INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

WILEY

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 26 (2019) 1786–1809
DOI: 10.1111/itor.12512

Learning monotone preferences using a majority rule sorting model

Olivier Sobrie[a,b], Vincent Mousseau[b] and Marc Pirlot[a]

[a] *Faculté Polytechnique, Université de Mons, rue de Houdain, 7000 Mons, Belgium*
[b] *LGI, CentraleSupélec, Université Paris-Saclay, Gif sut Yvette, France*
E-mail: olivier.sobrie@gmail.com [Sobrie]; vincent.mousseau@ecp.fr [Mousseau]; marc.pirlot@umons.ac.be [Pirlot]

Received 10 March 2017; received in revised form 9 October 2017; accepted 30 December 2017

Abstract

We consider the problem of learning a function assigning objects into ordered categories. The objects are described by a vector of attribute values and the assignment function is monotone w.r.t. the attribute values (monotone sorting problem). Our approach is based on a model used in multicriteria decision analysis (MCDA), called MR-Sort. This model determines the assigned class on the basis of a majority rule and an artificial object that is a typical lower profile of the category. MR-Sort is a simplified variant of the ELECTRE TRI method. We describe an algorithm designed for learning such a model on the basis of assignment examples. We compare its performance with choquistic regression, a method recently proposed in the preference learning community, and with UTADIS, another MCDA method leaning on an additive value function (utility) model. Our experimentation shows that MR-Sort competes with the other two methods, and leads to a model that is interpretable.

Keywords: multiple criteria decision analysis; classification; majority rule sorting; preference learning; heuristic

1. Introduction

Sorting problems frequently arise in real-life contexts. For instance, a committee is assigned the task of separating good projects from bad ones, a jury has to assign grades to students. To select the category in which an alternative (e.g., a project, a student) should be assigned, a common and intuitive approach consists in analyzing its characteristics recorded as the value of attributes, also called criteria (e.g., for grading a student, the marks obtained in the different subjects). In both examples above, categories are specified before sorting takes place and they are ordered to reflect the preference of the decision maker (DM). Furthermore, the grading of students typically is monotone w.r.t. the marks obtained. Better marks cannot lead to a worse grade.

These two properties, that is, sorting in *ordered* classes by rules that are *monotone* w.r.t. attribute values, characterize the kind of sorting problems we want to address with the algorithm presented in this paper. Formally, each alternative a in a set A is described as a vector $(a_1, \ldots, a_j, \ldots, a_n)$, where a_j is the value of a on the jth attribute. This value is an element of the attribute's scale X_j, endowed with a preference order $\succsim_j$. The set of alternatives A may thus be seen as a subset of the Cartesian product, $X = \prod_{j=1}^{n} X_j$. Sorting the alternatives in the ordered categories $C_p \succ \cdots \succ C_1$ (where $\succ$ describes the DM's preference order on the categories) amounts to define a function $g : A \subseteq X \to \{C_1, \ldots, C_p\}$. The sorting function g is *monotone* if an alternative a cannot be assigned to a less preferred category than an alternative b whenever a is at least as good as b on all attributes. Precisely, for all $a, b \in A$, with $a_j \succsim_j b_j$ for all j, we have $g(a) \succ g(b)$ or $g(a) = g(b)$.

In the preference learning (PL) community, learning such monotone assignment functions on the basis of assignment examples is referred to as *monotone learning* (Tehrani et al., 2012).

In the multiple criteria decision analysis (MCDA) community, sorting alternatives into categories on the basis of their evaluation on a family of criteria is one of the central problems (Figueira et al., 2013). In contrast with machine learning usage, MCDA puts the emphasis on interaction with the DM. Their preferences are usually elicited in the course of an interactive process and explicitly incorporated in the sorting model. There are two main categories of MCDA models that have been used for sorting purposes:

1. Methods based on the construction of an overall score or value function aggregating all attributes. The weighted sum and the additive value function models (Belton and Stewart, 2002, Chapter 6) belong to this category. Basically, an alternative is assigned to a category whenever its score (or value) reaches some lower threshold value and is smaller than the lower threshold of the category just above.
2. Methods based on outranking relations. They are inspired by social choice theory and rely on pairwise comparisons of alternatives. Well-known in this family are the ELECTRE methods and, in particular, ELECTRE TRI. In this framework, an alternative is assigned to a category if it is preferred to the lower profile of the category, but is not preferred to the lower profile of the category just above. The preference (called *outranking*) of an alternative over another is determined by means of a *concordance-nondiscordance* rule. Such a rule checks whether the former is at least as good as the latter on a sufficiently strong coalition of criteria and whether there is no criteria on which the former alternative is unacceptably worse than the latter.

For a detailed presentation of such methods, the reader is referred to Doumpos and Zopounidis (2002) and Zopounidis and Doumpos (2002).

Eliciting the parameters of a model in MCDA can either be done directly or indirectly. Since MCDA favors interactions with the DM, the parameters are often elicited through directly asking the DM questions that determine or restrict the range of variation of the model's parameters. Questioning procedures are often cognitively demanding while DMs are usually very busy. Therefore, indirect methods have been developed based on a learning set consisting, for instance, of assignment examples, in the case of a sorting problem. Several learning algorithms were proposed in the MCDA literature, in particular based on linear programing. It is the case of the UTADIS method developed by Jacquet-Lagrèze and Siskos (1982) (see also Doumpos and Zopounidis, 2002, Chapter 4).

However, in MCDA, learning sets are usually of small size and therefore, the learning algorithms developed within this field are not specially designed to handle large data sets. Hence, model parameters are typically underdetermined.

The aim of this paper is to present and test an algorithm designed to learn the parameters of an MCDA sorting model, called MR-Sort, which is based on an outranking relation. This algorithm is able to deal with large learning sets. We compare its performance with that of other algorithms developed within the machine learning community as well as with that of the UTADIS sorting method. Whenever possible, we also confront the results provided by our heuristic with the optimal solution obtained by solving a mixed integer formulation of the problem. For large instances, however, the latter is not feasible in reasonable computing times. The main advantage of using MCDA models, as compared with machine learning algorithms, is that the former allow for an interpretation of the explicit rules used for sorting the objects.

The paper is structured as follows. After a brief literature review in Section 2, we describe the MR-Sort model and its specificities in Section 3. Section 4 describes the algorithm elaborated to learn the parameters of this model. We provide here a complete description of this algorithm, which was previously more briefly presented in Sobrie et al. (2013). In the interval, some changes have been brought to the algorithm in view of enhancing its efficiency, which allows to apply it to larger data sets and benchmarks used in machine learning. In Section 5, the results obtained by testing the algorithm on real data sets are presented and its performance is compared with other MCDA and machine learning algorithms, both for binary and multiclass sorting. Finally, in Section 6, we conclude and outline some perspectives for further developments.

2. Literature review

Model-based PL consists in using models making assumptions about the structure of preference relations (Fürnkranz and Hüllermeier, 2010). In this section, we emphasize the link between the MCDA and PL domains (see also Sobrie, 2016).

2.1. Multiple criteria decision analysis and preference learning

MCDA models are designed to provide support to a DM facing a decision problem. Once their parameters have been elicited, models lead to a recommendation and allow to explain it to the DM. Usually, problems treated in MCDA only involve a small number of alternatives and the parameters of the model are determined by interacting with one or several DMs.

The involvement of a DM in the building of a preference model is a characteristic of MCDA. This feature makes it come close to *active learning* techniques in PL.

For several reasons, determining the value of the model's parameters by questioning the DM can prove difficult. Therefore, several algorithms were proposed to learn such parameters from holistic preference statements or assignment examples (e.g., Jacquet-Lagrèze and Siskos, 1982; Mousseau and Słowiński, 1998; Jacquet-Lagrèze and Siskos, 2001; Bous et al., 2010; Doumpos et al., 2014). These algorithms are often based on linear programing or mixed integer program (MIP). The latter

can only deal with relatively small data sets in which alternatives are described on few criteria or attributes (e.g., Leroy et al., 2011).

Usually, PL focusses on the learning and predictive performance of algorithms. It does not emphasize the interpretability of the results. Algorithms used to predict assignments, rankings, etc., are used as black boxes. Contrary to MCDA, problems dealt with in this subfield of machine learning generally involve large data sets and many attributes.

Using MCDA models in the context of machine learning is especially advisable in the case of *monotone PL*, that is, in case the position of an object in a ranking or its assignment to a category cannot become worse when the attributes that are relevant for its description take on better values (which implies that a "natural" order is defined on the values of the attributes). Monotonicity of the preference w.r.t. the values of the attributes (called criteria) is a fundamental property of most MCDA problems.

The use and perspectives of MCDA models in machine learning was discussed by Corrente et al. (2013). These authors emphasize in particular the usage of robust ordinal regression (ROR) in ranking problems.

In this paper, we focus on the sorting problem statement in which input attributes and classes are monotone. In machine learning, monotone classification has been introduced by Ben-David et al. (1989). Since then, several papers were published in the field of PL to deal with the sorting problem statement. Ben-David (1995) proposes a learning method based on decision trees. In Sill (1998), neural networks are used in the context of monotone classification. Chandrasekaran et al. (2005) develop an algorithm based on the so-called *isotonic separation*. In Dembczyński et al. (2006), a method using additive value functions, similar to UTADIS (Jacquet-Lagrèze and Siskos, 1982), is studied. Dembczyński et al. (2009) propose an algorithm based on the dominance rough set approach. Recently, Tehrani et al. (2012) developed an algorithm that allows to learn the parameters of a Choquet integral, which is used for sorting the items.

In MCDA, several sorting procedures exist. Zopounidis and Doumpos (2002) provide an overview of sorting methods in MCDA. Some of them, like UTADIS (Jacquet-Lagrèze and Siskos, 1982) and the dominance-based rough set approach (Greco et al., 2001), are designed to learn the preferences of the DM on the basis of assignment examples, but their performance has not been tested extensively on large size benchmark data sets.

2.2. Learning an ELECTRE TRI model

To our knowledge, none of the sorting algorithms proposed up to now in the machine learning field are based on ELECTRE models. In MCDA, multiple papers are devoted to learn the parameters of ELECTRE TRI models. Mousseau and Słowiński (1998) propose a linear program aiming to learn all the parameters of an ELECTRE TRI model. Mousseau et al. (2001) consider the problem of finding the weights and the majority threshold of an ELECTRE TRI model for which delimiting profiles are known beforehand. They propose a linear program on which they conduct some experiments. In Ngo The and Mousseau (2002), an MIP is suggested to learn the profiles of an ELECTRE TRI model for which the weights and the majority threshold are known beforehand. Other linear and MIP programs allowing to learn the vetoes of an ELECTRE TRI model are presented in Dias et al. (2002).

In Doumpos et al. (2009), a genetic algorithm is set up to learn the parameters of an ELECTRE TRI model. It can be transposed to learn the parameters of an MR-Sort model. However, this metaheuristic is not especially designed to adapt to the structure of the problem: its crossover and mutation operators are standard ones. In operational research, it is well known that a metaheuristic adapted to the problem structure performs better (Pirlot, 1996).

Learning the parameters of an MR-Sort model has been already studied in Leroy et al. (2011). The paper describes a mixed integer program (MIP) allowing to learn the parameters of an MR-Sort model on the basis of assignment examples. Cailloux et al. (2012) describe three MIPs aiming to learn the parameters of an MR-Sort model in the context of multiple DMs. These programs require even more binary variables than the MIP developed in Leroy et al. (2011). It is worthless to consider using one of these programs in the context of PL problems where large data sets are involved.

3. MR-Sort, an ordered classification method

MR-Sort is a method for assigning objects to ordered categories. Each object is described by a vector of attribute values. The attribute values can be meaningfully ordered, that is, there is an underlying order on each attribute scale, which is interpreted as a "better than" relation. Categories are determined by limit profiles, which are vectors of attribute values. The lower limit profile of a category is the upper limit profile of the category below. The MR-Sort rule works as follows. An object is assigned to a category if it is better than the lower limit profile of the category on a sufficiently large coalition of (weighted) criteria and this condition is not met with respect to the upper limit profile of this category. Obviously, MR-Sort is a monotone rule, that is, an object that is at least as good as another on all attributes cannot be assigned to a lower category.

The MR-Sort rule is a simplified version of the ELECTRE TRI procedure, a method that is used in MCDA to assign alternatives to predefined categories (Yu, 1992; Roy and Bouyssou, 1993). The underlying semantic is generally to assign the alternatives labels such as "good," "average," "bad,"....

To be more formal, let X be a set of objects evaluated on n-ordered attributes (or criteria). We assume that X is the Cartesian product of the attribute ranges, $X = X_1 \times X_2 \times \ldots \times X_n = \prod_{j=1}^{n} X_j$. An object a in X is thus a vector $(a_1, \ldots, a_j, \ldots, a_n)$, where $a_j \in X_j$ for all j.

The categories that the objects are assigned to by the MR-Sort model are denoted by C_h, with $h = 1, \ldots, p$. The category C_h is delimited by its lower limit profile b_{h-1} and its upper limit profile b_h, which is also the lower limit profile of category C_{h+1} (provided $h < p$). The profile b_h is the vector of attribute values $(b_{h,1}, \ldots, b_{h,j}, \ldots, b_{h,n})$, with $b_{h,j} \in X_j$ for all j.

By convention, the best category, C_p, is delimited by a fictive upper profile, b_p, and the worst one, C_1, by a fictive lower profile, b_0.

It is assumed that the profiles dominate one another, that is:

$$b_{h-1,j} \leq b_{h,j}, \quad h = 1, \ldots, p; \quad j = 1, \ldots, n.$$

Figure 1 provides a graphical representation of the profiles and categories of an MR-Sort model.

Using the MR-Sort procedure (without veto), an object is assigned to a category if its attribute values are at least as good as the category lower profile values on a weighted majority of criteria and this condition is not fulfilled when the object's attribute values are compared to the category

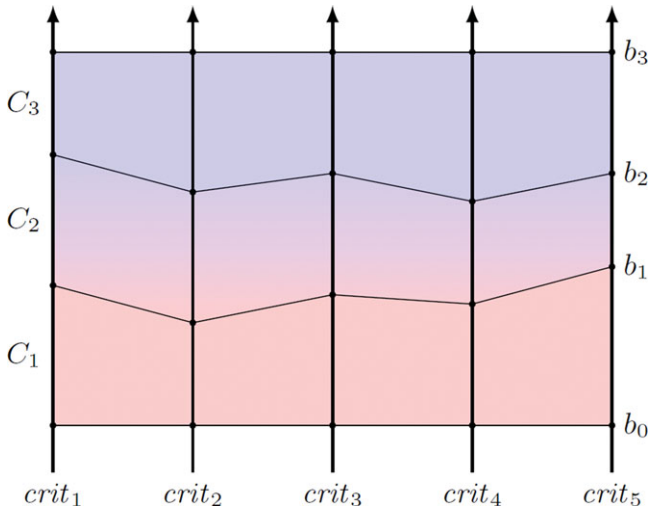

Fig. 1. Profiles and categories of an MR-Sort model (five criteria, three categories)

upper profile values. In the former case, we say that the object is *preferred* to the profile, while, in the latter, it is not. Formally, if an object $a \in X$ is *preferred* to a profile b_h, we denote this by $a \succsim b_h$. Object a is preferred to profile b_h whenever the following condition is fulfilled:

$$a \succsim b_h \Leftrightarrow \sum_{j:a_j \geq b_{h,j}} w_j \geq \lambda, \tag{1}$$

where w_j is the nonnegative weight associated with attribute j, for all j and λ sets a majority level. The weights satisfy the normalization condition $\sum_{j \in F} w_j = 1$; λ is called the *majority threshold*; it satisfies $\lambda \in [1/2, 1]$. The sum $\sum_{j:a_j \geq b_{h,j}} w_j$ is often called the *concordance index* and will be denoted by $\sigma(a, b_h)$. It measures the strength of the coalition of criteria backing the hypothesis that a is at least as good as the profile b_h. The threshold λ tells which coalitions are strong enough to conclude that indeed a is at least as good as b_h.

The preference relation $\succsim$ defined by (1) is called an *outranking* relation without veto or a *concordance* relation (Roy and Bouyssou, 1993; see also Bouyssou and Pirlot 2005, 2007, 2015 for an axiomatic description of such relations).

Consequently, the condition for an object $a \in X$ to be assigned to category C_h is written as:

$$\sum_{j:a_j \geq b_{h-1,j}} w_j \geq \lambda \quad \text{and} \quad \sum_{j:a_j \geq b_{h,j}} w_j < \lambda. \tag{2}$$

The MR-Sort assignment rule described above involves $pn + 1$ parameters, that is, n weights, $(p-1)n$ profiles evaluations, and one majority threshold. Note that the profiles b_0 and b_p are conventionally defined as follows: $b_{0,j}$ is a value such that $a_j \geq b_{0,j}$ for all $a \in X$ and $j = 1, \ldots, n$; $b_{p,j}$ is a value such that $a_j < b_{p,j}$ for all $a \in X$ and $j = 1, \ldots, n$.

A *learning set* A is a subset of objects $A \subseteq X$ for which an assignment is known. For $h = 1, \ldots, p$, A_h denotes the subset of objects $a \in A$, which are assigned to category C_h. The subsets A_h are disjoint; some of them may be empty.

Remark. The MR-Sort rule described above is a slightly particularized version of the *noncompensatory sorting model* studied by Bouyssou and Marchant (2007a, 2007b). With respect to these methods, MR-Sort has the following features: (i) no veto is considered and (ii) the coalitions of criteria that are sufficiently large[1] and imply that determining whether an object is preferred to a profile can be done by associating weights to the attributes and selecting a majority threshold through formula (1).

4. Learning the parameters of an MR-Sort model

We briefly introduce the method proposed in Leroy et al. (2011) to learn the parameters of an MR-Sort model by solving a MIP. We recall the limitations of such an approach in terms of computing time and memory space resources. We then describe the metaheuristic we have developed to achieve the same goal.

4.1. Mixed integer programing formulation

In Leroy et al. (2011), a linear program involving binary variables was used to learn the parameters of an MR-Sort model (without veto). The program tries to minimize the 0/1 loss, that is, it searches for a model that is compatible with as many examples as possible.

Learning the parameters of an MR-Sort model using linear programing cannot avoid using binary variables. The MIP proposed by Leroy et al. (2011) involves $m \cdot (2n + 1)$ binary variables, where m is the size of the learning set and n, the number of attributes.

The experimental results showed that learning the parameters of a model for data sets involving a large number of assignment examples, criteria, or categories, requires huge computing times (using the IBM ILOG CPLEX solver). With barely 100 alternatives, 5 criteria, and 3 categories (i.e., 1100 binary variables in the MIP), more than 100 seconds are needed to learn the parameters of an MR-Sort model.[2]

Due to these long computing times, using this MIP is not a feasible approach for the type of problem we want to handle, that is, problems involving large data sets. An option to overcome the computing time issue is to use relaxation techniques that allow to obtain an approximate solution (Minoux, 2008; Wolsey, 1998). Instead of exploring this path (which certainly deserves attention), we developed a new sophisticated—population based—metaheuristic that exploits as much as possible the specificities of the problem. This algorithm is described in the next section.

[1] In a general noncompensatory sorting method, the class of sufficient coalitions cannot always be described by an additive measure; it may require the use of a capacity. See Bouyssou and Marchant (2007a) for more detail.

[2] System used: Core 2 Duo P8700, running Gentoo Linux and CPLEX 12.5.

4.2. The metaheuristic

In Section 2, we gave an overview of several algorithms that were proposed to learn the parameters of ELECTRE TRI and MR-Sort models. In Sobrie et al. (2013), we described a preliminary version of a metaheuristic for learning MR-Sort models. This version needed improvements from an efficiency point of view in order to tackle large datasets, and was only validated on artificial data.

The present metaheuristic is grounded on the following two important observations:

- Given a set of profiles, learning the weights and the majority threshold of an MR-Sort model can easily be achieved by solving a linear program without binary variables.
- In contrast, given a set of weights and a majority threshold, learning the profiles values by means of linear programing requires using binary (0/1) variables.

To properly take the structure of the problem into account, that is, the ease of learning the weights and the majority threshold with a linear program and the difficulty to do the same for the profiles, we separate the algorithm in three components:

1. A heuristic that initializes a set of profiles;
2. a linear program learning the weights and the majority threshold of the model on the basis of fixed profiles; and
3. an heuristic adjusting the profiles to improve the quality of the model, while keeping the weights and majority threshold fixed.

The objective of the algorithm is to find a model restoring as many examples as possible. To assess the quality of the models, we use two indicators. The first is the classification accuracy (CA) criterion, which is defined as follows:

$$CA = \frac{\text{Number of assignment examples restored}}{\text{Total number of assignment examples}}. \tag{3}$$

The higher the value of the CA, the better the quality of the model. The second indicator is the area under curve (AUC), which quantifies the discriminating power of the algorithm to separate alternatives in different classes. Obviously, by optimizing successively the weights and threshold, then the profiles, again the weights and threshold, and so on, instead of optimizing all parameters simultaneously, there is no guarantee that a very good solution will be reached, even though the process is iterated. In order to enhance the chances to converge toward a very good solution, we adopt an evolutionary approach evolving a population of N_{mod} MR-Sort models.

The general architecture of our algorithm is described as Algorithm 1. The latter shows how the three components are combined to find a MR-Sort model that restores as well as possible the assignment examples in the learning set.

Algorithm 1 Metaheuristic to learn all the parameters of an MR-Sort model

Generate a population of N_{mod} models with profiles set by an initializing heuristic
repeat
  for all model M of the set **do**
    Learn the weights and majority threshold with a linear program, using the current profiles
    Adjust the profiles with a heuristic, using the current weights and threshold; repeat N_{it} times.
  end for
  Reinitialize the $\left\lfloor \frac{N_{mod}}{2} \right\rfloor$ models giving the bottom values of AUC
until Stopping criterion is met

First, a population of N_{mod} is generated and, for each model, the set of profiles are initialized by a specific heuristic. After the initialization phase, for each model M, the algorithm solves a linear program to find the weights and the majority threshold with fixed profiles (obtained in the initialization step). Then, for each model M, on the basis of the weights and majority threshold learned in the previous step, the metaheuristic adjusts the profiles with a randomized heuristic in order to maximize the number of examples compatible with the model. The randomized heuristic alters the profiles N_{it} times for each model M, after which the set of profiles restoring the largest number of assignment examples is selected. This process results in a new population of N_{mod} models. These are ordered by decreasing order of the AUC criterion. The top half of the models are retained while the bottom half (precisely $\lfloor \frac{N_{mod}}{2} \rfloor$ models) are reset using the initializing heuristic.

The algorithm stops either after having run a given number of times, denoted by N_o (fixed a priori), or when it has found at least one model that restores correctly all the assignment examples. If no model restores correctly all the assignment examples, the model giving the best AUC is returned.

In the next subsections, we detail the three components of the algorithm.

4.2.1. Profile initialization

The first step of the algorithm consists in the initialization of a set of profiles for each of the N_{mod} models in the population. The general idea of the heuristic designed to set the value $b_{h,j}$ of the profile b_h on criterion j is the following. This value is chosen in order to maximize the discriminating power of each criterion relatively to the alternatives in the learning set A. More precisely, we set $b_{h,j}$ in such a way that alternatives ranked in the category above b_h (i.e., C_{h+1}) typically have an evaluation greater than $b_{h,j}$ on criterion j and those ranked in the category below b_h (i.e. C_h) typically have an evaluation smaller than $b_{h,j}$.

In setting the initial profile values, we pay attention to the following aspects. First, for guaranteeing an equal treatment to all profiles, we chose to consider only C_h and C_{h+1} for determining b_h. The reason for this option is to balance the number of categories above and below the profile that are taken into account for determining this profile. For profiles b_1 and b_{p-1}, the only way of satisfying this requirement is to consider only one category above and one category below the profile.

The second issue is relative to the way the different categories are represented in the learning set. Consider the subsets A_h and A_{h+1} of alternatives in the learning set A that are assigned, respectively, to categories C_h and C_{h+1}. These subsets may be of quite different sizes. We weight the alternatives

by using the relative frequencies of A_h and A_{h+1} in order to control the influence of categories that are under- or overrepresented in the learning set.

The initializing heuristic is implemented as follows:

1. For each category, C_h, compute the frequency π_h with which alternatives in the learning set are assigned to category C_h, that is, $\pi_h = \frac{|A_h|}{|A|}$.
2. For each criterion and each profile b_h, a set of candidate profile values are selected. They correspond to the performances of alternatives in A assigned to categories C_h and C_{h+1}. The value of the profile, $b_{h,j}$, is chosen randomly among the candidate values with some probability. The probability of each candidate value is proportional to its likelihood to classify correctly alternatives of categories C_h and C_{h+1} on the basis of their performance on the sole criterion j. In view of balancing the influence of A_h and A_{h+1}, which may be of quite different sizes, the examples are assigned a weight that is inversely proportional to the size of the class they belong to.
3. The profiles are computed in descending order, enforcing the constraint that profile values on each criterion are ordered, that is, we have $b_{h+1,j} \geq b_{h,j}$, for all criterion j and profile h.

4.2.2. Learning the weights and the majority threshold

Assuming that the profiles are given, learning the weights and the majority threshold of a MR-Sort model from assignment examples is done by means of solving a linear program. The MR-Sort model postulates that the profiles dominate each other, that is, $b_{h+1,j} \geq b_{h,j}$ for all h and j, and the inequality is strict for at least one j. The constraints derived from the assignments of the alternatives in the learning set are expressed as follows:

$$\begin{aligned}
\textstyle\sum_{j:a_j \geq b_{h-1,j}} w_j - x_a + x'_a &= \lambda && \forall a \in A_h, h = 2, \ldots, p \\
\textstyle\sum_{j:a_j \geq b_{h,j}} w_j + y_a - y'_a &= \lambda - \varepsilon && \forall a \in A_h, h = 1, \ldots, p-1 \\
\textstyle\sum_{j=1}^{n} w_j &= 1 && \\
w_j &\in [0; 1] && j = 1, \ldots, n \\
\lambda &\in [0.5; 1] && \\
x_a, y_a, x'_a, y'_a &\in \mathbb{R}_0^+ && a \in A.
\end{aligned}$$

The small positive number ε is used for transforming strict inequalities into non strict ones. There are as many four tuples of variables x_a, y_a, x'_a, y'_a as there are alternatives in the learning set A. The value of $x_a - x'_a$ (resp. $y_a - y'_a$) represents the difference between the sum of the weights of the criteria belonging to the coalition in favor of $a \in A_h$ w.r.t. b_{h-1} (resp. b_h) and the majority threshold. If both $x_a - x'_a$ and $y_a - y'_a$ are positive, then the alternative a is assigned to the right category. In order to try to maximize the number of examples correctly assigned by the model, the objective function of the linear program minimizes the sum of x'_a and y'_a, that is, the objective function is $\min \sum_{a \in A}(x'_a + y'_a)$. Note however that such an objective function does not guarantee that the maximal number of examples are correctly assigned. Failing to meet this goal may be due to possible compensatory effects between constraints, that is, the program may favor a solution involving many small positive values of x'_a and y'_a over a solution involving large positive values of a few of these variables. Such a compensatory behavior could be avoided, but at the cost of

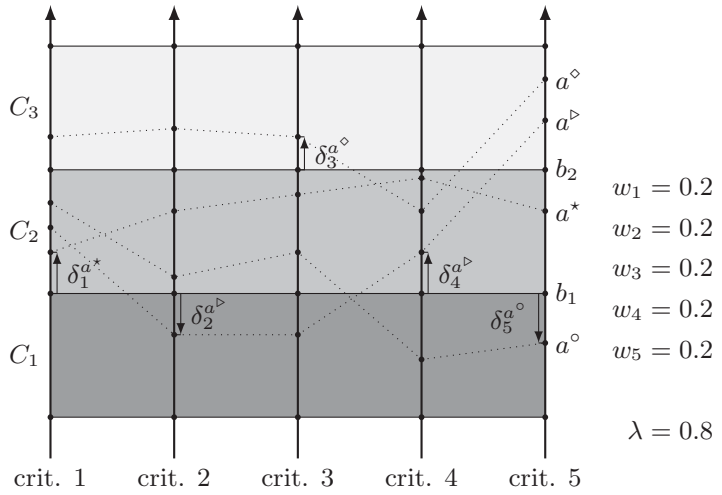

Fig. 2. Alternatives wrongly assigned because of profiles set too low or too high.

introducing binary variables, indicating each violation of the assignment constraints. We do not consider such formulations in order to keep computing times within reasonable limits.

4.2.3. Learning the profiles

Learning the profiles by using a mathematical programing formulation requires binary variables (Ngo The and Mousseau, 2002), leading to a MIP. In order to deal with problems involving large learning sets (e.g., 300 assignment examples, 10 criteria, and 5 categories), MIP is not an option, as discussed in Section 4.1. Therefore, we opt for a randomized heuristic algorithm that is described next.

For illustrative purposes, consider a model involving three categories and five criteria. Figure 2 represents the profiles and criteria as well as four alternatives, respectively, denoted as $a^\star$, $a^\triangleright$, $a^\diamond$, and a°. Criteria weights have been set equal ($w_j = 0.2$ for $j = 1, \ldots, 5$) and the majority threshold λ is set to 80%. Hence, an alternative is considered superior to a profile if it is at least as good as the profile on either four or five criteria.

Assume that the first three alternatives are misclassified by this model. The first alternative, $a^\star$, is assigned to category C_1 by the DM and to C_2 by the model. The second one, $a^\triangleright$, is assigned to category C_2 by the DM and to C_1 by the model and the third one, $a^\diamond$, is assigned to category C_1 by the DM and to C_3 by the model. Assuming fixed weights and majority threshold, this means that the profiles delimiting the categories are set either too high or too low on one or several criteria. Assume also that alternative a° is correctly assigned to category C_1 both by the DM and the model.

The idea implemented in the algorithm is to move up or down the profile value on some criterion in order to improve CA. We evaluate all possible moves of the profile on each attribute and select one likely to improve CA.

To be more precise, let us define several subsets of alternatives for each criterion j and each profile h and any positive value δ, which represents the size of a move:

$V_{h,j}^{+\delta}$ (resp. $V_{h,j}^{-\delta}$) : the sets of alternatives misclassified in C_{h+1} instead of C_h (resp. C_h instead of C_{h+1}), for which moving the profile b_h by $+\delta$ (resp. $-\delta$) on j results in a correct assignment. For instance, $a^{\triangleright}$ belongs to the set $V_{1,2}^{-\delta}$ on criterion 2 for $\delta \geq \delta_2^{a^{\triangleright}}$.

$W_{h,j}^{+\delta}$ (resp. $W_{h,j}^{-\delta}$) : the sets of alternatives misclassified in C_{h+1} instead of C_h (resp. C_h instead of C_{h+1}), for which moving the profile b_h by $+\delta$ (resp. $-\delta$) on j strengthens the criteria coalition in favor of the correct classification, but will not by itself result in a correct assignment. For instance, $a^{\star}$ belongs to the set $W_{1,1}^{+\delta}$ on criterion 1 for $\delta > \delta_1^{a^{\star}}$.

$Q_{h,j}^{+\delta}$ (resp. $Q_{h,j}^{-\delta}$) : the sets of alternatives correctly classified in C_{h+1} (resp. C_h) for which moving the profile b_h by $+\delta$ (resp. $-\delta$) on j results in a misclassification. For instance, a° belongs to the set $Q_{1,5}^{-\delta}$ on criterion 5 for $\delta > \delta_5^{a^{\circ}}$.

$R_{h,j}^{+\delta}$ (resp. $R_{h,j}^{-\delta}$) : the sets of alternatives misclassified in C_h instead of C_{h+1} (resp. C_{h+1} instead of C_h), for which moving the profile b_h by $+\delta$ (resp. $-\delta$) on j still strengthens the criteria coalition in favor of the incorrect classification. For instance, $a^{\triangleright}$ belongs to the set $R_{1,4}^{+\delta}$ on criterion 4 for $\delta > \delta_4^{a^{\triangleright}}$.

$T_{h,j}^{+\delta}$ (resp. $T_{h,j}^{-\delta}$) : the sets of alternatives assigned by the model to C_{h+1} or higher (resp. C_h or lower) but classified by the DM in a category below C_h (resp. to a category above C_{h+1}), for which moving the profile by $+\delta$ (resp. $-\delta$) on j strengthens the criteria coalition in favor of a classification that comes closer to the correct one. For instance $a^{\diamond}$ belongs to the set $T_{2,3}^{+\delta}$ on criterion 3 for $\delta > \delta_3^{a^{\diamond}}$.

In the above, subsets of type V and W contain alternatives that will tend to be better classified if we perform a given profile move. On the contrary, the assignment of alternatives in subsets of type Q will be worsened by the move; the (wrong) classification of alternatives in subsets of type R and T will not be altered by the move, but the latter goes "in the wrong direction" w.r.t. a correct classification of these alternatives. In order to formally define these sets, we introduce the following notation. A_h^l denotes the subset of misclassified alternatives that are assigned to category C_l by the model while the DM assigns them to category C_h. $A_{<h}^{>l}$ denotes the subset of misclassified alternatives that are assigned to a category above C_l by the model, while the DM assigns them to a category below C_h. And conversely for $A_{>h}^{<l}$. Finally, $\sigma(a, b_h) = \sum_{j:a_j \geq b_{h,j}} w_j$. We have, for any h, j and positive δ:

$$V_{h,j}^{+\delta} = \left\{a \in A_h^{h+1} : b_{h,j} + \delta > a_j \geq b_{h,j} \text{ and } \sigma(a, b_h) - w_j < \lambda\right\}$$

$$V_{h,j}^{-\delta} = \left\{a \in A_{h+1}^{h} : b_{h,j} - \delta < a_j < b_{h,j} \text{ and } \sigma(a, b_h) + w_j \geq \lambda\right\}$$

$$W_{h,j}^{+\delta} = \left\{a \in A_h^{h+1} : b_{h,j} + \delta > a_j \geq b_{h,j} \text{ and } \sigma(a, b_h) - w_j \geq \lambda\right\}$$

$$W_{h,j}^{-\delta} = \left\{a \in A_{h+1}^{h} : b_{h,j} - \delta < a_j < b_{h,j} \text{ and } \sigma(a, b_h) + w_j < \lambda\right\}$$

$$Q_{h,j}^{+\delta} = \left\{a \in A_{h+1}^{h+1} : b_{h,j} + \delta > a_j \geq b_{h,j} \text{ and } \sigma(a, b_h) - w_j < \lambda\right\}$$

$$Q_{h,j}^{-\delta} = \left\{a \in A_h^{h} : b_{h,j} - \delta < a_j < b_{h,j} \text{ and } \sigma(a, b_h) + w_j \geq \lambda\right\}$$

$$R_{h,j}^{+\delta} = \left\{a \in A_{h+1}^{h} : b_{h,j} + \delta > a_j \geq b_{h,j}\right\}$$
$$R_{h,j}^{-\delta} = \left\{a \in A_{h}^{h+1} : b_{h,j} - \delta < a_j < b_{h,j}\right\}$$
$$T_{h,j}^{+\delta} = \left\{a \in A_{<h}^{>h} : b_{h,j} + \delta > a_j \geq b_{h,j}\right\}$$
$$T_{h,j}^{-\delta} = \left\{a \in A_{>h+1}^{<h+1} : b_{h,j} - \delta < a_j \leq b_{h,j}\right\}.$$

The choice of a profile move is performed as follows. First, to avoid violations of the dominance rule between the profiles, the value of $+\delta$ or $-\delta$ is restricted to vary in the interval $[b_{h-1,j}, b_{h+1,j}]$. We then compute a *desirability index* $P(b_{h,j}^{+\delta})$ for each possible value $+\delta$ of a move of profile $b_{h,j}$. This index balances the alternatives that will be better off after the move and these on which the move will have a negative impact. The index is computed according to the following formula:

$$P(b_{h,j}^{+\delta}) = \frac{k_V\left|V_{h,j}^{+\delta}\right| + k_W\left|W_{h,j}^{+\delta}\right| + k_T\left|T_{h,j}^{+\delta}\right| + k_Q\left|Q_{h,j}^{+\delta}\right| + k_R\left|R_{h,j}^{+\delta}\right|}{d_V\left|V_{h,j}^{+\delta}\right| + d_W\left|W_{h,j}^{+\delta}\right| + d_T\left|T_{h,j}^{+\delta}\right| + d_Q\left|Q_{h,j}^{+\delta}\right| + d_R\left|R_{h,j}^{+\delta}\right|},$$

where k_V, k_W, k_T, k_Q, k_R, d_V, d_W, d_T, d_Q, and d_R are fixed constants. We define similarly $P(b_{h,j}^{-\delta})$. In the definition of $P(b_{h,j}^{+\delta})$ (resp. $P(b_{h,j}^{-\delta})$), the coefficients weighting the number of elements in the sets in the numerator are chosen so as to emphasize the arguments in favor of moving the value $b_{h,j}$ of profile b_h to $b_{h,j} + \delta$ (resp. $-\delta$), while the coefficients in the denominator emphasize the arguments against such a move. The values of the coefficients were empirically set as follows: $k_V = 2, k_W = 1, k_T = 0.1, k_Q = k_R = 0, d_V = d_W = d_T = 1, d_Q = 5, d_R = 1$.

The value $b_{h,j}$ of profile b_h on criterion j will possibly be moved to the value a_j of one of the alternatives a contained in $V_{h,j}^{+\delta}$, $V_{h,j}^{-\delta}$, $W_{h,j}^{+\delta}$, or $W_{h,j}^{-\delta}$. More precisely, it will be set to a_j or a value slightly above a_j. The exact new position of the profile is chosen so as to favor a correct assignment for a, taking into account the assignment rule (2). For instance, w.r.t. the situation illustrated in Fig. 2, the new value $b_{1,1} + \delta$ could be chosen just above the value of $a_1^\star$ so that criterion 1 would no longer belong to the coalition of criteria on which $a^\star$ is *at least as good* as b_1. Such a move would result in correctly assigning $a^\star$ to category C_1. If the move were driven by the position of alternative $a^\triangleright$ on criterion 2, then the new profile value $b_{1,2} + \delta$ would be set equal to the performance, $a_2^\triangleright$, of the alternative $a^\triangleright$ on criterion 2. Such a move would result in correctly assigning $a^\triangleright$ to C_2.

All such values a_j are located in the interval $[b_{h-1,j}, b_{h+1,j}]$. A subset of such values is chosen in a randomized way as follows. Among the set of values a_j, a value, denoted by $b'_{h,j}$, is chosen randomly. We denote by $d_{h,j}$ the difference $|b'_{h,j} - b_{h,j}|$. All values a_j located in $[b_{h-1,j}, b_{h,j} - d_{h,j}]$ and $[b_{h,j} + d_{h,j}, b_{h,j}]$ constitute a subset of candidate moves. The candidate move corresponds to the value a_j in the selected subset for which $P(b_{h,j}^{\Delta})$ is maximal, Δ being equal to $a_j - b_{h,j}$ (i.e., a positive or negative quantity). To decide whether to make the candidate move, a random number r is drawn uniformly in the interval [0, 1] and the value $b_{h,j}$ of profile b_h is changed if $P(b_{h,j}^{\Delta}) \geq r$.

This procedure is executed for all criteria and all profiles. Criteria are treated in random order and profiles in ascending order.

Algorithm 2 Randomized heuristic used for improving the profiles

```
for all profile b_h do
  for all criterion j chosen in random order do
    Choose, in a randomized manner, a sub-interval of [b_{h-1,j}, b_{h+1,j}]
    Select a position in this sub-interval for which P(b^Δ_{h,j}) is maximal
    Draw uniformly a random number r from the interval [0, 1].
    if r ≤ P(b^Δ_{h,j}) then
      Move b_{h,j} to the position corresponding to b_{h,j} + Δ
      Update the alternatives assignment
    end if
  end for
end for
```

5. Experiments

Our aim in these experiments is to analyze how our algorithm compares to the state-of-the-art in terms of performance in generalization. Such an analysis enables to appreciate the descriptive ability of the MR-Sort model as compared to other sorting models presented in the literature.

In this section, we first recall what we observed when we applied our algorithm to retrieve a model that was used to generate artificial data sets. Then, the algorithm is tested on real data sets.

We describe the experimental design and report the results. The algorithm and data sets used in this section are available at the following address: http://www.github.com/oso/pymcda.

5.1. Empirical validation on simulated data

In Sobrie et al. (2013), we conducted experiments to study the behavior of the algorithm on artificial data sets. These are produced as follows. An MR-Sort model is generated randomly (see Sobrie et al., 2013, for details). We draw at random vectors of evaluations representing the alternatives and we assign them using the MR-Sort model. Part of these vectors form the learning set and the rest constitutes the test set. We first use such data sets to determine an appropriate set of parameters (N_{mod}, N_{it}, N_o) for our heuristic algorithm.

We then conducted experiments to determine the number of assignment examples required to restore the model with a given accuracy. For a model involving 3 categories and 10 criteria, more than 400 examples are required to restore (on average) 95% of the assignments of a test set composed of 10,000 assignments. When the number of categories increases to five, more than 800 examples are required.

Real learning sets generally contain examples that are incompatible with an MR-Sort model. In order to assess the robustness of our algorithm to "assignment errors," we studied its behavior when such errors are introduced in the learning set. These "errors" were simulated by assigning some alternatives to a category different from that given by the MR-Sort model. When the learning set contains a proportion P of errors, we observe that the CA obtained with the learned model, that is, the proportion of test set alternatives correctly restored, converges toward $1 - P$. We also observe that most alternatives in the learning set that are wrongly assigned by the learned model are altered examples. In fact, the learned model corrects part of the introduced assignment errors.

Table 1
Data sets

Data set	No. of instances	No. of attributes	No. of categories
DBS	120	8	2
CPU	209	6	4
BCC	286	7	2
MPG	392	7	36
ESL	488	4	9
MMG	961	5	2
ERA	1000	4	4
LEV	1000	4	5
CEV	1728	6	4
ASA	898	16	4

5.2. Data sets and experimental design

For comparison purposes, we use the data sets that were considered by Tehrani et al. (2012) for testing the performance of a binary classifier based on the Choquet integral. These data sets were taken from two sources: the UCI machine learning repository and the WEKA repository (Machine Learning Group, University of Waikato, NZ). In addition, we consider also the ASA data set that was compiled and studied by Lazouni et al. (2013) (available at http:olivier.sobrie.be/shared/asa). The characteristics of all data sets are displayed in Table 1. All the attributes in these data sets are treated as monotone attributes.

The tests are conducted as follows. Each data set is randomly split in two disjoint parts. The first part is used as learning set and the second part as test set. The following size ratios between the learning set and the test set are considered: 20/80, 50/50, and 80/20. For each data set and each ratio, a random drawing of the learning set from the whole data set is repeated 100 times, yielding 100 instances of a partition of the data set in a learning set and a test set.

For each learning set instance, the algorithm finds a model that minimizes the 0/1 loss, that is, that is compatible with as many examples as possible. Afterwards, the alternatives in the test set are assigned by the learned model and the resulting assignments are compared to the original ones. This procedure is thus repeated 100 times for each data set and each relative size of the learning set.

Two indicators are computed to assess the quality of the learned models: the 0/1 loss and the AUC.

The performance of our heuristic algorithm is not only compared with the results obtained by Tehrani et al. (2012), but also with the exact solution of the MIP formulation (whenever it can be obtained) and with another previously mentioned MCDA method, UTADIS. For the reader's convenience, we briefly recall the principles of UTADIS, referring the reader to Jacquet-Lagrèze and Siskos (1982, 2001) and Zopounidis and Doumpos (2002) for further detail. This method is based on the learning of an additive value (or utility) function and thresholds that determine the minimal and maximal values of an alternative that is assigned to a given category. More formally, the UTADIS assignment rule reads as follows: for all $a \in X$,

$$a \in C_h \qquad \text{if } u(a) = \sum_{j=1}^{n} u_j(a_j) \in [U_{h-1}, U_h[, \tag{4}$$

where u_j are marginal value functions and U_{h-1} (resp. U_h) is the lower threshold value of category C_h (resp. C_{h+1}). UTADIS considers marginal value functions u_j that are monotone and piecewise linear. In our experiments, the domain of variation of each attribute is divided in three segments of equal length. Determining the marginal value at the break points is sufficient for determining the whole marginal value function u_j. Using piecewise linear marginal value functions enables to formulate the problem as a linear program and solve it with efficient solvers such as IBM ILOG CPLEX.

For all the experimentation, the MR-Sort metaheuristic is run with a population of 10 models ($N_{mod} = 10$) and the maximal number of iterations is fixed to 10 ($N_o = 10$). The outer loop of the metaheuristic, which adjusts the profiles and recomputes weights, is repeated 20 times ($N_{it} = 20$).

5.3. Binary classification

The algorithm developed by Tehrani et al. (2012) is designed for monotone sorting in two categories. In order to compare the performance of our algorithm with theirs, the assignments in the data sets presented in Table 1 are binarized by thresholding at the median, in the same way as was done by these authors. From these data sets, the parameters of an MR-Sort model are learned by using 20%, 50%, or 80% of the records as learning alternatives and the rest as test alternatives.

Our experimentation has two objectives. The first is to compare the quality of the MR-Sort models found by our metaheuristic with the ones obtained by an exact optimization method. Therefore, we solve the MIP formulation studied in Leroy et al. (2011) which minimizes the 0/1 loss of the model. Whenever the MIP solver is able to find a solution in the computing time allowed, we assess the learned models by comparing their average 0/1 loss on the test set.

Our second objective is to compare the performance of the proposed metaheuristic with that of other MCDA and machine learning algorithms, UTADIS (Jacquet-Lagrèze and Siskos, 1982; Doumpos and Zopounidis, 2002), a well-known MCDA method, and the choquistic regression (CR) (Tehrani et al., 2012), a method recently developed in the field of PL. To assess our metaheuristic, we use the average 0/1 loss and *AUC* computed on the test sets.

The *AUC* of an MR-Sort model with two categories C_1, C_2 is computed by comparing the concordance indices of the alternatives w.r.t. the profiles. The value of the *AUC* is given by equation (5).

$$AUC = \frac{1}{|A_1| \cdot |A_2|} \sum_{a^i \in A_2} \sum_{a^k \in A_1} \tau(a^i, a^k) \tag{5}$$

with A_1 (resp. A_2), the set of input alternatives classified in C_1 (resp. C_2). In the case of MR-Sort, we define $\tau(a^i, a^k)$ as follows:

$$\tau(a^i, a^k) = \begin{cases} 0 & \text{if } \sum_{j:a_j^i \geq b_{1,j}} w_j < \sum_{j:a_j^k \geq b_{1,j}} w_j \\ 0.5 & \text{if } \sum_{j:a_j^i \geq b_{1,j}} w_j = \sum_{j:a_j^k \geq b_{1,j}} w_j \\ 1 & \text{if } \sum_{j:a_j^i \geq b_{1,j}} w_j > \sum_{j:a_j^k \geq b_{1,j}} w_j. \end{cases}$$

Table 2
Average and standard deviation of the 0/1 loss (in percent) of the test set for learning sets of different sizes

Learning set	Data set	META	MIP	UTADIS	CR
20% data set	DBS	18.97 ± 4.23	19.77 ± 4.81	20.08 ± 5.33	17.13 ± 4.24
	CPU	9.94 ± 3.23	9.00 ± 3.45	6.52 ± 3.62	8.11 ± 1.03
	BCC	28.24 ± 2.73	26.78 ± 2.76	29.15 ± 3.07	27.75 ± 3.35
	MPG	20.25 ± 3.56	20.80 ± 3.26	22.25 ± 3.18	7.09 ± 1.93
	ESL	10.42 ± 1.71	10.75 ± 1.58	8.89 ± 1.60	6.82 ± 1.29
	MMG	16.97 ± 0.87	17.16 ± 1.40	18.40 ± 1.84	17.25 ± 1.20
	ERA	21.36 ± 2.05	20.93 ± 1.74	23.68 ± 1.87	28.89 ± 2.73
	LEV	16.74 ± 1.87	16.08 ± 1.73	16.54 ± 1.60	14.99 ± 1.22
	CEV	14.88 ± 1.35	–	13.00 ± 1.42	4.48 ± 0.89
	ASA	2.29 ± 1.09	–	3.69 ± 1.41	–
50% data set	DBS	16.23 ± 4.69	16.27 ± 4.26	14.80 ± 4.21	15.72 ± 4.16
	CPU	6.75 ± 2.37	6.40 ± 2.39	2.30 ± 2.38	4.64 ± 2.81
	BCC	27.50 ± 3.17	–	28.54 ± 2.46	26.87 ± 2.82
	MPG	17.81 ± 2.37	–	20.90 ± 2.36	5.77 ± 2.51
	ESL	10.04 ± 1.86	10.18 ± 1.55	7.83 ± 1.63	6.01 ± 1.26
	MMG	17.32 ± 1.51	–	17.58 ± 1.52	16.67 ± 1.44
	ERA	20.56 ± 1.73	19.58 ± 1.37	23.42 ± 1.71	28.44 ± 3.06
	LEV	15.92 ± 1.22	14.22 ± 1.54	15.56 ± 1.32	13.72 ± 1.25
	CEV	14.83 ± 0.95	–	13.24 ± 1.17	3.76 ± 0.59
	ASA	1.38 ± 0.61	–	2.47 ± 0.82	–
80% data set	DBS	15.92 ± 6.98	14.80 ± 8.11	12.80 ± 5.01	14.16 ± 6.81
	CPU	6.40 ± 3.04	5.98 ± 3.15	1.52 ± 2.14	2.12 ± 3.01
	BCC	26.77 ± 5.47	–	29.13 ± 5.10	24.96 ± 4.85
	MPG	16.86 ± 3.69	–	20.80 ± 3.88	5.51 ± 1.60
	ESL	10.01 ± 2.97	10.08 ± 2.47	7.44 ± 2.35	5.42 ± 2.18
	MMG	16.98 ± 2.79	–	17.34 ± 2.65	15.84 ± 2.51
	ERA	20.31 ± 2.50	18.56 ± 2.60	23.56 ± 2.92	28.13 ± 2.80
	LEV	16.16 ± 2.22	13.59 ± 1.85	15.72 ± 2.22	13.14 ± 1.76
	CEV	15.06 ± 1.66	–	13.36 ± 1.67	2.73 ± 0.89
	ASA	1.16 ± 1.74	–	2.11 ± 1.02	–

In the case of UTADIS, the value of AUC is also computed through formula (5), but $\tau(a^i, a^k)$ is defined differently: it compares the values of the alternatives, that is,

$$\tau(a^i, a^k) = \begin{cases} 0 & \text{if } u(a^i) < u(a^k) \\ 0.5 & \text{if } u(a^i) = u(a^k) \\ 1 & \text{if } u(a^i) > u(a^k). \end{cases}$$

5.3.1. Results

Table 2 shows the average 0/1 loss obtained on the test sets with the learned models. Table 3 shows the average value of the AUC. Each entry in these tables records the average value and standard deviation for 100 random splits of the data sets into learning and test sets.

Table 3
Average and standard deviation of the *AUC* (in percent) of the test set for learning sets of different sizes

Learning set	Data set	META	MIP	UTADIS	CR
20% data set	DBS	87.61 ± 4.62	86.37 ± 4.63	88.86 ± 4.96	92.90 ± 3.22
	CPU	95.31 ± 2.47	94.97 ± 2.62	97.89 ± 2.83	98.22 ± 1.21
	BCC	68.10 ± 4.58	71.55 ± 3.65	66.50 ± 5.27	64.00 ± 6.41
	MPG	83.37 ± 2.91	82.15 ± 3.68	81.62 ± 3.35	97.88 ± 1.60
	ESL	95.69 ± 1.14	95.10 ± 1.66	97.04 ± 0.95	96.70 ± 0.74
	MMG	88.28 ± 1.29	88.77 ± 1.51	86.50 ± 2.94	88.67 ± 1.23
	ERA	72.56 ± 2.38	71.82 ± 3.28	74.09 ± 1.75	76.69 ± 3.34
	LEV	85.30 ± 2.58	84.24 ± 2.91	87.07 ± 1.46	89.71 ± 0.98
	CEV	89.68 ± 1.16	–	92.35 ± 1.83	98.25 ± 0.80
	ASA	98.11 ± 1.25	–	98.73 ± 0.90	–
50% data set	DBS	90.74 ± 3.66	89.98 ± 3.36	93.25 ± 3.45	93.41 ± 2.28
	CPU	97.01 ± 1.40	96.45 ± 1.94	99.40 ± 1.31	99.20 ± 0.73
	BCC	69.29 ± 3.98	–	66.50 ± 52.7	69.12 ± 4.69
	MPG	83.37 ± 2.31	–	82.72 ± 2.43	98.18 ± 0.75
	ESL	96.40 ± 0.99	95.63 ± 1.14	97.47 ± 1.16	97.20 ± 0.84
	MMG	88.62 ± 1.38	–	86.67 ± 3.85	90.03 ± 1.32
	ERA	73.66 ± 2.33	71.67 ± 2.74	74.37 ± 2.11	77.05 ± 3.10
	LEV	87.21 ± 1.47	85.11 ± 2.19	87.46 ± 1.37	90.98 ± 1.03
	CEV	89.60 ± 0.73	–	93.39 ± 1.38	99.12 ± 0.24
	ASA	99.21 ± 0.72	–	99.48 ± 0.34	–
80% data set	DBS	90.19 ± 6.06	90.80 ± 6.73	94.76 ± 4.01	94.27 ± 4.43
	CPU	97.21 ± 2.19	96.56 ± 2.37	99.89 ± 0.30	99.71 ± 0.63
	BCC	70.56 ± 8.64	–	66.51 ± 6.59	73.49 ± 6.92
	MPG	86.13 ± 3.41	–	82.10 ± 4.34	98.55 ± 1.08
	ESL	96.13 ± 1.70	95.68 ± 1.65	97.78 ± 1.17	97.66 ± 1.50
	MMG	88.60 ± 2.65	–	86.82 ± 4.70	91.35 ± 2.33
	ERA	73.79 ± 3.51	72.42 ± 4.77	74.97 ± 4.02	76.70 ± 2.90
	LEV	86.63 ± 2.65	84.99 ± 3.32	87.41 ± 2.17	91.22 ± 2.02
	CEV	89.41 ± 1.35	–	93.99 ± 1.11	99.59 ± 0.27
	ASA	99.55 ± 0.64	–	99.64 ± 0.34	–

In these tables, column "learning set" displays the percentage of alternatives of the data set used by the algorithms as learning set. Column "META" shows the results obtained with the metaheuristic described in this paper. Column "MIP" contains the results obtained with the MIP described in Leroy et al. (2011). Column "UTADIS" displays the results obtained with UTADIS, and column "CR" contains the results obtained with the CR (Tehrani et al., 2012) on all the data sets, except ASA (not available).

In column "MIP," some cells are empty because the solver was not able to find a solution for at least one test instance in less than one hour. As compared to solving the MIP formulation, for the largest data set, that is, the CEV data set, the metaheuristic uses 50 seconds on average to find a model when the learning set consists of 80% of all the examples in the data set.

To assess the ability of the algorithm to find models restoring the assignment of a large number of examples, we compare the average 0/1 loss and *AUC* obtained with the MIP and the metaheuristic

for the test sets. Note that the MIP finds a MR-Sort model that is compatible with the largest possible number of examples from the learning set. There is no other MR-Sort model restoring correctly more assignment examples.

Table 2 shows that the 0/1 loss obtained by the exact algorithm is on average 1% smaller than by the metaheuristic. This is due to the fact that the MIP finds models restoring an optimal number of assignment examples (from the learning set), while the metaheuristic can remain stuck in local minima. However, this better performance does not hold for all data sets when applied to the test set. For instance, the MIP returns results slightly worse than the metaheuristic for the DBS and ESL data sets. This is probably due to an overfitting effect on the learning set.

We observe in Table 3 that the average *AUC* of the metaheuristic is close to the one of the MIP. This indicates that the quality of the classifiers obtained with the MIP and the metaheuristic are similar.

In order to see whether the algorithm proposed in this paper can be useful in the context of PL problems, we compare our results with two other methodologies: UTADIS (Jacquet-Lagrèze and Siskos, 1982; Doumpos and Zopounidis, 2002) and CR (Tehrani et al., 2012).

The performance of MR-Sort algorithms is close to the performance of UTADIS and CR for the DBS, CPU, BCC, MMG, and LEV data sets. The 0/1 losses observed on the test set differ by at most 4% on average. For the MPG and CEV data sets, CR clearly returns better average results (more than 10% better in terms of 0/1 loss) than the MR-Sort algorithms and UTADIS. This may be due to the capability of the Choquet integral to represent interactions between criteria (see, for example, Grabisch and Roubens, 2000). The assignments in the MPG and CEV data sets might require this type of modeling feature.

In contrast, for the ERA data set, the MR-Sort algorithms and UTADIS are definitely better than CR. Their advantage regarding the average 0/1 loss amounts to almost 8%.

As compared with UTADIS and CR, the average *AUC* value of the MR-Sort algorithms are worse. The difference is about 5% for DBS, CPU, BCC, ESL, MMG, and ERA data sets. For the MPG and CEV data sets, there is a marked advantage of CR over the other algorithms. We have seen that CR is also definitely better regarding 0/1 loss for the MPG data set, which suggests that the model underlying CR is better suited for representing the MPG data.

On the contrary, the average *AUC* of the ERA data set, for which the MR-Sort MIP and metaheuristic did better than CR in term of 0/1 loss, is worse with the MR-Sort MIP and metaheuristic than with CR. UTADIS does even better than MR-Sort algorithms in terms of *AUC* for this data set.

In order to better understand the latter results, the confusion matrices for the MR-Sort algorithms and UTADIS and all learning set sizes are displayed in Table 4 for the ERA data set (the confusion matrices relative to the other data sets can be found in Sobrie et al. 2015). These show the average distribution of the alternatives in the test sets in actual (C_1, C_2) versus predicted classes $(\hat{C}_1, \hat{C}_2)$. As compared with UTADIS, the MR-Sort algorithms classify correctly, on average, a higher number of instances from class C_1 and a lower number of instances from class C_2. We also note that there are, on average, more alternatives belonging to class C_2 than to class C_1. Alternatives misclassified by the MR-Sort algorithms mostly belong to category C_2. It is likely that the concordance index of some of these alternatives is equal or lower than that of some alternatives that are correctly classified in C_1. The contribution of these alternatives to the *AUC* index is therefore equal to 0.5 or 0, which decreases the value of the *AUC*.

Table 4
Confusion matrices of the test set for the (binarized) ERA data set

(a) META – ERA 20 %

	$\hat{C}_1$	$\hat{C}_2$
C_1	68.83 ±3.11	5.67 ±3.21
C_2	15.69 ±2.39	9.81 ±2.16

(b) MIP – ERA 20 %

	$\hat{C}_1$	$\hat{C}_2$
C_1	69.38 ±2.41	5.12 ±2.47
C_2	15.81 ±1.74	9.70 ±1.54

(c) UTADIS – ERA 20 %

	$\hat{C}_1$	$\hat{C}_2$
C_1	63.98 ±2.67	10.54 ±2.96
C_2	13.14 ±1.56	12.34 ±1.25

(d) META – ERA 50 %

	$\hat{C}_1$	$\hat{C}_2$
C_1	69.26 ±2.61	5.10 ±2.55
C_2	15.46 ±2.16	10.17 ±1.73

(e) MIP – ERA 50 %

	$\hat{C}_1$	$\hat{C}_2$
C_1	71.23 ±2.01	3.34 ±1.74
C_2	16.24 ±1.41	9.18 ±1.11

(f) UTADIS – ERA 50 %

	$\hat{C}_1$	$\hat{C}_2$
C_1	64.36 ±2.12	9.98 ±2.62
C_2	13.43 ±1.75	12.22 ±1.32

(g) META – ERA 80 %

	$\hat{C}_1$	$\hat{C}_2$
C_1	69.60 ±3.42	5.03 ±2.38
C_2	15.28 ±2.89	10.09 ±2.26

(h) MIP – ERA 80 %

	$\hat{C}_1$	$\hat{C}_2$
C_1	72.59 ±2.57	2.29 ±1.00
C_2	16.27 ±2.36	8.86 ±1.77

(i) UTADIS – ERA 80 %

	$\hat{C}_1$	$\hat{C}_2$
C_1	63.90 ±3.17	10.21 ±2.85
C_2	13.35 ±2.58	12.55 ±2.32

Actual class in rows, predicted class in columns.

It should be noted that the MR-Sort algorithms are designed in view of minimizing the 0/1 loss. They do not include specific mechanisms taking into account possible imbalance of classes in the learning set, which has an impact on AUC.

5.3.2. Comments

Computing time becomes quickly an issue with the MIP when the size of the learning set increases. It is therefore not an option to use it to deal with large data sets, which, in contrast, can easily be handled by the metaheuristic.

The metaheuristic, we developed performs better than UTADIS and CR for at least one data set (ERA). The same observation holds for the MIP. Regarding the 0/1 loss, we note that the MR-Sort model seems particularly well adapted for the ASA data set. This shows that for some types of data sets, a model-based approach like MR-Sort is well suited.

5.4. Model interpretation

An important feature of model-based PL is interpretability. This section aims to illustrate on the (binarized) ESL data set how an MR-Sort model may provide an interpretation of the classification rationale.

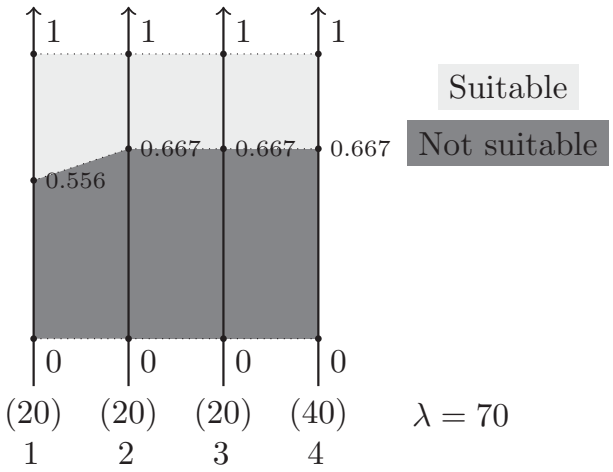

Fig. 3. An MR-Sort model obtained for the (binarized) ESL data set by using 50% of the instances as the learning set. The profile delimiting category "Suitable" from category "Not suitable" is represented by the dotted line. Criteria weights are displayed in parentheses below each criterion axis. The majority threshold is denoted by λ.

The ESL data set is composed of 488 instances that are evaluated on four attributes, denoted 1, 2, 3, and 4. Each instance corresponds to the profile of an applicant for a job. The applicants were evaluated by psychologists on the basis of psychometric tests and interviews. Attribute scales are normalized between 0 and 1. An overall score between 1 and 9 is assigned to each candidate. It represents the degree of suitability of the applicant for the job. The data set was "binarized," separating the applicants in two classes: "suitable" candidates have a score comprised between 6 and 9, while "not suitable" ones have a score equal to or lower than 5.

The model described in Fig. 3 was obtained using 50% of the instances as learning set and the rest as test set. This model is able to represent 93% of the assignments of the learning set instances with an AUC equal to 97.42%. In generalization, the model classifies correctly 88.11% of the test instances with an AUC of 95.34%.

On each criterion, the profile value separates the evaluations that contribute to assigning the candidates to the "suitable" class from the others. The assignment rule underlying the model presented in Fig. 3 has a simple formulation. A candidate is assigned to the "suitable" class if its performances are better than or equal to these of the profile on at least one of the three following criteria coalitions: {1, 2, 4}, {1, 3, 4}, or {2, 3, 4}. The sum of the criteria weights belonging to each of these coalitions is always greater than the majority threshold, set to 70. We have $w_1 + w_2 + w_4 = w_1 + w_3 + w_4 = w_2 + w_3 + w_4 = 80$. In other words, to be considered as "suitable," an applicant may have only one weakness, on one of the first three dimensions. As an illustration, a candidate assessed by the performance vector (0.889, 0.889, 0.5, 0.833) is considered "suitable" for the job since its evaluation is worse than the profile only on criterion 3 ($0.5 < 0.667$). On the contrary, an applicant characterized by the performance vector (0.222, 0.889, 0.5, 0.833) is considered "not suitable" for the job since it lies under the profile level on criteria 1 and 3 ($0.222 < 0.556$ on criterion 1 and $0.5 < 0.667$ on criterion 3).

The interested reader will find in Sobrie et al. (2016) a medical application of the MR-Sort method in the domain of preanesthesia patient assessment; Sobrie et al. (2016) also discuss the issue of interpretability of MR-Sort models.

6. Conclusion

In this paper, we proposed and studied a method for learning the assignment of objects evaluated on several attributes (or criteria) into ordered categories. This method is based on a well-understood preference model (Bouyssou and Marchant, 2007a, 2007b). For assigning objects, it takes into account the evaluation of the objects in an ordinal manner, that is, only the relative position of the evaluations w.r.t a vector of minimal requirements ("profile") matters.

The heuristic algorithm proposed for learning such a model on the basis of assignment examples was evaluated on a data sets benchmark. The first observation is that our heuristic provides good approximations of the MR-Sort model that can be learned by an exact method (MIP), whenever the latter can be computed in a reasonable amount of time. For most data sets, the classification performance of our algorithm is close to the best results obtained by state-of-the-art algorithms. It is definitely better for one of the data sets. Since the MR-Sort model relies on a specific form of regularity in the assignments, it is not surprising that some data sets can be better approximated using our algorithm than some others. What is worth noticing, actually, is that our heuristic behaves competitively on the whole benchmark.

Another positive feature of the MR-Sort model stems from the fact that the computed classifications can be explained to the user as the application of a compact and intuitive rule (see, e.g., Sobrie et al., 2016). This is linked with the origins of the model that has been initially used in preference *modeling* and decision aiding. In these domains, preferences are modeled by engaging into interactions with a DM (instead of being learned automatically on the basis of examples). Therefore, the preference models rely on intuitive concepts (e.g., limit profile, weights, majority threshold), which are used in the preference elicitation process. The resulting rules for comparing or sorting objects can be formulated in terms of the same concepts, which allows to explain their consequences to the DM. Understanding the model issued from an algorithm is likely to increase the trust of the user in the obtained classifier. Explainability is important, for example, in medical applications, but also in management and engineering applications.

Several extensions of the MR-Sort model, which enhance its expressivity without impairing its explainability, have yet to be explored in a PL perspective. One extension consists in introducing the possibility of vetoes. A *veto* (Bouyssou and Marchant, 2007a; Roy and Bouyssou, 1993) forbids an object from being assigned to a category if its performance is too much below the lower limit profile of the category on some criterion. Clearly, assignment rules combining a concordance condition (such as (2)) with a nonveto condition mitigates the purely ordinal interpretation of the criteria by creating additional anchor values. Besides the lower limit profile value on each criterion, we have to specify another value, beneath the profile value, which constitutes a minimal requirement in order to be eligible for assignment in a category.

Another extension of the MR-Sort model is known (Bouyssou and Marchant, 2007a, 2007b) as the noncompensatory sorting model (NCS). In this model, which may also accommodate nonveto conditions, the criteria weights are substituted by a *capacity*, which allows criteria interactions

(positive/negative synergies) to be modeled. These two types of extensions are the subject of ongoing investigations (see Sobrie et al., 2016; Meyer and Olteanu, 2017).

References

Belton, V., Stewart, T., 2002. *Muliple Criteria Decision Analysis: An Integrated Approach*. Kluwer Academic, Dordrecht.

Ben-David, A., 1995. Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning* 19, 1, 29–43.

Ben-David, A., Sterling, L., Pao, Y.-H., 1989. Learning and classification of monotonic ordinal concepts. *Computational Intelligence* 5, 1, 45–49.

Bous, G., Fortemps, P., Glineur, F., Pirlot, M., 2010. ACUTA: a novel method for eliciting additive value functions on the basis of holistic preference statements. *European Journal of Operational Research* 206, 2, 435–444.

Bouyssou, D., Marchant, T., 2007a. An axiomatic approach to noncompensatory sorting methods in MCDM, I: the case of two categories. *European Journal of Operational Research* 178, 1, 217–245.

Bouyssou, D., Marchant, T., 2007b. An axiomatic approach to noncompensatory sorting methods in MCDM, II: more than two categories. *European Journal of Operational Research* 178, 1, 246–276.

Bouyssou, D., Pirlot, M., 2005. A characterization of concordance relations. *European Journal of Operational Research* 167, 2, 427–443.

Bouyssou, D., Pirlot, M., 2007. Further results on concordance relations. *European Journal of Operational Research* 181, 505–514.

Bouyssou, D., Pirlot, M., 2015. A consolidated approach to the axiomatization of outranking relations: a survey and new results. *Annals of Operations Research* 229, 1, 159–212.

Cailloux, O., Meyer, P., Mousseau, V., 2012. Eliciting ELECTRE TRI category limits for a group of decision makers. *European Journal of Operational Research* 223, 1, 133–140.

Chandrasekaran, R., Ryu, Y.U., Jacob, V.S., Hong, S., 2005. Isotonic separation. *INFORMS Journal on Computing* 17, 4, 462–474.

Corrente, S., Greco, S., Kadzinski, M., Słowiński, R., 2013. Robust ordinal regression in preference learning and ranking. *Machine Learning* 93, 2–3, 381–422.

Dembczyński, K., Kotlowski, W., Słowiński, R., 2006. Additive preference model with piecewise linear components resulting from dominance-based rough set approximations. In Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds) *ICAISC*, Volume 4029 of *Lecture Notes in Computer Science*. Springer, Berlin, pp. 499–508.

Dembczyński, K., Kotlowski, W., Słowiński, R., 2009. Learning rule ensembles for ordinal classification with monotonicity constraints. *Fundamenta Informaticae* 94, 2, 163–178.

Dias, L., Mousseau, V., Figueira, J., Clímaco, J., 2002. An aggregation/disaggregation approach to obtain robust conclusions with ELECTRE TRI. *European Journal of Operational Research* 138, 1, 332–348.

Doumpos, M., Marinakis, Y., Marinaki, M., Zopounidis, C., 2009. An evolutionary approach to construction of outranking models for multicriteria classification: the case of the ELECTRE TRI method. *European Journal of Operational Research* 199, 2, 496–505.

Doumpos, M., Zopounidis, C., 2002. *Multicriteria Decision Aid Classification Methods*. Kluwer Academic, Dordrecht.

Doumpos, M., Zopounidis, C., Galariotis, E., 2014. Inferring robust decision models in multicriteria classification problems: an experimental analysis. *European Journal of Operational Research* 236, 2, 601–611.

Figueira, J.R., Greco, S., Roy, B., Słowiński, R., 2013. An overview of ELECTRE methods and their recent extensions. *Journal of Multi-Criteria Decision Analysis* 20, 1–2, 61–85.

Fürnkranz, J., Hüllermeier, E., 2010. Preference learning: an introduction. In Fürnkranz, J., Hüllermeier, E. (eds) *Preference Learning*. Springer-Verlag, Berlin, Heidelberg, pp. 1–17.

Grabisch, M., Roubens, M., 2000. Application of the Choquet integral in multicriteria decision making. In Grabisch, M., Murofushi, T., Sugeno, M., (eds) *Fuzzy Measures and Integrals—Theory and Applications*. Physica Verlag, Heidelberg, pp. 348–374.

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

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

Jacquet-Lagrèze, E., Siskos, Y., 2001. Preference disaggregation: 20 years of MCDA experience. *European Journal of Operational Research* 130, 2, 233–245.

Lazouni, M.E.A., El Habib Daho, M., Settouti, N., Chikh, M.A., Mahmoudi, S., 2013. Machine learning tool for automatic ASA detection. In Amine, A., Aït Mohamed, O., Bellatreche, L. (eds) *Modeling Approaches and Algorithms for Advanced Computer Applications*, volume 488 of *Studies in Computational Intelligence*. Springer, Berlin, pp. 9–16.

Leroy, A., Mousseau, V., Pirlot, M., 2011. Learning the parameters of a multiple criteria sorting method. In Brafman, R., Roberts, F., Tsoukiàs, A. (eds) *Algorithmic Decision Theory, Lecture Notes in Computer Science*, Vol. 6992. Springer, Berlin, pp. 219–233.

Meyer, P., Olteanu, A.L., 2017. Integrating large positive and negative performance differences into multicriteria majority-rule sorting models. *Computers & Operations Research* 81, 216–230.

Minoux, M., 2008. *Programmation Mathématique: Théorie et Algorithmes* (2nd edn). Collection technique et scientifique des télécommunications, Dunod.

Mousseau, V., Figueira, J., Naux, J.P., 2001. Using assignment examples to infer weights for ELECTRE TRI method: some experimental results. *European Journal of Operational Research* 130, 1, 263–275.

Mousseau, V., Słowiński, R., 1998. Inferring an ELECTRE TRI model from assignment examples. *Journal of Global Optimization* 12, 1, 157–174.

Ngo The, A., Mousseau, V., 2002. Using assignment examples to infer category limits for the ELECTRE TRI method. *Journal of Multi-criteria Decision Analysis* 11, 1, 29–43.

Pirlot, M., 1996. General local search methods. *European Journal of Operational Research* 92, 3, 493–511.

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

Sill, J., 1998. Monotonic networks. *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, Cambridge, MA, pp. 661–667.

Sobrie, O., 2016. *Learning preferences with multiple-criteria models*. PhD thesis, Université de Mons, Belgium and Université Paris Saclay, France.

Sobrie, O., Lazouni, M.E.A., Mahmoudi, S., Mousseau, V., Pirlot, M., 2016. A new decision support model for preanesthetic evaluation. *Computer Methods and Programs in Biomedicine* 133, 183–193.

Sobrie, O., Mousseau, V., Pirlot, M., 2013. Learning a majority rule model from large sets of assignment examples. In Perny, P., Pirlot, M., Tsoukiàs, A. (eds) *Algorithmic Decision Theory*, Third International Conference, ADT 2013, Lecture Notes in Artificial Intelligence, Vol. 8176. Springer, Brussels, Belgium, pp. 336–350.

Sobrie, O., Mousseau, V., Pirlot, M., 2015. A majority rule sorting model to deal with monotone learning sets. Research report 2015-02, École Centrale Paris. Available at http://www.lgi.ecp.fr/Biblio/PDF/CR-LGI-2015-02.pdf (last accessed February 5, 2018).

Sobrie, O., Mousseau, V., Pirlot, M., 2017. A population-based algorithm for learning a majority rule sorting model with coalitional veto. In Trautmann, H., Rudolph, G., Klamroth, K., Schütze, O., Wiecek, M., Jin, Y., Grimme, C. (eds), *Proceedings of the 9th international conference on Evolutionary Multi-Criterion Optimization,* volume 10173 of *Lecture Notes in Computer Science*. Springer, Berlin, pp. 575–589.

Tehrani, A.F., Cheng, W., Dembczyński, K., Hüllermeier, E., 2012. Learning monotone nonlinear models using the Choquet integral. *Machine Learning* 89, 1–2, 183–211.

Wolsey, L.A., 1998. *Integer Programming*. Wiley Series in Discrete Mathematics and Optimization. Wiley, New York.

Yu, W., 1992. *Aide multicritère à la décision dans le cadre de la problématique du tri: méthodes et applications*. PhD thesis, LAMSADE, Université Paris Dauphine, Paris.

Zopounidis, C., Doumpos, M., 2002. Multicriteria classification and sorting methods: a literature review. *European Journal of Operational Research* 138, 2, 229–246.