class:*
$$L_N^{t,\mathcal{U}}(a) = Max\left\{L^{DM}(a^*) : \exists U \in \mathcal{U}_{A^*}^t \text{ for which } U(a^*) \leq U(a),\ a^* \in A^*\right\}, \tag{29}$$

- *maximum necessary class:*
$$R_N^{t,\mathcal{U}}(a) = Min\left\{R^{DM}(a^*) : \exists U \in \mathcal{U}_{A^*}^t \text{ for which } U(a) \leq U(a^*),\ a^* \in A^*\right\}, \tag{30}$$

- *maximum possible class:* $R_P^{t,\mathcal{U}}(a) = Min\left\{R^{DM}(a^*) : \forall U \in \mathcal{U}_{A^*}^t,\ U(a) \leq U(a^*),\ a^* \in A^*\right\}$ (31)

Definition 7.2. *Given a set of exemplary assignments* $[L_t^{DM}(a^*), R_t^{DM}(a^*)], t = 1, \ldots s, a^* \in A^*$, *and corresponding sets* $\mathcal{U}_{A^*}^t$ *of compatible value functions of level* t, *for each* $a \in A$, *we define the possible assignment of level* $t, t = 1, ..., s$, $C_P^t(a)$ *as the set of indices of class* C_h *for which there exist at least one value function* $U \in \mathcal{U}_{A^*}^t$ *assigning* a *to* C_h *and the necessary assignment* $C_N^t(a)$ *as set of indices of class* C_h *for which all value functions* $U \in \mathcal{U}_{A^*}^t$ *assign* a *to* C_h, *that is:*

$$C_P^t(a) = \{h \in H \text{ such that } \exists U \in \mathcal{U}_{A^*}^t \text{ for which } h \in [L^U(a), R^U(a)]\} \tag{32}$$
$$C_N^t(a) = \{h \in H \text{ such that } \forall U \in \mathcal{U}_{A^*}^t \text{ it holds } h \in [L^U(a), R^U(a)]\} \tag{33}$$

In order to compute possible and necessary assignments $C_P^t(a)$ and $C_N^t(a)$, $t = 1, \ldots, s$ we can use an example based assignment sorting as well as a threshold based sorting.

Using example based assignment sorting we can proceed as follows. For all $a, b \in A$, we say that there is a necessary weak preference relation of level t, denoted by $a \succsim_t^N b$ $(t = 1, ..., s)$, if for all value functions $U \in \mathcal{U}_{A^*}^t$, we have $U(a) \geq U(b)$. Analogously, for all $a, b \in A$, we say that there is a possible weak preference relation of level t, denoted by $x \succsim_t^P y$ $(t = 1, ..., s)$, if for at least one value function $U \in \mathcal{U}_{A^*}^t$, we have $U(x) \geq U(y)$.

In order to compute possible and necessary weak preference relations $\succsim_t^P$ and $\succsim_t^N$, we can proceed as follows. For all $x, y \in A$, set of constraints $E_t(x, y)$ can be obtained from set $E_t^{A^*}$ by adjoining the constraints relative to the breakpoints introduced by those actions a, b that possibly do not belong to A^*. For each $t = 1, \ldots, s$, and binary preference relations $\succsim_t^N$ and $\succsim_t^P$, we have

$$x \succsim_t^N y \Leftrightarrow \begin{cases} min & d_t(x, y) = Min\{U(x) - U(y)\} \geq 0 \\ & \text{s.t. set } E_t(x, y) \text{ of constraints} \end{cases} \tag{34}$$

and

$$x \succsim_t^P y \Leftrightarrow \begin{cases} max & D_t(x, y) = Min\{U(x) - U(y)\} \geq 0 \\ & \text{s.t. set } E_t(x, y) \text{ of constraints} \end{cases} \tag{35}$$

Each time we pass from $\succsim_{t-1}$ to $\succsim_t$, $t = 1, \ldots, s-1$, we add to $E_{t-1}^{A^*}$ and, consequently, to $E_{t-1}(a, b)$, new constraints concerning pairs $(a^*, b^*) \in A^* \times A^*$, such that $L_t^{DM}(a) > R_t^{DM}(b)$ but not $L_{t-1}^{DM}(a) > R_{t-1}^{DM}(b)$, thus the computations of $d_t(x, y)$ and $D_t(x, y)$, for all $x, y \in A \times A$ proceed iteratively.

Binary preference relations $\succsim_t^N$ and $\succsim_t^P$, $t = 1, \ldots, s$ satisfies the following properties [GMS08]:

Proposition 7.1. *It holds:*

- $\succsim_t^N \subseteq \succsim_t^P$,
- $\succsim_t^N$ *is a complete preorder (i.e. transitive and strongly complete),*
- $\succsim_t^P$ *is strongly complete and negatively transitive.*
- $\succsim_{t-1}^N \subseteq \succsim_t^N$ *and* $\succsim_t^P \supseteq \succsim_{t-1}^P$, $t = 2, \ldots, s$.

On the basis of binary preference relations $\succsim_t^N$ and $\succsim_t^P$, $t = 1, \ldots, s$ it is possible to calculate iteratively the indices (28)-(31) in the same way as in Section 5.

8 Illustrative example

In this section, a real world multiple criteria sorting problem will be presented followed by the application of the UTADIS^{GMS} software.

A transport company is about to classify 76 buses into 4 pre-defined and preference ordered classes C_1-C_4, such that C_1 will group the buses being in the worst technical state, needing a vary major revision, C_2 will group the buses being in the lower-intermediate technical state, needing a major revision, C_3 will group the buses being in the upper-intermediate technical state, needing a minor revision, and C_4 will group the buses being in the best technical state, needing no revision. The buses were evaluated according to a total of 8 quantitative criteria reflecting their performance and technical parameters. The names and types of the criteria are given in Table 1, and the performance matrix for a subset of buses is presented in Table 2 (see [ZS95]).

Table 1: Table of criteria

Code	Name	Type
g_1	Maximum speed	gain
g_2	Compression pressure	gain
g_3	Blacking	cost
g_4	Torque	gain
g_5	Summer fuel consumption	cost
g_6	Winter fuel consumption	cost
g_7	Oil consumption	cost
g_8	Horse power	gain

Table 2: Performance matrix

Bus	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8
a_1	90	2.52	38	481	21.8	26.4	0.7	145
a_2	76	2.11	70	420	22.0	25.5	2.7	110
a_3	63	1.98	82	400	22.0	24.8	3.7	101
a_4	90	2.48	49	477	21.9	25.1	1.0	138
a_5	85	2.45	52	460	21.8	25.2	1.4	130
a_6	72	2.20	73	425	23.1	27.4	2.8	112
a_7	88	2.50	50	480	21.6	24.7	1.1	140
a_8	87	2.48	56	465	22.8	27.6	1.4	135
a_9	90	2.56	16	486	26.5	27.3	0.2	150
a_{10}	60	1.95	95	400	23.3	24.8	4.4	96
.	.	.	.	.	.	.	.	.
a_{74}	87	2.48	52	465	21.9	24.6	1.4	135
a_{75}	86	2.50	55	456	22.0	25.1	1.5	130
a_{76}	88	2.52	46	472	21.8	23.8	1.1	141

The diagnostic expert, assisted by a decision analyst, is able to make a more or less precise assignment of few buses to the classes of technical state. The state of these buses, called reference buses, is relatively well known to the expert, so they will serve to provide assignment examples in successive iterations. In the first iteration, the expert provides a typical example for each one of the four classes, and makes, moreover, an imprecise assignment of two other reference buses to the two intermediate classes. This preference information is given in Table 3. For this initial preference information, the possible and necessary assignments have been computed using the UTADIS^{GMS}

method implemented on the *DecisionDeck* Platform (`http://decision-deck.sourceforge.net/`) as an open source software.

Table 3: Exemplary assignments of some reference buses in the first iteration

Reference bus	Lower class (L^{DM})	Upper class (R^{DM})
a_1	C_2	C_2
a_5	C_3	C_3
a_7	C_4	C_4
a_{28}	C_2	C_3
a_{40}	C_1	C_1
a_{42}	C_2	C_3

The resulting possible assignments review all possible consequences of the preference information on sorting of the whole set of buses (see Table 4). One can see that the inferred model has assigned all reference buses to their respective desired classes: a_1 is assigned to C_2, a_5 to C_3, etc. For most of non-reference buses the possible assignment is non-univocal. Precisely, we have 9 non-reference buses assigned to single class C_1 or C_4, 12 and 38 buses assigned, respectively, to the range of two and three contiguous classes, and 11 busses which can be possibly assigned to all four classes.

Table 4: Possible assignments in the first iteration

$L_P^{\mathcal{U}^1}$	$R_P^{\mathcal{U}^1}$	Assigned buses
C_1	C_1	a_6, a_{23}, a_{40}, a_{60}, a_{62}, a_{69}
C_1	C_2	a_8, a_{14}, a_{17}, a_{19}
C_1	C_3	a_2, a_{10}, a_{12}, a_{21}, a_{24}, a_{26}, a_{27}, a_{30}, a_{34}, a_{36}, a_{38}, a_{39}, a_{45}, a_{46}, a_{47}, a_{48}, a_{50}, a_{53}, a_{63}, a_{66}, a_{67}
C_1	C_4	a_3, a_9, a_{11}, a_{15}, a_{16}, a_{20}, a_{31}, a_{52}, a_{58}, a_{68}, a_{70}
C_2	C_2	a_1
C_2	C_3	a_{25}, a_{28}, a_{42}
C_2	C_4	a_4, a_{13}, a_{22}, a_{33}, a_{37}, a_{41}, a_{43}, a_{44}, a_{55}, a_{56}, a_{59}, a_{64}, a_{65}, a_{71}, a_{73}, a_{74}, a_{75}
C_3	C_3	a_5
C_3	C_4	a_{32}, a_{35}, a_{51}, a_{54}, a_{57}, a_{61}, a_{76}
C_4	C_4	a_7, a_{18}, a_{29}, a_{49}, a_{72}

In the first iteration, when the preference information is yet rather poor, the possible weak preference relation used to assess the necessary assignments is very rich, i.e., for most of pairs of buses (a_i, a_j) it is true $a_i \succsim^P a_j$, as well as $a_j \succsim^P a_i$. Therefore, in most cases, one can observe that $R_N^{\mathcal{U}^1}(a) < L_N^{\mathcal{U}^1}(a)$ and, consequently, $C_N^1(a) = \emptyset$. Table 5 summarizes the non-empty necessary assignments in the first iteration. The necessary assignment is non-empty for the 4 reference buses and for 9 non-reference ones assigned to the extreme classes. Remember that the method does not guarantee that all imprecise assignments of reference actions are repeated in the necessary assignments. For example, the range of necessary assignment for bus a_{28} is empty because for some compatible value functions it is assigned to C_2 and for some others to C_3, but there is no compatible value function for which the assignment of a_{28} to both classes would be feasible). Obviously, for every bus a_i it holds $C_N^1(a) \subseteq C_P^1(a)$.

The UTADIS$^{\text{GMS}}$ method is intended to be used interactively, so that the DM and the analyst could provide assignment examples incrementally, looking at consequences of the introduced pref-

Table 5: Necessary assignments in the first iteration

$L_N^{\mathcal{U}^1}$	$R_N^{\mathcal{U}^1}$	Assigned buses
C_1	C_1	a_6, a_{23}, a_{40}, a_{60}, a_{62}, a_{69}
C_2	C_2	a_1
C_3	C_3	a_5
C_4	C_4	a_7, a_{18}, a_{29}, a_{49}, a_{72}

erence information on the possible assignments of all actions. Let us suppose that considering the results of the first iteration, our diagnostic expert feels confident that a_{30}, a_{19}, and a_{35} should be assigned to class C_1, C_2 and C_3, respectively. It appears, however, that the introduced preference information is inconsistent, and no additive value function is compatible with the reference assignments. The method states that the newly introduced assignments $(a_{30} \to C_1)$ and $(a_{19} \to C_2)$ cannot be represented together by any additive value function. Then, the DM may want either to pursue the analysis with this incompatibility, or to remove it. Suppose that our DM will remove one of the two inconsistent exemplary assignments, precisely the one concerning a_{19}. In consequence, the system becomes consistent, and the corresponding results are presented in Table 7.

Table 6: Additional exemplary assignments in the second iteration

Reference bus	Lower class (L^{DM})	Upper class (R^{DM})
a_{30}	C_1	C_1
a_{35}	C_3	C_3

One can observe that the assignments are more tight with the growth of the preference information. There are 10 non-reference buses, for which the possible assignments have changed with respect to the first iteration (those buses are marked in bold in the table). For example, a_{11} and a_{41} are possibly assigned to classes C_1, C_2, and C_3, but not to C_4, as previously. As far as necessary assignments are concerned, there are 5 additional buses assigned to the same class by all compatible value functions: reference bus a_{35} and 4 non-reference buses have been assigned to class C_1.

In the third iteration, the diagnostic expert provides two additional exemplary assignments (see Table 8). As a consequence, 4 non-reference buses got a tighter range of possible assignments (those buses are marked in bold in the table). Moreover, two of them are necessarily assigned to a non-empty range of classes which they were not in the previous iteration.

The illustration of the use of the UTADIS$^{\text{GMS}}$ method will be stopped at this stage. The DM may be satisfied with the results, or may want to pursue the iterative process, either by adding some new assignments of reference buses or by revising the previous judgments, which would result in new possible and necessary assignments.

9 Conclusion

In this paper we presented a new multiple criteria method called UTADIS$^{\text{GMS}}$ which sorts actions into ordered class. The underlying preference model is a set of monotone non decreasing additive value functions. The method applies to the case in which the classes are explicitly defined by thresholds on the value that separated consecutive classes, and to the case where classes are implicitly defined through the sorting examples. In order to specify the sorting model, the DM is asked to provide

Table 7: Possible and necessary assignments in the second iteration

$L_P^{\mathcal{U}^2}$	$R_P^{\mathcal{U}^2}$	Assigned buses
C_1	C_1	a_6, $\boldsymbol{a_{19}}$, a_{23}, $\boldsymbol{a_{30}}$, a_{40}, $\boldsymbol{a_{47}}$, a_{60}, a_{62}, $\boldsymbol{a_{63}}$, a_{69}
C_1	C_2	a_8, a_{14}, a_{17}
C_1	C_3	a_2, a_{10}, $\boldsymbol{a_{11}}$, a_{12}, $\boldsymbol{a_{15}}$, $\boldsymbol{a_{20}}$, a_{21}, a_{24}, a_{26}, a_{27}, $\boldsymbol{a_{31}}$, a_{34}, a_{36}, a_{38}, a_{39}, a_{45}, a_{46}, a_{48}, a_{50}, a_{53}, a_{66}, a_{67}, $\boldsymbol{a_{70}}$
C_1	C_4	a_3, a_9, a_{16}, a_{52}, a_{58}, a_{68}
C_2	C_2	a_1
C_2	C_3	a_{25}, a_{28}, $\boldsymbol{a_{41}}$, a_{42}, $\boldsymbol{a_{43}}$, $\boldsymbol{a_{64}}$
C_2	C_4	a_4, a_{13}, a_{22}, a_{33}, a_{37}, a_{44}, a_{55}, a_{56}, a_{59}, a_{65}, a_{71}, a_{73}, a_{74}, a_{75}
C_3	C_3	a_5, $\boldsymbol{a_{35}}$
C_3	C_4	a_{32}, a_{51}, a_{54}, a_{57}, a_{61}, a_{76}
C_4	C_4	a_7, a_{18}, a_{29}, a_{49}, a_{72}

$L_N^{\mathcal{U}^2}$	$R_N^{\mathcal{U}^2}$	Assigned buses
C_1	C_1	a_6, $\boldsymbol{a_{19}}$, a_{23}, $\boldsymbol{a_{30}}$, a_{40}, $\boldsymbol{a_{47}}$, a_{60}, a_{62}, $\boldsymbol{a_{63}}$, a_{69}
C_2	C_2	a_1
C_3	C_3	a_5, $\boldsymbol{a_{35}}$
C_4	C_4	a_{a7}, a_{18}, a_{29}, a_{49}, a_{72}

Table 8: Additional exemplary assignments in the third iteration

Reference bus	Lower class (L^{DM})	Upper class (R^{DM})
a_{26}	C_2	C_2
a_{74}	C_4	C_4

assignment examples (desired assignment for specific actions) which yield a set of compatible value function.

In reference to the set of sorting examples, the method provides, for each action a, two intervals of classes: the possible assignment corresponds to the interval of classes to which the action can be assigned considering any compatible value functions individually, while the necessary assignment corresponds to the interval of classes to which the action a can be assigned considering all compatible value functions simultaneously. The necessary and possible assignments are computed through the resolution of linear programs.

This new method provides new capabilities as compared to existing value based sorting methods in several ways: firstly, it consider possible and necessary assignments which no previous method did consider. Second, UTADIS$^{\text{GMS}}$ make use of a very general and flexible preference model as it considers all non-decreasing marginal value functions (rather than piece-wise linear marginal value function). Third, it considers, when computing assignments all value functions compatible with the assignment examples provided by the DM (and not only a representative one). Lastly, the method makes it possible to account for both the threshold based and example based sorting procedures. We envisage the following possible developments for future research:

- consideration of other type of preference information such as intensity of preferences and trade-

Table 9: Possible and necessary assignments in the third iteration

$L_P^{\mathcal{U}^3}$	$R_P^{\mathcal{U}^3}$	Assigned buses
C_1	C_1	a_6, a_{19}, a_{23}, a_{30}, a_{40}, a_{47}, a_{60}, a_{62}, a_{63}, a_{69}
C_1	C_2	a_8, a_{14}, a_{17}, $\boldsymbol{a_{27}}$
C_1	C_3	a_2, a_{10}, a_{11}, a_{12}, a_{15}, a_{20}, a_{21}, a_{24}, a_{34} a_{36}, a_{38}, a_{39}, a_{45}, a_{46}, a_{48}, a_{50}, a_{53}, a_{66}, a_{67}, a_{70}
C_1	C_4	a_3, a_9, a_{16}, a_{52}, a_{58}, a_{68}
C_2	C_2	a_1, $\boldsymbol{a_{26}}$
C_2	C_3	a_{25}, a_{28}, $\boldsymbol{a_{31}}$, a_{41}, a_{42}, a_{43}, a_{64}
C_2	C_4	a_4, a_{13}, a_{22}, a_{33}, a_{37}, a_{44}, a_{55}, a_{56}, a_{59}, a_{65}, a_{71}, a_{73}, a_{75}
C_3	C_3	a_5, a_{35}
C_3	C_4	a_{32}, a_{51}, a_{54}, a_{61}
C_4	C_4	a_7, a_{18}, a_{29}, a_{49}, $\boldsymbol{a_{57}}$, a_{72}, $\boldsymbol{a_{74}}$, $\boldsymbol{a_{76}}$

$L_N^{\mathcal{U}^3}$	$R_N^{\mathcal{U}^3}$	Assigned alternatives
C_1	C_1	a_6, a_{19}, a_{23}, a_{30}, a_{40}, a_{47}, a_{60}, a_{62}, a_{63}, a_{69}
C_2	C_2	a_1, $\boldsymbol{a_{26}}$
C_3	C_3	a_5, a_{35}
C_4	C_4	a_7, a_{18}, a_{29}, a_{49}, $\boldsymbol{a_{57}}$, a_{72}, $\boldsymbol{a_{74}}$, $\boldsymbol{a_{76}}$

offs,

- extension of the approach to group decision,
- application of this methodology to interactive optimization problems,
- application of the methodology to some classical classification problems such as bankruptcy risk or global risk evaluation and comparison with classical ordinal regression methodologies (UTADIS, MHDIS, etc.) and other competitive methodologies such as discriminant analysis, neural nets or support vector machine.
- improve the interactive process induced by the methodology, to support the DM in the elaboration of robust recommendations

Acknowledgements : The authors acknowledge the support of the COST Action IC0602 "Algorithmic Decision Theory", `www.algodec.org`. The authors also wish to thank Miłosz Kadziński who implemented the UTADIS$^{\mathbf{GMS}}$ method on the *DecisionDeck* Platform and run the calculations of the illustrative example.

References

[DGJL80] J.M. Devaud, G. Groussaud, and E. Jacquet-Lagreze. UTADIS: Une methode de construction de fonctions d'utilite additives rendant compte de jugements globaux. In *European working group on MCDA, Bochum , Germany*, 1980.

[FGS09] J. Figueira, S. Greco, and R. Slowinski. Building a set of additive value functions representing a reference preorder and intensities of preference: GRIP method. *European Journal of Operational Research*, 195(2):460–486, June 2009.

[GMS08] S. Greco, V. Mousseau, and R. Slowinski. Ordinal regression revisited: multiple criteria ranking using a set of additive value functions. *European Journal of Operational Research*, 191(2):415–435, December 2008.

[Jac95] E. Jacquet-Lagrèze. An application of the UTA discriminant model for the evaluation of sc R&D projects. In P. Pardalos, Y. Siskos, and C. Zopounidis, editors, *Advances in Multicriteria Analysis*, Nonconvex Optimization and its Applications, pages 203–211. Kluwer Academic, Dordrecht, 1995.

[KBO09] M. Koksalan and S. Bilgin Ozpeynirci. An interactive sorting method for additive utility functions. *Computers & Operations Research*, 36(9):2565–2572, September 2009.

[KR93] R.L. Keeney and H. Raiffa. *Decisions with multiple objectives: Preferences and value tradeoffs*. J. Wiley, New York, 1993. second edition.

[KU03] Murat Koksalan and Canan Ulu. An interactive approach for placing alternatives in preference classes. *European Journal of Operational Research*, 144:429–439, January 2003.

[MDF+03] 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, 2003.

[Roy96] B. Roy. *Multicriteria Methodology for Decision Aiding*. Kluwer Academic, Dordrecht, 1996.

[ZD00a] C. Zopounidis and M. Doumpos. Building additive utilities for multi-group hierarchical discrimination: The MHDIS method. *Optimization Methods & Software*, 14(3):219–240, 2000.

[ZD00b] C. Zopounidis and M. Doumpos. PREFDIS: a multicriteria decision support system for sorting decision problems. *Computers & Operations Research*, 27(7-8):779–797, June 2000.

[ZD02] C. Zopounidis and M. Doumpos. Multicriteria classification and sorting methods: A literature review. *European Journal of Operational Research*, 138(2):229–246, April 2002.

[ZS95] J. Zak and J. Stefanowski. Determining maintenance activities of motor vehicles using rough sets approach. In *Proceedings of the European Conference EUROMAINTENANCE ' 94*, pages 39–44. Amsterdam, 1995.

Appendices

A Proofs of section 3

A.1 Proof of Proposition 3.1

Proposition 3.1 *For any action $a \in A \setminus A^*$, the example-based sorting procedure provides an assignment $a \rightarrow [C_{L^U(a)}, C_{R^U(a)}]$, such that $L^U(a) \leq R^U(a)$.*

Proof. Let us suppose that :

$$L^U(a) > R^U(a) \tag{36}$$

Consider the sets $A^{*+} \subset A^*$ and $A^{*-} \subset A^*$ defined as $A^{*-} = \{a^* \in A^* : U(a^*) \leq U(a)\}$ and $A^{*+} = \{a^* \in A^* : U(a^*) \geq U(a)\}$. (36) mean that there exists $a^{*-} \in A^{*-}$ and $a^{*+} \in A^{*+}$ such that $L^{DM}(a^{*-}) = L^U(a)$, $R^{DM}(a^{*+}) = R^U(a)$ and $L^{DM}(a^{*-}) > R^{DM}(a^{*+})$. As the set of assignment examples is supposed to be consistent with U, $L^{DM}(a^{*-}) > R^{DM}(a^{*+})$ implies that $U(a^{*-}) > U(a^{*+})$. However, as $a^{*+} \in A^{*+}$ and $a^{*-} \in A^{*-}$, it holds that $U(a^{*-}) \leq U(a^{*+})$, which contradicts the hypothesis and concludes the proof. □

A.2 Proof of Proposition 3.2

Proposition 3.2 *The example-based sorting procedure assigns each reference action $a^* \in A^*$ to an interval of classes $[C_{L^U(a^*)}, C_{R^U(a^*)}]$, such that:*

$$L^U(a^*) \geq L^{DM}(a^*) \tag{37}$$
$$R^U(a^*) \leq R^{DM}(a^*) \tag{38}$$

Proof. According to (5) and (6) :

$$L^U(a^*) = Max\left\{L^{DM}(a^{*\prime}) : U(a^{*\prime}) \leq U(a^*),\ a^{*\prime} \in A^*\right\} \tag{39}$$
$$R^U(a^*) = Min\left\{R^{DM}(a^{*\prime}) : U(a^{*\prime}) \geq U(a^*),\ a^{*\prime} \in A^*\right\} \tag{40}$$

Since $U(a^*) \leq U(a^*)$, then $L^{DM}(a^*) \in \{L^{DM}(a^{*\prime}) : U(a^{*\prime}) \leq U(a^*),\ a^{*\prime} \in A^*\}$ and, therefore, $Max\left\{L^{DM}(a^{*\prime}) : U(a^{*\prime}) \leq U(a^*),\ a^{*\prime} \in A^*\right\} \geq L^{DM}(a^*)$, i.e. $L^U(a^*) \geq L^{DM}(a^*)$. Analogous proof holds for $R^U(a^*) \leq R^{DM}(a^*)$. □

A.3 Proof of Proposition 3.3

Proposition 3.3 *Consider the case where the DM provides precise assignment examples only, i.e., $L^{DM}(a^*) = R^{DM}(a^*)$, for each $a^* \in A^*$. Assuming the use of a single value function U in the example-based sorting procedure, if we choose, for each $h = 1, ..., p-1$, the threshold b_h^U in the interval $]Max_{a^* \rightarrow C_h}\{U(a^*)\}, Min_{a^* \rightarrow C_{h+1}}\{U(a^*)\}]$, we obtain a threshold-based sorting procedure that restores the assignment examples and assigns each non-reference action $a \in A \setminus A^*$ to a single class in the interval $[C_{L^U(a)}, C_{R^U(a)}]$ stemming from the example-based sorting procedure.*

Proof. As we suppose the assignment examples to be consistent with U, we have $\forall a^*, b^* \in A^*$, $U(a^*) \geq U(b^*) \Rightarrow R^{DM}(a^*) \geq L^{DM}(b^*)$. Moreover, since all assignment examples are precise ($L^{DM}(a^*) = R^{DM}(a^*)$), we will use $L^{DM}(a^*)$ to refer to the category to which the DM wishes to assign a^*. Therefore, the fact that assignment examples are consistent with U can be reformulated as $\forall a^*, b^* \in A^*, U(a^*) \geq U(b^*) \Rightarrow L^{DM}(a^*) \geq L^{DM}(b^*)$. If we choose the threshold values b_h^U in the interval $]Max_{a^* \to C_h}\{U(a^*)\}, Min_{a^* \to C_{h+1}}\{U(a^*)\}]$, for each $h = 1, ..., p-1$, then any $a \in A$ for which $U(a) \in [b_{h-1}^U, b_h^U[$ will be assigned by the threshold-based sorting procedure to category C_h (see Definition 3.1). As a consequence, the assignment examples will be restored by the threshold-based sorting procedure.

Considering a non-reference action $a \in A \setminus A^*$, we distinguish two cases:

i) $\exists h \in \{1, ..., p\}$ such that $U(a) \in [Min_{a^* \to C_h}\{U(a^*)\}, Max_{a^* \to C_h}\{U(a^*)\}]$,

ii) $\exists h \in \{1, ..., p-1\}$ such that $U(a) \in]Max_{a^* \to C_h}\{U(a^*)\}, Min_{a^* \to C_{h+1}}\{U(a^*)\}[$.

case i):
Considering Definition 3.3, we have $L^U(a) = R^U(a) = h$, i.e., a is assigned to class C_h by the example-based sorting procedure. In this case, the interval $[C_{L^U(a)}, C_{R^U(a)}]$ boils down to C_h.
Moreover, since $U(a) \in [Min_{a^* \to C_h}\{U(a^*)\}, Max_{a^* \to C_h}\{U(a^*)\}]$, a will be assigned to class C_h by the threshold-based sorting procedure, whatever the value of the thresholds
$b_{h-1} \in]Max_{a^* \to C_{h-1}}\{U(a^*)\}, Min_{a^* \to C_h}\{U(a^*)\}]$ and $b_h \in]Max_{a^* \to C_h}\{U(a^*)\}, Min_{a^* \to C_{h+1}}\{U(a^*)\}]$.

case ii):
Considering Definition 3.3, we have $L^U(a) = h$ and $R^U(a) = h+1$, i.e. a is assigned imprecisely to interval of classes $[C_h, C_{h+1}]$ by the example-based sorting procedure.
Moreover, since $U(a) \in]Max_{a^* \to C_h}\{U(a^*)\}, Min_{a^* \to C_{h+1}}\{U(a^*)\}[$, a will be assigned by the threshold-based sorting procedure to class C_{h+1} if $b_h \in]Max_{a^* \to C_h}\{U(a^*)\}, U(a)]$, or to class C_h if $b_h \in]U(a), Min_{a^* \to C_{h+1}}\{U(a^*)\}]$. □

A.4 Proof of Proposition 3.4

Proposition 3.4 *Consider the case where the DM provides possibly imprecise assignment examples, i.e., $L^{DM}(a^*) \leq R^{DM}(a^*)$ for each $a^* \in A^*$. Assuming the use of a single value function U in the example-based sorting procedure, if we choose, for each $h = 1, ..., p-1$, the threshold b_h^U in the interval $I(b_h^U) =]Max_{a^*: R^{DM}(a^*) \leq h}\{U(a^*)\}, Min_{a^*: L^{DM}(a^*) > h}\{U(a^*)\}[$, with $b_h^U < b_{h+1}^U$ we obtain a threshold-based sorting procedure that assigns each reference action $a^* \in A^*$ to a single class in the interval $[C_{L^{DM}(a^*)}, C_{R^{DM}(a^*)}]$, and assigns each non-reference action $a \in A \setminus A^*$ to a single class in the interval $[C_{L^U(a)}, C_{R^U(a)}]$ stemming from the example-based sorting procedure.*

Proof. The proof will consist of the five following points:

1. The set of possible values for b_h, $I(b_h^U)$, is a non-empty interval.

2. If $b_h^U \leq Max_{a^*: R^{DM}(a^*) \leq h}\{U(a^*)\}$, then the threshold-based sorting procedure assigns each $a^{*\prime} \in A^*$ such that $b_h^U \leq U(a^{*\prime}) \leq Max_{a^*: R^{DM}(a^*) \leq h}\{U(a^*)\}$ inconsistently with the statement of the DM, in the sense that $a^{*\prime}$ is assigned to a class C_k with $k > R^{DM}(a^{*\prime})$.

3. If $b_h^U > Min_{a^*: L^{DM}(a^*) > h}\{U(a^*)\}$, then the threshold-based sorting procedure assigns each $a^{*\prime} \in A^*$ such that $Min_{a^*: L^{DM}(a^*) > h}\{U(a^*)\} \leq U(a^{*\prime}) < b_h^U$ inconsistently with the statement of the DM, in the sense that $a^{*\prime}$ is assigned to a class C_k with $k < L^{DM}(a^{*\prime})$.

4. If $b_h^U \in I(b_h^U)$, $h = 1, ..., p-1$, then the threshold-based sorting procedure assigns each $a^* \in A^*$ to a single class C_k in the interval $[C_{L^{DM}(a^*)}, C_{R^{DM}(a^*)}]$, and each $a \in A \setminus A^*$ in a single class C_k in the interval $[C_{L^U(a)}, C_{R^U(a)}]$.

5. For each each $a^* \in A^*$, and for each $C_k \in [C_{L^{DM}(a^*)}, C_{R^{DM}(a^*)}]$, there exist thresholds $b_h^U \in I(b_h^U)$, $h = 1, ..., p-1$, such that the threshold-based sorting procedure assigns a^* to C_k; for each each $a \in A \setminus A^*$, and for each $C_k \in [C_{L^U(a)}, C_{R^U(a)}]$, there exist thresholds $b_h^U \in I(b_h^U)$, $h = 1, ..., p-1$, such that the threshold-based sorting procedure assigns a to C_k.

Proof of 1.
Consider $b^*, c^* \in A^*$ such that $U(b^*) = Max_{a^*:R^{DM}(a^*)\leq h}\{U(a^*)\}$ and $U(c^*) = Min_{a^*:L^{DM}(a^*)>h}\{U(a^*)\}$. Therefore, we have $L^{DM}(c^*) > h \geq R^{DM}(b^*)$. As we suppose the assignment examples provided by the DM are consistent with U (see (4)), we have $U(b^*) < U(c^*)$, i.e.,
$Max_{a^*:R^{DM}(a^*)\leq h}\{U(a^*)\} < Min_{a^*:L^{DM}(a^*)>h}\{U(a^*)\}$.

Proof of 2.
Consider $a^{*\prime} \in A^*$ such that $b_h^U \leq U(a^{*\prime}) \leq Max_{a^*:R^{DM}(a^*)\leq h}\{U(a^*)\}$.
Since $b_h^U \leq U(a^{*\prime})$, according to Definition 3.1, $a^{*\prime}$ should be assigned to class C_k with $k > h$.
As $U(a^{*\prime}) \leq Max_{a^*:R^{DM}(a^*)\leq h}\{U(a^*)\}$, we have $R^{DM}(a^{*\prime}) \leq h$.
Therefore, the threshold-based sorting procedure assigns $a^{*\prime}$ inconsistently with the DM's statement.

Proof of 3.
Consider $a^{*\prime} \in A^*$ such that $Min_{a^*:L^{DM}(a^*)>h}\{U(a^*)\} \leq U(a^{*\prime}) < b_h^U$.
Since $U(a^{*\prime}) < b_h^U$, according to Definition 3.1, $a^{*\prime}$ should be assigned to class C_k with $k \leq h$.
As $Min_{a^*:L^{DM}(a^*)>h}\{U(a^*)\} \leq U(a^{*\prime})$, we have $L^{DM}(a^{*\prime}) > h$.
Therefore, the threshold-based sorting procedure assigns $a^{*\prime}$ inconsistently with the DM's statement.

Proof of 4.
Consider $a^{*\prime} \in A^*$ such that $U(a^{*\prime}) \in [b_{k-1}^U, b_k^U[$ and, therefore, for Definition 3.1, $a^{*\prime} \to C_k$. Since, by hypothesis,

$$b_{k-1}^U \in]Max_{a^*:R^{DM}(a^*)\leq k-1}\{U(a^*)\}, Min_{a^*:L^{DM}(a^*)>k-1}\{U(a^*)\}]$$

and

$$b_k^U \in]Max_{a^*:R^{DM}(a^*)\leq k}\{U(a^*)\}, Min_{a^*:L^{DM}(a^*)>k}\{U(a^*)\}],$$

we have

$$Max_{a^*:R^{DM}(a^*)\leq k-1}\{U(a^*)\} < b_{k-1}^U \leq U(a^{*\prime}) < b_k^U \leq Min_{a^*:L^{DM}(a^*)>k}\{U(a^*)\}.$$

From

$$Max_{a^*:R^{DM}(a^*)\leq k-1}\{U(a^*)\} < U(a^{*\prime})$$

we get $R^{DM}(a^{*\prime}) \geq k$, while from

$$U(a^{*\prime}) < Min_{a^*:L^{DM}(a^*)>k}\{U(a^*)\}$$

we get $L^{DM}(a^{*\prime}) \leq k$. Thus, $C_k \in [C_{L^{DM}(a^{*\prime})}, C_{R^{DM}(a^{*\prime})}]$.

Analogously, for $a \in A \setminus A^*$ for which $U(a) \in [b_{k-1}^U, b_k^U[$ and, consequently, $a \to C_k$, we have $C_k \in [C_{L^U(a)}, C_{R^U(a)}]$.

Proof of 5.
Consider $a^{*\prime} \in A^*$ such that $C_k \in [C_{L^{DM}(a^{*\prime})}, C_{R^{DM}(a^{*\prime})}]$.

Let us fix the thresholds b_h^U, $h = 1, ..., p-1$, such that:

- $b_h^U < U(a^{*\prime})$ for each $h < k$
- $b_h^U \geq U(a^{*\prime})$ for each $h \geq k$

with $b_h^U < b_{h+1}^U$.
For the above thresholds, $a^{*\prime}$ is assigned by the threshold-based procedure to class C_k. Hence, we need to prove that there always exists a set of thresholds defined as above which verify

$$b_h^U \in I(b_h^U) \ =]Max_{a^*:R^{DM}(a^*)\leq h}\{U(a^*)\}, Min_{a^*:L^{DM}(a^*)>h}\{U(a^*)\}[, h = 1, ..., p-1. \quad (i)$$

Consider $h < k$. For (i) we have

$$U(a^{*\prime}) > b_h^U > Max_{a^*:R^{DM}(a^*)\leq h}\{U(a^*)\}$$

which holds because $R^{DM}(a^{*\prime}) \geq k > h$.

Consider $h \geq k$. For (i) we have

$$U(a^{*\prime}) \leq b_h^U < Min_{a^*:L^{DM}(a^*)>h}\{U(a^*)\}$$

which holds because $L^{DM}(a^{*\prime}) \leq k \leq h$. □

B Proofs of section 4

B.1 Proof of Proposition 4.2

Proposition 4.2 *Given a set A^* of assignment examples and a corresponding set $\mathcal{U}_{A^*}$ of compatible value functions, it holds, for each $a \in A$,*

- $L_P^{\mathcal{U}}(a) \leq L_N^{\mathcal{U}}(a)$
- $R_N^{\mathcal{U}}(a) \leq R_P^{\mathcal{U}}(a)$
- $L_P^{\mathcal{U}}(a) \leq R_P^{\mathcal{U}}(a)$

Moreover, for any $a \in A$, we have:

- $C_P(a) \subseteq [L_P^{\mathcal{U}}(a), R_P^{\mathcal{U}}(a)]$,
- *if* $L_N^{\mathcal{U}}(a) \leq R_N^{\mathcal{U}}(a)$, *then* $C_N(a) = [L_N^{\mathcal{U}}(a), R_N^{\mathcal{U}}(a)]$,
- *if* $L_N^{\mathcal{U}}(a) > R_N^{\mathcal{U}}(a)$, *then* $C_N(a) = \emptyset$.

Proof. Consider
$$\begin{cases} A_{all}^{*-}(a) = \{a^* \in A^* : \forall U \in \mathcal{U}_{A^*}, U(a^*) \leq U(a)\}, \\ A_{one}^{*-}(a) = \{a^* \in A^* : \exists U \in \mathcal{U}_{A^*} \text{ for which } U(a^*) \leq U(a)\}, \\ A_{all}^{*+}(a) = \{a^* \in A^* : \forall U \in \mathcal{U}_{A^*}, U(a^*) \geq U(a)\}, \\ A_{one}^{*+}(a) = \{a^* \in A^* : \exists U \in \mathcal{U}_{A^*} \text{ for which } U(a^*) \geq U(a)\}. \end{cases} \quad \forall a \in A$$

Observe that $A_{all}^{*-}(a) \subseteq A_{one}^{*-}(a)$ from which we get:

$$\begin{aligned} L_P^{\mathcal{U}}(a) &= Max_{a^* \in A^*} \left\{ L^{DM}(a^*) : \forall U \in \mathcal{U}_{A^*}, U(a^*) \leq U(a) \right\} \\ &\leq Max_{a^* \in A^*} \left\{ L^{DM}(a^*) : \exists U \in \mathcal{U}_{A^*} \text{ for which } U(a^*) \leq U(a) \right\} = L_N^{\mathcal{U}}(a) \end{aligned}$$

Analogously, from $A_{all}^{*+}(a) \subseteq A_{one}^{*+}(a)$, we get:

$$\begin{aligned} R_N^{\mathcal{U}}(a) &= min_{a^* \in A^*} \left\{ R^{DM}(a^*) : \exists U \in \mathcal{U}_{A^*} \text{ for which } U(a) \leq U(a^*) \right\} \\ &\geq min_{a^* \in A^*} \left\{ R^{DM}(a^*) : \forall U \in \mathcal{U}_{A^*} \text{ for which } U(a) \leq U(a^*) \right\} = R_P^{\mathcal{U}}(a) \end{aligned}$$

For all $U \in \mathcal{U}_{A^*}$ and for all $a^{*-} \in A_{all}^{*-}(a)$ and $a^{*+} \in A_{all}^{*+}(a)$ we have $U(a^{*-}) \leq U(a) \leq U(a^{*+})$ which implies $L^{DM}(a^{*-}) \leq R^{DM}(a^{*+})$. Therefore

$$max\{L^{DM}(a^*) : a^* \in A^{*-}(a)\} \leq min\{R^{DM}(a^*) : a^* \in A^{*+}(a)\}. \tag{41}$$

From the definition of $L_P^{\mathcal{U}}(a)$ and $R_P^{\mathcal{U}}(a)$, we get

$$L_P^{\mathcal{U}}(a) = min\{L^{DM}(a^*) : a^* \in A^{*-}(a)\}$$

and

$$R_P^{\mathcal{U}}(a) = max\{R^{DM}(a^*) : a^* \in A^{*+}(a)\},$$

such that (41) gives

$$L_P^{\mathcal{U}}(a) \leq R_P^{\mathcal{U}}(a).$$

Observe that for any $a \in A$ and for any $U \in \mathcal{U}_{A^*}$, $L^U(a) \geq L_P^{\mathcal{U}}(a)$ and $R^U(a) \leq R_P^{\mathcal{U}}(a)$, i.e.

$$[L^U(a), R^U(a)] \subseteq [L_P^{\mathcal{U}}(a), R_P^{\mathcal{U}}(a)]. \tag{42}$$

For (11), (42) gives

$$C_P(a) \subseteq [L_P^{\mathcal{U}}(a), R_P^{\mathcal{U}}(a)].$$

Observe that for any $a \in A$,

$$L_N^{\mathcal{U}}(a) = max\left\{L^U(a) : U \in \mathcal{U}_{A^*}\right\}$$

and

$$R_N^{\mathcal{U}}(a) = min\left\{R^U(a) : U \in \mathcal{U}_{A^*}\right\}.$$

For (11), if $L_N^{\mathcal{U}}(a) \leq R_N^{\mathcal{U}}(a)$, then $C_N(a) = [L_N^{\mathcal{U}}(a), R_N^{\mathcal{U}}(a)]$. Instead, again for (11), if $L_N^{\mathcal{U}}(a) > R_N^{\mathcal{U}}(a)$, then $C_N(a) = \emptyset$. □

B.2 Proof of Proposition 4.3

Proposition 4.3 *$C_N(a) \neq \emptyset$ if and only if the following strict continuity condition is satisfied: for all $a^*, b^* \in A^*$ such that $L^{DM}(a^*) > R^{DM}(b^*)$ there are no two compatible value functions $U, U' \in \mathcal{U}_{A^*}$ for which $U(a) \geq U(a^*)$ and $U'(b^*) \geq U'(a)$.*

Proof. Let us observe that if for $a \in A$, $C_N(a) = \emptyset$, then there exist $U, U' \in \mathcal{U}_{A^*}$ such that

$$[L^U(a), R^U(a)] \cap [L^{U'}(a), R^{U'}(a)] = \emptyset. \tag{43}$$

Supposing without loss of the generality that $L^U(a) > R^{U'}(a)$, (43) is equivalent to the fact that there exist $a^*, b^* \in A^*$ for which $L^{DM}(a^*) > R^{DM}(b^*)$, $U(a) \geq U(a^*)$ and $U'(b^*) \geq U'(a)$, such that

$$L^U(a) = max\{L^{DM}(c^*) : U(c^*) \leq U(a), c^* \in A^*\} \leq L^{DM}(a^*)$$

$$>$$

$$R^{DM}(b^*) \leq min\{R^{DM}(c^*) : U'(c^*) \geq U'(a), c^* \in A^*\} = R^{U'}(a).$$

Thus, there do not exist $U, U' \in \mathcal{U}_{A^*}$ for which $U(a) \geq U(a^*)$ and $U'(b^*) \geq U'(a)$, i.e. strict continuity holds, if and only if for all $U, U' \in \mathcal{U}_{A^*}$, $[L^U(a), R^U(a)] \cap [L^{U'}(a), R^{U'}(a)] \neq \emptyset$, and consequently $C_N(a) \neq \emptyset$. □

B.3 Proof of Proposition 4.4

Proposition 4.4 *The no jump property (12) holds, and therefore $C_P(a) = [L_P^U, R_P^U]$, if and only if the weak continuity (4.4) holds.*

Proof. Let us start by proving that weak continuity (4.4) is necessary for the no jump property (12). For contradiction, let us suppose that weak continuity does not hold and therefore that:

a) there exist $a^*, b^* \in A^*$ such that $L^{DM}(a^*) > R^{DM}(b^*)$, and $U', U'' \in \mathcal{U}_{A^*}$ for which $U'(a) \geq U'(a^*)$ and $U''(b^*) \geq U''(a)$, but

b) there does not exist $U''' \in \mathcal{U}_{A^*}$ for which $U'''(a^*) \geq U'''(a) \geq U'''(b^*)$.

Consider

$$\begin{cases} \mathcal{U}^{+}_{A^*}(a,a^*) = \{U \in \mathcal{U}_{A^*} : \text{ such that } U(a) \geq U(a^*)\} \\ \mathcal{U}^{-}_{A^*}(a,b^*) = \{U \in \mathcal{U}_{A^*} : \text{ such that } U(b^*) \geq U(a)\} \end{cases}$$

Since the set of assignment examples is consistent with all value functions $U \in \mathcal{U}_{A^*}$ according to (4), $L^{DM}(a^*) > R^{DM}(b^*)$ implies that for all $U \in \mathcal{U}_{A^*}$ we have $U(a^*) > U(b^*)$. Therefore b) implies that for all $U \in \mathcal{U}_{A^*}$, $U(a) \geq U(a^*) > U(b^*)$ or $U(a^*) > U(b^*) \geq U(a)$. This means that $\mathcal{U}^{+}_{A^*}(a,a^*) \cup \mathcal{U}^{-}_{A^*}(a,b^*) = \mathcal{U}_{A^*}$ and $\mathcal{U}^{+}_{A^*}(a,a^*) \cap \mathcal{U}^{-}_{A^*}(a,b^*) = \emptyset$. Since from a) $U'(a) \geq U'(a^*)$, $\mathcal{U}^{+}_{A^*}(a,a^*) \neq \emptyset$ because $U' \in \mathcal{U}^{+}_{A^*}(a,a^*)$. Analogously, since from a) $U''(b^*) \geq U''(a)$, $\mathcal{U}^{-}_{A^*}(a,b^*) \neq \emptyset$ because $U'' \in \mathcal{U}^{-}_{A^*}(a,b^*)$. For all $U^+ \in \mathcal{U}^{+}_{A^*}(a,a^*)$ and $U^- \in \mathcal{U}^{-}_{A^*}(a,b^*)$ we have

$$L^{U^+}(a) = Max\{L^{DM}(c^*) : U^+(c^*) \leq U^+(a),\ c^* \in A^*\} \geq L^{DM}(a^*)$$

$$>$$

$$R^{DM}(b^*) \geq Min\left\{R^{DM}(c^*) : U^-(c^*) \geq U^-(a),\ c^* \in A^*\right\} = R^{U^-}(a),$$

such that there is no $U \in \mathcal{U}_{A^*}$ for which $[L^U(a), R^U(a)] \cap [R^{DM}+1, L^{DM}-1] \neq \emptyset$, i.e. for all $U \in \mathcal{U}_{A^*}$ and $h \in [R^{DM}+1, L^{DM}-1]$, $h \notin [L^U(a), R^U(a)]$. This means that no jump property (12) does not hold. Thus we proved that (4.4) is necessary for the no jump property (12).
Now we prove that (4.4) is sufficient for the no jump property (12) to hold. Two cases are possible:

1) for all $a^*, b^* \in A^*$ such that $L^{DM}(a^*) > R^{DM}(b^*)$ there do not exist $U', U'' \in \mathcal{U}_{A^*}$ for which $U'(a) \geq U'(a^*)$ and $U''(b^*) \geq U''(a)$,

2) there exist $a^*, b^* \in A^*$ and $U', U'' \in \mathcal{U}_{A^*}$ such that $L^{DM}(a^*) > R^{DM}(b^*)$ and $U'(a) \geq U'(a^*)$ and $U''(b^*) \geq U''(a)$.

Case 1) For 4.3 $C_N(a) \neq \emptyset$. Remembering that

$$C_N(a) = \bigcap_{U \in \mathcal{U}_{A^*}} [L^U(a), R^U(a)],$$

$C_N(a) \neq \emptyset$ implies that for all $U', U'' \in \mathcal{U}_{A^*}$,

$$[L^{U'}(a), R^{U'}(a)] \cap [L^{U''}(a), R^{U''}(a)] \neq \emptyset, \tag{44}$$

because $C_N(a) \subseteq [L^{U'}(a), R^{U'}(a)] \cap [L^{U''}(a), R^{U''}(a)]$. Remembering that

$$C_P(a) = \bigcup_{U \in \mathcal{U}_{A^*}} [L^U(a), R^U(a)],$$

from (44) we have that (12) is satisfied, i.e. there are no jumps in the classes of possible assignment.
Case 2) Let us observe that

$$L^{U'}(a) = max\{L^{DM}(c^*) : U'(c^*) \leq U'(a), c^* \in A^*\} \leq L^{DM}(a*)$$

$$>$$

$$R^{DM}(b*) \leq min\{R^{DM}(c^*) : U''(c^*) \geq U''(a), c^* \in A^*\} = R^{U''}(a)$$

such that

$$[L^{U'}(a), R^{U'}(a)] \cap [L^{U''}(a), R^{U''}(a)] = \emptyset$$

and therefore there could be a jump with respect to some classes C_h belonging to $[L^{DM}(b^*), R^{DM}(a^*)]$. However, for weak continuity (4.4), we have that there exists $U''' \in \mathcal{U}_{\mathcal{A}^*}$ for which $U'''(a^*) \geq U'''(a) \geq U'''(b^*)$ and thus

$$L^{U'''}(a) = max\{L^{DM}(c^*) : U'''(c^*) \leq U'''(a), c^* \in A^*\} \geq L^{DM}(b^*)$$

and

$$R^{U'''}(a) = min\{R^{DM}(c*) : U'''(c^*) \geq U'''(a), c^* \in A^*\} \leq R^{DM}(a^*),$$

such that, for proposition 3.1,

$$L^{DM}(b^*) \leq L^{U'''}(a) \leq R^{U'''}(a) \leq R^{DM}(a^*).$$

If there does not exist any other $d^* \in A^*$ different from a^* and b^* such that $U'''(a^*) \geq U'''(d^*) \geq U'''(b^*)$, we have $L^{U'''}(a) = L^{DM}(b^*)$ and $R^{U'''}(a) = R^{DM}(a^*)$, such that

$$[L^{U'''}(a), R^{U'''}(a)] = [L^{DM}(b^*), R^{DM}(a^*)],$$

and therefore there is no jump with respect to classes C_h belonging to $[L^{DM}(b^*), R^{DM}(a^*)]$.

If instead there exists some other $d^* \in A^*$ different from a^* and b^*, such that $U'''(a^*) \geq U'''(d^*) \geq U'''(b^*)$, there could continue to be some jump with respect to to classes C_h belonging to $[L^{DM}(b^*), R^{DM}(a^*)]$, if

1) $U'''(a) \geq U'''(d^*)$ and $L^{DM}(d^*) > R^{DM}(b^*)$, or

2) $U'''(a) \leq U'''(d^*)$ and $R^{DM}(d^*) < L^{DM}(a^*)$.

In fact:

- in case 1), we would have

  $$L^{U'''}(a) = max\{L^{DM}(c^*) : U'''(c^*) \leq U'''(a), c^* \in A^*\} \geq L^{DM}(d^*) > R^{DM}(b^*)$$

  such that $[L^{U''}(a), R^{U''}(a)] \cap [L^{U'''}(a), R^{U'''}(a)] = \emptyset$;

- in case 2), we would have

  $$R^{U'''}(a) = min\{R^{DM}(c^*) : U'''(c^*) \geq U'''(a), c^* \in A^*\} \leq R^{DM}(d^*) < L^{DM}(a^*)$$

  such that $[L^{U'}(a), R^{U'}(a)] \cap [L^{U'''}(a), R^{U'''}(a)] = \emptyset$.

However, observe that for weak continuity (4.4), we have that there exist $U^{iv}, U^v \in \mathcal{U}_{A^*}$ for which $U^{iv}(a^*) \geq U^{iv}(a) \geq U^{iv}(d^*)$ and $U^v(d^*) \geq U^{iv}(a) \geq U^{iv}(b^*)$, such that we can repeat the same reasoning until there is no more space for some jump. □

B.4 Proof of Corollary 4.1

Corollary 4.1 *If the set $\mathcal{U}_{A^*}$ of compatible value functions satisfies the weak continuity (4.4), the example-based sorting procedure assigns each reference action $a^* \in A^*$ to an interval of classes $[L_P^U(a^*), R_P^U(a^*)] \subseteq [L^{DM}(a^*), R^{DM}(a^*)]$.*

Proof. Since we are supposing that weak continuity (4.4) holds and for 4.4, in this case $C_P(a) = [L_P^U(a^*), R_P^U(a^*)]$. Moreover, for proposition each compatible value functions in the corresponding example-based sorting procedure assigns each reference action $a^* \in A^*$ to an interval of classes such that

$$L^U(a^*) \geq L^{DM}(a^*) \tag{45}$$
$$R^U(a^*) \leq R^{DM}(a^*). \tag{46}$$

Therefore also

$$L_P^U(a^*) \geq L^{DM}(a^*) \tag{47}$$
$$R_P^U(a^*) \leq R^{DM}(a^*), \tag{48}$$

which gives $[L_P^U(a^*), R_P^U(a^*)] \subseteq [L^{DM}(a^*), R^{DM}(a^*)]$. □

B.5 Proof of Proposition 4.5

Proposition 4.5 *If the set $\mathcal{U}_{A^*}$ of compatible value functions is convex, the $C_P(a) = [L_P^{\mathcal{U}}(a), R_P^{\mathcal{U}}(a)]$, i.e. the no jump property (12) holds.*

Proof. We prove that if the set $\mathcal{U}_{A^*}$ of compatible value functions is convex, then weak continuity (4.4) is satisfied, because, on the basis of 4.4, this is sufficient for no jump property (12). Let us suppose that there exist $a^*, b^* \in A^*$ such that $L^{DM}(a^*) > R^{DM}(b^*)$ and $U, U' \in \mathcal{U}_{\mathcal{A}^*}$ for which $U(a) \geq U(a^*)$ and $U'(b^*) \geq U'(a)$. Since $\mathcal{U}_{\mathcal{A}^*}$ is convex, then for all $\alpha \in [0,1]$, also $\alpha U + (1-\alpha)U' \in \mathcal{U}_{A^*}$. Thus we look for $U'' = \alpha U + (1-\alpha)U'$ such that $U''(a^*) > U''(a) > U''(b^*)$, that is

$$U(a^*) + (1-\alpha)U'(a^*) > U(a) + (1-\alpha)U'(a) > \alpha U(b^*) + (1-\alpha)U'(b^*) \qquad (i)$$

Let us remember that since the set of assignment example is consistent, from $L^{DM}(a^*) > R^{DM}(b^*)$ we get $U(a^*) > U(b^*)$ and $U'(a^*) > U'(b^*)$. Thus from (i) we get

$$\alpha U(a^*) + (1-\alpha)U'(a^*) > \alpha U(a) + (1-\alpha)U'(a)$$

$$\Leftrightarrow$$

$$\alpha[U(a^*) - U'(a^*) - U(a) + U'(a)] > U'(a) - U'(a^*)$$

$$\Leftrightarrow$$

$$\alpha[U'(a^*) - U'a) + U(a) - U(a^*)] < U'(a^*) - U'a)$$

$$\Leftrightarrow$$

$$\alpha < \frac{U'(a^*) - U'a)}{[U'(a^*) - U'a) + U(a) - U(a^*)]}$$

and

$$\alpha U(a) + (1-\alpha)U'(a) > \alpha U(b^*) + (1-\alpha)U'(b^*)$$

$$\Leftrightarrow$$

$$\alpha[U(a) - U'(a) - U(b^*) + U'(b^*)] > U'(b^*) - U'(a)$$

$$\Leftrightarrow$$

$$\alpha[U(a) - U(b^*) + U'(b^*) - U'(a)] > U'(b^*) - U'(a)$$

$$\Leftrightarrow$$

$$\alpha > \frac{U'(b^*) - U'(a)}{[U(a) - U(b^*) + U'(b^*) - U'(a)]}.$$

This means that if $U'' = \alpha U + (1-\alpha)U'$ and we fix

$$\frac{U'(a^*) - U'a)}{[U'(a^*) - U'a) + U(a) - U(a^*)]} > \alpha > \frac{U'(b^*) - U'(a)}{[U(a) - U(b^*) + U'(b^*) - U'(a)]}$$

we get $U"(a^*) > U"(a) > U"(b^*)$, which concludes the proof. □

B.6 Proof of Proposition 4.6

Proposition 4.6 *If $\mathcal{U}_{A^*}$ is the set of all additive compatible value functions, then $C_P(a) = [L_P^{\mathcal{U}}(a), R_P^{\mathcal{U}}(a)]$, i.e. the no jump property (12) holds.*

Proof. On the basis of Proposition 4.5, we have simply to prove that if $\mathcal{U}_{A^*}$ is the set of all additive compatible value functions, then it is convex. Observe that, for any $U, U' \in \mathcal{U}_{\mathcal{A}^*}\mathcal{U}_{A^*}$ and all $a, b \in A$ if

$$U(a) = \sum_{j=1}^{n} u_j(g_j(a)) \geq \sum_{j=1}^{n} u_j(g_j(b)) = U(b),$$

and

$$U'(a) = \sum_{j=1}^{n} u'_j(g_j(a)) \geq \sum_{j=1}^{n} u'_j(g_j(b)) = U'(b),$$

then also

$$\sum_{j=1}^{n}(\alpha u_j(g_j(a)) + (1-\alpha)u_j(g_j(a)) \geq \sum_{j=1}^{n} \alpha u_j(g_j(b)) + (1-\alpha)u'_j(g_j(b))$$

Therefore, if $U, U' \in \mathcal{U}_{\mathcal{A}^*}$ are additive compatible value functions, also $U''(x) = \sum_{j=1}^{n} u''_j(g_j(a)$ such that for all $j \in G$ $u''_j(x) = (\alpha u_j(g_j(x)) + (1-\alpha)u_j(g_j(x))$ is an additive compatible value function. Since $U'' = \alpha U + (\alpha)U'$, then if $\mathcal{U}_{A^*}$ is the set of all additive compatible value functions, it is convex. □