A Population-Based Algorithm for Learning a Majority Rule Sorting Model with Coalitional Veto

Olivier Sobrie, Vincent Mousseau, Marc Pirlot

A Population-Based Algorithm for Learning a Majority Rule Sorting Model with Coalitional Veto

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

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

Abstract. MR-Sort (Majority Rule Sorting) is a multiple criteria sorting method which assigns an alternative a to category C^h when a is better than the lower limit of C^h on a weighted majority of criteria, and this is not true with the upper limit of C^h. We enrich the descriptive ability of MR-Sort by the addition of coalitional vetoes which operate in a symmetric way as compared to the MR-Sort rule w.r.t. to category limits, using specific veto profiles and veto weights. We describe a heuristic algorithm to learn such an MR-Sort model enriched with coalitional veto from a set of assignment examples, and show how it performs on real datasets.

1 Introduction

Multiple Criteria Sorting Problems aim at assigning alternatives to one of the predefined ordered categories $C^1, C^2, ..., C^p$, C^1 and C^p being the worst and the best category, respectively. This type of assignment method contrasts with classical supervised classification methods in that the assignments have to be monotone with respect to the scores of the alternatives. In other words, an alternative which has better scores on all criteria than another cannot be assigned to a worse category.

Many multiple criteria sorting methods have been proposed in the literature (see e.g., [1,2]). MR-Sort (Majority Rule Sorting, see [3]) is an outranking-based multiple criteria sorting method which corresponds to a simplified version of ELECTRE TRI where the discrimination and veto thresholds are omitted[1]. MR-Sort proved able to compete with state-of-the-art classification methods such as Choquistic regression [4] on real datasets.

[1] It is worth noting that outranking methods used for sorting are not subject to Condorcet effects (cycles in the preference relation), since alternatives are not compared in pairwise manner but only to profiles limiting the categories.

In the pessimistic version of ELECTRE TRI, veto effects make it possible to worsen the category to which an alternative is assigned when this alternative has very bad performances on one/several criteria. We consider a variant of MR-Sort which introduces possible veto effects. While in ELECTRE TRI, a veto involves a single criterion, we consider a more general formulation of veto (see [5]) which can involve a coalition of criteria (such a coalition can be reduced to a singleton).

The definition of such a "*coalitional veto*" exhibits a noteworthy symmetry between veto and concordance. To put it simple, in a two-category context (*Bad/Good*), an alternative is classified as *Good* when its performances are above the concordance profile on a sufficient majority of criteria, and when its performances are not below the veto profile for a sufficient majority of criteria. Hence, the veto condition can be viewed as the negation of a majority rule using a specific veto profile, and specific veto weights.

Algorithms to learn the parameters of an MR-Sort model without veto (category limits and criteria weights) have been proposed, either using linear programming involving integer variables (see [3]) or using a specific heuristic (see [6,7]). When the size of the learning set exceeds 100, only heuristic algorithms are able to provide a solution in a reasonable computing time.

Olteanu and Meyer [8] have developed a simulated annealing based algorithm to learn a MR-Sort model with classical veto (not coalitional ones).

In this paper, we propose a new heuristic algorithm to learn the parameters of a MR-Sort model with coalitional veto (called MR-Sort-CV) which makes use of the symmetry between the concordance and the coalitional veto conditions. Preliminary results obtained using an initial version of the algorithm can be found in [9]. The present work describes an improved version of the algorithm and the results of tests on real datasets involving two or more categories (while the preliminary version was only tested for classifying in two categories).

The paper is organized as follows. In Sect. 2, we recall MR-Sort the present work describes and define its extension when considering monocriterion veto and coalitional veto. After a brief reminder of the heuristic algorithm to learn an MR-Sort model, Sect. 3 is devoted to the presentation of the algorithm to learn an MR-Sort model with coalitional veto. Section 4 presents experimentations of this algorithm and Sect. 5 groups conclusions and directions for further research.

2 Considering Vetoes in MR-Sort

2.1 MR-Sort Model

MR-Sort is a method for assigning objects to ordered categories. It is a simplified version of ELECTRE TRI, another MCDA method [10,11].

The MR-Sort rule works as follows. Formally, let X be a set of objects evaluated on n ordered attributes (or criteria), $F = \{1, ..., n\}$. We assume that X is the Cartesian product of the criteria scales, $X = \prod_{j=1}^{n} X_j$, each scale X_j being completely ordered by the relation $\geq_j$. An object $a \in X$ is a vector $(a_1, \ldots, a_j, \ldots, a_n)$, where $a_j \in X_j$ for all j. The ordered categories which

the objects are assigned to by the MR-Sort model are denoted by C^h, with $h = 1, \ldots, p$. 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 $0 < h < p$). The profile b^h is the vector of criterion values $(b_1^h, \ldots, b_j^h, \ldots, b_n^h)$, with $b_j^h \in X_j$ for all j. We denote by $P = \{1, ..., p\}$ the list of category indices. 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, i.e.: $b_j^h \geq_j b_j^{h-1}$, for $h = \{1, \ldots, p\}$ and $j = \{1, \ldots, n\}$.

Using the MR-Sort procedure, an object is assigned to a category if its criterion values are at least as good as the category lower profile values on a weighted majority of criteria while this condition is not fulfilled when the object's criterion values are compared to the category upper profile values. In the former case, we say that the object is *weakly preferred* to the profile, while, in the latter, it is not[2]. Formally, if an object $a \in X$ is *weakly preferred* to a profile b^h, we denote this by $a \succcurlyeq b^h$. Object a is preferred to profile b^h whenever the following condition is met:

$$a \succcurlyeq b^h \Leftrightarrow \sum_{j:a_j \geq_j b_j^h} w_j \geq \lambda, \tag{1}$$

where w_j is the nonnegative weight associated with criterion 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*.

The preference relation $\succcurlyeq$ defined by (1) is called an *outranking* relation without veto or a *concordance* relation ([11]; see also [12,13] for an axiomatic description of such relations). Consequently, the condition for an object $a \in X$ to be assigned to category C^h reads:

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

The MR-Sort assignment rule described above involves $pn + 1$ parameters, i.e. n weights, $(p-1)n$ profiles evaluations and one majority threshold.

A *learning set* A is a subset of objects $A \subseteq X$ for which an assignment is known. For $h \in P$, 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.

2.2 MR-Sort-MV

In this section, we recall the traditional monocriterion veto rule as defined by [14,15]. In a MR-Sort model with *monocriterion veto*, an alternative a is "at least as good as" a profile b^h if it has at least equal to or better performances than b^h on a weighted majority of criteria and if it is not strongly worse than the profile on any criterion. In the sequel, we call b^h a *concordance profile* and we define "strongly worse than the profile" b^h by means of a veto profile

[2] "weak preference" means being "at least as good as".

$v^h = (v_1^h, v_2^h, ..., v_n^h)$, with $v_j^h \leq_j b_j^h$. It represents a vector of performances such that any alternative having a performance worse than or equal to this profile on any criterion would be excluded from category C^{h+1}. Formally, the assignment rule is described by the following condition:

$$a \succcurlyeq b^h \iff \sum_{j:a_j \geq_j b_j^h} w_j \geq \lambda \text{ and not } a\, V b^h,$$

with $a\, V b^h \iff \exists j \in F : a_j \leq_j v_j^h$. Note that non-veto condition is frequently presented in the literature using a veto threshold (see e.g. [16]), i.e. a maximal difference w.r.t. the concordance profile in order to be assigned to the category above the profile. Using veto profiles instead of veto thresholds better suits the context of multicriteria sorting. We recall that a profile b^h delimits the category C^h from C^{h+1}, with $C^{h+1} \succ C^h$; with monocriterion veto, the MR-Sort assignment rule reads as follows:

$$a \in C^h \iff \left[\sum_{j:a_j \geq_j b_j^{h-1}} w_j \geq \lambda \text{ and } \nexists j \in F : a_j \leq v_j^{h-1} \right] \text{ and } \left[\sum_{j:a_j \geq_j b_j^h} w_j < \lambda \text{ or } \exists j \in F : a_j \leq v_j^h \right]. \tag{3}$$

We remark that a MR-Sort model with more than 2 categories remains consistent only if veto profiles v_j^h do not overlap, i.e., are chosen such that $v_j^h \geq v_j^{h'}$ for all $\{h, h'\}$ s.t. $h > h'$. Otherwise, an alternative might be on the one hand in veto against a profile b^h, which prevents it to be assigned to C^{h+1} and, on the other hand, not in veto against b^{h+1}, which does not prevent it to be assigned to C^{h+2}.

2.3 MR-Sort-CV

We introduce here a new veto rule considering vetoes w.r.t. coalitions of criteria, which we call "*coalitional veto*". With this rule, the veto applies and forbids an alternative a to be assigned to category C^{h+1} when the performance of an alternative a is not better than v_j^h on a weighted majority of criteria.

As for the monocriterion veto, the veto profiles are vectors of performances $v^h = (v_1^h, v_2^h, ..., v_n^h)$, for all $h = \{1, .., p\}$. Coalitional veto also involves a set of nonnegative *veto weights* denoted z_j, for all $j \in F$. Without loss of generality, the sum of z_j is set to 1. Furthermore, a veto cutting threshold Λ is also involved and determines whether a coalition of criteria is sufficient to impose a veto. Formally, we express the coalitional veto rule $a\, V b^h$, as follows:

$$a\, V b^h \iff \sum_{j:a_j \leq_j v_j^h} z_j \geq \Lambda. \tag{4}$$

Using coalitional veto, the outranking relation of MR-Sort (2.2) is modified as follows:

$$a \succcurlyeq b^h \iff \sum_{j:a_j \geq_j b_j^h} w_j \geq \lambda \text{ and } \sum_{j:a_j \leq_j v_j^h} z_j < \Lambda. \tag{5}$$

Using coalitional veto with MR-Sort modifies the assignment rule as follows:

$$a \in C^h \iff \left[\sum_{j:a_j \geq_j b_j^{h-1}} w_j \geq \lambda \text{ and } \sum_{j:a_j \leq_j v_j^{h-1}} z_j < \Lambda \right] \text{ and } \left[\sum_{j:a_j \geq_j b_j^h} w_j < \lambda \text{ or } \sum_{j:a_j \leq_j v_j^h} z_j \geq \Lambda \right] \tag{6}$$

In MR-Sort, the coalitional veto can be interpreted as a combination of performances preventing the assignment of an alternative to a category. We call this new model, MR-Sort-CV.

The coalitional veto rule given in Eq. (5) is a generalization of the monocriterion rule. Indeed, if the veto cut threshold Λ is equal to $\frac{1}{n}$ (n being the number of criteria), and each veto weight z_j is set to $\frac{1}{n}$, then the veto rule defined in Eq. (4) corresponds to a monocriterion veto for each criterion.

2.4 The Non Compensatory Sorting (NCS) Model

In this subsection, we recall the non compensatory sorting (NCS) rule as defined by [14,15], which will be used in the experimental part (Sect. 4) for comparison purposes. These rules allow to model criteria interactions. MR-Sort is a particular case of these, in which criteria do not interact.

In order to take criteria interactions into account, it has been proposed to modify the definition of the global outranking relation, $a \succcurlyeq b^h$, given in (1). We introduce the notion of capacity. A *capacity* is a function $\mu : 2^F \to [0,1]$ such that:

- $\mu(B) \geq \mu(A)$, for all $A \subseteq B \subseteq F$ (monotonicity);
- $\mu(\emptyset) = 0$ and $\mu(F) = 1$ (normalization).

The Möbius transform allows to express the capacity in another form:

$$\mu(A) = \sum_{B \subseteq A} m(B), \tag{7}$$

for all $A \subseteq F$, with $m(B)$ defined as:

$$m(B) = \sum_{C \subseteq B} (-1)^{|B|-|C|} \mu(C) \tag{8}$$

The value $m(B)$ can be interpreted as the weight that is exclusively allocated to B as a whole. A capacity can be defined directly by its Möbius transform

also called "interaction". An interaction m is a set function $m : 2^F \to [-1, 1]$ satisfying the following conditions:

$$\sum_{j \in K \subseteq J \cup \{j\}} m(K) \geq 0, \quad \forall j \in F, J \subseteq F \backslash \{i\} \tag{9}$$

and

$$\sum_{K \subseteq F} m(K) = 1.$$

If m is an interaction, the set function defined by $\mu(A) = \sum_{B \subseteq A} m(B)$ is a capacity. Conditions (9) guarantee that μ is monotone [17].

Using a capacity to express the weight of the coalition in favor of an object, we transform the outranking rule as follows:

$$\begin{aligned} a \succcurlyeq b^h \Leftrightarrow \mu(A) \geq \lambda \text{ with } A = \{j : a_j \geq_j b_j^h\} \\ \text{and } \mu(A) = \sum_{B \subseteq A} m(B) \end{aligned} \tag{10}$$

Computing the value of $\mu(A)$ with the Möbius transform induces the evaluation of $2^{|A|}$ parameters. In a model composed of n criteria, it implies the elicitation of 2^n parameters, with $\mu(\emptyset) = 0$ and $\mu(F) = 1$. To reduce the number of parameters to elicit, we use a 2-additive capacity in which all the interactions involving more than 2 criteria are equal to zero. Inferring a 2-additive capacity for a model having n criteria requires the determination of $\frac{n(n+1)}{2} - 1$ parameters.

Finally, the condition for an object $a \in X$ to be assigned to category C^h can be expressed as follows:

$$\mu(F_{a,h-1}) \geq \lambda \quad \text{and} \quad \mu(F_{a,h}) < \lambda \tag{11}$$

with $F_{a,h-1} = \{j : a_j \geq_j b_j^{h-1}\}$ and $F_{a,h} = \{j : a_j \geq_j b_j^h\}$.

3 Learning MR-Sort

Learning the parameters of MR-Sort and ELECTRE TRI models has been studied in several articles [3,18–25]. In this section, we recall how to learn the parameters of a MR-Sort model using respectively an exact method [3] and a heuristic algorithm [18]. We then extend the heuristic algorithm to MR-Sort-CV.

3.1 Learning a Simple MR-Sort Model

It is possible to learn a MR-Sort model from a learning set using Mixed Integer Programming (MIP), see [3]. Such a MIP formulation is not suitable for large data sets because of the high computing time required to infer the MR-Sort parameters. In order to learn MR-Sort models in the context of large data sets,

a heuristic algorithm has been proposed in [18]. As for the MIP, the heuristic algorithm takes as input a set of assignment examples and their vectors of performances. The algorithm returns the parameters of a MR-Sort model.

The heuristic algorithm proposed in [18] works as follows. First a population of N_{mod} MR-Sort models is initialized. Thereafter, the following two steps are repeated iteratively on each model in the population:

$$
\begin{aligned}
&\min \textstyle\sum_{a \in A}(x'_a + y'_a)\\
&\text{s.t.}\\
&\textstyle\sum_{j:a_j \geq_j b_j^{h-1}} w_j - x_a + x'_a = \lambda \qquad \forall a \in A_h, h = \{2, ..., p\}\\
&\textstyle\sum_{j:a_j \geq_j b_j^{h}} w_j + y_a - y'_a = \lambda - \epsilon \quad \forall a \in A_h, h = \{1, ..., p-1\}\\
&\textstyle\sum_{j=1}^{n} w_j = 1\\
&w_j \in [0;1] \quad \forall j \in F\\
&\lambda \in [0;1]\\
&x_a, y_a, x'_a, y'_a \in \mathbb{R}_0^+\\
&\varepsilon \text{ a small positive number.}
\end{aligned}
\tag{12}
$$

1. A linear program optimizes the weights and the majority threshold on the basis of assignment examples while keeping the profiles fixed.
2. Given the inferred weights and the majority threshold, a heuristic adjusts the profiles of the model on the basis of the assignment examples.

After applying these two steps to all the models in the population, the $\lfloor \frac{N_{\text{mod}}}{2} \rfloor$ models restoring the least numbers of examples are reinitialized. These steps are repeated until the heuristic finds a model that fully restores all the examples or after a number of iterations specified a priori. This approach can be viewed as a sort of evolutionary metaheuristic (without crossover) in which a population of models is evolved.

The linear program designed to learn the weights and the majority threshold is given by (12). It minimizes a sum of slack variables, x'_a and y'_a, that is equal to 0 when all the objects are correctly assigned, i.e. assigned to the category defined in the input data set. We remark that the objective function of the linear program does not explicitly minimize the 0/1 loss but a sum of slacks. This implies that compensatory effects might appear, with undesirable consequences on the 0/1 loss. However in this heuristic, we consider that these effects are acceptable. The linear program doesn't involve binary variables. Therefore, the computing time remains reasonable when the size of the problem increases.

The objective function of the heuristic varying the profiles maximizes the number of examples compatible with the model. To do so, it iterates over each profile b^h and each criterion j and identifies a set of candidate moves for the profile, which correspond to the performances of the examples on criterion j located between profiles b^{h-1} and b^{h+1}. Each candidate move is evaluated as a function of the probability to improve the classification accuracy of the model. To assess whether a candidate move is likely to improve the classification of one or several objects, the examples which have an evaluation on criterion j located between the current value of the profile, b_j^h, and the candidate move, $b_j^h + \delta$ (resp. $b_j^h - \delta$), are grouped in different subsets:

$V_{h,j}^{+\delta}$ **(resp.** $V_{h,j}^{-\delta}$**):** the sets of objects 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.

$W_{h,j}^{+\delta}$ **(resp.** $W_{h,j}^{-\delta}$**):** the sets of objects 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.

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

$R_{h,j}^{+\delta}$ **(resp.** $R_{h,j}^{-\delta}$**):** the sets of objects 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 weakens the criteria coalition in favor of the correct classification but does not induce misclassification by itself.

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

A formal definition of these sets can be found in [18]. The evaluation of the candidate moves is done by aggregating the number of elements in each subset. Finally, the choice to move or not the profile on the criterion is determined by comparing the candidate move evaluation to a random number drawn uniformly. These operations are repeated multiple times on each profile and each criterion.

3.2 Learning a MR-Sort-CV Model

As compared with MR-Sort, a MR-Sort-CV model requires the elicitation of additional parameters: a veto profile, veto weights and a veto threshold. In (2), the MR-Sort condition $\sum_{j:a_j \geq_j b_j^{h-1}} w_j \geq \lambda$ is a necessary condition for an alternative to be assigned to a category at least as good as C^h. Basically, the coalitional veto rule can be viewed as a dual version of the majority rule. It provides a sufficient condition for being assigned to a category worse than C^h. An alternative a is assigned to such a category as soon as $\sum_{j:a_j \leq v_j^{h-1}} z_j \geq \lambda$. This condition has essentially the same form as the MR-Sort rule except that the sum is over the criteria on which the alternative's performance is at *most* as good as the profile (instead of "at least as good", for the MR-Sort rule).

In order to learn a MR-Sort-CV model, we modify the heuristic presented in the previous subsection as shown in Algorithm 1. The main differences with the MR-Sort heuristic are highlighted in grey.

In the rest of this section, we give some detail about the changes that have been brought to the heuristic.

Concordance weights optimization. The concordance profiles being given, the weights are optimized using the linear program (12). The sum of the error variables $x'_a + y'_a$ was the objective to be minimized. In the linear program,

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

Generate a population of N_{model} models with concordance profiles initialized with a heuristic
repeat
  for all model M of the set **do**
    Learn the concordance weights and majority threshold with a linear program, using the current concordance profiles
    Apply the heuristic N_{it} times to adjust the concordance profiles, using the current concordance weights and threshold.
    Initialize a set of veto profiles, taking into account the concordance profiles (in the first step and in case the coalitional veto rule was discarded in the previous step)
    Learn the veto weights and majority threshold with a linear program, using the current profiles
    Adjust the veto profiles by applying the heuristic N_{it} times, using the current veto weights and threshold
    Discard the coalitional veto if it does not improve classification accuracy
  end for
  Reinitialize the $\left\lfloor \frac{N_{model}}{2} \right\rfloor$ models giving the worst CA
until Stopping criterion is met

x'_a is set to a positive value whenever it is not possible to satisfy the condition which assigns a to a category at least as good as C^h, while a actually belongs to C^h. Impeding the assignment of positive values to x'_a amounts to favor false positive assignments. Hence, positive values of x'_a should be heavily penalized. In contrast, positive values of y'_a correspond to the case in which the conditions for assigning a to the categories above the profile are met while a belongs to the category below the profile. Positive values of y'_a need not be discouraged as much as those of x'_a and therefore we changed the objective function of the linear program into $\min \sum_{a \in A} 10x'_a + y'_a$.

Adjustment of the concordance profiles. In order to select moves by a quantity $\pm\delta$ applied to the profile level on a criterion, we compute a probability which takes into account the sizes of the sets listed at the end of Section 3.1. To ensure the consistency of the model, the candidate moves are located in the interval $[\max(b_j^{h-1}, v_j^h), b_j^{h+1}]$. In all cases, the movements which lower the profile ($-\delta$) are more favorable to false positive than the opposite movements. Therefore, all other things being equal (i.e. the sizes of the sets), the probability of choosing a downward move $-\delta$ should be larger than that of an upward move $+\delta$. The probability of an upward move is thus computed by the following formula

$$P(b_j^h + \delta) = \frac{2|V_{h,j}^{-\delta}| + 1|W_{h,j}^{-\delta}| + 0.1|T_{h,j}^{-\delta}|}{|V_{h,j}^{-\delta}| + |W_{h,j}^{-\delta}| + |T_{h,j}^{-\delta}| + 5|Q_{h,j}^{-\delta}| + |R_{h,j}^{-\delta}|}, \tag{13}$$

while that of a downward move is

$$P(b_j^h - \delta) = \frac{4|V_{h,j}^{+\delta}| + 2|W_{h,j}^{+\delta}| + 0.1|T_{h,j}^{+\delta}|}{|V_{h,j}^{+\delta}| + |W_{h,j}^{+\delta}| + |T_{h,j}^{+\delta}| + 5|Q_{h,j}^{+\delta}| + |R_{h,j}^{+\delta}|}. \tag{14}$$

The values appearing in (13) and (14) were determined empirically.

Initialization of veto profiles. A randomly selected veto profile is associated with each concordance profile. The veto profiles are initialized in ascending order, i.e. from the profile delimiting the worst category to the one delimiting the best category. The generation of a veto profile is done by drawing a random number in the interval $[b_j^h, v_j^{h-1}]$ on each criterion j. For the profile delimiting the worst categories, v_j^{h-1} corresponds to the worst possible performance on criterion j. Veto rules that do not prove useful at the end of the improvement phase (i.e., that do not contribute to reduce misclassification) are discarded and new veto profiles are generated during the next loop.

Veto weights optimization. The same type of linear program is used as for the concordance weights. One difference lies in the objective function in which we give the same importance to the positive and negative slack variables. Another difference lies in the constraints ensuring that an alternative $a \in C^h$ should not outrank the profile b^{h+1}. In order not to outrank the profile b^{h+1}, an alternative should either fulfill the condition $\sum_{j=1}^n w_j < \lambda$ or the condition $\sum_{j=1}^n z_j \geq \Lambda$ Since this is a disjunction of conditions, it is not necessary to ensure both. Therefore we do not consider the alternatives $a \in C^h$ that satisfy the first condition ($\sum_{j=1}^n w_j < \lambda$) in the linear program.

Adjustment of the veto profiles. The same heuristic as for the concordance profiles is used. The coefficients in (13) and (14) are modified in order to treat equally upward and downward moves.

4 Experiments

4.1 Datasets

In view of assessing the performance of the heuristic algorithm designed for learning the parameters of a MR-Sort-CV model, we use the algorithm to learn MR-Sort-CV models from several real data sets available at http://www.uni-marburg.de/fb12/kebi/research/repository/monodata, which serve as benchmark to assess monotone classification algorithms [4]. They involve from 120 to 1728 instances, from 4 to 8 monotone attributes and from 2 to 36 categories (see Table 1).

In our experiments, categories were initially binarized by thresholding at the median. We split the datasets in a twofold 50/50 partition: a learning set and a test set. Models are learned on the first set and evaluated on the test set; this is done 100 times on learning sets drawn at random.

4.2 Results Obtained with the Binarized Datasets

In a previous experimental study [6] performed on the same datasets, we compared the classification accuracy on the test sets obtained with MR-Sort and NCS. These results are reproduced in columns 2 and 3 of Table 2. No significant improvement in classification accuracy was observed when comparing NCS to MR-Sort. We have added the results for the new MR-Sort-CV heuristic in the fourth column of this table. In four cases, no improvement is observed as compared with MR-Sort. A slight improvement (of the order of 1%) is obtained in three cases (CPU, ESL, LEV). In one case (BCC), the results are slightly worse (2%).

Table 1. Data sets characteristics

Data set	#instances	#attributes	#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

4.3 Results Obtained with the Original Datasets

We also checked the algorithm for assigning alternatives to the original classes, in case there are more than two classes. Datasets with more than two classes

Table 2. Average and standard deviation of the classification accuracy on the test sets obtained with three different heuristics

Data set	MR-Sort	NCS	MR-Sort-CV
DBS	0.8377 ± 0.0469	0.8312 ± 0.0502	0.8390 ± 0.0476
CPU	0.9325 ± 0.0237	0.9313 ± 0.0272	0.9429 ± 0.0244
BCC	0.7250 ± 0.0379	0.7328 ± 0.0345	0.7044 ± 0.0299
MPG	0.8219 ± 0.0237	0.8180 ± 0.0247	0.8240 ± 0.0391
ESL	0.8996 ± 0.0185	0.8970 ± 0.0173	0.9024 ± 0.0179
MMG	0.8268 ± 0.0151	0.8335 ± 0.0138	0.8267 ± 0.0119
ERA	0.7944 ± 0.0173	0.7944 ± 0.0156	0.7959 ± 0.0270
LEV	0.8408 ± 0.0122	0.8508 ± 0.0188	0.8551 ± 0.0171
CEV	0.8516 ± 0.0091	0.8662 ± 0.0095	0.8516 ± 0.0665

are CPU, MPG, ESL, ERA, LEV and CEV. We did not consider MPG and ESL in which alternatives are respectively assigned to 36 and 9 categories. It is not reasonable indeed to aim at learning 35 (resp. 9) concordance and veto profiles on the basis of 392 (resp. 488) assignment examples in the dataset, half of them being reserved for testing purposes. The results obtained with the four remaining datasets are reported in Table 3.

Table 3. Average and standard deviation of the accuracy of classification in more than two categories obtained for the test sets

Data set	MR-Sort	MR-SortCV
CPU	0.8039 ± 0.0354	0.8469 ± 0.0426
ERA	0.5123 ± 0.0233	0.5230 ± 0.0198
LEV	0.5662 ± 0.0258	0.5734 ± 0.0213
CEV	0.7664 ± 0.0193	0.7832 ± 0.0130

We observe improvements w.r.t. MR-Sort on all four datasets. The gain ranges between a little less than 1% for ERA till 4% for CPU.

4.4 Results Obtained Using Randomly Generated MR-Sort-CV models

In view of the slight improvements w.r.t. MR-Sort obtained on the benchmark datasets, one may wonder whether this result should not be ascribed to the design of our heuristic algorithm. In order to check whether our algorithm is able to learn a MR-Sort-CV model in case the objects have been assigned to categories according to a hidden MR-Sort-CV rule, we performed the following experiments.

For a number of criteria ranging from 4 to 7, we generate at random a MR-Sort-CV model with two categories. We generate a learning set composed of 1000 random vectors of alternatives. Then we assign the alternatives to the two categories using the MR-Sort-CV model. We use the algorithm to learn a MR-Sort-CV model that reproduces as accurately as possible the assignments of the alternatives in the learning set. Having generated 10000 additional alternatives at random and having assigned them to a category using the generated MR-Sort-CV model, we compare these assignments with those produced by the learned model. We repeat this 100 times for each number of criteria. The average classification accuracy for the learning and the test sets is displayed in Table 4.

We observe that, on average, the learned model correctly restores more than 98.5% of the assignments in the learning set and more than 97.5% of the assignments in the test set. This means that our heuristic algorithm is effective in learning a MR-Sort-CV model when the assignments are actually made according to such a model.

Table 4. Average and standard deviation of the classification accuracy of MR-Sort-CV models learned on data generated by random MR-Sort-CV models with 2 categories and 4 to 7 criteria. The learning set is composed of 1000 alternatives and the test set is composed of 10000 alternatives.

#criteria	Learning set	Test set
4	0.9908 ± 0.01562	0.98517 ± 0.01869
5	0.9904 ± 0.01447	0.98328 ± 0.01677
6	0.9860 ± 0.01560	0.97547 ± 0.02001
7	0.9827 ± 0.01766	0.96958 ± 0.02116

An alternative explanation of the modest improvements yielded by using the MR-Sort-CV model is that the latter has limited additional descriptive power as compared to MR-Sort. We check this hypothesis by running the MR-Sort learning algorithm on the same artificial datasets generated by random MR-Sort-CV rules as in Table 4. The experimental results are reported in Table 5. By comparing Tables 4 and 5, we see that MR-Sort models are able to approximate very well MR-Sort-CV models. Indeed, the classification accuracy obtained with MR-Sort models is quite high on the test sets and only about 2% below that obtained with a learned MR-Sort-CV model.

Table 5. Average and standard deviation of the classification accuracy of MR-Sort models learned on data generated by random MR-Sort-CV models with 2 categories and 4 to 7 criteria. The learning set is composed of 1000 alternatives and the test set is composed of 10000 alternatives.

#criteria	Learning set	Test set
4	0.9760 ± 0.0270	0.9700 ± 0.0309
5	0.9713 ± 0.0275	0.9627 ± 0.0318
6	0.9645 ± 0.0248	0.9525 ± 0.0307
7	0.9639 ± 0.0264	0.9518 ± 0.0301

5 Conclusion

We have presented MR-Sort-CV, a new original extension of the MR-Sort ordered classification model. This model introduces a new and more general form of veto condition which applies on coalitions of criteria rather than a single criterion. This coalitional veto condition can be expressed as a "reversed" MR-Sort rule. Such a symmetry enables us to design a heuristic algorithm to learn an MR-Sort-CV model, derived from an algorithm used to learn MR-Sort.

The experimental results obtained on benchmark datasets show that there is no significant improvement in classification accuracy as compared with MR-Sort

in the case of two categories. The new model and the proposed learning algorithm lead to modest but definite gains in classification accuracy in the case of several categories. We checked that the learning algorithm was not responsible for the weak improvement of the assignment accuracy on the test sets. Therefore, the conclusion should be that the introduction of coalitional veto only adds limited descriptive ability to the MR-Sort model. This was also checked empirically.

The fact that coalitional veto (and this holds *a fortiori* for ordinary, single-criterion, veto) adds little descriptive power to the MR-Sort model is an important information by itself. In a learning context, the present study indicates that there is little hope to substantially improve classification accuracy by moving from a MR-Sort to a MR-Sort-CV model (as was also the case with the NCS model [6]). Note that small improvements in classification accuracy may be valuable, for instance, in medical applications (see e.g. [26]). Therefore it may be justified to consider MR-Sort-CV in spite of the increased complexity of the algorithm and of the model interpretation as compared to MR-Sort. It should be emphasized that the MR-Sort models lean themselves to easy interpretation in terms of rules [26]. The MR-Sort-CV model, although more complex, inherits this property since the coalitional veto condition is a reversed MR-Sort rule. Therefore, MR-Sort-CV models may be useful in specific applications.

References

1. Doumpos, M., Zopounidis, C.: Multicriteria Decision Aid Classification Methods. Kluwer Academic Publishers, Dordrecht (2002)
2. Zopounidis, C., Doumpos, M.: Multicriteria classification and sorting methods: a literature review. Eur. J. Oper. Res. **138**(2), 229–246 (2002)
3. Leroy, A., Mousseau, V., Pirlot, M.: Learning the parameters of a multiple criteria sorting method. In: Brafman, R.I., Roberts, F.S., Tsoukiàs, A. (eds.) ADT 2011. LNCS (LNAI), vol. 6992, pp. 219–233. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24873-3_17
4. Tehrani, A.F., Cheng, W., Dembczyński, K., Hüllermeier, E.: Learning monotone nonlinear models using the Choquet integral. Mach. Learn. **89**(1–2), 183–211 (2012)
5. Sobrie, O., Pirlot, M., Mousseau, V.: New veto relations for sorting models. Technical report, Laboratoire Génie Industriel, Ecole Centrale Paris Cahiers de recherche 2014–04, October 2014
6. Sobrie, O., Mousseau, V., Pirlot, M.: Learning the parameters of a non compensatory sorting model. In: Walsh, T. (ed.) ADT 2015. LNCS (LNAI), vol. 9346, pp. 153–170. Springer, Cham (2015). doi:10.1007/978-3-319-23114-3_10
7. Sobrie, O., Mousseau, V., Pirlot, M.: Learning the parameters of a majority rule sorting model taking attribute interactions into account. In: DA2pPL 2014 Workshop From Multiple Criteria Decision Aid to Preference Learning, pp. 22–30, Paris, France (2014)
8. Olteanu, A.L., Meyer, P.: Inferring the parameters of a majority rule sorting model with vetoes on large datasets. In: DA2pPL 2014 : From Multicriteria Decision Aid to Preference Learning, pp. 87–94 (2014)

9. Sobrie, O., Mousseau, V., Pirlot, M.: Learning MR-sort rules with coalitional veto. In: DA2pPL 2016: From Multicriteria Decision Aid to Preference Learning, p. 7 (2016)
10. 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)
11. Roy, B., Bouyssou, D.: Aide multicritère à la décision: méthodes et cas. Economica Paris (1993)
12. Bouyssou, D., Pirlot, M.: A characterization of concordance relations. Eur. J. Oper. Res. **167**(2), 427–443 (2005)
13. Bouyssou, D., Pirlot, M.: Further results on concordance relations. Eur. J. Oper. Res. **181**, 505–514 (2007)
14. Bouyssou, D., Marchant, T.: An axiomatic approach to noncompensatory sorting methods in MCDM, I: the case of two categories. Eur. J. Oper. Res. **178**(1), 217–245 (2007)
15. Bouyssou, D., Marchant, T.: An axiomatic approach to noncompensatory sorting methods in MCDM, II: more than two categories. Eur. J. Oper. Res. **178**(1), 246–276 (2007)
16. Roy, B.: The outranking approach and the foundations of ELECTRE methods. Theor. Decis. **31**, 49–73 (1991)
17. Chateauneuf, A., Jaffray, J.-Y.: Derivation of some results on monotone capacities by Mobius inversion. In: Bouchon, B., Yager, R.R. (eds.) IPMU 1986. LNCS, vol. 286, pp. 95–102. Springer, Heidelberg (1987). doi:10.1007/3-540-18579-8_8
18. Sobrie, O., Mousseau, V., Pirlot, M.: Learning a majority rule model from large sets of assignment examples. In: Perny, P., Pirlot, M., Tsoukiàs, A. (eds.) ADT 2013. LNCS (LNAI), vol. 8176, pp. 336–350. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41575-3_26
19. Mousseau, V., Słowiński, R.: Inferring an ELECTRE TRI model from assignment examples. J. Global Optim. **12**(1), 157–174 (1998)
20. Mousseau, V., Figueira, J.R., Naux, J.P.: Using assignment examples to infer weights for ELECTRE TRI method: some experimental results. Eur. J. Oper. Res. **130**(1), 263–275 (2001)
21. The, A.N., Mousseau, V.: Using assignment examples to infer category limits for the ELECTRE TRI method. J. Multi-criteria Decis. Anal. **11**(1), 29–43 (2002)
22. Dias, L., Mousseau, V., Figueira, J.R., Clímaco, J.: An aggregation/disaggregation approach to obtain robust conclusions with ELECTRE TRI. Eur. J. Oper. Res. **138**(1), 332–348 (2002)
23. 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. Eur. J. Oper. Res. **199**(2), 496–505 (2009)
24. Cailloux, O., Meyer, P., Mousseau, V.: Eliciting ELECTRE TRI category limits for a group of decision makers. Eur. J. Oper. Res. **223**(1), 133–140 (2012)
25. Zheng, J., Metchebon, S.A., Mousseau, V., Pirlot, M.: Learning criteria weights of an optimistic ELECTRE TRI sorting rule. Comput. OR **49**, 28–40 (2014)
26. Sobrie, O., Lazouni, M.E.A., Mahmoudi, S., Mousseau, V., Pirlot, M.: A new decision support model for preanesthetic evaluation. Comput. Methods Programs Biomed. **133**, 183–193 (2016)