Learning a majority rule model from large sets of assignment examples

Olivier Sobrie[1,2], Vincent Mousseau[1], and Marc Pirlot[2]

[1] École Centrale Paris
Grande Voie des Vignes
92295 Châtenay Malabry, France
olivier.sobrie@gmail.com, vincent.mousseau@ecp.fr
[2] Université de Mons, Faculté Polytechnique
9, rue de Houdain
7000 Mons, Belgique
marc.pirlot@umons.ac.be

Abstract. Learning the parameters of a Majority Rule Sorting model (MR-Sort) through linear programming requires to use binary variables. In the context of preference learning where large sets of alternatives and numerous attributes are involved, such an approach is not an option in view of the large computing times implied. Therefore, we propose a new metaheuristic designed to learn the parameters of an MR-Sort model. This algorithm works in two phases that are iterated. The first one consists in solving a linear program determining the weights and the majority threshold, assuming a given set of profiles. The second phase runs a metaheuristic which determines profiles for a fixed set of weights and a majority threshold. The presentation focuses on the metaheuristic and reports the results of numerical tests, providing insights on the algorithm behavior.

1 Introduction

Multiple criteria sorting procedures aim at assigning alternatives evaluated on multiple criteria to a category selected in a set of pre-defined and ordered categories. In this article we investigate the Majority Rule Sorting procedure (MR-Sort), a simplified version of the ELECTRE TRI sorting model [1, 2]. MR-Sort is directly inspired by the work of Bouyssou and Marchant who provide an axiomatic characterization [3, 4] of non-compensatory sorting methods. The general principle of MR-Sort is to assign alternatives by comparing their performances to those of profiles delimiting the categories. An alternative is assigned to a category "above" a profile if and only if it is at least as good as the profile on a (weighted) majority of criteria.

For using MR-Sort, several parameters need to be determined: the performance vector associated to each profile, the criteria weights and a majority threshold. It is not easy for a decision maker (DM) to assess such parameters. He often prefers to provide typical examples of assignments of alternatives to

categories. Several papers have been devoted to learning the parameters of such models on the basis of assignment examples. Mathematical programming techniques for learning part or all the parameters of an ELECTRE TRI model are described in [5–9] while [10] proposes a genetic algorithm designed for the same purpose. Learning the parameters of a MR-Sort model is dealt with in [11, 12].

None of these proposals can be considered suitable to our case, since we want to deal with *large* sets of assignment examples, having in mind the kind of sorting problems encountered in the field of preference learning [13], more precisely in the monotone learning subfield. [11] needs computing times as long as 25 seconds to learn the parameters of a MR-Sort model involving 5 criteria and 3 categories from a learning set containing 100 examples. This is no wonder since their algorithm is based on the resolution of a mixed integer program (MIP) in which the number of binary variables grows linearly with the number of examples. The experimental results in [11] show that the computing time increases very quickly with the number of examples.

From the previous work related to parameters learning for ELECTRE TRI models, we retain two main lessons. Firstly, learning only the weights and the majority threshold of a MR-Sort model can be done by solving a linear program without binary variables as done in [6]. On the other hand, as demonstrated in [7], learning only the profiles of such models by means of linear programming does require binary variables.

Based on these observations, we have designed an algorithm that computes the parameters of a MR-Sort model in order to assign as many as possible alternatives in the learning set to their category. This algorithm has two main components that are used repeatedly and alternatively. The first component learns optimal weights and majority threshold, in case the profiles limiting the categories are fixed. The second component adjusts the profiles for given weights and majority threshold.

To assess the new algorithm, we have set up a series of numerical experiments much in the spirit of [11]. The simulation experiments were designed in order to address the following questions:

Algorithm performance Given a MR-Sort model involving n criteria and p categories and a set of assignment examples compatible with this model, how fast does the algorithm find parameters of an MR-Sort model which restores the original assignments ?

Model Retrieval Given a MR-Sort model involving n criteria and p categories and a set of assignment examples compatible with this model, how many examples are required to obtain a model that is close to the original one?

Tolerance for errors Given a set of assignments, obtained through a MR-Sort model, in which errors have been added, to what extent do the errors perturb the algorithm?

Idiosyncrasy Each alternative of a set is assigned to a category by a rule that is not a MR-Sort rule (actually, they are assigned by an additive sorting rule). What's the ability of the algorithm to find a MR-Sort model restoring

as many examples as possible ? In other terms, is MR-Sort flexible enough to reproduce assignments by another type of sorting rule?

In the next section of this paper, we briefly recall the precise definition of the MR-Sort procedure. In section 3, we describe the algorithm that we have developed. Numerical experiments designed for testing the algorithm are described, their results summarized and commented on in section 4. We finally conclude this paper with some perspectives for further research in view of improving the current version of the algorithm.

2 MR-Sort procedure

The MR-Sort procedure is a simplified version of the ELECTRE TRI procedure [1, 2], based on the work of Bouyssou and Marchant developed in [3, 4].

Let X be a set of alternatives evaluated on n criteria, $F = \{1, 2, ..., n\}$. We denote by a_j the performance of alternative $a \in X$ on criterion j. The categories of the MR-Sort model, delimited by the profiles b_{h-1} and b_h, are denoted by C_h, where h denotes the category index. We convene that the best category C_p is delimited by a fictive upper profile, b_p, and the worst one by a fictive lower profile, b_0. The performances of the profiles are denoted by $b_{h,j}$, with $j = 1, ..., n$. It is assumed that the profiles dominate one another, i.e.:

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

Using the MR-Sort procedure (without veto), an alternative is assigned to a category if its performances are at least as good as the performances of the category's lower profile and worse than the performances of the category's upper profile on a weighted majority of criteria. In the former case, we say that the alternative is *preferred* to the profile, while, in the latter, it is not. Formally, an alternative $a \in X$ is *preferred* to profile b_h, and we denote it by aSb_h, if the following condition is met:

$$aSb_h \Leftrightarrow \sum_{j:a_j \geq b_{h,j}} w_j \geq \lambda,$$

where w_j for $j \in F$ are nonnegative weights attached to the criteria and satisfying the normalization condition $\sum_{j \in F} w_j = 1$; λ is the *majority threshold*; it satisfies $\lambda \in [1/2, 1]$. The preference relation S can be seen as an *outranking* relation without veto [2, 14, 15].

The condition for an alternative $a \in X$ to be assigned to category C_h is expressed as follows:

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

The MR-Sort assignment rule described above involves $pn + 1$ parameters, i.e. 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 $j \in F$; $b_{p,j}$ is a value such that $a_j < b_{p,j}$ for all $a \in X$ and $j \in F$.

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

3 The algorithm

3.1 Learning all the parameters

As demonstrated in [11], the problem of learning the parameters of a MR-Sort model on the basis of assignment examples can be formulated as a mixed integer program (MIP) but only instances of modest size can be solved in reasonable computing times. The MIP proposed in [11] contains $m \cdot (2n + 1)$ binary variables, with n, the number of criteria, and m, the number of alternatives. A problem involving 1000 alternatives, 10 criteria and 5 categories requires 21000 binary variables. For a similar program in [12], it is mentioned that problems with less than 400 binary variables can be solved within 90 minutes. Following these observations, we understand that MIP is not suitable for the applications we want to deal with. In [10], a genetic algorithm was proposed to learn the parameters of an ELECTRE TRI model. This algorithm could be transposed for learning the parameters of a MR-Sort model. However, it is well known [16] that genetic algorithms which take the structure of the problem into account to perform crossovers and mutations give better results. It is not the case of the genetic algorithm proposed in [10] since the authors' definitions of crossover and mutation operators are standard.

Learning only the weights and the majority threshold of an MR-Sort model on the basis of assignment examples can be done using an ordinary linear program (without binary or integer variables). On the contrary, learning profiles evaluations is not possible by linear programming without binary variables. Taking these observations into account, we propose an algorithm that takes advantage of the ease of learning the weights and the majority threshold by a linear program and adjusts the profiles by means of a dedicated heuristic.

The algorithm uses the following components (see Algorithm 1):

1. a heuristic for initializing the profiles;
2. a linear program learning the weights and the majority threshold, given the profiles;
3. a dedicated heuristic adjusting the profiles, given weights and a majority threshold.

In the next subsections we describe in more detail these three elements. The algorithm uses the latter two components iteratively: starting from initial profiles, we find the optimal weights and threshold for these profiles by solving a linear

program; then we adjust the profiles, using the heuristic, keeping the weights and threshold fixed; the profile adjustment operation is repeated N_{it} times. We call *main loop* the optimization of the weights and threshold followed by N_{it} iterations of the profiles adjustment operation. The main loop is executed until a stopping criterion is met.

Since the process of alternating the optimization of the weights and threshold, on the one hand, and several iterations of the (heuristic) optimization of the profiles, on another hand, is not guaranteed to converge to a good set of parameters, we implement the algorithm as an *evolutionary metaheuristic*, evolving, not a single MR-Sort model, but a population of them. The number of models in the population is denoted by N_{model}. After each application of the main loop to all models in the population, we assess the resulting population of models by using them to assign the alternatives in the learning set. The quality of a MR-Sort model is assessed by its *classification accuracy*:

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

At this stage, the algorithm reinitializes the $\left\lfloor \frac{N_{model}}{2} \right\rfloor$ models giving the worst CA. The stopping criterion of the algorithm is met either once the classification accuracy of some model in the population is equal to 1 or after a maximum number of iterations N_o (fixed a priori).

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

```
Generate a population of N_model models with profiles initialized with a 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 metaheuristic N_it times, using the current weights
    and threshold.
  end for
  Reinitialize the ⌊N_model/2⌋ models giving the worst CA
until Stopping criterion is met
```

3.2 Profiles initialization

The first step of the algorithm consists in initializing a set of profiles so that it can be used to learn a set of weights and a majority threshold. The general idea of the heuristic we designed to set the value $b_{h,j}$ of the profile b_h on criterion j is the following. We choose this value 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 profile values, the proportion of examples assigned to a category is taken into account so that the alternatives assigned to categories which are not often represented in the learning set have more importance. Note the initial value of a profile b_h is determined by only considering the examples assigned to the category just below and just above the profile i.e. the examples belonging respectively to the subsets A_h and A_{h+1} in the learning set A. 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. For guaranteeing an equal treatment of all profiles, we chose to consider only C_h and C_{h+1} for determining b_h.

The heuristic works as follows:

1. For each category C_h, compute the frequency π_h with which alternatives a in the learning set are assigned to category $C_h : \pi_h = \frac{|A_h|}{|A|}$.
2. For each criterion, the value of the profile $b_{h,j}$ is chosen s.t.:

$$\begin{aligned}\max_{b_{h,j}} \left[|a \in A_{h+1} : a_j \geq b_{h,j}| - |a \in A_{h+1} : a_j < b_{h,j}|\right](1 - \pi_{h+1}) \\ + \left[|a \in A_h : a_j < b_{h,j}| - |a \in A_h : a_j \geq b_{h,j}|\right](1 - \pi_h).\end{aligned}$$

The profiles are computed in descending order.

3.3 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, i.e. $b_{h+1,j} \geq b_{h,j}$ for all h and j, and the inequality is strict for at least one j. The constraints derived from the alternatives assignments are expressed as follows:

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

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, i.e. 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 do so may be due to possible compensatory effects between constraints, i.e. 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 introducing binary variables indicating each violation of the assignment constraints. We do not consider such formulations in order to limit computing times.

3.4 Learning the profiles

Learning the profiles by using a mathematical programming formulation requires binary variables, leading to a mixed integer program [7]. As we want to deal with problems involving large learning sets, 10 criteria and 3 to 5 categories, MIP is not an option. Therefore we opt for a randomized heuristic algorithm which is described below.

Consider a model having 2 categories and 5 criteria and assume that two alternatives are misclassified by this model. The one, a', is assigned in category C_1 by the DM and in C_2 by the model, while the other one, a'', is assigned in category C_2 by the DM and in C_1 by the model. Assuming fixed weights and majority threshold, it means that the profile delimiting the two categories, is either too high or too low on one or several criteria. In Figure 1, the arrows show the direction in which moving the profile in order to favor the correct classification of a' or a''. $\delta_j^{a'}$ (resp. $\delta_j^{a''}$) denotes the difference between the profile value $b_{1,j}$ and the alternative evaluations a'_j (resp. a''_j) on criterion j.

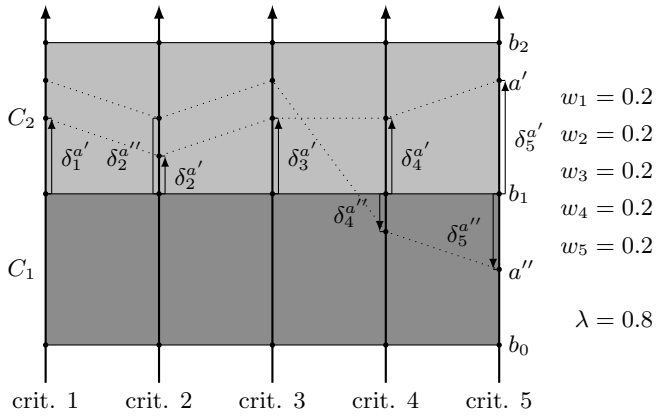


Fig. 1. Alternative wrongly assigned because of profile too low or too high

Based on these observations, we define several subsets of alternatives for each criterion j and each profile h and any positive value δ:

$V_{h,j}^{+\delta}$ **(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'' belongs to the set $V_{1,4}^{-\delta}$ on criterion 4 for $\delta \geq \delta_4^{a''}$.

$W_{h,j}^{+\delta}$ **(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 of $+\delta$ (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' belongs to the set $W_{1,1}^{+\delta}$ on criterion 1 for $\delta > \delta_1^{a'}$.

$Q_{h,j}^{+\delta}$ **(resp.** $Q_{h,j}^{-\delta}$**)** : the sets of alternatives correctly classified in C_{h+1} (resp. C_{h+1}) for which moving the profile b_h of $+\delta$ (resp. $-\delta$) on j results in a misclassification.

$R_{h,j}^{+\delta}$ **(resp.** $R_{h,j}^{-\delta}$**)** : 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 of $+\delta$ (resp. $-\delta$) on j weakens the criteria coalition in favor of the correct classification but does not induce misclassification by itself. For instance, a'' belongs to the set $R_{1,2}^{+\delta}$ on criterion 2 for $\delta > \delta_2^{a''}$.

$T_{h,j}^{+\delta}$ **(resp.** $T_{h,j}^{-\delta}$**)** : the sets of alternatives misclassified in a category higher than C_{h+1} (resp. in a category lower than C_h) for which the current profile evaluation weakens the criteria coalition in favor of the correct classification.

In order to formally define these sets we introduce the following notation. A_h^l denotes the subset of misclassified alternatives that are assigned in category C_l by the model while the DM assigns them in category C_h. $A_{<h}^{>l}$ denotes the subset of misclassified alternatives that are assigned in category higher than C_l by the model while the DM assigns them in a category below C_h. And similarly 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:

$$
\begin{aligned}
V_{h,j}^{+\delta} &= \left\{a \in A_h^{h+1} : b_{h,j} + \delta > a_j \geq b_{h,j} \text{ and } \sigma(a, b_h) - w_j < \lambda\right\} \\
V_{h,j}^{-\delta} &= \left\{a \in A_{h+1}^{h} : b_{h,j} - \delta < a_j < b_{h,j} \text{ and } \sigma(a, b_h) + w_j \geq \lambda\right\} \\
W_{h,j}^{+\delta} &= \left\{a \in A_h^{h+1} : b_{h,j} + \delta > a_j \geq b_{h,j} \text{ and } \sigma(a, b_h) - w_j \geq \lambda\right\} \\
W_{h,j}^{-\delta} &= \left\{a \in A_{h+1}^{h} : b_{h,j} - \delta < a_j < b_{h,j} \text{ and } \sigma(a, b_h) + w_j < \lambda\right\} \\
Q_{h,j}^{+\delta} &= \left\{a \in A_{h+1}^{h+1} : b_{h,j} + \delta > a_j \geq b_{h,j} \text{ and } \sigma(a, b_h) - w_j < \lambda\right\} \\
Q_{h,j}^{-\delta} &= \left\{a \in A_h^{h} : b_{h,j} - \delta < a_j < b_{h,j} \text{ and } \sigma(a, b_h) + w_j \geq \lambda\right\} \\
R_{h,j}^{+\delta} &= \left\{a \in A_{h+1}^{h} : b_{h,j} + \delta > a_j \geq b_{h,j}\right\} \\
R_{h,j}^{-\delta} &= \left\{a \in A_h^{h+1} : b_{h,j} - \delta < a_j < b_{h,j}\right\} \\
T_{h,j}^{+\delta} &= \left\{a \in A_{<h+1}^{>h+1} : b_{h,j} + \delta > a_j \geq b_{h,j}\right\} \\
T_{h,j}^{-\delta} &= \left\{a \in A_{>h}^{<h} : b_{h,j} - \delta < a_j \leq b_{h,j}\right\}
\end{aligned}
$$

To avoid violations of the dominance rule between the profiles, on each criterion j, $+\delta$ or $-\delta$ is chosen in the interval $[b_{h-1,j}, b_{h+1,j}]$. We define the value $P(b_{h,j}^{+\delta})$

which aggregates the number of alternatives contained in the sets described above as follows:

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

with k_V, k_W, k_T, k_Q, k_R, d_V, d_W, d_T, d_Q and d_R 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 are 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 below or slightly above a_j. The exact new position of the profile is chosen so as to favor a correct assignment for a.

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. 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 $P(b_{h,j}^{\Delta}) \leq r$.

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

Algorithm 2 summarizes how this randomized heuristic operates.

Algorithm 2 Randomized heuristic used for improving the profiles

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

4 Numerical experiments

4.1 Performance of the algorithm

Our first concern is to measure the performance of the algorithm and its convergence, i.e. how many iterations are needed to find a model restoring a majority of assignment examples and how much time is required to learn this model? To measure this, an experimental framework is set up:

1. An MR-Sort model M is generated randomly. The weights are uniformly generated as described in [17], i.e. $n-1$ random numbers are uniformly drawn from the interval $[0,1]$ and ranked s.t. $r_n = 1 > r_{n-1} \geq ... \geq r_1 > 0 = r_0$. Then weights are defined as follows: $w_j = r_j - r_{j-1}$, with $j = 1, ..., n$. The majority threshold is uniformly drawn from the interval $[1/2, 1]$. For the profiles evaluations, on each criterion $p-1$ random numbers are uniformly drawn from the interval $[0,1]$ and ordered s.t. $r'_{p-1} \geq ... \geq r'_1$. Profiles evaluations are determined by $b_{h,j} = r'_h$, $h = 1, ..., p-1$. Using model M as described by (1), each alternative can be assigned to a category. The resulting assignment rule is referred to as s_M.
2. A set of m alternatives with random performances on the n criteria is generated. The performances are uniformly and independently drawn from the $[0,1]$ interval. The set of generated alternatives is denoted by A. The alternatives in A are assigned using the rule s_M. The resulting assignments and the performances of the alternatives in the set A are given as input to the algorithm. They constitute the learning set.
3. On basis of the assignments and the performances of the alternatives in A, the algorithm learns a MR-Sort model which maximizes the classification accuracy. The model learned by the metaheuristic is denoted by M' and the corresponding assignment rule, $s_{M'}$.
4. The alternatives of the learning set A are assigned using the rule $s_{M'}$. The assignment resulting from this step are compared to the one obtained at step 2 and the classification accuracy $CA(s_M, s_{M'})$ is computed [3]

We test the algorithm with models having 10 criteria and 3 to 5 categories and learning sets containing 1000 assignment examples. These experiments are done on an Intel Dual Core P8700 running GNU/Linux with CPLEX Studio 12.5 and Python 2.7.3.

In Figure 2, the average value of $CA(s_M, s_{M'})$ obtained after repeating 10 times the experiment is shown. When the number of categories increases, we observe that the algorithm needs more iterations to converge to a model restoring correctly all assignment examples. This experiment shows that it is possible to find a model restoring 99% of the assignment examples in a reasonable computing time. On average two minutes are required to find the parameters of a model having 10 criteria and 5 categories with $N_{model} = 10$, $N_o = 30$ and $N_{it} = 20$.

[3] To assess the quality of a sorting model, other indices are also used, like the area under curve (AUC). In this paper, we use only the classification accuracy (CA) as quality indicator.

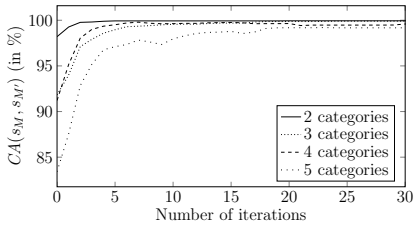


Fig. 2. Evolution of the classification accuracy of the learning set of a model composed of 10 criteria and a variable number of categories ($N_{model} = 10; N_o = 30; N_{it} = 20$)

4.2 Model retrieval

This experiment aims at answering the following question: How many assignment examples are required to obtain the parameters of a model which restores correctly a majority of assignment examples? This question has been already covered in [11] for models having no more than 5 criteria and 3 categories. To answer this question for models with more parameters we add a step to the test procedure described above:

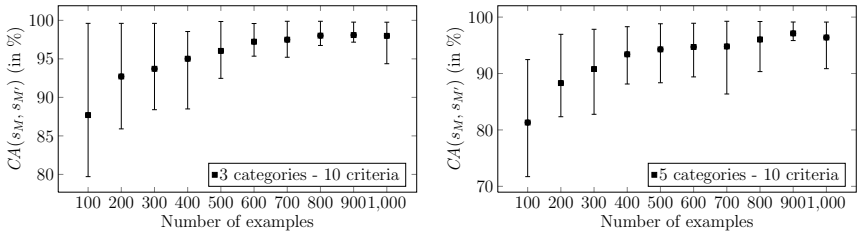
5. A set of 10000 random alternatives, B, is generated analogously to 2. We call this set, the generalization set. Alternatives of the set B are assigned by models M and M'. Finally the assignment obtained by models M and M' are compared and the classification accuracy $CA(s_M, s_{M'})$ is computed.

Figure 3(a) and 3(b) show the average, min and max $CA(s_M, s_{M'})$ of the generalization set after learning the parameters of models having 10 criteria and 3 or 5 categories on the basis of 100 to 1000 assignment examples. Figure 3(a) shows that 400 examples are sufficient to restore on average 95 % of the assignments for models having 3 categories, 10 criteria while 800 examples are needed for ones having 5 categories, 10 criteria (see Figure 3(b)). As expected, the higher the cardinality of the learning set, the more $CA(s_M, s_{M'})$ is high in generalization.

4.3 Tolerance for errors

To test the behavior of the algorithm when the learning set is not fully compatible with a MR-Sort model, a step is added in the test procedure after generating the assignment examples:

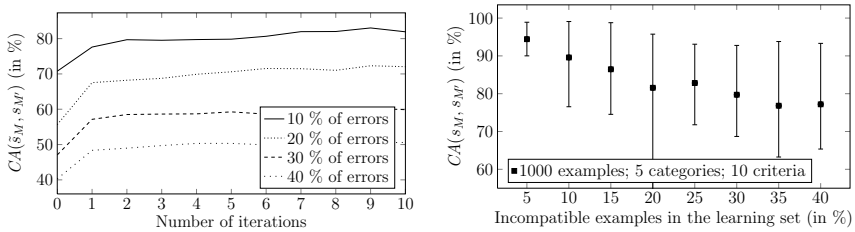
2' A proportion of errors is added in the assignments obtained using the model M. For each alternative of the learning set, its assignment is altered with probability P, the altered assignment example is uniformly drawn among the other categories. We denote by $\tilde{s}_M$ the rule producing the assignments with errors.



(a) 3 categories (b) 5 categories

Fig. 3. Evolution of the classification accuracy on the generalization set. A 10 criteria with 3 or 5 categories model has been learned on the basis of learning sets containing from 100 up to 1000 assignment examples ($N_{model} = 10; N_o = 30; N_{it} = 20$)

Tolerance for errors is tested by learning the parameters of a MR-Sort model having 5 categories and 10 criteria on the basis of 1000 assignment examples generated using $\tilde{s}_M$. In Figure 4(a), the average classification accuracy of the learning set is shown for 10 test instances with 10 to 40 % of errors in the learning set. We observe that $CA(\tilde{s}_M, s_{M'})$ converges to $1 - P$ when there are errors in the learning set. Among the assignment examples badly assigned by the model, a majority corresponds to altered examples. To see to what extent the errors affect the algorithm, we generate a generalization set that is assigned both by the rule s_M and $s_{M'}$. The resulting sets are compared and $CA(s_M, s_{M'})$ is computed. In Figure 4(b), average, minimal and maximal $CA(s_M, s_{M'})$ are shown for 10 test instances. We observe that for small numbers of errors, i.e. less than 20 %, the algorithm tends to modify the model s.t. $CA(s_M, s_{M'})$ is altered on average by the same percentage of error in generalization. When there are more than 20% of errors in the learning set, the algorithm is able to find a model giving a smaller proportion of assignment errors in generalization.



(a) Learning set (b) Generalization set

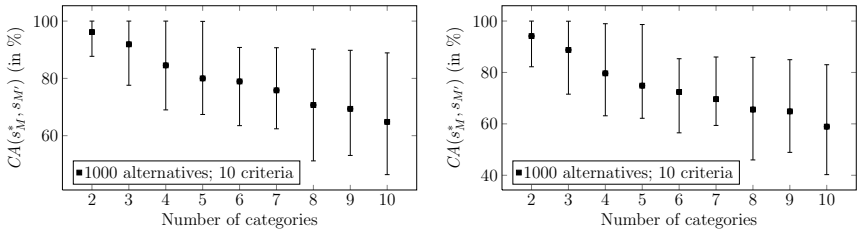
Fig. 4. Evolution of the classification accuracy ($CA(\tilde{s}_M, s_{M'})$) for the alternatives in the learning set (a) and the generalization set (b). A 5 categories and 10 criteria model is inferred on the basis of 1000 assignment examples containing 10 to 40 % of errors ($N_{model} = 10; N_o = 30; N_{it} = 20$)

4.4 Idiosyncratic behavior

This experiment aims at checking if an MR-Sort model is able to represent assignments that have been obtained by another sorting rule based on an additive value function (AVF-Sort model). In such a model, a marginal utility function u_j is associated to each criterion. In the chosen model, utility functions are piecewise linear and monotone. They are split in k parts in the criterion range $[g_{j*}, g_j^*]$, with g_{j*} the less preferred value and g_j^* the most preferred value on j, s.t. $u(g_{j*}) = 0$ and $u(g_j^*) = 1$. The end points of the piecewise linear functions are given by $g_j^l = g_{j*} + \frac{l}{k}\left(g_j^* - g_{j*}\right)$, with $l = 0, ..., k$. Marginal utility of an alternative a on criterion j is denoted by $u_j(a_j)$. The score of an alternative is given by the global utility function which is equal to $U(a) = \sum_{j=1}^{n} w_j u_j(a_j)$, with $U(a) \in [0, 1]$ and $\sum_{j=1}^{n} w_j = 1$. The higher the value of $U(a)$, the more a is preferred. Categories are delimited by ascending global utility values β^h, s.t. an alternative a is assigned in category h iff $\beta^{h-1} \leq U(a) < \beta^h$ with $\beta^0 = 0$ and $\beta^p = 1 + \epsilon$, ϵ being a small positive value. Such an additive model is used in the UTADIS method [18, 19]. We study the ability of our metaheuristic to learn an MR-Sort model from a learning set generated by an AVF-Sort model. To do so, we replace step 1 by:

1. A sorting model M based on an additive value function is randomly generated. To generate the weights, the same rule as for the MR-Sort model is used. For each value function, $k-1$ random are uniformly drawn from the interval $[0, 1]$ and ordered s.t. $r_k = 1 \geq r_{k-1} \geq ... \geq r_1 \geq 0 = r_0$, then end points are assigned as follows $u(g_j^l) = r_l$, with $l = 0, ..., k$. For the category limits β_h, $p-1$ random numbers are uniformly drawn from the interval $[0, 1]$ and then ordered s.t. $r_{p-1} \geq ... \geq r_1$. Category limits are given by $\beta_h = r_h$, $h = 1, ..., p-1$. The assignment rule is denoted by s_M^*.

Once the model generated, the alternatives are assigned by the model M and the metaheuristic tries to learn a MR-Sort model from the assignments obtained by M. To assess the ability of the heuristic to find a MR-Sort model restoring the maximum number of examples, we test it with 1000 assignment examples, on models composed of 10 criteria and 2 to 10 categories. We choose to use an AVF-Sort model in which each additive value function is composed of 3 segments. This experiment is repeated 10 times. Figure 5(a) presents the average, minimum and maximum $CA(s_M^*, s_{M'})$ of the learning set. The plot shows that the MR-Sort model is able to represent on average 80% of the assignment examples obtained with an AVF-Sort model when there are no more than 5 categories. We perform a generalization by assigning 10000 alternatives through the AVF-Sort model, M and through the learned MR-Sort model, M'. Figure 5(b) shows the average, minimum and maximum classification accuracy of the generalization set. These results confirm the behavior observed with the learning set. The ability to represent assignments obtained by an AVF-Sort model with an MR-Sort model is limited, even more when the number of categories increases.



(a) Learning set (b) Generalization set

Fig. 5. Evolution of the classification accuracy ($CA(s^*_M, s_{M'})$) for alternatives in the learning set (a) and the generalization set (b). An MR-Sort model is learned on the basis of assignment examples obtained with an AVF-Sort model having 10 criteria and 2 to 10 categories.

5 Conclusions and further research

In this paper we presented an algorithm that is suitable to learn a MR-Sort model from large sets of assignment examples. Unlike the MIP proposed in [11], it is possible to learn a model composed of 10 criteria and 5 categories that restores 99% of the examples of assignments in less than two minutes for a learning set composed of 1000 alternatives with no error.

In the case of a learning set containing a proportion of assignment errors, the experimentations showed that the algorithm finds a model giving on average a smaller or equal proportion of errors in generalization. We also found that assignment examples obtained by an AVF-Sort model are quite difficult to represent with an MR-Sort model. Further researches have to be done with the AVF-Sort model to see if it is able to learn a model that restores correctly a set of assignment examples obtained by an MR-Sort model.

The metaheuristic described in this paper does not cover MR-Sort models with vetoes. Learning the parameters of a MR-Sort model with vetoes deserves to be studied and will improve a bit the ability of the model to represent assignments.

Acknowledgment The authors thank two anonymous referees for their helpfull comments which contributed to improve the content of this paper. The usual caveat applies.

References

1. Yu, W.: Aide multicritère la décision dans le cadre de la problématique du tri: méthodes et applications. PhD thesis, LAMSADE, Université Paris Dauphine, Paris (1992)
2. Roy, B., Bouyssou, D.: Aide multicritère la décision: méthodes et cas. Economica Paris (1993)

3. Bouyssou, D., Marchant, T.: An axiomatic approach to noncompensatory sorting methods in MCDM, I: The case of two categories. European Journal of Operational Research **178**(1) (2007) 217–245
4. Bouyssou, D., Marchant, T.: An axiomatic approach to noncompensatory sorting methods in MCDM, II: More than two categories. European Journal of Operational Research **178**(1) (2007) 246–276
5. Mousseau, V., Slowinski, R.: Inferring an ELECTRE TRI model from assignment examples. Journal of Global Optimization **12**(1) (1998) 157–174
6. Mousseau, V., Figueira, J., Naux, J.P.: Using assignment examples to infer weights for ELECTRE TRI method: Some experimental results. European Journal of Operational Research **130**(1) (2001) 263–275
7. Ngo The, A., Mousseau, V.: Using assignment examples to infer category limits for the ELECTRE TRI method. Journal of Multi-criteria Decision Analysis **11**(1) (2002) 29–43
8. Dias, L., Mousseau, V., Figueira, J., Clímaco, J.: An aggregation/disaggregation approach to obtain robust conclusions with ELECTRE TRI. European Journal of Operational Research **138**(1) (2002) 332–348
9. Dias, L., Mousseau, V.: Inferring Electre's veto-related parameters from outranking examples. European Journal of Operational Research **170**(1) (2006) 172–191
10. Doumpos, M., Marinakis, Y., Marinaki, M., Zopounidis, C.: 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) (2009) 496–505
11. Leroy, A., Mousseau, V., Pirlot, M.: Learning the parameters of a multiple criteria sorting method. In Brafman, R., Roberts, F., Tsoukiàs, A., eds.: Algorithmic Decision Theory. Volume 6992 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2011) 219–233
12. Cailloux, O., Meyer, P., Mousseau, V.: Eliciting ELECTRE TRI category limits for a group of decision makers. European Journal of Operational Research **223**(1) (2012) 133–140
13. Fürnkranz, J., Hüllermeier, E.: Preference learning: An introduction. In Fürnkranz, J., Hüllermeier, E., eds.: Preference Learning. Springer-Verlag (2010) 1–17
14. Bouyssou, D., Pirlot, M.: A characterization of concordance relations. European Journal of Operational Research **167/2** (2005) 427–443
15. Bouyssou, D., Pirlot, M.: Further results on concordance relations. European Journal of Operational Research **181** (2007) 505–514
16. Pirlot, M.: General local search methods. European Journal of Operational Research **92**(3) (1996) 493–511
17. Butler, J., Jia, J., Dyer, J.: Simulation techniques for the sensitivity analysis of multi-criteria decision models. European Journal of Operational Research **103**(3) (December 1997) 531–546
18. Devaud, J., Groussaud, G., Jacquet-Lagrèze, E.: UTADIS: Une méthode de construction de fonctions d'utilité additives rendant compte de jugements globaux. In: European working group on MCDA, Bochum, Germany. (1980)
19. Doumpos, M., Zopounidis, C.: Multicriteria Decision Aid Classification Methods. Kluwer Academic Publishers, Dordrecht (2002)