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

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

[1]École Centrale de Paris - Laboratoire de Génie Industriel
[2]University of Mons - Faculty of engineering

April 12, 2013

1 Introduction

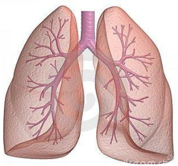
2 Algorithm

3 Experimentations

4 Conclusion

Introduction example

Application : Lung cancer



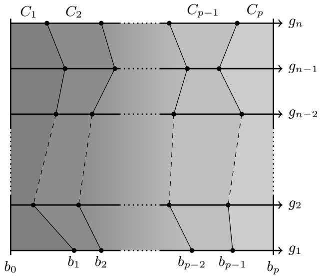
- 9394 patients analyzed
- Monotone attributes (number of cigarettes per day, age, ...)
- Output variable : no cancer, cancer, incurable cancer
- Predict the risk to get a lung cancer for other patients on basis of their attributes

MR-Sort procedure

Main characteristics

- Sorting procedure
- Simplified version of the ELECTRE TRI procedure [Yu, 1992]
- Axioms based [Slowínski et al., 2002, Bouyssou and Marchant, 2007a, Bouyssou and Marchant, 2007b]

Parameters



- Profiles' performances ($b_{h,j}$ for $h = 1, ..., p-1; j = 1, ..., n$)
- Criteria weights (w_j for $n = 1, ..., n$)
- Majority threshold (λ)

Number of parameters : $(2p-1)n+1$

Inference of the parameters

What already exists to infer MR-Sort parameters ?

- Mixed Integer Program learning the parameters of an MR-Sort model [Leroy et al., 2011]
- Metaheuristic to learn the parameters of an ELECTRE TRI model [Doumpos et al., 2009]
- Not suitable for large problems : computing time becomes huge when the number of parameters or examples increases

Our objective

- Learn a MR-Sort from a large set of assignment examples
- Efficient algorithm (i.e. can handle 1000 alternatives, 10 criteria, 5 categories)

Principe of our metaheuristic

Input parameters

- Assignment examples
- Performances of the examples on the n criteria

Objective

- Learn an MR-Sort model which is compatible with the highest number of assignment examples, i.e. maximize the classification accuracy,
  $CA = \frac{\text{Number of examples correctly restored}}{\text{Total number of examples}}$

Main parts of the algorithm

1. Initialization of a set of profiles
2. Learning the weights and majority threshold with a linear program
3. Adapt the profiles to increase the CA

Metaheuristic to infer all parameters

Algorithm

A population of N_{mod} models is initialized with the heuristic for the profiles
repeat
  Learn the weights and majority threshold with the linear program
  Run N_{meta} times the metaheuristic adjusting the profiles
  Reinitialize the $\frac{N_{mod}}{2}$ models having the worst CA
until Stop condition is met

Stop criterion

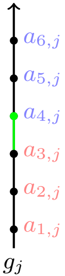
Stop criterion is met when one model has a CA equal to 1 or when the algorithm has run N_o times.

Initialization of the profiles

Principe

- By a heuristic
- On each criterion j, give to the profile a performance such that CA would be max for the alternatives belonging to h and $h+1$ if $w_j = 1$.
- Take the probability to belong to a category into account

Example 1 : Where should the profile be set on criterion j ?



$a_{i,j}$	Category
$a_{1,j}$	C_1
$a_{2,j}$	C_1
$a_{3,j}$	C_1
$a_{4,j}$	C_2
$a_{5,j}$	C_2
$a_{6,j}$	C_2

Category	$P(a_i \in C_h)$
C_1	$\frac{1}{2}$
C_2	$\frac{1}{2}$

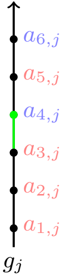
$$a_{3,j} < b_h \leq a_{4,j}$$

Initialization of the profiles

Principe

- By a heuristic
- On each criterion j, give to the profile a performance such that CA would be max for the alternatives belonging to h and $h+1$ if $w_j = 1$.
- Take the probability to belong to a category into account

Example 2 : Where should the profile be set on criterion j ?



$a_{i,j}$	Category
$a_{1,j}$	C_1
$a_{2,j}$	C_1
$a_{3,j}$	C_1
$a_{4,j}$	C_2
$a_{5,j}$	C_1
$a_{6,j}$	C_2

Category	$P(a_i \in C_h)$
C_1	$\frac{2}{3}$
C_2	$\frac{1}{3}$

$$a_{3,j} < b_h \leq a_{4,j}$$

Learning of the weights and majority threshold

Principe

- Maximizing the classification accuracy of the model
- Using a linear program with no binary variables

Linear program

$$\text{Objective : } \min \sum_{a_i \in A} (x_i' + y_i') \quad (1)$$

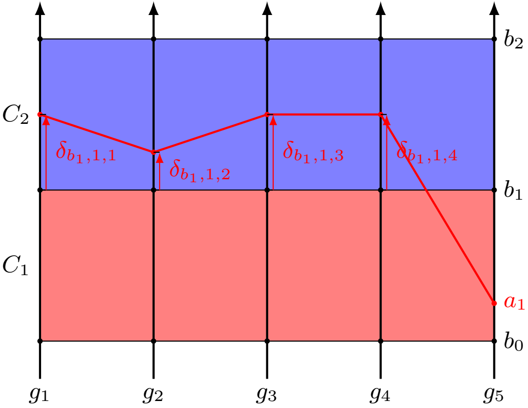
$$\sum_{\forall j | a_i S_j b_{h-1}} w_j - x_i + x_i' = \lambda \qquad \forall a_i \in A_h, h = \{2, ..., p-1\} \quad (2)$$

$$\sum_{\forall j | a_i S_j b_h} w_j + y_i - y_i' = \lambda - \delta \qquad \forall a_i \in A_h, h = \{1, ..., p-2\} \quad (3)$$

$$\sum_{j=1}^{n} w_j = 1 \quad (4)$$

Metaheuristic to adjust the profiles

Case 1 : Alternative a_1 classified in C_2 instead of C_1

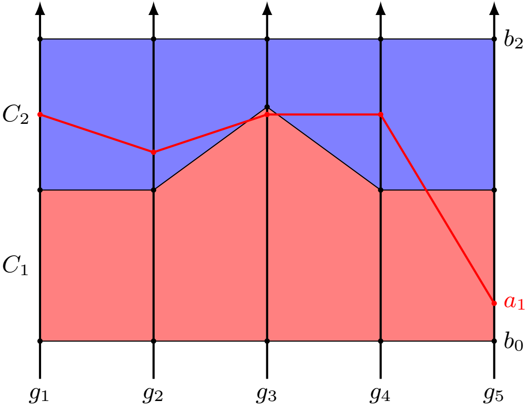


- a_1 is classified by the **DM** into category C_1
- a_1 is classified by the **model** into category C_2
- a_1 outranks b_1
- Profile too low on one or several criteria (in red)

$w_j = 0.2$ for $j = 1, ..., 5$; $\lambda = 0.8$

Metaheuristic to adjust the profiles

Case 1 : Alternative a_1 classified in C_2 instead of C_1

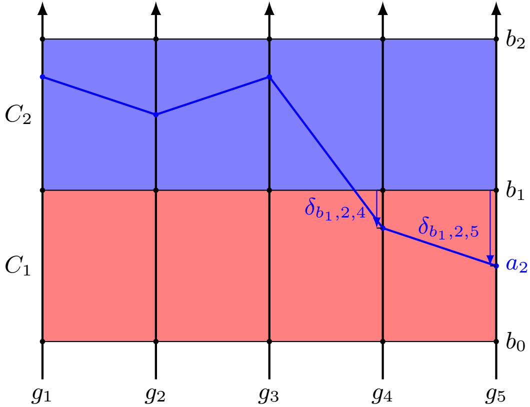


- a_1 is classified by the **DM** into category C_1
- a_1 is classified by the **model** into category C_2
- a_1 outranks b_1
- Profile too low on one or several criteria (in red)

$w_j = 0.2$ for $j = 1, ..., 5$; $\lambda = 0.8$

Metaheuristic to adjust the profiles

Case 2 : Alternative a_2 classified in C_1 instead of C_2

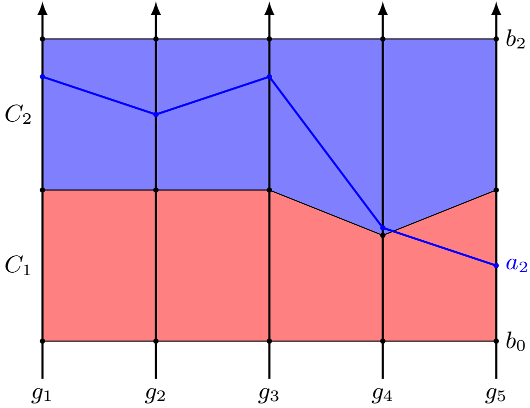


$w_j = 0.2$ for $j = 1, ..., 5$; $\lambda = 0.8$

- a_2 is classified by the **DM** into category C_2
- a_2 is classified by the **model** into category C_1
- a_2 doesn't outrank b_1
- Profile too high on one or several criteria (in blue)
- If profile moved by $\delta_{b_1,2,4}$ on g_4 and/or by $\delta_{b_1,2,5}$ on g_5, the alternative will be rightly classified

Metaheuristic to adjust the profiles

Case 2 : Alternative a_2 classified in C_1 instead of C_2

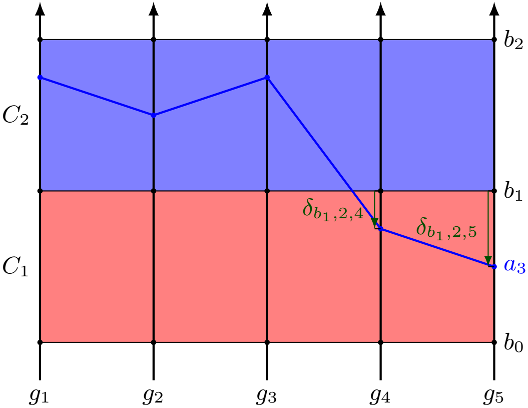


$w_j = 0.2$ for $j = 1, ..., 5$; $\lambda = 0.8$

- a_2 is classified by the **DM** into category C_2
- a_2 is classified by the **model** into category C_1
- a_2 doesn't outrank b_1
- Profile too high on one or several criteria (in blue)
- If profile moved by $\delta_{b_1,2,4}$ on g_4 and/or by $\delta_{b_1,2,5}$ on g_5, the alternative will be rightly classified

Metaheuristic to adjust the profiles

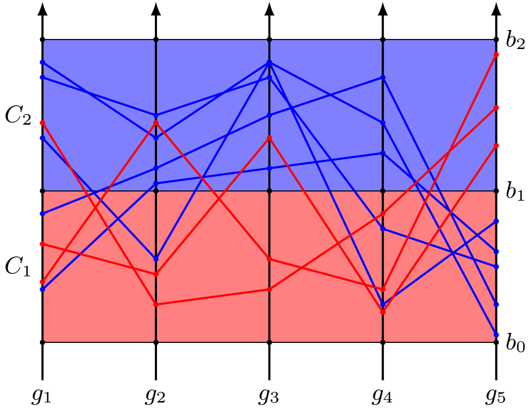
- $V_{h,j}^{\pm\delta}$: Set of alternatives classified into C_{h+1} instead of C_h or the contrary for which $b_{h,j}$ has a negative effect on the classification and for which moving the profile b_h of $\pm\delta$ on j will improve the classification



- $w_j = 0.2$ for $j = 1, ..., 5$
- $\lambda = 0.8$
- $a_3 \in A_{2 \leftarrow \text{DM}}^{1 \leftarrow \text{Model}}$

Metaheuristic to adjust the profiles

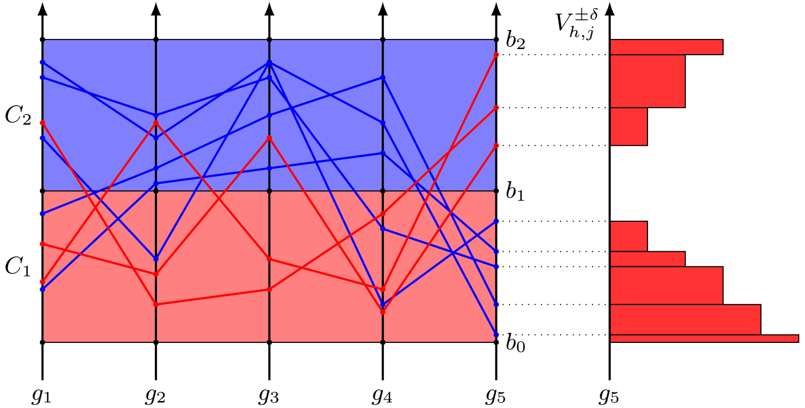
- $V_{h,j}^{\pm\delta}$: Set of alternatives classified into C_{h+1} instead of C_h or the contrary for which $b_{h,j}$ has a negative effect on the classification and for which moving the profile b_h of $\pm\delta$ on j will improve the classification



- $w_j = 0.2$ for $j = 1, ..., 5$
- $\lambda = 0.8$
- $a_3 \in A_{2 \leftarrow \text{DM}}^{1 \leftarrow \text{Model}}$

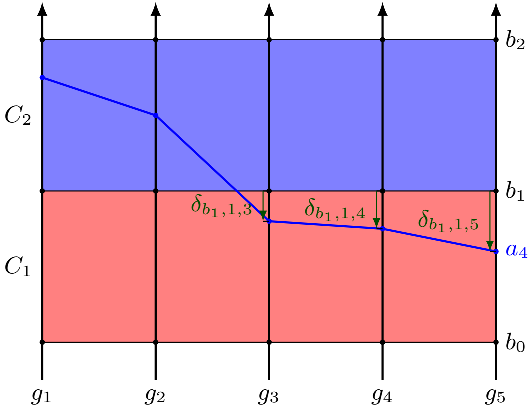
Metaheuristic to adjust the profiles

- $V_{h,j}^{\pm\delta}$: Set of alternatives classified into C_{h+1} instead of C_h or the contrary for which $b_{h,j}$ has a negative effect on the classification and for which moving the profile b_h of $\pm\delta$ on j will improve the classification



Metaheuristic to adjust the profiles

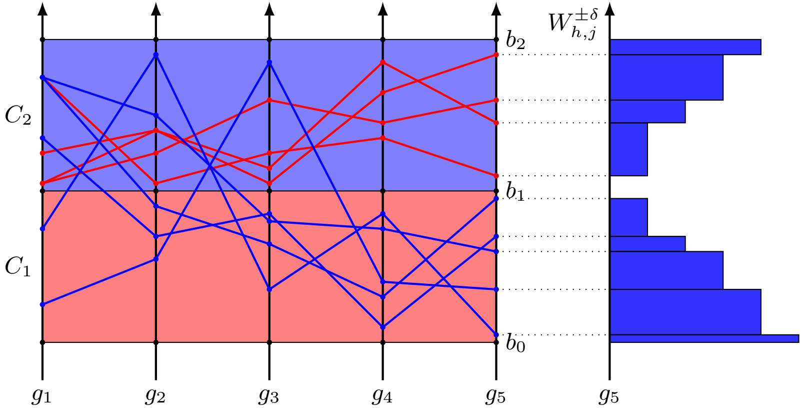
- $W_{h,j}^{\pm\delta}$: Set of alternatives classified into C_{h+1} instead of C_h or the contrary for which $b_{h,j}$ has a negative effect on the classification and for which moving the profile b_h of $\pm\delta$ on j will strengthen the criteria coalition in favor of the correct classification



- $w_j = 0.2$ for $j = 1, ..., 5$
- $\lambda = 0.8$
- $a_4 \in A_{1\leftarrow \text{DM}}^{2\leftarrow \text{Model}}$

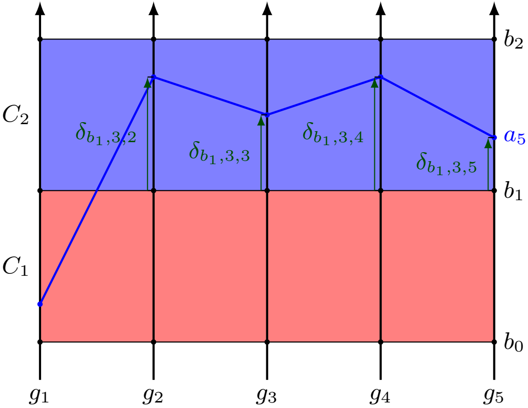
Metaheuristic to adjust the profiles

- $W_{h,j}^{\pm\delta}$: Set of alternatives classified into C_{h+1} instead of C_h or the contrary for which $b_{h,j}$ has a negative effect on the classification and for which moving the profile b_h of $\pm\delta$ on j will strengthen the criteria coalition in favor of the correct classification



Metaheuristic to adjust the profiles

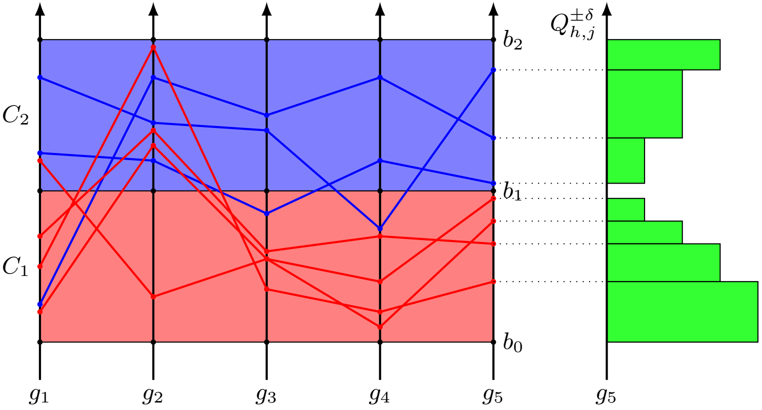
- $Q_{h,j}^{\pm\delta}$: Set of alternatives rightly classified into C_h or C_{h+1} for which $b_{h,j}$ has a positive effect on the classification and for which moving the profile b_h of $\pm\delta$ on j will degrade the classification



- $w_j = 0.2$ for $j = 1, ..., 5$
- $\lambda = 0.8$
- $a_5 \in A_{2\leftarrow DM}^{2\leftarrow Model}$

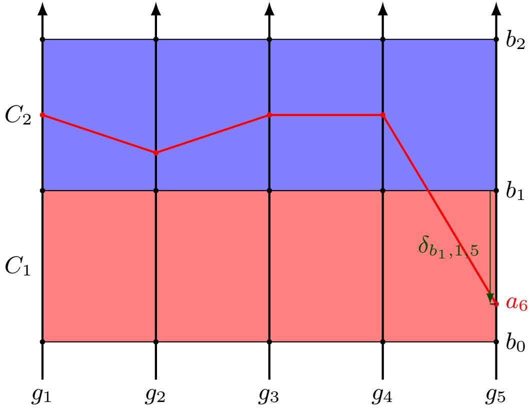
Metaheuristic to adjust the profiles

- $Q_{h,j}^{\pm\delta}$: Set of alternatives rightly classified into C_h or C_{h+1} for which $b_{h,j}$ has a positive effect on the classification and for which moving the profile b_h of $\pm\delta$ on j will degrade the classification



Metaheuristic to adjust the profiles

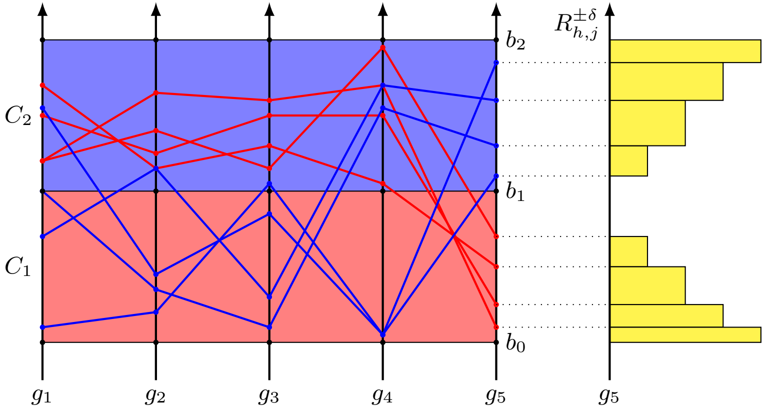
- $R_{h,j}^{\pm\delta}$: Set of alternatives classified into C_{h+1} instead of C_h or the contrary for which $b_{h,j}$ has a positive effect on the classification and for which moving the profile b_h of $\pm\delta$ on j will weaken the criteria coalition in favor of the correct classification



- $w_j = 0.2$ for $j = 1, ..., 5$
- $\lambda = 0.8$
- $a_6 \in A_{2 \leftarrow \text{DM}}^{1 \leftarrow \text{Model}}$

Metaheuristic to adjust the profiles

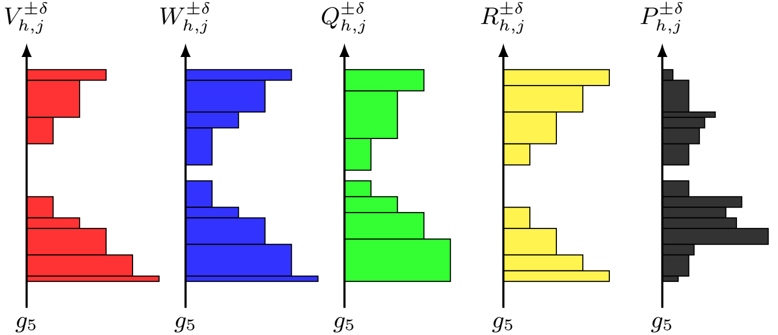
- $R^{\pm\delta}_{h,j}$: Set of alternatives classified into C_{h+1} instead of C_h or the contrary for which $b_{h,j}$ has a positive effect on the classification and for which moving the profile b_h of $\pm\delta$ on j will weaken the criteria coalition in favor of the correct classification



Metaheuristic to adjust the profiles

$$P(b_{1,j}^{\pm\delta}) = \frac{k_V|V_{h,j}^{\pm\delta}| + k_W|W_{h,j}^{\pm\delta}| + k_T|T_{h,j}^{\pm\delta}|}{d_V|V_{h,j}^{\pm\delta}| + d_W|W_{h,j}^{\pm\delta}| + d_T|T_{h,j}^{\pm\delta}| + d_Q|Q_{h,j}^{\pm\delta}| + d_R|R_{h,j}^{\pm\delta}|}$$

with : $k_V = 2, k_W = 1, k_T = 0.1, d_V = d_W = d_T = 1, d_Q = 5, d_R = 1$



Metaheuristic to adjust the profiles

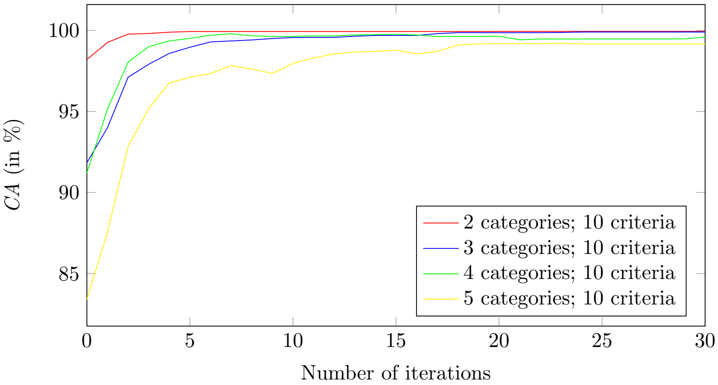
Overview of the complete algorithm

repeat
  for all profile **do**
    for all criterion (chosen randomly) **do**
      Choose profile's evaluation $b_{h,j}^{\pm L}$ which has the highest probability $P(b_{h,j}^{\pm L})$
      Draw a random number r in the interval $[0, 1]$
      if $P(b_{h,j}^{\pm L}) \geq r$ **then**
        Move the profile to the new value
      end if
    end for
  end for
until Stop condition is met

Experimentations

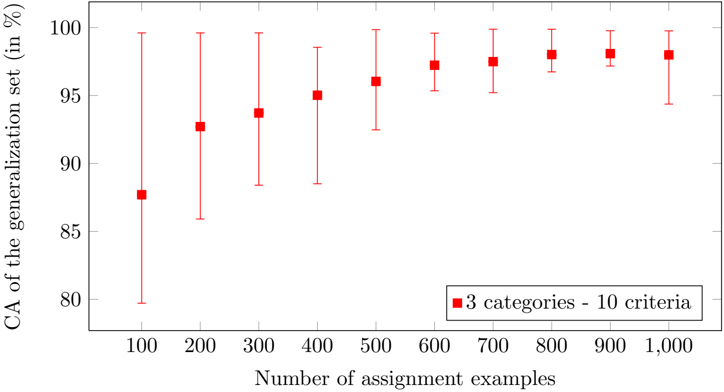
1. What's the efficiency of the algorithm ?
2. How much alternatives are required to learn a good model ?
3. What's the capability of the algorithm to restore assignment when there are errors in the examples ?
4. How the algorithm behaves on real datasets ?

Algorithm efficiency



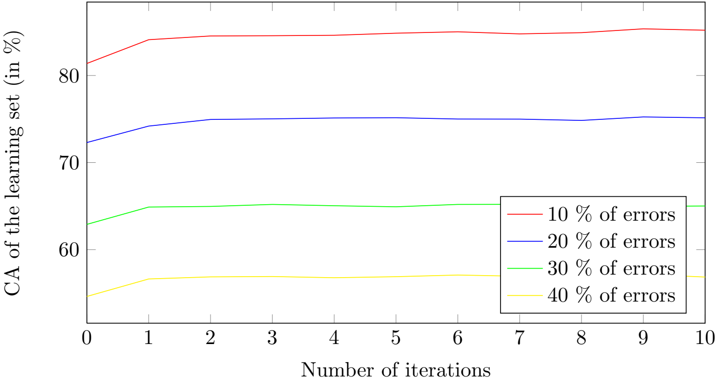
- Random model M generated
- Learning set : random alternatives assigned through the model M
- Model M' learned with the metaheuristic from the learning set

Model retrieval



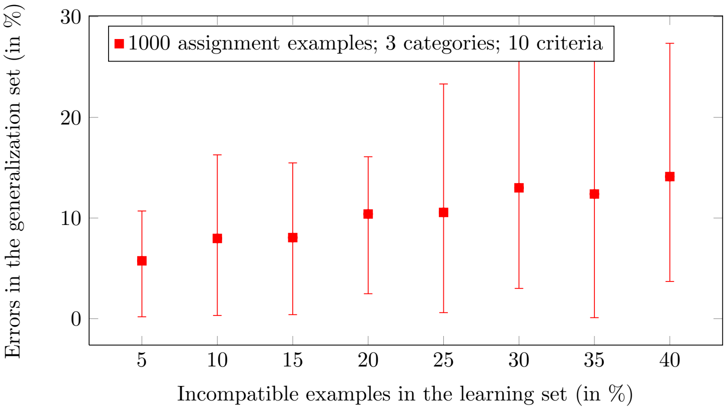
- Random model M generated
- Learning set : random alternatives assigned through model M
- Model M' learned with the metaheuristic from the learning set
- Generalization set : random alternatives assigned through M and M'

Tolerance for errors



- Random model M generated
- Learning set : random alternatives assigned through model M + errors
- Model M' learned with the metaheuristic from the learning set

Tolerance for errors



- Random model M generated
- Learning set : random alternatives assigned through model M + errors
- Model M' learned with the metaheuristic from the learning set
- Generalization set : random alternatives assigned through M and M'

Application on real datasets

Dataset	#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

- Instances split in two parts : learning and generalization sets
- Binarization of the categories

Application on real datasets - Binarized categories

Learning set	Dataset	MIP MR-SORT	META MR-SORT	LP UTADIS
20 %	DBS	0.9861 ± 0.0531	0.9586 ± 0.0410	0.9804 ± 0.0365
	CPU	0.9980 ± 0.0198	0.9883 ± 0.0200	1.0000 ± 0.0000
	BCC	0.8527 ± 0.0421	0.8060 ± 0.0559	0.7982 ± 0.0581
	MPG	0.8752 ± 0.0313	0.8564 ± 0.0406	0.8509 ± 0.0414
	ESL	0.9444 ± 0.0178	0.9345 ± 0.0213	0.9625 ± 0.0196
	MMG	0.8796 ± 0.0215	0.8704 ± 0.0232	0.8477 ± 0.0284
	ERA	0.8253 ± 0.0221	0.8218 ± 0.0211	0.7974 ± 0.0304
	LEV	0.8759 ± 0.0172	0.8690 ± 0.0220	0.8790 ± 0.0235
	CEV	-	0.9240 ± 0.0117	0.9230 ± 0.0123
50 %	DBS	0.9601 ± 0.0369	0.9381 ± 0.0276	0.9380 ± 0.0312
	CPU	0.9863 ± 0.0144	0.9755 ± 0.0157	1.0000 ± 0.0000
	BCC	-	0.7714 ± 0.0272	0.7590 ± 0.0246
	MPG	-	0.8357 ± 0.0269	0.8190 ± 0.0246
	ESL	0.9300 ± 0.0107	0.9241 ± 0.0116	0.9467 ± 0.0113
	MMG	-	0.8546 ± 0.0137	0.8395 ± 0.0155
	ERA	0.8157 ± 0.0106	0.8144 ± 0.0114	0.7841 ± 0.0200
	LEV	0.8668 ± 0.0100	0.8566 ± 0.0171	0.8604 ± 0.0137
	CEV	-	0.9232 ± 0.0067	0.9222 ± 0.0071
80 %	DBS	0.9464 ± 0.0162	0.9348 ± 0.0134	0.9206 ± 0.0170
	CPU	0.9797 ± 0.0123	0.9744 ± 0.0066	1.0000 ± 0.0000
	BCC	-	0.7672 ± 0.0170	0.7467 ± 0.0164
	MPG	-	0.8315 ± 0.0249	0.8124 ± 0.0132
	ESL	0.9231 ± 0.0058	0.9205 ± 0.0062	0.9436 ± 0.0068
	MMG	-	0.8486 ± 0.0079	0.8384 ± 0.0082
	ERA	0.8135 ± 0.0065	0.8097 ± 0.0067	0.7781 ± 0.0148
	LEV	0.8655 ± 0.0058	0.8466 ± 0.0270	0.8551 ± 0.0083
	CEV	-	0.9229 ± 0.0032	0.9226 ± 0.0034

Application on real datasets

	Dataset	MIP MR-SORT	META MR-SORT	LP UTADIS
20 %	CPU	0.7542 ± 0.0506	0.7443 ± 0.0559	0.8679 ± 0.0488
	ERA	-	0.5104 ± 0.0198	0.4856 ± 0.0169
	LEV	-	0.5528 ± 0.0274	0.5775 ± 0.0175
	CEV	-	0.7761 ± 0.0183	0.7719 ± 0.0153
50 %	CPU	-	0.8052 ± 0.0361	0.9340 ± 0.0266
	ERA	-	0.5216 ± 0.0180	0.4833 ± 0.0171
	LEV	-	0.5751 ± 0.0230	0.5889 ± 0.0158
	CEV	-	0.7833 ± 0.0180	0.7714 ± 0.0158
80 %	CPU	-	0.8055 ± 0.0560	0.9512 ± 0.0351
	ERA	-	0.5230 ± 0.0335	0.4824 ± 0.0332
	LEV	-	0.5750 ± 0.0344	0.5933 ± 0.0305
	CEV	-	0.7895 ± 0.0203	0.7717 ± 0.0259

Conclusion and further researches

- Comparison performances of UTADIS and MR-Sort
- Include vetoes in the algorithm
- Test it on other datasets

Thank you
for your attention !

References I

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

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

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

References II

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

Slowínski, R., Greco, S., and Matarazzo, B. (2002).
Axiomatization of utility, outranking and decision-rule preference models for multiple-criteria classification problems under partial inconsistency with the dominance principle.
Control and Cybernetics, 31(4) :1005–1035.

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