Report for the decision systems project at CentraleSupelec

Decision systems inference based on a Majority or a non-compensatory Rule Sorting

AJEGHRIR Mustapha
mustapha.ajeghrir@student-cs.fr

TAZI Nouamane
nouamane.tazi@student-cs.fr

KHALD Asmae
asmae.khald@student-cs.fr

February 1, 2022

1 Introduction

Decision systems inference aims to discover the rules used by a decision maker (DM) based on decision examples, those rules could be used in the future to make the same decision as the DM. In this paper, we are going to study two types of decision rules : Majority rule (MR-sort) and non-compensatory rule (NCS).

1.1 Definition of MR-sort

Mr-sort aims to classify items based on n criteria ($\mathcal{N}$) over $p+1$ classes. Lets consider $(b_1^h, b_2^h, \dots, b_n^h) \in R^n \quad h \in \{1, \dots, p\}$ to be the profiles of class C_h, we are considering the case where classes are monotone, so being in class C_h mean to be in all the classes equal or lower than h. Lets also consider $\{w_1, \dots, w_n\}$ to be the weights for each criterion with $\sum_{i \in \mathcal{N}} w_i = 1$ and $\lambda \in [0.5, 1]$ a fixed threshold. Being in a class is defined as :

$$x \in C_h \iff \sum_{i \in \mathcal{N}:\ x_i \geq b_i^h} w_i \geq \lambda$$

1.2 Definition of NCS

NCS is similar to MR-sort, except there is no Majority or weights. The necessary condition to be in class C_h is to verify all the clauses linked this class, Those clauses are in the following form :

$$x \in C_h \iff \bigwedge_{i \in \mathcal{N}} (x_i \geq b_i^h)$$

2 Our task

The project that have been assigned to us is to invert the MR-sort and NCS judgments, which mean to infer the decision parameters of a DM from its decisions. For MR-sort, this means to find :

$$\begin{array}{ll} \lambda & \\ w_i & \forall i \in \mathcal{N} \\ b_i^h & \forall i \in \mathcal{N}, \forall h \in \{1 \dots p\} \end{array}$$

For NCS this means to find :

$$b_i^h \quad \forall i \in \mathcal{N}, \forall h \in \{1 \dots p\}$$

The input of our problem is only a sample of the DM's judgments, for each sample we have access to its values for each criterion and the DM's decision over it.

3 inv-MR-Sort

As defined previously we have to find the decision parameters. A practical way to do this is by using Linear programming with Gurobi. All the code we have used and benchmarks done can be found in our GitHub repository [1].

3.1 Mathematical representation

In order solve this problem mathematically we should add a continuous variable c_{ij}^h for each sample j and criterion i, this variable should be equal to 1 only and only if $x_i \geq b_i^h$ for the sample j. In addition we should also add some slack variables x and y for each sample, the objective function would be a variable α which aims to maximize all the slack variables of the problem. The resulting linear program will be[1] :

[1] Lets denote $K = \{1, 2, \dots, p\}$ and $K_{no\_end} = \{1, 2, \dots, p-1\}$ and $K_{no\_no\_end} = \{1, 2, \dots, p-2\}$

$$
\begin{aligned}
&Maximize \quad \alpha \\
&subject\ to \sum_{i \in N} c_{ij}^{h} + x_j + \epsilon = \lambda && \forall a_j \in C_h, \forall h \in K_{no\_end} \\
&\sum_{i \in N} c_{ij}^{h-1} = \lambda + y_j && \forall a_j \in C_h, \forall h \in \{2, \ldots, k\} \\
&\alpha \leq x_j, \alpha \leq y_j && \forall a_j \in C^* \\
&c_{ij}^{l} \leq w_i && \forall i \in N, \forall a_j \in C_h, \forall h \in K, \forall l \in \{h, h-1\} \cap K_{no\_end} \\
&c_{ij}^{l} \leq \delta_{ij}^{l} && \forall i \in N, \forall a_j \in C_h, \forall h \in K, \forall l \in \{h, h-1\} \cap K_{no\_end} \\
&c_{ij}^{l} \geq \delta_{ij}^{l} - 1 + w_i && \forall i \in N, \forall a_j \in C_h, \forall h \in K, \forall l \in \{h, h-1\} \cap K_{no\_end} \\
&M\delta_{ij}^{l} + \epsilon \geq g_i(a_j) - b_i^{l} && \forall i \in N, \forall a_j \in C_h, \forall h \in K, \forall l \in \{h, h-1\} \cap K_{no\_end} \\
&M(\delta_{ij}^{l} - 1) \leq g_i(a_j) - b_i^{l} && \forall i \in N, \forall a_j \in C_h, \forall h \in K, \forall l \in \{h, h-1\} \cap K_{no\_end} \\
&\sum_{i \in N} w_i = 1, \quad \lambda \in [0.5, 1] \\
&w_i \in [0, 1] && \forall i \in N \\
&c_{ij}^{l} \in [0, 1], \delta_{ij}^{l} \in \{0, 1\} && \forall i \in N, \forall a_j \in C_h, \forall h \in K, \forall l \in \{h, h-1\} \cap K_{no\_end} \\
&x_j, y_j \in R && \forall a_j \in C^* \\
&\alpha \in R
\end{aligned}
$$

The main problem with this linear program is it doesn't take in consideration the possibility of mistakes from the DM. For example, the decision maker might make a mistake in an item, so instead of labeling it to class C_h he labels it to be in C'_h. In order to account for that, it is possible to add a new binary variable γ_j for each sample. The goal should be to maximise it, so it should always be equal to 1 except if there is a problematic sample.

$$
\begin{aligned}
&Maximize \quad \sum_{a_j \in C^*} \gamma_j \\
&subject\ to \sum_{i \in N} c_{ij}^h + \epsilon \leq \lambda + M(1-\gamma_j) && \forall a_j \in C_h, \forall h \in K_{no\_end} \\
&\sum_{i \in N} c_{ij}^{h-1} \geq \lambda - M(1-\gamma_j) && \forall a_j \in C_h, \forall h \in \{2, \ldots, k\} \\
&c_{ij}^l \leq w_i && \forall i \in N, \forall a_j \in C_h, \forall h \in K, \forall l \in \{h, h-1\} \cap K_{no\_end} \\
&c_{ij}^l \leq \delta_{ij}^l && \forall i \in N, \forall a_j \in C_h, \forall h \in K, \forall l \in \{h, h-1\} \cap K_{no\_end} \\
&c_{ij}^l \geq \delta_{ij}^l - 1 + w_i && \forall i \in N, \forall a_j \in C_h, \forall h \in K, \forall l \in \{h, h-1\} \cap K_{no\_end} \\
&M\delta_{ij}^l + \epsilon \geq g_i(a_j) - b_i^l && \forall i \in N, \forall a_j \in C_h, \forall h \in K, \forall l \in \{h, h-1\} \cap K_{no\_end} \\
&M(\delta_{ij}^l - 1) \leq g_i(a_j) - b_i^l && \forall i \in N, \forall a_j \in C_h, \forall h \in K, \forall l \in \{h, h-1\} \cap K_{no\_end} \\
&b_i^{h+1} \geq b_i^h && \forall i \in N, \forall h \in K_{no\_no\_end} \\
&\sum_{i \in N} w_i = 1, \quad \lambda \in [0.5, 1] \\
&w_i \in [0,1] && \forall i \in N \\
&c_{ij}^l \in [0,1], \delta_{ij}^l \in \{0,1\} && \forall i \in N, \forall a_j \in C_h, \forall h \in K, \forall l \in \{h, h-1\} \cap K_{no\_end} \\
&x_j, y_j \in R && \forall a_j \in C^*
\end{aligned}
$$

For more information about the implementation we recommend *Learning the Parameters of a Multiple Criteria Sorting Method Based on a Majority Rule* [2], or *Learning MR-Sort Models from Non-Monotone Data* [3]

3.2 Dataset generation and Noise strategy

3.2.1 Parameters generation

In the benchmarking part, we generate DM's decision parameters randomly in the correct structure.
Lambda : λ is generated with a uniform distribution between 0.5 and 1 : $\mathcal{U}(0.5, 1)$.
Criteria weights : $w_i \ \forall i \in \mathcal{N}$ are generated with a uniform distribution $\mathcal{U}(0, 1)$ then normalized to sumup to 1.
Profiles : If we consider $[a, b]$ to be the interval of possible values for our criteria. We generate $b_i^h \ \forall h \in K, i \in \mathcal{N}$ with a uniform distribution $\mathcal{U}\left(\frac{1}{6}a, \frac{5}{6}b\right)$ then we sort them for a fixed i to have the condition $b_i^h \leq b_i^{h+1} \ \forall h \in K_{no\_end}, i \in \mathcal{N}$

3.2.2 Sample generation strategy

For each sample we generate a random evaluation for each criterion by following the next steps:

1. We chose a random profile among the available ones
2. We generate a random values for each criterion with a normal dstribution $\mathcal{N}(0, 2)$
3. We assign to our sample the sum of the profile and the random noise

3.2.3 Noise strategy

In order to test the robustness of our inference, we also created noisy datasets where the class of a sample is randomly attributed with a probability $error\_rate$:

```
if random(0, 1) < error_rate :          # attribute random class
    sample_class = random_int(0, p+1)
else :                                  # attribute normal class
    sample_class = MR_sort(sample)
```

3.2.4 Quantization

When we use the full double precision floats, we get many problems regarding numerical precision in the solver, we tried to tune many Gurobi parameters like **OptimalityTol**, **FeasibilityTol**, **IntFeasTol** and **Quad**. But we found it to be more Time-efficient to work with quantized values, we used the following function for quantization $f(x) = round(x \times q)/q$ for $q := quantization\_facotr = 10^6$

3.3 Results

3.3.1 Results with no noise

When there is no noise, the α parameter aims to increase all the slack variables, therefore resulting in very good results as we can see the confusion matrix 1.

Generalisation	Predicted Class 0	Predicted Class 1
True Class 0	751	0
True Class 1	0	249

Reconstruction	Predicted Class 0	Predicted Class 1
True Class 0	806	1
True Class 1	0	193

Figure 1: Confusion matrix with the default configuration ($n = 6$, $p = 1$, $n_{generated} = 1000$). Inference time : $40s$

We have even done some benchmarks to better understand the effect of our parameters as we can see in the figure 2

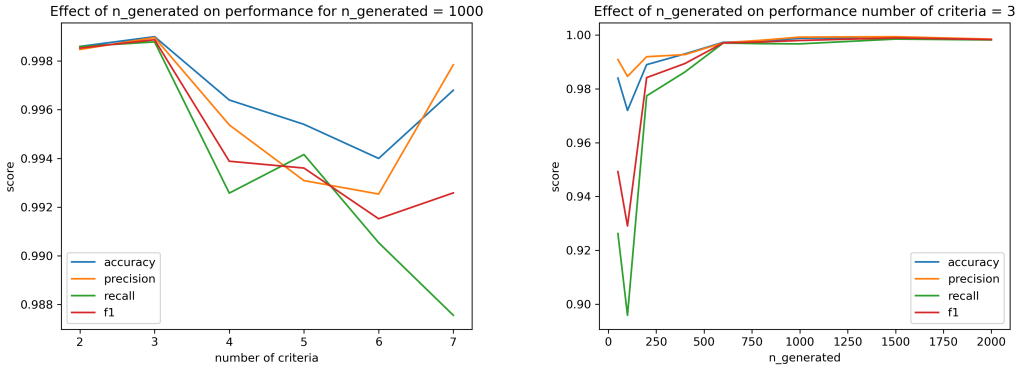

Figure 2: The MR-sort rules are first being inferred over a training split then tested over a testing split. The two splits are being generated by the same rules with only 2 classes ($p = 1$), results are the mean of 5 random runs. In the right we can see the effect of varying the number of elements in the training split $n_{generated}$, in the left we can see the effect of varying the number of criteria

Concerning Solving time, we can take a look at the figure 3. We can experimentally remark that the number of criteria increases the inference time exponentially while the number of samples only increases it linearly.

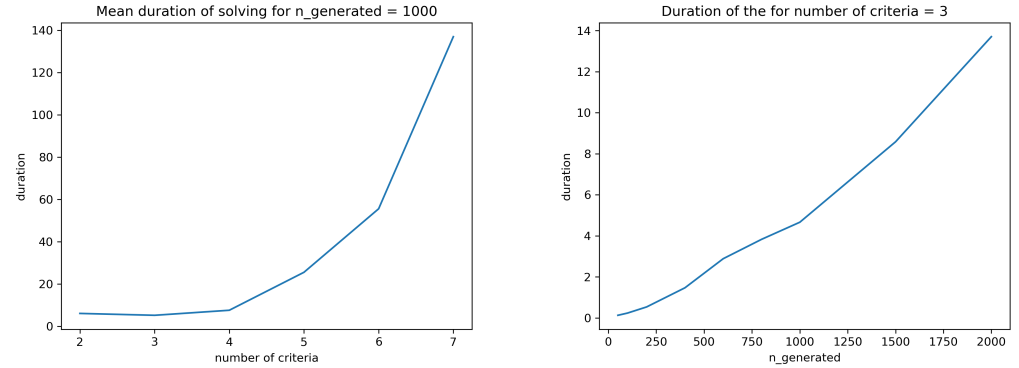

Figure 3: Duration of the MR-sort inference over the training split. The split is being generated by the same rules with only 2 classes ($p = 1$), results are the mean of 5 random runs. In the right we can see the effect of varying the number of elements in the training split $n_{generated}$, in the left we can see the effect of varying the number of criteria

3.3.2 Results with noise

When inferring over a noisy dataset or DM. We don't use slack variables, we only try to find a configuration that includes the maximum of samples, therefore the score is not being optimized. We can see in the confusion matrix 4 how the generalization ability drops with noise. But the reconstruction rate remains very high at 98% for 20% probability of error (17 real errors in this example).

Generalisation	Predicted Class 0	Predicted Class 1
True Class 0	138	6
True Class 1	0	56
Reconstruction	Predicted Class 0	Predicted Class 1
True Class 0	141	3
True Class 1	0	56

Figure 4: Confusion matrix with the default configuration ($n = 4$, $p = 1$, $n_{generated} = 200$, $N = 0.2$), N refers to the probability of random class attribution, if $N = 0.2$ this means 20% of the samples will get random class attribution regardless of the criteria. Inference time : $520s$

4 inv-NCS

The Non-Compensatory Sorting model aims at assigning alternatives evaluated on multiple criteria to one of the predefined ordered categories. It corresponds to a generalization of MR-sort. the inverse Non-Compensatory Sorting problem,takes as input a set of assignment examples, and computes an NCS sorting model which is consistent with this preference information. In other words, Inv-NCS learns the NCS parameters that perfectly match a set of desired outputs. In this work, we are interested in studying two SAT based representations for learning an NCS model from data, starting by just one profile and extending then a SAT formulation to multiple classes. After that, we will look for resolving a maxSAT formulation when the SAT one is unsatisfiable.

4.1 Mathematical representation

4.1.1 SAT formulation

a SAT formulation needs a set of clauses to be defined, thus a set of binary variables, that we defined as follows:

1. ' a ' variables, indexed by a criterion $i \in \mathcal{N}$, an exigence level $k \in [2.p]$ and a reference value $x \in X^\star$, represent the approved sets $\mathcal{A}_i^k$, with the following semantic: $a_{i,k,x} = 1 \Leftrightarrow x \in \mathcal{A}_i^k$ i.e. x is approved at level k according to i

2. ' t ' variables, indexed by a coalition of criteria $B \subseteq \mathcal{N}$ and an exigence level $k \in [2.p]$, represent the sufficient coalitions $\mathcal{T}^k$, with the following semantic: $t_{B,k} = 1 \Leftrightarrow B \in \mathcal{T}^k$ i.e. the coalition B is sufficient at level k

Given an instance of Inv-NCS with an assignment $\alpha : X^\star \to \left\{C^1 \prec \ldots \prec C^p\right\}$, the boolean function Φ_α^C with variables $\langle a_{i,k,x} \rangle_{i \in \mathcal{N}, k \in [2.p], x \in X^*}$ and $\langle t_{B,k} \rangle_{B \subseteq \mathcal{N}, k \in [2.p]}$, is defined as the conjunction of clauses:

$$
\begin{array}{l}
\Phi_\alpha^C = \phi_\alpha^{C1} \wedge \phi_\alpha^{C2} \wedge \phi_\alpha^{C3} \wedge \phi_\alpha^{C4} \wedge \phi_\alpha^{C5} \wedge \phi_\alpha^{C6} \\
\phi_\alpha^{C1} = \bigwedge_{i \in \mathcal{N}, k \in [2.p]} \bigwedge_{x' \ i} x \in X^* \\
\phi_\alpha^{C2} = \bigwedge_{i \in \mathcal{N}, k < k' \in [2.p], x \in X^*} (a_{i,k,X'} \vee \neg a_{i,k,X}) \\
\phi_\alpha^{C3} = \bigwedge_{B \subset B' \subseteq \mathcal{N}, k \in [2.p]} (t_{B',k} \vee \neg t_{B,k}) \\
\phi_\alpha^{C4} = \bigwedge_{B \subseteq \mathcal{N}, k < k' \in [2.p]} (t_{B,k} \vee \neg t_{B,k'}) \\
\phi_\alpha^{C5} = \bigwedge_{B \subseteq \mathcal{N}, k \in [2.p]} \bigwedge_{x \in \alpha^{-1}(C^{k-1})} \left(\bigvee_{i \in B} \neg a_{i,k,x} \vee \neg t_{B,k}\right) \\
\phi_\alpha^{C6} = \bigwedge_{B \subseteq \mathcal{N}, k \in [2.p]} \bigwedge_{x \in \alpha^{-1}(C^{k})} \left(\bigvee_{i \in B} a_{i,k,x} \vee t_{\mathcal{N} \setminus B,k}\right)
\end{array}
$$

Meaning of clauses:

• Clauses A1: if student validates a criterion i with evaluation k, then another student with criterion k'>k validates this criterion surely.
• Clauses A2: if student validates a criterion i with respect to the profile b'_h, then he must validate the criterion i with respect to the profile b_h $(h < h')$.
• Clauses A3: if B is sufficient then each B' containing B is sufficient.
• Clauses A4: if B is sufficient at level hp then B is sufficient at level $h < hp$.
• Clauses A5: if a student is in class h-1 and validates all criteria (i,h) in B, then B is not sufficient.
• Clauses A6: if a student is in class h and doesn't validate any criteria (i,h) in B, then complementary of B is sufficient.

4.1.2 MaxSAT formulation

When the SAT formulation isn't satisfiable with the learning set, the aim is to maximize the number of important clauses to satisfy; to do that, clauses must be weighted and another variable must be included:
' z ' variables, indexed by an alternative x, represent the set of alternatives properly classified by the inferred model, with the following semantic: $z_x = 1 \Leftrightarrow \alpha^{-1}(x) = NCS_\omega(x)$ i.e. the alternative x is properly classified.

These variables are introduced in some clauses to serve as switches: - For any exigence level $k \in [2.p]$, let $B \subseteq \mathcal{N}$ a coalition of criteria, and x an alternative assigned to C^{k-1} by α. If $z_k = 1$ and $B \subseteq \left\{i \in \mathcal{N} : x \in \mathcal{A}_i^k\right\}$ then $t_{B,k} = 0$. This leads to the following conjunction of clauses:

$$\phi_\alpha^{\widetilde{CS}} = \bigwedge_{B\subseteq\mathcal{N},k\in[2.p]} \bigwedge_{x\in\alpha^{-1}(C^{k-1})} \left(\bigvee_{i\in B} \neg a_{i,k,X} \vee \neg t_{B,k} \vee \neg z_X \right)$$

- For any exigence level $k \in [2.p]$, let $B \subseteq \mathcal{N}$ a coalition of criteria, and x an alternative assigned to C^k by α. If $z_k = 1$ and $B \subseteq \left\{i \in \mathcal{N} : x \in \mathcal{A}_i^k\right\}$ then $t_{\mathcal{N}\setminus B,k} = 0$. This leads to the following conjunction of clauses:

$$\phi_\alpha^{\widetilde{C6}} = \bigwedge_{B\subseteq\mathcal{N},k\in[2.p]} \bigwedge_{x\in\alpha^{-1}(C^{k})} \left(\bigvee_{i\in B} a_{i,k,x} \vee t_{\mathcal{N}\setminus B,k} \vee \neg z_x \right)$$

The objective in the MaxSAT formulation is to maximize the portion of alternatives properly classified, this is the subject of the following soft clause:

$$\phi_\alpha^{\text{goal}} = \bigwedge_{x\in X^*} z_x$$

Clauses composing the conjunctions $\phi_\alpha^{C1}, \phi_\alpha^{C2}, \phi_\alpha^{C3}, \phi_\alpha^{C4}, \phi_\alpha^{C5}$ and $\phi_\alpha^{\widetilde{C6}}$ are hard, associated to the weight $w_{\max}$, and we associate to $\phi_\alpha^{\text{goal}}$ the weight w_1 such that $w_{\max} > |X^\star|\, w_1$.

4.2 Dataset generation and Noise strategy

Learning data is a set of assignments; each row represents an alternative with its evaluations on each criterion and the category or class where it has to be assigned. Each row of alternative's evaluations is generated using a profile chosen randomly, with adding a random noise to each value that corresponds

to a criterion in the selected profile. Then,each set of evaluations generated (for one alternative) is assigned to a class/category depending on the arguments we have considered to generate data, and particularly the sufficient coalitions. We have also allowed some error when adding assignments in order to test the ability of the program to generalize.

We tried to ensure a balance between the classes in the learning set, i.e. no large difference in the number of assignments in each category, in order to have the best model.

alternative	criterion 0	criterion 1	criterion 2	criterion 3	class
0	8.02	14.22	9.84	9.03	0
1	8.96	10.78	11.07	9.03	0
2	12.61	10.76	15.98	10.54	2
3	10.13	10.77	9.93	8.92	1

Figure 5: set of data generated randomly and assigned depending on the arguments chosen including sufficient coalitions

4.3 Results

The output of the NCS resolution is a set of sufficient coalitions for each category. e.g.
Learnt sufficient coalitions:
(0, 1) at levels [1, 2]
(1, 2) at levels [1, 2]
(0, 1, 2) at levels [1, 2]

4.3.1 restoration ability

As expected, all SAT instances (without noise) are able to fully restore the learning sets; this result is an experimental validation of the theoretical work. Moreover, when learning a model from noisy learning sets (MaxSAT extension), we were able to infer NCS models with a restoration rate over 1 x, where x denotes the noise level in the learning set.

4.3.2 generalization ability

• an increase of the size of the learning set induces an improvement of the generalization index; such improvement occurs whatever the noise level (up to 20always possible to "capture the ground truth" with a sufficiently large learning set.
• an increase in the reference set noise level requires a larger learning set to keep the same generalization level. This implies that the "quality" of the learning set, have a significant impact on the required size of this learning set.

4.4 Single-peaked criterion

To solve the problem for single-peaked, we only had to change the clauses which ensured the monotony of the marks, which are the ϕ_{α}^{C1} clauses. So we replace them with this one

$$\phi_{\alpha\_SP}^{C1} = \bigwedge_{i\in\mathcal{N},k\in[2.p]} \bigwedge_{x_i' x" {}_i x\in X^*} (a_{i,k,X"} \vee \neg a_{i,k,X} \vee \neg a_{i,k,X'})$$

5 Conclusion

Our experimental results show that the duration of computation evolve exponentially with respect to the number of criteria for MR-Sort, linearly for Max-Sat, and linearly with respect to the size of the learning set. And the generalization index increases with the size of the learning set and decreases with the addition of number of criteria, all while restoring at least $1 - x$ of the data with x the noise percentage.

Finally, several computing scripts using *Gophersat* and *Gurobi* were proposed in our Github [1]. With the exact steps to reproduce the results presented in this paper.

References

[1] GitHub repository: `https://github.com/Mustapha-AJEGHRIR/projet_sys_decision`

[2] Learning the Parameters of a Multiple Criteria Sorting Method Based on a Majority Rule: `https://www.researchgate.net/publication/221367488_Learning_the_Parameters_of_a_Multiple_Criteria_Sorting_Method`

[3] Learning MR-Sort Models from Non-Monotone Data: `https://arxiv.org/pdf/2107.09668.pdf`

[4] D. Bouyssou, T. Marchant, 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).

[5] D. Bouyssou, T. Marchant, An axiomatic approach to noncompensatory sorting methods in MCDM, II: More than two categories, European Journal of Operational Research, 178(1):246–276, (2007).

[6] Eda Ersek Uyanik, Vincent Mousseau, Marc Pirlot, and Olivier Sobrie. Enumerating and categorizing positive boolean functions separable by a k-additive capacity. Discrete Applied Mathematics, 229:17-30, (2017).

[7] Agnes Leroy, Vincent Mousseau, and Marc Pirlot. Learning the parameters of a multiple criteria sorting method. Algorithmic Decision Theory, 219-233, (2011).

[8] Belahcene K., Labreuche C., Maudet N., Mousseau V., and Ouerdane, W. An efficient SAT formulation for learning multiple criteria non-compensatory sorting rules from examples, Computers Operations Research, 97, 58–71, (2018).