Learning the parameters of a multiple criteria sorting method from large sets of assignment examples

Olivier Sobrie[1,2,3] and **Vincent Mousseau**[2] and **Marc Pirlot**[3]

Abstract. ELECTRE TRI is a sorting method used in multiple criteria decision analysis. It assigns each alternative, described by a performance vector, to a category selected in a set of pre-defined ordered categories. Consecutive categories are separated by a profile. In a simplified version proposed and studied by Bouyssou and Marchant and called MR-Sort, a majority rule is used for assigning the alternatives to categories. Each alternative a is assigned to the lowest category for which a is at least as good as the lower profile delimiting this category for a majority of weighted criteria. In this paper, a new algorithm is proposed for learning the parameters of this model on the basis of assignment examples. In contrast with previous work ([7]), the present algorithm is designed to deal with large learning sets. Experimental results are presented, which assess the algorithm performances with respect to issues like model retrieval, computational efficiency and tolerance for error.

1 Introduction

ELECTRE TRI is a sorting method used in decision analysis to assign each alternative to a category. The categories are pre-defined and ordered. A simplified version, called MR-Sort (Majority Rule Sorting method) has been studied by Bouyssou and Marchant (see [1, 2]). Alternatives are assigned to a category based on a majority rule. Each category is associated a lower profile defining its boundary with the category below. An alternative is assigned to one of the categories above a profile as soon as its performances are at least as good as those of the profile for a weighted majority of criteria.

Methods for eliciting the parameters of such a sorting method on the basis of assignment examples already exist but are limited to relatively small datasets. The question we are interested in is whether it is possible to use such rules in the context of preference learning, assuming that the learning datasets consist of a large number of assignment examples. For instance, the dataset can be composed of students' grades (satis bene, cum laude, magna cum laude, summa cum laude) corresponding to their results in the different disciplines. The goal is then to learn a MR-Sort model that assigns a grade to a student whenever a vector of his/her results in the different courses is entered. Learning such a model amounts to compute the profiles limiting the categories, the criteria weights and the majority threshold on the basis of a list of students, their marks and the grade they have been assigned to by the jury.

In [7], learning all the parameters of the MR-Sort has been formulated as a mixed integer linear program This formulation has a drawback: it is not suitable for large learning sets since it requires computing times that grow rapidly with the number of assignment examples.

This paper presents a metaheuristic we devised to infer all the parameters of an MR-Sort model. It reports the results of experiments testing the following aspects of the algorithm performance:

Model retrieval Given a set of alternatives assigned by a known MR-Sort model, what is the ability of the algorithm to determine the parameters of a model assigning these alternatives as much as possible to the same categories as the original model?

Algorithm efficiency What is the practical complexity of the algorithm? Is it able to deal with large learning sets? How much time does it take to learn the parameters of a model for a given number of categories, criteria and assignment examples?

Tolerance for error The learning set given as input to the algorithm might contain errors, e.g. alternatives that should belong to some category considering its evaluations could be erroneously assigned to a different category. The question is: How does the algorithm react to learning sets that are not entirely compatible with a MR-Sort model? Has the algorithm the ability to correct assignment errors?

In the next section of this paper, we briefly recall the rules of the MR-Sort procedure and which difficulties are involved in the elicitation of its parameters. We also discuss previous work done in view of eliciting the parameters of the MR-Sort rule. In section 3, we present the new metaheuristic. The experiments designed for testing it are described in section 4; our first experimental results are commented. We conclude with some perspectives for further work in view of improving the current version of the algorithm.

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

April 26, 2013; version 3718b5b

2 Sorting procedure

2.1 MR-Sort model

The ELECTRE TRI procedure, originally developed in [13] (see also [12]), aims to assign every alternative a_i of a set $A = \{a_1, ..., a_m\}$ to one of the pre-defined and ordered categories going from C_1 to C_p, with C_1 the worst one and C_p the best one. Alternatives are evaluated on a set of n criteria; $a_{i,j}$ denotes the performance of a_i on criterion j ($F = \{1, .., n\}$). The criterion scales are assumed to be ordered in increasing order of the decision maker's preference. The assignment to a category is done by comparing each alternative performances to the performances of the $p-1$ profiles, delimiting the p categories, on each criterion. The profiles are denoted by b_h, $h = 1, \ldots, p-1$ and the performance of profile h on criterion j is $b_{h,j}$. The lower boundary of category C_h is profile b_{h-1}. For notational convenience, we sometimes use two trivial profiles b_0 and b_p. b_0 (resp. b_p) is the lower (resp. upper) profile of category C_1 (resp. C_p). For all j, the performance of b_0 on criterion j is the worst possible performance on this criterion, so that every alternative is at least as good as b_0 on all criteria. b_p plays a symmetric role in the sense that b_h is at least as good as every alternative on all criteria.

It is assumed that the profiles dominate each other, i.e.:

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

The original procedure presents some drawbacks. In particular, it involves numerous parameters which may play interrelated roles. Although several papers have been devoted to learning the parameters of such a model [9, 10, 8, 11, 5, 4], it is not advisable to use this method for learning preferences on the basis of large sets of assignment examples. In this article we consider a version of ELECTRE TRI called the *non-compensatory sorting model*. It is based on the work of Bouyssou and Marchant who have established an axiomatic characterization of this model in the case of two [1] or more categories [2].

To describe the assignment rule, we need to recall the definition of an *outranking* relation. An alternative a_i outranks a profile b_h if and only if there is a sufficient coalition of (weighted) criteria for which a_i is at least as good as b_h on each criterion of the coalition, and there is no criterion on which a_i is *so much worse* than b_h that compensating this disadvantage is impossible. The idea that some large disadvantages cannot be compensated usually is called a *veto*; it precludes asserting that a_i outranks b_h. The "at least as good" relation S_j on criterion j can be defined for instance by:

$$a_i S_j b_h \Leftrightarrow a_{i,j} \geq b_{h,j} \tag{2}$$

The veto relation V_j on criterion j can be defined as follows:

$$a_i \, V_j b_h \Leftrightarrow b_{h,j} < a_{i,j} - v_j(b_h). \tag{3}$$

If the sum of the weights w_j of the criteria j for which a_i is at least as good as b_h is larger than or equal to a majority (or concordance) threshold λ, and if there is no criterion on which there is a veto, then a_i outranks b_h.

The global outranking relation S is defined by:

$$a_i S b_h \Leftrightarrow \sum_{j \in S(a_i, b_h)} w_j \geq \lambda \text{ and } [\text{Not}[b_h \, V_j a_i], \forall j \in F] \quad \text{with } S(a_i, b_h) = \{j \in F : a_i S_j b_h\} \tag{4}$$

With ELECTRE TRI, there are two ways to determine to which category an alternative should be assigned: they are called the pessimistic and the optimistic approach. We only describe the pessimistic approach (the only one that was characterized in [1, 2]) since it is the one used in the algorithm described below. The pessimistic procedure consists in comparing a_i to the profiles b_k for $k = p-1, p-2, \ldots, 1$ successively; if b_h is the first profile such that $a_i S b_h$, the alternative a_i is assigned to the category C_{h+1}. If the alternative a_i doesn't outrank any profile, then it is assigned to the worst category, C_1.

In this paper, we consider models without vetoes. Hence the conditions for an alternative a_i to be assigned to category C_h can be expressed as follows:

$$\sum_{j \in S(a_i, b_{h-1})} w_j \geq \lambda \quad \text{and} \quad \sum_{j \in S(a_i, b_h)} w_j < \lambda \tag{5}$$

As in [7], we call a model assigning alternatives to a category using such a rule, a *Majority Rule Sorting Model* (MR-Sort).

2.2 Elicitation of ELECTRE TRI parameters

Several published articles deal with learning the parameters of a traditional ELECTRE TRI model. In [9], it is proposed to infer the whole set of parameters of an ELECTRE TRI model from assignment examples by using a nonlinear programming formulation. In [8], the authors describe a way to learn the weights of an ELECTRE TRI model with a linear program. Article [11] deals with the inference of the profiles from assignment examples. Once again a linear program is used. In [6], a genetic algorithm is developed in order to learn the whole set of parameters of a traditional ELECTRE TRI model.

Recently, in [7], a mixed integer linear program has been proposed to infer all the parameters of an MR-Sort model. The linear program has been tested with 10 to 100 examples of assignments, 3 to 5 criteria and 2 to 3 categories. The experiments made show that a large number of assignment examples is needed to retrieve a model that represents the preferences of the decision maker with a reasonable accuracy. However, with the mixed integer linear program proposed in [7], the computing time quickly grows with the number of assignment. In [3], three mixed linear programs are used to find a set of weights or profiles which reflect as much as possible the preferences of multiple decision makers.

Using the linear programs developed in [7] and [3] is not an option in our case because we want to deal with large numbers of assignment examples and models having more than 5

criteria and 2 categories. The new approach proposed below aims at dealing with such models.

3 Inferring the parameters of a MR-Sort model

In this section we detail a new algorithm that aims to learn the whole set of parameters of an MR-Sort model. Initially, a set of random profiles dominating one another is generated. The proposed algorithm is composed of two main steps:

1. Using the current profiles, find a set of weights and a majority threshold maximizing the number of assignment examples compatible with the model;
2. Adjust the profiles in order to maximize the number of assignment examples compatible with the model.

The goal of the algorithm is to obtain the parameters of a model reflecting as much as possible the preferences of the decision maker, i.e. a model that restores as much as possible the assignments of the examples given as input. To measure the performance of the algorithm, we compute the classification accuracy CA of the final model, which is defined as:

$$CA = \frac{\text{Number of examples correctly restored}}{\text{Total number of examples}} \tag{6}$$

In this section, we first describe the linear program used to learn the weights. Then we describe the metaheuristic used to improve the position of the profiles. Finally, the coupling of the linear program and the metaheuristic is explained.

3.1 Inferring the weights and the majority threshold

Finding the weights and the majority threshold of an MR-Sort model with fixed profiles doesn't require mixed integer programming. The problem can be easily formulated as a simple linear program.

We denote by A_h the set of alternatives assigned by the DM to the category C_h. As the profiles dominate each other, the constraints for an alternative a_i to be assigned to the category C_h can be expressed as follows:

$$\sum_{\forall j | a_i S_j b_{h-1}} w_j - x_i + x'_i = \lambda \qquad \forall a_i \in A_h, \quad h = \{2, ..., p-1\} \tag{7}$$

$$\sum_{\forall j | a_i S_j b_h} w_j + y_i - y'_i = \lambda - \delta \qquad \forall a_i \in A_h, \quad h = \{1, ..., p-2\} \tag{8}$$

with:

$$\sum_{j=1}^{n} w_j = 1 \tag{9}$$

$$\lambda \in [0.5; 1] \tag{10}$$

$$w_j \in [0; 1] \qquad \forall j \in F \tag{11}$$

$$x_i \in \mathbb{R}_0^+ \qquad \forall a_i \tag{12}$$

$$y_i \in \mathbb{R}_0^+ \qquad \forall a_i \tag{13}$$

$$x'_i \in \mathbb{R}_0^+ \qquad \forall a_i \tag{14}$$

$$y'_i \in \mathbb{R}_0^+ \qquad \forall a_i \tag{15}$$

The value $x_i - x'_i$ (resp. $y_i - y'_i$) represents the difference between the sum of the weights of the criteria belonging to coalition in favor of $a_i \in A_h$ w.r.t. b_{h-1} (resp. b_h) and the majority threshold. If both $x_i - x'_i$ and $y_i - y'_i$ are positive, then the alternative a_i is assigned to the right category. In order to try to get a maximum number of compatible alternatives, the objective function of the linear program minimizes the sum of x'_i and y'_i:

$$\min \sum_{a_i \in A} (x'_i + y'_i) \tag{16}$$

However this objective function does not guarantee to return a set of weights and a majority threshold which maximize the number of alternatives assigned to the category indicated by the DM. This is due to the fact that the objective function allows for compensatory effects between constraints.

3.2 Inferring the profiles

Trying to learn the profiles using an exact method is not easy because conditions (5) cannot be formulated as linear constraints. Exact methods have been proposed and studied in [7]. They require mixed integer programming solvers. As we want to deal with models having more than 5 criteria and 2 categories, the use of a linear program with binary variables is not an option due to quickly exploding computing times. Therefore we opted for developing a metaheuristic, which is described below.

3.2.1 Idea of the metaheuristic

Consider a model with 5 criteria, 2 categories, C_1 and C_2 (C_2 being the best and C_1 the worst category). We assume that the criteria weights are known. Let a_1 and a_2 be 2 misclassified alternatives (see Figure 1). The profile delimiting the two categories is denoted by b_1; b_0 and b_2 correspond respectively to the worst and the best possible (fictive) alternative on the five criteria. Imagine that a_1 is wrongly assigned by the procedure to the category C_2 instead of C_1. This means that the profile has too low levels on one or several criteria. In contrast, an alternative a_2 wrongly assigned to category C_1 instead of C_2 means that the profile level is too high on one or several criteria (we recall that the weights are considered as known). On Figure 1, an arrow shows the direction in which moving

the profile in order to potentially assign the two alternatives to the right category.

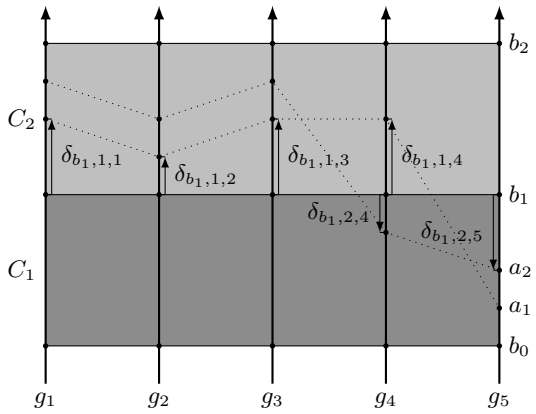

Figure 1. Alternative wrongly assigned because of the profile too low or too high

We denote by A_1^2 (resp. A_2^1) the set of alternatives wrongly classified in C_2 (C_1, respectively) by the inferred model while the DM assigns them to category C_1 (resp. C_2). The sets of alternatives correctly classified in C_1 and C_2 are denoted respectively by A_1^1 and A_2^2. With a two categories model, each alternative belongs to one of the four sets, A_1^1, A_2^1, A_1^2 or A_2^2. An alternative belonging to A_2^1 (resp. A_1^2) indicates that the profile level is too high (resp. too low) on one or several criteria (assuming that we do not change the weights).

Regarding the relative position of the evaluation $a_{i,j}$ of alternative a_i on criterion j and the profile evaluation $b_{1,j}$ and considering the assignment of the alternative, we can distinguish 8 cases (see Figure 2); $\delta_{1,i,j}$ represents the distance between the profile level and the alternative evaluation on criterion j.

In the 8 cases represented in Figure 2, we see that the difference between the value of the profile $b_{1,j}$ and the performance of the alternative $a_{i,j}$ can have a positive (cases a, d, e, h) or a negative (cases b, c, f, g) influence on the classification. We denote by $W_{1,j}$ the set of alternatives wrongly assigned by the model and for which the criterion j is not in favor of the correct assignment due to the current profile level. The set $R_{1,j}$ contains the alternatives for which evaluation of b_1 favors the assignment to the right class.

$$\begin{aligned} W_{1,j} &= \{a_i \in A_1^2 : a_{i,j} \geq b_{1,j}\} \\ &\quad \cup \{a_i \in A_2^1 : a_{i,j} < b_{1,j}\} \end{aligned} \tag{17}$$

$$\begin{aligned} R_{1,j} &= \{a_i \in A_1^2 : a_{i,j} < b_{1,j}\} \\ &\quad \cup \{a_i \in A_1^1 : a_{i,j} < b_{1,j}\} \\ &\quad \cup \{a_i \in A_2^1 : a_{i,j} \geq b_{1,j}\} \\ &\quad \cup \{a_i \in A_2^2 : a_{i,j} \geq b_{1,j}\} \end{aligned} \tag{18}$$

The alternatives contained in the sets $W_{1,j}$ and $R_{1,j}$ give an indication about how the profile should be moved on criterion j to potentially increase the classification accuracy of the

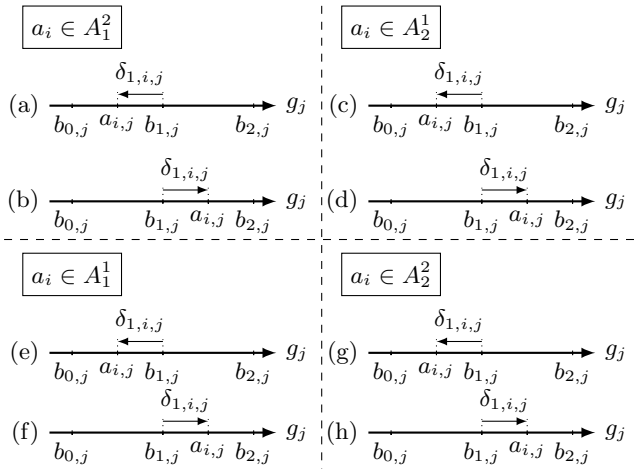

Figure 2. Given the evaluation of an alternative and of the profile on a criterion j, 8 possible cases regarding the alternative assignment

model. In order to assess the advantage of the different possible moves of the profile level, the space between the profiles levels $b_{1,j}$ and $b_{0,j}$ on criterion j is split into k sub-intervals by means of k subdivision points denoted by $b_{1,j}^{-l}$ for $l = 1, \ldots, k$. The same is done between $b_{1,j}$ and $b_{2,j}$ by means of k subdivision points denoted by $b_{1,j}^{+l}$ for $l = 1, \ldots, k$. We consider these $2k$ subdivision points scattered on both sides of $b_{1,j}$ as the candidate moves for the profile level $b_{1,j}$. Then, histograms similar to those shown in Figure 3 are constructed for each criterion j.

The bars lengths in the first (resp. second) histogram represent the number of alternatives in the set $W_{1,j}^{\pm l}$ (resp. $R_{1,j}^{\pm l}$) where $W_{1,j}^{\pm l}$ (resp. $R_{1,j}^{\pm l}$) denotes the set of alternatives belonging to $W_{1,j}$ (resp. $R_{1,j}$) the evaluation of which, on criterion j, is located between the current value of the profile, $b_{1,j}$, and the potential new value, $b_{1,j}^{\pm l}$. In the last histogram, the bars lengths represent what is formally a probability P defined by:

$$P(b_{1,j}^{\pm l}) = \frac{|W_{1,j}^{\pm l}|}{|W_{1,j}^{\pm l}| + |R_{1,j}^{\pm l}|} \tag{19}$$

If we move the profile level $b_{1,j}$ to $b_{1,j}^{\pm l}$, the number $|W_{1,j}^{\pm l}|$ will decrease by $|W_{1,j}^{\pm l}| - |R_{1,j}^{\pm l}|$ and the number $|R_{1,j}^{\pm l}|$ will increase by the same quantity. If the quantity $|W_{1,j}^{\pm l}| - |R_{1,j}^{\pm l}|$ is positive, the number of correctly assigned alternatives with their evaluation on the right side of the profile will tend to increase while the profile level is moved to $b_{1,j}^{\pm l}$. Of course, the number of correctly assigned alternatives will not mechanically increase by $|W_{1,j}^{\pm l}| - |R_{1,j}^{\pm l}|$ since the corresponding change in the profile level only concerns criterion j. We use the probabilities $P(b_{1,j}^{\pm l})$ as indicators of the potential gain in correct classification that can be expected from a move of the profile level on some criterion. The probabilities associated with profile b_1 on criterion j are computed and the value $L \in \{-k, \ldots, -1, 1, \ldots, k\}$ for which the probability of $b_{1,j}^{L}$ is

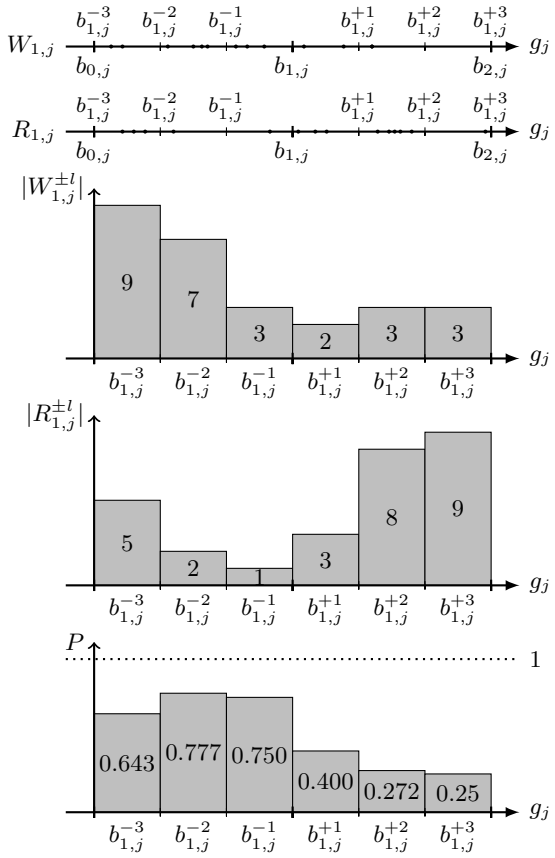

Figure 3. Histogram of the evaluations of misclassified alternatives on criterion j

maximal is recorded. Then a random number r is drawn from the uniform distribution on $[0, 1]$. If the value of r is smaller than $P(b_{1,j}^{\pm L})$, then the profile is moved to $b_{1,j}^{\pm L}$, otherwise the profile is not moved at all. The same operation is performed for each criterion.

One loop of the metaheuristic in the case of a model with 2 categories can be summarized by the following algorithm:

for all $j \in \{1, ..., n\}$ **do**
  Compute $P(b_{1,j}^{\pm l})$, $\forall l$
  Find L such that $P(b_{1,j}^{L}) = \max_l(P(b_{1,j}^{\pm l}))$
  Draw a random number r from the uniform distribution $[0, 1]$
  if $r < (P(b_{1,j}^{L}))$ **then**
    $b_{1,j} = b_{1,j}^{L}$
  end if
end for

When there are more than two categories, a similar algorithm is applied to each profile with a slightly adapted definition of W and R:

$$\begin{aligned} W_{h,j} &= \left\{a_i \in A_h^{h+1} : b_{h+1,j} > a_{i,j} \geq b_{h,j}\right\} \\ &\quad \cup \left\{a_i \in A_{h+1}^{h} : b_{h-1,j} < a_{i,j} < b_{h,j}\right\} \end{aligned} \tag{20}$$

$$\begin{aligned} R_{h,j} &= \left\{a_i \in A_h^{h+1} : b_{h-1,j} \leq a_{i,j} < b_{h,j}\right\} \\ &\quad \cup \left\{a_i \in A_h^{h} : b_{h-1,j} \leq a_{i,j} < b_{h,j}\right\} \\ &\quad \cup \left\{a_i \in A_{h+1}^{h} : b_{h+1,j} > a_{i,j} \geq b_{h,j}\right\} \\ &\quad \cup \left\{a_i \in A_{h+1}^{h+1} : b_{h+1,j} > a_{i,j} \geq b_{h,j}\right\} \end{aligned} \tag{21}$$

for $h = 1, ..., p-1$. In these definitions, 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. Note that the definitions of $W_{h,j}$ and $R_{h,j}$ only take into account the alternatives for which the class assigned by the DM and the model either coincide or are nearest neighbor. Definitions which take into account all misclassified alternatives have been experimented and have led to inferior results in terms of convergence of the algorithm.

3.2.2 Parameters setting and tactical details

In the metaheuristic outlined above, several parameters and implementation details influence the convergence. The following options have been chosen.

Objective function and stopping criterion In the proposed algorithm, the objective function aims at maximizing the classification accuracy of the model. The stopping criterion is met once the classification accuracy is equal to 1 or when the algorithm has run for N_{it} loops.

Number and position of the subdivision points $b_{i,j}^{\pm l}$ The interval in which the value of the profile b_h can vary is subdivided in $2k$ subintervals. The number k and the way of subdividing the interval (equal vs. unequal subintervals) must be specified.

Probability function In the present version, the probability (19) only takes into account the number of alternatives rightly or wrongly assigned to one of the two categories neighboring the profile.

Treatment order of the profiles When there are more than two categories, we have to specify the order in which the algorithm handles the profiles. In this paper, they are treated in ascending order of their labels, i.e. b_1, b_2, ..., b_{p-1}.

3.3 Inferring all the parameters

To infer all the parameters of the MR-Sort procedure, the linear program and the metaheuristic, described in the previous paragraphs, are combined.

First, a set of N_{mod} MR-Sort models is generated. Each model is initialized with a set of random profiles. Then, for each model, the following two operations are repeated at most N_o times:

1. Given the current profiles, the weights and a majority threshold are learned by using the linear program.
2. Given the current values of the weights and the threshold, the profiles are improved by running the metaheuristic N_{it} times. The classification accuracy CA, is computed after each loop. After the N_{it} loops, the profiles giving the best CA are kept.

After the 2 steps learning procedure has been applied to the N_{mod} models, the algorithm keeps only the $N_{mod}/2$ models giving the best CA and $N_{mod}/2$ new models are randomly generated. The algorithm is stopped once a model has a CA equal to 1 or when the algorithm has run N_o times.

4 Experimentations

In this section, we address the validation issues presented in the introduction, i.e. model retrieval, algorithm efficiency and tolerance for errors. We successively test the linear program used to infer the weights and the majority threshold, the metaheuristic used to infer the profiles and the metaheuristic allowing to infer the whole set of parameters.

To test the algorithm partially and globally, we use a common testing procedure:

1. A random MR-Sort model M is generated. It is determined by a set of weights, normalized to 1, a set of profiles with evaluations on the n criteria between 0 and 1 and a majority threshold whose value is picked in the interval $[0.5, 1]$. All values are drawn from uniform distributions. Using model M as described by (5), 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 performance values are drawn uniformly and independently 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. In case we only infer part of the parameters of the rule, the other parameters are given as input to the algorithm, e.g. for the inference of the profiles, the weights and the majority threshold are given as input. Then the algorithm runs and tries to maximize the number of assignments compatible with the output resulting from step 2. The resulting model is denoted by M'. The alternatives in the set A are assigned to a category by the model M'. Formally, the assignment rule is denoted by $s_{M'}$. We compute the classification accuracy $CA(s_M, s_{M'})$ according to Equation 6, i.e. $CA(s_M, s_{M'}) = \frac{|\{a \in A : s_M(a) = s_{M'}(a)\}|}{|A|}$.

The last step allows to study the efficiency of the algorithm either by examining the computing time needed to learn the parameters or by observing the algorithm convergence behavior. In order to answer to the two other questions posed in the introduction, additional steps are required:

4. After learning the parameters, a set of 10000 alternatives with random performances (drawn independently from the uniform distribution in the $[0, 1]$ interval) is generated. It is denoted by B. This set is used in the generalization phase.
5. The alternatives contained in the set B are assigned using rules s_M and s'_M and the classification accuracy of model M' is computed.

These two steps allow us to address the model retrieval issue. To see how the algorithm behaves when the learning set contains errors, an additional step is needed:

2' A proportion of error is added in the assignment resulting from rule s_M. We denote by $\tilde{s}_M$ the rule producing the assignments with errors.

After learning the parameters of the MR-Sort model, two values of classification accuracy can be computed to analyze the algorithm behavior in the presence of errors. On the one hand, the value $CA(s_M, s_{M'})$ gives an indication on the ability of the algorithm to correct the errors in assignment examples. On the other hand, the value $CA(\tilde{s}_M, s_{M'})$ gives an indication on the ability of the algorithm at finding a model fitting the learning set given as input.

The experimentations presented are made on an Intel Core 2 P8700 PC running Gentoo Linux, CPLEX version 12 and Python 2.7.2. All experiments are repeated on 10 random instances and the values displayed in the graphics below are averages over these 10 instances.

4.1 Inference of the weights and majority threshold

4.1.1 Computing time

To see how much time is needed to learn the weights and the majority threshold, the linear program, described in subsection 3.1 is tested for models with 3 categories and 5, 7, 10 or 20 criteria with 1000 to 10000 assignment examples. The profiles that are used are the correct ones, i.e. those used in the rule s_M that assigns the alternatives in the learning set.

Solving large continuous variables linear programs using a solver like CPLEX can be done very efficiently. However, a pre-treatment of the linear constraints is required in order to reduce the computing time needed to encode the constraints into the solver. The pre-processing consists in consists in filtering the constraints 7 in view of eliminating the redundant ones.

The experimental results are displayed in Figure 4. It shows that less than 1 second is needed to learn the parameters of a model having 3 categories and 10 criteria, even when the learning set is as large as 10000 alternatives. However we see that the computing time increases with the number of criteria. This is due to the fact that the number of non-redundant constraints quickly grows when the number of criteria is increased.

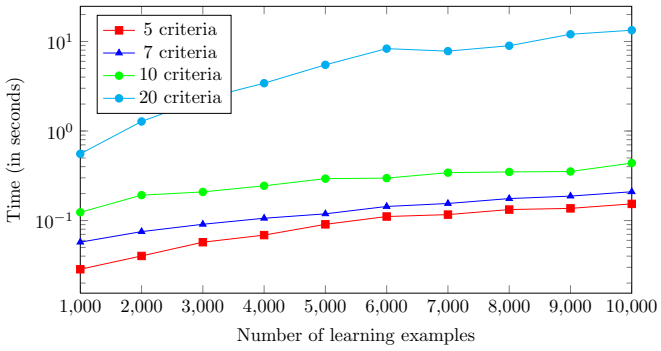

Figure 4. Computing time for learning the weights and the majority threshold of a model with 3 categories and 5, 7, 10 or 20 criteria

4.1.2 Model retrieval

What is the number of alternatives needed to obtain a good set of weights and majority threshold for a model with a given number of categories and criteria (assuming that we start with the right profiles)?

The algorithm is tested on 3 categories and 10 criteria models with learning sets involving 100 to 1000 assignment examples. The inferred model ($s_{M'}$) is used "in generalization" to assign 10000 randomly generated alternatives. These assignments are compared with those made by the original rule (s_M), yielding an assessment of the classification accuracy. The evolution of the classification accuracy is shown on Figure 5.

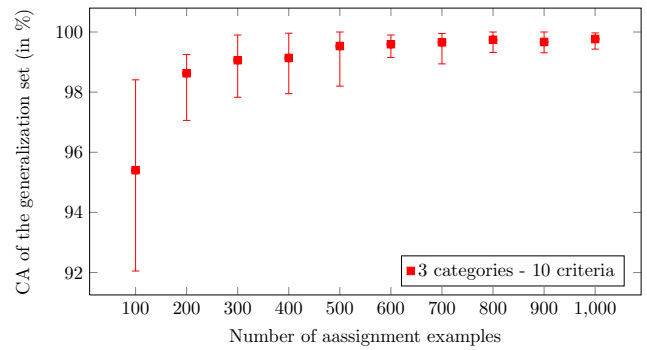

Figure 5. Evolution of the classification accuracy of models having 3 categories and 10 criteria when the learning set contains 100 to 1000 alternatives

As we can see from the plot, the linear program returns weights and a threshold that allow to assign the alternatives in a similar way as the original model even for relatively small learning sets. The classification accuracy is above 95 % for 200 assignment examples; it quickly reaches a classification accuracy close to 100 % when the number of alternatives increases.

4.1.3 Tolerance for error

Starting with learning sets in which the alternatives have been assigned according to rule s_M, we introduce random assignment errors. More precisely, a certain proportion of the alternatives are reassigned to another category chosen uniformly at random among all the other categories. We investigate how the algorithm reacts.

The algorithm for learning the weights and a threshold is tested on 3 categories and 10 criteria models when a proportion of 5 to 40 % of assignment errors are introduced in learning sets composed of 1000 assignment examples. Once the parameters have been learned, we compare the original model and the learned one on the manner they assign the alternatives in the learning set.

Figure 6 displays the average, minimal and maximal values of the classification accuracy obtained for learning sets containing 5 to 40 % of erroneous assignments. Since the number of assignment errors made by the learned model usually is smaller than the number of assignment errors introduced in the learning set, we conclude that the algorithm selects weights and a threshold in such a way that some of the errors introduced in the learning set are corrected, thus obtaining a classification accuracy $CA(s_M, s_{M'})$ that is generally better than 100 % minus the assignment error rate in the learning set.

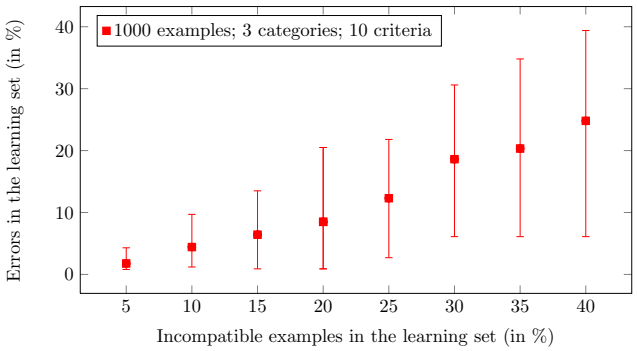

Figure 6. Evolution of the number of assignment errors made by the learned model for the alternatives in the learning set. The original model has 3 categories and 10 criteria and the learning set contains 5 to 40 % of erroneous assignments

A further issue is the following. Are the alternatives wrongly assigned by the learned model mostly alternatives that have been erroneously reassigned to introduce terrors in the learning set? Or, on the opposite, does the learned model create many new assignment errors? In the set of alternatives wrongly assigned with the learned weights and majority threshold, what's the percentage of alternatives that were degraded in this set? By looking in the set of alternatives incorrectly assigned by the function $s_{M'}$, we see that these alternatives are mainly ones that were not errors. For instance, in a case in which the learning set is composed of 1000 alternatives, erroneously assigned for 10% of them, among the 5%

of errors obtained by assigning the alternatives of the learning set by means of M', only 0.5% correspond to errors introduced in the learning set. We conclude that the algorithm is able to correct introduced assignment errors, but will in general create other errors.

4.2 Inference of the profiles

4.2.1 Strategy for moving the profiles

As emphasized in section 3.2.2, the convergence of the algorithm is influenced by several parameters. Among these, we now focus on the size of the intervals and the number of intervals determining the possible moves for the profiles. We present the evolution of the classification accuracy in connection with 3 different strategies for defining the potential profiles moves.

1. Equally spaced subdivisions between the profiles.

$$b_{h,j}^{+l} = b_{h,j} + \frac{l}{k} \cdot (b_{h+1,j} - b_{h,j}) \tag{22}$$

$$b_{h,j}^{-l} = b_{h,j} - \frac{l}{k} \cdot (b_{h,j} - b_{h-1,j}) \tag{23}$$

   with k the number of sub-intervals and $l \in \{1, ...k\}$.
2. Spacing between two subdivisions increasing as a function of the distance to the profile.

$$b_{h,j}^{+l} = b_{h,j} + \frac{e^l}{\sum_{i=1}^{k} e^i} \cdot (b_{h+1,j} - b_{h,j}) \tag{24}$$

$$b_{h,j}^{-l} = b_{h,j} - \frac{e^l}{\sum_{i=1}^{k} e^i} \cdot (b_{h,j} - b_{h-1,j}) \tag{25}$$

3. Spacing between subdivisions increasing as a function of the distance to the profile; Number of intervals increasing as a function of the classification accuracy of the model.

We see on Figure 7 that the third strategy guarantees a faster convergence. It is the one that is used in the rest of the experimentations.

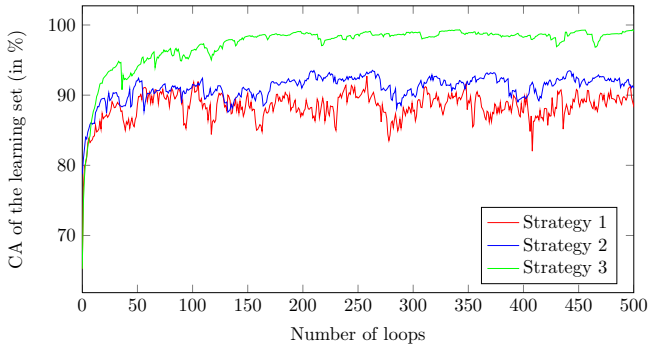

Figure 7. Evolution of the classification accuracy for 3 categories and 10 criteria models, depending on the strategy adopted for moving the profiles

4.2.2 Model retrieval

The experiments are made on models having 3 categories and 10 criteria. Using the right weights and threshold (those of model M), the profiles are learned on the basis of learning sets involving from 100 up to 1000 assignment examples. The resulting model M' is then used to assign 10000 randomly generated alternatives. Their assignment by M' is then compared with their assignment using M. The average, minimal and maximal values of the classification accuracy for the 10 instances are plotted on Figure 8.

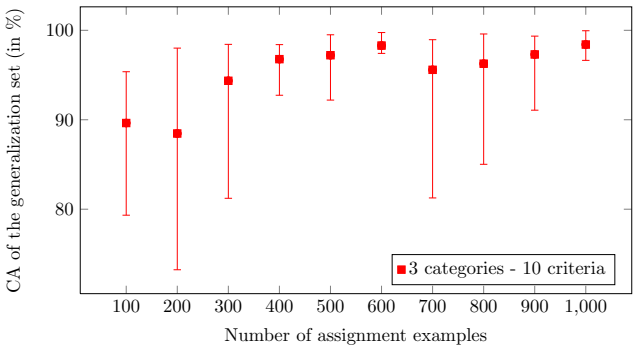

Figure 8. Evolution of the classification accuracy in generalization (10000 alternatives) for models having 3 categories and 10 criteria; size of learning set: 100 up to 1000 alternatives

With 1000 alternatives in the learning set, the classification accuracy of the alternatives contained in the generalization set is on average close to 100 %. Unlike the linear program used to find the weights and the majority threshold, the metaheuristic requires more examples to return an appropriate set of profiles. This can be explained on the one hand by the number of parameters to be determined is larger (when there are more than two categories) and on the other hand by the fact that the metaheuristic can remain stuck in a local minimum.

4.2.3 Tolerance for error

Experiments are made on models having 3 categories and 10 criteria; the learning set involves 1000 alternatives. A proportion of random erroneous reassignments is applied to the learning set. Model M' is learned on the basis of this corrupted learning set and then, the assignments of the alternatives in the learning set by model M' are compared to those produced by the corrupted rule $\tilde{s}_M$. Figure 9 indicates that the percentage of errors obtained after assigning the alternatives through the learned model tends to be higher than the percentage of errors introduced in the learning set.

In the case of learning sets with 10% of introduced errors, the classification accuracy $CA(\tilde{s}_M, s_{M'})$ is more or less equal to 85% after 50 loops of the algorithm. Looking at the 15% of alternatives erroneously assigned, we observe that 9.5% are alternatives that have been reassigned (i.e. belong to the assignment errors introduced in the learning set). This indicates

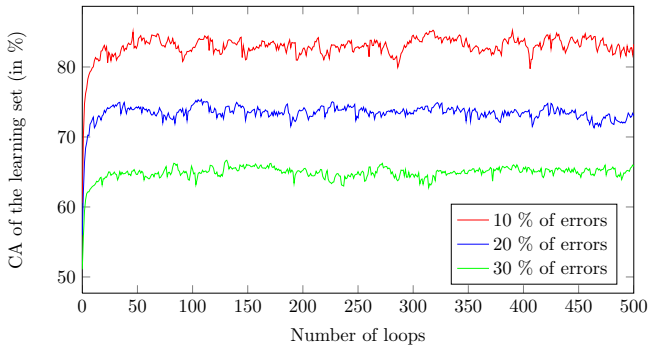

Figure 9. Evolution of the classification accuracy w.r.t. the erroneous learning set ($CA(\tilde{s}_M, s_{M'})$) used to learn the profiles of models having 3 categories and 10 criteria with 1000 learning alternatives containing 10 to 30 % of incompatible assignments

that the algorithm has the ability to identify wrong assignments and adjust the parameters on the basis of the learning examples which are not corrupted. However, the algorithm also introduces some additional errors while assigning the alternatives in the learning set.

The observations just made let us expect good results in generalization, as the learned model M' seems to be close to the original–uncorrupted–model M. To challenge this feeling, we compare the assignments obtained by the learned model M' and the original model M on a set of 10000 randomly generated alternatives.

Figure 10 shows that the metaheuristic has a good capacity to retrieve assignment examples even in the presence of errors in the learning set. For instance, with 10 % of errors in the learning set, the model learned by means of the algorithm correctly assigns 97.5 % of the alternatives in the generalization set.

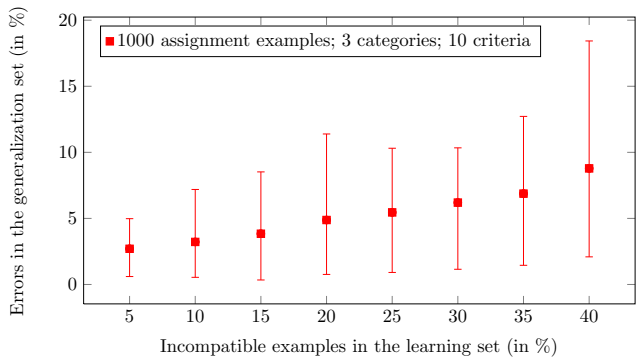

Figure 10. Evolution of the classification errors in the generalization set (10000 alternatives) after learning the profiles of models having 3 categories and 10 criteria on the basis of a set of 1000 assignment examples containing 5 to 40 % assignment errors

4.3 Inference of all parameters

For the inference of the whole set of parameters, the same experiments as for the partial inferences have been performed.

4.3.1 Convergence of the algorithm

Our first concern is to study the convergence behavior of the combined algorithm. The program described for the inference of all parameters of a MR-Sort model is tested for models having 3 categories and 10 criteria. The algorithm is run 100 times ($N_o = 100$) on a population of 10 models ($N_{mod} = 10$). For each loop, the linear program is run once and the metaheuristic 20 times (N_{it}).

Figure 11 displays the average classification accuracy of the alternatives in the learning set after each loop. We observe that the strongest improvement in the classification accuracy is obtained during the first iteration. It is then not needed to run the algorithm for too long because the gain in classification accuracy will not be substantial. In the example presented, 10 iterations of the combined algorithm appears to be sufficient.

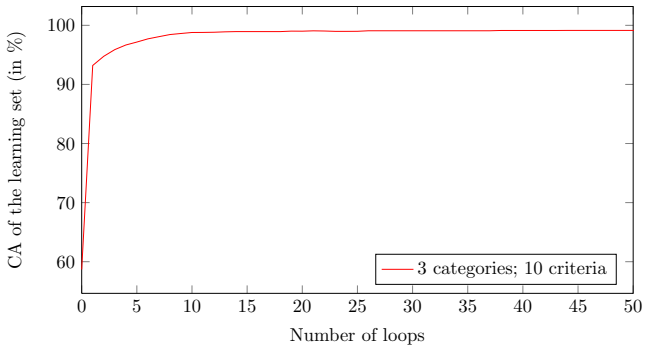

Figure 11. Evolution of the classification accuracy of the alternatives in the learning set used to learn the profiles of a model having 3 categories and 10 criteria with 1000 assignment examples ($N_{mod} = 10; N_o = 100; N_{it} = 20$)

4.3.2 Model retrieval

How many examples should we consider in the learning set in order to be able to infer a model that gives a fair representation of the decision maker's preferences? The experimentation is performed for 3 categories and 10 criteria models. The learning sets involve from 100 up to 1000 assignment examples. We study the classification accuracy of the learned model M' by comparing the assignments of 10000 randomly generated alternatives. Figure 12 shows the classification accuracy $CA(s_M, s_{M'})$ in generalization.

The plot shows us that the learned model is tuned precisely enough to produce a classification accuracy superior to 90 % on average when at least 300 assignment examples are used. As expected, the larger the number of assignment examples used as input, the more precise the model.

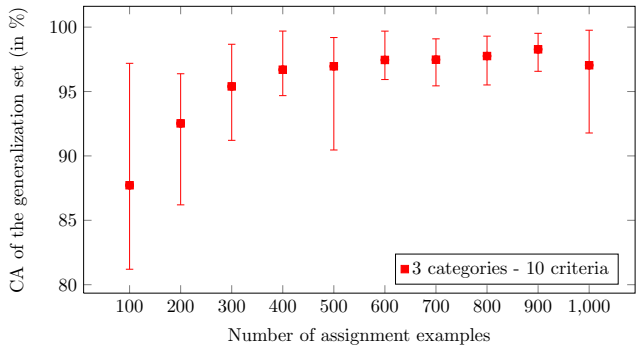

Figure 12. Evolution of the classification accuracy on the generalization set. A 3 categories and 10 criteria model has been learned on the basis of learning sets containing from 100 up to 1000 assignment examples ($N_{mod} = 10; N_o = 100; N_{it} = 20$)

4.3.3 Tolerance for error

We also want to know the capacity of the algorithm to return appropriate values for the parameters when the learning set contains erroneous assignments. The experimental setting is the same as before.

Using the learned model M' to assign the examples in the learning set yields the results displayed on Figure 13.

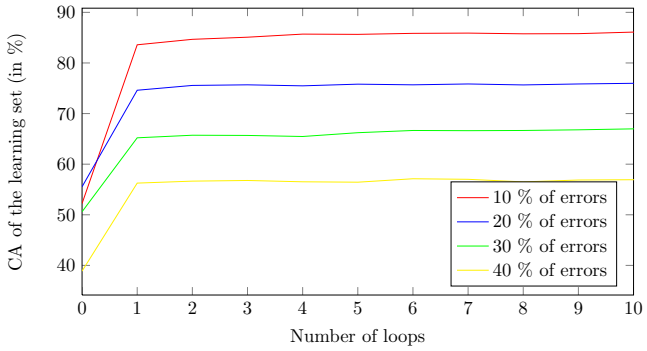

Figure 13. Evolution of the classification accuracy ($CA(\tilde{s}_M, s_{M'})$) for the alternatives in the learning set. A 3 categories and 10 criteria model is inferred on the basis of 1000 assignment examples containing 10 to 30 % of errors ($N_{mod} = 10; N_o = 100; N_{it} = 20$)

We see that the classification accuracy reflects the percentage of errors in the learning set. For 10 % of errors in the learning set given in input, the classification accuracy of the learning set after learning the model stays between 85 % and 90 %.

To assess the ability of the algorithm to identify incorrectly assigned alternatives, we study its behavior in generalization by randomly generating 10000 alternatives and comparing their assignment by the original model M and the learned model M'. The results are shown on Figure 14.

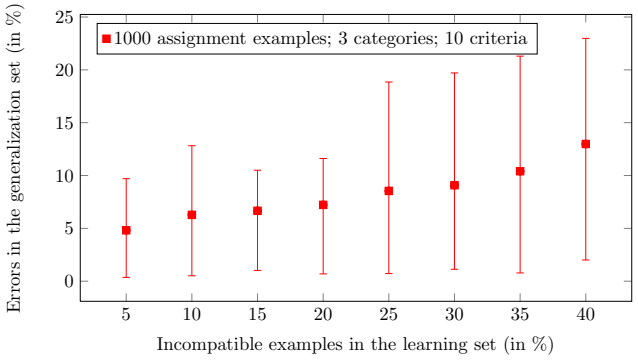

Figure 14. Evolution of the classification errors on the generalization set (10000 alternatives) after learning the whole set of parameters on the basis 1000 assignment examples ($N_{mod} = 10; N_o = 100; N_{it} = 20$)

The percentage of errors in the learning set is on average attenuated by the metaheuristic. For instance with 20 % of errors in the learning set, the average error is around 8 % in the generalization set with the learned model. However, the metaheuristic sometimes returns models producing a percentage of assignment errors superior to the error rate imposed on the learning set. For instance, with 5 % of errors in the learning set, the metaheuristic returned once 10 % of errors in the generalization set. This demonstrates that there is still room for improving the algorithm, which may currently fail to converge to a learned model sufficiently close to the original one.

5 Conclusion and further research issues

In this article we presented a new metaheuristic to learn the whole set of parameters of an MR-Sort procedure. Several experiments have been performed for testing the behavior of the metaheuristic when large learning sets are used as input.

In the experimentations, we observed that we can obtain good results for reasonably complex models i.e. typically those involving 3 categories and 10 criteria. The metaheuristic can retrieve the parameters of such models from 500 assignment examples with a classification accuracy close to 95 %. When the assignment of the examples in the learning set is not fully compatible with a MR-Sort model, the metaheuristic is still able to return a reasonable approximation of the "true" model by an adequate MR-Sort model. In generalization, we saw that the percentage of assignment errors made by the learned model is smaller than the percentage of assignment errors introduced the learning set.

However, the experiments have also shown that additional work is needed to improve the algorithm behavior in some cases. When there are no assignment errors in the learning set, it happens that the metaheuristic used for learning the profiles does not converge towards a classification accuracy of 100 % even after more than 500 loops. Several tactical options for implementing the metaheuristic have been presented in section 3.2.2 but only two of them have been studied. Other

parameters like the probability function used for choosing the profiles moves deserve to be studied in the perspective of improving the algorithm performance.

This paper does not cover the case of a MR-Sort model with vetoes as described in section 2.1. Learning the parameters of such model is another challenge.

REFERENCES

[1] 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), 217–245 (2007)

[2] 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), 246–276 (2007)

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

[4] Dias, L., Mousseau, V.: Inferring Electre's veto-related parameters from outranking examples. European Journal of Operational Research 170(1), 172–191 (2006)

[5] 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), 332–348 (2002)

[6] 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), 496–505 (2009)

[7] 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, Lecture Notes in Computer Science, vol. 6992, pp. 219–233. Springer Berlin / Heidelberg (2011)

[8] 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), 263–275 (2001)

[9] Mousseau, V., Slowinski, R.: Inferring an ELECTRE TRI model from assignment examples. Journal of Global Optimization 12(1), 157–174 (1998)

[10] Mousseau, V., Slowinski, R., Zielniewicz, P.: A user-oriented implementation of the ELECTRE TRI method integrating preference elicitation support. Computers & OR 27(7-8), 757–777 (2000)

[11] 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), 29–43 (2002)

[12] Roy, B., Bouyssou, D.: Aide multicritère à la décision: méthodes et cas. Economica Paris (1993)

[13] Yu, W.: Aide multicritère à la décision dans le cadre de la problématique du tri: méthodes et applications. Ph.D. thesis, LAMSADE, Université Paris Dauphine, Paris (1992)