Learning preferences with multiple-criteria models

Olivier Sobrie

NNT : 2016SACLC057

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ DE MONS
ET DE L'UNIVERSITÉ PARIS-SACLAY

préparée à CentraleSupélec

ÉCOLE DOCTORALE N°573
Interfaces : approches interdisciplinaires / fondements, applications et innovation

Spécialité de doctorat : Informatique

par

M. Olivier SOBRIE

Learning preferences with multiple-criteria models

Thèse présentée et soutenue à Mons (Belgique), le 21 juin 2016

Membres du jury:

Prof. Yves De Smet	Université Libre de Bruxelles	*Rapporteur*
Prof. Philippe Fortemps	Université de Mons	*Directeur de thèse*
Prof. Nicolas Gillis	Université de Mons	*Examinateur*
Dr. Christophe Labreuche	Thales Research & Technology	*Rapporteur*
Prof. Saïd Mahmoudi	Université de Mons	*Président du jury*
Prof. Vincent Mousseau	Université Paris-Saclay	*Directeur de thèse*
Prof. Wassila Ouerdane	Université Paris-Saclay	*Examinateur*
Prof. Marc Pirlot	Université de Mons	*Co-Directeur de thèse*

université PARIS-SACLAY
ÉCOLE DOCTORALE
INTERFACES
Approches Interdisciplinaires : Fondements, Applications et Innovation

Titre : Apprentissage de préférences á l'aide de modèles multi-critères
Mots-clés : apprentissage, preferences, aide á la décision

Résumé :
L'aide multicritère à la décision (AMCD) vise à faciliter et améliorer la qualité du processus de prise de décision. Les méthodes d'AMCD permettent de traiter les problèmes de choix, rangement et classification. Ces méthodes impliquent généralement la construction d'un modèle. Déterminer les valeurs des paramètres de ces modèles n'est pas aisé. Les méthodes d'apprentissage indirectes permettent de simplifier cette tâche en apprenant les paramètres du modèle de décision à partir de jugements émis par un décideur tels que "l'alternative a est préférée à l'alternative b" ou "l'alternative a doit être classifiée dans la meilleure catégorie". Les informations données par le décideur sont généralement parcimonieuses. Le modèle d'AMCD est appris au cours d'un processus interactif entre le décideur et l'analyste. L'analyste aide le décideur à formuler et revoir ses jugements si nécessaire. Le processus s'arrête une fois qu'un modèle satisfaisant les préférences du décideur a été trouvé.
Le "preference learning" (PL) est un sous domaine du "machine learning" qui s'intéresse à l'apprentissage des préférences. Les algorithmes de ce domaine sont capables de traiter de grands jeux de données et sont validés au moyen de jeux de données artificiels et réels. Les jeux de données traités en PL sont généralement collectés de différentes sources et sont entachés de bruit. Contrairement à l'AMCD, il existe peu ou pas d'interaction avec l'utilisateur en PL. Le jeu de données fourni en entrée à l'algorithme est considéré comme un échantillon éventuellement bruité d'une "réalité" ou "vérité de terrain". Les algorithmes utilisés dans ce domaine ont des propriétés statistiques fortes leur permettant de s'affranchir du bruit dans ces jeux de données.
Dans cette thèse, nous développons des algorithmes d'apprentissage permettant d'apprendre les paramètres de modèles d'AMCD. Plus précisément, nous développons une métaheuristique afin d'apprendre les paramètres d'un modèle appelé MR-Sort ("majority rule sorting"). Cette métaheuristique est testée sur des jeux de données artificiels et réels utilisés dans le domaine du PL. Nous utilisons cet algorithme afin de traiter un problème concret dans le domaine médical. Ensuite nous modifions la métaheuristique afin d'apprendre les paramètres d'un modèle plus expressif appelé NCS ("non-compensatory sorting"). Finalement, nous développons un nouveau type de règle de veto pour les modèles MR-Sort et NCS qui permet de prendre les coalitions de critères en compte. La dernière partie de la thèse introduit les méthodes d'optimisation semi-définie positive (SDP) dans le contexte de l'aide multicritère à la décision. Précisément, nous utilisons l'optimisation SDP afin d'apprendre les paramètres d'un modèle de fonction de valeur additive.

Title: Learning preferences with multiple-criteria models
Keywords: learning, preferences, multiple-criteria decision analysis

Abstract:
Multiple-criteria decision analysis (MCDA) aims at providing support in order to make a decision. MCDA methods allow to handle choice, ranking and sorting problems. These methods usually involve the elicitation of models. Eliciting the parameters of these models is not trivial. Indirect elicitation methods simplify this task by learning the parameters of the decision model from preference statements issued by the decision maker (DM) such as "alternative a is preferred to alternative b" or "alternative a should be classified in the best category". The information provided by the decision maker are usually parsimonious. The MCDA model is learned through an interactive process between the DM and the decision analyst. The analyst helps the DM to modify and revise his/her statements if needed. The process ends once a model satisfying the preferences of the DM is found.
Preference learning (PL) is a subfield of machine learning which focuses on the elicitation of preferences. Algorithms in this subfield are able to deal with large data sets and are validated with artificial and real data sets. Data sets used in PL are usually collected from different sources and are subject to noise. Unlike in MCDA, there is little or no interaction with the user in PL. The input data set is considered as a noisy sample of a "ground truth". Algorithms used in this field have strong statistical properties that allow them to filter noise in the data sets.

In this thesis, we develop learning algorithms to infer the parameters of MCDA models. Precisely, we develop a metaheuristic designed for learning the parameters of a MCDA sorting model called majority rule sorting (MR-Sort) model. This metaheuristic is assessed with artificial and real data sets issued from the PL field. We use the algorithm to deal with a real application in the medical domain. Then we modify the metaheuristic to learn the parameters of a more expressive model called the non-compensatory sorting (NCS) model. After that, we develop a new type of veto rule for MR-Sort and NCS models which allows to take criteria coalitions into account. The last part of the thesis introduces semidefinite programming (SDP) in the context of multiple-criteria decision analysis. We use SDP to learn the parameters of an additive value function model.

Acknowledgments

Achieving this thesis wouldn't have been possible without the help of several persons. Now it's time to look backwards and to thank all these persons who contributed in one way or another in the accomplishment of this work.

First I want to thank my two advisors, Professors Marc Pirlot and Vincent Mousseau for giving me the opportunity to accomplish this thesis. Their support, encouragement and valued advice helped me a lot. It was a great pleasure to work with them and I'll keep great memories of our collaboration.

I am grateful to the members of the jury who accepted to participate in the defense of this thesis. I thank Professor Saïd Mahmoudi and Doctor Amine Lazouni for their collaboration in the elaboration of Chapter 4. I'm also grateful to Professor Nicolas Gillis for his great support in the elaboration of Chapter 7.

I would like to thank my colleagues and friends from Mons and Paris, more particularly Valérie, Arnaud, Jinyan, Massimo with whom it was always a pleasure to take part in conferences. I also thank the secretaries of Mons and Paris, respectively Laurence Wouters and Delphine Martin for their administrative support.

I thank all my friends who supported me in the achievement of this thesis. I'm also grateful to my family. I especially want to thank Virginie who read my thesis and helped to improve substantially my English. I also thank my godmother Michèle who corrected English errors and misspellings in Chapter 4.

Finally, I want to thank my girlfriend, Maria, who supported me during these 4 years. Without her help it would not have been possible to accomplish this work. I thank her for her patience and encouragements.

Si l'on sait exactement ce qu'on va faire, à quoi bon le faire?
— Pablo Picasso

Contents

Acronyms

AHP analytic hierarchy process.

ASA American society of anesthesiologists.

AUC area under the curve.

AVF additive value function.

AVF-Sort additive value function sorting.

BF Boolean function.

CA classification accuracy.

CR Choquistic regression.

DM decision maker.

DRSA dominance rough set approach.

DSA doctors specialized in anesthesia.

ECTS european credit transfer and accumulation system.

ELECTRE élimination et choix traduisant la réalité.

EWG European working group.

GRIP generalized regression with intensities of preference.

KNN k-nearest neighbor.

LP linear program.

M.H.DIS multi-group hierarchical discrimination method.

MAVT multi-attribute value theory.

MBF monotone Boolean function.

MCDA multiple-criteria decision analysis.

MIP mixed integer program.

ML machine learning.

MLP multilayer perceptron.

MR-Sort majority rule sorting.

MR-Sort-CV MR-Sort model with coalitional veto.

MSC minimal sufficient coalition.

NCS non-compensatory sorting.

ORCLASS ordinal classification.

PL preference learning.

PSD positive semidefinite.

RBF radial basis function.

ROC receiving operating characteristic.

ROR robust ordinal regression.

SC sufficient coalition.

SDP semidefinite programming.

SOS sum of squares.

SVM support vector machine.

TOMASO tool for ordinal multi-attribute sorting and ordering.

UTA utilités additives.

UTADIS utilités additives discriminantes.

List of notations

Λ	Majority threshold of the veto relation.
λ	Majority threshold of the concordance relation.
A	Set of alternatives.
$A^* = \{a^{(1)}, \ldots, a^{(m)}\}$	Learning set composed of m alternatives.
A^{*h}	Subset of alternatives of the learning set A^* assigned to category C^h.
A^h	Subset of alternatives of the set A assigned to category C^h.
A_l^{*h}	Subset of alternatives of the learning set A^* assigned to category C^h according to the decision maker but assigned to category C^l by the model.
$A_{>l}^{*<h}$	Subset of alternatives of the learning set A^* assigned to a category worse than C^h according to the decision maker but assigned to a category better than C^l by the model.
$A_{<l}^{*>h}$	Subset of alternatives of the learning set A^* assigned to a category better than C^h according to the decision maker but assigned to a category worse than C^l by the model.
a_j	Performance of the alternative a on the criterion j.
$\overline{a_j}$	Best performance on criterion j.
$\underline{a_j}$	Worst performance on criterion j.
b^h	Profile delimiting category C^{h-1} from C^h.
$\tilde{b}^h$	Profile characterizing category C^h.
b_j^h	Performance of the profile b^h on the criterion j.
$\tilde{b}_j^h$	Performance of the profile $\tilde{b}^h$ on the criterion j.

C Set of categories going from C^1 to C^p.

$C(a)$ Category associated to the alternative a in the data set.

$C_M(a)$ Category associated to the alternative a by the model M.

C^h Category C^h.

$C^{<h}$ Categories worse than C^h.

$C^{>h}$ Categories better than C^h.

$C^{\leq h}$ Categories worse than and including C^h.

$C^{\geq h}$ Categories better than and including C^h.

C_μ Choquet integral with regard to capacity μ.

$g_j : A \to X_j$ Value function of the criterion j.

$H = \{1, \ldots, p-1\}$ Indices of the profiles of in a sorting model using limiting profiles.

$I, J \subseteq N$ Subsets of criteria of N.

m Number of alternatives in the learning set A^*.

$m(J)$ Möbius index of the subset of criteria J.

$N = \{1, \ldots, n\}$ Set of criteria.

n Number of criteria.

$N_{a \geq b^h}$ Subset of criteria of N on which a has a better score than b^h.

p Number of categories.

$\mu(J)$ Capacity of the subset of criteria J.

$U(a)$ Score of the alternative a in an additive value function model.

U^h Threshold delimiting category C^{h-1} from category C^h in an additive value function sorting model.

u_j Marginal value function of criterion j.

$u_j(a_j)$ Marginal value of alternative a on criterion j.

v^h Veto profile associated to the profile b^h.

v_j^h Veto threshold value associated to the profile b^h on criterion j.

w_j Weight of criterion j.

$X = \prod_{j=1}^{n} X_j$ Cartesian product of criteria scales.

X_j Evalutations of the alternatives on criterion j.

z_j Veto weight of criterion j.

Chapter 1

Introduction

In operations research, multiple-criteria decision analysis (MCDA) aims at helping users confronted to decision problems by the use of formal models. In MCDA, the user is also called decision maker (DM). While MCDA methods do not aim to substitute the DM and make decision on his/her behalf, they allow to structure the decision problem and give an insight into the DM's preferences. Decision problems considered in MCDA are of different types. For instance, a municipality has to choose the location of a new sports facility. This problem can be formulated as a choice problem in which the best solution is to be selected. MCDA also allows to deal with ranking and sorting problems. A ranking problem consists in ordering a set alternatives from the worst to the best according to the DM's preferences. For instance, a jury has to order students based on their academic results. Sorting problems consist in assigning each alternative to a category selected among a set of predefined and ordered categories. For instance, a surgeon has to classify patients in categories relative to their health status. In this thesis, we focus mainly on sorting problems.

In MCDA, the DM is often supported by a decision analyst since his/her preferences are generally not clearly defined in his/her mind at the beginning of the decision process. The role of the decision analyst is to interact with the DM in order to help him/her to reveal his/her preferences. The preferences are then formalized in a model. MCDA models involve parameters whose values should be elicited. For a DM it is often difficult to directly elicit these parameters since this requires a clear understanding of the decision model. Usually the DM is more likely to provide preference judgments of the type "alternative a is preferred to b" or "alternative a is classified in the best category" than specifying values for preference parameters. To support the elicitation of the DM's preferences, MCDA techniques have been developed in order to infer the preference parameters from preference statements. For instance, in the context of sorting, the analyst may

ask the DM to sort a subset of alternatives. Then the analyst uses an algorithm that learns the parameters of the sorting model in order to best restore the assignment examples provided by the DM. Once the model has been learned, the analyst helps the DM to modify or refine his/her statements if needed. The process continues iteratively until the DM is satisfied with the model and its outcomes.

More specifically, we illustrate a MCDA decision problem with the following example. Consider a family who has to choose an accommodation for holidays in France. They use a web engine in order to get a list of possible accommodations in the region they plan to visit. They want to rank the list of 1000 accommodations returned by the search engine from the worst to the best. However the family doesn't want to consider each accommodation individually. The family requests support from an analyst. The first task of the analyst consists of identifying the criteria that are relevant for the family. In this example, it can be for instance the size of the accommodation, the distance to the sea, the price per night, etc. Then the decision analyst requests the family for a ranking of 20 accommodations selected among the 1000 returned by the search engine. Based on the ranking provided by the family, the analyst infers the parameters of a decision model thanks to a learning algorithm. This decision model is then used to rank all the remaining 980 accommodations. Taking into account the satisfaction of the family regarding the model and the resulting ranking, the analyst may ask the family for more information or to revise the provided statements. The process continues until the family is satisfied with the ranking provided by the model. Generally in MCDA, the number of preference statements available for learning the model is limited. In this example, the family provides a ranking for a small subset of alternatives. The use of a MCDA model aims at formalizing the preferences of the family in order to rank a larger set of accommodations. The objective is to maximize the satisfaction of the family with respect to the obtained ranking. This goal is achieved by selecting an appropriate objective function when inferring the parameters. For instance, a possible objective function may be to find a model minimizing the number of inversions in the ranking given as input. In order to make sure to satisfy the DM, it is important to interact with him/her.

Preference learning (PL) is a subfield of machine learning (ML) dedicated to problems involving preferences. Problems in PL are usually split in several categories: label ranking, instance ranking, object ranking, etc. Problems in this field typically involve large data sets. PL algorithms rely on strong statistical properties and there is generally no or limited interactions with the user.

As an example of PL problem, consider a database containing 1000 patients which have been evaluated on multiple attributes for preoperative assessment. For each patient, a doctor has stated whether or not the patient can be accepted for surgery based on his/her evaluations on the different attributes. This data set

is used as input of preference learning algorithms for inferring the parameters of a model that best matches the database. The model can then be used to predict whether a patient should be accepted for surgery. Each learning algorithm has an objective that is clearly defined, e.g. maximizing the classification accuracy (CA). The model learned by the algorithm is generally used as a blackbox and its performance is assessed by computing a loss function which has been defined beforehand, e.g. the CA. Unlike in MCDA, the outcome of the model is usually used as is since there are no or limited interactions with the user.

MCDA and PL share a common goal: representing the preferences of users. As shown in the examples, the paths to achieve this goal are different. Algorithms in PL are designed to work with large data sets and no or limited interactions with the user while in MCDA it is the opposite. In this thesis, we try to bridge the gap between MCDA and PL. Precisely, we try to take advantage of the validation techniques used in PL in order to develop an algorithm for learning a MCDA sorting model from large data sets. We then use this algorithm to handle a typical PL problem involving several hundreds of alternatives. In this context, we show the advantage of the MCDA sorting model that is easily interpretable with a set of compact and intuitive rules. Then we enrich the expressivity of the model by taking criteria interactions into account. Later, we create a new type of veto rule taking criteria interactions into account. Finally we bring new optimization techniques, namely semidefinite programming (SDP), to MCDA and PL.

Organization of this manuscript

This manuscript is organized as follows.

Chapter 2 – State of the art

We introduce the basic notions of MCDA and PL. We recall the aim of the methods in these fields and the type of problems that are treated in both fields. We detail the MCDA methods designed for sorting alternatives in ordered categories. We recall the existing MCDA learning algorithms used in the sorting context. The ones used in this thesis are detailed more in depth. In this chapter, we also describe some PL algorithms that are dedicated to the inference of sorting models. Finally, we analyze the main differences and similarities between MCDA and PL.

Chapter 3 – Learning a MR-Sort model from large data sets

In the third chapter, we describe a metaheuristic dedicated to learning the parameters of a MCDA sorting model called majority rule sorting (MR-Sort) model. The metaheuristic infers the parameters of a MR-Sort model based on a set of assignment examples given as input. The metaheuristic and its variants are detailed. We study the performance of the metaheuristic in terms of computing time, tolerance for errors, model retrieval and the capability of the model to restore the assignments obtained with another sorting procedure (idiosyncrasy). Finally the chapter closes with experimental results on benchmarks proposed in the PL literature.

Chapter 4 – Case study: preoperative patient classification

We use the metaheuristic described in the previous chapter to deal with a medical application. In a first step, the application consists of determining a score relative to the health of a patient before surgery. In a second step, this score is used together with other elements to decide whether a patient should be accepted or refused for surgery. We infer the parameters of MR-Sort models from a data set composed of 898 patients. We compare the results obtained with MR-Sort to the ones obtained with other MCDA and PL algorithms. Finally, we discuss the advantage of using interpretable models such as MR-Sort in such a type of application.

Chapter 5 – Learning the parameters of a NCS model

The MR-Sort model is able to deal with problems in which criteria do not interact. In this chapter, we use an extension of MR-Sort which takes criteria interactions into account. This model is called the non-compensatory sorting (NCS) model. We provide the formulations of a mixed integer program (MIP) and a metaheuristic that are designed for learning the parameters of such a model from a set of assignment examples. We test the metaheuristic on real data sets. Finally we try to evaluate the gain in terms of expressivity that NCS provides as compared to MR-Sort.

Chapter 6 – New veto rule for outranking models

In this chapter, we enrich the MR-Sort assignment rule with non-veto conditions. In MR-Sort, categories are separated by profiles ordered by dominance. Each profile is a vector of evaluation on the set of criteria involved in the decision problem. An alternative is assigned to a category if its performances are at least as good as the lower profile on a sufficiently large set of criteria and not at least

as good as the upper profile. Standard veto enables to increase the expressivity of the classification rule by adding exceptions when alternatives are too weak on a specific criterion. We propose a new veto rule based on coalitions of criteria which extends the flexibility of the veto.

Chapter 7 – UTA-poly and UTA-splines

This chapter is devoted to the study of additive value function (AVF) models. In such models, a numerical score is computed for each alternative. It is computed by additively aggregating the scores of the alternative on the different criteria involved in the decision problem. The score of an alternative on a criterion is determined by a value function which either increases or decreases monotonically as a function of the criterion value. Determining explicitly the shape of the value functions is not an easy task for a DM. That's why several methods have been developed in order to infer these functions from a set of preference statements given by the DM. In this chapter, we propose a new method to infer a set of additive functions from statements given by the DM. The inferred additive value functions are polynomials or splines for which the monotonicity is guaranteed. We discuss the advantage of our method and experiment it on both artificial and real data sets.

Chapter 2

State of the art

In this chapter we recall the definition of some important notions in multiple-criteria decision analysis (MCDA). We list the type of problems treated in MCDA as well as the different families of methodologies and the different approaches to elicit the parameters of MCDA models.

In the second section of the chapter, we detail in depth the MCDA sorting algorithms that are used in this thesis.

The third section gives an overview of other well-known MCDA sorting procedures.

In the fourth section, we recall some algorithms used to learn the parameters of MCDA sorting models on the basis of assignment examples. We detail in depth the algorithms that infer the parameters of models that are used in this thesis.

The fifth section of the chapter introduces Preference Learning (preference learning (PL)), a subfield of machine learning (ML). As for MCDA, we describe the type of problems treated in PL. After that we describe performance indices used in ML to evaluate the quality of a sorting.

In the sixth section, we detail some preference learning algorithms dedicated to sorting problems. We describe more in depth the ones that are used in the context of monotone data sets.

Finally, the last section of the chapter describes the links and differences between MCDA and PL.

2.1 Multiple-Criteria Decision Analysis

In this section we recall the main concepts in multiple-criteria decision analysis. We give the aim of this field of operational research and the type of problem treated within this field.

2.1.1 Aim of multiple-criteria decision analysis methodologies

MCDA methodologies aim at helping one or several persons confronted to a decision problem. In MCDA, the person confronted to a decision is called decision maker (DM). The purpose of MCDA methodologies is to provide models to the DM(s) in order to help him to get answers to his questions regarding the decision he/she has to take.

The construction of MCDA models is done by an analyst in conjunction with the DM through interactions called *decision process*. As mentioned by Roy and Bouyssou (1993), the dynamic of the decision process is composed of highlights which will contribute to the construction of the global decision. Taking part to a decision process as a MCDA analyst consists in helping the DM to define the decision problem by identifying variables having an influence on the decision, helping him to make compromises, etc. There is a strong link between the decision process and the decision itself. Often, the outcome of a decision analyst is a model which has been constructed along the decision process. This MCDA model is not neutral and therefore does not provide an universal truth but only a truth that is as compatible as possible with DM(s) preferences. The purpose is not to provide one truth to a decision problem but rather to give to the DM a model reflecting his preferences in which the influence of his possibly arbitrary judgments are reduced. Roy (1985, 1996) defines decision aid as follows:

Definition 1. *Decision aiding is the activity of the person who, through the use of explicit but not necessarily completely formalized models, helps obtain elements of responses to the questions posed by a stakeholder of a decision process. These elements work towards clarifying the decision and usually towards recommending, or simply favoring, a behavior that will increase the consistency between the evolution of the process and this stakeholder's objectives and value system.*

2.1.2 Problem setting

MCDA aims at answering several types of decision problems. Roy (1985, 1996) identified the following three types of decision problems.

Choice problem In choice problem, the DM has to make a choice between different alternatives evaluated on multiple attributes. MCDA helps the DM to choose among all the possibilities by extracting the best alternative(s) among all existing alternatives.

Example 1. *For the construction of a new airport, a mayor has to choose between five locations in his city. Of course, the DM, i.e. the mayor, wants to build the new airport at the best possible place. In this context, the role*

of the MCDA analyst is to provide a model returning the best locations to the DM.

Ranking problem In this type of problem, the DM wants to get a ranking of the alternatives from the best to the worst, with possibly equivalence between two or more alternatives. The outcome of a MCDA method can be either a partial or a complete ranking of the set of alternatives.

Example 2. *In order to award medals to the best three participants of a contest, a jury has to elaborate a classification of the participants to determine the contest winners. The ranking is based on the evaluation of the candidates on multiple attributes. By using a MCDA method, one can determine a complete ranking of participants such that it is possible to attribute the gold, silver and bronze medals to respectively the first, the second and the third participant to the contest.*

Sorting problem The problem of sorting consists in assigning alternatives to predefined categories. These categories are ordered from the worst to the best. The outcome of a MCDA sorting method is an assignment of the alternatives among the different classes.

Example 3. *A jury has to decide whether students have succeeded or failed their school year based on their evaluations in different courses. In this context, a MCDA model can help the jury to deliberate in order to have consistent decisions for all the students. The model can for instance assign each student to one of the following ordered classes: "accepted" or "refused".*

2.1.3 Notion of alternative and criterion

Criterion and alternative are two important notions in MCDA. We recall here their definitions.

In MCDA, a decision maker is often confronted to different solutions to a problem. These solutions are also called alternatives or actions. In the examples presented above, we already presented some instances of alternatives: possible locations for an airport, participants to a contest or students. Roy (1985, 1996) defines an action as follows:

Definition 2. *An action "a" is the representation of a possible contribution to the comprehensive decision that can be considered autonomously with respect to the decision process development state and that can serve as an application point for the decision aid. The application point is, then, sufficient to characterize "a".*

Sometimes, we also speak about fictive alternative to designate an alternative which does not actually exist but is used for reasoning purposes in the decision process. This type of alternative is useful in some MCDA methods.

In MCDA, the alternatives are evaluated on multiple attributes, called criteria. A criterion is a function that assigns an evaluation to each alternative involved in the decision problem. The coding of a criterion is often made such that the preference either increases or decreases as a function of the criterion value. The preference scale associated with a criterion should be monotone and each criterion should have a preference direction: the value of the criterion should be either minimized or maximized. For instance, in the context of choosing a car, disregarding all other criteria, a DM usually prefers a cheap car to an expensive one. It is the task of the MCDA analyst to elicit these criteria in order to use them in a model. Roy (1985) defines the notion of criterion as follows.

Definition 3. *A criterion is a function g_j assigning a value to each alternative of a decision problem such that it becomes possible for a DM to perform pairwise comparisons and express a preference in favor of one alternative based on this value.*

In this manuscript, we adopt the following notations. The Cartesian product of criteria scales is denoted by $X = \prod_{j=1}^{n} X_j$ in which X_j represents all the possible values of a criterion j. We denote by

$$g_j : A \to X_j$$

the function assigning a value to each alternative of the set A. We denote by A a finite set of alternatives for which a ranking or a sorting has to be determined. We denote by $a \in A$ an alternative a contained in the set A. We denote by $a_j = g_j(a)$ the performance of an alternative a on criterion j. The performance vector associated to each alternative a is denoted by $a = (a_1, \ldots, a_n)$;

2.1.4 Preference relation

There exist different types of preference relation between two alternatives. Roy and Bouyssou (1993) list four types of preferences:

1. Indifference: the two alternatives are considered equivalent;
2. Strict preference: one alternative is considered as strictly preferred to another one;
3. Weak preference: the set of arguments in favor of one alternative against another is not strong enough to say that the first alternative is preferred to the second;

4. Incomparability: two alternatives cannot be compared because their evaluations are very contrasted.

2.1.5 Families of methodologies

MCDA methods are generally classified in two families. The first family encompasses methods based on multi-attribute value theory (MAVT) (Keeney and Raiffa, 1976) while the second includes methods based on outranking relations (Roy, 1991). We give some details about these methods below.

Multi-attribute value theory methods

The aim of MAVT methods is to assign a score to each alternative and to compare the alternatives with each other or against a threshold based on their score. The score of an alternative is built by summing up its weighted scores on the set of criteria on which it is evaluated, possibly after transforming them. A simple example of MAVT method is the weighted sum. MAVT methods include all decision methods that have been developed based on the MAVT, for instance AHP (analytic hierarchy process), UTA (utilités additives), etc.

Outranking methods

This family of methods has been developed by Roy in order to address problems for which MAVT methods were not well adapted (Bouyssou, 2009). Roy (1968) developed the ELECTRE (élimination et choix traduisant la réalité) method to deal with choice problems. Roy and Bertier (1973) developed ELECTRE II which allows to deal with ranking problems. Later, Yu (1992); Roy and Bouyssou (1993) developed ELECTRE TRI which allows to deal with sorting problems.

In outranking methods, a preference relation, called outranking relation is built between pairs of alternatives evaluated on multiple criteria. An outranking relation as defined by Roy (1991) is a binary relation on the set of alternatives A denoted by $\succcurlyeq$. An alternative a outranks another one, b, i.e. $a \succcurlyeq b$, if there are strong enough arguments to declare that a is at least as good as b and if there is no essential reason to refute that statement. Outranking methods include methods like ELECTRE, PROMETHEE and RUBIS.

Choice of the multiple-criteria decision analysis method

When confronted to a decision problem, an analyst may wonder which MCDA method he/she should use in order to help the DM in the best possible way. The choice of the method depends on multiple factors including the nature of the decision problem, the knowledge of the DM, the properties of the MCDA

method, etc. This question is treated in papers such as for instance De Montis et al. (2005); Ouerdane (2011); Cinelli et al. (2014).

2.2 Two families of multiple-criteria decision analysis sorting methods

The focus of this thesis is on sorting problems. We consider only sorting problems in which the categories are predefined. Note that clustering addresses problems in which the categories are not not predefined (see e.g. De Smet et al., 2012; Meyer and Olteanu, 2013).

Several MCDA methods have been developed in order to deal with sorting problems. We recall here two MCDA methods dedicated to the sorting problems which are used in this thesis: the ELECTRE TRI model (and two of its variants) and the additive value function sorting model.

2.2.1 ELECTRE TRI

ELECTRE TRI is an outranking sorting procedure proposed by Yu (1992). The method aims at assigning each alternative of a set to a category selected among a set of pre-defined and ordered categories. Each alternative in the set is evaluated on a set of monotone criteria. The category in which an alternative is assigned to is chosen by comparing the alternative performances to the performances of profiles delimiting the categories.

Compared to other outranking methods, like ELECTRE II used in ranking problems, ELECTRE TRI does not require to compare every pair of alternatives. This has an importance in terms of computing time. Consider a problem involving 100 alternatives. If we want to rank these alternatives, we have to compare them in pairs. It involves 4950 ($\frac{100\times 99}{2}$) comparisons. To sort 100 alternatives in p categories with ELECTRE TRI, one has to compare these alternatives to $(p-1)$ profiles delimiting the p categories. It leads at most to $100\times(p-1)$ comparisons, i.e. $\frac{99}{2(p-1)}$ times less comparisons. For large data sets, using ELECTRE TRI saves a lot of computing time compared to ELECTRE II.

Another advantage of ELECTRE TRI is that it directly provides a sorting of the alternatives in ordered classes while the outcome of a ranking method as ELECTRE II is usually not a complete ranking of the alternatives. The outranking relation of ELECTRE II does not guarantee the transitivity and the completeness. ELECTRE TRI allows to obtain a complete ranking of the alternatives by assigning them in classes.

Formally we denote by A the set of alternatives that have to be assigned to a category selected among p ordered categories, $C^p \succ C^{p-1} \succ \ldots \succ C^1$. Alternatives in the set are evaluated on multiple attributes which are known to

have a monotone preference scale, i.e. the higher the value of an alternative on criterion j the better it is or the contrary. The set of criteria is denoted by $N = \{1, \ldots, n\}$ and j denotes one of the criteria in the set N. To know in which category an alternative should be assigned, its performances are compared to the ones of the $p-1$ profiles delimiting the p categories. We denote by b^h the profile delimiting the category C^h from C^{h+1}. $H = \{1, \ldots, p-1\}$ denotes the set of profile indices. Figure 2.1 illustrates the profiles and categories of an ELECTRE TRI model.

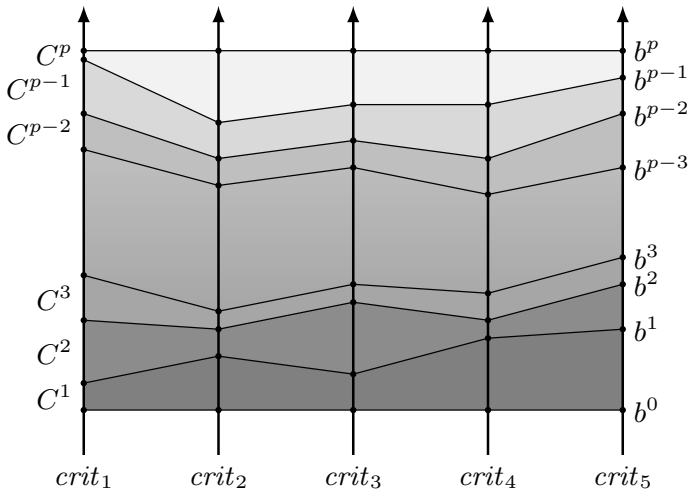


Figure 2.1: Criteria, profiles and categories of an ELECTRE Tri model.

Alternatives are compared to the profiles delimiting the categories. In ELECTRE TRI, an alternative a is considered better than or equal to a profile b^h if the credibility index of the assertion "a is at least as good as b^h", denoted by $\sigma(a, b^h)$, is greater than or equal to a cut threshold, λ. The credibility index is the aggregation of two indices, called concordance and non-discordance degrees.

The concordance degree values the strength of the criteria coalition supporting the assertion "a is at least as good as b^h", denoted by $a \succcurlyeq b^h$. This index corresponds to the aggregation of partial concordance indices computed for each criterion j, which represent the degree to which an alternative a is considered at least as good as the profile b^h on criterion j. The computation of partial concordance index, denoted by $\mathcal{C}_j(a, b^h)$ is done by comparing on criterion j, the alternative performance a_j to the profile performance, denoted by b^h_j. Two thresholds are involved in the computation of $\mathcal{C}_j(a, b^h)$:

- an indifference threshold, denoted by q^h_j, such that $q^h_j \geq 0$ which corresponds to the minimal difference between a_j and b^h_j from which the assertion $a \succcurlyeq b$ becomes completely true on criterion j;

- a preference threshold, denoted by p_j^h, such that $p_j^h \geq q_j^h$, which represents the minimal difference between a_j and b_j^h, from which the assertion $a \succcurlyeq b^h$ becomes partially true on criterion j.

Note that indifference and preference thresholds may vary depending on the criterion value. We consider here fixed thresholds in order to simplify the notations. Finally, the value of $\mathcal{C}_j(a, b^h)$ is obtained as follows:

$$\mathcal{C}_j(a, b^h) = \begin{cases} 0 & \text{if } a_j \leq b_j^h - p_j^h, \\ 1 & \text{if } a_j \geq b_j^h - q_j^h, \\ \frac{a_j - b_j^h + p_j^h}{p_j^h - q_j^h} & \text{otherwise.} \end{cases} \tag{2.1}$$

The concordance degree, denoted $\mathcal{C}(a, b^h)$ is finally computed by weighting and aggregating the partial concordance indices as follows:

$$\mathcal{C}(a, b^h) = \sum_{j=1}^{n} w_j \cdot \mathcal{C}_j(a, b^h). \tag{2.2}$$

For convenience, the criteria weights are usually normalized, i.e. $\sum_{j=1}^{n} w_j = 1$.

The discordance degree values the strength of the criteria coalition rejecting the assertion $a \succcurlyeq b^h$. The non-discordance degree is the complement of this value. We denote the non-discordance degree by $\mathcal{ND}(a, b^h)$. To compute the non-discordance degree, the performances of the alternative are compared to these of the profiles and a partial discordance index, $D_j(a, b^h)$ is computed for each criterion. The determination of $\mathcal{D}_j(a, b^h)$ involves two thresholds:

- the preference threshold, p_j^h;
- the veto threshold, v_j^h, which represents the maximal difference between b_j^h and a_j tolerated before rejecting the assertion $a \succcurlyeq b^h$.

The value of $\mathcal{D}_j(a, b^h)$ is computed as follows:

$$\mathcal{D}_j(a, b^h) = \begin{cases} 1 & \text{if } a_j \leq b_j^h - v_j^h, \\ 0 & \text{if } a_j \geq b_j^h - p_j^h, \\ \frac{b_j^h - p_j^h - a_j}{v_j^h - p_j^h} & \text{otherwise.} \end{cases}$$

The non-discordance index is computed by aggregating the partial discordance degrees and the concordance degree as follows:

$$\mathcal{ND}(a, b^h) = \prod_{j \in N : \mathcal{D}_j(a, b^h) > \mathcal{C}(a, b^h)} \frac{1 - \mathcal{D}_j(a, b^h)}{1 - \mathcal{C}(a, b^h)}. \tag{2.3}$$

The credibility degree $\sigma(a, b^h)$ of the assertion $a \succcurlyeq b^h$ is finally determined by multiplying the concordance degree by the non-discordance degree as given in the following equation.

$$\sigma(a, b^h) = \mathcal{C}(a, b^h) \cdot \mathcal{ND}(a, b^h). \tag{2.4}$$

Finally, an alternative a outranks a profile b^h if its credibility threshold $\sigma(a, b^h)$ is greater than a threshold value λ fixed usually in the interval $]0.5, 1]$. Formally, we have:

$$a \succcurlyeq b^h \iff \sigma(a, b^h) \geq \lambda. \tag{2.5}$$

We define two fictive profiles b^0 and b^p which correspond respectively to the worst and best possible possible performances on all the criteria. In ELECTRE TRI, alternatives are assigned to a category by using one of the two following assignment procedures.

Pessimistic procedure It consists in comparing each alternative a successively to $b^{p-1}, \ldots, b^1, b_0$. With l being the first index such that $a \succcurlyeq b^l$, a is assigned to C^{l+1}.

Optimistic procedure It consists in comparing each alternative a successively to $b^1, b^2, \ldots, b^p$. With l being the first index such that $b^l \succcurlyeq a$ and not $a \succcurlyeq b^l$, a is assigned to C^l.

2.2.2 Majority rule sorting model

In this subsection, we begin by presenting the drawbacks of ELECTRE TRI. Then we describe the majority rule sorting (MR-Sort) model, a simplified version of ELECTRE TRI.

Drawbacks of ELECTRE TRI

ELECTRE TRI presents some drawbacks from our point of view. In this paragraph we present some of them.

The first one is the number of parameters of the model. ELECTRE TRI requires to elicit n weights, $p-1$ profiles evaluated on the set of criteria, one majority threshold as well as at least n indifference, preference and veto thresholds. In total it makes $4np-3n+1$ parameters to be determined[1]. As stated previously, a DM often has difficulties to elicit directly the parameters of a MCDA model. It is all the more true that the number of parameters increases. In case of indirect

[1]If we consider a different threshold for each profile then it adds $3np$ parameters to be determined.

elicitation, the high number of parameters has an impact on the computing time of the solution.

ELECTRE TRI can be subject to small compensatory effects when the performances of an alternative a are such that a is not strictly preferred to a profile b^h on several criteria, i.e. the performance of a_j is such that $b_j^h - p_j^h < a_j < b_j^h - q_j^h$ on several criteria. A weak performance on one criterion can be compensated by a stronger one on another criterion. Indeed, consider a profile b^h and two alternatives, a and b, having the same credibility index with respect to profile b^h. Assume that the performances of a and b are equal on all the criteria of the model, except on criteria 1 and 2. On both criteria the performances of a and b are located in the interval $[b^h - p_j^h, b^h - q_j^h]$. Since the credibility indices of a and b are equal and their performances are equal on all the criteria $j \in N \backslash \{1, 2\}$, the contributions of partial concordance indices on criteria 1 and 2 to the global concordance index are equal. Therefore the following equality holds:

$$w_1 \cdot \mathcal{C}_1(a, b^h) + w_2 \cdot \mathcal{C}_2(a, b^h) = w_1 \cdot \mathcal{C}_1(b, b^h) + w_2 \cdot \mathcal{C}_2(b, b^h).$$

The ratio $\frac{w_2}{w_1}$ can be expressed as follows:

$$\frac{w_2}{w_1} = \frac{\mathcal{C}_1(b, b^h) - \mathcal{C}_1(a, b^h)}{\mathcal{C}_2(a, b^h) - \mathcal{C}_2(b, b^h)}. \tag{2.6}$$

The partial concordance indices of $a^{(1)}$ and $a^{(2)}$ on criteria 1 and 2 are equal to:

$$\mathcal{C}_1(a, b^h) = \frac{a_1 - b_1^h + p_1^h}{p_1^h - q_1^h},$$
$$\mathcal{C}_2(a, b^h) = \frac{a_2 - b_2^h + p_2^h}{p_2^h - q_2^h},$$
$$\mathcal{C}_1(b, b^h) = \frac{b_1 - b_1^h + p_1^h}{p_1^h - q_1^h},$$
$$\mathcal{C}_2(b, b^h) = \frac{b_2 - b_2^h + p_2^h}{p_2^h - q_2^h}.$$

By substituting these values in Equation (2.6), we obtain:

$$\frac{w_2}{w_1} = -\frac{p_2^h - q_2^h}{p_1^h - q_1^h} \cdot \frac{a_1 - b_1}{a_2 - b_2}.$$

In this Equation, we observe that a difference of $a_1 - b_1$ on criterion 1 is compensated by a difference of $a_2 - b_2$ on criterion 2. As ELECTRE methods are well-known for their ability to reduce compensatory effects, it is difficult to legitimate assignments obtained with ELECTRE TRI if compensatory effects occur between criteria.

We also remark that the concordance degree evolves as a linear function when the difference between the performance of the alternative a and the profile b^h is in the interval $[q_j^h,\ p_j^h]$ on a criterion j. The same remark holds for the partial discordance index when the difference between a and b^h on j is in the interval $[q_j^h, p_j^h]$. This linearity is debatable. Indeed the preference function could be concave or convex instead of linear in the interval $[q_j^h, p_j^h]$.

In ELECTRE TRI, the discordance index leads to a non-linearity of the credibility index. The non-linearity of this index increases the complexity of the model and makes it more difficult to explain. In MCDA, parameters of the model are used to legitimate the output of a method. Due to the complexity of the credibility index, it is difficult to explain the output of the method to a DM. Moreover, this index needs to be linearized when using standard linear programming tools in order to infer the parameters of the method (e.g. Mousseau and Dias, 2003).

Majority rule sorting model

Bouyssou and Marchant (2007a,b) presented an axiomatic characterization of the so called non-compensatory sorting (NCS) models. Their papers propose strong theoretical justifications for such a type of model. A link between this type of model and the pessimistic version of ELECTRE TRI is outlined according to Bouyssou and Marchant (2007a,b). The outranking rule can be formulated as follows: An alternative a *outranks* a profile b^h if the following two conditions are satisfied:

1. a has better performances than b^h on a sufficiently large coalition of criteria;

2. a is not significantly worse than b^h on any criterion $j \in N$.

MR-Sort is a further simplification of the NCS model. With MR-Sort we assume that "sufficiently large coalitions of criteria" can be determined by a weight w_j associated with each criterion and a majority threshold λ. A coalition of criteria J is "sufficiently large" if $\sum_{j\in J} w_j \geq \lambda$. Formally, the outranking relation of MR-Sort (2.5) reads:

$$a \succcurlyeq b^h \iff \sum_{j:a_j \geq b_j^h} w_j \geq \lambda \text{ and } \nexists j \in N : a_j < b_j^h - v_j^h. \tag{2.7}$$

The axioms given in Bouyssou and Marchant (2007a,b) are described for the pessimistic assignment procedure. Therefore, we use MR-Sort exclusively with this assignment procedure. Using Equation (2.7) as outranking rule, the pes-

simistic assignment rule reads:

$$a \in C^h \iff \left[\sum_{j:a_j \geq b_j^{h-1}} w_j \geq \lambda \text{ and } \nexists j \in N : a_j < b_j^{h-1} - v_j^{h-1} \right] \text{ and } \left[\sum_{j:a_j \geq b_j^h} w_j < \lambda \text{ or } \exists j \in N : a_j < b_j^h - v_j^h \right]. \quad (2.8)$$

In this thesis, we mainly use MR-Sort without veto, therefore the assignment rule given in Equation (2.8) can be simplified as follows:

$$a \in C^h \iff \sum_{j:a_j \geq b_j^{h-1}} w_j \geq \lambda \text{ and } \sum_{j:a_j \geq b_j^h} w_j < \lambda. \quad (2.9)$$

Compared with ELECTRE TRI, MR-Sort is not subject to compensatory effects, which better complies with the aim of outranking methods. Moreover MR-Sort assignment rules are easier to explain to a DM. Indeed the method involves less parameters and the outranking rule does not involve indifference and preference thresholds.

Such a model contrasts with additive value functions models such as UTADIS (Jacquet-Lagrèze and Siskos, 1982; Doumpos and Zopounidis, 2002). It belongs to a class of decision models referred to as non-compensatory in the literature (Fishburn, 1976; Bouyssou, 1986), because it just takes into account whether or not an evaluation is above the profile value, not by how much it passes or misses this profile value. These methods are well suited to criteria assessed on ordinal scales.

2.2.3 Non-compensatory sorting model

The MR-Sort model is a particular case of the definition of non-compensatory sorting models axiomatized by Bouyssou and Marchant (2007a,b). MR-Sort is only able to deal with decision problems in which there is no interaction between criteria. The NCS model proposed by Bouyssou and Marchant (2007a,b) is more general and can handle criteria interactions. In this model, the strength of a coalition of criteria is not just the sum of the weights associated to individual criteria. We first show the limitations of MR-Sort on a fictive decision problem. Then we describe a mathematical formulation of the model.

Limitation of MR-Sort

Before describing the NCS model, we show the limits of MR-Sort on an example.

Table 2.1: Evaluation of students and their acceptance/refusal status.

	Math	Physics	Chemistry	History	A/R
James	11	11	9	9	A
Robert	9	9	11	11	A
John	11	9	9	11	R
Pierre	9	11	11	9	R

Example 4. *Consider an application in which a committee for a higher education program has to decide the admission of students based on their evaluations in 4 courses: math, physics, chemistry and history. To be accepted in the program, the committee considers that a student should have a sufficient majority of evaluations above 10/20. From the committee point of view, courses (criteria) coalitions don't have the same importance. The strength of a coalition of courses varies as a function of the courses belonging to the coalition. The committee stated that the following subsets are the minimal coalitions of courses in which the evaluation should be above 10/20 in order to be accepted: {math, physics}, {math, chemistry} and {chemistry, history}. To illustrate this rule, Table 2.1 shows evaluations of several students and, for each student, whether he/she is accepted or refused.*

Representing the assignments given in Table 2.1 by using a MR-Sort model with profiles fixed at 10/20 in each course is impossible. There are no additive weights allowing to model such rules. On the one hand, the acceptance of James and Robert lead to the following constraints:

$$\begin{cases} w_1 + w_2 & \geq \lambda, \\ w_3 + w_4 & \geq \lambda. \end{cases}$$

By summing these inequalities, we have:

$$w_1 + w_2 + w_3 + w_4 \geq 2\lambda. \tag{2.10}$$

On the other hand, the refusal of John and Pierre lead to the following constraints:

$$\begin{cases} w_1 + w_4 & < \lambda, \\ w_2 + w_3 & < \lambda. \end{cases}$$

After summing these inequalities, we obtain:

$$w_1 + w_2 + w_3 + w_4 < 2\lambda. \tag{2.11}$$

Constraints (2.10) *and* (2.11) *can't be satisfied both at the same time. It is therefore not possible to use simple additive weights to model these assignment rules.*

The example given above showed that MR-Sort is not adapted to handle such type of problems since it does not allow to model attribute interactions. It limits the capacity of the model to represent some situations.

Notion of capacity

The model described hereafter uses capacities. Therefore we introduce the definition of a capacity.

Definition 4. *A capacity is a function* $\mu : 2^N \to [0,1]$ *such that:*

- $\mu(K) \geq \mu(J)$*, for all* $J \subseteq K \subseteq N$ *(monotonicity) ;*
- $\mu(\emptyset) = 0$ *and* $\mu(N) = 1$ *(normalization).*

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

$$\mu(J) = \sum_{K \subseteq J} m(K) \quad \forall J \subseteq N \quad \text{with } m(K) = \sum_{L \subseteq K} (-1)^{|K|-|L|} \mu(L). \tag{2.12}$$

The value $m(K)$ can be interpreted as the weight that is exclusively allocated to K as a whole. A capacity can be defined directly by its Möbius transform also called Möbius interaction. A Möbius interaction or Möbius mass m is a set function $m : 2^N \to [-1,1]$ satisfying the following conditions:

$$\sum_{K \subseteq J \cup \{j\}} m(K) \geq 0 \quad \forall j \in N, J \subseteq N \backslash \{j\} \quad \text{and} \sum_{K \subseteq N} m(K) = 1. \tag{2.13}$$

If m is a Möbius interaction, the set function defined by $\mu(J) = \sum_{K \subseteq J} m(K)$ is a capacity. Conditions (2.13) guarantee that μ is monotone (Chateauneuf and Jaffray, 1986).

Non-compensatory sorting model

Using a capacity to express the strength of a criteria coalition in favor of an object, we transform the outranking rule (2.7) as follows:

$$a \succcurlyeq b^h \Leftrightarrow \mu(J) \geq \lambda \text{ and } \nexists j \in N : a_j < b_j^h - v_j^h \tag{2.14}$$

with $J = \{j \in N : a_j \geq b_j^h\}$ and $\mu(J) = \sum_{K \subseteq J} m(K)$. Computing the value of $\mu(J)$ with the Möbius transform requires the evaluation of $2^{|J|}$ parameters. In a model involving n criteria, it involves the elicitation of 2^n parameters, with $\mu(\emptyset) = 0$ and $\mu(N) = 1$. To reduce the number of parameters to be elicited, we generally use 2-additive capacities 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.

In this thesis, we use the NCS model without veto. Equation (2.14) can then be simplified as follows:

$$a \succcurlyeq b^h \Leftrightarrow \mu(J) \geq \lambda. \tag{2.15}$$

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

$$\mu(N_{a \geq b^{h-1}}) \geq \lambda \quad \text{and} \quad \mu(N_{a \geq b^h}) < \lambda \tag{2.16}$$

with $N_{a \geq b^{h-1}} = \{j \in N : a_j \geq b_j^{h-1}\}$ and $N_{a \geq b^h} = \{j \in N : a_j \geq b_j^h\}$.

This model fits with the definition of a NCS model given in Bouyssou and Marchant (2007a,b). We note that MR-Sort is a special case of a NCS model in which a simple additive capacity is used.

2.2.4 Additive value function sorting model

Additive value function sorting (AVF-Sort) models belong to the family of MAVT methods. In such type of model, a numeric score is attributed to each alternative. Additive value function models are based on the theory of multi-attribute value functions (Keeney and Raiffa, 1976).

In an additive value function model, a value function u_j is associated to each criterion $j \in N$ of the decision problem. Each value function aims at modeling the preference of the decision maker on one criterion. These value functions, called marginals, are monotone, i.e. the slope of the curve remains either positive or negative on the criterion domain. The slope is positive if the preference increases when the criterion value increases and negative if the preference decreases when the criterion value decreases. An example of a marginal is shown in Figure 2.2. The marginal value associated to an alternative a on a criterion j is denoted by $u_j(a_j)$. We denote by $\underline{a_j}$ (resp. $\overline{a_j}$) the worst (resp. best) performance on criterion j. One may assume that $0 \leq u_j(a_j) \leq 1$ and $u_j(\underline{a_j}) = 0$, $u_j(\overline{a_j}) = 1$.

In order to obtain a global value reflecting the score of an alternative a, marginal values, $u_j(a_j)$, are summed up into a global value $U(a)$:

$$U(a) = \sum_{j=1}^{n} w_j \cdot u_j(a_j). \tag{2.17}$$

Each marginal value function is given a weight w_j which represents the importance of the criterion from the DM's point of view. Weights are determined such

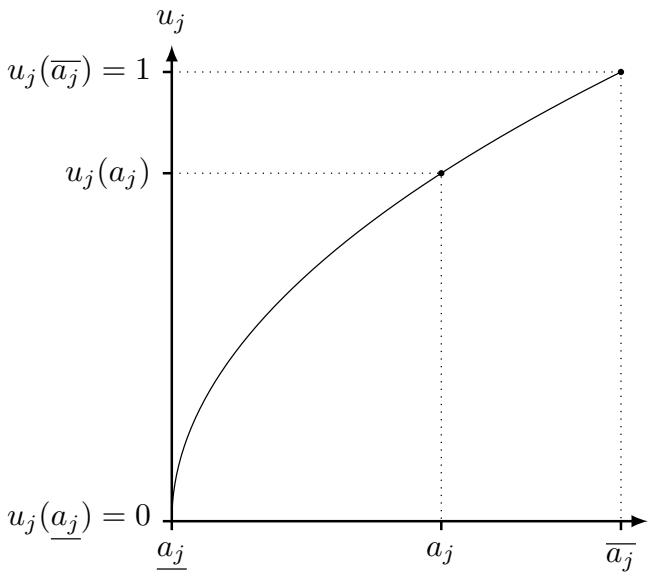


Figure 2.2: Example of marginal value function for criterion j.

that $\sum_{j=1}^{n} w_j = 1$. To simplify the notation, we directly integrate the weights in the marginal value functions as follows:

$$u_j^*(a_j) = w_j \cdot u_j(a_j).$$

Hence, Equation (2.17) can then be reformulated as follows:

$$U(a) = \sum_{j=1}^{n} u_j^*(a_j). \tag{2.18}$$

A particular but often used shape for marginal value functions is to assume they are piecewise linear. Such functions can approximate any shape of marginal value function. The marginal value functions are often defined as piecewise linear functions. Breakpoints of the value function u_j are placed on the criterion domain $[\underline{a_j}, \overline{a_j}]$ such that it is split in η_j equal parts, with $g_j^0 = \underline{a_j}$, $g_j^{\eta_j} = \overline{a_j}$ and $g_j^l = g_j^0 + \frac{l}{n_j} \cdot (g_j^{\eta_j} - g_j^0)$. Utility functions are constructed such that the worst performance on criterion j has a value equal to 0, $u_j(g_j^0) = 0$, and the best one has a value equal to 1, $u_j(g_j^{n_j}) = 1$. This is for the case of increasing marginals. The case of marginals decreasing with the value of the criterion can be described symmetrically.

When using piecewise linear functions, the marginal value $u_j(a_j)$ of an alternative a can be expressed as follows (for a criterion to be maximized). Let

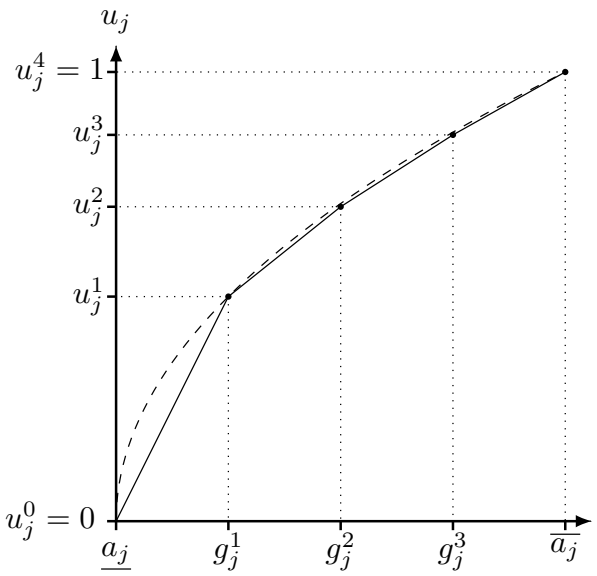


Figure 2.3: Approximation of a marginal value function for a criterion j (to be maximized) through a piecewise linear function.

g_j^L, with $l = \{1, \ldots, n_j\}$, be the first breakpoint for an alternative a such that $a_j \leq g_j^L$, the value of $u_j(a_j)$ is given by:

$$u_j(a_j) = u_j(g_j^{L-1}) + \left(\frac{a_j - g_j^{L-1}}{g_j^L - g_j^{L-1}} \right) \left(u_j(g_j^L) - u_j(g_j^{L-1}) \right). \tag{2.19}$$

In such a model, it is easy to compare alternatives based on their value $U(a)$. One can say that an alternative a is preferred to another b if the value of a, $U(a)$, is greater than the value of b, $U(b)$. Similarly an alternative a is indifferent to another b if the value of a is equal to the value of b. Formally it reads:

$$a \succ b \iff U(a) > U(b), \text{and}$$
$$a \sim b \iff U(a) = U(b).$$

These comparisons are used to establish a ranking of the alternatives.

In the sorting context, additive value function values are compared to thresholds delimiting the ordered categories. For a model involving p categories, $p + 1$ thresholds are required to fix the boundaries of the categories. We denote them by U^h, with $h = \{0, \ldots, p\}$. We have $U^0 = 0$ and $U^p = 1 + \epsilon$, with ϵ a small constant strictly greater than 0. Each category C^h is delimited by a lower and an upper threshold, respectively U^{h-1} and U^h. Using this model, an alternative

a is assigned to a category C^h if the following two conditions are fulfilled:

$$a \in C^h \iff \begin{cases} U(a) \geq U^{h-1}, \text{and} & \text{(2.20a)} \\ U(a) < U^h. & \text{(2.20b)} \end{cases}$$

2.3 Other multiple-criteria decision analysis sorting methods

In this section, we give an overview of other MCDA sorting methods. A complete review of MCDA sorting methods has been done in Zopounidis and Doumpos (2002).

2.3.1 Trichotomic segmentation

Trichotomic segmentation procedure was introduced by Moscarola and Roy (1977). The aim of the method is to sort each alternative of a set in one of the three following categories: "accepted", "rejected" or "need additional examination". We denote these categories respectively by C^A, C^R and $C^?$.

Each action is evaluated on n criteria by q experts. Each expert k is invited to evaluate the importance of the n criteria by assigning a weight w_j^k to each criterion. Then he/she has to define two profiles evaluated on n criteria. The first one $\mathcal{A}$ discriminates acceptable projects from others. A second one $\mathcal{R}$ discriminates rejected projects from others. The profiles defined by the experts are denoted respectively by $b^{\mathcal{A},k}$ and $b^{\mathcal{R},k}$. Each profile is a vector of performances on the n criteria. The value of profile $b^{\mathcal{A},k}$ (resp. $b^{\mathcal{R},k}$) on criterion j is denoted by $b_j^{\mathcal{A},k}$ (resp. $b_j^{\mathcal{R},k}$). Each expert k is asked to assess the value of a veto threshold $v_j^{h,k}$ on each criterion j with respect to each profile $b^{h,k}$, $h \in \{\mathcal{A}, \mathcal{R}\}$.

As in ELECTRE methods, concordance and non-discordance indices have to be calculated for each alternative and each profile. These indices have the same formulation as in ELECTRE TRI (Equations (2.2) and (2.3)). They only differ in the definition of the partial concordance and discordance indices. Formally the concordance index of an alternative a against a profile $b^{h,k}$, $h \in \{\mathcal{A}, \mathcal{R}\}$, for the expert k reads:

$$\mathcal{C}^k(a, b^{h,k}) = \sum_{j=1}^{n} w_j^k \cdot \mathcal{C}_j^k(a, b^{h,k}) \quad \text{with} \quad \mathcal{C}_j^k(a, b^{h,k}) = \begin{cases} 1 & \text{if } a_j \geq b_j^{h,k}, \\ 0 & \text{if } a_j < b_j^{h,k}, \end{cases}$$

and the non-discordance index reads:

$$\mathcal{ND}^k(a, b^{h,k}) = \prod_{j \in N : \mathcal{D}_j^k(a, b^{h,k}) > \mathcal{C}^k(a, b^{h,k})} \frac{1 - \mathcal{D}_j^k(a, b^{h,k})}{1 - \mathcal{C}^k(a, b^{h,k})},$$

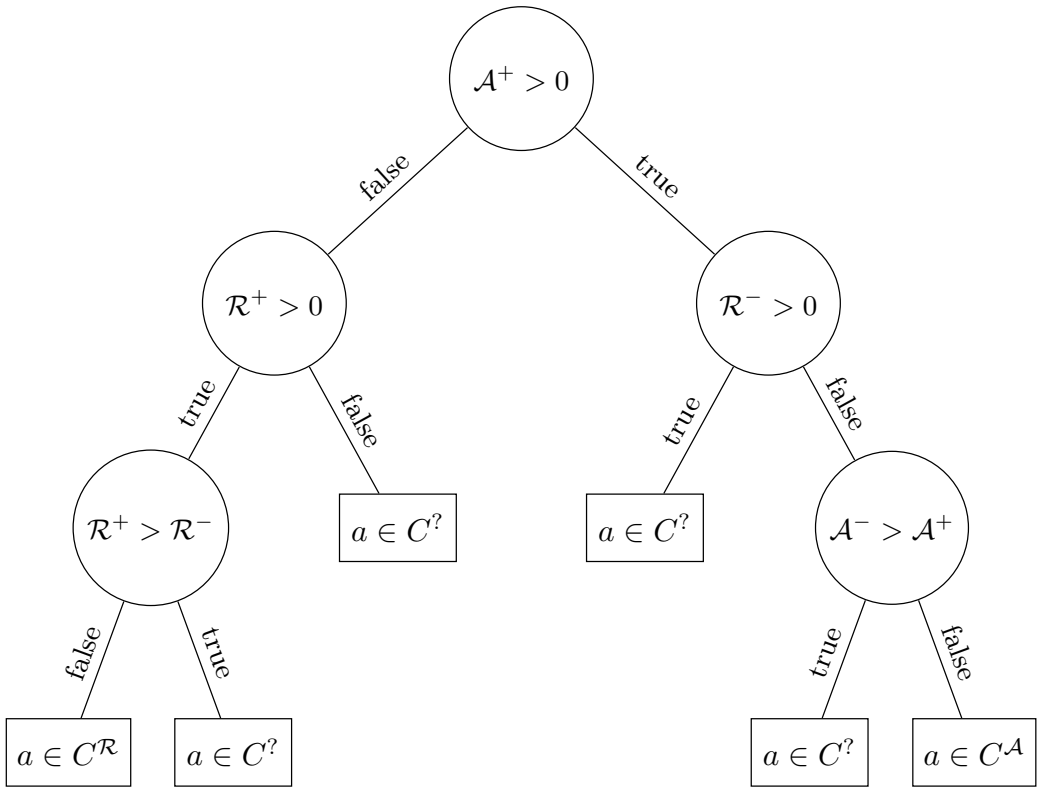


Figure 2.4: Trichotomic decision tree.

with

$$\mathcal{D}_j^k(a, b^{h,k}) = \begin{cases} 1 & \text{if } a_j \leq b_j^{h,k} - v_j, \\ 0.5 & \text{if } a_j = b_j^{h,k} - v_j, \\ 0 & \text{if } a_j \geq b_j^{h,k} - v_j. \end{cases}$$

Finally a credibility index of the relation $a \succcurlyeq b^{h,k}$ is computed as in ELECTRE TRI:

$$\sigma^k(a, b^{h,k}) = \mathcal{C}(a, b^{h,k}) \cdot \mathcal{ND}(a, b^{h,k})$$

The value $\sigma^k(b^{h,k}, a)$ can be computed similarly by permuting a and $b^{h,k}$ in the equations above. For each expert k, four indices are computed: $\sigma^k(a, b^{\mathcal{A},k})$, $\sigma^k(b^{\mathcal{A},k}, a)$, $\sigma^k(a, b^{\mathcal{R},k})$ and $\sigma^k(b^{\mathcal{R},k}, a)$.

Unlike in classical ELECTRE methods, this procedure considers several experts. To determine the assignment of an alternative a to a category, the decision tree given in Figure 2.4 is used. In the tree, $\mathcal{A}^+$, $\mathcal{A}^-$, $\mathcal{R}^+$ and $\mathcal{R}^-$ represent respectively the number of experts for which the indices $\sigma^k(a, b^{\mathcal{A},k})$ $\sigma^k(b^{\mathcal{A},k}, a)$,

$\sigma^k(a, b^{\mathcal{R},k})$ and $\sigma^k(b^{\mathcal{R},k}, a)$ exceed a pre-defined threshold λ. An alternative a is assigned to category $C^{\mathcal{A}}$ if $\mathcal{A}^+ > 0$, $\mathcal{A}^+ \geq \mathcal{A}^-$ and $\mathcal{R}^- = 0$. a is assigned to category $C^{\mathcal{R}}$ if $\mathcal{A}^+ = 0$, $\mathcal{R}^+ > 0$ and $\mathcal{R}^- > \mathcal{R}^+$. In all other cases a is assigned to $C^?$.

2.3.2 nTOMIC

Massaglia and Ostanello (1991) developed an interactive system called nTOMIC which is designed to support all stages of a segmentation process.

The method consists in computing two marginal indices for each alternative on each criterion. These marginals are called respectively "goodness" and "badness" credibility index. We denote them $\mathcal{G}_j(a, b^+)$ and $\mathcal{B}_j(a, b^-)$. Building these marginals involves the determination of two levels on each criterion: b_j^+ and b_j^-. The value b_j^+ corresponds to what is considered as a "good" score while b_j^- corresponds to what is considered as a "bad" score. An indifference threshold q_j^+ (resp. q_j^-) and a discrimination threshold s_j^+ (resp. s_j^-) are associated with each level b_j^+ (resp. b_j^-). These thresholds are chosen such that $0 \leq q_j^+ \leq s_j^+$, $0 \leq q_j^- \leq s_j^-$ and $b_j^+ + s_j^+ \leq b_j^- - s_j^-$. The performance of an alternative a_j is qualified as "certainly good", "not good" or "fairly good" respectively if $a_j \geq b_j^+ - q_j^+$, $a_j \leq b_j^+ - s_j^+$ or $b_j^+ - q_j^+ > a_j > b_j^+ - s_j^+$. Similarly, a performance a_j is qualified as "certainly bad", "not bad" or "fairly bad" respectively if $a_j \leq b_j^- + q_j^-$, $a_j \geq b_j^- + s_j^-$ or $b_j^- + q_j^- < a_j < b_j^- + s_j^-$. The two marginal indices are then computed similarly to the concordance and discordance indices in ELECTRE TRI. Formally, it reads:

$$\mathcal{G}_j(a, b_j^+) = \begin{cases} 0 & \text{if } a_j \leq b_j^+ - s_j^+, \\ 1 & \text{if } a_j \geq b_j^+ - q_j^+, \\ \frac{b_j^+ - s_j^+ - a_j}{s_j^+ - q_j^+} & \text{otherwise,} \end{cases}$$

and

$$\mathcal{B}_j(a, b_j^-) = \begin{cases} 0 & \text{if } a_j \geq b_j^- + s_j^-, \\ 1 & \text{if } a_j \leq b_j^- + q_j^-, \\ \frac{a_j - b_j^- + s_j^-}{s_j^- - q_j^-} & \text{otherwise.} \end{cases}$$

The marginal indices are then aggregated in two credibility indices: a credibility of "goodness", denoted by $\mathcal{G}(a, b_j^+)$ and a credibility of "badness", denoted by $\mathcal{B}(a, b_j^-)$. It is done as follows. A weight w_j is associated with each criterion j. Then "goodness" and "badness" credibility indices are defined as follows:

$$\mathcal{G}(a, b^+) = \sum_{j=1}^{n} w_j \cdot \mathcal{G}_j(a, b_j^+) \qquad \text{and} \qquad \mathcal{B}(a, b^-) = \sum_{j=1}^{n} w_j \cdot \mathcal{B}_j(a, b_j^+).$$

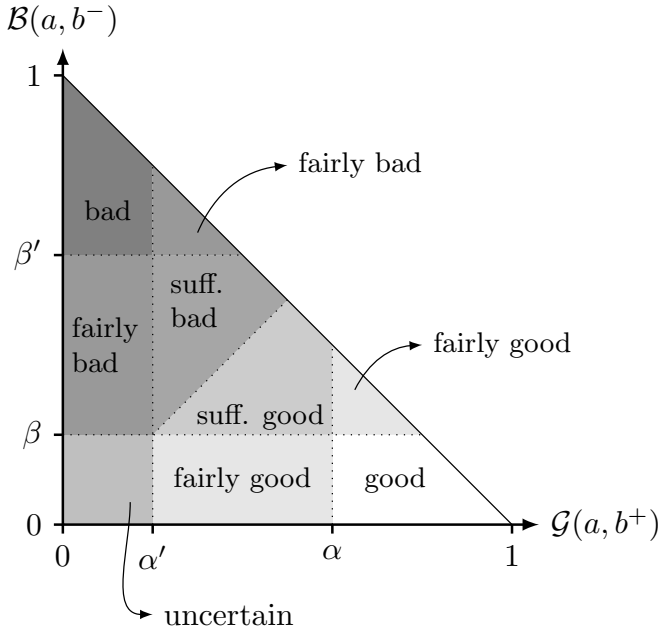


Figure 2.5: nTOMIC: example of partition of the aggregation space.

The two indices above do not take compensations between good and bad scores into account. In order to take compensations into account, Massaglia and Ostanello (1991) introduce a credibility of "goodness" and a credibility of "badness". The credibility of "goodness" is computed as follows:

$$\mathcal{G}^*(a,b^+) = \mathcal{G}(a,b^+) \prod_{j \in N: \mathcal{B}(a,b_j^+) > \mathcal{G}(a,b_j^+)} \frac{1-\mathcal{B}(a,b_j^+)}{1-\mathcal{G}(a,b_j^+)}.$$

The credibility of "badness" is computed as follows:

$$\mathcal{B}^*(a,b^-) = \mathcal{B}(a,b^-) \prod_{j \in N: \mathcal{G}(a,b_j^-) > \mathcal{B}(a,b_j^-)} \frac{1-\mathcal{G}(a,b_j^-)}{1-\mathcal{B}(a,b_j^-)}.$$

We observe that these indices are computed in a similar way as the credibility index in ELECTRE TRI.

After computing the credibility indices, the procedure consists of defining the number of classes in which the alternatives have to be classified. Classes are separated by thresholds which are defined by the DM. Figure 2.5 shows an example of a classification rule assuming a non-compensatory logic with two thresholds for each credibility index. In the figure, the separation thresholds are

denoted by α, α', β and β'. Several classification rules can be defined depending on the DM's attitude.

2.3.3 PROAFTN

This outranking approach has been conceived by Belacel (2000). The method builds fuzzy indices indicating to what extent an alternative belongs to a category. The fuzzy index is based on the concordance and non-discordance principle used in ELECTRE methods.

The method uses class prototypes, i.e. alternatives representing a category. The set of prototypes associated to a class C^h is denoted by $B^h = \{b^{h,1}, \ldots, b^{h,L_h}\}$ with $b^{h,k}$ the k^{th} prototype and L_h the number of prototypes. Each prototype $b^{h,k}$ corresponds to a performance vector on all the criteria involved in the decision problem.

In PROAFTN, a global indifference index is computed for each alternative against each prototype by aggregating a set of partial indifference indices. In the sequel, we denote the indifference relation between an alternative a and a prototype $b^{h,k}$ by $a \sim b^{h,k}$. The method computes partial indifference relations as follows. An interval $[\Delta^-(b_j^{h,k}), \Delta^+(b_j^{h,k})]$ is associated to each profile $b_j^{h,k}$ where $\Delta^-(b_j^{h,k}) \leq b_j^{h,k} \leq \Delta^+(b_j^{h,k})$. An alternative a is said indifferent to prototype $b^{h,k}$ on criterion j if $\Delta^-(b_j^{h,k}) \leq a_j \leq \Delta^+(b_j^{h,k})$. Two thresholds $p^-(b_j^{h,k})$ and $p^+(b_j^{h,k})$ are introduced. They define the limit from which an alternative is not indifferent anymore to prototype $b^{h,k}$. If $\Delta^-(b_j^{h,k}) - p^-(b_j^{h,k}) \leq a_j \leq \Delta^-(b_j^{h,k})$ or $\Delta^+(b_j^{h,k}) \leq a_j \leq \Delta^+(b_j^{h,k}) + p^+(b_j^{h,k})$ then the alternative is said to be weakly indifferent to the prototype $b^{h,k}$ on the criterion j. Finally, if the performance of a on criterion j doesn't respect both conditions then a is said to be "not indifferent" to $b^{h,k}$ on criterion j. The degree of indifference of an alternative a with respect to a prototype $b^{h,j}$ is computed through a partial indifference degree $\mathcal{I}_j(a, b_j^h)$ which is defined as follows:

$$\mathcal{I}_j(a, b^{h,k}) = \min\{\mathcal{I}_j^-(a, b^{h,k}), \mathcal{I}_j^+(a, b^{h,k})\},$$

where $\mathcal{I}_j^-(a, b^{h,k})$ and $\mathcal{I}_j^+(a, b_j^{h,k})$ are computed as follows:

$$\mathcal{I}_j^-(a, b^{h,k}) = \begin{cases} 0 & \text{if } a_j \leq \Delta^-(b_j^{h,k}) - p^-(b_j^{h,k}), \\ 1 & \text{if } a_j \geq \Delta^-(b_j^{h,k}), \\ \frac{a_j - \Delta^-(b_j^{h,k}) + p^-(b_j^{h,k})}{p^-(b_j^{h,k})} & \text{otherwise,} \end{cases}$$

$$\mathcal{I}_j^+(a, b^{h,k}) = \begin{cases} 0 & \text{if } a_j \geq \Delta^+(b_j^{h,k}) + p^+(b_j^{h,k}), \\ 1 & \text{if } a_j \leq \Delta^+(b_j^{h,k}), \\ \frac{\Delta^+(b_j^{h,k}) + p^+(b_j^{h,k}) - a_j}{p^+(b_j^{h,k})} & \text{otherwise.} \end{cases}$$

The partial indifference relations are then aggregated in a comprehensive indifference relation as follows:

$$\mathcal{I}(a, b^{h,k}) = \sum_{j=1}^{n} w_j^h \cdot \mathcal{C}_j(a, b^{h,k}),$$

where $w_j^h \in [0, 1]$ represents the relevance of criterion j for the assignment of an alternative in C^h; a value of 1 means strong relevance of the criterion j and 0, the contrary. These weights are without loss of generality normalized between 0 and 1 so that their sum is equal to 1.

As in other ELECTRE methods, PROAFTN uses a discordance index. Two veto thresholds $v^-(b_j^{h,k})$ and $v^+(b_j^{h,k})$ are defined for each prototype $b_j^{h,k}$. These satisfy $v^-(b_j^{h,k}) \geq p^-(b_j^{h,k})$ and $v^+(b_j^{h,k}) \geq p^+(b_j^{h,k})$. Then partial discordance indices $\mathcal{D}_j(a, b^{h,k})$ are computed as in ELECTRE. The discordance index $\mathcal{D}(a, b^{h,k})$ is aggregated from a set of partial discordance indices $\mathcal{D}_j(a, b^{h,k})$. The value of $\mathcal{D}_j(a, b^{h,k})$ is finally obtained as follows:

$$\mathcal{D}_j(a, b^{h,k}) = \max\{\mathcal{D}_j^-(a, b^{h,k}), \mathcal{D}_j^+(a, b^{h,k})\},$$

where $\mathcal{D}_j^-(a, b^{h,k})$ and $\mathcal{D}_j^+(a, b_j^{h,k})$ are computed as follows:

$$\mathcal{D}_j^-(a, b^{h,k}) = \begin{cases} 0 & \text{if } a_j \geq \Delta^-(b_j^{h,k}) - p_j(b^{h,k}), \\ 1 & \text{if } a_j \leq \Delta^-(b_j^{h,k}) - v_j(b^{h,k}), \\ \frac{\Delta^-(b_j^{h,k}) - p^-(b_j^{h,k}) - a_j}{p^-(b_j^{h,k}) - v^-(b_j^{h,k})} & \text{otherwise,} \end{cases}$$

$$\mathcal{D}_j^+(a, b^{h,k}) = \begin{cases} 0 & \text{if } a_j \leq \Delta^+(b_j^{h,k}) + p_j(b^{h,k}), \\ 1 & \text{if } a_j \geq \Delta^+(b_j^{h,k}) + v_j(b^{h,k}), \\ \frac{a_j - \Delta^+(b_j^{h,k}) - p^+(b_j^{h,k})}{v^+(b_j^{h,k}) - p^+(b_j^{h,k})} & \text{otherwise.} \end{cases}$$

The value of the non-discordance index is derived from it as follows:

$$\mathcal{ND}(a, b^{h,k}) = 1 - \prod_{j=1}^{n} \left(1 - \mathcal{D}_j(a, b^{h,k})\right)^{w_j^h}.$$

The indifference and discordance indices are aggregated into an index $\sigma(a, b^{h,k})$ representing the credibility of the relation $a \sim b^{h,k}$. The credibility index is computed as follows:

$$\sigma(a, b^{h,k}) = \mathcal{I}(a, b^{h,k}) \cdot \mathcal{ND}(a, b^{h,k}).$$

Finally, a fuzzy degree indicating the degree to which an alternative a belongs to a category C^h is obtained by taking the maximal value of $\sigma(a, b^{h,k})$ among all the prototypes $b^{h,k}$ associated to a category C^h, i.e.,

$$\sigma(a \in C^h) = \max\{\sigma(a, b^{h,k}) : k = \{1, \ldots, L_h\}\}.$$

An alternative a is assigned to the category C^h for which the fuzzy degree $\sigma(a, b^{h,k})$ is the largest:

$$a \in C^h \iff \sigma(a \in C^h) = \max\{\sigma(a \in C^l) : l = \{1, \ldots, p\}\}.$$

The relationship with other outranking approaches is clear. PROAFTN uses a concordance and discordance relation as in ELECTRE III and ELECTRE TRI. However PROAFTN assigns an alternative to a category using an indifference relation with respect to this category. In PROAFTN, the categories are not necessary ordinal.

2.3.4 ELECTRE TRI-C and ELECTRE TRI-nC

ELECTRE TRI-C is a variant of ELECTRE TRI which uses central profiles in order to characterize categories. ELECTRE TRI-C has been proposed by Almeida-Dias et al. (2011). In this model, a reference profile is associated to each category going from C^1 to C^p with $C^p \succ C^{p-1} \succ \ldots \succ C^1$. The reference profile denoted by $\tilde{b}^h$ is the action which is the most representative of a category C^h. Two fictitious profiles $\tilde{b}^0$ and $\tilde{b}^{p+1}$ are added to the list of profiles, with $\tilde{b}^0_j$ the worst possible performance on criterion j and $\tilde{b}^{p+1}_j$ the best possible performance on criterion j. The method uses a credibility index as in ELECTRE TRI. An indifference (q_j), a preference (p_j) and possibly a veto threshold (v_j) are associated to each criterion $j \in N$.

In ELECTRE TRI-C it is supposed that the profiles strictly dominate one another as in ELECTRE TRI. This condition is however not sufficient. Indeed, consider two consecutive profiles $\tilde{b}^h$ and $\tilde{b}^{h+1}$, if the performances of these profiles are such that $0 \leq \tilde{b}^{h+1}_1 - \tilde{b}^h_j \leq q_j$ for each criterion $j \in N$, then we have $\sigma(\tilde{b}^h, \tilde{b}^{h+1}) = 1$ which means that the two profiles are not distinct. According to the value taken by $\sigma(\tilde{b}^h, \tilde{b}^{h+1})$, Almeida-Dias et al. (2011) define weak ($\sigma(\tilde{b}^h, \tilde{b}^{h+1}) < 1$), strict ($\sigma(\tilde{b}^h, \tilde{b}^{h+1}) < 0.5$) and hyper-strict ($\sigma(\tilde{b}^h, \tilde{b}^{h+1}) = 0$) separability conditions.

ELECTRE TRI-C requires the definition of a selection function denoted by $\rho(a, \tilde{b}^h)$. This selection function allows to select between two consecutive categories the one to which a should be assigned. The value of $\rho(a, \tilde{b}^h)$ should depend on the values of $\sigma(a, \tilde{b}^h)$ and $\sigma(\tilde{b}^h, a)$. For instance, one can use $\rho(a, \tilde{b}^h) = \min\{\sigma(a, b^h), \sigma(b^h, a)\}$ as selection function.

To assign an alternative a to a category C^h, ELECTRE TRI-C proposes two joint assignment rules, one descending and one ascending. We describe these rules below.

Descending rule Compare alternative a successively to $\tilde{b}^p, \tilde{b}^{p-1}, \ldots, \tilde{b}^1, \tilde{b}^0$ until the first value h such that $\sigma(a, \tilde{b}^h) \geq \lambda$:

(a) For $h = p$, select C^p as a possible category to assign a

(b) For $0 < h < p$, if $\rho(a, b^h) > \rho(a, b^{h+1})$, then select C^h as a possible category to assign a; otherwise select C^{h+1}.

(c) For $h = 0$, select C^1 as a possible category to assign a.

Ascending rule Compare alternative a successively to $\tilde{b}^0, \tilde{b}^1, \ldots, \tilde{b}^{p-1}, \tilde{b}^p$ until the first value h such that $\sigma(\tilde{b}^h, a) \geq \lambda$:

(a) For $h = 1$, select C^1 as a possible category to assign a.

(b) For $1 < h < p+1$, if $\rho(a, b^h) > \rho(a, b^{h-1})$, then select C^h as a possible category to assign a, otherwise, select C^{h-1}.

(c) For $h = p + 1$, select C^h as a possible category to assign a.

The two assignment procedures are used jointly, meaning that if an alternative a is assigned to two different categories, C^k by the descending rule and C^l by the ascending rule, then a is assigned to an interval of categories delimited by C^k and C^l.

Bouyssou and Marchant (2015) showed that the relationships between ELECTRE TRI-C and ELECTRE TRI are complex: some assignments obtained with one method can not be obtained by the other and vice-versa. They propose to search for a method working with central profiles which is closer to ELECTRE TRI.

ELECTRE TRI-nC is a variant of ELECTRE TRI-C in which a category is characterized by several reference actions. More information about this method can be found in Almeida-Dias et al. (2012).

2.3.5 FlowSort

FlowSort is a sorting model that is based on PROMETHEE, an outranking method proposed by Brans et al. (1984) that handles ranking problems. The PROMETHEE method has been adapted in order to handle sorting problems by Nemery and Lamboray (2008). In FlowSort, categories are either separated by limiting profiles or characterized by central profiles.

Consider an alternative a that has to be assigned to a category. Categories are defined by central or limiting profiles denoted by $\mathcal{R}$. We denote by $\mathcal{R}_a$ the

set containing the profiles and the alternative a, i.e., $\mathcal{R}_a = \mathcal{R} \cup \{a\}$. In FlowSort, two types of flows are computed:

1. a positive outranking flow which represents to what extent the alternative a outranks the profiles $\mathcal{R}$;

2. a negative outranking flow which represents to what extent the alternative a is outranked by the profiles $\mathcal{R}$.

The positive and negative flows of an alternative $x \in \mathcal{R}_a$ are computed with the following formula:

$$\phi^+_{\mathcal{R}_a}(x) = \frac{1}{|\mathcal{R}_a| - 1} \sum_{y \in \mathcal{R}_a} \pi(x, y),$$
$$\phi^-_{\mathcal{R}_a}(x) = \frac{1}{|\mathcal{R}_a| - 1} \sum_{y \in \mathcal{R}_a} \pi(y, x),$$

in which $\pi(x, y)$ (resp. $\pi(y, x)$) is a preference degree function which evaluates the preference strength of an alternative x (resp. y) over another one y (resp. x). We refer the reader to Brans and Vincke (1985) for further details on the definition of $\pi(x, y)$.

As in ELECTRE TRI, categories can be separated by limiting profiles. We denote by b^h the profile delimiting the category C^{h-1} from C^h. In case of limiting profiles, there exist two assignment rules:

$$a \in C^h \qquad \text{if } \phi^+_{\mathcal{R}_a}(b^h) \geq \phi^+_{\mathcal{R}_a}(a) > \phi^+_{\mathcal{R}_a}(b^{h-1}),$$
$$a \in C^h \qquad \text{if } \phi^-_{\mathcal{R}_a}(b^h) < \phi^-_{\mathcal{R}_a}(a) \leq \phi^-_{\mathcal{R}_a}(b^{h-1}).$$

The procedure can lead to two different assignments for an alternative a. It is also possible to combine positive and negative flows so that an alternative is imposed to exactly one category. The combination of both flows gives the net flow:

$$\phi_{\mathcal{R}_a}(x) = \phi^+_{\mathcal{R}_a}(x) - \phi^-_{\mathcal{R}_a}(x). \tag{2.21}$$

Using the net flow, the assignment rule becomes:

$$a \in C^h \qquad \text{if } \phi_{\mathcal{R}_a}(b^h) \geq \phi_{\mathcal{R}_a}(a) > \phi_{\mathcal{R}_a}(b^{h-1}).$$

Like ELECTRE TRI-C and ELECTRE TRI-nC, FlowSort is able to work with central profiles characterizing the categories. We denote by $\tilde{b}^h$ the profile characterizing the category C^h. Using such type of profiles, the assignment rules

become:

$$a \in C^h \quad \text{if } \frac{\phi^+_{\mathcal{R}_a}(\tilde{b}^h) + \phi^+_{\mathcal{R}_a}(\tilde{b}^{h-1})}{2} < \phi^+_{\mathcal{R}_a}(a) \leq \frac{\phi^+_{\mathcal{R}_a}(\tilde{b}^h) + \phi^+_{\mathcal{R}_a}(\tilde{b}^{h+1})}{2},$$
$$a \in C^h \quad \text{if } \frac{\phi^-_{\mathcal{R}_a}(\tilde{b}^h) + \phi^-_{\mathcal{R}_a}(\tilde{b}^{h-1})}{2} \geq \phi^-_{\mathcal{R}_a}(a) > \frac{\phi^-_{\mathcal{R}_a}(\tilde{b}^h) + \phi^-_{\mathcal{R}_a}(\tilde{b}^{h+1})}{2}.$$

Again, an alternative a can be assigned to two different categories by the two rules given above. By using the net flow given in Equation (2.21), it is possible to assign an alternative to a single category:

$$a \in C^h \quad \text{if } \frac{\phi_{\mathcal{R}_a}(\tilde{b}^h) + \phi_{\mathcal{R}_a}(\tilde{b}^{h-1})}{2} < \phi_{\mathcal{R}_a}(a) \leq \frac{\phi_{\mathcal{R}_a}(\tilde{b}^h) + \phi_{\mathcal{R}_a}(\tilde{b}^{h+1})}{2}.$$

2.3.6 TOMASO

Marichal et al. (2005) proposed a new method for sorting alternatives into categories. It is called TOMASO (tool for ordinal multi-attribute sorting and ordering). The method works in two steps. The first one consists in computing a score for each alternative on each criterion. The second step consists in aggregating these scores with a Choquet integral.

Consider a set of m alternatives A evaluated on n criteria. We denote by X_j the list of possible alternative evaluations on criterion j. The set X_j composed of n_j values is totally ordered, we have $X_j = \left\{x_j^{(n_j)} \succ x_j^{(n_j-1)} \succ \ldots \succ x_j^{(1)}\right\}$.

Two approaches are considered to compute the score of an alternative a on a criterion j. It is up to the analyst to decide which one to use. The first approach builds the score of an alternative based on all the alternatives of the set A:

$$Sc_j(a) = \frac{\Omega_j(a) + (m-1)}{2 \cdot (m-1)},$$

where $\Omega_j(a)$ is the number of alternatives that a is preferred to on criterion j minus the number of alternatives which are preferred to a, i.e., $\Omega_j(a) = |\{x \in A : a_j \succ x_j\}| - |\{x \in A : x_j \succ a_j\}|$. The second approach builds the score of an alternative independently from the other alternatives. It is defined as:

$$Sc_j(a) = \frac{r-1}{n_j - 1},$$

with r such that $a_j = x_j^{(r)}$.

A Choquet integral is then used to aggregate all the scores of each alternative a. Consider a capacity μ, i.e. a monotone function $\mu : 2^N \to [0,1]$ fulfilling

$\mu(\emptyset) = 0$ and $\mu(N) = 1$. The Choquet integral is defined as follows:

$$C_\mu(a) = \sum_{j=1}^{n} Sc_{\tau(j)}(a) \cdot \left[\mu(N_{\tau(j)}) - \mu(N_{\tau(j+1)})\right],$$

where τ is a permutation of the indices $\{1, \ldots, n\}$ such that $0 \leq Sc_{\tau(1)}(a) \leq \ldots \leq Sc_{\tau(n)}(a)$ and $N_{(j)} = \{j, \ldots, n\}$. $\mu(N_{\tau(j)})$ denotes the capacity of the subset of criteria $N_{\tau(j)} \subseteq N$. The assignment of an alternative is then obtained by comparing the fuzzy measure $C_\mu(a)$ to thresholds delimiting the categories.

2.3.7 Summary

In this section we gave an overview of MCDA methods for sorting alternatives in categories. The methods can be distinguished along several dimensions.

Except AVF-Sort methods, all the sorting methods presented up to now belong to the family of outranking methods.

We can discriminate the methods by the type of profiles that are used. In ELECTRE TRI, MR-Sort, NCS, UTADIS, trichotomic segmentation, nTOMIC and TOMASO the categories are separated by limiting profiles. Each profile is a boundary between two consecutive categories. ELECTRE TRI-C, ELECTRE TRI-nC and PROAFTN use central profiles. Each central profile represents a typical example of the category. FlowSort is a method that works either with central or limiting profiles.

Most methods presented in this section are based on the concordance and discordance principles. It is the case for ELECTRE TRI, MR-Sort, NCS, ELECTRE TRI-C, ELECTRE TRI-nC and PROAFTN. In these methods, the assignment of an alternative is obtained by comparing it successively to each profile. In FlowSort the assignment of an alternative is done by using flows as scores as in UTADIS.

TOMASO is the only sorting method presented here that proposes the use of capacities instead of additive weights.

2.4 Learning the parameters of multiple-criteria decision analysis models

In this section, we first describe direct and indirect methods for the elicitation of MCDA models parameters. Then a large part of the section describes some indirect elicitation methods for sorting models. After that, one part of the section describes how the robustness of the model outcome is assessed and guaranteed in MCDA. Finally the last part of the section describes how inconsistencies are handled in MCDA.

2.4.1 Direct and indirect parameters elicitation

In the literature, we distinguish direct and indirect methods for the elicitation of model parameters.

Direct methods consist in eliciting the parameters of the method by interacting with the decision maker. For instance, the analyst asks the DM for a percentage corresponding to the importance given by the DM to each criterion. The main difference between direct and indirect methods is that in indirect methods, the parameters of the model are inferred from a set of preference and indifference judgments.

Eliciting directly the method parameters is not easy. That is why indirect elicitation techniques have been developed. For instance, Simos (1990); Simos and Maystre (1990) proposed a simple procedure that allows to determine indirectly the weights of ELECTRE methods by using a set of cards. An improvement of the procedure has been proposed by Figueira and Roy (2002). Recently some robustness concerns about these methods have been outlined by Siskos and Tsotsolas (2015). Another example of indirect elicitation is the method of indifference judgments used to elicit an additive value function asks the DM to assess the value of a criterion that equilibrates the comparison of two alternatives. The marginal value functions are elicited through such judgments, that may deal with alternatives. The parameters of the method do not appear explicitly.

Among indirect elicitation techniques, some allow to learn all the model parameters and some others are dedicated to learn only a subset of the method parameters. As an example, in recent years, several papers have been dedicated to the parameters inference for ELECTRE TRI. Mousseau and Słowiński (1998) proposed a nonlinear program in order to determine all the parameters of the method based on assignment examples. In Ngo The and Mousseau (2002) only the profiles of an ELECTRE TRI model are learned on the basis of assignment examples.

2.4.2 Learning the parameters of ELECTRE TRI and MR-Sort

In recent years, several papers dealt with the learning of ELECTRE TRI and MR-Sort parameters.

The first paper devoted to the learning of ELECTRE TRI parameters has been proposed by Mousseau and Słowiński (1998). The learning algorithm takes as input a set of assignment examples and their associated vector of performances with respect to the problem criteria. The paper shows the difficulties to learn the parameters of ELECTRE TRI without veto. The main difficulty is the non-linearity of the partial concordance indices. Indeed, it makes the concordance index not differentiable which prevents the use of gradient optimization algo-

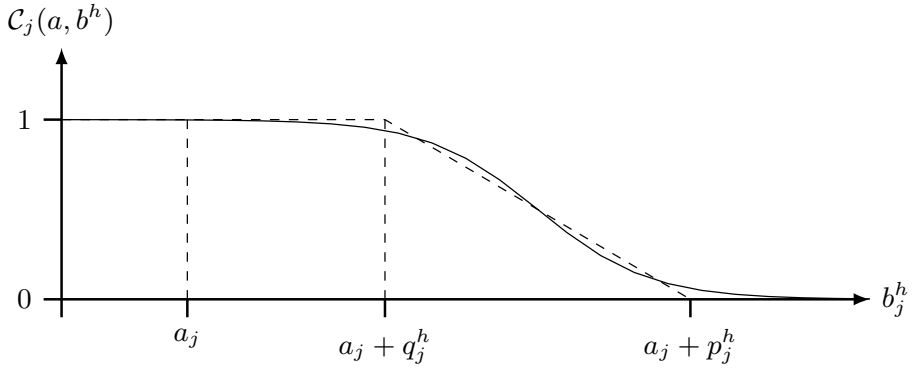


Figure 2.6: Partial concordance function approximated by a sigmoid.

rithms. In order to tackle this difficulty, Mousseau and Słowiński (1998) propose to approximate the partial concordance indices by sigmoid functions as shown in Figure 2.6.

Learning all the parameters of an ELECTRE TRI model involves the determination of a lot of parameters. It requires a lot of cognitive effort from the user. Mousseau et al. (2001) consider the subproblem of finding the weights and the cutting threshold of an ELECTRE TRI model with fixed profiles and indifference and preference thresholds. In the paper, a linear program (LP) is proposed and some experiments are conducted. It shows that learning only a subpart of the ELECTRE TRI model simplifies the problem. Less assignment examples are required to obtain good results.

Ngo The and Mousseau (2002) proposed a mixed integer program (MIP) in order to infer the profiles of an ELECTRE TRI model with fixed weights and thresholds. The MIP presented in the paper finds the partial concordance indices in a first step. The second steps consists in deducing the values of the profiles from the partial concordance indices. They propose to use this MIP in combination with the LP of Mousseau et al. (2001) in order to determine the whole set of parameters of an ELECTRE TRI model.

Mousseau and Słowiński (1998), Mousseau et al. (2001) and Ngo The and Mousseau (2002) consider only ELECTRE TRI models without vetoes. Dias and Mousseau (2006) present a manner to learn vetoes of an ELECTRE TRI model with fixed profiles, thresholds and weights. In the paper, two subproblems are treated. The first one considers the inference of veto parameters for a single criterion. The second considers the inference of all veto parameters for multiple criteria at the same time.

Doumpos et al. (2009) proposed a metaheuristic in order to learn all the parameters of an ELECTRE TRI model, including the veto thresholds. They de-

veloped a genetic algorithm in order to learn all the parameters of an ELECTRE TRI model at the same time. The interest of this approach is that it allows to deal with larger data sets than MIP based algorithms.

As shown in Section 2.2.1, ELECTRE TRI presents some drawbacks, one of them being the large number of parameters to determine. MR-Sort is a simplified version of ELECTRE TRI which keeps the philosophy of ELECTRE TRI with the advantage of using less parameters. Leroy et al. (2011) propose a MIP in order to learn the parameters of such a model based on assignment examples. The experimental results presented in the paper show that the MIP is able to find MR-Sort models which perform well in generalization. However, the experiments show the limitation of such an algorithm in terms of computing time. For a small problem involving 5 categories and 3 criteria, more than 100 seconds are required to restore all the parameters of a MR-Sort model on the basis of 100 assignment examples.

Damart et al. (2007) are the first to consider the problem of learning the parameters of an ELECTRE TRI model in the context of multiple decision makers. They propose an approach that aims at determining a set of fictitious alternatives that contain enough information to obtain a model that is satisfactory for all the DMs. The procedure is applied to an illustrative example.

Later, Cailloux et al. (2012) developed two MIPs in order to learn the parameters of a MR-Sort model in the context of multiple DMs. The first MIP aims at finding a set of profiles that is common to all the decision makers. The second MIP learns a set of weights compatible with the preferences of each DM. The paper presents experimental results on real and fictitious applications.

2.4.3 Learning the parameters of additive value function models

Learning the parameters of an additive value function model has been covered in several papers. In this subsection, we recall the main papers that are devoted to the inference of parameters of an additive value function model. We first recall the inference procedures that have been developed for ranking problems, then we cover sorting problems.

Ranking problems

Jacquet-Lagrèze and Siskos (1982) are the first to propose a disaggregation procedure in order to infer the parameters of a set of additive value functions. They proposed a LP in order to infer the additive value functions from a given ranking. They called the method UTA (utilités additives). We recall here the principle of this approach as we use it later in this thesis.

Consider a ranking of m alternatives from the best to the worst. This set of alternatives constitutes the learning set and is denoted by A^*. Given two consecutive alternatives a and b in the ranking, the preference relation between a and b is either "a is preferred to b" or "a is indifferent to b". Let $\mathcal{P}$ be the set containing all the pairs (a, b) such that a is preferred to b and $\mathcal{I}$ the set containing all the pairs (a, b) such that a is indifferent to b. To translate the ranking into a linear program, Jacquet-Lagrèze and Siskos (1982) introduce one error variable per alternative, such that the biased value of an alternative a, $U'(a)$, is expressed as $U'(a) = U(a) + \sigma(a)$. For each pair of alternatives $(a, b) \in \mathcal{P}$, we have $U'(a) - U'(b) > 0$ and each pair $(a, b) \in \mathcal{I}$, we have $U'(a) - U'(b) = 0$. Finally, the UTA linear program can be expressed as follows. The LP minimizes the following objective function:

$$\min_{u_j^*} \sum_{a \in A^*} \sigma(a), \tag{2.22}$$

such that:

$$\begin{cases} U(a) - U(b) + \sigma(a) - \sigma(b) & \geq & \varepsilon & \forall (a, b) \in \mathcal{P}, \\ U(a) - U(b) + \sigma(a) - \sigma(b) & = & 0 & \forall (a, b) \in \mathcal{I}, \\ \sum_{j=1}^{n} u_j^*(\overline{a_j}) & = & 1, & \\ u_j^*(\underline{a_j}) & = & 0 & \forall j \in N, \\ \sigma(a) & \geq & 0 & \forall a \in A^*, \\ u_j^* & \text{monotonic} & & \forall j \in N, \end{cases} \tag{2.23}$$

with ε a small positive value.

The third and fourth constraints given in Equation (2.23) ensure the normalisation of the value functions. The constraints can be easily formulated in a linear programming solver when piecewise linear marginal value functions are used (see Figure 2.3 for an example of such a function). Indeed, it is easy to ensure monotonicity of a piecewise linear function u_j^* by ensuring that for each breakpoint g_j^l, with $l = \{1, \ldots, \eta_j\}$, we have $u_j^*(g_j^l) \geq u_j(g_j^{l-1})$.

UTA is at the origin of several other disaggregation procedures. An improvement of UTA, called UTASTAR, has been proposed by Siskos and Yanacopoulos (1985). This variant of UTA replaces the error variable associated to an alternative a, $\sigma(a)$, by two positive error variables $\sigma^+(a)$ and $\sigma^-(a)$. The UTASTAR LP minimizes the following objective:

$$\min_{u_j^*} \sum_{a \in A^*} \left(\sigma^+(a) + \sigma^-(a)\right) \tag{2.24}$$

such that:

$$\left\{\begin{array}{rcll} U(a)-U(b)+\sigma^+(a)-\sigma^-(a) & & & \\ -\sigma^+(b)+\sigma^-(b) & \geq & \varepsilon & \forall (a,b)\in\mathcal{P}, \\ U(a)-U(b)+\sigma^+(a)-\sigma^-(a) & & & \\ -\sigma^+(b)+\sigma^-(b) & = & 0 & \forall (a,b)\in\mathcal{I}, \\ \sum_{j=1}^{n}\sum_{l=1}^{\eta_j} u_j^*(g_j^l) & = & 1, & \\ \sigma^+(a),\sigma^-(a) & \geq & 0 & \forall a\in A^*, \\ u_j^*(g_j^l) & \geq & 0 & l=\{1,\ldots,\eta_j\}, j=\{1,\ldots,n\}, \end{array}\right. \tag{2.25}$$

with ε a small positive value.

In UTASTAR, the existence of near optimal solutions is tested by running for each criterion j the above LP with another objective maximizing the value function j and with a constraints ensuring that the sum of slack variables is bounded by the optimal value found previously with the original LP. All the value functions found for a criterion j are then aggregated in order to obtain a mean value function. Such a method allows to obtain a central solution. This variant of UTA allows to reduce the number of ranking errors as shown experimentally by Siskos and Yanacopoulos (1985). Note that ACUTA (Bous et al., 2010) is another method that returns a central solution. We refer the reader to Jacquet-Lagrèze and Siskos (2001) for a review of UTA variants developed up to 2001.

In recent years, other UTA based methods have been developed. Słowiński et al. (2005) developed UTA$^{\text{GMS}}$ which considers all possible additive value functions compatible with the DM's preferences. It aims at obtaining a robust model. It consists in computing the analytic center of the polyhedron formed by the set of constraints deduced from the sets $\mathcal{P}$ and $\mathcal{I}$.

Sorting problems

It is easy to transpose the inference of an additive value function model to sorting problems. We recall here the principles of UTADIS (utilités additives discriminantes) and its variants.

The UTADIS method (Devaud et al., 1980; Jacquet-Lagrèze and Siskos, 1982) aims at finding an additive value functions that restores as good as possible the preferences of a DM. The method takes as input a set of assignment examples and gives an AVF-Sort model as output. The inference of model parameters from assignment examples is achieved with a LP. As in UTASTAR, the method associates a positive and negative error, $\sigma^+(a)$ and $\sigma^-(a)$, to each alternative a of the learning set A^*. Consider a model involving p categories such that $C^p \succ C^{p-1} \succ \ldots \succ C^1$. Each pair of consecutive categories C^h and C^{h+1} is separated by a threshold denoted by U^h, with $h=\{1,\ldots,p-1\}$. We denote by

A^{*h}, the set of alternatives of the learning set A^* that are assigned to category C^h. The objective function of the UTADIS linear program is given by:

$$\min_{u_j^*} \sum_{a \in A} \left(\sigma^+(a) + \sigma^-(a)\right) \tag{2.26}$$

such that:

$$\begin{cases} U(a) + \sigma^+(a) & \geq & U^{h-1} & \forall a \in A^{*h}, h = \{2, \ldots, p\}, \\ U(a) - \sigma^-(a) & \leq & U^h - \varepsilon & \forall a \in A^{*h}, h = \{1, \ldots, p-1\}, \\ U^h & \geq & U^{h-1} & h = \{2, \ldots, p-1\}, \\ \sum_{j=1}^{n} u_j^*(\overline{a_j}) & = & 1, & \\ u_j^*(a_j) & = & 0, & \\ \overline{U^h} & \in & [0,1] & h = \{2, \ldots, p-1\}, \\ \sigma^+(a) & \geq & 0 & \forall a \in A^*, \\ \sigma^-(a) & \geq & 0 & \forall a \in A^*, \\ u_j^* & \text{monotonic} & & \forall j \in N, \end{cases} \tag{2.27}$$

with ε a small positive value.

As in UTA and UTASTAR, the monotonicity of u_j^* is ensured by using piecewise linear functions and by guarantying that $u_j^*(g_j^l) \geq u_j^*(g_j^{l-1})$ for $l = \{1, \ldots, \eta_j\}$ and $j = \{1, \ldots, n\}$.

Zopounidis and Doumpos (2000) recall some extensions of UTADIS proposed in the past. One consists in transforming the LP into a MIP in which the objective is modified in order to minimize the misclassification errors. Another MIP variant introduces a cost for each misclassification. In this thesis, we use the UTADIS formulation given by Equation (2.26) and Equation (2.27).

2.4.4 Learning algorithms for other sorting models

This part of the section describes more in depth algorithms that are used for learning the parameters of sorting models. Some of these models have been presented in Section 2.3.

M.H.DIS

Multi-group hierarchical discrimination method (M.H.DIS) has been proposed by Zopounidis and Doumpos (2000). Compared to other MCDA inference methods, it separates progressively the best alternatives from all the others. M.H.DIS uses additive value functions having the following form:

$$U^h(a) = \sum_{j=1}^{n} u_j^{*h}(a_j),$$

where $U^h(a)$ denotes the global value of classifying a in category C^h and $u_j^h(a_j)$ denotes the marginal value function of classifying a in C^h on criterion j. Marginal value $u_j^h(a_j)$ is an increasing or decreasing function on the criterion scale. Similarly, the additive value function $U^{\sim h}$ is defined as the value function of not classifying a in category C^h. The decision to classify an alternative a in a category C^h is made based on these two values:

$$a \in C^h \iff U^h(a) > U^{\sim h}(a), \text{ and}$$
$$a \notin C^h \iff U^h(a) < U^{\sim h}(a).$$

The case $U^h(a) = U^{\sim h}(a)$ is considered as a misclassification.

In a model involving p categories, the M.H.DIS procedure requires the elicitation of $2(p-1)$ value functions. The elicitation of these functions is done through mathematical programming based on a set of assignment examples. The method uses two LP and one MIP. As in UTA and UTADIS, the marginals are piece-wise linear functions. However, their breakpoints are not equally spaced on the criterion domain as in UTA and UTADIS. The positions of the breakpoints correspond to the evaluations of the alternatives that have to be sorted. We refer the reader to Zopounidis and Doumpos (2000) for more details about the mathematical programs.

To determine the assignment of an alternative a, a hierarchical process is applied. First, the value of $U^p(a)$ and $U^{\sim p}(a)$ are computed. If $U^p(a) > U^{\sim p}(a)$, then a is assigned to C^p and the procedure stops. Otherwise, if $U^p(a) < U^{\sim p}(a)$, then the values of a for the category $p-1$ are computed and the same rule is applied. The procedure continues as long as there is no category h such that $U^h(a) > U^{\sim h}(a)$.

Interactive procedure for selecting acceptable alternatives in the presence of multiple criteria

Ulu and Köksalan (2001) proposed a procedure partitioning alternatives into two classes: accepted or refused. This interactive algorithm aims at reducing the number of questions a DM has to answer.

The algorithm tries to discriminate the alternatives in the two categories on the basis of previous DM's preferences and dominance relationships. The dominance relationship is exploited as follows. Consider two performance vectors $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$ evaluated on the n criteria. We denote by a^T (resp. b^T) the transpose of vector a (resp. b). The following statements holds:

- If a is known to be accepted and is dominated by b on all criteria, then b is also accepted;

- If a is known to be refused and dominates b on all criteria, then b is also refused;

- If a is known to be accepted, b does not dominate a on all the criteria and if it is not possible to find a set of positive weights $w = (w_1, \ldots, w_n)$ such that $w \cdot a^{\mathsf{T}} > w \cdot b^{\mathsf{T}}$ then b is also accepted;

- If a is known to be refused, b is not dominated by a on all the criteria and if it is not possible to find a set of positive weights $w = (w_1, \ldots, w_n)$ such that $w \cdot b^{\mathsf{T}} > w \cdot a^{\mathsf{T}}$ then b is also refused.

The last two tests can be done through linear programming. These four relations allow to identify accepted and refused alternatives without having to ask the DM. Based on these information, an algorithm is built in order to split the set of alternatives in a set containing the accepted ones and another containing the rejected ones.

Outranking approach based on reference profiles

Köksalan et al. (2009) propose an outranking approach similar to ELECTRE TRI. They consider that it is difficult for a DM to determine category limits and easier for him to provide examples of alternatives that belong to a category. They propose a new outranking approach that does not require the elicitation of category profiles.

The first step in the algorithm consists in asking the DM to classify a subset of alternatives in the different categories involved in the MCDA problem. We denote by A the set of alternatives that have to be assigned and A^* denotes the subset of alternatives that have been assigned by the DM. They advocate the assignment of equal numbers of alternatives in each category. Then a LP is used in order to identify and eliminate incompatible assignments. An example of incompatible assignment occurs when two alternatives assigned to two different categories outrank each other.

The second step of the process consists in assigning each alternative of A in a category. It is done based on the reference assignments given by the DM using a MIP. The MIP tries to find model parameters such that an alternative of a higher category outranks an alternative of a lower category. The objective of the MIP is to minimize the violation of these constraints.

In the paper, they also propose to consider assigning alternatives to a range of categories. It is done by modifying the constraints in the MIP.

ORCLASS

Larichev and Moshkovich (1997) developed ORCLASS (ordinal classification), a MCDA method which does not use parameters. Decision maker's preferences are described through decision rules. ORCLASS is described as a verbal decision analysis method (Moshkovich et al., 2005) based on cognitive psychology principles.

The method consists of first identifying the criteria and the list of possible values on these criteria. Criteria values are sorted in ascending order from the worst value to the best. The decision maker preferences are then used to determine a classification rule. As an example, consider a decision problem involving two criteria, $N = \{1, 2\}$, and two classes C^1 and C^2, such that $C^2 \succ C^1$. Let $X = X_1 \times X_2$ be the Cartesian product of criteria scales. We suppose that each criterion has three possible values: $X_1 = \{x_1^1, x_1^2, x_1^3\}$ and $X_2 = \{x_2^1, x_2^2, x_2^3\}$. Values of X_1 (resp. X_2) are ordered such that $x_1^1 < x_1^2 < x_1^3$ (resp. $x_2^1 < x_2^2 < x_2^3$). We consider without loss of generality that the criteria values have to be maximized. Given this, one can assume that an alternative characterized by the vector of performances (x_1^3, x_2^3) is classified in the category C^2 while an alternative characterized by the vector of performances (x_1^1, x_2^1) is classified in C^1. To determine the boundaries of the categories, performance vectors are presented to the decision maker and the classification rule is determined by applying dominance relations. As an example, consider that the decision maker specified that (x_1^1, x_2^2) belongs to class C^2. From this statement, one can deduce that (x_1^1, x_2^3), (x_1^2, x_2^2), (x_1^3, x_2^2) and (x_1^2, x_2^3) are classified in C^2 through dominance relations. On the contrary, if the DM chooses to assign the performance vector (x_1^1, x_2^2) to the category C^1, no other assignment can be deducted. Classification tables similar to the one presented in Table 2.2 are constructed. Each cell of the table corresponds to a particular vector of performance. Each cell contains two numbers, the left one corresponds to the number of assignments that can be deduced if the associated performance vector is classified in C^1 and the right one corresponds to the number of assignments that can be deduced if the performance vector is classified in C^2. In the case of the example above, knowing that (x_1^1, x_2^2) belongs to C^1 has no further consequence while, assigning it to C^2 yields four consequences. Hence we write 1|5 in the corresponding cell of the board. ORCLASS limits the number of questions the DM has to answer by considering primarily the vectors that allow to deduce the more information. Table 2.2 is updated after each DM statement about a given performance vector. It is necessary to check the consistency of the DM to avoid inconsistencies. If an inconsistency is discovered, it has to be solved in accordance with the DM.

One of the advantage of ORCLASS is that it does not require the elicitation of parameters. The DM doesn't have to understand how the sorting model works and is directly confronted to his preferences. A second advantage of such a type

Table 2.2: Example of classification board used in the ORCLASS method.

	x_1^1	x_1^2	x_1^3
x_2^1	C^1	1 | 5	2 | 2
x_2^2	1 | 5	4 | 4	5 | 1
x_2^3	2 | 2	5 | 1	C^2

of model is that incompatibilities are directly detected. Unfortunately, when the number of classes, criteria or criteria values increases, the classification rules become complex and the number of questions to ask the DM heavily increases.

Dominance rough set approach

Rough sets (Pawlak, 1982) have been adapted and introduced in MCDA by Greco et al. (2001) under the name "dominance rough set approach (DRSA)". This approach characterizes the membership to categories using a collection of upwards and downwards unions of categories.

Consider a set of objects A, evaluated on a set of n criteria $N = \{1, \ldots, n\}$. Each object $a \in A$ is classified in a category C^h selected among a set of p categories. Categories are ordered, such that $C^p \succ C^{p-1} \succ \ldots \succ C^1$. We denote by $A^{\geq h}$ the set of alternatives in A assigned to a category better than or equal to C^h. Similarly, $A^{\leq h}$ denotes the set of alternatives in A assigned to a category worse than or equal to C^h.

We define the dominance relation as follows. Let $P \subseteq N$ be a subset of criteria. An alternative $a \in A$ dominates an other $b \in A$ with respect to criteria in P if a is better than b on each criterion $j \in P$, i.e. $a_j \geq b_j$. Formally it reads:

$$a \succcurlyeq_P b \iff a_j \geq b_j, \forall j \in P.$$

We denote by $D_P^+(a) = \{b \in A : b \succcurlyeq_P a\}$ the subset of alternatives in A that dominate a. The set $D_P^-(a) = \{b \in A : a \succcurlyeq_P b\}$ denotes the subset of alternatives in A that are dominated by a.

The dominance relation defined above is used to determine the P-lower and P-upper approximations of upward and downward unions of dominated sets. For a set $A^{\geq h}$ these downwards and upwards sets are denoted respectively by $\underline{P}(A^{\geq h})$ and $\overline{P}(A^{\geq h})$ and defined as:

$$\underline{P}(A^{\geq h}) = \{a \in A : D_P^+(a) \subseteq A^{\geq h}\},$$
$$\overline{P}(A^{\geq h}) = \{a \in A : D_P^-(a) \cap A^{\geq h} \neq \emptyset\}.$$

Similarly, $\underline{P}(A^{\leq h})$ and $\overline{P}(A^{\leq h})$ are defined as:

$$\underline{P}(A^{\leq h}) = \{a \in A : D_P^+(a) \subseteq A^{\leq h}\},$$
$$\overline{P}(A^{\leq h}) = \{a \in A : D_P^-(a) \cap A^{\leq h} \neq \emptyset\}.$$

After having determined the lower and upper category approximations, a set of rules can be deduced. By using this set of rules, one can predict the category of an alternative.

This technique has the advantage to provide a set of rules directly to the DM. There is no need to explain a model to the DM. Indeed, the assignment of an alternative can be simply explained by reading the rules defining category boundaries. Rules that are deduced are for instance of the type "a is assigned to category C^2 if its performance is greater than 0.5 on criterion 1 and if its performance is greater than 0.7 on criterion 2". However, when the number of rules increases, it becomes difficult to read them and to understand the reason why an alternative has been assigned to a given category.

2.4.5 Robustness of the models

In MCDA, the robustness of the elicited model has always been an important issue. The concept of robustness has been introduced by Roy (1998). Later, Vincke (1999) formalized this concept. In MCDA, the robustness of the entire procedure is considered. Indeed, robustness can be handled at different levels since there are several sources of uncertainties in the decision process. Stewart (2005) identifies two potential sources of uncertainties in MCDA:

1. external uncertainties: they correspond to the uncertainty of the consequences for a particular choice: for instance the decision to build an airport at a certain place may create new job opportunities in this area;

2. internal uncertainties: they reflect the lack of knowledge and the instability of the DM which are reflected in the judgement that he/she provides.

All these uncertainties can be treated through a sensitivity analysis after applying a given MCDA methodology.

In recent years, robust ordinal regression (ROR) has gained interest in MCDA. Greco et al. (2010b) developed the concept ROR for disaggregation procedures. Usually disaggregation procedures provide only the parameters of one model that is compatible with the preferences of the DM provided as input. However there is in general more than one set of parameters that allow to model the preferences of the DM. ROR aims at using all the sets of parameters that are compatible with the DM's preferences in order to produce robust recommendations. The methodology has been applied to several MCDA methods.

ROR was first applied to UTA methods by Greco et al. (2008). The method has been called UTA$^{\text{GMS}}$. Unlike UTA, which considers only one set of additive value functions compatible with DM's preferences, UTA$^{\text{GMS}}$ provides the set of additive value functions that are compatible with the preferences of the DM. As in UTA, UTA$^{\text{GMS}}$ takes as input a set of pairwise comparisons of a set of reference alternatives $A^* \subseteq X$ and uses linear programming. The outcome of the UTA$^{\text{GMS}}$ procedure are two rankings of the alternatives A, namely a necessary and a possible ranking. For any pair $a, b \in A$:

1. in the necessary ranking, a is ranked at least as good as b if $U(a) \geq U(b)$ for all value functions compatible with the preferences of the DM;

2. in the possible ranking, a is ranked at least as good as b if $U(a) \geq U(b)$ for at least one value function compatible with the preferences of the DM.

An extension of UTA$^{\text{GMS}}$ called generalized regression with intensities of preference (GRIP) has been proposed later by Figueira et al. (2009).

ROR has been further extended to sorting problems by Greco et al. (2010a). They created a new method called UTADIS$^{\text{GMS}}$ derived from UTADIS. In this method, it is supposed that the DM provides imprecise assignments in the form of intervals. For instance, the DM says that an alternative a is assigned between category C^1 and C^3. UTADIS$^{\text{GMS}}$ uses linear programming in order to compute a set of additive value functions compatible with the preferences of the DM. The method computes for each alternative $a \in A$ two sets of assignments:

1. the set of possible assignments which corresponds to the list of categories to which a is assigned by at least one of the value functions;

2. the set of necessary assignments which corresponds to the list of categories to which a is assigned by all the value functions.

Angilella et al. (2010) introduced non-additive robust ordinal regression. This variant takes criteria interactions into account by using the Choquet integral. Later Greco et al. (2011) applied ROR in the context of outranking relations. Recently, the concept of ROR has also been brought to the machine learning field by Corrente et al. (2013). It has been extended to DRSA by Słowiński et al. (2014).

2.4.6 Handling inconsistencies

When a model is learned based on assignment examples, it may happen that the model is incompatible with some preference statements of the decision maker. Inconsistencies can be due to several reasons.

One of the possible root cause of incompatible assignments is that the model is not well-suited to the problem. In that case, the model chosen to represent the DM's preferences has to be reconsidered. Some research papers deal with the selection of the MCDA model (see e.g. Ozernoy, 1987, 1992).

Another case causing inconsistencies occurs when the DM's preferences are not firmly established in his/her mind. This results in inconsistencies in preference statements. For instance, in the case of a sorting problem, these inconsistencies arise when the DM classifies an alternative a dominated by another one b in a better category than b. In MCDA, some works have been dedicated to the detection of inconsistencies with respect to a specific model. Some methods designed to fix these inconsistencies have been proposed, e.g. MACBETH (Bana e Costa and Vansnick, 1994).

Mousseau et al. (2003) proposed two different methods to cope with inconsistencies with respect to a MCDA model. They assume that the information provided by the DM are imprecise and represented as constraints on the parameters of a MCDA model. An interactive process is considered in which the DM starts the first iteration with very limited information and adds successively new constraints to the model. The process stops when the DM is satisfied with the model that has been learned. During the process, it can happen that the addition of a constraint reduces the space of solutions to the empty set. At this moment, it is interesting to identify the subsets of constraints that make the problem infeasible. That is the purpose of the two methods proposed by Mousseau et al. (2003). They consist respectively in a MIP and a LP that identify the list of minimal subsets of constraints that induce inconsistencies with respect to the model. In the paper, the algorithms are tested in the context of the elicitation of the parameters of an ELECTRE TRI model.

An extension of the algorithms proposed by Mousseau et al. (2003) has been developed by Mousseau et al. (2006). Instead of removing constraints, the approach proposed by Mousseau et al. (2006) consists in relaxing the constraints. In practice, when learning the parameters of a sorting problem, it amounts to enlarging the list of possible assignments for an alternative.

2.5 Preference learning

In this section, we present the main concepts in preference learning (PL). We recall the type of problems treated in PL and the indicators that allow to assess the quality of a solution obtained with a PL algorithm.

2.5.1 Purpose of preference learning

Preference learning is a subfield of machine learning (ML). In PL, predictive models are learned on the basis of observed preference information. Preference learning algorithms are conceived in order to handle large data sets. The problems treated in this field might involve several hundreds of instances. The methods and algorithms used in PL usually rely on strong statistical foundations.

There exist various types of preference learning problems. Fürnkranz and Hüllermeier (2010) differentiate problems along several dimensions. We describe these dimension below.

Representation of the preferences Preferences can be represented in different ways. One can split these preferences in two families: absolute and relative preferences. The former consists in assessing alternatives while the latter consists in comparing alternatives with each other. Two kinds of absolute preferences are distinguished:

- binary preference: an alternative is considered either as "good" or "bad";
- value function: a score is assigned to each alternative, representing its degree of preference; the score can be either numeric or ordinal.

Different types of relative preference exist. One distinguishes two classes:

- total order: an object can be compared to any other alternative of the set;
- partial order: some objects are incomparable.

Type of objects The type of alternatives that are treated in preference learning can be presented in different forms: identifier, feature vector, structured objects, ...

Type of training input The training input can be constituted of relative or absolute preferences. As an example, the training input can be a total order. Moreover the training input can be complete or incomplete. For instance some feature vectors might be incomplete.

2.5.2 Supervised learning problems with distinguished input and output spaces

Hüllermeier (2014) distinguishes supervised learning problems in which the input and output spaces are clearly distinguished from the others. In this type of problems, the input instances are mapped to preference models. The input

space of the problem usually consists of a set of instances associated to a feature vector. We denote it by X and $x = (x_1, x_2, \ldots, x_m) \in X$ represents a vector of features. In these problems, the output space is usually complex and structured. The output space can be, for instance, a set of permutations of labels, a set of combinations of labels. We describe in the next paragraphs some supervised learning problems as a function of the representation of the output space.

Multi-label classification

Multi-label classification has been introduced by Tsoumakas and Katakis (2007). Consider a set of labels denoted by $\mathcal{L} = \{\theta_1, \theta_2, \ldots, \theta_k\}$. There exists no natural order of the labels. Multi-label classification aims at assigning a subset of relevant labels $\mathcal{L}_x \subseteq \mathcal{L}$ to each instance $x \in X$. The output space $S_\mathcal{L}$ consists in the power set of the labels. The training instances given as input to the algorithm is usually of the form $(x, \mathcal{L}_x)$, where x is a feature vector and $\mathcal{L}_x$ represents a subset of labels associated with the instance x.

Example 5. *As an example of a multi-label classification problem, consider a set of labels $\mathcal{L}$ representing different types of activities: hiking, tennis, football and basket. The input space consists of a set of students who have to select a list of sports they like. The set of students constitutes the input space X of the problem. For a student x, a possible subset of activities that he/she likes $\mathcal{L}_x$ might be [football, basket]. The multi-label classification algorithm takes as input a list of tuples constituted of a student and the list of sports he/she likes, e.g. $\{(John, [football, basket]); (Marc, [hiking]); (Peter, [tennis, basket])\}$.*

Multi-label ranking

Multi-label ranking is a variation on multi-label classification. The training input given to the algorithm remains the same as for multi-label classification. In this type of problems, the predicted output of the algorithm is a ranking of the labels for an instance x denoted by π_x. The output space $S_\mathcal{L}$ of the learned function consists of the set of permutations of all the labels.

Example 6. *Applied to Example 5 introduced above, a multi-label classification algorithm will compute a ranking over the labels for each alternative. For instance, a possible prediction of the algorithm for a student x is football $\succ$ basket $\succ$ tennis $\succ$ hiking.*

Graded multi-label classification

Graded multi-label classification has been introduced by Cheng et al. (2010b). This variant of multi-label classification consists of assigning a grade to each

label. The grade represents the degree of membership of the label. A high grade corresponds to a strong membership of the label to the subset $\mathcal{L}_x$. The degree of membership of a label θ in a set $\mathcal{L}_x$ is denoted by $\mathcal{L}_x(\theta)$. Graded multi-label classification algorithms learn a function f_x that assigns a grade to each label in $\mathcal{L}_x$ for each instance $x \in X$.

Example 7. *Applied to Example 5 introduced in the paragraph about multi-label classification, it consists of assigning a grade to each label of the set $\mathcal{L}$. If the score is a number, ranging between 1 and 3, a possible output for a student x can be $f_x(football) = 2, f_x(basket) = 3, f_x(tennis) = 1, f_x(hiking) = 2$. This implies that he/she prefers basket over football and hiking which are tied in his preferences. Tennis is the least preferred sport for this student.*

Graded multi-label ranking

This slight variation of graded multi-label classification consists of predicting a ranking over the labels for each instance $x \in X$. The difference lies in the input given to the algorithm. In multi-label ranking, a subset of labels is associated to each instance. In graded multi-label ranking a score is associated to each label for each instance. The output remains the same as in multi-label ranking.

Label ranking

This family of problems has been introduced by Hüllermeier et al. (2008). Label ranking problems take as input a set of training instances which are usually represented by a feature vector. A set of labels $\mathcal{L}$ is defined. Each learning instance is associated to a set of pairwise preferences among the labels. For an instance x, the pairwise preferences are of the form $\theta^{(i)} \succ_x \theta^{(j)}$, with $\theta^{(i)}, \theta^{(j)} \in \mathcal{L}$. Preferences over labels are expressed in a relative way. For an instance x, all pairs of labels are not necessarily compared. A label ranking algorithm tries to learn a function that maps each instance of the set X to a ranking $\succ_x$ of $\mathcal{L}$.

2.5.3 Other learning problems

Other learning problems include problems in which there is no clear distinction between the input and output spaces. We describe these type of problems in the next paragraphs.

Instance ranking

Instance ranking consists in assigning a label $\theta^{(i)} \in \mathcal{L}$ to an instance $x \in X$. The labels in $\mathcal{L}$ have a natural order such that $\theta^{(k)} \succ \theta^{(k-1)} \succ \ldots \succ \theta^{(2)} \succ \theta^{(1)}$. The training input given to the algorithm consists in a set of instances coupled

to a label, i.e. pairs $(x, \theta^{(i)})$. The goal is to learn a function predicting the score of an alternative based on its performances on the different attributes. Instance ranking problems are identical to sorting problems in MCDA.

Object ranking

Object ranking consists in finding a ranking function f that allows to rank a finite set of objects A. A typical ranking function associates a score to each alternative and then sorts the alternatives by score. Algorithms in this field take as input a set of pairwise preferences on the objects in the form $a \succ b$, with $a, b \in A^*$, with $A^* \subseteq X$. The performances of objects $a, b \in A^*$ are usually also given as input to the algorithm. A ranking function is learned based on this information and then used to predict the ranking of other objects. Object ranking problems are similar to ranking problems in MCDA.

Collaborative filtering

Collaborative filtering consists of using the information provided by a group of users over a finite set of objects in order to determine the preferences of other users for these objects. This family of problems has been introduced by Goldberg et al. (1992). The training instances given to the algorithm consists in incomplete score vectors over a set of objects. Collaborative filtering algorithms try to predict the scores that are not known for an object.

Dyadic prediction

Dyadic prediction has been introduced by Menon and Elkan (2010). It is a more general form of collaborative filtering. In such type of problems, the input given to the algorithm consists in pairs, called dyads, associated to a label. Each instance in the pair is evaluated on a set of attributes which are not necessarily the same for both instances. The goal of the algorithm is to predict a label for other pairs.

2.5.4 Loss functions and model evaluation methods

In this part, we review the list of loss functions used to assess assignments obtained using sorting models.

We consider a set of m alternatives A evaluated on n criteria. These alternatives are naturally assigned to one of the categories selected among a set of p ordered categories C going from C^1 to C^p, such that $C^p \succ C^{p-1} \succ \ldots \succ C^1$. Let $s_M : A \to C$. We denote by $s_M(a)$ the assignment of the alternative a obtained

with model M. Similarly, let $s_{M'} : A \to C$, we denote by $s_{M'}(a)$ the assignment of the alternative a obtained through model M'.

Usually, the assignment obtained with a model M' is compared to the "ground truth" represented by model M. It is the case in the measures defined below.

Classification accuracy, error rate

The classification accuracy (CA) indicates the proportion of alternatives that are correctly assigned by model M. Formally, it is defined as:

$$CA = \frac{|a \in A : s_M(a) = s_{M'}(a)|}{|a \in A|}. \tag{2.28}$$

The error rate indicates the ratio of alternatives that are incorrectly assigned by the model M. It corresponds to the complement of the CA and it is defined as:

$$ER = 1 - CA = \frac{|a \in A : s_M(a) \neq s_{M'}(a)|}{|a \in A|}. \tag{2.29}$$

This quality index is the most used when assessing the quality of a classifier. However it does not give a good indication about the discriminating power of the classifier. We illustrate this through an example.

Example 8. *For instance, consider a set of alternatives assigned either to category C^1 or C^2. In this set, 99% of the alternatives are assigned to C^1 according to the model M. If a model M' assigns all the alternatives to C^1, then CA is equal to 99%. This value can be misleading since the classifier aims to be efficient while it is not since it cannot discriminate the 99% alternatives that belong to C^1 from the ones that belong to C^2.*

Confusion matrix

A confusion matrix is a table in which each row contains the number of alternatives that are actually assigned to a given class according to the ground truth and each column contains the number of alternatives that are assigned by the model to a given class.

The schema of a confusion table for a problem involving two classes $C^2 \succ C^1$ is given in Table 2.3. The class C^1 is considered as the "negative" class while C^2 is considered as the "positive" class. The notion of confusion matrix can easily be extended to problems involving more than 2 categories.

The first cell in the table contains the number of true negative instances, i.e. the number of instances that have been correctly classified in C^1 by the model. Similarly the bottom right cell shows the number of alternatives correctly

Table 2.3: Confusion matrix.

Ground truth \ Predicted class	C^1	C^2
C^1	True negative	False positive
C^2	False negative	True positive

classified in C^2 by the model. The top right and bottom left respectively indicate the number of alternatives that have been wrongly assigned by the model to C^2 and C^1.

Tables of confusion allow to observe the efficiency of the classifier to discriminate alternatives from different classes. It shows whether the classifier makes big errors, i.e. classifies alternatives far from their ground truth category. This is even clearer when the number of categories p is large, in particular larger than 2. The table also indicates if the classifier is neutral, pessimistic or optimistic, i.e. if incorrect assignments are fairly spread between categories or if the algorithm assigns more likely to worse or better categories.

From a confusion matrix it is possible to compute several indices reflecting the precision and efficiency of a classifier. The formula to compute some of these indices are given below:

$$\text{True positive rate} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}}, \tag{2.30}$$

$$\text{False positive rate} = \frac{\text{False positive}}{\text{False positive} + \text{True negative}}, \tag{2.31}$$

$$\text{Precision} = \frac{\text{True positive}}{\text{True positive} + \text{False positive}}. \tag{2.32}$$

Confusion tables offer a lot of information. However it requires some analysis in order to be able to assess the efficiency of a classifier since it does not deliver a single index contrary to CA.

Receiving operating characteristic

The receiving operating characteristic (ROC) curve (Fawcett, 2006) provides a measure of the efficiency of a binary classifier to discriminate alternatives.

An example of ROC graph is displayed in Figure 2.7. The abscissa and ordinate values correspond respectively to the false positive rate (Equation (2.31)) and to the true positive rate (Equation (2.30)). Each classifier corresponds to a point in this graph. The more to the North West of the graph the point is

located, the more efficient the classifier is at discriminating alternatives from the lower and upper classes. For instance, in Figure 2.7, the classifier M' is more efficient than M since its true positive rate is better. It reports also less false positive.

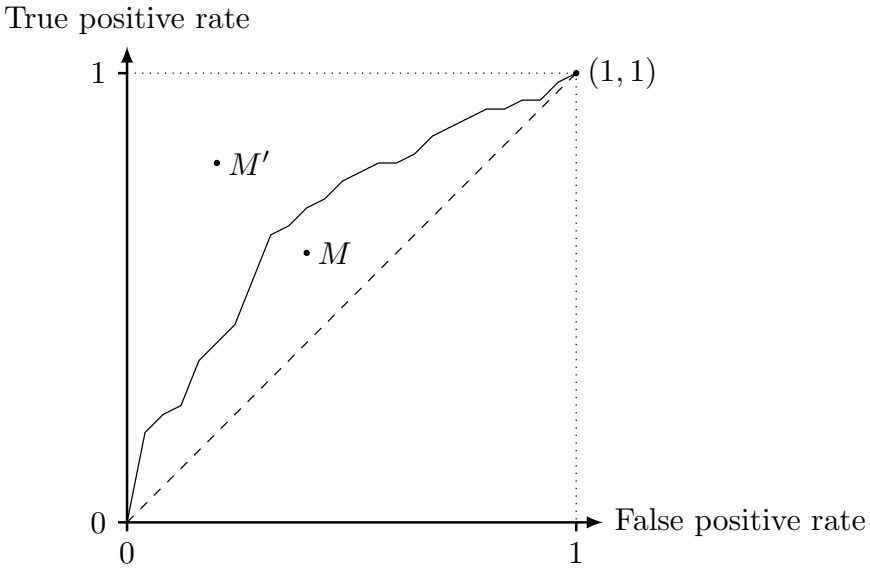


Figure 2.7: Receiving operating characteristic (ROC) curve.

When the classifier computes a score and uses a threshold to discriminate alternatives between two classes (e.g. logistic regression, see below), it is possible to plot a curve as shown on Figure 2.7 by varying the threshold value. Fawcett (2006) proposes an efficient method to plot the ROC curve of a classifier. First it consists in sorting the alternatives by descending value of their score. As the score of an alternative does not change, the assignment of this alternative remains monotone. Indeed, an alternative cannot be assigned to a worse category if the threshold decreases. Then the value of the threshold is increased progressively and the true positive and false positive rates are updated accordingly.

We remark that the ROC curve shows the performance of a classifier independently of the distribution in classes. Indeed, a variation of the number of positive instances does not affect the ROC curve since it is respectively the true positive and false positive rate that are taken into account.

Area under the curve

The area under the curve (AUC) corresponds to the area under a ROC curve. It represents the probability that a classifier will rank a randomly chosen positive

instance higher than a randomly chosen negative one. The value of the AUC is comprised between 0 and 1.

The value of the AUC can be computed through different estimators. Faraggi and Reiser (2002) made an empirical study of these estimators. In our experiments, we use the Mann-Whitney approach which provides an unbiased non-parametric estimator for computing the AUC. For a binary classifier, sorting alternatives in two classes C^2 and C^1, with $C^2 \succ C^1$, the AUC can be expressed as follows:

$$AUC = \frac{1}{|A^1| \cdot |A^2|} \sum_{a^i \in A^2} \sum_{a^k \in A^1} I_{f(a^i) > f(a^k)} \tag{2.33}$$

where A^1 (resp. A^2) denotes the set of alternatives classified in C^1 (resp. C^2) according to the ground truth, f is a classifier function and $I_{f(a_i)>f(a_k)}$ is an indicator function that is equal to 1 when $f(a_i) > f(a_k)$ is true and equal to 0 otherwise.

Waegeman et al. (2008) generalized this formulation of the AUC for p categories, with $C^p \succ C^{p-1} \succ \ldots \succ C^1$. The approach proceeds by dichotomy. It consists in building successively a set of twofold partitions, $(C^{\leq h}, C^{>h})$ for h going from 1 to $p-1$. In each twofold partition, the first set $C^{\leq h}$ denotes the set of categories below and including C^h. The second set $C^{>h}$ denotes the set of categories above C^h. Waegeman et al. (2008) compute an AUC for each twofold partition, this AUC is denoted by AUC^h and is computed as follows:

$$AUC^h = \frac{1}{|A^{\leq h}| \cdot |A^{>h}|} \sum_{a^i \in A^{>h}} \sum_{a^k \in A^{\leq h}} I_{f(a^i) > f(a^k)}$$

where $A^{\leq h}$ (resp. $A^{>h}$) denotes the set of alternatives classified below and in C^h (resp. above C^h) according to the ground truth, f is a classifier function and $I_{f(a_i)>f(a_k)}$ is an indicator function that is equal to 1 when $f(a_i) > f(a_k)$ is true and equal to 0 otherwise. The global AUC value is computed by averaging the $p-1$ AUC^h values as follows:

$$AUC = \frac{1}{p-1} \sum_{h=1}^{p-1} AUC^h. \tag{2.34}$$

2.6 Monotone learning in a sorting context

In this thesis, we focus mainly on sorting problems in which the monotonicity of attributes is guaranteed. In this part of the chapter, we describe some algorithms dedicated to sorting problems.

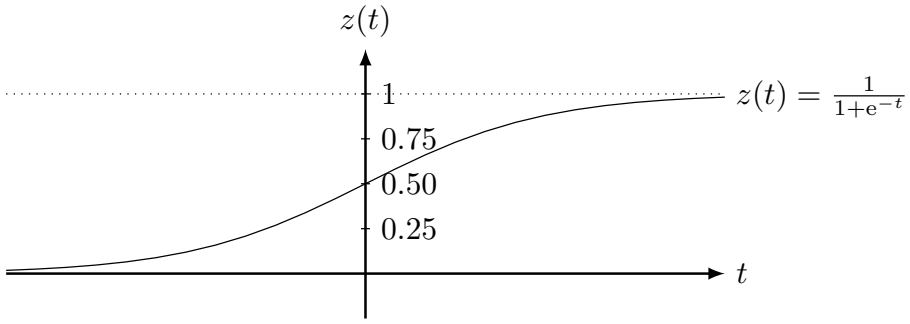


Figure 2.8: Logistic function.

2.6.1 Logistic regression

Logistic regression (Cox, 1958) is used in statistics to determine the probability of an outcome based on input variables, also called predictive variables. The input variables are either continuous or binary. Depending on the output variable, logistic regression can be binomial, ordinal or multinomial. In binomial or binary logistic regression, the output variable has only two possible values (e.g. "win" or "loss"). In multinomial logistic regression, the output variable can have more than 2 values that are not ordered (e.g. "car", "bus" or "train"). In ordinal logistic regression, the output variables are ordered (e.g. "good", "medium" or "bad"). Binary ordinal logistic regression is a particular case of ordinal regression where the output is binary.

Below we recall what a logistic function is and then we describe how the logistic regression is used in classification.

Logistic function

Logistic regression models are using logistic functions. A logistic function can take any real value as input and always outputs a value between 0 and 1. The logistic function $z(t)$ is the function having the following form:

$$z(t) = \frac{1}{1 + e^{-t}} \tag{2.35}$$

where variable t is usually expressed as a linear function of a linear combination of explanatory variables $x_1, x_2, \ldots, x_n$:

$$t = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n. \tag{2.36}$$

An example of such a function is displayed in Figure (2.8). By replacing t in

Equation (2.35), one can express the logistic function as follows:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta_0 - \theta_1 x_1 - \theta_2 x_2 - \ldots - \theta_n x_n}}.$$

The value of this function can be interpreted as the probability of an outcome y for a given value of the input variables $x_1, x_2, \ldots, x_n$. We denote by $P(y = 1|x)$ the probability of having $y = 1$ associated to the input vector x. The probability of having y equal to zero is denoted by $P(y = 0|x)$ and is the complement of $P(y = 1|x)$, i.e. $P(y = 0|x) = 1 - P(y = 1|x)$.

Using logistic regression for classification

The logistic regression is used in the context of preference learning where the monotonicity of the attributes is guaranteed. One way to use this model is to consider that an alternative x is classified in class $y = 1$ if the probability $P(y = 1|x)$ is greater than 0.5, otherwise it is classified in $y = 0$.

In such a model, the parameters that have to be determined are the values of θ_0 to θ_n. Learning these parameters can be done based on a learning set composed of pairs (x, y) in which x is a vector of performances and y the binary output value associated to x. Inferring the values of $\theta_0, \theta_1, \ldots, \theta_n$ implies the definition of a cost function. In logistic regression, the cost function used is given in the following equation:

$$Cost(h_\theta(x), y) = -y \cdot \log(h_\theta(x)) - (1 - y) \cdot \log(1 - h_\theta(x)).$$

The value of the cost function is equal to $-\log(h_\theta(x))$ when the value of y is equal to 1. Otherwise, if $y = 0$, the value of $Cost(h_\theta(x), y)$ is equal to $\log(1 - h_\theta(x))$. It means that the cost is huge when the value of $y = 1$ (resp. $y = 0$) if the value of $h_\theta(x)$ is close to 0 (resp. 1). On the contrary, the cost value is small for $y = 1$ (resp. $y = 0$) if $h_\theta(x)$ is close to 1 (resp. 0). This cost function is convex which means that a standard gradient descent method can be applied for finding $\theta_0, \theta_1, \ldots, \theta_n$. For a learning set composed of m pairs $(x^{(i)}, y^{(i)})$, with $i = 1, \ldots, m$, the objective is:

$$\min_\theta -\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})). \tag{2.37}$$

For problems in which more than two classes are involved, a logistic regression is applied for each class. Finally, the alternative is assigned to the class in which the value of the associated logistic function has the highest value.

2.6.2 Choquistic regression

In logistic regression, the logistic function (2.35) does not take attribute interactions into account. Tehrani et al. (2012) proposed to modify the logistic function in order to take attribute interactions into account. To do so, they advocate the use of the Choquet integral through a new model called "Choquistic model". Before, the Choquet integral has been widely studied in the domain of multiple-criteria decision analysis (Grabisch and Roubens, 2000; Grabisch and Labreuche, 2010) but not in machine learning.

The Choquistic model consists in replacing the linear function given in Equation (2.36) by a Choquet integral. The Choquet integral of a vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is defined as:

$$C_\mu(x) = \sum_{j=1}^{n} [x_{\tau(j)} - x_{\tau(j-1)}] \cdot \mu(\{\tau(i), \ldots, \tau(n)\}), \tag{2.38}$$

where τ is a permutation of the indices $\{1, \ldots, n\}$ such that $0 \leq x_{\tau(1)} \leq x_{\tau(2)} \leq \ldots \leq x_{\tau(n)}$ and $\mu(\{\tau(i), \ldots, \tau(n)\})$ is the capacity of the subset of criteria $\{\tau(i), \ldots, \tau(n)\}$ as described in Definition 4.

In the Choquistic regression, the variable t in logistic function (2.35) is replaced as follows:

$$h_{C_\mu}(x) = \frac{1}{1 + e^{-\gamma(C_\mu(x) - \beta)}}, \tag{2.39}$$

where $\gamma, \beta \in \mathbb{R}$ are two constants. The value of β defines the threshold separating the two categories of the model. If the value of $C_\mu(x)$ is greater than β then the alternative is classified in the "good" category, otherwise it is classified in the "bad" category.

Tehrani et al. (2012) propose a manner to elicit the parameters of this model through the standard gradient-based optimization methods with constraints. This paper deals with problems in which only two categories are considered. Tehrani et al. (2011) also developed the Choquistic regression for problems involving more than two categories.

2.6.3 Decision tree based algorithms for ordinal classification

Potharst and Bioch (1999) developed an algorithm which generates a binary decision tree that enables to classify alternatives in ordered categories. In a binary decision tree, the data set is split in two disjoint sets at each node by testing one of the attributes against a threshold value. The test done at each node on each alternative $a \in A^*$ has the following form: $a_j \leq \beta$ for some $\beta \in X_j$.

An example of a binary decision tree is shown in Figure 2.9. In this tree, three criteria and three categories, with $C^3 \succ C^2 \succ C^1$, are involved. We

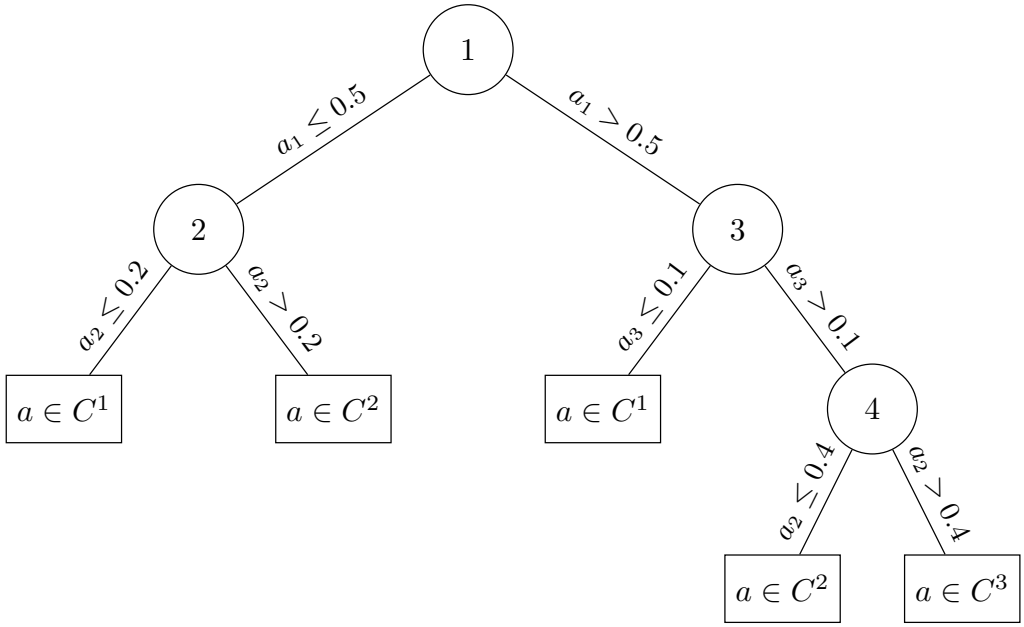


Figure 2.9: Binary decision tree.

describe briefly the assignment procedure. The first node splits the alternatives space in two disjoint subsets on the basis of the value on the first criterion: the first subset (node 2) contains alternatives of X that have a score lower than 0.5 on criterion 1 and the second subset (node 3) is the complement. In node 2, the alternatives space is split again in two disjoint subsets: the first one contains alternatives that have a score smaller than or equal to 0.1, the second one contains the alternatives having a score greater than 0.1. Alternatives of the first subset are then assigned to category C^1 and the ones in the second subset are assigned to C^2. Similarly node 3 and node 4 split the space by discriminating the alternatives on, respectively, the third and fourth criteria.

An advantage of decision trees over some other preference learning algorithms is that it is easy to interpret the assignment rule resulting from the decision tree. Indeed, the assignment rule of a decision tree can be described as a disjunction of conjunctions. For instance, for the tree shown on Figure 2.9, an alternative a is assigned to category C^1 if the following condition is fulfilled:

$$a \in C^1 \iff [(a_1 \leq 0.5) \wedge (a_2 \leq 0.2)] \vee [(a_1 > 0.5) \wedge (a_3 \leq 0.1)].$$

Potharst and Bioch (1999) described an algorithm that allows to induce such a binary decision tree based on an input data set. The data set A^* that is given as input to the algorithm has to be monotone, i.e. for all $a, b \in A^*$ such that $a_j \geq b_j$ for every $j \in N$ we have $C(a) \geq C(b)$. The algorithm splits recursively

the input data sets A^* and ensures that the monotonicity of the assignment rules holds. Algorithm 1 shows the structure of the algorithm.

Algorithm 1 Decision tree algorithm for ordinal classification.

```
function GenerateTree(T, A*)
    Update(T, A*)
    if Stop(T) then
        AssignCategory(T)
    else
        (T_l, T_r) = Split(T, A*)
        GenerateTree(T_l, A*)
        GenerateTree(T_r, A*)
    end if
end function
```

Let x, y be two instance vectors of X, we have $x \leq y$ if and only if $x_j \leq y_j$ for all $j \in N$. Similarly, we have $x < y$ if and only if $x_j \leq y_j$ for all $j \in N$ and $x_j < y_j$ on a least one $j \in N$. In Algorithm 1, T represents a subset of the alternatives space X: $T = \{x \in X : t^- < x \leq t^+\}$, with t^- and t^+, two instance vectors of X such that $t_j^- \leq t_j^+$ for all $j \in N$. The algorithm is composed of 4 main parts:

1. The update function which ensures the monotonicity. It consists in adding the worst possible and the best possible instance vectors of T in the data set if it is not already in it.

2. The stopping rule which determines when to stop splitting the alternatives space. The splitting stops once the worst and the best instances of T are assigned to the same category.

3. The assignment rule which assigns a category to a leaf.

4. The split rule which divides the data set in two disjoint subsets based on the score on one criterion j.

Later, Potharst and Feelders (2002) proposed a variant of the algorithm which is able to handle non-monotone data sets.

van de Kamp et al. (2009) presented a new algorithm learning monotone binary decision trees. The algorithm proposed in this paper can be combined with any standard classification tree algorithm. It takes as input a tree and returns a monotone decision tree as output. In order to ensure the monotonicity of the tree, the algorithm uses antitonic regression. It was shown that the isotonic classification tree algorithm performs better than standard trees on real data sets.

Giove et al. (2002) proposed another type of binary decision tree. Instead of assigning the object to a particular class, the method consists in assigning the object to a lower or upper class range with a probability. At each node a test of the type $a_j \geq \beta$, with β a constant, is made on an attribute j. The first subnode contains the alternatives passing the test and the second the other ones. In the first subnode, the alternatives that do not pass the test are assigned with a probability index to all the downward unions of categories, i.e. $C^{\leq h}$, with $h = \{1, \ldots, p\}$. Similarly, the alternatives of the second node are assigned with a probability index to all the unions of upward categories $C^{\geq h}$, with $h = \{1, \ldots, p\}$. The advantage of this algorithm is that it does not require any hypothesis on the input data set, i.e. the attributes should be necessarily monotone.

2.7 Links and differences between multiple-criteria decision analysis and preference learning

This section describes major links and differences between preference learning and multiple-criteria decision analysis. A list of these differences and links has been emphasized in Hüllermeier (2014) and in Corrente et al. (2013).

2.7.1 Size of the problems

Problems in PL often involve several hundreds of alternatives both in the learning and test sets. For instance, the data sets used in Duivesteijn and Feelders (2008) contain from 200 to about 1000 alternatives. In monotone learning, a subfield of PL, this size of problem is also common. Other papers deal with applications based on data sets containing several thousands of alternatives (Cheng et al., 2009, 2010a) or even with several millions of alternatives (Bache and Lichman, 2013).

In MCDA, the decision problems are generally small. Datasets studied in this field usually contain few alternatives. As an example, in Norese and Carbone (2014), ELECTRE TRI has been used to assess the marginal and overall activation of 21 Italian airports in reaction to a crisis. Another example is Figueira et al. (2011) who used ELECTRE TRI-C in the context of assisted reproduction. In this application, the data sets consisted of 51 couples.

Problems in PL usually involve a lot of attributes. For instance in Hüllermeier et al. (2008), more than 20 attributes were used in the context of label ranking. In MCDA, the number of criteria is usually more limited, but much more engineered, i.e. the semantic of the criteria is very well known and the definition of the criteria is better established than in PL. For instance in Metchebon Takougang (2010), the criteria chosen to study the deterioration of the landscape in the Loulouka

watershed have been studied in depth and chosen in accordance with the nature of the decision problem.

In PL, the lack of interpretability of the models is usually balanced by the data set size. The inferred models rely on a larger set of information than in MCDA.

2.7.2 Interpretability and accuracy

In PL, the objective is to determine the parameters of an estimator that reflects as well as possible the ground truth, no matter the complexity of the underlying model. There are weak assumptions about the models used. The emphasis is put on the performances of the algorithm. In sorting algorithms, the emphasis is put on the outcome of the algorithm in terms of classification accuracy, area under the curve, etc. For instance, in Daniels and Kamp (1999); Duivesteijn and Feelders (2008) the emphasis is put on the performance of the algorithms in terms of error rate.

Usually, in PL, less importance is given to the parameters of the estimator than in MCDA. Indeed, the aim of PL algorithms is to maximize an objective function that is clearly defined, not to find a model that can be explained to a DM. The models learned in PL are generally more difficult to interpret. However it is not always the case. For instance Potharst and Bioch (1999) developed an algorithm that infers a decision tree. Such a type of model is easily interpretable by a DM.

In MCDA, the emphasis is put on the interpretability of the model. It is usually preferred to have a model that is understandable and interpretable than one that is better in terms of classification accuracy but difficult to interpret. When an inference procedure is used to learn the parameters of a MCDA model, generally, the learned parameters are presented to the DM. The model is also explained to the DM and, if needed, the analyst asks more questions to the DM in order to refine or modify the learned model. Such a type of approach is used in the algorithms presented in Mousseau et al. (2001, 2003). It works as follows. The DM defines some assignment examples, linear programming techniques are then used to learn the parameters of the MCDA model. The parameters learned by the algorithm are then presented to the DM and he/she decides whether or not some assignment examples have to be modified or if other constraints should be added. Then, if needed, the LP or MIP is run again until the DM is satisfied with the model.

In MCDA, the models used are well-founded. These models have strong interpretation skills and the type of preferences that can be represented by a model are well-known and studied. For instance, Bouyssou and Marchant (2007a,b) studied the type of preferences that can be represented by a NCS model. Other

papers deal with the type of preferences that can be represented more generally with outranking methods (Bouyssou and Pirlot, 2005, 2007).

2.7.3 Monotonicity of the attributes

In PL it is not always assumed that attributes are monotonic. Most algorithms in PL have been developed without any assumption about the monotonicity of the attributes (see e.g. Chandrasekaran et al., 2005). This makes these algorithms more flexible, i.e. able to deal with more data sets but it also makes them generally less interpretable and convincing if the attribute scale of the problem is monotone.

In MCDA, attributes are generally used as criteria. By definition it implies the monotonicity of the preference scale.

In PL, the field regrouping algorithms dealing with monotonic data sets is called *Monotone Learning*. In the last years, some papers gave more importance to this particular field in PL, see e.g. Potharst and Bioch (1999); Tehrani et al. (2011).

2.7.4 Learning methods

Algorithms designed to learn the parameters of MCDA models are not always conceived in order to deal with large data sets. In MCDA, a majority of the learning procedure relies on linear programming or mixed integer programming (see e.g. Zopounidis and Doumpos, 2000). Some MCDA learning algorithms become inefficient when the size of the problems increases. For instance in Leroy et al. (2011), a MIP has been proposed in order to infer the parameters of a MR-Sort model. The program becomes inefficient when the size of the model grows (number of criteria and categories) or when a lot of examples are given in input to the algorithm. However, in the last years, some efforts have been made in order to be able to handle larger problems.

In PL, the algorithms are usually based on non-linear and statistical methods. For instance in Tehrani et al. (2012), quadratic programming is used to determine the parameters of a model. These methods performs better than MIP for large problems.

2.7.5 Interaction with the decision maker

In MCDA, there is usually a strong interaction with the decision maker during the learning process. The inference of the model occurs by doing multiple iterations in interaction with one or several DMs. The determination of the model parameters is either done by eliciting them directly with the DM (direct elicitation) or by inferring the parameters from preference statements expressed by the DM (indirect elicitation). An example of simple indirect elicitation is given

in Simos (1990) where a procedure allowing to elicit the weights of a model by interacting with a DM has been presented. The procedure consists in asking the DM to rank a set of cards representing the criteria in order to determine empirically the weights of the criteria. Note that an improvement of this procedure has been proposed in Figueira and Roy (2002). In indirect elicitation procedures, the DM is often a participant in the elicitation procedure. If the learned model is not consistent or is incompatible with some DM's statements then he/she can revise some of the assignments he/she gave as input. Such a type of procedure is used for instance in Ngo The and Mousseau (2002). Another example is the interactive approach proposed in Dias et al. (2002) in order to obtain robust conclusions in interaction with a decision maker.

In PL, the preference information found in the data sets is considered as a ground truth. Even if inconsistencies are identified in the data sets, the information is not corrected in an interaction with a DM. The data sets are used as they are without any interaction with a DM. In PL, the efficiency of the algorithms are assessed by comparing the assignments, ranking, etc. to the ones of the learning data sets which are considered one of the ground truth.

2.7.6 Robustness

Since the early days of MCDA and up to now, importance has been granted to the robustness of the solutions found by the disaggregation algorithms (Vincke, 1999; Roy, 2010). In Greco et al. (2010b), robust ordinal regression has been introduced in order to provide the DM with a list of all the possible and necessary consequences derived from the DM's preference statements (and assuming a given model, for instance, an additive value function). This approach allows to have an idea of how the model is constrained by the DM's preference statements and gives the opportunity to propose several models as output.

In PL, less focus is put on the robustness of the model. PL algorithms generally provide only one solution to the problem and usually no sensitivity analysis is performed. However, note that ROR has been recently introduced in the machine learning field by Corrente et al. (2013).

Chapter 3

Learning a majority rule sorting model from large data sets

In this chapter, we present a metaheuristic designed for learning the parameters of a majority rule sorting (MR-Sort) model on the basis of large sets of assignment examples. The first part of the chapter describes the purpose of the metaheuristic and the strategy adopted in order to learn efficiently the parameters of a MR-Sort model. The second part of the chapter details the components of the metaheuristic and the different strategies considered for each component. Finally, the end of this chapter is devoted to experimental tests with artificial and real data sets.

3.1 Purpose of the metaheuristic

Learning the parameters of a MR-Sort model can be achieved in different ways. In this section, we describe why we chose to learn the parameters with a metaheuristic.

3.1.1 Handling large data sets

The aim of the metaheuristic described here is to learn the whole set of parameters of a MR-Sort model without veto on the basis of assignment examples. Examples given as input to the metaheuristic are vectors of performances for which an assignment is known. The metaheuristic should be able to handle large data sets similar to the ones used in the field of preference learning (PL). As mentioned in Chapter 2, PL data sets can involve several hundreds of examples.

Up to now, there is no algorithm that allows to learn the parameters of a MR-Sort model when large data sets are involved. In Chapter 2 we reminded

some algorithms devoted to learn globally or partially the parameters of such a model. None of them is able to deal efficiently with large data sets.

In Leroy et al. (2011), a linear program involving binary variables was used to learn the parameters of a MR-Sort model (without veto). The program tries to minimize the 0/1 loss, i.e. it searches for a model that is compatible with as many examples as possible. Learning the parameters of a MR-Sort model with linear programming requires the use of binary variables. The mixed integer program (MIP) proposed by Leroy et al. (2011) involves $m(2n+1)$ binary variables, where m is the size of the learning set and n the number of attributes. Experimental results showed that learning the parameters of a model for data sets involving a large number of assignment examples, criteria or categories, requires huge computing times with the MIP (using the IBM ILOG CPLEX solver). With barely 100 alternatives, 5 criteria and 3 categories (i.e. 1100 binary variables in the MIP), more than 100 seconds are needed to learn the parameters of a MR-Sort model[1]. Due to these long computing times, the MIP is not able to deal with problems involving large data sets.

3.1.2 Interpretability of the model

To our knowledge, none of the sorting algorithms used in the domain of preference learning are based on ELECTRE models. This type of models provides the advantage to be easily interpretable. The assignment obtained in output can be more easily explained with the value of the model parameters. As we explained in the previous chapter, this is an advantage when the output of preference learning algorithm has to be justified. For instance, in the medical sector, doctors may ask for justification for the assignment of a patient in a given category. With MR-Sort, the assignment of an alternative to a category can be explained with compact and intuitive rules.

3.2 Past researches and strategy for the elaboration of the metaheuristic

In Chapter 2, we presented past researches dedicated to the global or partial inference of MR-Sort parameters. Due to the long computing time required for learning the parameters of a MR-Sort with the MIP developed in Leroy et al. (2011), using this algorithm is not a feasible approach for the type of problems we want to handle, i.e. problems involving large data sets. An option to overcome the computing time issue is to use relaxation techniques that allow the obtainment of approximate solutions (Wolsey, 1998; Minoux, 2008). Instead of

[1]System used: Core 2 Duo P8700, running Gentoo Linux and CPLEX 12.5

exploring this path (which certainly deserves attention) we developed a new sophisticated - population based - metaheuristic which exploits as much as possible the specificities of the problem.

A metaheuristic presented by Doumpos et al. (2009) learns the parameters of an ELECTRE TRI model. Since MR-Sort is a light version of ELECTRE TRI which involves less parameters, it is possible to adapt this metaheuristic in order to learn the parameters of a MR-Sort model. However, the genetic algorithm described by Doumpos et al. (2009) uses standard mutation, cross-over and selection operators. In operation research, it is well-known that a metaheuristic adapted to the structure of the problem performs better (Pirlot, 1996).

In order to have an efficient algorithm which exploits as much as possible the structure of the problem, we try to take advantage of past research. Mousseau et al. (2001) showed that it was easy to learn the weights and majority threshold of a MR-Sort model with linear programming technique. It does not involve the use of integer variables, therefore the problem can be solved efficiently with the simplex algorithm. On the contrary, past studies have shown that learning the profiles of a MR-Sort model is not so easy. The MIP presented by Ngo The and Mousseau (2002) involves the use of binary variables in order to learn the profiles. It implies that the computing time becomes quickly prohibitive when the size of the problem increases.

In summary, we extract two pieces of information from past research:

- Given a set of profiles, learning the weights and the majority threshold of a MR-Sort model can easily be achieved using a linear program without binary variables.

- In contrast, given a set of weights and a majority threshold, learning the profiles values by means of linear programming requires using binary (0/1) variables.

Based on these two pieces of information, we develop a metaheuristic which is described in depth in the next section.

3.3 Description of the metaheuristic

In this section, we detail in depth the metaheuristic and its variants. We begin with the description of the overall structure of the metaheuristic. Then we describe its components in detail.

3.3.1 The metaheuristic

To properly take the structure of the problem into account, i.e. the ease of learning the weights and the majority threshold with a linear program and the difficulty to do the same for the profiles, we separate the algorithm in three components:

1. A heuristic which initializes a set of profiles;

2. A linear program learning the weights and the majority threshold of the model based on fixed profiles;

3. A heuristic adjusting the profiles to improve the quality of the model, while keeping the weights and majority threshold fixed.

The objective of the algorithm is to find a model restoring as many examples as possible. To assess the quality of the models, we use two indicators. The first is the classification accuracy (CA) criterion, which is defined by equation (2.28). The higher the value of the classification accuracy, the better the quality of the model. The second indicator is the area under the curve (AUC) defined by Equation (2.33), which quantifies the discriminating power of the algorithm to separate alternatives in different classes. Obviously, by optimizing successively the weights and threshold, then the profiles, again the weights and threshold and so on, instead of optimizing all parameters simultaneously, there is no guarantee that a very good solution will be reached, even though the process is iterated. In order to enhance the chances to converge towards a very good solution, we adopt an evolutionary approach evolving a population of N_{mod} MR-Sort models.

The general architecture of our algorithm is described as Algorithm 2. It shows how the three components are combined to find a MR-Sort model that restores as well as possible the assignment examples in the learning set.

Algorithm 2 Metaheuristic to learn all the parameters of a MR-Sort model

Generate a population of N_{mod} models with profiles set by an initializing heuristic
repeat
  for all model M of the set **do**
    Learn the weights and majority threshold with a linear program, using the current profiles
    Adjust the profiles with a heuristic, using the current weights and threshold; repeat N_{it} times.
  end for
  Reinitialize the $\left\lfloor \frac{N_{mod}}{2} \right\rfloor$ models giving the bottom values of CA or AUC.
until Stopping criterion is met

First a population of N_{mod} MR-Sort model is generated and, for each model, the set of profiles are initialized by a specific heuristic. After the initialization phase, for each model M, the algorithm solves a linear program to find the weights and the majority threshold with fixed profiles (obtained in the initialization step). Then, for each model M, on the basis of the weights and majority threshold learned in the previous step, the metaheuristic adjusts the profiles with a randomized heuristic in order to maximize the number of examples compatible with the model. The randomized heuristic alters the profiles N_{it} times for each model M, after which the set of profiles restoring the largest number of assignment examples is selected. This process results in a new population of N_{mod} models. These are ordered by decreasing order quality. The top half of the models are retained while the bottom half (precisely $\left\lfloor \frac{N_{mod}}{2} \right\rfloor$ models) are reset using the initializing heuristic.

The algorithm stops either after having run a given number of times, denoted by N_o (fixed a priori), or when it has found at least one model that restores correctly all the assignment examples. If no model restores correctly all the assignment examples, the best model is returned.

During our researches, we used two variants of this metaheuristic. Both are described in the next paragraphs.

Variant 1: maximization of the CA

The first variant of the metaheuristic consists in using the classification accuracy as quality criterion of a model M. It means that the selection of the models to reinitialize is done by comparing their CA. A model M is in that case considered better than another one M' if the CA of M is superior to the CA of M'. The best model returned by the metaheuristic is the one which has the highest CA.

Variant 2: maximization of the AUC

The second variant of the metaheuristic consists in using the AUC as quality criterion for a model M. In that case, the selection process is done by comparing the models with each other on the basis of their AUC. A model M is in that case considered better than another one M' if the AUC of M is superior to the AUC of M'. The model returned by the metaheuristic is the one having the highest AUC.

3.3.2 Profiles initialization

The first step of the algorithm consists in the initialization of a set of profiles for each of the N_{mod} models in the population. The general idea of the heuristic designed to set the value b_j^h of the profile b^h on criterion j is the following. This

value is chosen in order to maximize the discriminating power of each criterion, relatively to the alternatives in the learning set A. More precisely, we set b_j^h in such a way that alternatives ranked in the category above b^h (i.e. C^{h+1}) typically have an evaluation greater than b_j^h on criterion j and those ranked in the category below b^h (i.e. C^h) typically have an evaluation smaller than b_j^h.

In setting the initial profile values, we paid attention to the following aspects. Firstly, to guarantee an equal treatment to all profiles, we chose to consider only C^h and C^{h+1} to determine b^h. The reason for this option is to balance the number of categories above and below the profiles that are taken into account for determining this profile. For profiles b^1 and b^{p-1}, the only way to satisfy this requirement is to consider only one category above and one category below the profile.

The second issue is relative to the way the different categories are represented in the learning set. Consider the subsets A^{*h} and A^{*h+1} of alternatives in the learning set A that are assigned, respectively, to categories C^h and C^{h+1}. These subsets may be of quite different sizes. We weight the alternatives by using the relative frequencies of A^{*h} and A^{*h+1} in order to control the influence of categories that are under- or over-represented in the learning set.

The initializing heuristic is implemented as follows:

1. For each category C^h, compute the frequency π_h with which alternatives in the learning set are assigned to category C^h, i.e., $\pi_h = \frac{|A^{*h}|}{|A^*|}$.

2. For each criterion and each profile b^h, a set of candidate profile values are selected. They correspond to the performances of alternatives in A assigned to categories C^h and C^{h+1}. The value of the profile, b_j^h, is chosen randomly among the candidate values with some probability. The probability of each candidate value is proportional to its likelihood to classify correctly alternatives of categories C^h and C^{h+1} based on their performance on the sole criterion j. In order to balance the influence of A^{*h} and A^{*h+1}, which may be of quite different sizes, the examples are assigned a weight that is inversely proportional to the size of the class they belong to.

3. The profiles are computed in descending order, enforcing the constraint that values of the profiles on each criterion are ordered, i.e., we have $b_j^{h+1} \geq b_j^h$, for all criterion j and profile h.

3.3.3 Learning the weights and the majority threshold

Assuming that the profiles are given, learning the weights (w_j, for all $j \in N$) and the majority threshold (λ) of a MR-Sort model from assignment examples is done by means of solving a linear program. The MR-Sort model postulates that

the profiles dominate each other, i.e. $b_j^{h+1} \geq b_j^h$ for all h and j, and the inequality is strict for at least one j. The constraints derived from the assignments of the alternatives in the learning set are expressed as follows:

$$\begin{cases} \sum_{j:a_j \geq b_j^{h-1}} w_j - x_a + x'_a = \lambda & \forall a \in A^{*h}, h = \{2, \ldots, p\}, \\ \sum_{j:a_j \geq b_j^h} w_j + y_a - y'_a = \lambda - \varepsilon & \forall a \in A^{*h}, h = \{1, \ldots, p-1\}, \\ \sum_{j=1}^n w_j = 1, & \\ w_j \in [0;1] & j = 1, \ldots, n, \\ \lambda \in [0.5;1], & \\ x_a, y_a, x'_a, y'_a \in \mathbb{R}_0^+ & \forall a \in A^*. \end{cases} \tag{3.1}$$

The small positive number ε is used for transforming strict inequalities into non strict ones. There are as many 4-tuples of variables x_a, y_a, x'_a, y'_a as there are alternatives in the learning set A. The value of $x_a - x'_a$ (resp. $y_a - y'_a$) represents the difference between the sum of the weights of the criteria belonging to the coalition in favor of $a \in A^{*h}$ with regard to b^{h-1} (resp. b^h) and the majority threshold. If both $x_a - x'_a$ and $y_a - y'_a$ are positive, then the alternative a is assigned to the right category. In order to try to maximize the number of examples correctly assigned by the model, the objective function of the linear program minimizes the sum of x'_a and y'_a, i.e. the objective function is $\min \sum_{a \in A}(x'_a + y'_a)$. Note however that such an objective function does not guarantee that the maximal number of examples are correctly assigned. Failing to meet this goal may be due to possible compensatory effects between constraints, i.e., the program may favor a solution involving many small positive values of x'_a and y'_a over a solution involving large positive values of a few of these variables. Such a compensatory behavior could be avoided, but at the cost of introducing binary variables indicating each violation of the assignment constraints. We do not consider such formulations in order to keep computing times within reasonable limits.

3.3.4 Learning the profiles

Learning the profiles by using a mathematical programming formulation requires binary variables (Ngo The and Mousseau, 2002), leading to a MIP. In order to deal with problems involving large learning sets (e.g. 300 assignment examples, 10 criteria and 5 categories), MIP is not an option, as discussed in Section 3.1.1. Therefore we opt for a randomized heuristic algorithm. We describe two variants of the heuristic.

For illustrative purposes, consider a model involving 3 categories and 5 criteria. Figure 3.1 represents the profiles and criteria as well as four alternatives respectively denoted as $a^\star, a^\triangleright, a^\diamond$ and a°. Criteria weights have been set equal

($w_j = 0.2$ for $j = 1, \ldots, 5$) and the majority threshold λ is set to 80%. Hence, an alternative is considered superior to a profile if it is at least as good as the profile on either four or five criteria.

Assume that the first three alternatives are misclassified by this model. The first alternative, $a^\star$, is assigned to category C^1 according to the decision maker (DM) and to C^2 by the model. The second one, $a^\triangleright$, is assigned to category C^2 according to the DM and to C^1 by the model and the third one, $a^\diamond$, is assigned to category C^1 by the DM and to C^3 by the model. Assuming fixed weights and majority threshold, this means that the profiles delimiting the categories, are set either too high or too low on one or several criteria. Assume also that alternative a° is assigned to category C^1 both by the DM and the model.

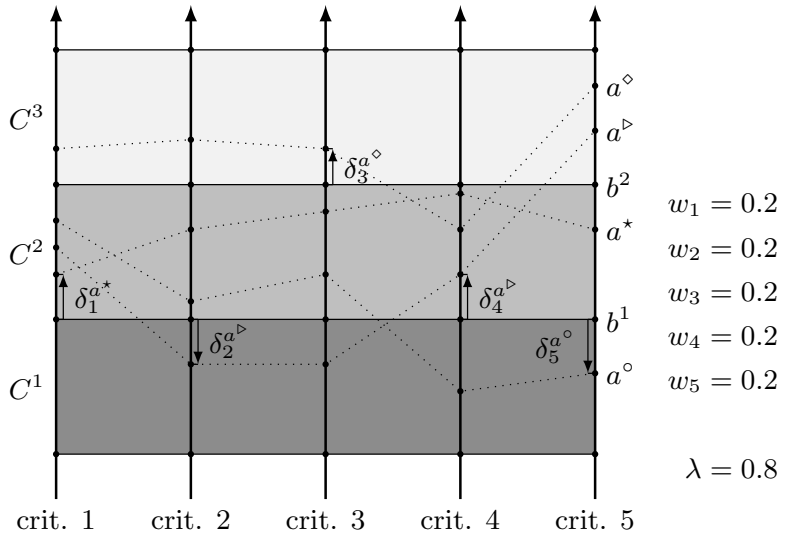


Figure 3.1: Alternatives wrongly assigned because of profiles set too low or too high

The idea implemented in the algorithm is to move up or down the profile value on some criterion in order to improve classification accuracy. We evaluate all possible moves of the profile on each attribute and select one likely to improve classification accuracy.

Variant 1

Consider only two categories, C^1 and C^2, in the model presented in Figure 3.1 and an alternative a. We denote by a_j, the evaluation of a on the criterion j. Regarding the position of a on criterion j, the assignment of a and a move of the

profile b^1 by $+\delta$ or $-\delta$, we can distinguish 8 cases (see Figure 3.2). We denote by A_l^{*h} the subset of alternatives of the learning set that are assigned to the category C^h by the decision maker but assigned in category C^l by the model.

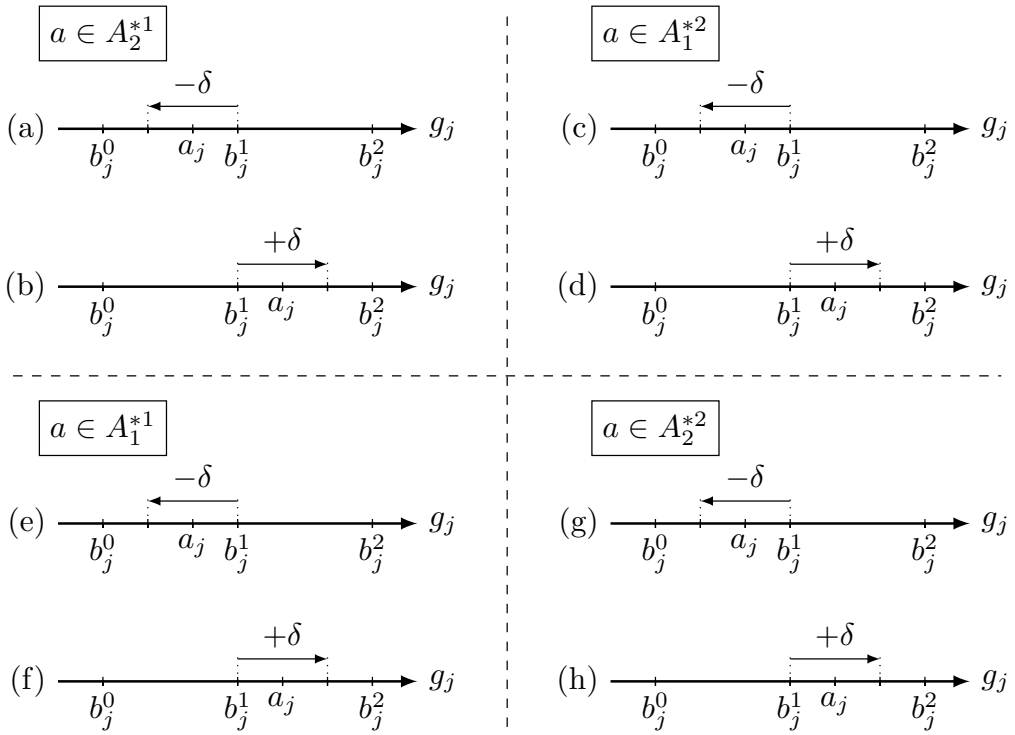


Figure 3.2: Given the evaluation of an alternative a and of the profile b^1 on a criterion j, 8 possible cases regarding the alternative assignment, the current profile value on j and a move of $+\delta$ or $-\delta$.

In the 8 cases represented in Figure 3.2, we see that a move of b^1 by $+\delta$ or $-\delta$ on criterion j can have a positive (cases b, c, f, g) or a negative (cases a, d, e, h) influence on the classification. We identify different subsets:

$W_{1,j}^{+\delta}$ **(resp.** $W_{1,j}^{-\delta}$**)** : the set of alternatives wrongly assigned by the model and for which moving the profile b^1 by $+\delta$ (resp. $-\delta$) is in favor of the correct assignment. It is for instance the case of $a^{\triangleright}$ if the profile is moved by $\delta_2^{a^{\triangleright}}$ on criterion 2.

$R_{1,j}^{+\delta}$ **(resp.** $R_{1,j}^{-\delta}$**)** : the set of alternatives for which moving the profile b^1 by $+\delta$ (resp. $-\delta$) on j does not favor the assignment to the correct class. It is for instance the case of a° if the profile is moved by $\delta_5^{a^{\circ}}$ on criterion 5.

Formally, the subsets read:

$$\begin{aligned}
W_{1,j}^{+\delta} &= \left\{a \in A_2^{*1} : b_j^1 + \delta > a_j \geq b_j^1\right\} \\
W_{1,j}^{-\delta} &= \left\{a \in A_1^{*2} : b_j^1 - \delta \leq a_j < b_j^1\right\} \\
R_{1,j}^{+\delta} &= \left\{a \in A_1^{*2} : b_j^1 + \delta > a_j \geq b_j^1\right\} \cup \left\{a \in A_2^{*2} : b_j^1 + \delta > a_j \geq b_j^1\right\} \\
R_{1,j}^{-\delta} &= \left\{a \in A_2^{*1} : b_j^1 - \delta \leq a_j < b_j^1\right\} \cup \left\{a \in A_1^{*1} : b_j^1 - \delta \leq a_j < b_j^1\right\}
\end{aligned}$$

In order to assess the advantages of the different possible moves of the profile level, the space between the profiles levels b_j^1 and b_j^0 on criterion j is split into k sub-intervals by means of k subdivision points denoted by $b_j^1 - \delta_l$ for $l = 1, \ldots, k$. The same is done between b_j^1 and b_j^2 by means of k subdivision points denoted by $b_j^1 + \delta_l$ for $l = 1, \ldots, k$. We consider these $2k$ subdivision points scattered on both sides of b_j^1 as the candidate moves for the profile level b_j^1. Then, histograms similar to those shown in Figure 3.3 are constructed for each criterion j. We denote by $W_{1,j}^{\pm\delta_l}$ (resp. $R_{1,j}^{\pm\delta_l}$) the union of $W_{1,j}^{+\delta_l}$ (resp. $R_{1,j}^{+\delta_l}$) and $W_{1,j}^{-\delta_l}$ (resp. $R_{1,j}^{-\delta_l}$). The bars lengths in the first histogram represent the number of alternatives in the set $W_{1,j}^{\pm\delta_l}$ (resp. $R_{1,j}^{\pm\delta_l}$). In the last histogram, the bars lengths represent what is formally a *desirability index* P defined by:

$$P(b_j^1 \pm \delta_l) = \frac{\left|W_{1,j}^{\pm\delta_l}\right|}{\left|W_{1,j}^{\pm\delta_l}\right| + \left|R_{1,j}^{\pm\delta_l}\right|} \tag{3.2}$$

If we move the profile level b_j^1 to $b_j^1 \pm \delta_l$, the number $|W_{1,j}^{\pm\delta_l}|$ will decrease by $|W_{1,j}^{\pm\delta_l}| - |R_{1,j}^{\pm\delta_l}|$ and the number $|R_{1,j}^{\pm\delta_l}|$ will increase by the same quantity. If the quantity $|W_{1,j}^{\pm\delta_l}| - |R_{1,j}^{\pm\delta_l}|$ 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 $b_j^1 + \delta_l$. Of course, the number of correctly assigned alternatives will not mechanically increase by $|W_{1,j}^{\pm\delta_l}| - |R_{1,j}^{\pm\delta_l}|$ since the corresponding change in the profile level only concerns criterion j. We use the probabilities $P(b_j^1 \pm \delta_l)$ as indicators of the potential gain in correct classification that can be expected from a move of the profile level on some criteria. 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 $b_j^1 \pm \Delta$ is 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 $P(b_j^1 \pm \Delta)$, then the profile is moved to $b_j^1 \pm \Delta$, otherwise the profile is not moved at all. The same operation is performed for each criterion.

One loop of the heuristic moving the profiles in the case of a model with 2 categories can be summarized by Algorithm 3.

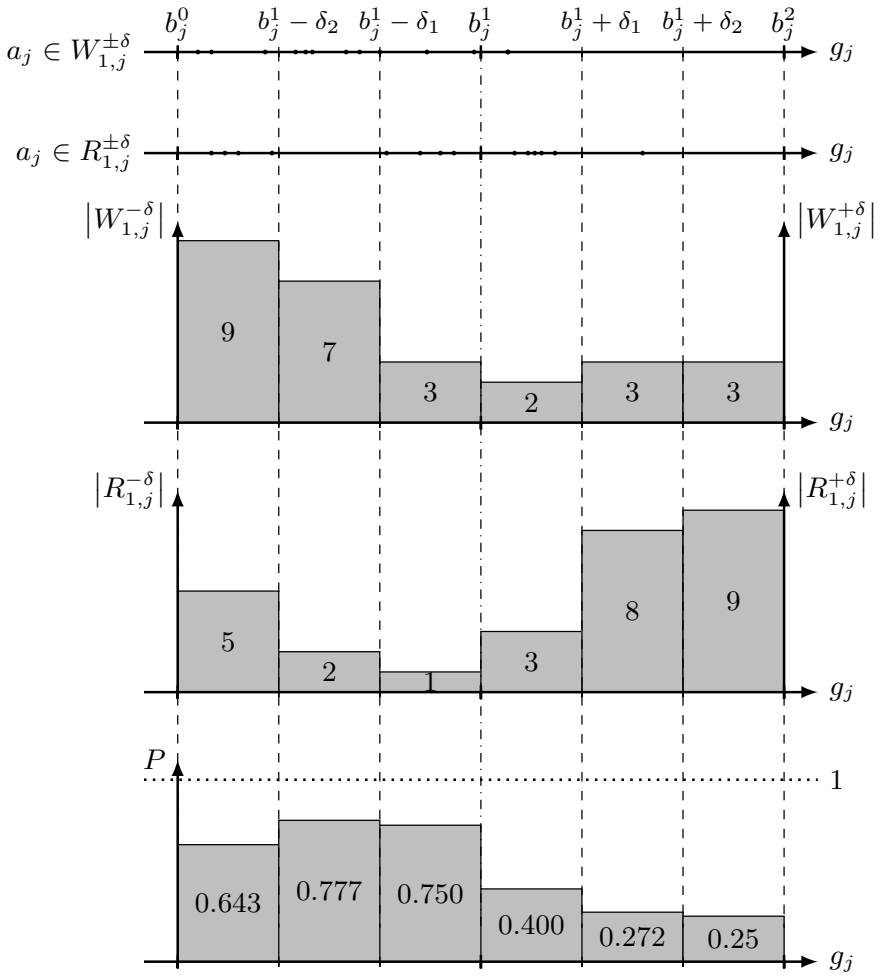


Figure 3.3: Histogram of the evaluations of misclassified alternatives on criterion j

When more than two categories are involved, a similar algorithm is applied to each profile. The *desirability index* (Equation (3.2)) is adapted as follows:

$$P(b_j^h \pm \delta_l) = \frac{\left|W_{h,j}^{\pm\delta_l}\right|}{\left|W_{h,j}^{\pm\delta_l}\right| + \left|R_{h,j}^{\pm\delta_l}\right|}$$

Algorithm 3 Heuristic moving the profile of a MR-Sort model

for all $j \in \{1, \ldots, n\}$ **do**
  Compute $P(b_j^1 \pm \delta_l)$, $\forall l$.
  Find L such that $P(b_j^1 \pm \Delta) = \max_l P(b_j^1 \pm \delta_l)$.
  Draw a random number r from the uniform distribution $[0, 1]$.
  if $r < P(b_j^1 \pm \Delta)$ **then**
    Move b_j^1 to the position corresponding to $b_j^1 \pm \Delta$.
    Update the alternatives assignment.
  end if
end for

with:

$$W_{h,j}^{+\delta} = \left\{a \in A_{h+1}^{*h} : b_j^h + \delta > a_j \geq b_j^h\right\}$$
$$W_{h,j}^{-\delta} = \left\{a \in A_h^{*h+1} : b_j^h - \delta \leq a_j < b_j^h\right\}$$
$$R_{h,j}^{+\delta} = \left\{a \in A_h^{*h+1} : b_j^h + \delta > a_j \geq b_j^h\right\} \cup \left\{a \in A_{h+1}^{*h+1} : b_j^h + \delta > a_j \geq b_j^h\right\}$$
$$R_{h,j}^{-\delta} = \left\{a \in A_{h+1}^{*h} : b_j^h - \delta \leq a_j < b_j^h\right\} \cup \left\{a \in A_h^{*h} : b_j^h - \delta \leq a_j < b_j^h\right\}$$

Variant 2

The second variant of the heuristic uses the value of the concordance index stating that an alternative a is as good as a profile b^h in the computation of the *desirability index*. To be more precise, let us define several subsets of alternatives for each criterion j and each profile h and any positive value δ, which represents the size of a move:

$V_{h,j}^{+\delta}$ **(resp.** $V_{h,j}^{-\delta}$**)** : the sets of alternatives misclassified in C^{h+1} instead of C^h (resp. C^h instead of C^{h+1}), for which moving the profile b^h by $+\delta$ (resp. $-\delta$) on j results in a correct assignment. For instance, $a^\triangleright$ belongs to the set $V_{1,2}^{-\delta}$ on criterion 2 for $\delta \geq \delta_2^{a^\triangleright}$.

$W_{h,j}^{+\delta}$ **(resp.** $W_{h,j}^{-\delta}$**)** : the sets of alternatives misclassified in C^{h+1} instead of C^h (resp. C^h instead of C^{h+1}), for which moving the profile b^h 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. For instance, $a^\star$ belongs to the set $W_{1,1}^{+\delta}$ on criterion 1 for $\delta > \delta_1^{a^\star}$.

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

misclassification. For instance, a° belongs to the set $Q_{1,5}^{-\delta}$ on criterion 5 for $\delta > \delta_5^{a^\circ}$.

$R_{h,j}^{+\delta}$ **(resp.** $R_{h,j}^{-\delta}$**)** : the sets of alternatives misclassified in C^h instead of C^{h+1} (resp. C^{h+1} instead of C^h), for which moving the profile b^h by $+\delta$ (resp. $-\delta$) on j still strengthens the criteria coalition in favor of the incorrect classification. For instance, $a^\triangleright$ belongs to the set $R_{1,4}^{+\delta}$ on criterion 4 for $\delta > \delta_4^{a^\triangleright}$.

$T_{h,j}^{+\delta}$ **(resp.** $T_{h,j}^{-\delta}$**)** : the sets of alternatives assigned by the model to C^{h+1} or higher (resp. C^h or lower) but classified by the DM in a category below C^h (resp. to a category above C^{h+1}), for which moving the profile by $+\delta$ (resp. $-\delta$) on j strengthens the criteria coalition in favor of a classification that comes closer to the correct one. For instance $a^\diamond$ belongs to the set $T_{2,3}^{+\delta}$ on criterion 3 for $\delta > \delta_3^{a^\diamond}$.

In the above, subsets of type V and W contain alternatives that will tend to be better classified if we perform a given profile move. On the contrary, the assignment of alternatives in subsets of type Q will be worsened by the move; the (wrong) classification of alternatives in subsets of type R and T will not be altered by the move, but the latter goes "in the wrong direction" with regard to a correct classification of these alternatives. In order to formally define these sets we introduce the following notation. A_l^{*h} denotes the subset of misclassified alternatives that are assigned to category C^l by the model while the ground truth states that they are assigned to category C^h. $A_{>l}^{*<h}$ denotes the subset of misclassified alternatives that are assigned to a category above C^l by the model while the ground truth states that they are assigned to a category below C^h. And conversely for $A_{<l}^{*>h}$. Finally, $\sigma(a, b^h) = \sum_{j:a_j \geq b_j^h} w_j$. We have, for any h, j

and positive δ:

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

The choice of a profile move is as follows. First, to avoid violations of the dominance rule between the profiles, the value of $+\delta$ or $-\delta$ is restricted to vary in the interval $[b_j^{h-1}, b_j^{h+1}]$. We then compute a *desirability index* $P(b_j^h + \delta)$ for each possible value $+\delta$ of a move of profile b_j^h. This index balances the alternatives that will be better off after the move and those on which the move will have a negative impact. The index is computed according to the following formula:

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

where k_V, k_W, k_T, k_Q, k_R, d_V, d_W, d_T, d_Q and d_R are fixed constants. We define similarly $P(b_j^h - \delta)$. In the definition of $P(b_j^h + \delta)$ (resp. $P(b_j^h - \delta)$), the coefficients weighting the number of elements in the sets in the numerator are chosen so as to emphasize the arguments in favor of moving the value b_j^h of profile b^h to $b_j^h + \delta$ (resp. $-\delta$), while the coefficients in the denominator emphasize the arguments against such a move. The values of the coefficients were empirically set as follows: $k_V = 2, k_W = 1, k_T = 0.1, k_Q = k_R = 0, d_V = d_W = d_T = 1, d_Q = 5, d_R = 1$.

The value b_j^h of profile b^h on criterion j will possibly be moved to the value a_j of one of the alternatives a contained in $V_{h,j}^{+\delta}$, $V_{h,j}^{-\delta}$, $W_{h,j}^{+\delta}$ or $W_{h,j}^{-\delta}$. More precisely, it will be set to a_j or a value slightly above a_j. The exact new position of the profile is chosen so as to favor a correct assignment for a, taking into account the assignment rule (2.9). For instance, with regard to the situation illustrated in Figure 3.1, the new value $b_1^1 + \delta$ could be chosen just above the value of $a_1^\star$ so that criterion 1 would no longer belong to the coalition of criteria on which $a^\star$ is *at least as good* as b^1. Such a move would result in correctly assigning $a^\star$ to

category C^1. If the move was driven by the position of alternative $a^\triangleright$ on criterion 2, then the new profile value $b_2^1 + \delta$ would be set equal to the performance, $a_2^\triangleright$, of the alternative $a^\triangleright$ on criterion 2. Such a move would result in correctly assigning $a^\triangleright$ to C^2.

All such values a_j are located in the interval $[b_j^{h-1}, b_j^{h+1}]$. A subset of such values is chosen in a randomized way as follows. Among the set of values a_j, a value, denoted by $b_j'^h$, is chosen randomly. We denote by d_j^h the difference $|b_j'^h - b_j^h|$. All values a_j located in $[b_j^{h-1}, b_j^h - d_j^h]$ and $[b_j^h + d_j^h, b_j^h]$ constitute a subset of candidate moves. The candidate move corresponds to the value a_j in the selected subset for which $P(b_j^h + \Delta)$ is maximal, Δ being equal to $a_j - b_j^h$ (i.e. a positive or negative quantity). To decide whether to make the candidate move, a random number r is drawn uniformly in the interval $[0, 1]$ and the value b_j^h of profile b^h is changed if $P(b_j^h + \Delta) \geq r$.

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

Algorithm 4 summarizes how this randomized heuristic operates.

Algorithm 4 Randomized heuristic used for improving the profiles

```
for all profile b^h do
    for all criterion j chosen in random order do
        Choose, in a randomized manner, a sub-interval of [b_j^{h-1}, b_j^{h+1}].
        Select a position in this sub-interval for which P(b_j^h + Δ) is maximal.
        Draw uniformly a random number r from the interval [0, 1].
        if r ≤ P(b_j^h + Δ) then
            Move b_j^h to the position corresponding to b_j^h + Δ.
            Update the alternatives assignment.
        end if
    end for
end for
```

3.4 Experimental results with artificial data sets

We performed several tests with the metaheuristic in order to understand how it behaves when the input settings and its parameters vary. The first tests consist in studying each component of the metaheuristic. It is done by generating artificial data sets and by modifying the parameters of the model and the ones of the algorithm. We assess the computing time, the model retrieval and the tolerance for error. The aim of these tests is to determine whether or not the strategy is appropriate for learning the parameters of a MR-Sort model.

After testing the components of the algorithm, we test the metaheuristic itself. The tests are done with artificial data sets. As for the test of its components, we observe how the algorithm behaves by varying the parameters of the model and the ones of the metaheuristic.

Finally, we perform experiments with the metaheuristic with real data sets and we compare the metaheuristic with other sorting algorithms.

3.4.1 Experimental setup

In order to assess the algorithm and its components, we setup an experimental framework. The experimental framework aims at assessing the algorithm in terms of computing time, model retrieval and tolerance for errors.

Computing time

The aim of our algorithm is to learn a model on the basis of large data sets in a reasonable amount of time. The computing time varies with the size of the problem. In order to assess the algorithm in terms of computing time, we use the following test procedure:

1. A MR-Sort model M is generated randomly. The weights are uniformly generated as described in Butler et al. (1997). It consists in drawing uniformly $n-1$ random numbers from the interval $[0,1]$ and ranked such that $r_n = 1 > r_{n-1} \geq \ldots \geq r_1 > 0 = r_0$. Then weights are defined as follows: $w_j = r_j - r_{j-1}$, with $j = 1, \ldots, n$. The majority threshold λ is uniformly drawn from the interval $[1/2, 1]$. For the profiles evaluations, on each criterion $p-1$ random numbers are uniformly drawn from the interval $[0,1]$ and ordered such that $r'_{p-1} \geq \ldots \geq r'_1$. Profiles evaluations are determined by $b_{h,j} = r'_h$, $h = 1, \ldots, p-1$. Using model M as described by Equation (2.9), each alternative can be assigned to a category. The resulting assignment rule is referred to as s_M.

2. A set of m alternatives with random performances on the n criteria is generated. The performances are uniformly and independently drawn from the $[0,1]$ interval. The set of generated alternatives is denoted by A^*. The alternatives in A^* are assigned using the rule s_M. The resulting assignments and the performances of the alternatives in the set A^* are given as input to the algorithm. They constitute the learning set.

3. On the basis of the assignments and the performances of the alternatives in A^*, the algorithm learns a MR-Sort model which maximizes the classification accuracy. The model learned by the metaheuristic is denoted by M' and the corresponding assignment rule, $s_{M'}$.

The computing time is assessed by measuring the CPU time required to perform the step 3.

Convergence of the algorithm

To assess the convergence of the algorithm, we compare the assignments obtained with models M and M' and compute the classification accuracy. In our first experimentations, we consider only the classification accuracy as quality index of the model. Formally, the following step is added to the test procedure:

4. The alternatives of the learning set A^* are assigned using the rule $s_{M'}$. The assignments resulting from this step are compared to the ones obtained at step 2 and the classification accuracy is computed according to Equation (2.28) in which A is replaced by A^*. We denote it by $CA^*(s_M, s_{M'})$.

The experimentation is done for learning sets of different sizes. Then the value $CA^*(, s_M, s_{M'})$ shows the ratio of examples that are restored.

Model retrieval

Testing model retrieval aims at determining how many examples are needed to be able to find the parameters of a MR-Sort model that reflects as well as possible the preferences of a DM. In other words, this experiment aims at answering the following question: How many assignment examples are required to obtain the parameters of a model restoring correctly a large proportion of assignment examples? To answer this question we add a step to the test procedure described above:

5. A set of 10000 random alternatives, A, is generated in a similar way as in step 2. We call this set the generalization set. Alternatives of the set A are assigned by models M and M'. Finally the assignments obtained by models M and M' are compared and the classification accuracy is computed according to Equation (2.28). We denote it by $CA(s_M, s_{M'})$.

The value of $CA(s_M, s_{M'})$ shows how the algorithm behaves in generalization.

Tolerance for errors

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

2' A proportion of errors is added in the assignments obtained using the model M. For each alternative of the learning set, its assignment is altered with

probability P, the altered category assigned to an example is uniformly drawn among the other categories. We denote by $\tilde{s}_M$ the rule producing the assignments with errors.

The next three steps are then applied as previously but the rule s_M is substituted by $\tilde{s}_M$.

3.4.2 Experiments on the components of the algorithm

In the text above we exposed the test procedure that has been used in order to test the metaheuristic. We recall that the metaheuristic is composed of three main components that have been described in Section 3.3. In this subsection, we present the results obtained with regard to the computing time, model retrieval and tolerance for error of the linear program used to learn the weights and majority threshold and the heuristic improving the profiles.

Linear program learning the weights and majority threshold

We show here how the linear program (LP) used to learn the weights and majority threshold of a MR-Sort model behaves with regard to computing time, model retrieval and tolerance for errors.

Computing time We assess the ability of the linear program to restore a set of weights and majority threshold in a reasonable amount of time. To proceed, we follow the experimental procedure described in Section 3.4.1.

To see how much time is needed to learn the weights and the majority threshold, the linear program, described in Section 3.3.3 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. The experiment has been repeated 10 times, each time with a different learning set A^* and model M.

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 filtering the constraints of the LP given in Equation (3.1) in order to eliminate the redundant ones.

Figure 3.4 shows the average computing time required when learning the parameters of a MR-Sort model composed of 3 categories and 5 to 20 criteria on the basis of a learning set containing 1000 to 10000 assignment examples. We observe 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

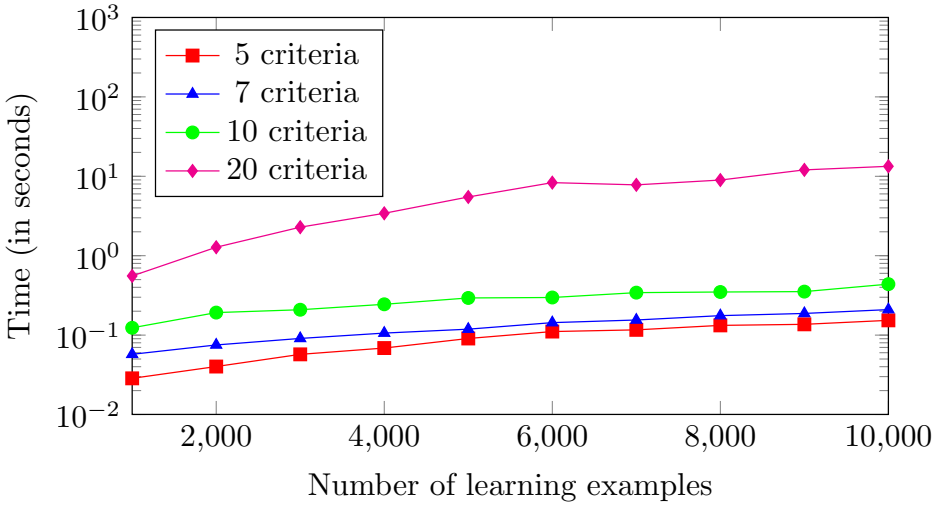


Figure 3.4: Average computing time needed to learn the weights and the majority threshold of a MR-Sort model involving 3 categories and 5, 7, 10 or 20 criteria with the LP described in Section 3.3.3. The learning set contains from 1000 to 10000 alternatives.

of criteria. This is due to the fact that the number of non-redundant constraints quickly grows when the number of criteria is increased.

Model retrieval Model retrieval aims at finding the number of alternatives that are required in order to infer the parameters of a model composed of p categories and n criteria such that it restores as well as possible the preferences of a DM.

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. This experiment has been repeated 10 times, each time with a different learning set A^*, model M and test set A. The evolution of the classification accuracy is shown in Figure 3.5.

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. This indicates that the original model has been well approximated.

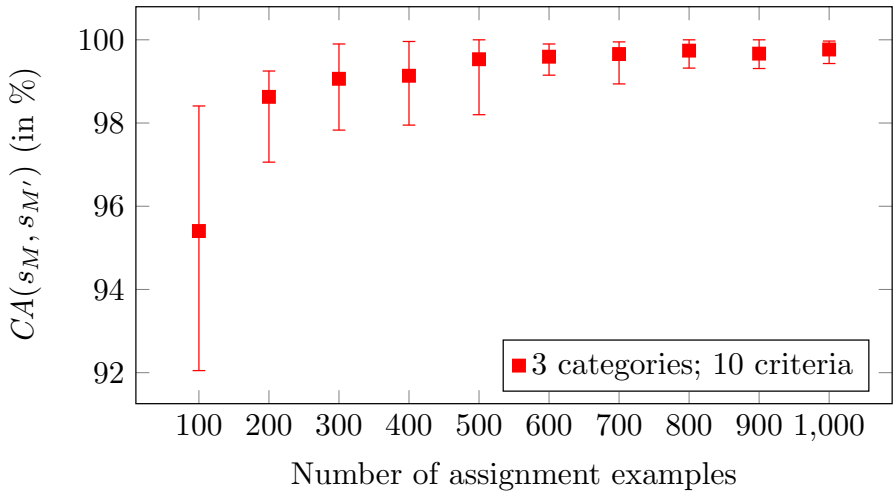


Figure 3.5: Average, minimum and maximum CA of the test set containing 10000 alternatives. The learning set contains from 100 to 1000 assignment examples. The weights and the majority threshold of the MR-Sort model involving 3 categories and 10 criteria are learned with the LP described in Section 3.3.3.

Tolerance for errors The tolerance for error of the LP is assessed by adding errors in the learning set as described in Section 3.4.1.

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 the 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 test set. This experiment has been repeated 10 times, each time with a different learning set A^*, model M and test set A.

Figure 3.6 shows 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 is usually 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.

A further issue is the following. Are the alternatives wrongly assigned by the learned model mostly alternatives that have been erroneously reassigned to introduce errors 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

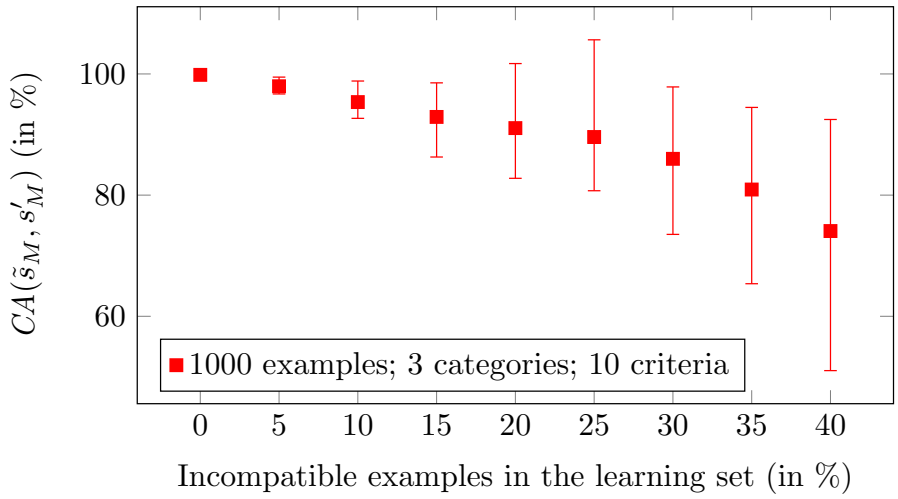


Figure 3.6: Average, minimum and maximum CA of the test set containing 10000 alternatives. The learning set contains 1000 assignment examples with 0 to 40% of erroneous assignments. The weights and majority threshold of the MR-Sort model involving 3 categories and 10 criteria are learned with the LP described in Section 3.3.3.

the learned weights and majority threshold, what's the percentage of alternatives that were degraded in this set? By looking at 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.

Heuristic for improving the profiles

As for the linear program inferring the weights and majority threshold, we performed the same experiments on the heuristic improving the profiles. In this subsection we study how the heuristic adjusting the profiles behaves. First we observe how it converges over the iterations. Then we test computing time, model retrieval and tolerance for errors.

Convergence of the algorithm We compare the two variants of the heuristic algorithm presented in Section 3.3.4. This experiment was repeated 10 times, each time with a different learning set A^* and model M.

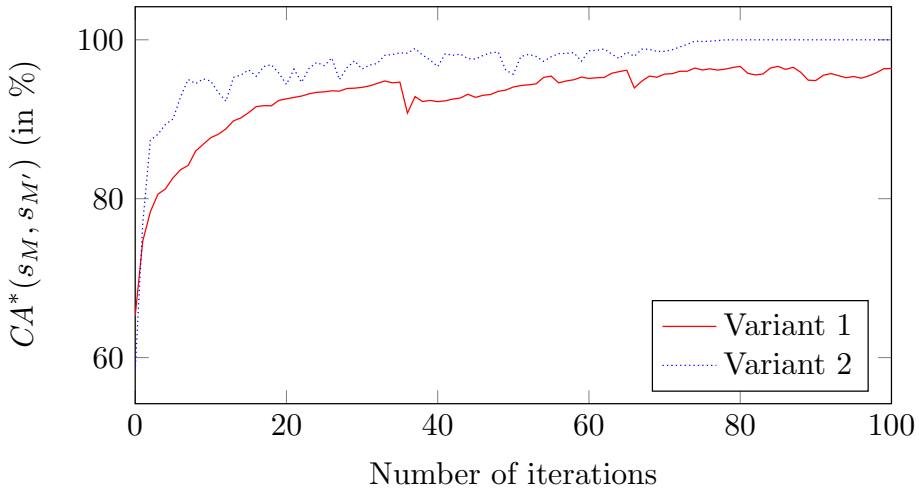


Figure 3.7: Evolution of the average CA over the iterations with a learning set containing 10000 assignment examples. The learned model involves from 2 to 5 categories and 10 criteria. The plot shows the evolution of the CA of the learning set for the two variants of the heuristic described in Section 3.3.4.

Figure 3.7 shows the evolution of the average classification accuracy of the learning set when learning a MR-Sort composed of 3 categories and 10 criteria on the basis of a learning set composed of 10000 assignment examples. We observe that the second variant of the heuristic is more efficient than the first one. Indeed, after 100 iterations, the algorithm reaches a CA equal to 100%. Therefore, we opt for the second variant of the heuristic in the sequel.

Computing time As for the LP inferring the weights and majority threshold, we test the rapidity of the metaheuristic to restore the profiles of the model.

Figure 3.8 shows that less than 10 seconds are needed to learn a model composed of 10 criteria and 3 categories on the basis of a learning set composed of 1000 alternatives. With 10000 examples of assignments, the computing time remains reasonable: it takes on average 35 seconds to restore the profiles.

Model retrieval We test the model retrieval on a model composed of 3 categories and 10 criteria. Figure 3.9 shows the average, minimum and maximum classification accuracy of the test set composed of 10000 alternatives. The results show that the learned model is able to restore on average more than 95% of the assignments when the learning set is composed of 200 examples. With 500 alternatives in the learning set, the classification accuracy increases to 97%.

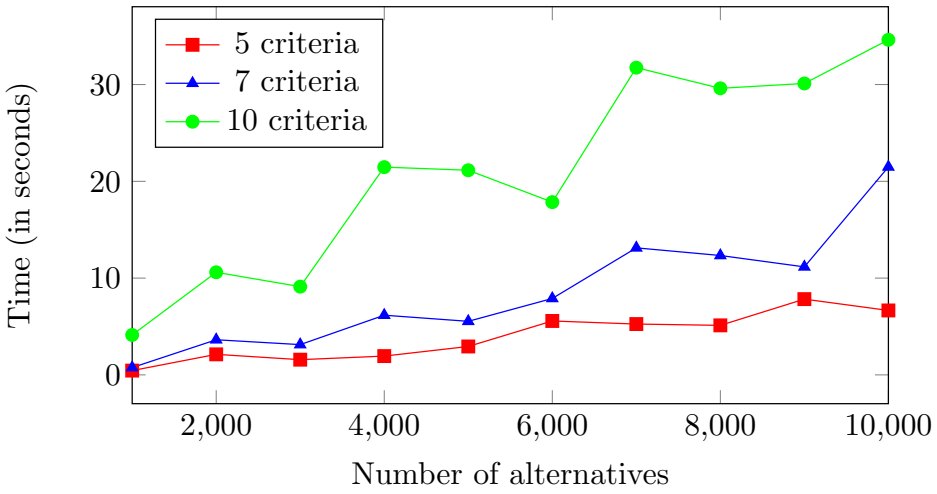


Figure 3.8: Average computing time needed to learn the profiles of a MR-Sort model involving 3 categories and 5, 7 or 10 criteria with the heuristic described in Section 3.3.4. The learning set contains from 1000 to 10000 alternatives.

These results show that the algorithm is able to restore a "good" model, i.e. one that restore a large number of assignments of the test set, with relatively few examples of assignments.

Tolerance for errors The tolerance for errors is tested on a MR-Sort model composed of 10 criteria and 3 categories with a learning set containing 1000 assignment examples. The probability of errors in the learning set varies from 0 to 40%.

Figure 3.10 shows the evolution of the classification accuracy in generalization. We observe that the algorithm is not perturbed too much by the errors in the learning set. Indeed, when the learning set is composed of 40% of altered assignments, the average classification accuracy remains close to 90%. However, we observe that the distance between the minimal and maximal value of $CA(A, \tilde{s}_M, s_{M'})$ grows. Nevertheless it remains smaller than the error rate of the learning set. We conclude that the algorithm is capable of identifying the noise introduced in the learning set.

3.4.3 Experiments with the metaheuristic

Experiments on the components of the metaheuristic showed that they are able to handle errors and to restore the preference of a DM with a relatively small amount of assignment examples. We now perform the same type of tests with

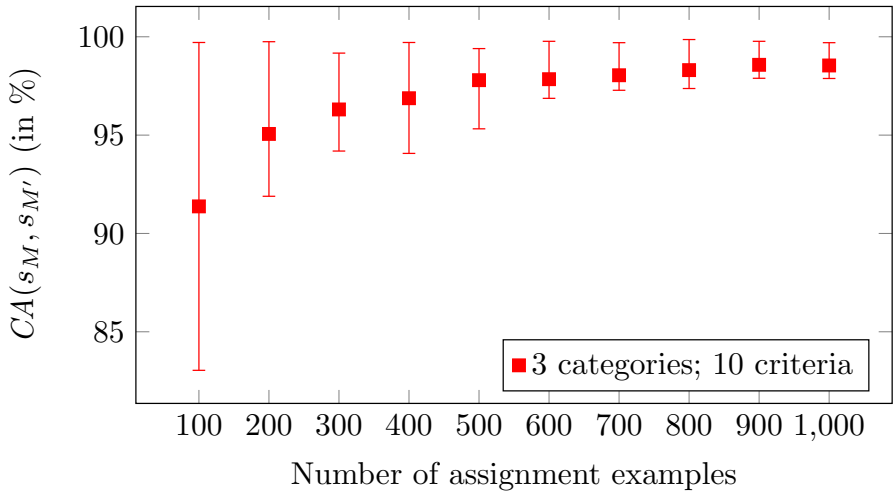


Figure 3.9: Average, minimum and maximum CA of the test set containing 10000 alternatives. The learning set contains from 100 to 1000 assignment examples. The profiles of the MR-Sort model involving 3 categories and 10 criteria are learned with the heuristic described in Section 3.3.4.

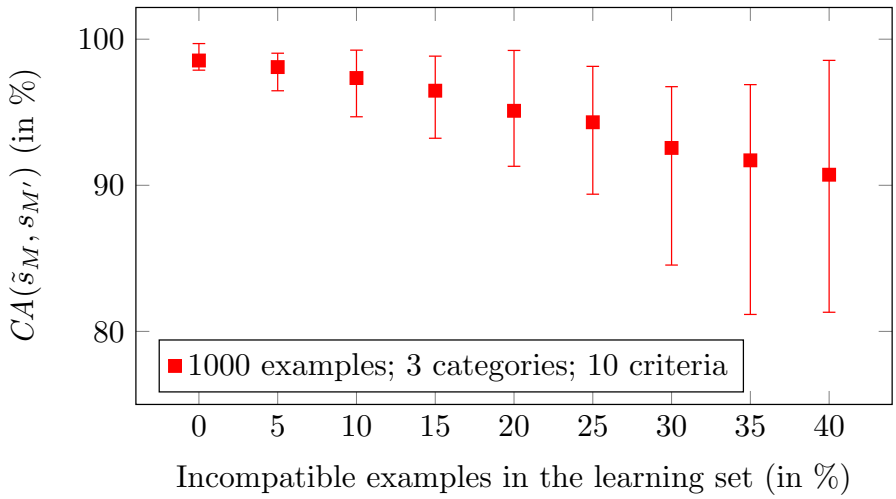


Figure 3.10: Average, minimum and maximum CA of the test set containing 10000 alternatives. The learning set contains 1000 assignment examples with 0 to 40% of erroneous assignments. The profiles of the MR-Sort model involving 3 categories and 10 criteria are learned with the heuristic described in Section 3.3.4.

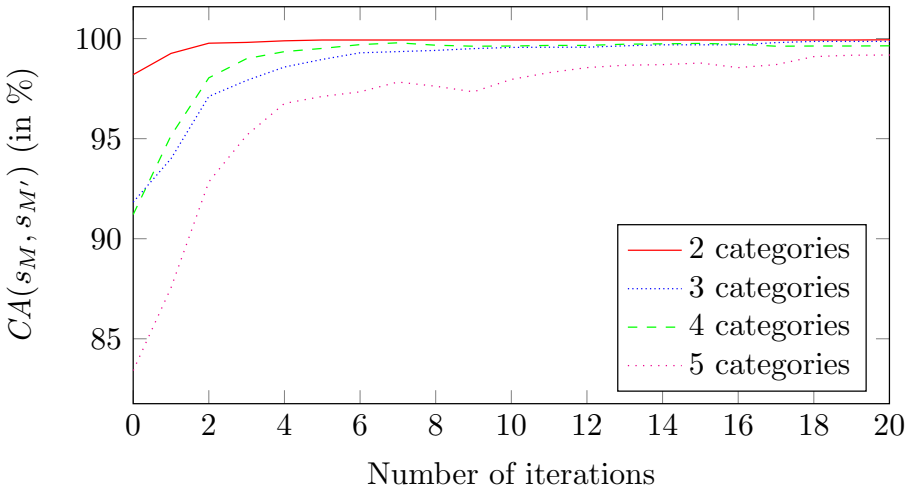


Figure 3.11: Evolution of the average CA over the iterations with a learning set involving 10000 assignment examples. The learned model involves 3 categories and 10 criteria. The plot shows the evolution of the CA of the learning set with the model learned by the metaheuristic described in Section 3.3.1.

the complete metaheuristic. We used the variant 1 of the metaheuristic which consists of maximizing the classification accuracy of the learning set. All the experiments have been done with the following parameters: $N_{mod} = 10$, $N_o = 30$ and $N_{it} = 20$.

Convergence of the algorithm

To test the convergence of the metaheuristic, we treat decision problems involving 10 criteria and 2 to 5 categories. This is the typical size of problems we want to be able to handle with the metaheuristic.

In Figure 3.11, the average value of $CA(s_M, s_{M'})$ obtained after repeating 10 times the experiment presented in Section 3.4.1 is shown. When the number of categories increases, we observe that the algorithm needs more iterations to converge to a model restoring correctly all assignment examples. This experiment shows that it is possible to find a model restoring 99% of the assignment examples in a reasonable computing time. On average two minutes are required to find the parameters of a model having 10 criteria and 5 categories with $N_{mod} = 10$, $N_o = 30$ and $N_{it} = 20$.

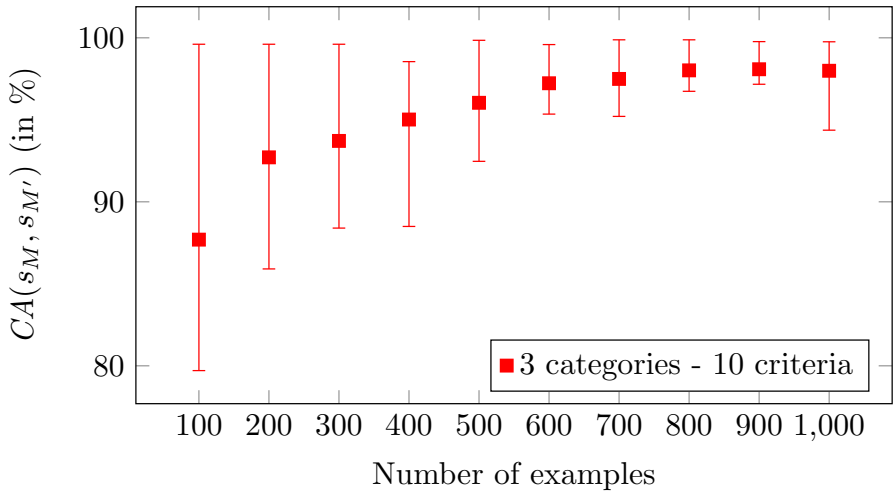


Figure 3.12: Average, minimum and maximum CA of the test set containing 10000 alternatives. The learning set contains from 100 to 1000 assignment examples. The parameters of the MR-Sort model involving 3 categories and 10 criteria are learned with the metaheuristic described in Section 3.3.1.

Model retrieval

Figures 3.12 and 3.13 show the average, min and max $CA(s_M, s_{M'})$ of the generalization set after learning the parameters of models having 10 criteria and 3 or 5 categories based on 100 to 1000 assignment examples. Figure 3.12 shows that 400 examples are sufficient to restore on average 95 % of the assignments for models having 3 categories, 10 criteria while 800 examples are needed for ones having 5 categories, 10 criteria (see Figure 3.13). As expected, the higher the cardinality of the learning set, the higher $CA(s_M, s_{M'})$ in generalization.

Tolerance for errors

Tolerance for errors is tested by learning the parameters of a MR-Sort model having 5 categories and 10 criteria on the basis of 1000 assignment examples generated using $\tilde{s}_M$. In Figure 3.14, the average classification accuracy of the learning set is shown for 10 test instances with 10 to 40 % of errors in the learning set. We observe that $CA(\tilde{s}_M, s_{M'})$ converges to $1 - P$ when there are errors in the learning set. Among the assignment examples badly assigned by the model, a majority corresponds to altered examples. To see to what extent the errors affect the algorithm, we generate a test set that is assigned both by the rule s_M and $s_{M'}$.

The resulting sets are compared and $CA(s_M, s_{M'})$ is computed. In Figure

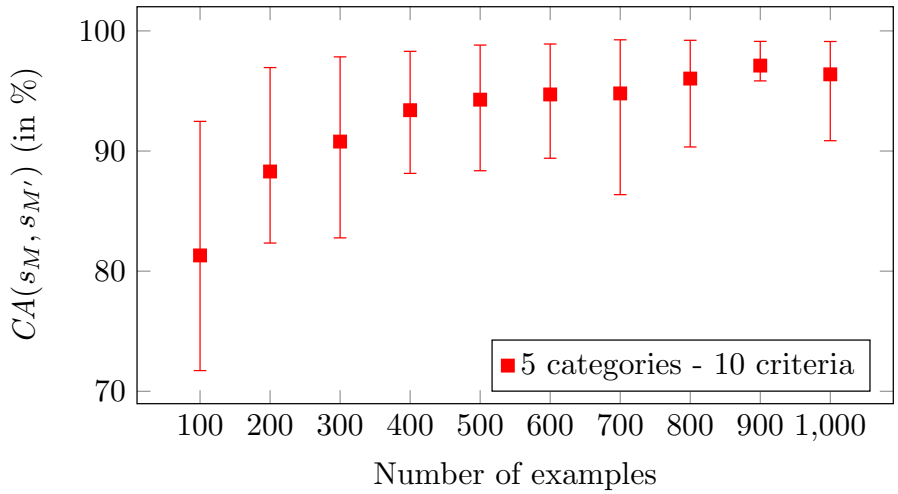


Figure 3.13: Average, minimum and maximum CA of the test set containing 10000 alternatives. The learning set contains from 100 to 1000 assignment examples. The parameters of the MR-Sort model involving 5 categories and 10 criteria are learned with the metaheuristic described in Section 3.3.1.

3.15, average, minimal and maximal $CA(s_M, s_{M'})$ are shown for 10 test instances. We observe that for small numbers of errors, i.e. less than 20 %, the algorithm tends to modify the model such that $CA(s_M, s_{M'})$ is altered on average by the same percentage of error in generalization. When there are more than 20% of errors in the learning set, the algorithm is able to find a model giving a smaller proportion of assignment errors in generalization.

Idiosyncratic behavior

This experiment aims at checking if a MR-Sort model is able to represent assignments that have been obtained by another sorting rule based on an additive value function.

We use an additive value function sorting (AVF-Sort) model as described in Section 2.2.4. In such a model, a marginal value function u_j is associated to each criterion. In this experiment, the marginal value functions u_j are piecewise linear. Each function is composed of k segments.

We study the ability of our metaheuristic to learn a MR-Sort model from a learning set generated with an AVF-Sort model. To do so, we replace step 1 by:

1. A sorting model M based on an additive value function is randomly generated. To generate the weights, the same rule as for the MR-Sort model is used. For each value function, $k-1$ random numbers are uniformly drawn

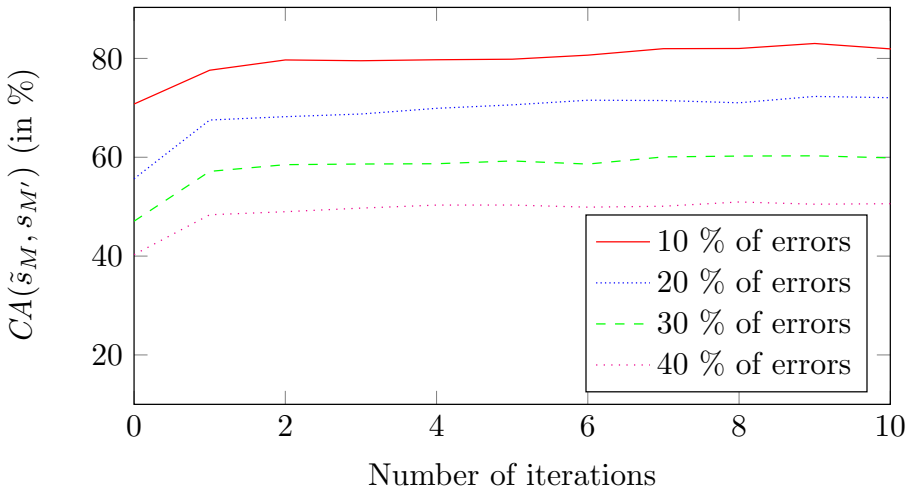


Figure 3.14: Evolution of the average CA of the learning set composed of 1000 alternatives. The learning set contains from 100 to 1000 assignment examples with 0 to 40% of erroneous assignments. The models learned with the metaheuristic described in Section 3.3.1 involve 5 categories and 10 criteria.

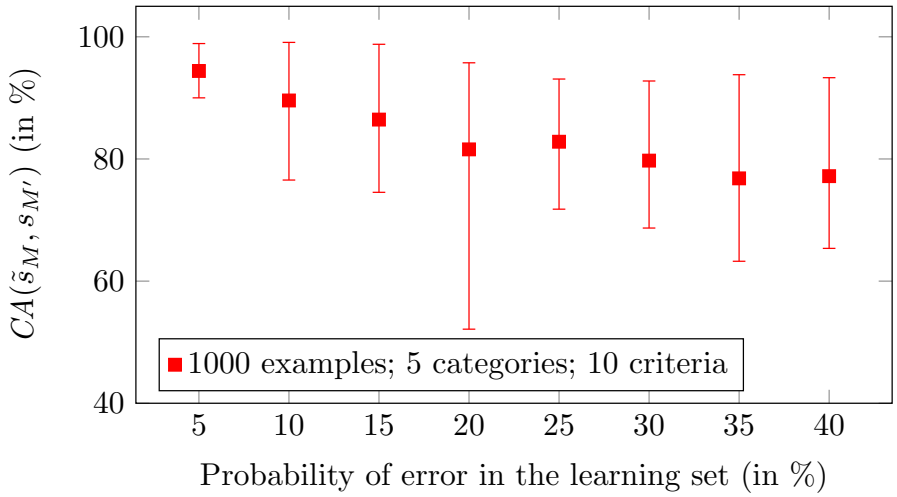


Figure 3.15: Average, minimum and maximum CA of the test set composed of 10000 alternatives. The learning set contains from 100 to 1000 assignment examples with 0 to 40% of erroneous assignments. The models learned with the metaheuristic described in Section 3.3.1 involve 5 categories and 10 criteria.

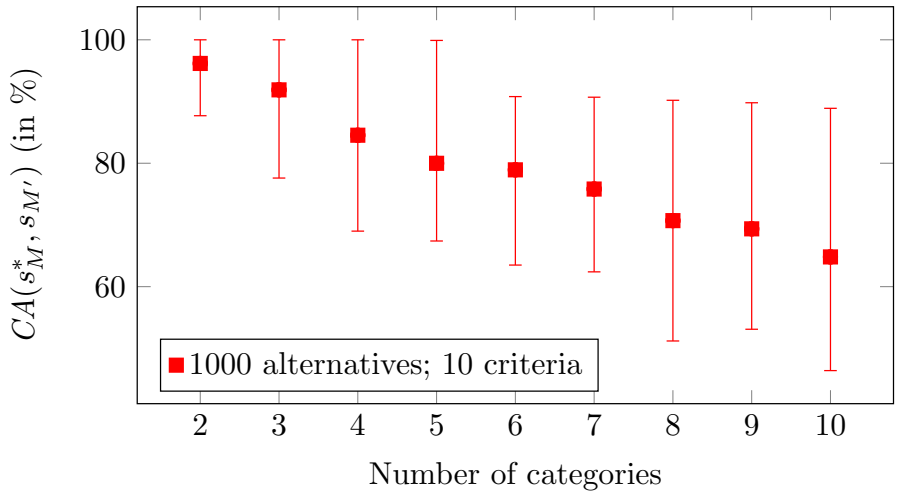


Figure 3.16: Average, minimum and maximum CA of the learning set containing 1000 assignment examples obtained with an AVF-Sort model involving 10 criteria and 2 to 10 categories. The MR-Sort model is learned with the metaheuristic described in Section 3.3.1.

from the interval $[0, 1]$ and ordered such that $r_k = 1 \geq r_{k-1} \geq \ldots \geq r_1 \geq 0 = r_0$, then end points are assigned as follows $u_j(x_j^l) = r_l$, with $l = 0, \ldots, k$. For the category limits U^h, $p-1$ random numbers are uniformly drawn from the interval $[0, 1]$ and then ordered such that $r_{p-1} \geq \ldots \geq r_1$. Category limits are given by $U^h = r_h$, $h = 1, \ldots, p-1$. The assignment rule is denoted by s^*_M.

Once the model has been generated, the alternatives are assigned by the model M and the metaheuristic tries to learn a MR-Sort model from the assignments obtained by M.

To assess the ability of the heuristic to find a MR-Sort model restoring the maximum number of examples, we test it with 1000 assignment examples, on models composed of 10 criteria and 2 to 10 categories. We choose to use an AVF-Sort model in which each additive value function is composed of 3 segments. This experiment is repeated 10 times.

Figure 3.16 presents the average, minimum and maximum $CA(s^*_M, s_{M'})$ of the learning set. The plot shows that the MR-Sort model is able to represent on average 80% of the assignment examples obtained with an AVF-Sort model when there are no more than 5 categories.

We perform a generalization by assigning 10000 alternatives through the AVF-Sort model, M and through the learned MR-Sort model, M'. Figure 3.17 shows the average, minimum and maximum classification accuracy of the generalization

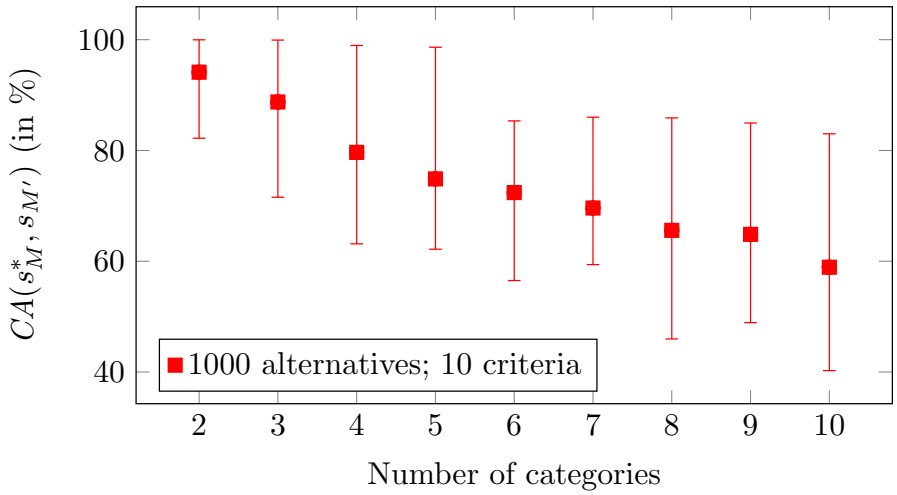


Figure 3.17: Average, minimum and maximum CA of the test set containing 10000 assignment examples obtained with an AVF-Sort model involving 10 criteria and 2 to 10 categories. The MR-Sort model is learned with the metaheuristic described in Section 3.3.1.

set. These results confirm the behavior observed with the learning set. The ability to represent assignments obtained by an AVF-Sort model with a MR-Sort model is limited, even more when the number of categories increases.

3.5 Experimental results with real data sets

In this section, we study the performance of the metaheuristic with real data sets. We observe to what extent the metaheuristic is able to restore a model representing as well as possible the preference information in the data sets. For these experiments we used the second variant of the metaheuristic maximizing the AUC criterion.

3.5.1 Data sets and experimental design

For comparison purposes, we use the data sets that were considered by Tehrani et al. (2012) for testing the performance of a binary classifier based on the Choquet integral. These data sets were taken from two sources: the UCI machine learning repository and the WEKA repository. In addition, we consider also the ASA data set which was compiled and studied by Lazouni et al. (2013b) (available at `http://olivier.sobrie.be`). The characteristics of all data sets

are displayed in Table 3.1. All the attributes in these data sets are treated as monotone attributes.

Table 3.1: Data sets

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
ASA	898	16	4

The tests are conducted as follows. Each data set is randomly split in two disjoint parts. The first part is used as learning set and the second part as test set. The following ratios between the size of the learning set and the size of the test set are considered: 20/80, 50/50, 80/20. For each data set and each ratio, a random drawing of the learning set from the whole data set is repeated 100 times, yielding 100 instances of a partition of the data set in a learning set and a test set.

For each learning set instance, the algorithm finds a model that minimizes the 0/1 loss, i.e. that is compatible with as many examples as possible. Afterwards, the alternatives in the test set are assigned by the learned model and the resulting assignments are compared to the original ones. This procedure is thus repeated 100 times for each data set and each relative size of the learning set.

Two indicators are computed to assess the quality of the learned models: the 0/1 loss and the AUC.

The performance of our heuristic algorithm is not only compared with the results obtained by Tehrani et al. (2012), but also with the exact solution of the MIP formulation (whenever it can be obtained) and with another previously mentioned multiple-criteria decision analysis (MCDA) method, UTADIS (see Section 2.4.3 in Chapter 2).

During all the experimentation, the MR-Sort metaheuristic is run with a population of 10 models ($N_{mod} = 10$) and the maximal number of iterations is fixed to 10 ($N_o = 10$). The outer loop of the metaheuristic, which adjusts the profiles and recomputes weights, is repeated 20 times ($N_{it} = 20$).

3.5.2 Binary classification

The algorithm developed by Tehrani et al. (2012) is designed for monotone sorting in two categories. In order to compare the performance of our algorithm with theirs, the assignments in the data sets presented in Table 3.1 are binarized by thresholding at the median, in the same way these authors did. From these data sets, the parameters of a MR-Sort model are learned by using 20, 50 or 80 percent of the records as learning alternatives and the rest as test alternatives.

Our experimentation has two objectives. The first is to compare the quality of the MR-Sort models found by our metaheuristic with the ones obtained by an exact optimization method. Therefore, we solve the mixed integer programming formulation studied in Leroy et al. (2011) which minimizes the 0/1 loss of the model. Whenever the MIP solver is able to find a solution in the computing time allowed, we assess the learned models by comparing their average 0/1 loss on the test set.

Our second objective is to compare the performance of the proposed metaheuristic with that of other MCDA and machine learning algorithms, UTADIS (Jacquet-Lagrèze and Siskos, 1982; Doumpos and Zopounidis, 2002), a well-known MCDA method described in Section 2.4.3 in Chapter 2, and the Choquistic regression (CR) (Tehrani et al., 2012), a method recently developed in the field of preference learning. To assess our metaheuristic, we use the average 0/1 loss and AUC computed on the test sets.

The general definition of AUC has been reminded in Section 2.5.4. The AUC of a MR-Sort model with two categories C^1, C^2 is computed by comparing the concordance indices of the alternatives with regard to the profiles. The value of the MR-Sort AUC is given by the following equation:

$$AUC = \frac{1}{|A^1| \cdot |A^2|} \sum_{a^i \in A^2} \sum_{a^k \in A^1} \tau(a^i, a^k) \tag{3.3}$$

with A^1 (resp. A^2), the set of input alternatives classified in C^1 (resp. C^2). In the case of MR-Sort, we define $\tau(a^{(i)}, a^{(k)})$ as follows:

$$\tau(a^i, a^k) = \begin{cases} 0 & \text{if } \sum_{j:a_j^{(i)} \geq b_j^1} w_j < \sum_{j:a_j^{(k)} \geq b_j^1} w_j, \\ 0.5 & \text{if } \sum_{j:a_j^{(i)} \geq b_j^1} w_j = \sum_{j:a_j^{(k)} \geq b_j^1} w_j, \\ 1 & \text{if } \sum_{j:a_j^{(i)} \geq b_j^1} w_j > \sum_{j:a_j^{(k)} \geq b_j^1} w_j. \end{cases}$$

In the case of UTADIS, the value of AUC is also computed through formula (3.3) but $\tau(a^{(i)}, a^{(k)})$ is defined differently: it compares the values of the alternatives,

i.e.

$$\tau(a^i, a^k) = \begin{cases} 0 & \text{if } u(a^{(i)}) < u(a^{(k)}), \\ 0.5 & \text{if } u(a^{(i)}) = u(a^{(k)}), \\ 1 & \text{if } u(a^{(i)}) > u(a^{(k)}). \end{cases}$$

Results

Table 3.2 shows the average 0/1 loss obtained on the test sets with the learned models. Table 3.3 shows the average value of the AUC. Each entry in these tables records the average value and standard deviation for 100 random splits of the data sets into learning and test sets.

In these tables, column "Size" displays the percentage of alternatives of the data set used by the algorithms as learning set. Column "META" shows the results obtained with the metaheuristic described in this chapter. Column "MIP" contains the results obtained with the MIP described in Leroy et al. (2011). Column "UTADIS" displays the results obtained with UTADIS and column "CR" contains the results obtained with the CR (Tehrani et al., 2012) on all the data sets, except ASA (not available).

In column "MIP", some cells are empty because the solver was not able to find a solution for one test instance in less than one hour. As compared to solving the MIP formulation, for the largest data set, i.e. the CEV data set, the metaheuristic uses 50 seconds on average to find a model when the learning set consists of 80% of all the examples in the data set.

To assess the ability of the algorithm to find models restoring the assignment of a large number of examples, we compare the average 0/1 loss and AUC obtained with the MIP and the metaheuristic for the test sets. Note that the MIP finds a MR-Sort model that is compatible with the largest possible number of examples from the learning set. There is no other MR-Sort model restoring correctly more assignment examples.

Table 3.2 shows that the 0/1 loss obtained by the exact algorithm is on average 1% smaller than by the metaheuristic. This is due to the fact that the MIP finds models restoring an optimal number of assignment examples (from the learning set) while the metaheuristic can remain stuck in local minima. However this better performance does not hold for all data sets when applied to the test set. For instance, the MIP returns results slightly worse than the metaheuristic for the DBS and ESL data sets. This is probably due to an overfitting effect on the learning set.

We observe in Table 3.3 that the average AUC of the metaheuristic is close to the one of the MIP. This indicates that the quality of the classifiers obtained with the MIP and the metaheuristic are similar.

Table 3.2: Average and standard deviation of the 0/1 loss (in percent) of the test set for learning sets of different sizes.

Size	Data set	META	MIP	UTADIS	CR
20 %	DBS	18.97 ± 4.23	19.77 ± 4.81	20.08 ± 5.33	17.13 ± 4.24
	CPU	9.94 ± 3.23	9.00 ± 3.45	6.52 ± 3.62	8.11 ± 1.03
	BCC	28.24 ± 2.73	26.78 ± 2.76	29.15 ± 3.07	27.75 ± 3.35
	MPG	20.25 ± 3.56	20.80 ± 3.26	22.25 ± 3.18	7.09 ± 1.93
	ESL	10.42 ± 1.71	10.75 ± 1.58	8.89 ± 1.60	6.82 ± 1.29
	MMG	16.97 ± 0.87	17.16 ± 1.40	18.40 ± 1.84	17.25 ± 1.20
	ERA	21.36 ± 2.05	20.93 ± 1.74	23.68 ± 1.87	28.89 ± 2.73
	LEV	16.74 ± 1.87	16.08 ± 1.73	16.54 ± 1.60	14.99 ± 1.22
	CEV	9.37 ± 1.12	-	7.94 ± 0.59	4.48 ± 0.89
	ASA	2.29 ± 1.09	-	3.69 ± 1.41	-
50 %	DBS	16.23 ± 4.69	16.27 ± 4.26	14.80 ± 4.21	15.72 ± 4.16
	CPU	6.75 ± 2.37	6.40 ± 2.39	2.30 ± 2.38	4.64 ± 2.81
	BCC	27.50 ± 3.17	-	28.54 ± 2.46	26.87 ± 2.82
	MPG	17.81 ± 2.37	-	20.90 ± 2.36	5.77 ± 2.51
	ESL	10.04 ± 1.86	10.18 ± 1.55	7.83 ± 1.63	6.01 ± 1.26
	MMG	17.32 ± 1.51	-	17.58 ± 1.52	16.67 ± 1.44
	ERA	20.56 ± 1.73	19.58 ± 1.37	23.42 ± 1.71	28.44 ± 3.06
	LEV	15.92 ± 1.22	14.22 ± 1.54	15.56 ± 1.32	13.72 ± 1.25
	CEV	9.36 ± 1.19	-	7.99 ± 0.91	3.76 ± 0.59
	ASA	1.38 ± 0.61	-	2.47 ± 0.82	-
80 %	DBS	15.92 ± 6.98	14.80 ± 8.11	12.80 ± 5.01	14.16 ± 6.81
	CPU	6.40 ± 3.04	5.98 ± 3.15	1.52 ± 2.14	2.12 ± 3.01
	BCC	26.77 ± 5.47	-	29.13 ± 5.10	24.96 ± 4.85
	MPG	16.86 ± 3.69	-	20.80 ± 3.88	5.51 ± 1.60
	ESL	10.01 ± 2.97	10.08 ± 2.47	7.44 ± 2.35	5.42 ± 2.18
	MMG	16.98 ± 2.79	-	17.34 ± 2.65	15.84 ± 2.51
	ERA	20.31 ± 2.50	18.56 ± 2.60	23.56 ± 2.92	28.13 ± 2.80
	LEV	16.16 ± 2.22	13.59 ± 1.85	15.72 ± 2.22	13.14 ± 1.76
	CEV	9.66 ± 1.74	-	7.99 ± 1.32	2.73 ± 0.89
	ASA	1.16 ± 1.74	-	2.11 ± 1.02	-

Table 3.3: Average and standard deviation of the AUC (in percent) of the test set for learning sets of different sizes.

Size	Data set	META	MIP	UTADIS	CR
20 %	DBS	87.61 ± 4.62	86.37 ± 4.63	88.86 ± 4.96	92.90 ± 3.22
	CPU	95.31 ± 2.47	94.97 ± 2.62	97.89 ± 2.83	98.22 ± 1.21
	BCC	68.10 ± 4.58	71.55 ± 3.65	66.50 ± 5.27	64.00 ± 6.41
	MPG	83.37 ± 2.91	82.15 ± 3.68	81.62 ± 3.35	97.88 ± 1.60
	ESL	95.69 ± 1.14	95.10 ± 1.66	97.04 ± 0.95	96.70 ± 0.74
	MMG	88.28 ± 1.29	88.77 ± 1.51	86.50 ± 2.94	88.67 ± 1.23
	ERA	72.56 ± 2.38	71.82 ± 3.28	74.09 ± 1.75	76.69 ± 3.34
	LEV	85.30 ± 2.58	84.24 ± 2.91	87.07 ± 1.46	89.71 ± 0.98
	CEV	76.13 ± 3.69	-	83.29 ± 2.47	98.25 ± 0.80
	ASA	98.11 ± 1.25	-	98.73 ± 0.90	-
50 %	DBS	90.74 ± 3.66	89.98 ± 3.36	93.25 ± 3.45	93.41 ± 2.28
	CPU	97.01 ± 1.40	96.45 ± 1.94	99.40 ± 1.31	99.20 ± 0.73
	BCC	69.29 ± 3.98	-	66.50 ± 52.7	69.12 ± 4.69
	MPG	83.37 ± 2.31	-	82.72 ± 2.43	98.18 ± 0.75
	ESL	96.40 ± 0.99	95.63 ± 1.14	97.47 ± 1.16	97.20 ± 0.84
	MMG	88.62 ± 1.38	-	86.67 ± 3.85	90.03 ± 1.32
	ERA	73.66 ± 2.33	71.67 ± 2.74	74.37 ± 2.11	77.05 ± 3.10
	LEV	87.21 ± 1.47	85.11 ± 2.19	87.46 ± 1.37	90.98 ± 1.03
	CEV	75.72 ± 2.82	-	84.50 ± 1.92	99.12 ± 0.24
	ASA	99.21 ± 0.72	-	99.48 ± 0.34	-
80 %	DBS	90.19 ± 6.06	90.80 ± 6.73	94.76 ± 4.01	94.27 ± 4.43
	CPU	97.21 ± 2.19	96.56 ± 2.37	99.89 ± 0.30	99.71 ± 0.63
	BCC	70.56 ± 8.64	-	66.51 ± 6.59	73.49 ± 6.92
	MPG	86.13 ± 3.41	-	82.10 ± 4.34	98.55 ± 1.08
	ESL	96.13 ± 1.70	95.68 ± 1.65	97.78 ± 1.17	97.66 ± 1.50
	MMG	88.60 ± 2.65	-	86.82 ± 4.70	91.35 ± 2.33
	ERA	73.79 ± 3.51	72.42 ± 4.77	74.97 ± 4.02	76.70 ± 2.90
	LEV	86.63 ± 2.65	84.99 ± 3.32	87.41 ± 2.17	91.22 ± 2.02
	CEV	75.74 ± 3.91	-	85.00 ± 2.46	99.59 ± 0.27
	ASA	99.55 ± 0.64	-	99.64 ± 0.34	-

In order to see whether the algorithm proposed in this paper can be useful in the context of preference learning problems, we compare our results with two other methodologies: UTADIS (Jacquet-Lagrèze and Siskos, 1982; Doumpos and Zopounidis, 2002) and CR (Tehrani et al., 2012).

The performance of MR-Sort algorithms is close to the performance of UTADIS and CR for the DBS, CPU, BCC, MMG, LEV and CEV data sets. The 0/1 losses observed on the test set differ by at most 4% on average. For the MPG data set, CR clearly returns better average results (more than 10% better in terms of 0/1 loss) than the MR-Sort algorithms and UTADIS. This may be due to the capability of the Choquet integral to represent interactions between criteria (see e.g. Grabisch and Roubens, 2000). The assignments in the MPG data set might require this type of modeling feature.

In contrast, for the ERA data set, the MR-Sort algorithms and UTADIS are definitely better than CR. Their advantage regarding the average 0/1 loss amounts to almost 8%.

As compared with UTADIS and CR, the average AUC value of the MR-Sort algorithms are worse. The difference is about 5% for DBS, CPU, BCC, ESL, MMG and ERA data sets. For the MPG and CEV data sets, there is a marked advantage of CR over the other algorithms. We have seen that CR is also definitely better regarding 0/1 loss for the MPG data set, which suggests that the model underlying CR is better suited for representing the MPG data.

On the contrary, the average AUC of the ERA data set, for which the MR-Sort MIP and metaheuristic did better than CR in term of 0/1 loss, is worse with the MR-Sort MIP and metaheuristic than with CR. UTADIS does even better than MR-Sort algorithms in terms of AUC for this data set.

In order to better understand the latter results, the confusion matrices for the MR-Sort algorithms and UTADIS and all learning set sizes are displayed in Table 3.4 for the ERA data set. The confusion matrices relative to the other data sets can be found in Appendix A. These show the average distribution of the alternatives in the test sets in actual (C^1, C^2) versus predicted classes ($\hat{C}^1$, $\hat{C}^2$). As compared with UTADIS, the MR-Sort algorithms classify correctly, on average, a higher number of instances from class C^1 and a lower number of instances from class C^2. We also notice that there are, on average, more alternatives belonging to class C^2 than to class C^1. Alternatives misclassified by the MR-Sort algorithms mostly belong to category C^2. It is likely that the concordance index of some of these alternatives is equal or lower than the one of some alternatives which are correctly classified in C^1. The contribution of these alternatives to the AUC index is therefore equal to 0.5 or 0, which decreases the value of the AUC.

It should be noted that the MR-Sort algorithms are designed in order to minimize the 0/1 loss. They do not include specific mechanisms taking into

Table 3.4: Confusion matrices of the test set for the (binarized) ERA data set. Actual class in rows, predicted class in columns.

(a) META - ERA 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	68.83 ±3.11	5.67 ±3.21
C^2	15.69 ±2.39	9.81 ±2.16

(b) MIP - ERA 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	69.38 ±2.41	5.12 ±2.47
C^2	15.81 ±1.74	9.70 ±1.54

(c) UTADIS - ERA 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	63.98 ±2.67	10.54 ±2.96
C^2	13.14 ±1.56	12.34 ±1.25

(d) META - ERA 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	69.26 ±2.61	5.10 ±2.55
C^2	15.46 ±2.16	10.17 ±1.73

(e) MIP - ERA 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	71.23 ±2.01	3.34 ±1.74
C^2	16.24 ±1.41	9.18 ±1.11

(f) UTADIS - ERA 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	64.36 ±2.12	9.98 ±2.62
C^2	13.43 ±1.75	12.22 ±1.32

(g) META - ERA 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	69.60 ±3.42	5.03 ±2.38
C^2	15.28 ±2.89	10.09 ±2.26

(h) MIP - ERA 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	72.59 ±2.57	2.29 ±1.00
C^2	16.27 ±2.36	8.86 ±1.77

(i) UTADIS - ERA 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	63.90 ±3.17	10.21 ±2.85
C^2	13.35 ±2.58	12.55 ±2.32

account possible imbalance of classes in the learning set, which has an impact on AUC.

Comments

Computing time becomes quickly an issue with the MIP when the size of the learning set increases. It is therefore not an option to use it to deal with large data sets, which, in contrast, can easily be handled by the metaheuristic.

The metaheuristic we developed performs better than UTADIS and CR for at least one data set (ERA). The same observation holds for the MIP. Regarding the 0/1 loss, we note that the MR-Sort model seems particularly well adapted for the ASA data set. This shows that for some types of data sets, a model-based approach like MR-Sort is well suited.

3.5.3 More than two categories

Some of the data sets involving more than two categories, namely CPU, ERA, LEV and CEV, were binarized to allow for comparison with the CR algorithm in the previous section. In this section we use these data sets with their original number of categories. All these data sets have 4 categories, except LEV which involves 5 categories.

Experimental setup

The same testing procedure as for the binary classification is applied. A model is learned on the basis of a subset of assignment examples, the learning set, and the alternatives in the test set are assigned by means of the learned model. These assignments are compared to the original classification.

When more than two categories are involved, formula (3.3) is no longer suited for computing the AUC. To obtain an AUC value for models with more than two categories, we proceed similarly as in Doumpos et al. (2009) and Waegeman et al. (2008). We denote by $A^{\leq h}$ (resp. $A^{>h}$), the subset of alternatives that are assigned to a category below or equal to (resp. above) C^h. The value AUC_h is computed for $h = 1, \ldots, p-1$, as follows:

$$AUC_h = \frac{1}{|A^{\leq h}| \cdot |A^{>h}|} \sum_{a^i \in A^{>h}} \sum_{a^k \in A^{\leq h}} \tau_h(a^{(i)}, a^{(k)}), \tag{3.4}$$

with

$$\tau_h(a^i, a^k) = \begin{cases} 0 & \text{if } \sum_{j:a_j^{(i)} \geq b_j^h} w_j < \sum_{j:a_j^{(k)} \geq b_j^h} w_j \\ 0.5 & \text{if } \sum_{j:a_j^{(i)} \geq b_j^h} w_j = \sum_{j:a_j^{(k)} \geq b_j^h} w_j \\ 1 & \text{if } \sum_{j:a_j^{(i)} \geq b_j^h} w_j > \sum_{j:a_j^{(k)} \geq b_j^h} w_j. \end{cases}$$

The value of AUC is computed as the average value of the AUC_h for $h = 1, \ldots, p-1$.

Results

For each data set, the average 0/1 loss of the test set is displayed in Table 3.5 and the AUC in Table 3.6. Note that for more than 2 categories the MIP version of the MR-Sort algorithm cannot be solved in a reasonable time and therefore, it does not appear in the tables. We observe that the average 0/1 loss is higher than in the binary case, both for the MR-Sort metaheuristic and UTADIS when there are more than two categories. This is not surprising since on the one hand, the data are more complex and, on the other hand, we chose to use a single

Table 3.5: Average and standard deviation of the 0/1 loss (in percent) on the test set for different sizes of the learning set with the original number of categories.

Size	Data set	META	UTADIS
20 %	CPU	24.57 ± 4.79	13.21 ± 4.88
	ERA	49.15 ± 1.76	51.44 ± 1.69
	LEV	46.52 ± 3.14	42.25 ± 1.75
	CEV	23.92 ± 1.73	22.81 ± 1.53
	ASA	7.12 ± 1.91	9.23 ± 1.57
50 %	CPU	19.61 ± 3.54	6.60 ± 2.66
	ERA	48.77 ± 2.33	51.67 ± 1.71
	LEV	43.48 ± 2.58	41.11 ± 1.58
	CEV	23.36 ± 1.93	22.86 ± 1.58
	ASA	4.82 ± 1.28	7.13 ± 1.15
80 %	CPU	20.76 ± 6.06	4.88 ± 3.51
	ERA	48.25 ± 3.74	51.76 ± 3.32
	LEV	43.17 ± 4.12	40.67 ± 3.05
	CEV	22.76 ± 2.52	22.83 ± 2.59
	ASA	4.32 ± 1.71	7.09 ± 1.61

set of weights for all categories. An alternative option would indeed consist in learning a series of binary classifiers that restore the classification in more than two categories. In such a case, each binary classifier would use its own set of weights. Such a model would certainly lead to smaller 0/1 loss values but the number of parameters would be larger and the interpretability of the model would be impoverished.

In terms of 0/1 loss, UTADIS performs better than the MR-Sort metaheuristic for the CPU and LEV data sets while it is the opposite for the two other data sets, ERA and CEV. Table 3.6 shows better AUC values obtained by UTADIS than by MR-Sort for all data sets.

We observe in Table 3.5 that the error rate on the ERA data set was smaller with the MR-Sort metaheuristic than with UTADIS and, inversely, that the AUC was better with UTADIS. The confusion matrices in Table 3.7 show us that the MR-Sort algorithm, on average, classifies correctly more alternatives of the categories C^1 and C^2 than UTADIS while it is the contrary for alternatives of categories C^3 and C^4.

The confusion matrices for all data sets are available in Appendix A. They show that the assignments obtained with the MR-Sort metaheuristic tend to be more pessimistic than those obtained with UTADIS. In the confusion matrices,

Table 3.6: Average and standard deviation of the AUC (in percent) on the test set for different sizes of the learning set with the original number of categories.

Size	Data set	META	UTADIS
20 %	CPU	96.49 ± 1.50	98.36 ± 2.75
	ERA	76.44 ± 2.05	79.86 ± 0.90
	LEV	84.00 ± 1.81	87.91 ± 0.85
	CEV	87.16 ± 1.39	91.25 ± 1.09
	ASA	97.89 ± 0.65	98.64 ± 0.48
50 %	CPU	97.47 ± 0.83	99.48 ± 1.34
	ERA	77.21 ± 1.97	80.27 ± 1.32
	LEV	85.46 ± 1.59	87.98 ± 1.31
	CEV	87.32 ± 0.98	91.19 ± 1.10
	ASA	98.54 ± 0.56	99.14 ± 0.39
80 %	CPU	97.39 ± 1.26	99.88 ± 0.20
	ERA	76.90 ± 3.07	80.76 ± 2.44
	LEV	85.81 ± 2.40	88.31 ± 2.78
	CEV	87.23 ± 1.58	91.42 ± 1.45
	ASA	98.78 ± 0.61	99.32 ± 0.35

we see that the MR-Sort model makes less errors of more than one category than UTADIS for the learning alternatives belonging to the classes C^1 and C^2. In the set of alternatives misclassified by the MR-Sort algorithm, more examples are assigned to lower categories and less are assigned to higher categories than when using UTADIS. We note that MR-Sort uses the pessimistic assignment rule of ELECTRE TRI which may explain that behavior.

3.6 Chapter conclusion

In this chapter we have presented a metaheuristic dedicated to the inference of the parameters of a MR-Sort model. We described two variants of the metaheuristic: one that maximizes the classification accuracy of the learning set and another that maximizes the area under the curve of the learning set.

Each component of the metaheuristic has been tested with artificial data sets. We tested the model retrieval, the tolerance for errors and we measured the computing time required to learn the parameters.

Experiments have shown that the computing time does not become prohibitive when the size of the learning set increases. The metaheuristic is therefore able to deal with large data sets similar to the ones found in the preference learning

Table 3.7: Confusion matrices for the test set of the ERA data set.

(a) META - ERA 20 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	29.16 ±3.92	10.09 ±4.01	1.91 ±1.36	0.39 ±0.42
C^2	13.47 ±4.34	14.55 ±4.21	4.44 ±2.84	0.49 ±0.69
C^3	5.91 ±1.66	8.05 ±2.53	5.11 ±2.19	1.55 ±1.17
C^4	0.32 ±0.19	0.83 ±0.49	1.71 ±0.80	2.03 ±0.63

(b) UTADIS - ERA 20 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	28.35 ±1.84	8.11 ±2.41	4.45 ±1.43	0.54 ±0.38
C^2	13.51 ±2.28	9.75 ±2.59	8.97 ±2.47	0.81 ±0.72
C^3	5.84 ±1.00	4.66 ±1.56	7.89 ±1.63	2.23 ±1.15
C^4	0.21 ±0.12	0.36 ±0.21	1.82 ±0.69	2.50 ±0.58

(c) META - ERA 50 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	29.76 ±4.29	9.82 ±4.58	1.60 ±1.07	0.23 ±0.34
C^2	14.32 ±5.30	14.26 ±5.00	4.14 ±2.22	0.24 ±0.54
C^3	6.22 ±1.99	7.97 ±2.78	5.34 ±2.13	1.23 ±1.21
C^4	0.31 ±0.26	0.74 ±0.43	1.96 ±0.84	1.87 ±0.56

(d) UTADIS - ERA 50 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	27.81 ±1.51	9.02 ±1.55	4.24 ±0.97	0.42 ±0.23
C^2	13.33 ±2.04	10.08 ±1.68	9.07 ±1.71	0.61 ±0.34
C^3	5.84 ±1.03	4.40 ±1.17	8.20 ±1.16	2.10 ±0.96
C^4	0.17 ±0.15	0.42 ±0.25	1.79 ±0.73	2.52 ±0.68

(e) META - ERA 80 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	29.36 ±4.43	10.24 ±4.44	1.44 ±1.15	0.21 ±0.38
C^2	13.70 ±5.52	15.04 ±5.53	4.52 ±2.96	0.14 ±0.44
C^3	5.99 ±2.24	7.94 ±2.82	5.62 ±2.52	1.08 ±1.38
C^4	0.27 ±0.37	0.85 ±0.61	1.90 ±1.03	1.74 ±0.88

(f) UTADIS - ERA 80 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	28.18 ±2.65	8.96 ±1.96	4.06 ±1.42	0.38 ±0.40
C^2	13.71 ±2.55	9.95 ±2.43	9.05 ±2.42	0.61 ±0.59
C^3	6.05 ±1.59	4.14 ±1.34	8.38 ±1.96	1.95 ±1.15
C^4	0.09 ±0.19	0.41 ±0.39	1.88 ±1.04	2.25 ±0.94

field.

Testing the model retrieval of each component amounts to determine the number of examples that are needed to find the parameters of a MR-Sort model reflecting as well as possible the preferences of a DM. The results of the experiments with artificial data sets showed that 400 assignment examples enable to restore 95% of the assignments of a test set composed of 10000 alternatives for a model composed of 3 categories and 10 criteria.

Testing the tolerance for errors aims at determining how the algorithm behaves when there are errors in the data sets. The results showed that the metaheuristic was able to restore a large number of examples even in the presence of errors. We observed that the classification accuracy of the learning set was converging to $1 - P$, with P being the proportion of errors in the learning set. The alternatives that were wrongly restored by the metaheuristic are mainly altered examples.

We also tested the idiosyncratic behavior of the metaheuristic, i.e. its ability to find a model restoring the assignments obtained with another sorting model. To test the idiosyncratic behavior, an AVF-Sort model has been used. We observed that the metaheuristic was able to restore on average more than 90% of the assignments when the learning set is composed of 1000 examples assigned with an AVF-Sort model.

Finally, we assessed the metaheuristic with real data sets issued from the preference learning field. We compared the results obtained with the metaheuristic to the results obtained with an exact formulation of the problem with a MIP. We also compared the results to the ones obtained with other PL and MCDA algorithms. The first observation is that our heuristic provides good approximations of the MR-Sort model that can be learned by an exact method (MIP), whenever the latter can be computed in a reasonable amount of time. For most data sets, the classification performance of our algorithm is close to the best results obtained by state-of-the-art algorithms. It is definitely better for one of the data sets. Since the MR-Sort model relies on a specific form of regularity in the assignments, it is not surprising that some data sets can be better approximated using our algorithm than some others. What is worth noticing, actually, is that our heuristic behaves competitively on the whole benchmark.

A remarkable feature of the model and of the algorithm is related to its ability to yield multi-category assignments in one step. It allows to dispense with learning a series of binary classifiers in order to restore a classification in more than two categories.

Another positive feature of the MR-Sort model stems from the fact that the computed classifications can be explained to the user as the application of a compact and intuitive rule. This is linked with the origins of the model which has been initially used in preference *modeling* and decision aiding. In those domains,

preferences are modeled by engaging into interactions with a decision maker (instead of being learned automatically on the basis of examples). Therefore, the preference models rely on intuitive concepts (e.g. limit profile, weights, majority threshold), which are used in the preference elicitation process. The resulting rules for comparing or sorting objects can be formulated in terms of the same concepts, which allows to explain their consequences to the DM. Understanding the model issued from an algorithm is likely to increase the trust of the user in the obtained classifier. Explainability is important e.g. in medical applications but also in management and engineering applications.

Chapter 4

Case study: preoperative patient classification

In this chapter, we present a medical application in the field of anesthesia. The application consists in determining automatically the score reflecting the health of a patient based on patient's characteristics on several attributes. Then the patient score is used in conjunction with other criteria to determine whether or not he/she is accepted for surgery. We use the metaheuristic described in Chapter 3 in order to find the parameters of a majority rule sorting (MR-Sort) model that enables to predict the health score of a patient and whether or not he/she should be accepted for surgery. We compare the results with other machine learning (ML) algorithms. Finally we describe the parameters of one MR-Sort model and show that it is easily interpretable by doctors.

4.1 Context

In recent years, the principal challenges related to the field of anesthesia and intensive care consists in reducing both anesthetic risks and mortality rate, as well as providing better and more efficient assistance to doctors specialized in anesthesia (DSA).

With regards to anesthesiology, assessing the patient's physical health state is a crucial step before deciding whether or not he/she can be accepted for surgery. This information allows DSA to identify the anesthesia type and determine the ease of the tracheal intubation. The American society of anesthesiologists (ASA) proposed a commonly used system to determine the patient's health state which is called ASA physical status classification system. It consists of classifying patients in one of the six categories going from "healthy person" to "brain-dead person

whose organs are being removed for donor purposes". The scale is composed of the following categories:

ASA 1 : Healthy person,

ASA 2 : Mild systemic disease,

ASA 3 : Severe systemic disease,

ASA 4 : Severe systemic disease that is a constant threat to life.

ASA 5 : A moribund person who is not expected to survive without the surgery.

ASA 6 : A declared brain-dead person whose organs are being removed for donor purposes.

Up to now, the classification of a patient does not rely on a well-established and objective method but on the subjective advice of one or several doctors. That's why we propose here to use a multiple-criteria decision analysis (MCDA) method in order to determine the ASA score and acceptance or refusal of the patient for surgery.

In medicine, MCDA methods can be used for various applications going from cancer diagnosis to health care settings. Among medicine applications based on decision aid methods, we observe that many of them are using the AHP (analytic hierarchy process) method (Liberatore and Nydick, 2008). Few of them are using other families of MCDA methods like outranking ones or methods based on additive value function.

In anesthesiology scientific literature, very few works related to preoperative patient classification were carried out. In Lazouni et al. (2013a,b), this issue was dealt with by using several machine learning algorithms. A drawback of such algorithms is that they are not easily interpretable by doctors. Indeed, as said in Section 2.7, these algorithms are used as blackboxes and are difficult to interpret. It is sometimes difficult to understand the patient classification by referring to the algorithm parameters. For instance, with a neural network algorithm like the multilayer perceptron, it is difficult to understand the classification by looking at synapses values and activation functions, even more if the model involves many variables.

As far as we know and according to our exploration of the existing literature, there is no work dealing with multiple criteria decision analysis and ASA physical status classification.

In this chapter, we propose to use the MR-Sort MCDA procedure to determine the ASA score of a patient evaluated on several attributes. We suggest to learn the parameters of MR-Sort models on the basis of a large database containing information about 898 patients evaluated on multiple criteria and who have been

classified in one of the six ASA classes. The idea behind is to obtain a set of models that can be easily interpreted by doctors. Note that our database contains only four of the six ASA score classes, given that ASA scores 5 and 6 have not been collected because the hospitals from which we collected our data are not included within the ones donating organ.

4.2 Literature review

In this section, we give an overview of existing works involving MCDA in medicine. Then we list a set of the decision support systems that have been developed in anesthesiology.

4.2.1 Multiple criteria decision analysis in medicine

In medicine, the decision aid methods are used for various applications going from cancer diagnosis and treatment (West et al., 2005; Carter et al., 1999), to the selection of technologies in health care settings (Chatburn and Primiano, 2001). Many medical applications are based on AHP. Liberatore and Nydick (2008) presented an overview of existing applications using AHP methods. However, few articles are dealing with other MCDA methods like the ELECTRE methods. Figueira et al. (2011) used the ELECTRE TRI-C sorting procedure in the context of assisted reproduction. Couples are assigned to categories which correspond to the number of embryos that have to be transferred back to the uterus of the woman in order to obtain a single pregnancy. To the best of our knowledge, there are no works dealing with the multiple criteria decision analysis and ASA score determination by using the above mentioned method while the ASA score is widely used by all DSA during their pre-anesthetic examinations.

Since in medicine it is preferred to use well-know methods that allow to explain a choice (Goodman and Ahn, 1999), we advocate the use of an outranking model, based on MR-Sort, and the use of an additive value function model.

4.2.2 Decision support systems for anesthesia

Glance et al. (2012) proposed a probabilistic model to evaluate the surgical mortality. The objective of this work is to predict the patient's mortality after a non-cardiac surgery, in order to diminish the operation risks. This system calculates the risk score in an empirical way by using three descriptors, which are the ASA score, the type of surgery either of high or intermediate risk and whether or not the surgery is urgent. A huge database composed of 298,772 patients has been gathered from different hospitals between 2005 and 2007, resulting as follows:

- patients with a risk score under 5 had a mortality risk under 0.5%;
- patients with a risk score between 5 and 6 had a mortality risk between 1.5% and 4%;
- patients with a risk score over 6 had a mortality risk of more than 10%.

An automatic system capable of predicting the operation anesthetic risk has been developed by Karpagavalli et al. (2009). This system assesses three classifications techniques based on supervised learning. The assessment has been done by using the WEKA software for the three following classification techniques: classification and regression trees, neural networks and Bayesian naïve classification. The database used contained 362 patients evaluated on 37 descriptors.

Lutz (2008) developed an automatic system classifying patients in different anesthetic risk levels. A modified version of the method has been presented by Hussmann and Russel (1997). One of the descriptors used to predict the patient's risk level is the ASA score of the patient.

Donati et al. (2004) developed a new model in order to predict the operative risk for the patients. The objective of this work was to predict the mortality and the morbidity of patients on the basis of the ASA classification. A database composed of 1936 patients built on the input of two hospitals had been used to predict the operative risks by means of a machine learning algorithm, called logistic regression.

Several machine learning algorithms have been used by Lazouni et al. (2013a) in order to predict patients' ASA score. Lazouni et al. (2013b) used five supervised machine learning techniques: support vector machine (SVM), radial basis function (RBF), C4.5 decision tree classifier, k-nearest neighbor (KNN) and multilayer perceptron (MLP). The comparison of the supervised machine learning algorithms done by Lazouni et al. (2013b) shows that the C4.5 decision tree classifier gives the best results for ASA score prediction. An additional algorithm used in this work is majority voting which consists of assigning the patient to the category to which it has been assigned by a majority of the 5 other machine learning algorithms. This last algorithm allows to obtain even better results than with the C4.5 decision tree classifier. A database containing 898 patients evaluated on multiple attributes has been used to evaluate the classifiers. The paper first deals with the determination of the ASA score of each patient. The second concern consisted of determining whether or not the patients are accepted for surgery. The third one was the selection of the type of anesthetic method either general or local. The last one consisted of determining whether the patient's tracheal intubation is easy or hard. For each algorithm and each concern, cross validation was performed.

4.2.3 Performance of machine learning algorithms for the determination of the ASA score

In the present work, we are interested in the determination of the ASA score and patient acceptance or refusal for surgery. We remind the main results of Lazouni et al. (2013a) regarding these two cases. The attributes taken into account in order to learn a model predicting the ASA score are listed in Table 4.1. The table describes the domain of each attribute and whether they should be maximized and/or minimized. For the determination of patient acceptance or refusal for surgery, 3 attributes, given in Table 4.2, are taken into account.

Table 4.1: List of attributes taken into account for the prediction of the ASA score in Lazouni et al. (2013a).

Attribute	Domain (Unit)	Direction
Age	[0-105] (year)	min.
Diabetic	{0,1}	min.
Hypertension	{0,1}	min.
Respiratory failure	{0,1}	min.
Heart failure	{0,1}	min.
Heart rate	[55-123] (bpm)	max. min.
Heart rate steadiness	{0,1}	max.
Pacemaker	{0,1}	min.
Atrioventricular block	{0,1}	min.
Left ventricular hypertrophy	{0,1}	min.
Oxygen saturation	[43-100] (%)	max.
Blood glucose level	[0.5-3.8] (g/l)	max. min.
Systolic blood pressure	[9-20.5] (cm Hg)	min.
Diastolic blood pressure	[5-13] (cm Hg)	min.

Table 4.3 shows the average classification accuracy of the test set for the prediction of the ASA score and acceptation or refusal of patient for surgery

Table 4.2: List of attributes taken into account for the prediction of acceptance or refusal of a patient for surgery in Lazouni et al. (2013a).

Attribute	Domain (Unit)	Direction
ASA score	[0-4]	min.
Cerebrovascular accident	[0-2]	min.
Myocardial infarction	[0-2]	min.

when 70 % of the data set is used as learning set. The method returning the best results for both cases is majority voting. It restores 93.59% of the assignments for ASA score prediction and 94.07% of the assignments for acceptance or refusal for surgery.

Table 4.3: Average classification accuracy of the test set when 70% of the examples in the data set are used as learning set for the prediction of ASA score and acceptance/refusal for surgery.

Learning algorithm	ASA score	Acceptance/Refusal
SVM	0.8752	0.9142
C4.5	0.9154	0.9012
KNN	0.8468	0.9085
MLP	0.8927	0.9292
RBF	0.8333	0.8981
Majority voting	0.9259	0.9407

4.3 Using the majority rule sorting model for the prediction of the ASA score and patient acceptance or refusal for surgery

Compared to other machine learning algorithms, the MR-Sort rule can be more easily interpreted. It is possible to describe the model as a set of simple rules. In this chapter, we use the MR-Sort metaheuristic presented in Chapter 3 to learn the parameters of MR-Sort models predicting the ASA score of a patient and whether or not he/she is accepted for surgery. To address the two cases, we reuse the data set of Lazouni et al. (2013a) which involves 898 patients. Table 4.4 gives the distribution of the patients in the data set among the first four ASA classes. No patient has an ASA score above 4 and a majority of them has an ASA score below 3.

The ASA score of a patient is determined based on the 14 attributes in Table 4.1. The acceptance or refusal for surgery is determined on the basis of 3 attributes (see Table 4.2) including patient's ASA score.

As using a MR-Sort model requires attributes which are monotone, it implies that some attributes of Table 4.1 have to be modified in order to have a monotonic scale. Indeed, attributes "Heart rate" and "Blood glucose level" are not monotone. The preference for these attributes increases and then decreases as a function of the attribute value. As an example, a person with a heart rate of 70 beats per minutes (bpm) is preferred to someone who has a heart rate of 50 bpm and to

Table 4.4: Number of patients per ASA score.

ASA score	Number of instances (proportion in percents)
ASA 1	211 (23 %)
ASA 2	396 (44 %)
ASA 3	239 (27 %)
ASA 4	52 (6 %)

Table 4.5: Number of patients accepted and refused.

Patient status	Number of instances (proportion in percents)
Accepted	762 (85 %)
Refused	136 (15 %)

Table 4.6: Attributes split in two in order to determine the ASA score with a MR-Sort model.

Attribute		Domain (Unit)	Direction
Heart rate	Bradycardia	[50-70] (bpm)	max.
	Tachycardia	[70-123] (bpm)	min.
Blood glucose level	Hypoglycemia	[0.5-0.92] (g/l)	max.
	Hyperglycemia	[0.92-3.8] (g/l)	min.

someone with a heart rate of 100 bpm. In order to have criteria for which the preference either increases or decreases as a function of its value, the attributes are split in four sub-attributes: "Bradycardia", "Tachycardia", "Hypoglycemia" and "Hyperglycemia". Table 4.6 lists the four criteria and whether it should be maximized or minimized.

Attributes used to determine the acceptance or refusal of a patient for surgery (Table 4.2) are all monotone. There is no need to transform any of them.

4.4 Quality of ASA score and acceptance prediction using the majority rule sorting model

To assess whether or not MR-Sort gives better results than other machine learning algorithms, we perform a cross validation on the data set and compare our results to the ones obtained by Lazouni et al. (2013a). The cross validation is done by

using successively 30%, 50%, 70% of the database as learning set and the rest as test set. The split between learning and test alternatives is done at random. For a given size of learning and test sets, the cross validation is repeated 100 times, each time with different learning and test sets.

The comparison of the machine learning algorithms with the MR-Sort metaheuristic is done by measuring two indices:

1. classification accuracy (classification accuracy (CA)), see Equation (2.28);
2. area under the curve (area under the curve (AUC)), see Equation (3.3).

First, we assess the ability of the MR-Sort metaheuristic to return a model that is compatible with the highest number of examples. The results are given in Table 4.7. Compared to the results obtained with other machine learning algorithms (see Table 4.3), we observe that the classification accuracy with MR-Sort is significantly better. Indeed, the classification accuracy is improved by almost 4% with MR-Sort compared to the majority voting algorithm used in Lazouni et al. (2013a). We also note that the value of the area under the curve is high which means that the model can efficiently discriminate alternatives from different classes.

Table 4.7: Prediction of the ASA score: average classification accuracy of the learning and test sets for different size of learning set (30%, 50%, 70% of the data set).

		Learning set	Test set
CA	30%	0.9862 ± 0.0064	0.9469 ± 0.0124
	50%	0.9829 ± 0.0053	0.9553 ± 0.0101
	70%	0.9810 ± 0.0045	0.9615 ± 0.0129
AUC	30%	0.9958 ± 0.0029	0.9830 ± 0.0067
	50%	0.9950 ± 0.0022	0.9858 ± 0.0053
	70%	0.9943 ± 0.0021	0.9878 ± 0.0053

The same experiment is performed for the prediction of the patient acceptance or refusal for surgery. Table 4.8 shows the results obtained with the MR-Sort metaheuristic. We observe that the model is able to restore 92% of the examples. Compared to the majority voting algorithm used in Lazouni et al. (2013a), it is about 2% less efficient. Regarding the area under the curve, we note that the algorithm is less efficient than for the prediction of the ASA score.

In order to improve the efficiency of MR-Sort for the prediction of patient acceptance or refusal for surgery, we consider replacing the attribute ASA by the list of 14 attributes used to determine the ASA score of a patient. The

Table 4.8: Prediction of Patient Acceptance/Refusal for Surgery using three attributes for different sizes (30%, 50%, 70%) of the data set.

		Learning set	Test set
CA	30%	0.9268 ± 0.0121	0.9207 ± 0.0097
	50%	0.9252 ± 0.0084	0.9241 ± 0.0092
	70%	0.9259 ± 0.0055	0.9235 ± 0.0129
AUC	30%	0.7604 ± 0.0377	0.7509 ± 0.0162
	50%	0.7521 ± 0.0235	0.7513 ± 0.0246
	70%	0.7536 ± 0.0148	0.7507 ± 0.0346

experimental results are displayed in Table 4.9. We observe that the CA and AUC are substantially improved by replacing the ASA criterion by the 14 criteria used to determine it. The model gives a CA that is up to 3 percents better when the learning set grows. But the main difference lies in the value of the AUC which increases from 75% with 3 criteria to 91% with 16 criteria. Compared to the best results obtained with the machine learning algorithms in Lazouni et al. (2013b), we consistently observe a gain of more than one percent with MR-Sort.

It demonstrates that using more criteria helps to improve the quality of the model. Using the sole ASA attribute results in a descriptive loss which results in worse performances.

Table 4.9: Prediction of patient acceptance/refusal for surgery using 18 attributes for different sizes (30%, 50%, 70%) of the data set.

		Learning set	Test set
CA	30%	0.9794 ± 0.0086	0.9347 ± 0.0156
	50%	0.9701 ± 0.0063	0.9475 ± 0.0113
	70%	0.9668 ± 0.0049	0.9525 ± 0.0133
AUC	30%	0.9672 ± 0.0272	0.9129 ± 0.0338
	50%	0.9486 ± 0.0267	0.9188 ± 0.0277
	70%	0.9281 ± 0.0277	0.9085 ± 0.0377

4.5 Explaining predictions and interpretability

Machine learning algorithms often operate as blackboxes. It is difficult for the user to interpret the resulting models. Compared to ML algorithms, MR-Sort is a model whose parameters can be interpreted in order to explain the assignments.

In this subsection, we use the full ASA data set as learning set for the MR-Sort metaheuristic in order to learn models restoring as many examples as possible. We select one of the models learned with the metaheuristic and describe it.

4.5.1 Reduction of the number of attributes

To simplify the model as much as possible, we identify attributes having the least influence in the model. Therefore, we initialize 100 instances of the metaheuristic. Each instance is initialized at random with a population of 20 MR-Sort models. The metaheuristic is configured to run 20 times the heuristic improving the profiles and 20 times the heuristic adjusting the profiles. For each instance of the metaheuristic, we keep the model restoring the highest number of assignments. After running 100 instances of the metaheuristic, we obtain a list of 100 models. We observe that several criteria are not used in the models found by the metaheuristic. The histogram given in Figure 4.1 shows the number of times each criterion has been discarded among the 100 MR-Sort models. With 16 criteria, we observe that "Bradycardia", "Tachycardia" and "Hyperglycemia" are three attributes that are discarded in more than 75% of the models. It shows that some criteria do not add a lot of information for the determination of the ASA score.

To simplify the model, we apply a leave-one-out procedure to remove some attributes from the model. It consists in repeating the experiment by removing one attribute at a time from the data set. For a data set involving 16 attributes, we thus repeat the experiment 16 times, each time with another subset of 15 attributes. After applying the leave-one-out procedure, we compute the average classification accuracy of the models for the different subsets. Finally, we remove the attribute decreasing the least the classification accuracy. The same procedure is repeated for the 15 criteria and so on. We take care to keep attributes that are considered important by doctors for the determination of the ASA score. As an example, "Oxygen saturation" is not often used in the models but we choose to keep this criterion because doctors consider that it is an important parameter in the determination of the ASA score.

Figure 4.2 shows the evolution of the average classification accuracy (CA) and area under the curve (AUC) when using from 16 to 5 attributes in the model.

We observe that the classification accuracy and area under the curve slightly decrease when attributes are removed. The drop is more important when the model goes from 8 to 7 attributes and the classification accuracy declines with more than one percent. The area under the curve remains stable up to 7 criteria. It decreases with more than one percent when the number of attributes goes from 7 to 6. Going from 6 to 5 criteria results in a decrease of more than 3 percents of the AUC.

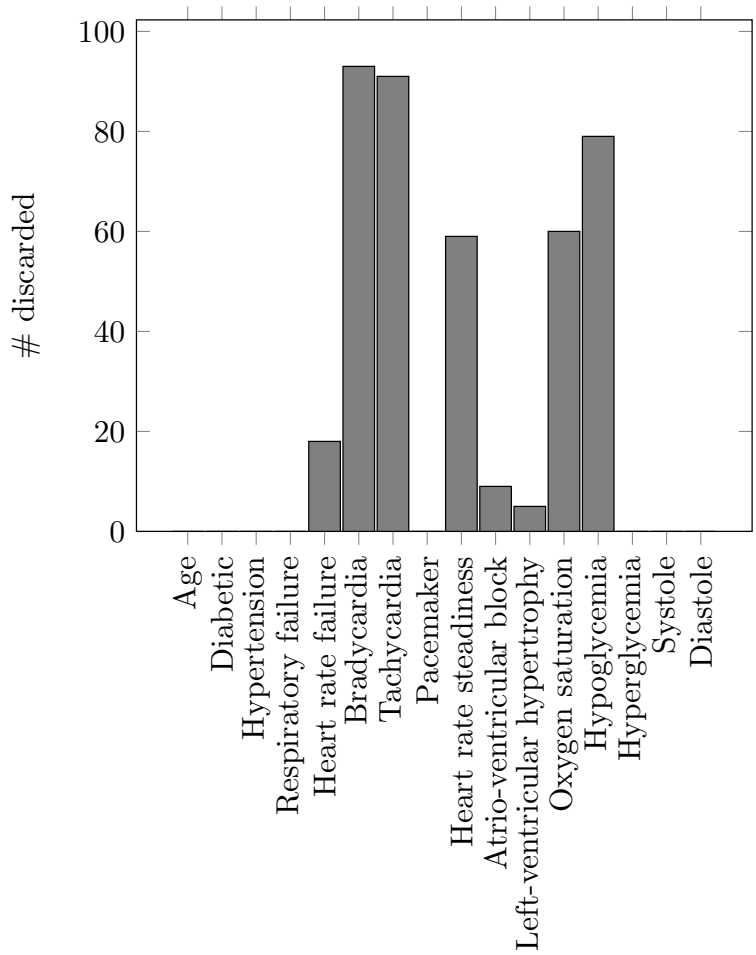


Figure 4.1: Number of times each criterion was discarded among the 100 best models corresponding to the 100 instances of the metaheuristic.

4.5.2 Interpretability of the model parameters

In general, a majority rule can be represented by different sets of additive weights and majority thresholds.

For instance, consider the set of additive weights and the majority thresholds in Table 4.10. Using these weights in a MR-Sort model composed of 2 categories C^1 and C^2, with $C^2 > C^1$, an alternative is assigned to category C^2 if it is at as good as the profile on two of the three criteria. Indeed, for each of these coalitions, we have:

- $w_1 + w_2 > \lambda$;
- $w_1 + w_3 > \lambda$;

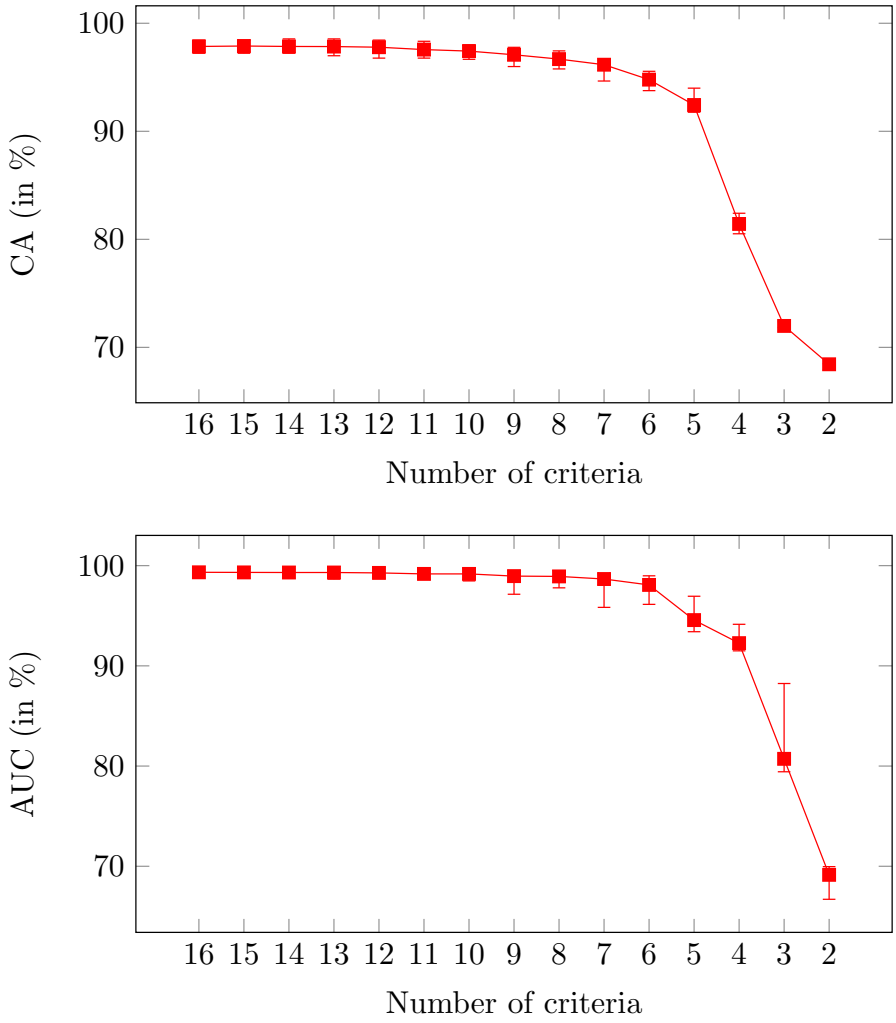


Figure 4.2: Evolution of the classification accuracy (CA) and area under the curve (AUC) when decreasing the number of attributes from 16 to 5.

Table 4.10: Example of a set of weights and majority threshold of a MR-Sort model. The weights are denoted by w_j with $j = \{1, 2, 3\}$ and the majority threshold is denoted by λ.

w_1	w_2	w_3	λ
0.49	0.11	0.40	0.50

- $w_2 + w_3 > \lambda$.

These criteria coalitions are called "winning coalitions" and we denote them by $\{w_1, w_2\}$, $\{w_1, w_3\}$ and $\{w_2, w_3\}$.

It is not easy to interpret the impact of each criterion by reading the weights and majority threshold given in Table 4.10. Actually, all criteria have the same decision power. The differences in the weights of the various criteria seem arbitrary. We conclude that these weights do not clearly show the relative importance of each criterion.

With a model involving 3 criteria, it is not difficult to identify the winning coalitions by enumerating all the possible coalitions of criteria and by verifying whether these coalitions fulfill the condition $\sum_{j=1}^{3} w_j \geq \lambda$. It makes in total 2^3 (8) criteria coalitions to check. However when the number of criteria increases, the number of coalitions to check quickly grows. For 7 criteria, it makes 2^7 (128) coalitions of criteria to check.

In order to get a set of weights that are more easily interpretable by a decision maker (DM), we use a post-treatment after learning the parameters of the MR-Sort model. The post-treatment consists in first identifying the winning coalitions and then running a mixed integer program (MIP) in order to obtain a set of weights that is compatible with the original rule and that reflects the power of each criterion.

Identifying the winning coalition is done by verifying the inequality $\sum_{j=1}^{n} w_j \geq \lambda$ for each possible coalition. We denote by $\mathcal{G}$ the set of winning coalitions, i.e. the ones fulfilling the condition $\sum_{j=1}^{n} w_j \geq \lambda$. Similarly $\mathcal{B}$ denotes the set of losing coalitions, i.e. the ones that are not fulfilling the above condition. These coalitions are then given to the MIP which infers a set of weights that is more easily interpretable by a DM. The MIP tries to maximize the slack between the majority threshold and winning and losing coalitions. It infers integer weights and majority threshold. The weights sum up to a pre-defined integer value. The MIP formulation is the following:

$$\max_{w_j, \lambda} \alpha$$

such that:

$$\begin{cases} \sum_{j\in\mathcal{C}} w_j - \alpha & \geq & \lambda & \forall \mathcal{C} \in \mathcal{G}, \\ \sum_{j\in\mathcal{C}} w_j + \alpha & \leq & \lambda - \epsilon & \forall \mathcal{C} \in \mathcal{B}, \\ \sum_{j=1}^{n} w_j & = & W, & \end{cases}$$

with:

$$\begin{cases} \lambda & \in & [0, W] \text{ (integer)}, & \\ w_j & \in & [0, W] \text{ (integer)} & \forall j \in N, \end{cases}$$

and ϵ a small positive value (e.g. 10^{-5}), W an integer value defined a priori.

After using the MIP with the rules defined by weights and majority threshold given in Table 4.10, we obtain the weights and majority threshold given in Table 4.11. We fixed the value of W to 100 so that the weights sum up to 100. The weights obtained with the MIP represent more equitably the importance of each criterion in the model. Indeed each criterion has an importance which is equal to more or less one third of the weight sum which means an equal importance of each criterion. The majority threshold is also chosen so that it is easy to state whether or not a coalition of criteria is better than the threshold or not.

Table 4.11: Weights and majority threshold obtained with the MIP finding a set of representative weights and majority threshold. The value of W was fixed to 100.

w_1	w_2	w_3	λ
34	33	33	51

The post-treatment presented in this subsection has been used on all the models found by the metaheuristic in the context of the case study presented in this chapter. By default, the value of W was fixed to 100. However, sometimes the MIP was not able to return a set of integer weights that sum up to 100 and fulfills the rules of the model. In that case, we multiply the value of W by ten and run the program again. We perform this operation as long as the MIP restores a set of weights that is compatible with the rules implied by the original set of weights and majority threshold.

Finding a set of additive weights and a majority threshold which represents the importance of the criteria in a model is something that deserves attention in our view. Indeed, we think that the cognitive effort of a DM to understand the model can be reduced if the parameters of the model are chosen appropriately. This question was debated during the 71st meeting of the European working group (EWG) on MCDA and deserves more attention.

4.5.3 Majority rule sorting model for the prediction of the ASA score

In agreement with doctors, we choose to keep a model using 7 attributes in order to predict the ASA score of a patient. The attributes taken into account are "Age", "Diabetic", "Hypertension", "Oxygen saturation", "Hyperglycemia", "Systole" and "Diastole". The metaheuristic instances provide several models having similar classification accuracy and area under the curve. Some of these models are given in appendix B.

Among the 100 best MR-Sort models obtained, we keep the one represented in Figure 4.3. This model can restore the ASA score of 96,21% of the patients in the learning set, with an AUC equal to 98.48%. The confusion matrix is given in Table 4.12.

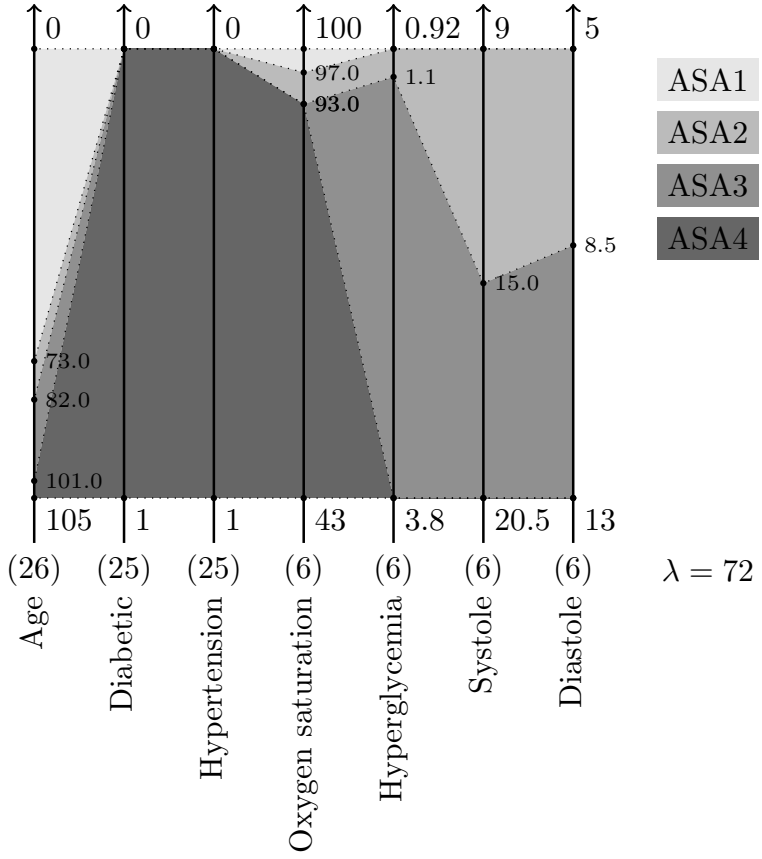


Figure 4.3: MR-Sort model for the prediction of the ASA score. Values between parentheses below the axis are the weights of the criteria.

Using MR-Sort, patient performances are compared to the profiles delimiting

Table 4.12: Prediction of patient ASA score: confusion matrix.

	$\widehat{\text{ASA 1}}$	$\widehat{\text{ASA 2}}$	$\widehat{\text{ASA 3}}$	$\widehat{\text{ASA 4}}$
ASA 1	202	9	0	0
ASA 2	11	382	3	0
ASA 3	6	5	228	0
ASA 4	0	0	0	52

the categories in ascending order, i.e. the comparison begins with the profile delimiting the worst category. To be assigned to a category, a patient should be at least as good as the lower profile of this category and not as good as its upper profile. In the model illustrated in Figure 4.3, a patient is as good as the profile if his/her performances are at least equal to those of the profile on each criterion of one of these four criteria coalitions:

1. {Age, Diabetic, Hypertension};
2. {Age, Diabetic, Hyperglycemia, Oxygen saturation, Systole, Diastole};
3. {Age, Hypertension, Hyperglycemia, Oxygen saturation, Systole, Diastole};
4. {Diabetic, Hypertension, Hyperglycemia, Oxygen saturation, Systole, Diastole}.

A patient is assigned to a category above ASA 4 if his/her performances are as good as the performances of the profile delimiting the category ASA 3 from ASA 4. In other words, a patient has always a score better than 4 if he/she satisfies the three following conditions:

1. he/she is 105 years old or younger;
2. he/she is not diabetic;
3. he/she doesn't suffer from hypertension.

The ASA score of a patient is also better than 4 if he/she satisfies two of these conditions in conjunction with a level of oxygen saturation equal or above 93%. On the contrary, a patient who does not satisfy two of the three conditions listed above is always assigned to category ASA 4. A patient is assigned to a category better than 3 if he/she satisfies the three following conditions:

1. he/she is 82 years old or younger;
2. he/she is not diabetic;

3. he/she doesn't suffer from hypertension.

The ASA score of a patient is also better than 3 if he/she satisfies two of the above conditions in conjunction with:

1. an oxygen saturation level equal or greater than 93%;
2. a small hyperglycemia characterized by a blood glucose level equal or smaller than 1.10 g/l;
3. a systole level equal or smaller than 15 cm Hg;
4. a diastole level equal or smaller than 8.5 cm Hg.

Finally, a patient is always classified in category ASA 1 if the following three conditions are met:

1. he/she is 73 years old or younger;
2. he/she is not diabetic;
3. he/she doesn't suffer from hypertension.

A patient is also assigned to ASA 1 if he/she satisfies two of these conditions in conjunction with:

1. an oxygen saturation level equal or greater than 97%;
2. No hyperglycemia, characterized by a blood glucose level equal or smaller than 0.92 g/l;
3. a systole level equal or smaller than 9.4 cm Hg;
4. a diastole level equal or smaller than 5.4 cm Hg.

4.5.4 Majority rule sorting model for the prediction of patient acceptance/refusal for surgery

The acceptance or refusal of a patient for surgery is made on the basis of his/her performance on the three criteria listed in Table 4.2. As for the prediction of the ASA score, we use the full data set as learning set to obtain a MR-Sort model interpretable by doctors. We run the metaheuristic a hundred times to obtain a set of models. By using the 898 patients of the database as learning set, we obtain an average classification accuracy equal to 0.9254 and an AUC equal to 0.7537. Among the 100 best models given by the metaheuristic instances, a large majority of them are identical. We show one of these models in Figure 4.4.

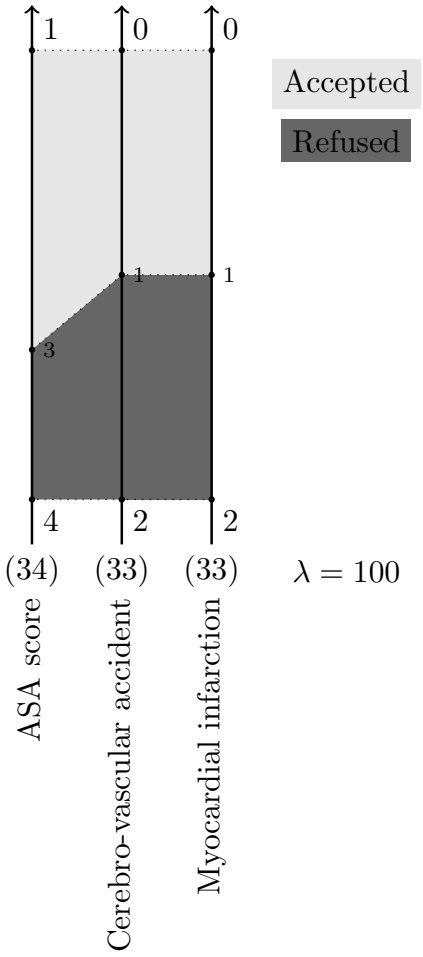


Figure 4.4: MR-Sort model for the prediction of the patient acceptance or refusal for surgery. Values in parentheses below the axis are the weights of the criteria.

Using this model, a patient is accepted for surgery if he/she has at least as good performance as the limiting profile on all the criteria since $\lambda = 100$.

Following this model, a patient is accepted for surgery if he/she fulfills the two following conditions:

1. his/her ASA score is better than 4;

2. he/she hasn't been subject to a cerebro-vascular accident or myocardial infarction.

The confusion matrix is given in Table 4.13. We note that the method is optimistic. All patients that should be accepted for surgery are correctly classified

by the model. However, the model accepts a large proportion of patients that should be refused for surgery. In the determination of patient acceptance or refusal for surgery, the use of a MR-Sort model seems to be less relevant than for the prediction of the ASA score.

Table 4.13: Prediction of patient acceptance/refusal for surgery: confusion matrix.

	$\widehat{\text{Accepted}}$	$\widehat{\text{Refused}}$
Accepted	762	0
Refused	67	69

4.6 Chapter conclusion

The development of the medical computer-aided diagnosis systems is becoming nowadays a very motivating research field. Numerous researchers working in the field of artificial intelligence are trying to build interpretable intelligent automatic systems able to help doctors in their routine clinical work.

In this chapter we have introduced a MCDA sorting method in a medical application. This system, which consists in the prediction of the ASA score and use it to decide whether the patient is accepted or refused for surgery, is designed mainly for doctors specialized in anesthesia and ease a great part of their pre-anesthetic examination.

The results obtained with MR-Sort showed an improvement of the classification accuracy compared to machine learning algorithms. The experiments showed that the accuracy was improved by more than 5 percents for the determination of the ASA score. For the prediction of patient acceptance or refusal for surgery, we observed that MR-Sort is not better than the ML algorithms when using the ASA score as an attribute. However, MR-Sort performs better when the ASA score is replaced by the sixteen attributes that are used to determine it.

As mentioned in Chapter 3, a MR-Sort model is easy to interpret compared to other ML algorithms. In this chapter we showed that the model can be described by a set of simple rules.

We conclude that the proposed methods can be practically used in pre-anesthetic examinations to help doctors specialized in anesthesia assess the risks for the patients. In our future work, we intend to enrich the database by adding patients that are not often represented in it (e.g. newborn children, patients with a pacemaker, etc.).

To simplify the model as much as possible and ease its interpretation, we reduced the number of attributes required to determine the ASA score. For the prediction of the patient acceptance or refusal for surgery we noticed that the

results were better if all the attributes involved in the prediction of the ASA score were used. Note that we can proceed as for the ASA score prediction to reduce the number of attributes and simplify the model.

Chapter 5

Learning the parameters of a non-compensatory sorting model

With a majority rule sorting (MR-Sort) model, the assignment of an alternative to a category is done by comparing its performances to the performances of the profiles delimiting the categories. In this method, an alternative is considered as good as a profile if it has at least equal performances as the profile on a weighted majority of criteria. The coalition of criteria in favor of an alternative against a profile is represented by the sum of weights of the criteria on which the alternative is better than the profile. This model does not handle criteria interactions since criteria weights are just summed up. In order to improve the expressiveness of the model, we substitute additive weights by capacities in order to represent the power of coalitions of criteria. In this chapter, we describe two manners to learn the parameters of such a model. The first one is based on mixed integer programming and the second one is an extension of the metaheuristic presented in Chapter 3. We perform experimental tests on the same data sets as in Chapter 3 and compare the results.

5.1 Motivations

In this chapter, we consider a sorting model that satisfies the requirements of the non-compensatory sorting model described in Section 2.2.3 and characterized by Bouyssou and Marchant (2007a,b). The model is a generalization of MR-Sort presented in Section 2.2.2. In MR-Sort, categories are separated by profiles which are vectors of performances on the different criteria. Each criterion of the model is associated a weight representing its importance or its voting power. Using this model, without veto, we assign an alternative to a category if it is considered at least as good as the category lower profile and not at least as good as the category

upper profile. An alternative is considered as good as a profile if its performances are at least as good as the profile performances on a weighted majority of criteria. In MR-Sort, the weighted majority of criteria is reached if the sum of weights of the criteria on which the alternative is at least as good as the profile is greater than a threshold.

Consider a MR-Sort model involving 4 criteria (c_1, c_2, c_3 and c_4) and 2 ordered categories ($C^2 > C^1$), separated by a profile b_1. Using this model, an alternative is assigned to the "good" category (C^2) iff its performances are as good as the profile b_1 on at least one of the four following minimal criteria coalitions: $c_1 \wedge c_2$, $c_3 \wedge c_4$, $c_1 \wedge c_4$ and $c_2 \wedge c_4$. A coalition of criteria is said to be minimal if removing any criterion is enough to reject the assertion "alternative a is as good as profile b". With an MR-Sort model, this can be achieved by selecting, for instance, the following weights and majority threshold: $w_1 = 0.3$, $w_2 = 0.2$, $w_3 = 0.1$, $w_4 = 0.4$ and $\lambda = 0.5$. It leads to the four following constraints:

$$\begin{cases} w_1 + w_2 & = & \lambda, \\ w_3 + w_4 & = & \lambda, \\ w_1 + w_4 & > & \lambda, \\ w_2 + w_4 & > & \lambda. \end{cases}$$

All the other coalitions of criteria, which are not supersets of the four minimal coalitions listed above, are not sufficient to be considered as good as b_1 (e.g. $w_1 + w_3 < \lambda$).

Assume now that we want a model for which the two minimal sufficient criteria coalitions are: $c_1 \wedge c_2$ and $c_3 \wedge c_4$. Modeling this classification rule with an MR-Sort model is impossible. To model these rules, we have to choose weights w_j, $j = 1, \ldots, 4$, summing up to 1, such that $w_1 + w_2 \geq \lambda$ and $w_3 + w_4 \geq \lambda$. Summing these two inequalities yields $1 \geq 2\lambda$. If we want these coalitions to be the only minimal sufficient ones, we must also have : $w_1 + w_3 < \lambda$, $w_1 + w_4 < \lambda$, $w_2 + w_3 < \lambda$ and $w_2 + w_4 < \lambda$. Summing these four inequalities yields $2 < 4\lambda$. Hence, there exist no weights and majority threshold for which the 2 above coalitions are the only two minimal sufficient coalitions. In order to be able to represent such a type of rule, we consider in this paper an extension of MR-Sort allowing to model interactions between criteria. This formulation expresses the majority rule of MR-Sort by using a capacity like in the Choquet integral (Grabisch, 1996). This model is called the non-compensatory sorting model (non-compensatory sorting (NCS) model). It was introduced and characterized by Bouyssou and Marchant (2007a,b).

In this chapter, we aim at studying the additional descriptive ability of the NCS model as compared to MR-Sort. We present two manner to infer the parameters of such a model on the basis of assignment examples. We assess this experimentally on real data sets.

5.2 Mixed integer program for learning the parameters of a non-compensatory sorting model

In this section we propose a mixed integer program (MIP) that has been developed in order to learn the parameters of a NCS model on the basis of a set of assignment examples and their associated vectors of performances. We first remind how the parameters of a MR-Sort model can be learned through linear programming (see Leroy et al., 2011). Then we modify the MIP in order to learn the parameters of a NCS model.

5.2.1 Mixed integer program for learning the parameters of a majority rule sorting model

Leroy et al. (2011) developed a mixed integer program taking a set of assignment examples and their vector of performances as input and finding the parameters of a MR-Sort model such that the largest possible number of examples are restored by the inferred model. We recall the main steps to obtain the MIP formulation.

We denote by A^* the set of examples given as input to the learning algorithm. The condition for an object a to be assigned to category C^h (Equation (2.9)) can be written as follows:

$$a \in C^h \iff \begin{cases} \sum_{j=1}^{n} c_{a,j}^{h-1} \geq \lambda & \text{with } c_{a,j}^{h-1} = \begin{cases} w_j & \text{if } a_j \geq b_j^{h-1}, \\ 0 & \text{otherwise,} \end{cases} \\ \sum_{j=1}^{n} c_{a,j}^{h} < \lambda & \text{with } c_{a,j}^{h} = \begin{cases} w_j & \text{if } a_j \geq b_j^{h}, \\ 0 & \text{otherwise.} \end{cases} \end{cases}$$

In order to make these constraints linear, we introduce for each variable $c_{a,j}^l$, with $l = \{h-1, h\}$, a binary variable $\delta_{a,j}^l$ that is equal to 1 when the performance of the object a is at least as good as or better than the performance of the profile b_l on criterion j and 0 otherwise. To obtain the value of $\delta_{a,j}^l$, we add the following constraints, where M is an arbitrary large positive constant (M should be chosen greater than $\overline{a_j} - \underline{a_j}$):

$$M(\delta_{a,j}^l - 1) \leq a_j - b_j^l < M \cdot \delta_{a,j}^l. \tag{5.1}$$

By using the value $\delta_{a,j}^l$, the values of $c_{a,j}^l$ are obtained as follows:

$$\begin{cases} c_{a,j}^l & \geq 0, \\ c_{a,j}^l & \leq w_j, \\ c_{a,j}^l & \leq \delta_{a,j}^l, \\ c_{a,j}^l & \geq \delta_{a,j}^l - 1 + w_j. \end{cases}$$

The objective function of the MIP consists in maximizing the number of examples compatible with the learned model, i.e. minimizing the 0/1 loss function. In order to model this, new binary variables, γ_a for all $a \in A^*$, are introduced. The value of γ_a is equal to 1 if object a is assigned to the expected category, i.e. the category it is assigned to in the learning set, and equal to 0 otherwise. To obtain the correct value of the γ_a variables, two additional constraints are added:

$$\begin{cases} \sum_{j=1}^{n} c_{a,j}^{h-1} & \geq \lambda + M(\gamma_a - 1), \\ \sum_{j=1}^{n} c_{a,j}^{h} & < \lambda - M(\gamma_a - 1). \end{cases}$$

The objective function of the MIP is then to maximize the sum of γ_a.

Finally, the combination of the objective with all the constraints leads to the following MIP:

$$\max_{w_j, \lambda, b_j^h} \sum_{a \in A^*} \gamma_a,$$

such that:

$$\begin{cases} \sum_{j=1}^{n} c_{a,j}^{h-1} & \geq & \lambda + M(\gamma_a - 1) & \forall a \in A^{*h}, h = \{2, \ldots, p\}, \\ \sum_{j=1}^{n} c_{a,j}^{h} & < & \lambda - M(\gamma_a - 1) & \forall a \in A^{*h}, h = \{1, \ldots, p-1\}, \\ a_j - b_j^l & < & M \cdot \delta_{a,j}^l & \forall j \in N, \forall a \in A^{*h}, \forall h \in H, l = \{h-1, h\}, \\ a_j - b_j^l & \geq & M(\delta_{a,j}^l - 1) & \forall j \in N, \forall a \in A^{*h}, \forall h \in H, l = \{h-1, h\}, \\ c_{a,j}^l & \leq & \delta_{a,j}^l & \forall j \in N, \forall a \in A^{*h}, \forall h \in H, l = \{h-1, h\}, \\ c_{a,j}^l & \leq & w_j & \forall j \in N, \forall a \in A^{*h}, \forall h \in H, l = \{h-1, h\}, \\ c_{a,j}^l & \geq & \delta_{a,j}^l - 1 + w_j & \forall j \in N, \forall a \in A^{*h}, \forall h \in H, l = \{h-1, h\}, \\ b_j^h & \geq & b_j^{h-1} & \forall j \in N, h = \{2, \ldots, p-1\}, \\ \sum_{j=1}^{n} w_j & = & 1, & \end{cases}$$

with:

$$\begin{cases} \delta_{a,j}^l & \in & \{0,1\} & \forall j \in N, \forall a \in A^{*h}, \forall h \in H, l = \{h-1, h\}, \\ c_{a,j}^l & \in & [0,1] & \forall j \in N, \forall a \in A^{*h}, \forall h \in H, l = \{h-1, h\}, \\ b_j^h & \in & \mathbb{R} & \forall j \in N, \forall h \in H, \\ \gamma_a & \in & \{0,1\} & \forall a \in A^* \\ w_j & \in & [0,1] & \forall j \in N, \\ \lambda & \in & [0.5, 1]. & \end{cases}$$

5.2.2 Formulation of the mixed integer program for learning a non-compensatory model

As compared to a MR-Sort model, a NCS model involves more parameters. In a standard MR-Sort model, a weight is associated to each criterion, which makes overall n parameters to elicit. With a NCS model limited to two-additive capacities, the computation of the strength of a coalition of criteria involves the weights of the criteria in the coalition and the pairwise interactions (Möbius coefficients) between these criteria. Overall there are $\frac{n(n+1)}{2} - 1$ coefficients. In the two-additive case, let us denote by m_j the weight of criterion j and by $m_{j,k}$ the Möbius interactions between criteria j and k. The capacity $\mu(J)$ of a subset of criteria J is obtained as: $\mu(J) = \sum_{j \in J} m_j + \sum_{\{j,k\} \subseteq J} m_{j,k}$. The constraints (2.13) on the interaction read:

$$m_j + \sum_{k \in J} m_{j,k} \geq 0 \qquad \forall j \in N, \forall J \subseteq N \backslash \{j\} \tag{5.2}$$

and $\sum_{j \in N} m_j + \sum_{\{j,k\} \subseteq N} m_{j,k} = 1$.

The number of monotonicity constraints (5.2) evolves exponentially as a function of the number of criteria, n. In Hüllermeier and Tehrani (2013), two other formulations are proposed in order to reduce significantly the number of constraints ensuring the monotonicity of the capacities. The first formulation reduces the number of constraints to $2n^2$ but leads to a non linear program. The second formulation reduces the number of constraints to $n^2 + 1$ without introducing non linearities but adds n^2 extra variables.

With a 2-additive MR-Sort model, the constraints for an alternative a to be assigned to a category h (Equation (2.16)) can also be expressed as follows:

$$\begin{cases} \sum_{j=1}^{n} c_{a,j}^{h-1} + \sum_{j=1}^{n} \sum_{k=1}^{j} c_{a,j,k}^{h-1} & \geq \lambda + M(\gamma_a - 1), \\ \sum_{j=1}^{n} c_{a,j}^{h} + \sum_{j=1}^{n} \sum_{k=1}^{j} c_{a,j,k}^{h} & < \lambda - M(\gamma_a - 1), \end{cases} \tag{5.3}$$

with:

- $c_{a,j}^{h-1}$ (resp. $c_{a,j}^{h}$) equals m_j if the performance of alternative a is at least as good as the performance of profile b^{h-1} (resp. b^h) on criterion j, and equals 0 otherwise;
- $c_{a,j,k}^{h-1}$ (resp. $c_{a,j,k}^{h}$) equals $m_{j,k}$ if the performance of alternative a is at least as good as the performance of profile b^{h-1} (resp. b^h) on criteria j and k, and equals 0 otherwise.

For all $a \in A^*$, $j \in N$ and $l \in H$, constraints (5.2) imply that $c_{a,j}^{l} \geq 0$ and that $c_{a,j,k}^{l} \in [-1, 1]$. The values of $c_{a,j}^{h-1}$ and $c_{a,j}^{h}$ are obtained in a similar way as it is

done for learning the parameters of a standard MR-Sort model by replacing the weights with the corresponding Möbius coefficients (5.4).

$$\begin{cases} c^l_{a,j} & \geq 0, \\ c^l_{a,j} & \leq m_j, \\ c^l_{a,j} & \leq \delta^l_{a,j}, \\ c^l_{a,j} & \geq \delta^l_{a,j} - 1 + m_j. \end{cases} \tag{5.4}$$

However it is not the case for the variables $c^{h-1}_{a,j,k}$ and $c^h_{a,j,k}$, because they involve two criteria. To linearize the formulation, we introduce new binary variables, $\Delta^l_{a,j,k}$ equal to 1 if alternative a has better performances than profile b_l on criteria j and k and equal to 0 otherwise. We obtain the value of $\Delta^l_{a,j,k}$ thanks to the conjunction of constraints given in (5.1) and the following constraints:

$$2\Delta^l_{a,j,k} \leq \delta^l_{a,j} + \delta^l_{a,k} \leq \Delta^l_{a,j,k} + 1.$$

In order to obtain the value of $c^l_{a,j,k}$, which can be either positive or negative, for all $l \in H$, we decompose the variable in two parts, $\alpha^l_{a,j,k}$ and $\beta^l_{a,j,k}$ such that $c^l_{a,j,k} = \alpha^l_{a,j,k} - \beta^l_{a,j,k}$ with $\alpha^l_{a,j,k} \geq 0$ and $\beta^l_{a,j,k} \geq 0$. The same is done for $m_{j,k}$ which is decomposed as follows: $m_{j,k} = m^+_{j,k} - m^-_{j,k}$ with $m^+_{j,k} \geq 0$ and $m^-_{j,k} \geq 0$. The values of $\alpha^l_{a,j,k}$ and $\beta^l_{a,j,k}$ are obtained thanks to the following constraints:

$$\begin{cases} \alpha^l_{a,j,k} & \leq \Delta^l_{a,j,k}, \\ \alpha^l_{a,j,k} & \leq m^+_{j,k}, \\ \alpha^l_{a,j,k} & \geq \Delta^l_{a,j,k} - 1 + m^+_{j,k}, \\ \beta^l_{a,j,k} & \leq \Delta^l_{a,j,k}, \\ \beta^l_{a,j,k} & \leq m^-_{j,k}, \\ \beta^l_{a,j,k} & \geq \Delta^l_{a,j,k} - 1 + m^-_{j,k}. \end{cases}$$

Finally, we obtain the following MIP:

$$\max_{m_j, m^+_{j,k}, m^-_{j,k}, \lambda, b^h_j} \sum_{a \in A^*} \gamma_a,$$

such that:

$$
\begin{cases}
\displaystyle\sum_{j=1}^{n}\left(c_{a,j}^{h-1}+\sum_{k=1}^{j}\alpha_{a,j,k}^{h-1}-\sum_{k=1}^{j}\beta_{a,j,k}^{h-1}\right) \geq \lambda+M(\gamma_a-1) & \\
\qquad\qquad \forall a\in A^{*h}, h=\{2,\ldots,p\}, & \\
\displaystyle\sum_{j=1}^{n}\left(c_{a,j}^{h}+\sum_{k=1}^{j}\alpha_{a,j,k}^{h}-\sum_{k=1}^{j}\beta_{a,j,k}^{h}\right) < \lambda-M(\gamma_a-1) & \\
\qquad\qquad \forall a\in A^{*h}, h=\{1,\cdots,p-1\}, & \\
m_j+\displaystyle\sum_{k\in J}(m_{j,k}^{+}-m_{j,k}^{-}) \geq 0 & \\
\qquad\qquad \forall j\in N, \forall J\subseteq N\backslash\{j\}, & \\
\displaystyle\sum_{j=1}^{n} m_j+\sum_{j=1}^{n}\sum_{k=1}^{j}(m_{j,k}^{+}-m_{j,k}^{-}) = 1, & \\
b_j^h \geq b_j^{h-1} & \forall j\in N, h=\{2,\ldots,p\},\\
c_{a,j}^{l} \leq \delta_{a,j}^{l} & \forall j\in N, \forall a\in A^{*h}, \forall h\in H, l=\{h-1,h\},\\
c_{a,j}^{l} \leq m_j & \forall j\in N, \forall a\in A^{*h}, \forall h\in H, l=\{h-1,h\},\\
c_{a,j}^{l}-m_j \geq \delta_{a,j}^{l}-1 & \forall j\in N, \forall a\in A^{*h}, \forall h\in H, l=\{h-1,h\},\\
a_j-b_j^{l} < M\cdot\delta_{a,j}^{l} & \forall j\in N, \forall a\in A^{*h}, \forall h\in H, l=\{h-1,h\},\\
a_j-b_j^{l} \geq M(\delta_{a,j}^{l}-1) & \forall j\in N, \forall a\in A^{*h}, \forall h\in H, l=\{h-1,h\},\\
\delta_{a,j}^{l}+\delta_{a,k}^{l} \geq 2\Delta_{a,j,k}^{l} & \forall\{j,k\}\in N: k<j, \forall a\in A^{*h}, \forall h\in H,\\
 & \qquad l=\{h-1,h\},\\
\delta_{a,j}^{l}+\delta_{a,k}^{l} \leq \Delta_{a,j,k}^{l}+1 & \forall\{j,k\}\in N: k<j, \forall a\in A^{*h}, \forall h\in H,\\
 & \qquad l=\{h-1,h\},\\
\alpha_{a,j,k}^{l} \leq \Delta_{a,j,k}^{l} & \forall\{j,k\}\in N: k<j, \forall a\in A^{*h}, \forall h\in H,\\
 & \qquad l=\{h-1,h\},\\
\alpha_{a,j,k}^{l} \leq m_{j,k}^{+} & \forall\{j,k\}\in N: k<j, \forall a\in A^{*h}, \forall h\in H,\\
 & \qquad l=\{h-1,h\},\\
\alpha_{a,j,k}^{l}+m_{j,k}^{+} \geq \Delta_{a,j,k}^{l}-1 & \forall\{j,k\}\in N: k<j, \forall a\in A^{*h}, \forall h\in H,\\
 & \qquad l=\{h-1,h\},\\
\beta_{a,j,k}^{l} \leq \Delta_{a,j,k}^{l} & \forall\{j,k\}\in N: k<j, \forall a\in A^{*h}, \forall h\in H,\\
 & \qquad l=\{h-1,h\},\\
\beta_{a,j,k}^{l} \leq m_{j,k}^{-} & \forall\{j,k\}\in N: k<j, \forall a\in A^{*h}, \forall h\in H,\\
 & \qquad l=\{h-1,h\},\\
\beta_{a,j,k}^{l}-m_{j,k}^{-} \geq \Delta_{a,j,k}^{l}-1 & \forall\{j,k\}\in N: k<j, \forall a\in A^{*h}, \forall h\in H,\\
 & \qquad l=\{h-1,h\},
\end{cases}
$$

with:

$$\begin{cases} c^l_{a,j} \in [0,1] & \forall j \in N, \forall a \in A^{*h}, \forall h \in H, l = \{h-1, h\}, \\ \delta^l_{a,j} \in \{0,1\} & \forall j \in N, \forall a \in A^{*h}, \forall h \in H, l = \{h-1, h\}, \\ \alpha^l_{a,j,k}, \beta^l_{a,j,k} \in [0,1] & \forall \{j,k\} \in N : k < j, \forall a \in A^{*h}, \forall h \in H, \\ & l = \{h-1, h\}, \\ \Delta^l_{a,j,k} \in \{0,1\} & \forall \{j,k\} \in N : k < j, \forall a \in A^{*h}, \forall h \in H, \\ & l = \{h-1, h\}, \\ m_j \in [0,1] & \forall j \in N, \\ m^+_{j,k}, m^-_{j,k} \in [0,1] & \forall j \in N, \forall k \in N, k < j, \\ b^h_j \in \mathbb{R} & \forall j \in N, \forall h \in H, \\ \gamma_a \in \{0,1\} & \forall a \in A^*, \\ \lambda \in [0,1]. \end{cases}$$

5.3 Metaheuristic for learning the parameters of a non-compensatory sorting model

The MIP described in the previous section requires a lot of binary variables and is therefore not well-suited for large problems. In the present section, we describe an adaptation of the metaheuristic described in Chapter 3 in order to learn the parameters of a NCS model. Like for the MIP in the previous section, we limit the model to 2-additive capacities in order to reduce the number of coefficients as compared to a model with a general capacity.

One of the components that needs to be adapted in the metaheuristic in order to be able to learn a 2-additive NCS model is the linear program that infers the weights and the majority threshold (3.1). Like in the MIP described in the previous section, we use the Möbius transform to express capacities. In order to infer Möbius coefficients, m_j and $m_{j,k}$, $\forall j, \forall k$ with $k < j$, we modify the linear program as shown in the following equation:

$$\min_{m_j, m_{j,k}, \lambda} \sum_{a \in A^*} (x'_a + y'_a)$$

such that:

$$\begin{cases} \displaystyle\sum_{j:a_j \geq b_j^{h-1}}^{n} \left(m_j + \sum_{k:a_k \geq b_k^{h-1}}^{j} m_{j,k} \right) - x_a + x'_a = \lambda & \forall a \in A^{*h}, \\ & h = \{2, \ldots, p\}, \\ \displaystyle\sum_{j:a_j \geq b_j^{h}}^{n} \left(m_j + \sum_{k:a_k \geq b_k^{h}}^{j} m_{j,k} \right) + y_a - y'_a + \varepsilon = \lambda & \forall a \in A^{*h}, \\ & h = \{1, \ldots, p-1\}, \\ \displaystyle\sum_{j=1}^{n} m_j + \sum_{j=1}^{n} \sum_{k=1}^{j} m_{j,k} = 1, & \\ \displaystyle m_j + \sum_{k \in J} m_{j,k} \geq 0 & \forall j \in N, \forall J \subseteq N \backslash \{j\}, \end{cases}$$

with:

$$\begin{cases} \lambda \in [0.5; 1] & \\ m_j \in [0, 1] & \forall j \in N \\ m_{j,k} \in [-1, 1] & \forall j \in N, \forall k \in N, k < j \\ x_a, y_a, x'_a, y'_a \in \mathbb{R}_0^+ & a \in A^* \\ \varepsilon \text{ a small positive number.} & \end{cases}$$

The value of $x_a - x'_a$ (resp. $y_a - y'_a$) represents the difference between the capacity of the criteria belonging to the coalition in favor of $a \in A^{*h}$ with regard to b^{h-1} (resp. b^h) and the majority threshold. If both $x_a - x'_a$ and $y_a - y'_a$ are positive, then object a is assigned to the correct category. In order to try to maximize the number of examples correctly assigned by the model, the objective function of the linear program minimizes the sum of x'_a and y'_a, i.e. the objective function is $\min \sum_{a \in A^*} (x'_a + y'_a)$.

The metaheuristic adjusting the profile also needs some adaptations in order to take capacities into account. More precisely, the formal definition of the sets in which objects are classified for computing the candidate move evaluation should be adapted. The semantics of the sets, given in Section 3.3.4 remains identical. The formal definitions of these sets have to be adapted as follows to take into

account the capacity:

$$
\begin{aligned}
V_{h,j}^{+\delta} &= \left\{a \in A_{h+1}^{*h} : b_j^h + \delta > a_j \geq b_j^h \text{ and } \mu(N_{a \geq b^h} \backslash \{j\}) < \lambda\right\}, \\
V_{h,j}^{-\delta} &= \left\{a \in A_h^{*h+1} : b_j^h - \delta < a_j < b_j^h \text{ and } \mu(N_{a \geq b^h} \cup \{j\}) \geq \lambda\right\}, \\
W_{h,j}^{+\delta} &= \left\{a \in A_{h+1}^{*h} : b_j^h + \delta > a_j \geq b_j^h \text{ and } \mu(N_{a \geq b^h} \backslash \{j\}) \geq \lambda\right\}, \\
W_{h,j}^{-\delta} &= \left\{a \in A_h^{*h+1} : b_j^h - \delta < a_j < b_j^h \text{ and } \mu(N_{a \geq b^h} \cup \{j\}) < \lambda\right\}, \\
Q_{h,j}^{+\delta} &= \left\{a \in A_{h+1}^{*h+1} : b_j^h + \delta > a_j \geq b_j^h \text{ and } \mu(N_{a \geq b^h} \backslash \{j\}) < \lambda\right\}, \\
Q_{h,j}^{-\delta} &= \left\{a \in A_h^{*h} : b_j^h - \delta < a_j < b_j^h \text{ and } \mu(N_{a \geq b^h} \cup \{j\}) \geq \lambda\right\}.
\end{aligned}
$$

The formal definitions of the sets $R_{h,j}^{+\delta}$, $R_{h,j}^{-\delta}$, $T_{h,j}^{+\delta}$ remain the same as for the simple additive MR-Sort model as well as the function computing the evaluations taking into account the size of the sets.

The rest of the metaheuristic remains unchanged.

5.4 Experiments

The use of the MIP for learning a NCS model is limited because of the large number of binary variables involved. It contains more binary variables than the MIP learning the parameters of a simple additive MR-Sort model. Experiments reported in Leroy et al. (2011) have demonstrated that the computing time required to learn the parameters of a standard MR-Sort model having a small number of criteria and categories from a small set of assignment examples becomes quickly prohibitive. Therefore we cannot expect to be able to treat large problems using the MIP for learning NCS models.

To assess the performance of the metaheuristic designed for learning the parameters of a NCS model, we use it to learn NCS models from the real data sets used in Chapter 3.

In our first experiment, we use 50% of the alternatives in the data sets as learning set and the rest as test set. We learn MR-Sort and NCS models using both metaheuristics. We repeat this procedure for 100 random splits of the data sets in learning and test sets. We observe from Table 5.1 that the classification accuracy obtained with the NCS metaheuristic is on average comparable to the one obtained with the MR-Sort metaheuristic. The use of a more expressive model does not help much to improve the classification accuracy of the test set.

In a second experiment, we check the ability of MR-Sort and NCS to restore the whole data set. To do so, we run both metaheuristics 100 times. The average classification accuracy and standard deviation of the learning set are given in Table 5.2. The NCS metaheuristic does not always give better results than the MR-Sort one in restoring the learning set examples. Except for the MPG data

Table 5.1: Average and standard deviation of the classification accuracy of the test set when using 50 % of the examples as learning set and the rest as test set.

Data set	Metaheuristic MR-Sort	Metaheuristic NCS
DBS	0.8377 ± 0.0469	0.8312 ± 0.0502
CPU	0.9325 ± 0.0237	0.9313 ± 0.0272
BCC	0.7250 ± 0.0379	0.7328 ± 0.0345
MPG	0.8219 ± 0.0237	0.8180 ± 0.0247
ESL	0.8996 ± 0.0185	0.8970 ± 0.0173
MMG	0.8268 ± 0.0151	0.8335 ± 0.0138
ERA	0.7944 ± 0.0173	0.7944 ± 0.0156
LEV	0.8408 ± 0.0122	0.8508 ± 0.0188
CEV	0.9064 ± 0.0119	0.9118 ± 0.0263

Table 5.2: Average and standard deviation of the classification accuracy of the learning set when using the MR-Sort and NCS models learned on the whole data set.

Data set	#attributes	Metaheuristic MR-Sort	Metaheuristic NCS
DBS	8	0.9268 ± 0.0096	0.9326 ± 0.0087
CPU	6	0.9643 ± 0.0048	0.9703 ± 0.0091
BCC	7	0.7605 ± 0.0147	0.7761 ± 0.0085
MPG	7	0.8419 ± 0.0099	0.8389 ± 0.0069
ESL	4	0.9164 ± 0.0033	0.9168 ± 0.0042
MMG	5	0.8419 ± 0.0099	0.8409 ± 0.0091
ERA	4	0.8035 ± 0.0052	0.8027 ± 0.0053
LEV	4	0.8501 ± 0.0082	0.8643 ± 0.0038
CEV	6	0.9005 ± 0.0141	0.9172 ± 0.0101

set, we observe a slight advantage (of the order of one standard deviation) in favor of NCS when the number of attributes is at least 6. There is almost no difference for the data sets described by 4 or 5 attributes and for MPG (7 attributes).

Average computing times of the results in Table 5.2 are displayed in Table 5.3. Learning a NCS model can take up to almost 3 times as much as learning a simple MR-Sort model.

The above experiments on benchmark data sets available in the literature failed to show a clear advantage at using NCS rather than MR-Sort. This raises the following question. Which type of data set would reveal a gain of expressivity provided by NCS over MR-Sort? We investigate this question in the next section.

Table 5.3: Average computing time (in seconds) required to find a solution with MR-Sort and NCS metaheuristics when using all the examples as learning set.

Data set	Metaheuristic MR-Sort	Metaheuristic NCS
DBS	3.0508	6.9547
CPU	3.1646	5.2069
BCC	3.3700	7.7545
MPG	4.4136	9.9294
ESL	3.8466	7.2495
MMG	6.1481	13.4848
ERA	5.9689	14.4875
LEV	5.8986	13.2356
CEV	11.1122	31.7042

5.5 Gain in descriptive power with the non-compensatory sorting model

As the results obtained with NCS models are inconclusive, we try to find a reason to this by quantifying the gain in descriptive power when passing from additive weights to capacities.

5.5.1 Notion of minimal sufficient coalition

We remind the definition of a sufficient coalition of criteria[1].

Definition 5. *In a NCS model, a subset of criteria J is said to be a sufficient coalition (SC) if its capacity is greater than the majority threshold λ, i.e. when the condition $\mu(J) \geq \lambda$ is fulfilled (see Equation* (2.15)*).*

Proposition 1. *Any family of sufficient coalitions can be represented as the set of subsets $J \subseteq \{1, \ldots, n\}$ verifying*

$$\mu(J) \geq \lambda,$$

for some capacity μ and threshold $\lambda \geq 0$. Conversely, if μ is a capacity and λ is a nonnegative number, the set of subsets J satisfying the inequality $\mu(J) \geq \lambda$ is a family of sufficient coalitions.

Proof. A family of sufficient coalitions is a family of subsets such that any subset containing a subset of the family is itself in the family. Define a nonnegative set

[1] Also called "winning coalition of criteria"

Table 5.4: Known values of the Dedekind numbers $D(n)$.

n	$D(n)$
0	2
1	3
2	6
3	20
4	168
5	7 581
6	7 828 354
7	2 414 682 040 998
8	56 130 437 228 687 557 907 788

function μ letting $\mu(J) = 1$ if J is a sufficient coalition and 0 otherwise. One can see that μ is monotone, and therefore a capacity, due to the characteristic properties of the families of sufficient coalitions. It is also normalized. Define the threshold $\lambda = 0.5$. Clearly $\mu(J) \geq 0.5$ iff J is a sufficient coalition. The proof of the converse is also straightforward. □

The set of SC may be large (typically exponential in n), but one can avoid listing all its elements.

Definition 6. *A minimal sufficient coalition (MSC) is a SC which is not properly included in another SC. Knowing the set of MSC allows to determine all SC since a coalition is sufficient as soon as it contains a MSC.*

A family of MSC is any collection of subsets of $\{1, \ldots, n\}$ such that none of them is included in another. In other words, a set of MSC is an antichain in the set of subsets of $\{1, \ldots, n\}$ (partially) ordered by inclusion. It is well-known that the number of antichains in the power set of $\{1, \ldots, n\}$ is $D(n)$, the nth Dedekind number (The OEIS Foundation Inc. (2009), sequence A000372). These numbers grow extremely rapidly with n and no exact closed form is known for them. These numbers have been computed up to $n = 8$; their values appear in Table 5.4.

The Dedekind numbers are also the number of monotone (more precisely, positive (Crama and Hammer, 2011)) Boolean functions (BFs) in n variables. Clearly, the set of sufficient coalitions can be represented as the set of n-dimensional Boolean vectors which give the value 1 to a monotone Boolean function (MBF), and conversely. Another application of the Dedekind numbers is in game theory. They are the numbers of simple games with n players in minimal winning form (von Neumann and Morgenstern, 1972; Kurz and Tautenhahn, 2013).

Table 5.5: Number of inequivalent families of sufficient coalitions of n criteria.

n	$R(n)$
0	2
1	3
2	5
3	10
4	30
5	210
6	16 353
7	490 013 148

One way of simplifying the study of the families of sufficient coalitions consists in keeping only one representative of each class of equivalent families of SC. Two families will be considered as equivalent, or isomorphic, if they can be transformed one into the other just by permuting the labels of the criteria. For instance, consider the following family of minimal SC for $n = 4$: $\{2, 4\}, \{2, 3\}, \{1, 3, 4\}$. It consists of 2 subsets of 2 criteria and one of 3 criteria. There are 12 equivalent families that can be obtained from this one, by permuting the criteria labels (the criterion which does not show up in the set of 3 criteria can be chosen in 4 different ways and the two criteria which distinguish the two pairs can be chosen in 3 different ways). The number $R(n)$ of *inequivalent* families of SC is known for $n = 0$ to $n = 7$ (The OEIS Foundation Inc. (2009), sequence A003182). $R(7)$ was only recently computed by Stephen and Yusun (2014). Table 5.5 lists the known values of $R(n)$.

Finally we recall Sperner's theorem (Caspard et al., 2012, p. 116-118), a result that will be useful in the process of generating all possible families of SC. The maximal size of an antichain in the power set of a set of n elements is $\binom{n}{\lfloor n/2 \rfloor}$. Hence the latter is the maximal number of sets in a family of minimal SC.

5.5.2 Listing the families of minimal sufficient coalitions

For generating all families of MSC and selecting a representative of each class of equivalent families, we follow a strategy similar to the one used by Stephen and Yusun (2014). We describe it briefly. The families of MSC can be partitioned according to their type (called "profile" by Stephen and Yusun (2014)). The type of a family of MSC is an integer vector $(k_1, k_2, \ldots, k_n)$, where k_i represents the number of coalitions of i criteria in the family. For instance, the family $\{2, 4\}, \{2, 3\}, \{1, 3, 4\}$, for $n = 4$, is of the type $(0, 2, 1, 0)$, since it involves two coalitions of 2 criteria and one of 3 criteria. For any feasible type, $\sum_{i=1}^{n} k_i \leq$

Table 5.6: Number of inequivalent families of minimal sufficient coalitions.

Type	Representative	# equivalent
$(1,0,0)$	$\{\{1\}\}$	3
$(2,0,0)$	$\{\{1\},\{2\}\}$	3
$(3,0,0)$	$\{\{1\},\{2\},\{3\}\}$	1
$(0,1,0)$	$\{\{1,2\}\}$	3
$(1,1,0)$	$\{\{1\},\{2,3\}\}$	3
$(0,2,0)$	$\{\{1,3\},\{2,3\}\}$	3
$(0,3,0)$	$\{\{1,2\},\{1,3\},\{2,3\}\}$	1
$(0,0,1)$	$\{\{1,2,3\}\}$	1
Total	8	18

$\binom{n}{\lfloor n/2 \rfloor}$, due to Sperner's theorem.

The listing algorithm roughly proceeds as follows:

1. generate all type vectors $(k_1, k_2, \ldots, k_n)$ in lexicographic increasing order;
2. for each type, generate all families of subsets of $\{1, \ldots, n\}$ having the right type and eliminate those that are not antichains, i.e. those involving a subset that is included in another subset;
3. for each type and for each family of this type, the list of remaining families is screened for detecting families that are equivalent, counting them and eliminating them from the list of families of the type considered.

This algorithm outputs a list containing a representative of each class of equivalent families of MSC for each type.

Example 9. *For $n = 3$, the inequivalent families of MSC, with their number of equivalent versions, are displayed in Table 5.6.*

Some remarks:

1. there exist two additional families which do not appear in Table 5.6:
   - the empty family, corresponding to the case in which no coalition is sufficient, which means, for a sorting procedure, that all objects are assigned to the "bad" category;
   - the family of which the sole element is the empty set; this means that all coalitions are sufficient, even the empty one, and consequently, all objects are sorted in the "good" category.

Adding these two extreme cases to the counts in the last line of Table 5.6 yields values that are consistent with Tables 5.5 and 5.4.

2. for $n = 3$, every possible class type has a single representative. For larger values of n, this is no longer the case. For instance, for $n = 4$, we have 3 inequivalent representatives for type $(0, 3, 0, 0)$:

Type	Representative	# equivalent
$(0,3,0,0)$	$\{\{1,3\},\{1,2\},\{3,4\}\}$	12
$(0,3,0,0)$	$\{\{2,4\},\{1,2\},\{1,4\}\}$	4
$(0,3,0,0)$	$\{\{2,4\},\{3,4\},\{1,4\}\}$	4

These three inequivalent families are the three sorts of non-isomorphic 3-edge graphs on 4 vertices.

3. in the sequel, in the absence of ambiguity, we shall drop the brackets around the coalitions and the commas separating the elements of a coalition in order to simplify the description of a family of SC; for instance, the first family of type (0,3,0,0) above will be denoted by: $\{13, 12, 34\}$ instead of $\{\{1,3\},\{1,2\},\{3,4\}\}$.

The algorithm sketched above can be made more efficient by implementing the following properties (see Stephen and Yusun, 2014, lemma 2.4 for a proof), linking the families of MSC:

1. There is a one-to-one correspondence between families consisting exclusively of k_i MSC of cardinality i and families consisting exclusively of $\binom{n}{i} - k_i$ MSC of cardinality i. In other terms, there is a bijection between the families of the type $(0, \ldots, 0, k_i, 0, \ldots, 0)$ and these of the type $(0, \ldots, 0, \binom{n}{i} - k_i, 0, \ldots, 0)$. For instance, in Table 5.6, generating family $\{12\}$ of type (0,1,0), automatically yields family $\{13, 23\}$ of type (0,2,0). The number of representatives in both types are identical (three, in the latter example).

2. If a family of MSC on n criteria contains at least one singleton, then the remaining MSC of the family do not involve this criterion and hence belong to a type of family of MSC on $n - 1$ criteria. In the example of $n = 3$, knowing the families of MSC on 2 criteria allows to generate the families on three criteria for which one criterion alone is a sufficient coalition. For instance, if criterion 1 alone is sufficient, one can build all families in which 1 is a MSC by putting together with 1 each family of MSC on criteria 2 and 3, i.e.: {}, {2},{3},$\{2, 3\}$ and {23}. This, however, will not allow to

directly compute the number of representatives of each type, since some families, involving more than one singleton as MSC, can be generated in several ways. For instance, $\{1, 2\}$ will be obtained both when starting from the singleton 1 as a MSC and completing this family by MSC included in $\{2, 3\}$, and, starting from the singleton 2 and completing this family by MSC extracted from $\{1, 3\}$.

3. There is a one-to-one correspondence between families of MSC belonging to type $(k_1, k_2, \ldots, k_{n-1}, 0)$ and these belonging to the "reverse" type $(k_{n-1}, \ldots, k_2, k_1, 0)$. For instance, starting from the family $\{1, 2\}$ belonging to type (2,0,0) and taking the complement of each MSC, one obtains the family $\{23, 13\}$, which belongs to (0,2,0). This correspondence allows to halve the computations for $D(n)$ and $R(n)$.

Using this algorithm on a cluster of computers, we have computed the list of all inequivalent families of MSC for $n = 2$ to $n = 6$. The results, grouped by type, are available at `http://olivier.sobrie.be`. For illustrative purposes, the case $n = 4$ is given in Appendix C.1.

5.5.3 Representation of coalitions with k-additive capacities

Our main goal in this section is to partition the set of families of MSC, for fixed n, in categories $\mathcal{C}_k$, which are defined as follows.

Verifying the representability of a rule by a k-additive capacity

Definition 7. *A family of sufficient coalitions belongs to class $\mathcal{C}_k$ if*

1. *it is the set of all subsets J of $\{1, \ldots, n\}$ satisfying an inequality of the type: $\mu(A) \geq \lambda$, where μ is a normalized k-additive capacity and λ a non-negative real number;*

2. *k is the smallest integer for which such an inequality is valid.*

It is clear that all equivalent families of MSC belong to the same class $\mathcal{C}_k$. Hence it is sufficient to check for one representative of each class of equivalent families of MSC whether or not it belongs to $\mathcal{C}_k$.

The checking strategy is the following. For each inequivalent family of MSC (listed as explained in Section 5.5.2), we iteratively check whether it belongs to class $\mathcal{C}_k$, starting from $k = 1$ and incrementing k until the test is positive. We know, by Proposition 1, that this will occur at the latest for $k = n$. The test can be formulated as a linear program. Basically, we have to write constraints imposing that $\mu(A) \geq \lambda$ for each sufficient coalition J and that the same inequality is not satisfied for all other coalitions, which will be called insufficient coalitions. It

is enough to write these sorts of constraints only for the minimal sufficient coalitions and for the maximal insufficient coalitions. The set of minimal sufficient (resp. maximal insufficient) coalitions will be denoted by *SCMin* (resp. *ICMax*).

To formulate the problem as a linear program, we use formula (2.12), which expresses the value of the capacity μ as a linear combination of its associated Möbius function m. This enables to control the parameter k which fixes the k-additivity of the capacity. When checking whether a family of MSC belongs to class $\mathcal{C}_k$, we set the values of the variables $m(K)$ to 0 for all sets K of cardinality superior to k; the remaining values of the interaction function are the main variables in the linear program. The following constitutes the general scheme of the linear programs used for each class $\mathcal{C}_k$:

$$\max \lambda, \tag{5.5}$$

such that:

$$\begin{cases} \mu(J) \geq \lambda & \forall J \in \mathit{SCMin}, \\ \mu(J) \leq \lambda - \varepsilon & \forall J \in \mathit{ICMax}, \\ \mu(J) = \sum_{K \subseteq J} m(K) & \forall J \in \mathit{SCMin} \cup \mathit{ICMax} \\ \sum_{K: i \in K \subseteq J} m(K) \geq 0, & \forall i \in N, \forall J \subseteq N, \\ \sum_{K \subseteq N} m(K) = 1, & \\ m(K) \geq 0 & \forall K \subseteq N : |K| = 1, \end{cases} \tag{5.6}$$

with:

$$\begin{cases} m(K) \in \mathbb{R} & \forall K \subseteq N, \\ \lambda \geq 0, & \end{cases} \tag{5.7}$$

and ε a small positive value.

Results

For $n = 1$ to 6 and for each family in the list of inequivalent families of MSC, we checked whether this family belongs to $\mathcal{C}_k$, starting with $k = 1$ and incrementing its value until the check is positive. The results are presented in Table 5.7 for the number and proportion of inequivalent families in classes $\mathcal{C}_2$ and $\mathcal{C}_3$. The families that are not in these classes belong to class $\mathcal{C}_1$. Up to $n = 6$, inclusively, there are no families in classes $\mathcal{C}_4$ or above, which means that all families can be represented using a 3-additive capacity (in the worst case). Up to $n = 5$, inclusively, 2-additive capacities are sufficient. Table 5.8 represents a similar

Table 5.7: Number and proportion of inequivalent families of MSC that are representable by a 2- or 3-additive capacity.

n	$R(n)$	$\mathcal{C}_2$	$\mathcal{C}_3$
0	2	0 (00.00 %)	0 (00.00 %)
1	3	0 (00.00 %)	0 (00.00 %)
2	5	0 (00.00 %)	0 (00.00 %)
3	10	0 (00.00 %)	0 (00.00 %)
4	30	3 (10.00 %)	0 (00.00 %)
5	210	91 (43.33 %)	0 (00.00 %)
6	16 353	15 240 (93.19 %)	338 (02.07 %)

Table 5.8: Number and proportion of all families of MSC that are representable by a 2- or 3-additive capacity.

n	$D(n)$	$\mathcal{C}_2$	$\mathcal{C}_3$
0	2	0 (00.00 %)	0 (00.00 %)
1	3	0 (00.00 %)	0 (00.00 %)
2	6	0 (00.00 %)	0 (00.00 %)
3	20	0 (00.00 %)	0 (00.00 %)
4	168	18 (10.71 %)	0 (00.00 %)
5	7 581	4 294 (56.64 %)	0 (00.00 %)
6	7 828 354	7 584 196 (96.88 %)	145 502 (01.86 %)

information but each family in the list of inequivalent families is weighted by the size of the equivalence class it represents. In other words, this is the result that would have been obtained by checking all families of MSC for belonging to class $\mathcal{C}_1$, $\mathcal{C}_2$ or $\mathcal{C}_3$.

The information displayed in Table 5.7 (resp. 5.8) is represented in graphical form in Figure 5.1 (resp. 5.2). The cases of 0, 1 and 2 criteria are not represented since all families can be represented by a 1-additive capacity. These figures clearly show that the proportion of families that can be represented by means of a 1-additive capacity, i.e., by additive weights, decreases quite rapidly with the number of criteria. In contrast, the proportion of families that can be represented by a 2-additive capacity grows up to more than 93% from $n = 3$ to $n = 6$. The proportions slightly differ depending on whether only inequivalent families or all families are taken into account. One can observe that the proportion of families in class $\mathcal{C}_2$ is a bit larger when considering all families (Table 5.8 and Figure 5.2).

As a matter of illustration, we describe a few examples for $n = 4$ and $n = 6$.

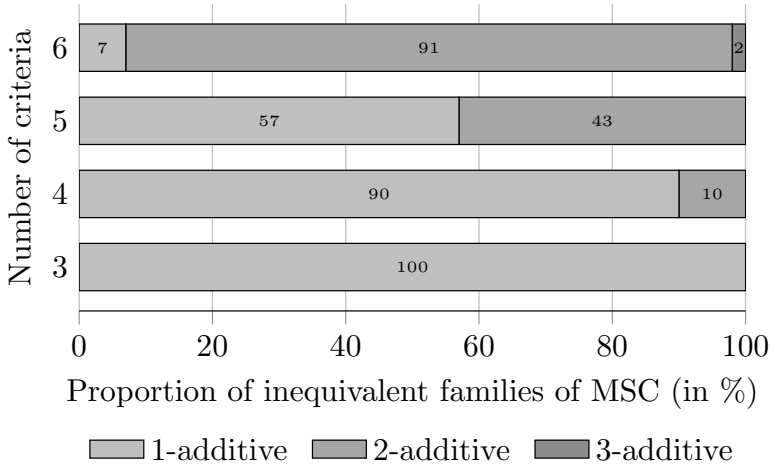


Figure 5.1: Proportion of inequivalent families of MSC in classes $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$.

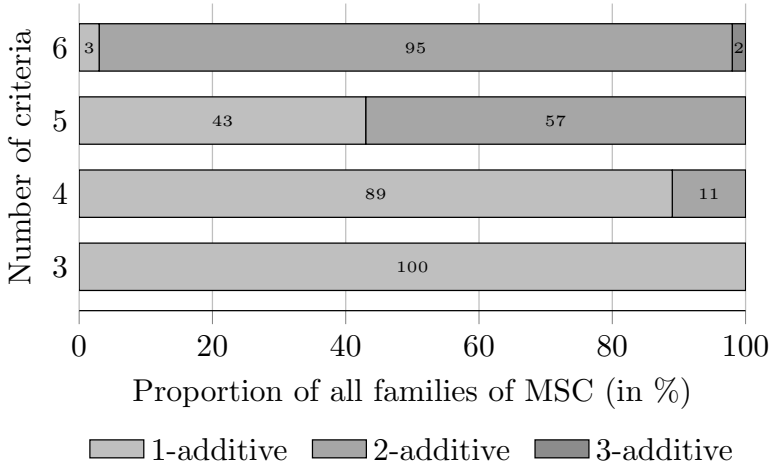


Figure 5.2: Proportion of all families of MSC in classes $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$.

The list of all inequivalent MSC for $n = 5$, which are not representable by a 1-additive capacity, is in Appendix C.2. The categorization in classes $\mathcal{C}_k$ is available at `http://olivier.sobrie.be` for $n = 4, 5, 6$.

Example 10. *Here are the three families of MSC on 4 criteria that cannot be represented using a 1-additive capacity (they can be by a 2-additive capacity).*

Type	*Representative*	*# equivalent*
$(0, 2, 0, 0)$	$\{23, 14\}$	3
$(0, 3, 0, 0)$	$\{13, 12, 34\}$	12
$(0, 4, 0, 0)$	$\{13, 14, 23, 24\}$	3

These three inequivalent families yield, by permutations of the criteria labels, a total of 18 families that can only be represented using a 2-additive capacity.

The first inequivalent family is precisely the example that we used in Section 5.1 to show that not all families of SC can be represented by a 1-additive capacity. In contrast, it can be represented, for instance, by setting $m_1 = m_2 = m_3 = m_4 = 0$ *and* $m_{12} = m_{34} = 5/10$*, while the other pairwise interactions* m_{13}*,* m_{14}*,* m_{23} *and* m_{24} *are set to 0. Setting the threshold* λ *to* $5/10$ *(or any value greater than 0) allows to separate the sufficient coalitions from the insufficient. This representation is by no means unique. We construct another capacity by setting* $m_1 = m_2 = m_3 = m_4 = 1/3$*,* $m_{13} = m_{14} = m_{23} = m_{24} = -1/12$ *and* $m_{12} = m_{34} = 0$*. We have:* $\mu(13) = \mu(14) = \mu(23) = \mu(24) = 7/12$ *while* $\mu(12) = \mu(34) = 2/3$*. Setting the threshold* λ *to a value greater than* $7/12$ *also separates the sufficient from the insufficient coalitions.*

Note that the first and the last example are complementary in the sense of the first property allowing to speed up the enumeration of the families of MSC described at the end of Section 5.5.2. Both these families are composed of pairs of criteria; the two pairs in the first family are disjoint from the four in the third family and all pairs are either in one or the other family. In such a situation, it is clear that both families belong to the same class $\mathcal{C}_k$*.*

Example 11. *Here are two examples of inequivalent families of MSC on 6 criteria that are not representable by a 2-additive capacity but require a 3-additive capacity. There are 338 such inequivalent families which yield, through permutations, a total of 145 502 families*[2]*. A simple example is of the type* $(0, 0, 4, 0, 0, 0)$*. The MSC are* $\{136, 234, 125, 456\}$*. There are 30 equivalent families that can be derived from this family by permutation. Another, much more complex example*

[2]If all permutations of the criteria labels were yielding different families, the total number of families would be $338 \times 720 = 243\,360$

is of the type $(0, 1, 7, 1, 0, 0)$. *The MSC are*

$$\{135, 256, 345, 36, 234, 456, 1245, 146, 123\}.$$

There are 360 families that are equivalent to this one through permutations.

In the 338 families, no MSC consists of a single criterion; none of them involves 5 criteria. The largest number of MSC in a family is 16, the maximal cardinality of a family of MSC on 6 criteria being the Sperner number 20.

5.5.4 Comparison between the majority rule sorting model and the non-compensatory sorting model

Among NCS assignment rules, some can be exactly represented by additive weights and a threshold (the MR-Sort rules), while the others require a non-additive capacity and a threshold. We call the latter non-additive NCS rules. These are not MR-Sort rules but they can be approximated by a MR-Sort model. The experiment described below aims at assessing how well a non-additive NCS rule can be approximated by a MR-Sort rule.

Consider a NCS model assigning alternatives to two categories, C_1 and C_2. For a given profile, the set of all possible alternatives can be partitioned in 2^n subsets, where n is the number of criteria. Each of these subsets is characterized by one of the 2^n relative positions of an alternative with regard to the profile. On each criterion, the performance of an alternative is either at least as good as the profile or worse. Due to the ordinal nature of the NCS rule, all alternatives that share the same relative position with regard to the profile (i.e. all alternatives in the same class of the partition in 2^n subsets) are assigned to the same category. If we assume that the evaluations of the alternatives on all criteria range in the $[0, 1]$ interval, we can set the profile values to 0.5 on all criteria. The set of n-dimensional Boolean vectors is composed of exactly one example of each possible relative position with regard to the profile.

Our experiments are conducted as follows.

1. We modify the MIP described in Section 5.2.1 to learn only the weights and the majority threshold of a MR-Sort model on the basis of fixed profiles and assignment examples. The objective function of the MIP remains the minimization of the $0/1$ loss.

2. We generate all possible NCS rules for $n = 4, 5, 6$ criteria. For more detail about how this can be done, see Ersek Uyanık et al. (2014); the list of all non-equivalent NCS rules is available at `http://olivier.sobrie.be`. Each non-additive NCS rule, is used to assign the set of n-dimensional Boolean vectors to one of the two categories (using the 0.5 constant profile). These sets of representative alternatives constitute our learning sets.

Table 5.9: Average, minimum and maximum 0/1 loss of the learning sets after learning additive weights and the majority threshold of a MR-Sort model.

n	% non-additive	MR-Sort		
		min.	max.	avg.
4	11 %	6.2 %	6.2 %	6.2 %
5	57 %	3.1 %	9.4 %	3.9 %
6	97 %	1.6 %	12.5 %	4.8 %

3. The modified MIP is used to learn the weights and majority threshold of a MR-Sort model, which restores as well as possible the assignments made by the non-additive NCS rule.

The results of the experimentation are displayed in Table 5.9. Each row of the table contains the results for a given number of criteria, $n = 4, 5, 6$. The second column shows the percentage of non-additive NCS rules among all possible rules for each given number of criteria. The last three columns contain the min, max and average percentage of the 2^n examples assigned by non-additive rules that cannot be restored by a simple additive model.

We observe that a MR-Sort model on 4 criteria is, in the worst case, not able to restore 6.2% of the examples in the learning set (1 example out of 16). With 5 and 6 criteria, the maximum 0/1 loss increases respectively to 9.4% (3 examples out of 32) and 12.5% (8 examples out of 64).

Note that these proportions were obtained using learning sets in which each type of relative position with regard to the profile is represented exactly once. Therefore these conclusions should be valid for learning sets in which all types of relative positions are approximately equally represented. On a test set, the difference in classification performance between a non-additive NCS rule and its approximation by a MR-Sort rule can be amplified, or, on the contrary, can fade, depending on the proportion of the test alternatives belonging to the various types of relative positions with regard to the profile.

Table 5.9 reveals another important information. The proportion of non-additive NCS rules among all NCS rules quickly grows with the number of attributes: from 11% of 2-additive NCS rules for $n = 4$ to 97% for $n = 6$. It hence becomes more and more likely that a NCS rule is not a MR-Sort one when n grows.

The results in Table 5.9 could help to better understand the relatively poor gains observed in the previous section when comparing the metaheuristic algorithm for learning a 2-additive NCS model and a MR-Sort model. We noticed that the classification accuracy of the learned NCS rule tended to be slightly

better for the data sets involving at least 6 attributes. The lack of an advantage for data sets involving 4 attributes might be due to the relative scarcity of non-additive NCS rules for $n = 4$ (11%). When a gain is obtained, it is tiny, which might result from the fact that the approximation of a non-additive NCS rule by a MR-Sort rule is relatively good, at least up to $n = 6$. Investigating the NCS for $n \geq 7$ model in a systematic way, using the same method as we did in our last experiments, is almost impossible due to the extremely fast growth of the number of possible NCS rules (see Ersek Uyanık et al. (2014)). It is however arguable that non-additive NCS rules could be at an advantage, as compared to MR-Sort rules, when the number of attributes is at least as large as 6.

5.6 Chapter conclusion

In this chapter we have introduced two algorithms that enable to learn the parameters of a NCS model. The first algorithm, a MIP, enables to obtain an optimal solution which maximizes the compatibility of the learning set with the model. The MIP is unfortunately not suitable for large data sets since it uses many binary variables. That's why we proposed a modified version of the metaheuristic presented in Chapter 3 in order to learn a NCS model. With this metaheuristic it is possible to handle large data sets. However it is not guaranteed that the solution obtained with the metaheuristic is optimal.

We tested the metaheuristic for learning a NCS model on real data sets issued from the preference learning field. The results have been compared with the ones obtained with the metaheuristic of Chapter 3. It showed that using a NCS model on these data sets does not help a lot. Indeed, the average classification accuracy and area under the curve are close to the values obtained with the MR-Sort metaheuristic.

In view of the results obtained with the NCS metaheuristic, we tried to find the reasons why performances are not that much improved when passing from a model using additive weights to a model using capacities. We have shown that a capacity is a particular representation of monotone Boolean function. In other words, there is a monotone Boolean function associated to each capacity. Capacities can be represented by a list of MSC. Listing the monotone Boolean functions amounts to list all possible monotone Boolean functions. In this thesis, we listed all the MSC for a number of criteria up to 6. We observed that the proportion of MSC that are representable with a 1-additive capacity decreases when the number of criteria increases. For 6 criteria, we noticed that almost all the families of MSC are representable with 2-additive capacities and all are representable with a 3-additive capacity.

The analysis showed that it is interesting to use 2-additive and 3-additive capacities to increase the descriptive power of the model. However it does not

explain why the gain is small when passing from additive weights to capacities.

Chapter 6

New veto rule for outranking models

Outranking methods are based on the principles of concordance and non-discordance: an alternative a is considered "at least as good as" another alternative if a majority of criteria supports this statement (concordance) and if no criterion strongly disagrees with it (non-discordance). In the literature, multiple strategies have been adopted in order to model the concordance and non-discordance principles. Up to now in this thesis, we never considered non-discordance relation, i.e. veto, neither in the majority rule sorting (MR-Sort) model nor in the non-compensatory sorting (NCS) model. In the NCS model defined by Bouyssou and Marchant (2007a,b), the veto applies for the assertion "a is preferred to profile b^h" when the score of the alternative a "much worse than" the profile b^h on any criterion j, i.e. if a is below the profile b^h minus a veto threshold value v_j on any criterion j. A veto threshold v_j is possibly associated to each criterion j. In this chapter, we consider a new veto rule applying when the performances of an alternative a is "much worse than" the profile b^h on a subset of criteria. We begin this chapter with an introductory example in order to describe the concept of "coalitional veto". Then we give an overview of the types of veto found in the literature. After that we describe formally the new veto rule we introduce. Afterwards, we present a mixed integer formulation designed for learning the parameters of a MR-Sort model using coalitional veto. Finally, we propose a strategy in order to extend the metaheuristic described in Chapter 3 for learning the parameters of a MR-Sort model using coalitional veto.

6.1 Introductory example

.

As an introductory example, consider a decision maker (DM) who is responsible for deciding whether or not a set of 100 students succeeded their academic year. The students are assessed in 10 courses: mathematics, physics, chemistry, biology, finance, law, management, computer, sociology and marketing. Each student receives a score ranging between 0 and 20 in each course. Consider a subset of 5 students, James, John, Michael, Robert and David who obtained the ratings given in Table 6.1.

Table 6.1: Grades of the students in the five courses.

	mathematics	physics	chemistry	biology	finance	law	management	computer	sociology	marketing
James	**8**	17	15	18	17	15	19	18	14	15
John	16	13	17	16	18	18	14	16	18	**8**
Michael	17	18	14	17	**8**	14	17	18	16	**8**
Robert	18	17	19	**8**	**8**	15	15	19	19	**8**
David	19	18	17	**8**	13	**8**	19	17	15	**8**

In the european credit transfer and accumulation system (ECTS), each course is associated to a number of credits. In this example, we consider that all courses have the same number of credits. A student obtains the credits associated to a course if he/she has an evaluation greater than or equal to 10/20 in this course. The DM established that a student succeeds his year if he/she gets a score greater than or equal to 10/20 on at least 7 out of the 10 courses and if the list of courses in which he/she got an evaluation lower than 9/20 is not too important.

The DM, who decides whether a student is accepted, issues the following judgments for the 5 students. Given the scores of James, the DM considers that he should be refused because of his bad rating in mathematics, which is a course that matters very much for the next school year. On the contrary, the DM considers that John, who got a bad evaluations in marketing, should be accepted. He/She considers that marketing is not a significant course for being rejected and that the score of John should not be a veto for him to pass to the next year. Following the DM, Michael, who got bad results in marketing and finance, should be accepted. The DM states that a student as Robert having ratings below 9/20 in chemistry, marketing and finance should be refused. However, the DM judges that David can pass to the next year, even with the three bad ratings in biology, finance and marketing.

Modelling the preferences of the DM with a classical MR-Sort model without

veto is not possible. We remind that all the courses, i.e. criteria, have the same importance, i.e. number of ECTS credits. In MR-Sort, we model this by choosing equal weights (0.1) for all the criteria. As stated by the DM, a necessary condition to accept a student is that his/her evaluations are at least better than or equal to 10/20 in 7 courses. To model this requirement with a MR-Sort model, we define a profile having performances equal to 10/20 on all the criteria and the majority threshold is set to 0.7. Using the model as such, i.e. without veto, enables to restore the assignments of John, Michael and David. On the contrary, James and Robert are not correctly assigned since they are accepted disregard the contrary opinion of the DM.

In order to correct the assignments of James and Robert, we introduce the binary veto as defined by Bouyssou and Marchant (2007a,b) for the NCS model. We remind that a binary veto is effective for the assertion "a is at least as good as b^h" when the performance of an alternative a is worse than the profile b^h minus a veto threshold v_j on any criterion j. Correcting the assignment of James is done by setting a veto threshold at 9/20 in mathematics. Only Robert remains assigned in the wrong category, i.e. he is still accepted to pass to the next year with this model. In order to correct the assignment of Robert, one can consider setting a veto threshold at 9/20 either in biology, finance or marketing. However, if a veto threshold is set on any of these three courses, then the assignments of other students become incorrect. If a binary veto threshold is set to 9/20 in biology, then David is refused. If a binary veto threshold is set to 9/20 in finance, then Robert is refused. Finally, if a binary veto threshold is set to 9/20 in marketing, then Michael and John are also refused. This example shows that it is not possible to represent DM's acceptance rule with a standard binary veto. We understand the need for a more general veto rule.

In this example, all the courses have the same importance. However, when a student has bad ratings, i.e. ratings below 10/20, the DM considers that the courses do not have the same importance, e.g. mathematics is more important than marketing. Moreover, he/she also considers that coalitions of criteria on which a student has bad ratings do not have the same importance, e.g. the coalition [biology, finance, marketing] is more important than the coalition [biology, law, marketing]. We propose therefore a new veto rule which enables to differentiate coalitions of criteria.

6.2 Literature review

ELECTRE methods (Roy, 1968; Roy and Bouyssou, 1993) are based on the concordance and non-discordance principles. In these methods, an alternative a is preferred to another one b if two conditions are fulfilled:

1. There is a sufficient coalition of criteria on which a has at least as good performances as b;

2. The alternative a is not much worse than b on one or several attributes.

When the second condition does not hold, we say that there is "veto" for the assertion "a is preferred to b".

In ELECTRE I and ELECTRE II, veto applies for the assertion "a is preferred to b" if the difference between a and b, in favor of b, on any criterion j is greater than a veto threshold v_j, i.e. $b_j - a_j \geq v_j$. In ELECTRE III and ELECTRE TRI, the assertion "a is preferred to b" is valued with a fuzzy index called credibility threshold. This credibility index has been reminded in Chapter 2, Equation (2.4). In these methods, the veto doesn't always apply in an "all or nothing" way, instead it weakens the credibility index as a non linear function of the difference between the performances of the two alternatives and the concordance threshold. Roy and Bouyssou (1993) show that choosing veto thresholds close to the preference thresholds in conjunction with a credibility threshold close to 1 increases the conjunctive or disjunctive character of the ELECTRE TRI procedure. Bouyssou and Pirlot (2009) present several axiom of outranking relations based on concordance and non-discordance principles as in ELECTRE I and ELECTRE II.

Using ELECTRE III and ELECTRE TRI requires the elicitation of additional thresholds (indifference and preference thresholds) in order to compute the concordance and non-discordance indices. While they think that outranking methods are well adapted for environmental problems, Rogers and Bruen (1998) note that the use of ELECTRE methods might be seen as unconvincing if the values of these thresholds are too subjective. They propose a new approach to determine the indifference, preference and veto thresholds which takes uncertainty and human aspect into account. Similarly, Maciej (2004) notes that a DM can only evaluate the performances of alternatives in a probabilistic way. He proposes a procedure based on stochastic dominance to compute a credibility index in a similar way as for ELECTRE III.

To avoid to determine explicitly the veto thresholds in ELECTRE III and ELECTRE TRI, Dias and Mousseau (2006) proposed to learn the veto parameters through mathematical programming from statements provided by the DM. In the paper, they consider both learning the veto parameters criterion per criterion or all the veto parameters simultaneously.

Perny and Roy (1992) introduced fuzzy outranking relations in order to handle uncertain knowledge and conflicting preferences. The proposed model allows to compare pair of alternatives. In another paper, Perny (1998) proposes a method based on the fuzzy outranking relation in order to assign alternatives in predefined and possibly ordered categories.

Roy and Słowiński (2008) proposed two new credibility indices for ELECTRE III and ELECTRE TRI allowing to take two new effects into account. The first effect called "reinforced preference" increases the value of the credibility index when an alternative a is judged "very strongly preferred" to another alternative b. It is achieved by increasing the value of the partial concordance index (2.1) on criteria which are strongly in favor of the alternative a. The second effect, called "counter-veto" aims at reducing the veto effect for criteria on which the difference between the evaluations of a and b is greater than a value called counter-veto threshold.

In order to take conflicting information into account in preference statements, i.e. arguments in favour or in disfavour of a proposition a is preferred to b, Fortemps and Słowiński (2002) extend quadrivalent logic introduced by Belnap (1977). Quadrivalent logic consists in considering four states of knowledge with regard to a proposition: ignorance, truthfulness, falsity and contradiction. Fortemps and Słowiński (2002) propose to extend quadrivalent logic to handle the relevance of preference information in the context of ranking problems. For an assertion of the type "a is preferred to b", they propose to compute four outranking relations corresponding to the ignorance, truthfulness, falsity and contradiction of the assertion.

6.3 Veto rules

In this section, we recall the traditional binary veto rule as defined by Bouyssou and Marchant (2007a,b). Then we introduce a new veto rule which enables to take coalition of criteria into account.

6.3.1 Binary veto

In a MR-Sort model with binary 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. Formally, it is described in Chapter 2 by Equation (2.7). This equation can be expressed as follows:

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

with:

$$a\, V b^h \iff \exists j \in N : a_j \leq b_j^h - v_j^h,$$

where v_j^h denotes the veto threshold associated to the profile b^h on the criterion j. We remind that a profile b^h delimits the category C^h from C^{h+1}, with $C^{h+1} \succ C^h$.

With binary veto, the MR-Sort assignment rule corresponds to Equation (2.8) given in Chapter 2. We define the concept of *veto profile*:

Definition 8. *A veto profile v^h associated to a profile b^h is a vector of performances in which the performance on each criterion is equal to $b_j^h - v_j^h$.*

We remark that a MR-Sort model with more than 2 categories remains consistent only if veto profiles do not overlap. Otherwise, an alternative might be on one hand in veto against a profile b^h which prevents it to be assigned in C^{h+1} and on the other hand not in veto against b^{h+1} which do not prevent it to be assigned in C^{h+2}. It leads to the following proposition.

Proposition 2. *In MR-Sort, the binary veto rule remains consistent if the veto parameters v_j^h are chosen such that $b_j^h - v_j^h \geq b_j^k - v_j^k$ for all $\{h,k\} \in J$ with $h > k$.*

Proof. Assume v_j^h and v_j^k such that $b_j^h - v_j^h < b_j^k - v_j^k$ with $h > k$. Consider an alternative a such that $\sum_{a_j \geq b_j^h} w_j \geq \lambda$ which has worse performance than b^h on any criterion j such that $b_j^h - v_j^h < a_j < b_j^k - v_j^k$. Since a is below the veto profile v^k, we have $a\,V b^k$ which means that it cannot be assigned to category C^{k+1}. However, since a is above the veto profile v^h on j, it can be assigned in C^{h+1}. This assignment is inconsistent since $C^{h+1} \succ C^{k+1}$. On the contrary, if v_j^h and v_j^k are such that $b_j^h - v_j^h \geq b_j^k - v_j^k$ and the performance of a on criterion j is such that $b_j^h - v_j^h \geq a_j \geq b_j^k - v_j^k$, then we have either $a\,V b^h$ and $a\,V b^k$ if $a_j = b_j^h - v_j^h = b_j^k - v_j^k$ or $a\,V b^h$ and not $a\,V b^k$ if $b_j^h - v_j^h \geq a_j > b_j^k - v_j^k$. □

6.3.2 Coalitional veto

We introduce a new veto rule which considers coalitions of criteria. We call it "coalitional veto". With this rule, the veto applies for the assertion "a is preferred to b^h" when the performance of an alternative a is worse than $b_j^h - v_j^h$ on a weighted majority of criteria.

As in the binary veto, a veto threshold v_j^h is associated to each criterion j and each profile b^h. Coalitional veto involves other parameters. Firstly, it involves a set of veto weights denoted by z_j, for all $j \in N$. Without loss of generality, the sum of z_j is set to 1. Secondly, it involves a veto cut thresholds Λ determining whether a coalition of criteria is sufficient to impose a veto. Formally, we express the coalitional veto rule $a\,V b^h$, as follows:

$$a\,V b^h \iff \sum_{j:a_j \leq b_j^h - v_j^h} z_j \geq \Lambda. \tag{6.1}$$

Using coalitional veto, the outranking relation of MR-Sort (2.7) is modified as follows:

$$a \succcurlyeq b^h \iff \sum_{j:a_j \geq b_j^h} w_j \geq \lambda \text{ and } \sum_{j:a_j \leq b_j^h - v_j^h} z_j < \Lambda. \tag{6.2}$$

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

$$\begin{aligned} a \in C^h \iff & \left[\sum_{j:a_j \geq b_j^{h-1}} w_j \geq \lambda \text{ and } \sum_{j:a_j \leq b_j^{h-1} - v_j^{h-1}} z_j < \Lambda \right] \\ & \text{and } \left[\sum_{j:a_j \geq b_j^h} w_j < \lambda \text{ or } \sum_{j:a_j \leq b_j^h - v_j^h} z_j < \Lambda \right] \end{aligned} \tag{6.3}$$

In MR-Sort, the veto can be interpreted as a combination of performances preventing the assignment of an alternative to a category.

This new veto rule enables to obtain a model that is capable to restore the classification rule described in the example of Section 6.1. We call this new model, MR-Sort model with coalitional veto (MR-Sort-CV).

In the sorting context, the following propositions hold.

Proposition 3. *If there is veto for the assertion "a is preferred to b^h", then there is also veto for the assertion "a is preferred to b^k" for any $k > h$, with $b_j^h - v_j^h \geq b_j^k - v_j^k$ for all $j \in N$.*

Proof. We have $a\, V b^h$ if a has worse performance than the veto profile v^h on a weighted majority of criteria. Proposition 2 ensures a dominance relation between the veto profiles. Therefore, alternative a which has worse performances than v^h on a majority of criteria will also have worse performances than the veto profile v^k with $k > h$ on at least an as large majority of criteria. □

Proposition 4. *If there is veto for the assertion "a is preferred to b^h", then the veto also applies for the assertion "a' is preferred to b^h" if a' is weaker than a on all the criteria.*

Proof. Alternative a is in veto condition against b^h because it is worse than the veto profile a subset of criteria K such that $\sum_{j \in K} z_j \geq \Lambda$. As the alternative a' has worse performances than a on all the criteria, the subset of criteria K' on which a is worse than v^h includes the subset K. Therefore we have $\sum_{j \in K'} z_j \geq \sum_{j \in K} z_j \geq \Lambda$. □

Proposition 5. *There is never veto for the assertion "a is preferred to b^h" if a has at least as good performances as b^h on a subset of criteria K such that $\sum_{j\in K} z_j > 1-\Lambda$.*

Proof. Consider an alternative a better than a profile b^h on a subset of criteria K such that $\sum_{j\in K} z_j > 1-\Lambda$. It implies that $\sum_{j\in N\setminus K} z_j < \Lambda$ and therefore the veto never applies for the assertion "a is preferred to b^h". □

Proposition 6. *A subset of criteria K such that $\sum_{j\in K} w_j \geq \lambda$ has no influence in the veto rule of a MR-Sort model in which $\lambda > 0.5$.*

Proof. Consider an alternative a having worse performances than a profile b^h on a subset of K criteria such that $\sum_{j\in K} w_j \geq \lambda$. By definition of the MR-Sort model, we have $\sum_{j\in N\setminus K} w_j = 1 - \sum_{j\in K} w_j \leq 1-\lambda < \lambda$, as $\lambda > 0.5$. Therefore a does not outrank b^h no matter the veto rule. □

6.3.3 Links between binary veto and coalitional veto

The coalitional veto rule given in Equation (6.2) is a generalization of the binary rule. Indeed, if the veto cut thresholds Λ is equal to $\frac{1}{n}$, with n being the number of criteria, and each veto weight z_j is equal to $\frac{1}{n}$, then the veto rule defined in Equation (6.1) works as a binary veto.

6.3.4 Coalitional veto with same concordance and veto weights

Using coalitional veto in the outranking relation (6.2) implies the definition of $n(p-1)+n+1$ extra parameters. In total, $2(np+1)$ parameters have to be determined in a MR-Sort model using coalitional veto. In order to reduce the number of parameters, one can use the same weights for the veto and concordance profiles. It reduces the number of parameters to $2(np+1)-n$. However it also limits the flexibility of the model (e.g. the introductory example cannot be solved by using this simplified version). Formally, it amounts to impose $z_j = w_j$ for all $j \in N$. The outranking rule (6.2) then reads:

$$a \succcurlyeq b^h \iff \sum_{j:a_j \geq b_j^h} w_j \geq \lambda \text{ and } \sum_{j:a_j \leq b_j^h - v_j^h} w_j < \Lambda.$$

Proposition 7. *In a MR-Sort model with coalitional veto using equal veto weights, the veto rule has influence only if the veto threshold should be chosen such that $\Lambda \leq 1-\lambda$.*

Proof. Assume that $\sum_{j\in K} w_j \geq \lambda$ when $K = \{j : a_j \geq b_j^h\}$. Then $\sum_{j\in N\setminus K} w_j = 1 - \sum_{j\in K} w_j \leq 1 - \lambda$. Hence if $1 - \lambda < \Lambda$, then the coalitional veto can never be triggered. Thus $1 - \lambda \geq \Lambda$. □

Note that the above theorem gives a necessary but not sufficient condition: Λ should be chosen such that $\Lambda \leq 1 - \min \sum_{j\in K} w_j$ with $K : \sum_{j\in K} w_j \geq \lambda$.

6.4 Mixed integer program for learning the parameters of a majority rule sorting model using coalitional veto

In this section, we describe a mixed integer program (MIP) for learning the parameters of a MR-Sort model using coalitional veto. First we describe the MIP for learning the parameters of a MR-Sort model using coalitional veto. Then we describe a variant of the mixed integer program in which the veto weights are the same as the one used for the concordance.

6.4.1 MR-Sort-CV model using independent weights

Learning the parameters of a MR-Sort model without veto using mixed integer programming has been already studied in Leroy et al. (2011). In this section, we propose a linear program designed for learning the parameters of a MR-Sort model with coalitional veto.

The assignment of an alternative into category C^h holds if conditions of Equation (6.3) are satisfied. We express these conditions as follows:

$$a \in C^h \iff \begin{cases} \sum_{j:a_j \geq b_j^{h-1}} w_j \geq \lambda & \text{and } \sum_{j:a_j \leq b_j^{h-1} - v_j^{h-1}} z_j < \Lambda, \\ \sum_{j:a_j \geq b_j^h} w_j < \lambda & \text{or } \sum_{j:a_j \leq b_j^h - v_j^h} z_j \geq \Lambda. \end{cases}$$

In order to model this using linear constraints, we rewrite these conditions as follows:

$$a \in C^h \iff \begin{cases} \sum_{j=1}^n c_{a,j}^{h-1} \geq \lambda & \text{and } \sum_{j=1}^n \mu_{a,j}^{h-1} < \Lambda, \\ \sum_{j=1}^n c_{a,j}^h < \lambda & \text{or } \sum_{j=1}^n \mu_{a,j}^h \geq \Lambda. \end{cases} \tag{6.4}$$

with $c_{a,j}^l$ and $\mu_{a,j}^l$ for $l = \{h-1, h\}$ such that:

$$c_{a,j}^l = \begin{cases} w_j & \text{if } a_j \geq b_j^l, \\ 0 & \text{if } a_j < b_j^l, \end{cases}$$

and

$$\mu_{a,j}^l = \begin{cases} z_j & \text{if } a_j \leq b_j^l - v_j^l, \\ 0 & \text{if } a_j > b_j^l - v_j^l. \end{cases}$$

In order to make these constraints linear, two binary variables are introduced:

$$\delta^l_{a,j} = \begin{cases} 1 & \text{if } a_j \geq b^l_j, \\ 0 & \text{if } a_j < b^l_j, \end{cases}$$

and

$$\nu^l_{a,j} = \begin{cases} 1 & \text{if } a_j \leq b^l_j - v^l_j, \\ 0 & \text{if } a_j > b^l_j - v^l_j. \end{cases}$$

The value $\delta^l_{a,j}$ and $\nu^l_{a,j}$ are obtained thanks to the following constraints:

$$\begin{cases} a_j - b^l_j & < M\delta^l_{a,j}, \\ a_j - b^l_j & \geq M(\delta^l_{a,j} - 1), \\ a_j - b^l_j + v^l_j & > -M\nu^l_{a,j}, \\ a_j - b^l_j + v^l_j & \leq M(1 - \nu^l_{a,j}). \end{cases}$$

Finally the value of $c^l_{a,j}$ and $\mu^l_{a,j}$ are obtained as follows:

$$\begin{cases} c^l_{a,j} & \leq \delta^l_{a,j}, \\ c^l_{a,j} & \leq w_j, \\ c^l_{a,j} & \geq \delta^l_{a,j} - 1 + w_j, \\ \mu^l_{a,j} & \leq \nu^l_{a,j}, \\ \mu^l_{a,j} & \leq z_j, \\ \mu^l_{a,j} & \geq \delta^l_{a,j} - 1 + z_j. \end{cases}$$

Using the values of $c^l_{a,j}$ and $\mu^l_{a,j}$, it is possible to determine the concordance and coalitional veto.

We define two objectives for the mathematical program:

1. The first objective consists in maximizing the compatibility of the model with the learning set, i.e. restoring the assignments of a maximum number of examples with the learned model.

2. The second objective consists in minimizing the number of veto. The program tries to learn a model in which the veto applies to as few examples as possible.

The two objectives are used in a lexicographic order. The priority is given to the first objective which maximizes the compatibility of the learning set with the learned model.

To achieve the first objective, it is needed to add a binary variable for each alternative of the learning set that indicates whether the alternative is assigned in the correct category. The variable is denoted by γ_a and is such that:

$$\gamma_a = \begin{cases} 1 & \text{if } a \text{ is assigned in the correct category,} \\ 0 & \text{if } a \text{ is assigned in an incorrect category.} \end{cases}$$

In a model without veto, the value of γ_a is obtained by modifying the left part of the constraints given at Equation (6.4) as follows:

$$a \in C^h \iff \begin{cases} \sum_{j=1}^{n} c_{a,j}^{h-1} \geq \lambda + M(\gamma_a - 1), \\ \sum_{j=1}^{n} c_{a,j}^{h} < \lambda - M(\gamma_a - 1). \end{cases} \tag{6.5}$$

To take the veto into account, two binary variables are introduced for each example of the learning set. The binary variables associated to each example indicate whether veto effect applies regarding the upper and lower profiles of the category in which the example is assigned. The variable is denoted by ω_a^l, with $l = \{h-1, h\}$. It takes the following value:

$$\omega_a^l = \begin{cases} 1 & \text{if veto applies for alternative } a \text{ against profile } l, \\ 0 & \text{otherwise.} \end{cases}$$

Formally, we have:

$$\omega_a^l = \begin{cases} 1 & \text{if } \sum_{j=1}^{n} \mu_{a,j}^l \geq \Lambda, \\ 0 & \text{if } \sum_{j=1}^{n} \mu_{a,j}^l < \Lambda. \end{cases}$$

The value of ω_a^l is obtained thanks to the following constraints:

$$\begin{cases} \sum_{j=1}^{n} \mu_{a,j} - \Lambda \geq M(\omega_a^h - 1), \\ \sum_{j=1}^{n} \mu_{a,j} - \Lambda < M\omega_a^h. \end{cases}$$

In order to take the veto effect into account, variables ω_a^{h-1} and ω_a^h are added in constraints given in Equation (6.5):

$$a \in C^h \iff \begin{cases} \sum_{j=1}^{n} c_{a,j}^{h-1} - \omega_a^{h-1} \geq \lambda + M(\gamma_a - 1), \\ \sum_{j=1}^{n} c_{a,j}^{h} - \omega_a^{h} < \lambda - M(\gamma_a - 1). \end{cases}$$

Recall that we want to maximize the compatibility of the learning set with the learned model and minimize the veto effect. To satisfy these requirements, we use the following objective function:

$$\max_{\substack{w_j, \lambda, b_j^h \\ z_j, \Lambda, v_j^h}} \sum_{a \in A} \gamma_a - \frac{1}{2\,|a \in A \backslash A^1|} \sum_{a \in A \backslash A^1} \omega_a^{h-1} - \frac{1}{2\,|a \in A \backslash A^p|} \sum_{a \in A \backslash A^p} \omega_a^h$$

where A^1 (resp. A^p) denotes the set of alternatives assigned in C^1 (resp. C^p). The first part of the objective function maximizes the sum of γ_a in order to maximize the compatibility of the model with the largest possible number of alternatives. The second part of the objective function minimizes the sum of ω_a^l with $l = \{h-1, h\}$ in order to minimize the number of veto effects. We remark the component of the second part of the objective function are weighted such that the MIP will give priority to the maximization of the model compatibility with the learning set.

The objective function coupled to the constraints listed above lead to the following mixed integer program:

$$\max_{\substack{w_j,\lambda,b_j^h \\ z_j,\Lambda,v_j^h}} \sum_{a\in A} \gamma_a - \frac{1}{2\left|a \in A \backslash A^1\right|} \sum_{a\in A\backslash A^1} \omega_a^{h-1} - \frac{1}{2\left|a \in A \backslash A^p\right|} \sum_{a\in A\backslash A^p} \omega_a^h,$$

such that:

$$\left\{\begin{array}{rcll}
\sum_{j=1}^n c_{a,j}^{h-1} - \omega_a^{h-1} & \geq & \lambda + M(\gamma_a - 1) & \forall a \in A^h, \forall h \in H, \\
\sum_{j=1}^n c_{a,j}^{h} - \omega_a^{h} & < & \lambda - M(\gamma_a - 1) & \forall a \in A^h, \forall h \in H \\
a_j - b_j^l & < & M\delta_{a,j}^l & \forall a \in A^h, \forall h \in H, \\
 & & & l = \{h-1, h\} \setminus \{0, p\}, \forall j \in N, \\
a_j - b_j^l & \geq & M(\delta_{a,j}^l - 1) & \forall a \in A^h, \forall h \in H, \\
 & & & l = \{h-1, h\} \setminus \{0, p\}, \forall j \in N, \\
a_j - b_j^l + v_j^l & > & -M\nu_{a,j}^l & \forall a \in A^h, \forall h \in H, \\
 & & & l = \{h-1, h\} \setminus \{0, p\}, \forall j \in N, \\
a_j - b_j^l + v_j^l & \leq & M(1 - \nu_{a,j}^l) & \forall a \in A^h, \forall h \in H, \\
 & & & l = \{h-1, h\} \setminus \{0, p\}, \forall j \in N, \\
c_{a,j}^l & \leq & \delta_{a,j}^l & \forall a \in A^h, \forall h \in H, \\
 & & & l = \{h-1, h\} \setminus \{0, p\}, \forall j \in N, \\
c_{a,j}^l & \leq & w_j & \forall a \in A^h, \forall h \in H, \\
 & & & l = \{h-1, h\} \setminus \{0, p\}, \forall j \in N, \\
c_{a,j}^l & \geq & \delta_{a,j}^l - 1 + w_j & \forall a \in A^h, \forall h \in H, \\
 & & & l = \{h-1, h\} \setminus \{0, p\}, \forall j \in N, \\
\mu_{a,j}^l & \leq & \nu_{a,j}^l & \forall a \in A^h, \forall h \in H, \\
 & & & l = \{h-1, h\} \setminus \{0, p\}, \forall j \in N, \\
\mu_{a,j}^l & \leq & z_j & \forall a \in A^h, \forall h \in H, \\
 & & & l = \{h-1, h\} \setminus \{0, p\}, \forall j \in N, \\
\mu_{a,j}^l & \geq & \nu_{a,j}^l - 1 + z_j & \forall a \in A^h, \forall h \in H, \\
 & & & l = \{h-1, h\} \setminus \{0, p\}, \forall j \in N, \\
\sum_{j=1}^n \mu_{a,j}^l - \Lambda & \geq & M(\omega_a^l - 1) & \forall a \in A^h, \forall h \in H, \\
 & & & l = \{h-1, h\} \setminus \{0, p\}, \\
\sum_{j=1}^n \mu_{a,j}^l - \Lambda & < & M\omega_a^l & \forall a \in A^h, \forall h \in H, \\
 & & & l = \{h-1, h\} \setminus \{0, p\}, \\
\sum_{j=1}^n w_j & = & 1, & \\
\sum_{j=1}^n z_j & = & 1, & \\
b_j^h & \geq & b_j^{h-1} & h = \{2, \ldots, p-1\}, \forall j \in J, \\
b_j^h - v_j^h & \geq & b_j^{h-1} - v_j^{h-1} & h = \{2, \ldots, p-1\}, \forall j \in J,
\end{array}\right.$$

with:

$$\begin{cases} w_j & \in & [0,1] & & \forall j \in N, \\ z_j & \in & [0,1] & & \forall j \in N, \\ c^l_{a,j} & \in & [0,1] & \forall a \in A^h, \forall h \in H, l = \{h-1,h\} \setminus \{0,p\}, \forall j \in N, \\ \mu^h_{a,j} & \in & [0,1] & \forall a \in A^h, \forall h \in H, l = \{h-1,h\} \setminus \{0,p\}, \forall j \in N, \\ \delta^h_{a,j} & = & \{0,1\} & \forall a \in A^h, \forall h \in H, l = \{h-1,h\} \setminus \{0,p\}, \forall j \in N, \\ \gamma_a & = & \{0,1\} & & \forall a \in A, \\ \nu^h_{a,j} & = & \{0,1\} & \forall a \in A^h, \forall h \in H, l = \{h-1,h\} \setminus \{0,p\}, \forall j \in N, \\ \omega^l_a & = & \{0,1\} & \forall a \in A^h, \forall h \in H, l = \{h-1,h\} \setminus \{0,p\}, \forall j \in N, \\ \lambda & \in & [0.5,1], & & \\ \Lambda & \in & [0,1], & & \\ b^h_j & \in & \mathbb{R} & & \forall h \in H, \forall j \in N. \end{cases}$$

6.4.2 MR-Sort-CV model with veto profiles on concordance profiles

A particular case of the model consists in choosing veto profiles having performances equal to the performances of lower concordance profiles, i.e. $v^h_j = b^h_j - b^l_j$ for any $l < h$. In this particular case, no veto profile is associated to the profile delimiting the worst category from the others, i.e. b^1 which delimits C^1 from C^2 (with $C^2 \succ C^1$). In order to obtain such a model, we modify the MIP presented above as follows. We introduce $h-1$ binary variable $\rho_{h,l,j}$ for each profile performance b^h_j. It takes the following value:

$$\rho_{h,l,j} = \begin{cases} 1 & \text{if } v^h_j = b^h_j - b^l_j \text{ for any } l < h, \\ 0 & \text{if } v^h_j \neq b^h_j - b^l_j \text{ for any } l < h. \end{cases}$$

We impose $\sum_{l=1}^{h-1} \rho_{h,l,j} \leq 1$. Finally, the value of v^h_j is obtained by adding the following constraints to the MIP of Section 6.4.1:

$$\begin{cases} b^h_j - v^h_j & \leq & b^l_j + M(1-\rho_{h,l,j}) & h = \{2,\ldots,p-1\}, \\ & & & l = \{1,\ldots,h-1\}, \forall j \in N, \\ b^h_j - v^h_j & \geq & b^l_j - M(1-\rho_{h,l,j}) & h = \{2,\ldots,p-1\}, \\ & & & l = \{1,\ldots,h-1\}, \forall j \in N, \\ b^1_j - v^1_j & = & \underline{a_j} & \forall j \in N, \\ \sum_{l=1}^{h-1} \rho_{h,l,j} & \leq & 1 & h = \{2,\ldots,p-1\}, \forall j \in N, \\ \rho_{h,l,j} & \in & \{0,1\} & \forall h, l = \{0,\ldots,h-1\}, \forall j \in N. \end{cases}$$

Note that adding these constraints does not simplify the formulation of the MIP described above since it adds new binary variables in the program.

6.4.3 Example

In this subsection, we illustrate how the MIP described in Section 6.4.1 works on a small example. To test this inference program, we build a MR-Sort model which will be used to determine whether a student is accepted or refused for next year based on his/her ratings in five courses evaluated on a scale ranging between 0 and 20. The DM states that a student is accepted if the following two conditions are fulfilled:

1. The student has a score greater than or equal to 10/20 on at least 3 out of the 5 courses;

2. His/Her scores are not lower than or equal to 8/20 on more than one course.

If one of these two conditions is not fulfilled, then the student is refused.

To model the DM preference, we define a profile b^1 delimiting the category *accepted* from the category *refused*. This profile divides the domain of each criterion in two parts, such that $b_j^1 = 10$ for $j = \{1, \ldots, 5\}$. The DM considers that all the courses have the same importance. We model this by choosing the weights such that $w_j = \frac{1}{5}$ for $j = \{1, \ldots, 5\}$. The majority threshold λ is set to $\frac{3}{5}$ such that a student should have ratings greater than or equal to the profile b^1 on at least 3 out of the 5 criteria to be potentially accepted.

To model the second condition, a veto threshold is added on every profile such that $v_j^1 = 2$ for $j = \{1, \ldots, 5\}$. Veto weights are chosen equal to the concordance weights. The threshold Λ is chosen equal to $\frac{2}{5}$ so that a student is refused if he/she has a score smaller than or equal to 8/20 on more than one course.

To show that the MIP described in the previous section works as expected, we first generate a set of 30 $(= 2^5 - 2)$ example students that allows to learn the concordance relation. The scores of the students are chosen just above and below 10/20 such that all the possible coalitions are represented. Students with scores overall above or below 10/20 are not necessary in the learning set because we know whether the concordance is satisfied or not for them. In order to learn the veto threshold, we define a second set of students representing all the possible veto coalitions. Table 6.2 contains the scores of the 30 students and the category they are assigned in. The scores and their associated assignments are given as input to the MIP.

Table 6.2: Students evaluations and assignments.

Student	c_1	c_2	c_3	c_4	c_5	Category
$a^{(1)}$	9	9	9	9	11	refused
$a^{(2)}$	9	9	9	11	9	refused

Continued on next page

Table 6.2 – *Continued from previous page*

Student	c_1	c_2	c_3	c_4	c_5	Category
$a^{(3)}$	9	9	9	11	11	refused
$a^{(4)}$	9	9	11	9	9	refused
$a^{(5)}$	9	9	11	9	11	refused
$a^{(6)}$	9	9	11	11	9	refused
$a^{(7)}$	9	9	11	11	11	accepted
$a^{(8)}$	9	11	9	9	9	refused
$a^{(9)}$	9	11	9	9	11	refused
$a^{(10)}$	9	11	9	11	9	refused
$a^{(11)}$	9	11	9	11	11	accepted
$a^{(12)}$	9	11	11	9	9	refused
$a^{(13)}$	9	11	11	9	11	accepted
$a^{(14)}$	9	11	11	11	9	accepted
$a^{(15)}$	9	11	11	11	11	accepted
$a^{(16)}$	11	9	9	9	9	refused
$a^{(17)}$	11	9	9	9	11	refused
$a^{(18)}$	11	9	9	11	9	refused
$a^{(19)}$	11	9	9	11	11	accepted
$a^{(20)}$	11	9	11	9	9	refused
$a^{(21)}$	11	9	11	9	11	accepted
$a^{(22)}$	11	9	11	11	9	accepted
$a^{(23)}$	11	9	11	11	11	accepted
$a^{(24)}$	11	11	9	9	9	refused
$a^{(25)}$	11	11	9	9	11	accepted
$a^{(26)}$	11	11	9	11	9	accepted
$a^{(27)}$	11	11	9	11	11	accepted
$a^{(28)}$	11	11	11	9	9	accepted
$a^{(29)}$	11	11	11	9	11	accepted
$a^{(30)}$	11	11	11	11	9	accepted
$a^{(31)}$	11	11	11	11	7	accepted
$a^{(32)}$	11	11	11	7	11	accepted
$a^{(33)}$	11	11	7	11	11	accepted
$a^{(34)}$	11	7	11	11	11	accepted
$a^{(35)}$	7	11	11	11	11	accepted
$a^{(36)}$	11	11	11	7	7	refused
$a^{(37)}$	11	11	7	11	7	refused
$a^{(38)}$	11	7	11	11	7	refused

Continued on next page

Table 6.2 – *Continued from previous page*

Student	c_1	c_2	c_3	c_4	c_5	Category
$a^{(39)}$	7	11	11	11	7	refused
$a^{(40)}$	11	11	7	7	11	refused
$a^{(41)}$	11	7	11	7	11	refused
$a^{(42)}$	7	11	11	7	11	refused
$a^{(43)}$	11	7	7	11	11	refused
$a^{(44)}$	7	11	7	11	11	refused
$a^{(45)}$	7	7	11	11	11	refused

The MIP infers model parameters based on the assignment examples given in Table 6.2. The parameters of the MR-Sort model found by the linear program are given in Table 6.3. This model enables to restore all the assignments examples given as input.

Table 6.3: Parameters of the model found by the MR-Sort-CV MIP.

	c_1	c_2	c_3	c_4	c_5
b_1	9.0001	9.0001	9.0001	9.0001	9.0001
v_1	0.0002	0.0002	0.0002	0.0002	0.0002
w_j	0.2	0.2	0.2	0.2	0.2
λ			0.6		
Λ			0.4		

We ran the MIP presented in Leroy et al. (2011) to see how many examples can be restored with a MR-Sort model without veto. The parameters of the MR-Sort model returned by the MIP are given in Table 6.4. With this model, it is possible to restore 39 out of the 45 assignment examples, i.e. 86% of the examples. This illustrative example shows that the MR-Sort-CV increases the expressivity of a classical MR-Sort model.

Table 6.4: Parameters of the model found by the MR-Sort MIP.

	c_1	c_2	c_3	c_4	c_5
b_1	9.0001	9.0001	9.0001	9.0001	7.0001
w	0.249975	0.249975	0.249975	0.249975	0.0001
λ			0.750025		

6.5 Metaheuristic for learning the parameters of a MR-Sort-CV model

We know that inferring the parameters of a MR-Sort model with a MIP takes a lot of computing time and can therefore be only considered for small problems (Leroy et al., 2011). A MR-Sort-CV involves more parameters than a MR-Sort model. The formulation of the MIP described in Section 6.4 includes even more variables and constraints than the MIP for learning a MR-Sort model without veto (Leroy et al., 2011). Consequently, the computing time needed to infer the parameters of a MR-Sort-CV model will be at least as large as for learning the parameters of a MR-Sort model without veto.

In this section we describe a new strategy in order to integrate the inference of veto parameters in the algorithm designed for learning a MR-Sort model from large set of examples which has been described in Chapter 3 (Algorithm 2). Algorithm 5 gives a global overview of the proposed strategy for integrating the inference of the coalitional veto in this algorithm.

Algorithm 5 Metaheuristic to learn all the parameters of an MR-Sort-CV model.

Generate a population of N_{model} models with profiles initialized with a heuristic
repeat
  for all model M of the set **do**
    Learn the weights and majority threshold with a linear program, using the current profiles
    Adjust the profiles with an heuristic N_{it} times, using the current weights and threshold.
    Alternatives subject to veto are identified
    A set of veto weights and majority threshold are learned with a linear program.
    A set of veto profiles are computed
  end for
  Reinitialize the $\lfloor \frac{N_{model}}{2} \rfloor$ models giving the worst CA
until Stopping criterion is met

After running the heuristic adjusting the profiles, veto parameters are identified with a heuristic algorithm which works as follows:

1. For each profile b^h, all the alternatives of the learning set that should be assigned to C^h assigned by the current model to a category better than C^h are identified. We denote this set of alternatives by $A^{*h}_{>h}$. For instance, in a model involving three categories ($C^3 \succ C^2 \succ C^1$), it consists of identifying the alternatives belonging to the sets A^{*1}_2, A^{*1}_3 and A^{*2}_3. Note that

all the alternatives identified at this step are not necessarily wrongly classified because of a missing veto. Indeed some of the alternatives might be wrongly assigned because the profile is too low on some criteria or because its assignment is altered.

2. Similarly, for each category C^h, we identify all the alternatives that are correctly assigned by the model to C^h having at least one performance below the lower profile of C^h, i.e. b^{h-1}. We denote this set of alternatives by $A^{*h}_{h\lhd}$. The assignments of these alternatives should not be altered by a veto.

3. The alternatives of the sets $A^{*h}_{>h}$ and $A^{*h}_{h\lhd}$ are used as input of a modified version of the linear program (LP) described in Section 3.3.3. The constraints of this LP are modified as follows:

$$\begin{cases} \sum_{j:a_j<b_j^h} z_j - x_a + x'_a & = \Lambda & \forall a \in A^{*h}_{>h}, h = \{1,\dots,p-1\}, \\ \sum_{j:a_j\geq b_j^{h-1}} z_j + y_a - y'_a & = \Lambda - \varepsilon & \forall a \in A^{*h}_{h\lhd}, h = \{2,\dots,p\}, \\ \sum_{j=1}^{n} z_j & = 1, & \\ z_j & \in [0;1] & j = 1,\dots,n, \\ \Lambda & \in [0.5;1], & \\ x_a, x'_a & \in \mathbb{R}_0^+ & a \in A^{*h}_{>h}, \\ y_a, y'_a & \in \mathbb{R}_0^+ & a \in A^{*h}_{h\lhd}. \end{cases}$$

The objective function of the LP remains the same as the one of the LP described in Section 3.3.3, i.e. minimizing the sum of slack variables x'_a and y'_a.

4. Once a set of veto weights and a veto cut threshold have been learned, a veto profile is computed for each profile b^h with $h = \{1,\dots,p-1\}$. Of course the dominance constraints between veto profiles are respected. To compute the veto profile, we use a modified version of the heuristic used in Chapter 3 to compute the concordance profiles.

Note that this is a simple draft of the algorithm. More things have to be precised in order to obtain a full implementation. It is for instance needed to modify the linear program that infer the weights and majority threshold and the heuristic that modifies the profiles in order to take the veto into account.

Another approach consists in proceeding as in Section 6.4.2, i.e. using the same performances as lower concordance profiles for the veto profile. This approach reduces the space of solutions but simplifies the problem. For instance, the veto profile associated to the profile b^2 can take only two values on each criterion j: either $b_j^2 - v_j^2 = b_j^1$ or $b_j^2 - v_j^2 = \underline{a_j}$. One can also consider using same veto and concordance weights in order to simplify the problem.

6.6 Chapter conclusion

In this chapter, we presented a new type of veto rule. We called it "coalitional veto". It amounts to consider that an alternative is in veto against a profile if it is worse than a veto profile on a subset of criteria. A mixed integer program designed for the inference of the parameters of a MR-Sort model using coalitional veto has been proposed. The MIP is able to deal with small data sets as illustrated in this chapter. The outline of an extension of the metaheuristic dedicated to learn the parameters of a MR-Sort-CV model has been also presented in this chapter.

We see two directions for further research. The first one consists of characterizing axiomatically this new veto type in order to understand it better. The second direction of research consists of finding new techniques in order to learn all the parameters of a MR-Sort-CV model at the same time without involving too much computing time.

Chapter 7

UTA-poly and UTA-splines: additive value functions with polynomial marginals

In Chapter 2, we presented the additive value function (AVF) model (Section 2.2.4). In such a model, a numerical value is associated to each alternative involved in the decision problem. It is computed by aggregating the scores of the alternatives on the different criteria. The score of an alternative on a criterion is determined by a marginal value function that evolves monotonically as a function of the performance of the alternative on this criterion. Determining the shape of those marginals is not easy for a decision maker (DM). It is easier for him/her to make statements such as "alternative a is preferred over alternative b". In order to help the DM, UTA disaggregation procedures use linear programming to approximate the marginals by piecewise linear functions based only on such statements. In this chapter, we propose to infer polynomials and splines instead of piecewise linear functions for the marginals. In this aim, we use semidefinite programming (SDP) instead of linear programming. We illustrate this new elicitation method and present some experimental results with artificial and real data sets.

7.1 UTA-poly: additive value functions with polynomial marginals

In this section we present a new way to elicit marginal value functions using semidefinite programming. We first give the motivations for using this new method. Then we describe it.

7.1.1 Motivation

UTA methods use piecewise linear functions to model the marginal value functions. Opting for such functions shapes allows to use the linear programs presented in Chapter 2, Section 2.4.3, and linear programming solvers to infer an additive value ranking or sorting model. However by considering piecewise linear marginals with breakpoints at predefined places, original UTA methods have two important drawbacks: these options limit the interpretability and the flexibility of the additive value model.

Interpretability

There is a longstanding tradition in Economics, especially in the classical theory of consumer behavior (see e.g. Silberberg and Suen (2001)), which assumes that utility (or value) functions are differentiable and interpret their first and second (partial) derivatives in relation with the preferences and behavior of the customer. Multiple criteria decision analysis, based on value functions, stems from the same tradition. Trade-offs or marginal rates of substitution are generally thought of as changing smoothly (see e.g. Keeney and Raiffa (1976), p. 83 : "Throughout we assume that we are in a well-behaved world where all functions have smooth second derivatives"). Although piecewise linear marginals can provide good approximations for the value of any derivable function, they are not fully satisfactory as an explanatory model. This is especially the case when the breakpoints are fixed arbitrarily (e.g. equally spaced in the criterion domains). Such a choice may well fail to correctly reflect the DM's feelings about where the marginal rate of substitution starts to grow more quickly (resp. to diminish) or shows an inflexion. In other words, the qualitative behavior of the first and second derivatives of the "true" marginal value function might be poorly approximated by resorting to piecewise linear models, while this behavior might have an intuitive meaning for the DM. Therefore, considering piecewise linear marginals might lead to final models that fail to convince the DM even though they fit the learning set accurately.

Flexibility

Restricting the shape of the marginals to piecewise linear functions may hamper the expressivity of the additive value function model. This is especially detrimental when large learning sets are available as is the case in machine learning (ML) applications[1].

[1] It is seldom so in multiple-criteria decision analysis (MCDA) applications where the size of the learning set rarely exceeds a few dozens of records.

The following *ad hoc* case aims to illustrate the loss in flexibility incurred due to the piecewise linear hypothesis. Consider a ranking problem in which alternatives are assessed on two criteria. The DM states that the top-ranked alternatives are a, b, which are tied (rank 1), followed by c (rank 2) while d is strictly less preferred than the others (rank 3). The evaluations and ranks of these alternatives are displayed in Table 7.1.

Table 7.1: Example of an alternatives ranking that is not representable with a UTA model (one linear piece per marginal).

alternative	criterion 1	criterion 2	rank
a	100	0	1
b	0	100	1
c	25	75	2
d	75	25	3

Assume that we plan to use a UTA model with marginals involving a single linear piece (i.e. a weighted sum). Such an UTA model cannot at the same time distinguish c and d and express that a and b are tied. The fact that a and b are tied indeed implies that the criteria weights are equal (we can set them to 0.5 without loss of generality). The value on each marginal varies from 0 to 0.5. The worst value (0) corresponds to the worst performance (0) and the best value (0.5) to the best performance (100) on each criterion (see the marginal value functions represented by dashed lines in Figure 7.1). Using these marginals, the scores of the four alternatives are obtained through linear interpolation and displayed in Table 7.2. We observe that all alternatives receive the same value 0.5. It is therefore not possible to discriminate alternatives c and d.

Table 7.2: UTA and UTA-poly scores of the alternatives described in Table 7.1 with the UTA and UTA-poly marginals represented in Figure 7.1.

	a	b	c	d
UTA score	0.5	0.5	0.5	0.5
UTA-poly score	0.5	0.5	0.46	0.33

In case polynomials are allowed for, instead of piecewise linear functions, to model the marginals, the DM's preferences can be accurately represented. Figure 7.1 shows the case of polynomials of degree 3 used as marginals (plain line). The scores of the alternatives computed with these marginals are displayed in Table 7.2. They comply with the DM's preferences.

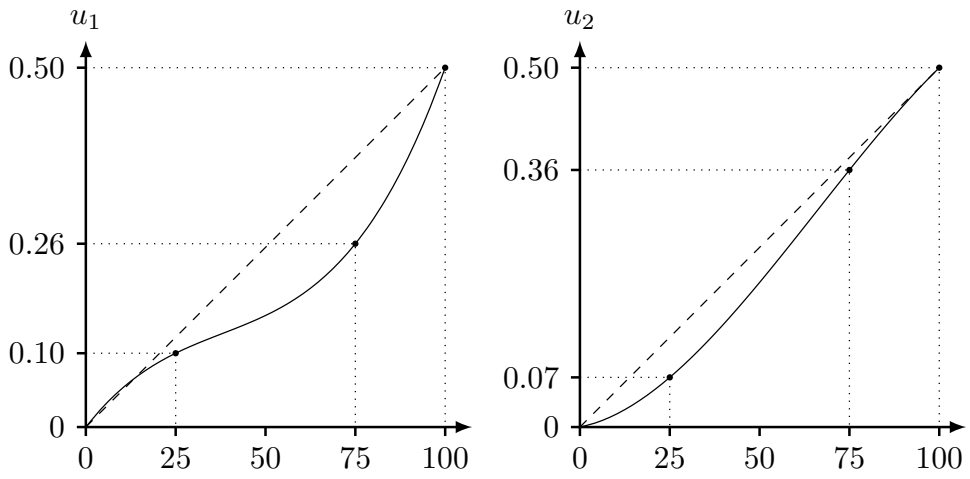


Figure 7.1: Example of UTA and UTA-poly value functions. The dashed lines correspond to the UTA piecewise linear function and the plain lines correspond to polynomials of degree 3.

Obviously it would have been possible to reproduce the DM's ranking using more than one linear piece marginals in an UTA model. However, when the breakpoints are fixed in advance, it is easy to construct an example, similar to the above one, in which the DM's ranking cannot be reproduced using a linear function between successive breakpoints while a polynomial (spline) will do.

The two methods introduced below, UTA-poly in the rest of this section and UTA-splines in Section 7.2, replace the piecewise linear marginals of UTA by polynomials and polynomial splines, respectively.

7.1.2 Basic facts about non-negative polynomials

In the last few years, significant improvements have been made in formulating and solving optimization problems in which constraints are expressed in the form of polynomial (in)equalities and with a polynomial objective function; see, e.g., Henrion and Lasserre (2003); Henrion et al. (2009). These new techniques are useful for various applications; see Lasserre (2009) and the references therein. A problem arising in many applications, including the present one, is to guarantee the non-negativity of functions of several variables. In our case, we have to make sure not only that marginals are non negative but also that they are non-decreasing, i.e. that their derivative is non-negative. Testing the non-negativity of a polynomial of several variables and of a degree equal to or greater than 4 is NP-hard (Murty and Kabadi, 1987). Parrilo (2003) proposed an approach based

on convex optimization techniques in order to find an approximate solution to this problem.

The approach proposed by Parrilo (2003) is based on the following theorem about non-negative polynomials.

Theorem 1 (Hilbert)**.** *A polynomial $F : \mathbb{R}^n \to \mathbb{R}$ is non-negative if it is possible to decompose it as a sum of squares (SOS):*

$$F(z) = \sum_s f_s^2(z) \qquad \text{with } z \in \mathbb{R}^n. \tag{7.1}$$

The condition given above is sufficient but not necessary, there exist non-negative polynomials that cannot be decomposed as a sum of squares (Blekherman, 2006). However, it has been proved by Hilbert that a non-negative polynomial of one variable is always a sum of squares (Parrilo, 2003). We give the proof here because it is remarkably simple and elegant.

Theorem 2 (Hilbert)**.** *A non-negative polynomial in one variable is always a SOS.*

Proof. Consider a polynomial of degree D, $p(x) = p_0 + p_1 x + p_2 x^2 + \ldots + p_D x^D$. Since $p(x)$ is non-negative, D must be even. The value of p_D should be greater than 0, otherwise $\lim_{x\to\infty} p(x) = -\infty$. As every polynomial of degree D admits D roots, one can write $p(x)$ as follows:

$$p(x) = p_D \prod_{i=1}^{m} (x - z_i)(x - \bar{z}_i) \prod_{j=1}^{n} (x - t_j)^{\alpha_j}$$

in which z_i and $\bar{z}_i$ for $i = \{1, \ldots, m\}$ are pairs of conjugate complex numbers and t_j for $j = \{1, \ldots, n\}$ are distinct real numbers where $D = 2m + \sum_{j=1}^{n} \alpha_j$. All the values of the exponents α_j are even. Indeed, consider a subset of k indices, $\{\Delta_1, \ldots, \Delta_k\}$, such that $\alpha_{\Delta_1}, \ldots, \alpha_{\Delta_k}$ are odd. Let τ be a permutation of these indices such that $t_{\tau(\Delta_1)} < \ldots < t_{\tau(\Delta_k)}$. For $x \in \left]t_{\tau(\Delta_{k-1})}, t_{\tau(\Delta_k)}\right[$, we would have $\prod_{j=1}^{n}(x - t_j)^{\alpha_j} < 0$, a contradiction. As all the values α_j are even, we can rewrite $p(x)$ as follows:

$$p(x) = \left(\sqrt{p_D} \prod_{i=1}^{l} (x - z_i)\right) \left(\sqrt{p_D} \prod_{i=1}^{l} (x - \bar{z}_i)\right)$$

in which some pairs $(z_i, \bar{z}_i)$ have no imaginary part. Let $\left(\sqrt{p_D} \prod_{i=1}^{l}(x - z_i)\right) = q(x) + ir(x)$ and $\left(\sqrt{p_D} \prod_{i=1}^{l}(x - \bar{z}_i)\right) = q(x) - ir(x)$ where i is the imaginary part of the complex number and $q(x), r(x)$, two polynomials with real coefficients.

Finally, the product of these two terms gives a sum of two squares: $p(x) = [q(x)]^2 + [r(x)]^2$. □

Let us consider the problem of determining a non-negative polynomial p of one variable x and degree D. We use the following canonical form to represent this polynomial:

$$\begin{aligned} p(x) &= p_0 + p_1 x + p_2 x^2 + \ldots + p_D x^D \\ &= \sum_{i=0}^{D} p_i \cdot x^i. \end{aligned} \tag{7.2}$$

To guarantee the non-negativity of this polynomial, we have to ensure that it can be represented as a sum of squares like in Equation (7.1). Note that a non-negative polynomial will always have an even degree since either the limit at positive or negative infinity of a polynomial of odd degree is negative. Let $d = \frac{D}{2}$, the polynomial $p(x)$ reads:

$$p(x) = \sum_s q_s^2(x) = \sum_s \left[\sum_{i=0}^{d} b_s^i x^i \right]^2.$$

Defining $b_s^{\mathsf{T}} = \begin{pmatrix} b_s^0 & b_s^1 & \ldots & b_s^d \end{pmatrix}$ and $\mathbf{x}^T = \begin{pmatrix} 1 & x & \ldots & x^d \end{pmatrix}$ (where T stands for the matrix transposition operation), we can express $p(x)$ as follows:

$$\begin{aligned} p(x) &= \sum_s \left(b_s^{\mathsf{T}} \mathbf{x} \right)^2 = \sum_s \mathbf{x}^{\mathsf{T}} b_s b_s^{\mathsf{T}} \mathbf{x} = \mathbf{x}^{\mathsf{T}} \left[\sum_s b_s b_s^{\mathsf{T}} \right] \mathbf{x} = \mathbf{x}^{\mathsf{T}} Q \mathbf{x} \\ &= \begin{pmatrix} 1 \\ x \\ \vdots \\ x^d \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} q_{0,0} & q_{0,1} & \cdots & q_{0,d} \\ q_{1,0} & q_{1,1} & \cdots & q_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ q_{d,0} & q_{d,1} & \cdots & q_{d,d} \end{pmatrix} \begin{pmatrix} 1 \\ x \\ \vdots \\ x^d \end{pmatrix}. \end{aligned}$$

Note that the matrix $Q = \sum_s b_s b_s^{\mathsf{T}}$ is symmetric and positive semidefinite (PSD), which we denote $Q \succeq 0$, since $\mathbf{x}^{\mathsf{T}} Q \mathbf{x} = \sum_s \left(b_s^{\mathsf{T}} \mathbf{x} \right)^2 \geq 0$ for all $\mathbf{x} \in \mathbb{R}^{d+1}$. Therefore, to ensure that $p(x)$ is non-negative, it is necessary to find a matrix Q of dimension $(d+1) \times (d+1)$ such that $p(x) = \mathbf{x}^{\mathsf{T}} Q \mathbf{x}$ and $Q \succeq 0$. It turns out that this condition is also sufficient. This follows from the following lemma.

Lemma 3. $Q \succeq 0 \iff \exists H : Q = H \cdot H^{\mathsf{T}}$.

The above decomposition is called the Cholesky decomposition of matrix Q (see Appendix E). To summarize, *a polynomial $p(x)$ in one variable is non-negative if and only if there exists $Q \succeq 0$ such that $p(x) = \mathbf{x}^{\mathsf{T}} Q \mathbf{x}$.*

The coefficients of the polynomial expressed in its canonical form (7.2) are obtained by summing the off-diagonal entries of the matrix Q, as follows:

$$\begin{cases} p_0 = q_{0,0}, \\ p_1 = q_{1,0} + q_{0,1}, \\ p_2 = q_{2,0} + q_{1,1} + q_{0,2}, \\ \vdots \\ p_{2d-1} = q_{d,d-1} + q_{d-1,d}, \\ p_{2d} = q_{d,d}. \end{cases}$$

We can express the value of the coefficients of the polynomial as follows:

$$p_i = \begin{cases} \sum_{g=0}^{i} q_{g,i-g} & i = \{0, \ldots, d\}, \\ \sum_{g=i-d}^{d} q_{g,i-g} & i = \{d, \ldots, 2d\}. \end{cases} \tag{7.3}$$

The value of p_d can be computed with both expressions. Finding a non-negative univariate polynomial consists in finding a semidefinite positive matrix Q. Summing the off-diagonal entries of this matrix allows to control the coefficients of the polynomial.

In some applications, it is not necessary to ensure the non-negativity of the polynomial on $\mathbb{R}$ but only in an interval $[v_1, v_2]$. If the non-negativity constraint has to be guaranteed only in a given interval $[v_1, v_2]$ for a polynomial $p(x)$, then the following theorem is useful.

Theorem 4 (Hilbert)**.** *A polynomial $p(x)$ in one variable x is non-negative in the interval $[v_1, v_2]$, if and only if $p(x) = (x - v_1) \cdot q(x) + (v_2 - x) \cdot r(x)$ where $q(x)$ and $r(x)$ are SOS.*

Given the above theorem, if we want to ensure the non-negativity of the polynomial $p(x)$ of degree D on the interval $[v_1, v_2]$, we have to find two matrices Q and R of size $d + 1$, with $d = \left\lfloor \frac{D}{2} \right\rfloor$ ($\lfloor \cdot \rfloor$ being the nearest integer down), that are PSD. We denote these matrices and their indices as follows:

$$Q = \begin{pmatrix} q_{0,0} & q_{0,1} & \cdots & q_{0,d} \\ q_{1,0} & q_{1,1} & \cdots & q_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ q_{d,0} & q_{d,1} & \cdots & q_{d,d} \end{pmatrix}, \quad R = \begin{pmatrix} r_{0,0} & r_{0,1} & \cdots & r_{0,d} \\ r_{1,0} & r_{1,1} & \cdots & r_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ r_{d,0} & r_{d,1} & \cdots & r_{d,d} \end{pmatrix}.$$

Since Q and R are two PSD matrices, the products $\mathbf{a_j}^\mathsf{T} Q \mathbf{a_j}$ and $\mathbf{a_j}^\mathsf{T} R \mathbf{a_j}$, with $\mathbf{a_j}^\mathsf{T} = \begin{pmatrix} 1 & a_j & \ldots & a_j^d \end{pmatrix}$, are always non-negative.

To obtain a polynomial $p(x)$ that is non-negative in the interval $[v_1, v_2]$, its coefficients have to be chosen such that:

$$\begin{cases} p_0 = v_2 \cdot r_{0,0} - v_1 \cdot q_{0,0}, \\ p_1 = q_{0,0} - r_{0,0} + v_2 \cdot (r_{1,0} + r_{0,1}) - v_1 \cdot (q_{1,0} - q_{0,1}), \\ p_2 = (q_{1,0} + q_{0,1}) - (r_{1,0} + r_{0,1}) + v_2 \cdot (r_{2,0} + r_{1,1} + r_{0,2}) \\ \qquad\quad -v_1 \cdot (q_{2,0} + q_{1,1} + q_{0,2}), \\ \vdots \\ p_{2d-1} = (q_{d,d-2} + q_{d-1,d-1} + q_{d-2,d}) - (r_{d,d-2} + r_{d-1,d-1} + r_{d-2,d}) \\ \qquad\quad +v_2 \cdot (r_{d,d-1} + r_{d-1,d}) - v_1 \cdot (q_{d,d-1} + q_{d-1,d}), \\ p_{2d} = (q_{d,d-1} + q_{d-1,d}) - (r_{d,d-1} + r_{d-1,d}) + v_2 \cdot r_{d,d} - v_1 \cdot q_{d,d}, \\ p_{2d+1} = q_{d,d} - r_{d,d}. \end{cases}$$

If the degree D of the polynomial $p(x)$ is even then the value of p_{2d+1} is equal to 0. The values p_i can be expressed in the following more compact form:

$$p_i = \begin{cases} v_2 \cdot r_{0,0} - v_1 \cdot q_{0,0} & i = 0, \\ \sum_{g=0}^{i-1}(q_{g,i-1-g} - r_{g,i-1-g}) & \\ \qquad + \sum_{g=0}^{i}(v_2 \cdot r_{g,i-g} - v_1 \cdot q_{g,i-g}) & i = \{1, \ldots, d\}, \\ \sum_{g=i-d-1}^{d}(q_{g,i-1-g} - r_{g,i-1-g}) & \\ \qquad + \sum_{g=i-d}^{d}(v_2 \cdot r_{g,i-g} - v_1 \cdot q_{g,i-g}) & i = \{d+1, \ldots, 2d\}, \\ q_{d,d} - r_{d,d} & i = 2d+1. \end{cases}$$

7.1.3 Semidefinite programming applied to UTA methods

In the perspective of building more natural marginal value functions, we use SDP to learn polynomial marginals instead of piecewise linear ones. SDP has become a standard tool in convex optimization, being a generalization of linear programming and second-order cone programming. It allows to optimize linear functions over an affine subspace of the set of PSD matrices; see, e.g., Vandenberghe and Boyd (1996) and the references therein.

There are two variants of the new UTA-poly method. Firstly, we describe the approach that consists in using polynomials that are overall monotone, i.e. monotone on the set of all real numbers. Then we describe the second approach considering polynomials that are monotone only on a given interval.

Enforcing monotonicity of the marginals on the set of real numbers

In the new proposed model, we define the value function on each criterion j as a polynomial of degree D_j:

$$u_j^*(a_j) = \sum_{i=0}^{D_j} p_{j,i} \cdot a_j^i. \tag{7.4}$$

To be compliant with the requirements of the theory of additive value functions, the polynomials used as marginals should be non-negative and monotone over the criteria domains. To ensure monotonicity, the derivative of the marginal value function has to be non-negative, hence we impose that the derivative of each value function is a sum of squares. The degree of the derivative is therefore even which implies that D_j is odd. This requirement reads:

$$\begin{aligned} u_j^{*\prime} &= p_{j,1} + 2p_{j,2} \cdot a_j + 3p_{j,3} \cdot a_j^2 + \ldots + D_j p_{j,D_j} \cdot a_j^{D_j - 1} \\ &= \mathbf{a_j}^\mathsf{T} Q_j \mathbf{a_j}, \end{aligned}$$

with Q_j a PSD matrix of dimension $(d_j+1) \times (d_j+1)$, $\mathbf{a_j}$ a vector of size (d_j+1) with $d_j = \frac{D_j - 1}{2}$:

$$Q_j = \begin{pmatrix} q_{j,0,0} & q_{j,0,1} & \cdots & q_{j,0,d_j} \\ q_{j,1,0} & q_{j,1,1} & \cdots & q_{j,1,d_j} \\ \vdots & \vdots & \ddots & \vdots \\ q_{j,d_j,0} & q_{j,d_j,1} & \cdots & q_{j,d_j,d_j} \end{pmatrix}, \quad \mathbf{a_j} = \begin{pmatrix} 1 \\ a_j \\ \vdots \\ a_j^{d_j} \end{pmatrix}.$$

By using SDP, we impose the matrix Q to be semidefinite positive and we set the following constraints on the $p_{j,i}$ values, for $i \geq 1$:

$$\begin{cases} p_{j,1} = q_{j,0,0}, \\ 2p_{j,2} = q_{j,1,0} + q_{j,0,1}, \\ 3p_{j,3} = q_{j,2,0} + q_{j,1,1} + q_{j,0,2}, \\ \vdots \\ (2d_j)p_{j,2d_j} = q_{j,d_j,d_j-1} + q_{j,d_j-1,d_j}, \\ (2d_j+1)p_{j,2d_j+1} = q_{j,d_j,d_j}. \end{cases}$$

In UTA-poly, the marginal value functions and monotonicity conditions on marginals given in Equation (2.23) and (2.27) are replaced by the following con-

straints:

$$\begin{cases} U(a) = \sum_{j=0}^{n} \sum_{i=0}^{D_j} p_{j,i} \cdot a_j^i & \forall a \in A, \\ Q_j \quad \text{PSD} & \forall j \in N, \\ (i+1)p_{j,i+1} = \sum_{g=0}^{i} q_{j,g,i-g} & i = \{0, \ldots, d_j\}, \forall j \in N, \\ (i+1)p_{j,i+1} = \sum_{g=i-d_j}^{d_j} q_{j,g,i-g} & i = \{d_j+1, \ldots, 2d_j\}, \forall j \in N. \end{cases} \tag{7.5}$$

The optimization program composed of the objective given in Equation (2.22) and the set of constraints given in Equations (2.23) and (7.5) can be solved using convex programming, more precisely, semidefinite programming (Parrilo, 2003). We refer to this new mathematical program as to UTA-poly. An explicit UTA-poly formulation for a simple problem involving 2 criteria and 3 alternatives is provided in Appendix D for illustrative purposes.

Enforcing monotonicity of the marginals on the criteria domains

Ensuring the monotonicity of each marginal on the domain of each criterion (instead of the whole real line) is sufficient to satisfy the requirements of the additive value function model. To do so, we use Theorem 4 and only impose the non-negativity of the marginal derivative on the domain $[\underline{a_j}, \overline{a_j}]$ of each criterion. This results in the following condition on the derivative $u_j^{*\prime}$ of the polynomial u_j^*, for all j:

$$\begin{aligned} u_j^{*\prime} &= p_{j,1} + 2p_{j,2} \cdot a_j + 3p_{j,3} \cdot a_j^2 + \ldots + D_j p_{j,D_j} \cdot a_j^{D_j - 1} \\ &= (a_j - \underline{a_j}) \mathbf{a_j}^{\mathsf{T}} Q_j \mathbf{a_j} + (\overline{a_j} - a_j) \mathbf{a_j}^{\mathsf{T}} R_j \mathbf{a_j}. \end{aligned}$$

In the above equation, Q_j and R_j are two PSD matrices of size $(d_j+1) \times (d_j+1)$ and $\mathbf{a_j}$ a vector of size $d_j + 1$, where $d_j = \left\lfloor \frac{D_j - 1}{2} \right\rfloor$:

$$Q_j = \begin{pmatrix} q_{j,0,0} & q_{j,0,1} & \cdots & q_{j,0,d_j} \\ q_{j,1,0} & q_{j,1,1} & \cdots & q_{j,1,d_j} \\ \vdots & \vdots & \ddots & \vdots \\ q_{j,d_j,0} & q_{j,d_j,1} & \cdots & q_{j,d_j,d_j} \end{pmatrix}, R_j = \begin{pmatrix} r_{j,0,0} & r_{j,0,1} & \cdots & r_{j,0,d_j} \\ r_{j,1,0} & r_{j,1,1} & \cdots & r_{j,1,d_j} \\ \vdots & \vdots & \ddots & \vdots \\ r_{j,d_j,0} & r_{j,d_j,1} & \cdots & r_{j,d_j,d_j} \end{pmatrix}.$$

The value $p_{j,i}$ for $i \geq 1$ are obtained as follows:

$$\begin{cases} p_{j,1} = \overline{a_j} \cdot r_{j,0,0} - \underline{a_j} \cdot q_{j,0,0}, \\ 2p_{j,2} = q_{j,0,0} - r_{j,0,0} + \overline{a_j} \cdot (r_{j,1,0} + r_{j,0,1}) - \underline{a_j} \cdot (q_{j,1,0} + q_{j,0,1}), \\ 3p_{j,3} = (q_{j,1,0} + q_{j,0,1}) - (r_{j,1,0} + r_{j,0,1}) + \overline{a_j} \cdot (r_{j,2,0} + r_{j,1,1} + r_{j,0,2}) \\ \qquad\qquad -\underline{a_j} \cdot (q_{j,2,0} + q_{j,1,1} + q_{j,0,2}) \\ \vdots \\ (2d_j)p_{j,2d_j} = (q_{j,d_j,d_j-2} + q_{j,d_j-1,d_j-1} + q_{j,d_j-2,d_j}) \\ \qquad\qquad -(r_{j,d_j,d_j-2} + r_{j,d_j-1,d_j-1} + r_{j,d_j-2,d_j}) \\ \qquad\qquad +\overline{a_j} \cdot (r_{j,d_j,d_j-1} + r_{d_j-1,d_j}) - \underline{a_j} \cdot (q_{j,d_j,d_j-1} + q_{j,d_j-1,d_j}), \\ (2d_j+1)p_{j,2d_j+1} = (q_{j,d_j,d_j-1} + q_{j,d_j-1,d_j}) - (r_{j,d_j,d_j-1} + r_{j,d_j-1,d_j}) \\ \qquad\qquad +\overline{a_j} \cdot r_{j,d_j,d_j} - \underline{a_j} \cdot q_{j,d_j,d_j}, \\ (2d_j+2)p_{j,2d_j+2} = q_{j,d_j,d_j} - r_{j,d_j,d_j}. \end{cases}$$

If the degree D_j is odd, then we have $p_{j,2d_j+2} = 0$ since $2d_j + 2 > D_j$.

In convex programming, in order to have polynomial marginals that are monotone on an interval, the monotonicity constraints in UTA have to be replaced by the following ones:

$$\begin{cases} U(a) = \sum_{j=0}^{n} \sum_{i=0}^{D_j} p_{j,i} a_j^i & \forall a \in A, \\ Q_j, R_j \quad \text{PSD} & \forall j \in N, \\ p_{j,1} = \overline{a_j} \cdot r_{j,0,0} - \underline{a_j} \cdot q_{j,0,0}, & \\ (i+1)p_{j,i+1} = \sum_{g=0}^{i-1}(q_{j,g,i-g} - r_{j,g,i-g}) & \\ \qquad + \sum_{g=0}^{i}(\overline{a_j} \cdot r_{j,g,i-1-g} - \underline{a_j} \cdot q_{j,g,i-1-g}) & \\ \qquad\qquad i = \{0, \ldots, d_j\}, & \forall j \in N, \\ (i+1)p_{j,i+1} = \sum_{g=i-d_j-1}^{d_j}(q_{j,g,i-1-g} - r_{j,g,i-1-g}) & \\ \qquad + \sum_{g=i-d_j}^{d_j}(\overline{a_j} \cdot r_{j,g,i-g} - \underline{a_j} q_{j,g,i-g}) & \\ \qquad\qquad i = \{d_j + 1, \ldots, 2d_j\}, & \forall j \in N, \\ (2d_j+2)p_{j,2d_j+2} = q_{d_j,d_j} - r_{d_j,d_j} & \forall j \in N. \end{cases} \tag{7.6}$$

The optimization program composed of the objective given in Equation (2.22) and the set of constraints given in Equation (2.23) and (7.5) can be solved using semidefinite programming.

Similarly, it is possible to use monotonicity constraints given in Equation (7.5) in conjunction with Equation (2.26) and (2.27) in order to learn an additive value function sorting model. We call this inference program UTADIS-poly.

7.2 UTA-splines: additive value functions with splines marginals

In this section we describe a variant of UTA-poly which consists of using several polynomials for each value function. We first recall some theory about splines. Then we describe the new method called UTA-splines.

7.2.1 Splines

We recall here the definition of a spline. We detail the ones that are the most commonly used.

Definition

A spline of degree D_s is a function Sp that interpolates the set of points (x_i, y_i) for $i = 0, \ldots, q$, with $x_0 < x_1 < \ldots < x_q$ such that:

- $Sp(x_i) = y_i$ for $i = 0, \ldots, q$;
- Sp is a set of polynomials of degree equal to or smaller than D_s, on each interval $[x_i, x_{i+1}[$ (at least one of the polynomials has a degree equal to D_s);
- the derivative of Sp are continuous up to a given degree D_c on $[x_0, x_q]$.

The degree of a spline corresponds to its highest polynomial degree. If all the polynomials have the same degree, the spline is said to be uniform.

The continuity of the spline at the connection points is ensured up to a given derivative. Usually, the continuity of the spline is guaranteed up to the second derivative ($D_c = 2$). It ensures the continuity of the slope and concavity at the connection points.

Cubic splines

The most common uniform splines are the ones of degree 3 ($D_s = 3$), also called cubic splines. A cubic spline consists of a set of third degree polynomials which are continuous up to the second derivative at their connection points.

We denote by s_i the i^{th} polynomial of the spline going from connection point x_i to connection point x_{i+1}. Formally, each polynomial s_i of the spline has the following form:

$$s_i(x) = s_{i,0} + s_{i,1}x + s_{i,2}x^2 + s_{i,3}x^3.$$

The use of cubic splines requires the determination of four parameters: $s_{i,0}$, $s_{i,1}$, $s_{i,2}$ and $s_{i,3}$. If the spline interpolates q points, there are overall $4 \cdot (q-1)$ parameters to determine.

Imposing the equality up to the second derivative at the connection points amounts to enforce the following constraints:

$$\left\{\begin{array}{rcll} s_i(x_i) & = & y_i & i=\{0,\ldots,q-1\}, \\ s_i(x_{i+1}) & = & y_{i+1} & i=\{0,\ldots,q-1\}, \\ s_i'(x_{i+1}) & = & s_{i+1}'(x_{i+1}) & i=\{0,\ldots,q-2\}, \\ s_i''(x_{i+1}) & = & s_{i+1}''(x_{i+1}) & i=\{0,\ldots,q-2\}. \end{array}\right. \tag{7.7}$$

Since there are $4q-2$ constraints and $4q$ parameters, two degrees of freedom remain. They can be set in different ways. For instance, one can impose $s_0''(x_0)=0$ and $s_{q-1}''(x_q)=0$. This corresponds to imposing zero curvature at both endpoints of the spline.

7.2.2 UTA-splines: using splines as marginals

We give some detail on how using splines to model marginal value functions of an additive value function model. We formulate a semidefinite program that learns the parameters of such a model.

Overview

Using splines continuous up to either the first or the second derivative instead of piecewise linear functions for the marginal value functions aims at obtaining more natural functions around the breakpoints.

With UTA-poly, the flexibility of the model is improved by using polynomials of higher degrees. In order to further improve the flexibility of the model, we propose now to hybridize the original UTA method which splits the criterion domain into k equal parts with the UTA-poly approach which uses polynomials to model the marginal value functions. We call this new disaggregation procedures UTA-splines. The UTA-splines method combines the use of piecewise functions for the marginals (as in UTA) and the use polynomials (as in UTA-poly) for each piece of the function.

Compared to UTA, in UTA-splines the continuity of the marginal can be ensured up to the any derivative at the connection points. It enables to obtain more natural marginals which have a continuous curvature.

Description of UTA-splines

In UTA-splines, we model marginals as uniform splines of degree D_s. Formally the marginal of criterion j reads:

$$u_j^*(a_j) = Sp_j^{D_s,k}(a_j)$$

where $Sp_j^{D_s}$ denotes a uniform spline of degree D_s composed of k pieces. Each piece of the spline $Sp_j^{D_s,k}(a_j)$ is a polynomial of degree D_s denoted by $s_{j,l}(a_j)$, $l = \{1, \ldots, k\}$. Formally it reads:

$$s_{j,l}(a_j) = s_{j,l,0} + s_{j,l,1}a_j + s_{j,l,2}a_j^2 + \ldots + s_{j,l,D_s}a_j^{D_s}.$$

The pairs (g_j^{l-1}, u_j^{l-1}) and (g_j^l, u_j^l) denote respectively the coordinates of the initial and final points of the piece l of the spline. The points g_j^l for $l = 1$ to $k-1$ partition the criterion domain $[\underline{a_j}, \overline{a_j}]$ in subintervals. We set $\underline{a_j} = g_j^0$ and $\overline{a_j} = g_j^k$. Hence the piece $s_{j,l}$ of the spline is defined on the interval $[g_j^{l-1}, g_j^l]$. The spline $s_{j,l}$ takes the value u_j^{l-1} (resp. u_j^l) on g_j^{l-1} (resp. g_j^l). The continuity of the spline at the connection points is ensured by imposing the two following constraints:

$$\begin{cases} s_{j,l}(g_j^{l-1}) = u_j^{l-1} & l = \{1, \ldots, k\}, \\ s_{j,l}(g_j^l) = u_j^l & l = \{1, \ldots, k\}. \end{cases}$$

Usually, the continuity of the marginals is ensured up to the second derivative so that slope and concavity at the connection points remain continuous. To ensure the continuity of the first derivative, the following constraints are added:

$$s'_{j,l}(g_j^l) = s'_{j,l+1}(g_j^l) \quad l = \{1, \ldots, k-1\}.$$

Similarly, the following constraints are added to ensure the continuity of the second derivative:

$$s''_{j,l}(g_j^l) = s''_{j,l+1}(g_j^l) \quad l = \{1, \ldots, k-1\}.$$

Of course, it is possible to ensure the continuity of the second derivative only if the marginal polynomials have a degree equal to or higher than 3. More generally, it is possible to ensure the continuity of the polynomials up to the i^{th} derivative only if the polynomials have a degree equal to or higher than $i+1$.

As in UTA-poly, the main difficulty in UTA-splines is to find polynomials which ensure the monotonicity of the marginals. To achieve this, we use the results set out in Section 7.1.2. Recall that the non-negativity of an univariate polynomial is ensured if it can be expressed as a sum of squares. The monotonicity of the marginals is therefore ensured by imposing the non-negativity of their derivatives on an interval. Formally, for the piece l of the spline associated to criterion j, it reads:

$$s'_{j,l}(a_j) = s_{j,l,1} + 2s_{j,l,2}a_j + \ldots + D_s s_{j,l,D_s} a_j^{D_s-1} \geq 0.$$

We impose $s'_{j,l}(a_j)$ to be a sum of two SOS as specified in Theorem 4. Formally it reads:

$$s'_{j,l}(a_j) = (x - g_j^{l-1}) \cdot q_{j,l}(a_j) + (g_j^l - x) \cdot r_{j,l}(a_j),$$

with $q_{j,l}(a_j)$ and $r_{j,l}(a_j)$ two polynomials that can be expressed as sums of squares.

Using SDP, we impose two square matrices $Q_{j,l}$ and $R_{j,l}$ of size $d = \left\lceil \frac{D_s-1}{2} \right\rceil + 1$ to be PSD. Hence, $q_{j,l}(a_j) = \mathbf{a_j}^\mathsf{T} Q_{j,l} \mathbf{a_j}$ and $r_{j,l}(a_j) = \mathbf{a_j}^\mathsf{T} R_{j,l} \mathbf{a_j}$, with $\mathbf{a_j}^\mathsf{T} = \begin{pmatrix} 1 & a_j & \dots & a_j^d \end{pmatrix}$, are two non-negative polynomials.

The value of the polynomial coefficients $s_{j,l,0}, \dots, s_{j,l,D_s}$ are obtained by combining the off-diagonal terms of the matrices.

As for UTA-poly, it is possible to proceed similarly to learn an additive value function sorting model by using the monotonicity constraints described in this section in conjunction with Equations (2.26) and (2.27). We call such an inference procedure UTADIS-splines.

Link between UTA-splines, UTA-poly and UTA

We note that UTA-splines is a generalization of UTA. Indeed, UTA is a particular case of UTA-splines in which splines of the first degree are used.

A similar link exists between UTA-splines and UTA-poly. Indeed, if UTA-splines is used to learn marginals composed of exactly one piece then it is equivalent to the UTA-poly formulation.

7.3 Illustrative example

In this section, we illustrate UTA-poly and UTA-splines on an small instance of a ranking problem. In the first subsection we briefly present the context of the problem. Then we infer the parameters of UTA-poly models and compare the marginals obtained with UTA-poly to the original ones. Finally we perform the same experiment with UTA-splines. To formulate and solve the SDP we used CVX, a Matlab software for *disciplined convex programming* (Grant and Boyd, 2014). The source code of UTA-poly and UTA-splines is available at the following address: `http://olivier.sobrie.be`.

7.3.1 Context of the problem

A family plans to spend a one week holiday in France. They use a search engine which returns a list of 1000 possible accommodations. To avoid reviewing the whole list and save time, the family calls a MCDA analyst. The first task of the analyst consists of determining which criteria matter to the family. They identify the following three criteria:

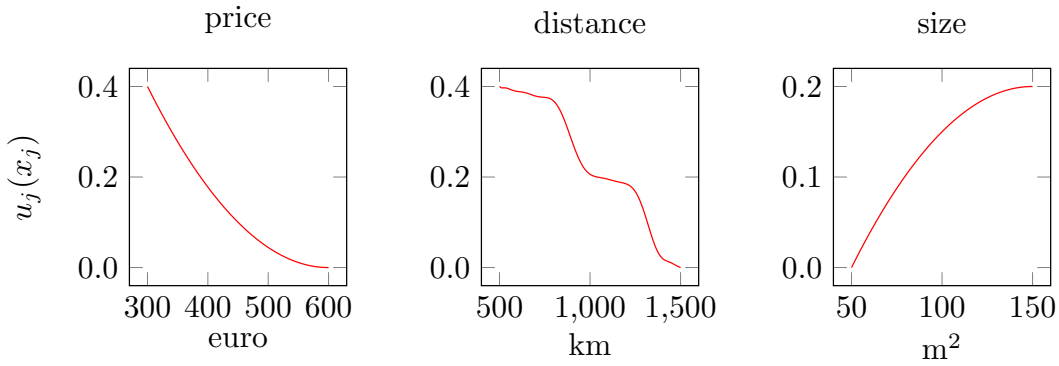


Figure 7.2: True marginal value functions modeling the family's preferences.

- Price: the price of the renting in euros, which should be minimized;
- Distance: the distance from home in kilometers, which should be minimized;
- Size: the size of the accommodation in square meters, which should be maximized.

The family cannot evaluate the importance of the criteria and doesn't want to enter into a formal elicitation procedure. In contrast, they are ready to make some overall statements that could be used by a model learning method.

Let us assume that the preferences of the family can be represented by an additive value function and that the marginals are displayed in Figure 7.2. These functions are polynomials of degree 2 (u_1 and u_3) and 15 (u_2).

7.3.2 UTA-poly

In order to learn the marginals given in Figure 7.2, the family ranks a subset of 50 alternatives chosen randomly in the list according to the unveiled marginal functions displayed in Figure 7.2.

The 49 informative pairwise comparisons are used to learn, using UTA-poly, an additive value function model with polynomials of degree one to ten. The inferred value function yields a ranking of the 50 alternatives. Hence, we can observe the similarity of the initial and inferred rankings. The evolution of the Spearman distance and Kendall Tau of these rankings is given in Figure 7.3. We observe that increasing the degree of the polynomial increases the accuracy of the model. Indeed, the values of the Spearman distance and Kendall Tau grow as a function of the degree of the marginals.

In a second step, the analyst asks the family to include 50 other alternatives in the ranking. The analyst provides a set of 99 pairwise comparisons to UTA-poly.

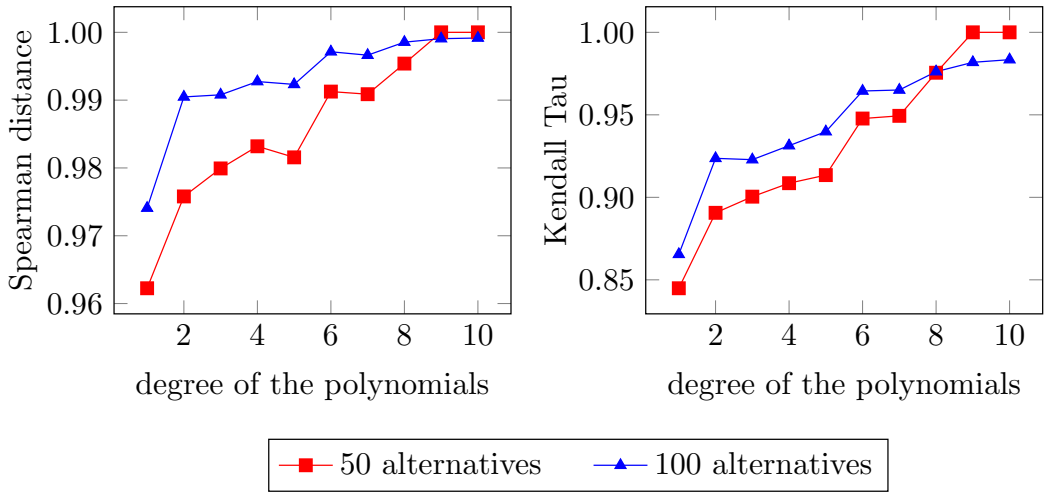


Figure 7.3: Evolution of the Spearman distance and Kendall Tau of the learning set as a function of the degree of the marginal polynomials for learning sets composed of 50 and 100 examples.

As in the first step, polynomials of degree one to ten are learned. We observe in Figure 7.3 that the accuracy of the model is improved with more pairwise comparisons when the marginals have a small degree (smaller than 8). With more examples we see that the Spearman distance and Kendall Tau are slightly better when marginals degree is small and slightly worse when marginals degree is superior to 9.

For illustrative purpose we show in Figure 7.4 the marginals learned on the basis of 100 examples with polynomials of degree 2, 6 and 10. We see that the marginals u_1 and u_3 are well approximated with polynomials of degree 2 to 10. The major difference is observed for u_2. Using a polynomial of degree 2 approximates roughly the curve. The two "steps" of u_2 cannot be better approximated by a polynomial of the second degree since there is no inflexion point with such a polynomial. The real marginal has at least two inflexions where the steps are located. With a polynomial of degree 6 we see that the approximation of this curve is improved but it does not perfectly fit the real marginal. Indeed the slope is less steep between the inflexion points. With a polynomial of degree 10 the learned marginal almost perfectly fits the real marginal. The inflexion of the curve happens at the same places and the slopes are similar.

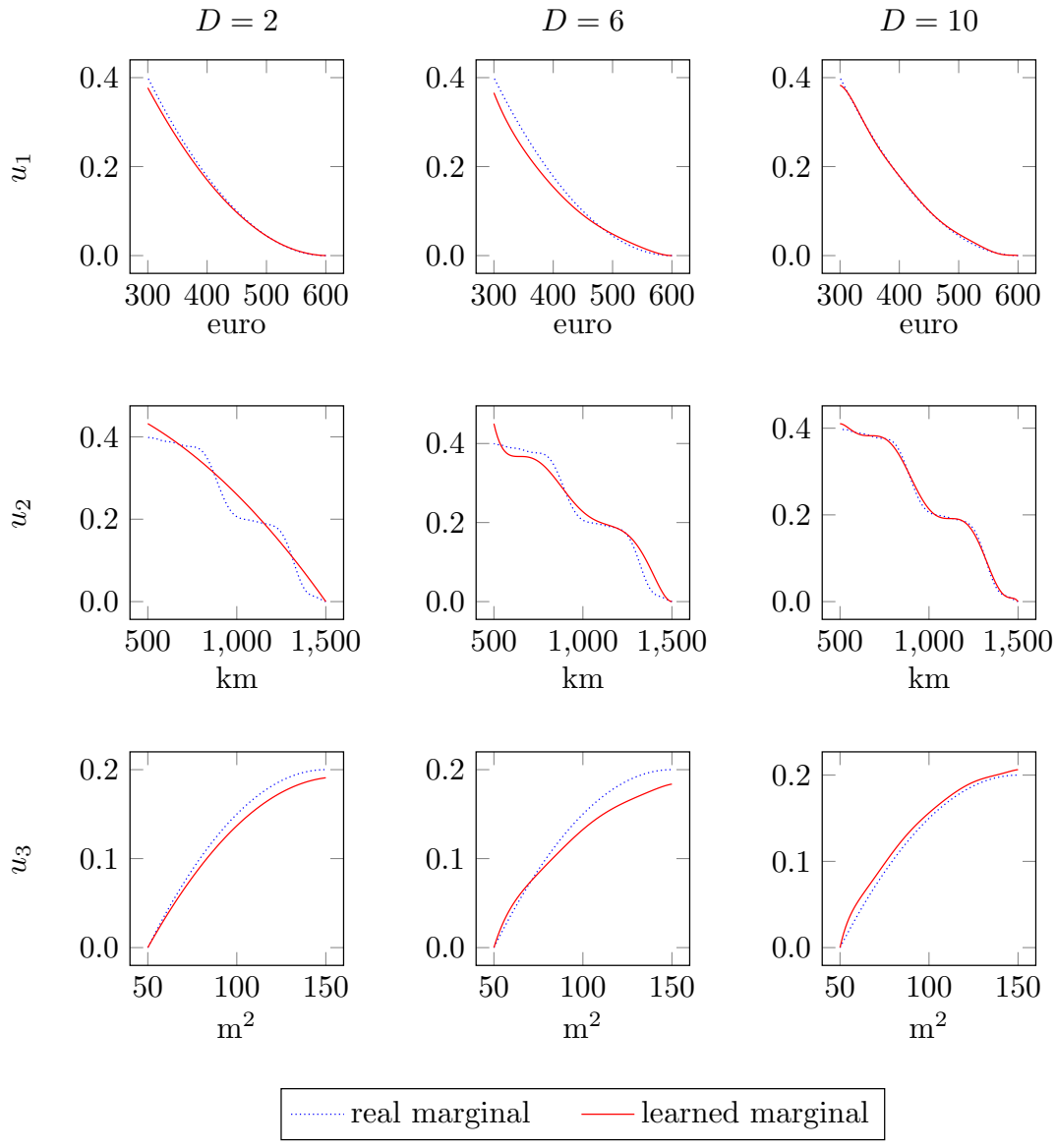


Figure 7.4: Value functions learned by UTA-poly on the basis of a learning set composed of 100 examples with polynomials of degree $D = 2, 6$ and 10.

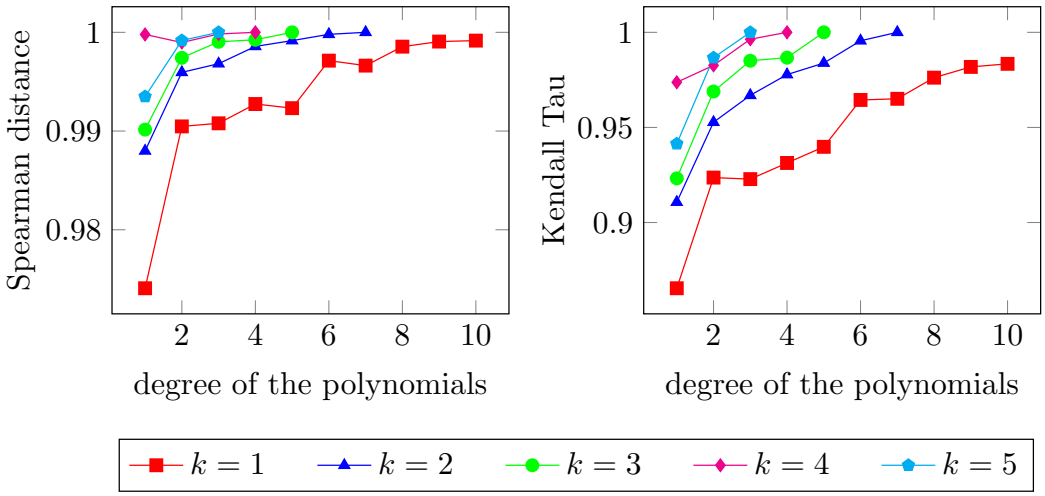


Figure 7.5: Evolution of the Spearman distance and Kendall Tau of the learning set as a function of the degree of the marginal polynomials for learning sets composed of 100 examples with 1 to 5 polynomials per marginal.

7.3.3 UTA-splines

As for UTA-poly, we perform some experimentations with UTA-splines on the application described above. We vary the number of pieces and the polynomial degrees of UTA-splines and observe the variation in accuracy. We also study the impact of the continuity degree on the splines.

Figure 7.5 shows the evolution of the average Spearman distance and Kendall Tau on the learning set. We note that increasing the number of pieces usually has a positive influence on the way UTA-splines succeeds in restoring the original ranking. UTA-splines is able to restore the original ranking with smaller polynomial degrees when the number of pieces increases. However it is not always the case. For instance, when using polynomials of degree 1, a UTA model composed of 4 pieces performs better than one using 5 pieces. With polynomials of degree greater than 1, UTA-splines always performs better when the number of pieces is larger.

For illustrative purpose, we show in Figure 7.6 the marginals obtained with splines of degree $D = 1$ to 3. The continuity of the splines at the breakpoints (D_c) is enforced up to $D - 1$. With polynomials of degree 3, we observe that the learned marginals tightly fit the real marginals.

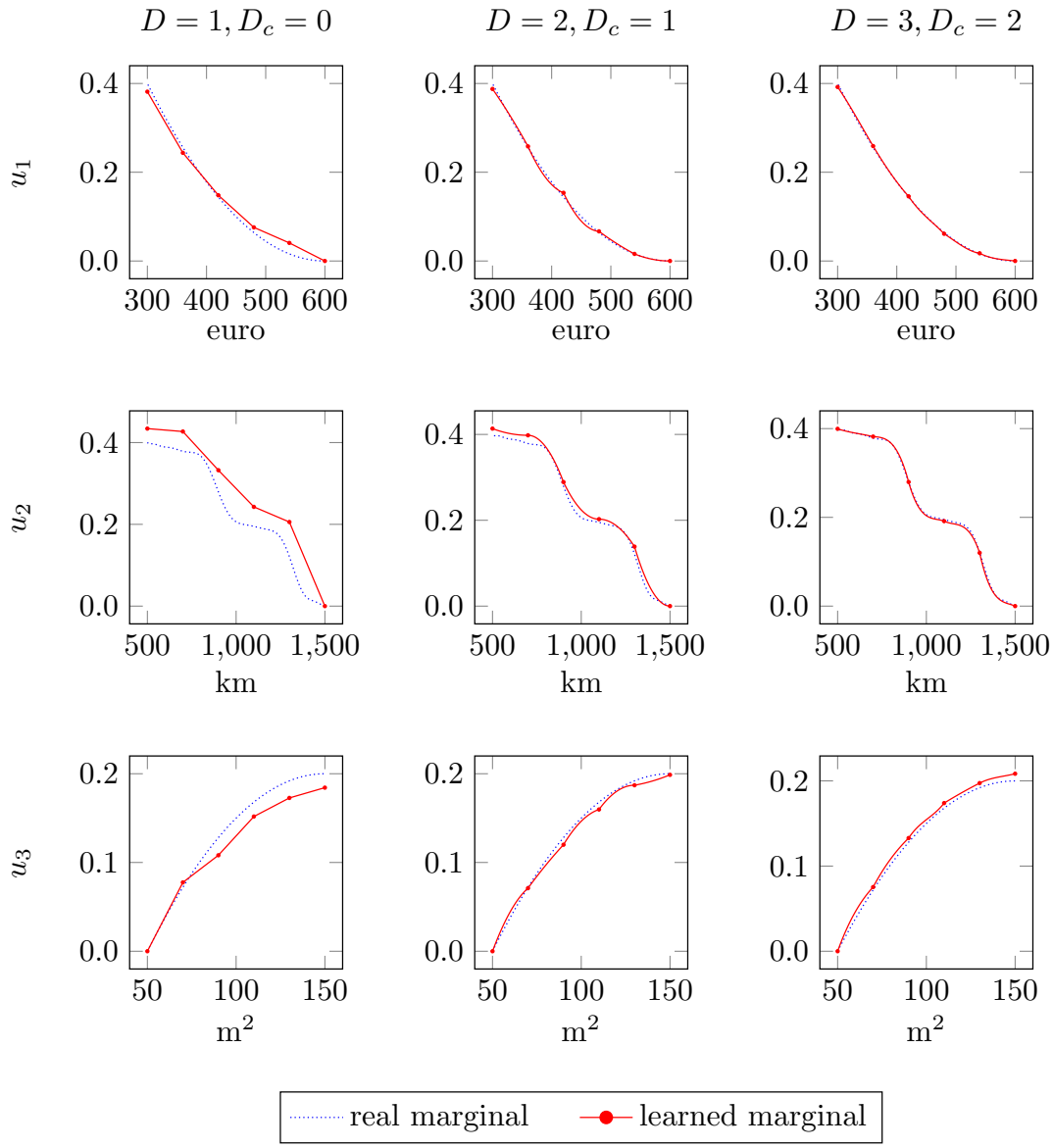


Figure 7.6: Value functions learned by UTA-splines on the basis of a learning set composed of 100 examples with polynomials of degree $D = 1$ to 3 and marginals composed of 5 polynomials ($k = 5$). The continuity of the spline (D_c) is enforced up to $D - 1$.

7.4 Experiments with artificial data sets

So as to understand the behavior of UTA-poly and UTA-splines, we performed experiments with artificial data sets. These experiments aim at studying the ability of the methods to retrieve a ranking from a set of pairwise comparisons and the computing time. In the experiments, we vary different parameters of UTA-poly and UTA-splines: degree of the polynomials (D), number of pieces (k), the continuity at breakpoints (D_c) and the number of alternatives in the learning set (m). As in the previous Section, we formulate and solve the SDP we used CVX, a Matlab software for *disciplined convex programming* (see Grant and Boyd, 2014).

7.4.1 Experimental setup

Our experimental strategy is the following. We start from an hypothetical additive value model denoted M, and generate a set of alternatives (called learning set). Then we simulate the behavior of a DM ranking these alternatives, while having the model M in mind. Hence, we build a ranking on the learning set.

We compute an additive value model using UTA-poly and UTA-splines compatible with the ranking of the learning set. We then compare the inferred models with the model M. To do so, we randomly generate another set of alternatives (test set), and we compute the ranking of this test set obtained by the model M and by the inferred model. We then compute the Spearman distance (Spearman, 1904) and the Kendall Tau (Kendall, 1938) to evaluate how close the inferred rankings are to the original ones.

We considered 8 different models M, chosen to represent a wide variety of value functions (structure and forms of the marginals). Four of these models are composed of 3 criteria (Figure 7.7), while the four others involve 5 criteria (Figure 7.8). As shown in Figures 7.7 and 7.8, the marginals are of different types: piecewise linear functions, sigmoids, exponentials and polynomials of degree 2, 3 and 15.

For a given model M and a seed s, the experimental procedure is the following:

1. The random generator is initialized with the seed s.

2. A set of m performances vectors (alternatives) is generated. It constitutes the learning set A^*. Each component a_j of a performances vector $a = (a_1, a_2, \ldots, a_n) \in A^*$ is generated by drawing n random numbers uniformly in $[0, 1]$.

3. The score $U(a)$ is computed for each vector of performances $a \in A^*$ using the value model M. A pre-order on these alternatives is derived from their

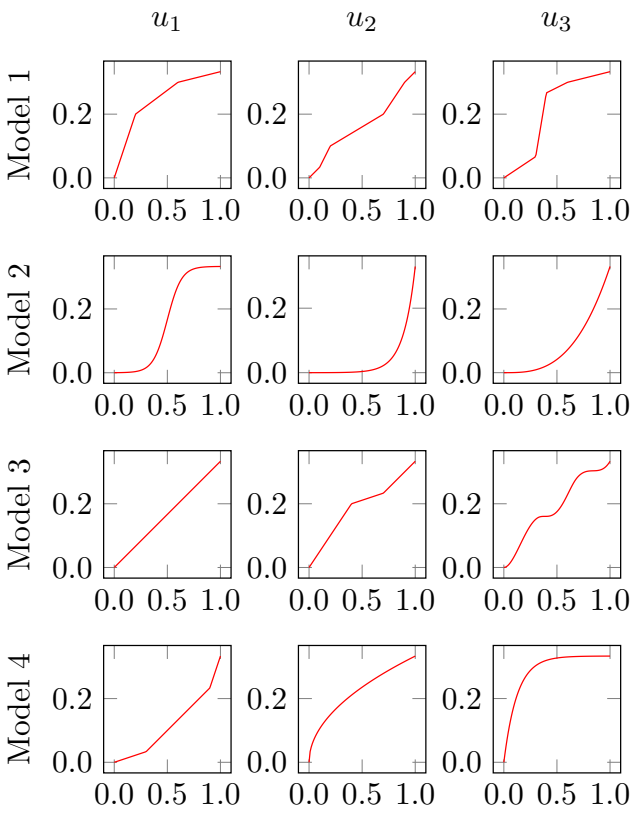


Figure 7.7: Four additive value function models composed of 3 criteria.

scores. Given a ranking π^* of the alternatives in A^*, we denote by π^*_i the alternative ranked in the i^{th} position. We have $\pi^*_1 \succcurlyeq \pi^*_2 \succcurlyeq \ldots \succcurlyeq \pi^*_{m-1}$.

4. A list of $m-1$ pairwise comparisons is induced from the complete ranking π^*. It is done by comparing each pair of consecutive alternatives in the ranking. In a ranking π^*, it consists in comparing π^*_i to π^*_{i+1}, either by an indifference ($\pi^*_i \sim \pi^*_{i+1}$) or a preference ($\pi^*_i \succ \pi^*_{i+1}$). We denote by $\mathcal{P}^*$ the set containing the pairs of alternatives (a, b) such that $a \succ b$. $\mathcal{I}^*$ denotes the set containing the pairs (a, b) such that $a \sim b$.

5. The sets A^*, $\mathcal{P}^*$ and $\mathcal{I}^*$ are given as input to UTA-splines/UTA-poly. The algorithm learns an additive value function model M' in which the marginals are composed of k polynomials of degree D. The breakpoints of the polynomials are equally spaced on the criterion domain. The continuity is guaranteed up to the D_c^{th} derivative at the breakpoints.

6. A test set of 1000 alternatives A is generated similarly as for the learning

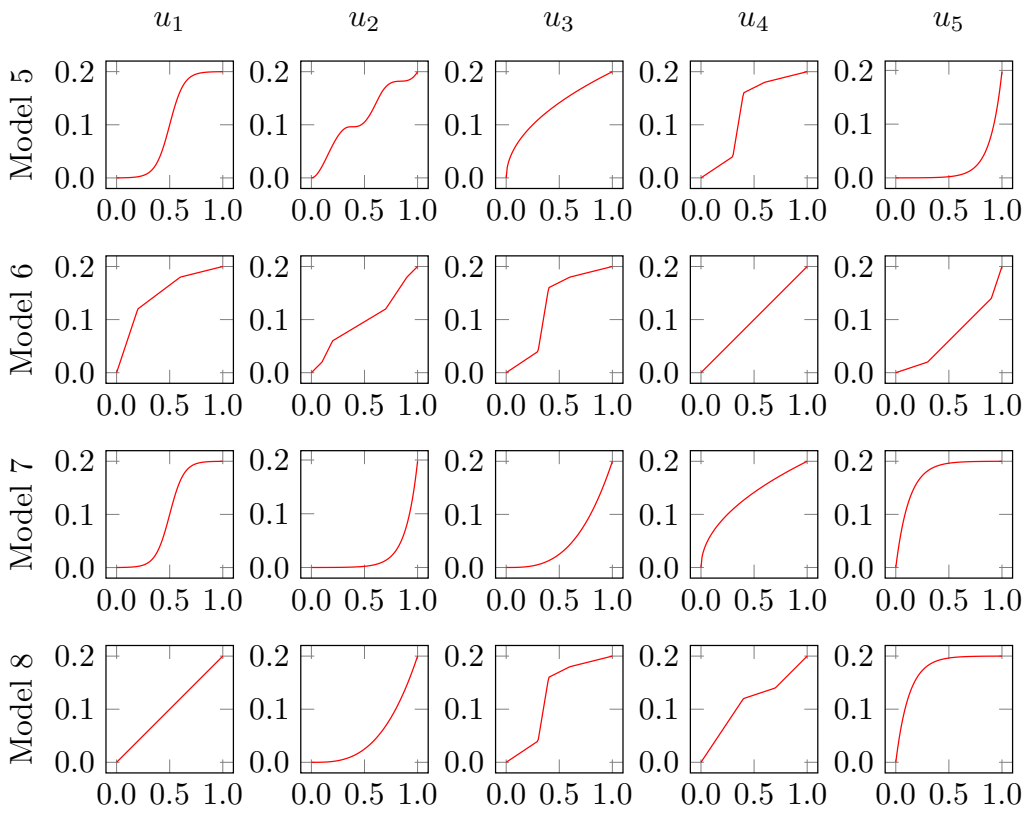


Figure 7.8: Four additive value function models composed of 5 criteria.

set. The alternatives in A are ranked with models M and M'. The obtained ranking π and $\hat{\pi}$ are then compared by computing the Spearman distance $SD(\pi, \hat{\pi})$ (see Spearman, 1904) and the Kendall Tau $KT(\pi, \hat{\pi})$ (see Kendall, 1938).

7.4.2 Model retrieval

We tested UTA-poly and UTA-splines with the models shown in Figures 7.7 and 7.8. Results provided in this Section are mean values over the 8 different models tested. We varied the degree of the polynomials (D), the number of pieces (k), the continuity at the breakpoints (D_c). We varied the size of the learning set (m) between 10 and 100 alternatives. The test set was composed of 1000 alternatives. For each setting, we ran the test procedure described above with 10 random seeds.

This experiment shows how the number of comparisons impacts the ability to elicit the parameters of a model M composed of n criteria. The experiment also shows the impact of the number of pieces per marginal and of the degree of the

polynomial.

UTA-poly

The first test consists in testing UTA-poly with only one piece per marginal ($k = 1$). We show in Figure 7.9 the average Spearman distance and Kendall Tau of the test set of the models composed of 3 criteria when the degree of the learned marginals (D) varies from 1 (which corresponds to a weighted sum) to 4. The values of the Spearman distance and Kendall Tau increase as a function of the number of alternatives in the learning set. For the same number of examples in the learning set, the quality of the ranking is improved as the degree of the polynomial increases. We observe the same behavior with models involving 5 criteria (Figure 7.10). Detailed results per model are available in Appendix F.

UTA-splines

In the second test, we varied the number of pieces per marginals (k) from 1 to 5 and used polynomials of degree 3. The continuity at the breakpoints is ensured up to the second derivative. Figure 7.11 shows the average Spearman distance and Kendall Tau of the test set for the models composed of 3 criteria. We observe that increasing the number of pieces helps to increase the accuracy of the model. With models composed of 5 criteria (see Figure 7.12), we observe the same behavior. It depicts a general trend for the model presented in Figures 7.7 and 7.8. Nevertheless one has to be cautious to overfitting effects when the number of pieces increases and to the position of the breakpoints. Indeed increasing the number of pieces increases the number of parameters of the model and its flexibility which may lead to overfitting. In Appendix F we present the detailed results for each model of Figure 7.7 and 7.8.

7.4.3 Computing time

The computing time strongly depends on the number of constraints and variables that are involved. The number of constraints and variables are expressed by the following equations:

$$\#\text{constraints} = m + n + 2nk + nkD + (1 + D_c)n(k-1),$$
$$\#\text{variables} = nk(D+1) + 2nk\left\lceil \frac{D}{2} \right\rceil^2 + 2m.$$

We give in Table 7.3 the number of constraints and variables for different problem sizes.

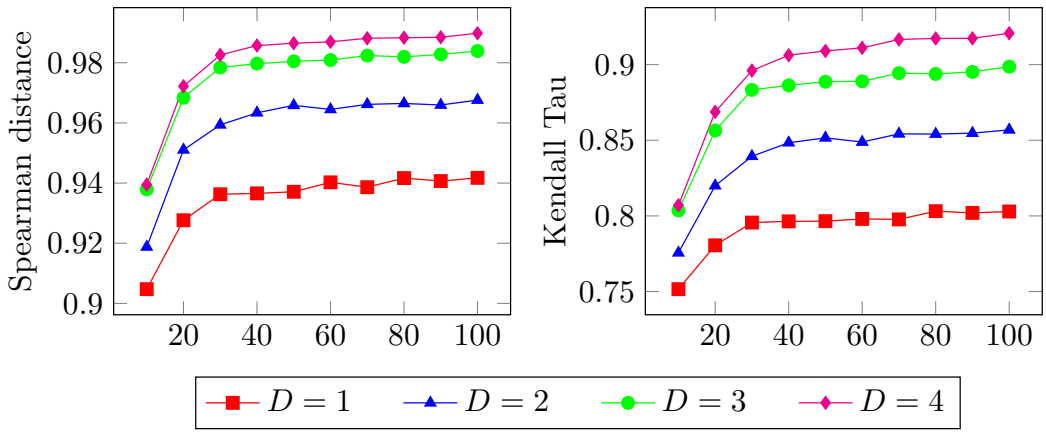


Figure 7.9: Average Spearman distance and Kendall Tau of the test set with the models involving 3 criteria learned by UTA-poly when the degree of the marginals vary between 1 and 4.

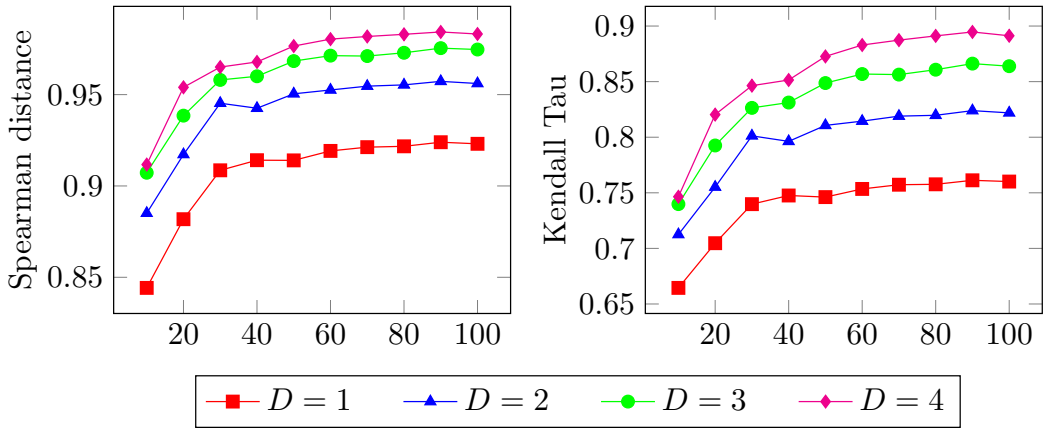


Figure 7.10: Average Spearman distance and Kendall Tau of the test set with the models involving 5 criteria learned by UTA-poly when the degree of the marginals vary between 1 and 4.

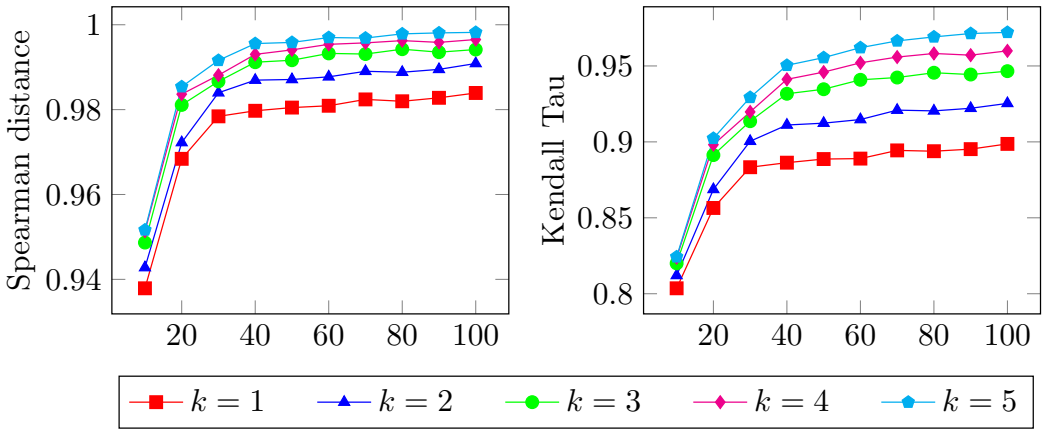


Figure 7.11: Average Spearman distance and Kendall Tau of the test set with the models involving 3 criteria learned by UTA-splines with marginals composed of polynomials of the third degree. The continuity at the breakpoints is ensured up to the second derivative.

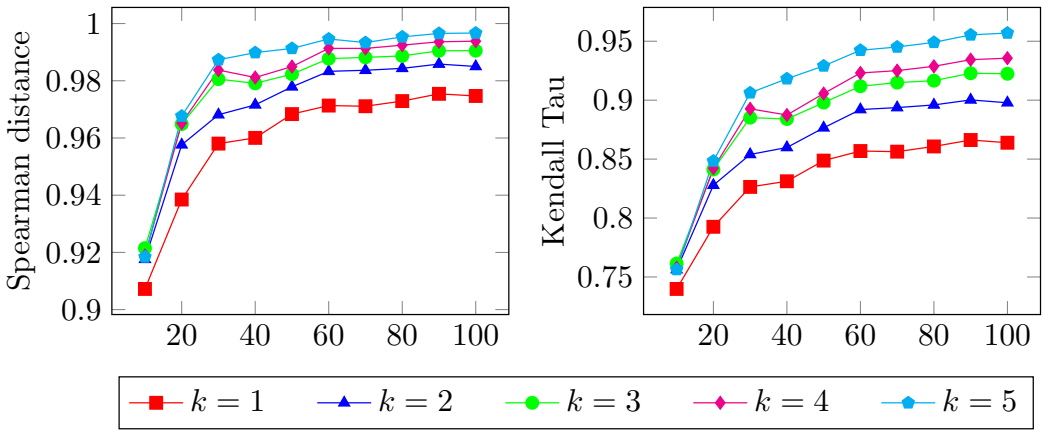


Figure 7.12: Average Spearman distance and Kendall Tau of the test set with the models involving 5 criteria learned by UTA-splines with marginals composed of polynomials of the third degree. The continuity at the breakpoints is ensured up to the second derivative.

Table 7.3: Number of constraints and variables for different problem sizes and average computing time with standard deviation.

m	n	k	D	D_c	#const.	#var.	computing time (sec.)
10	3	1	1	0	22	32	0.48 ± 0.15
10	3	5	1	0	70	80	1.02 ± 0.34
10	3	1	4	0	31	59	0.86 ± 0.19
10	3	5	4	2	139	215	1.96 ± 0.29
10	5	5	4	2	225	345	2.99 ± 0.36
100	3	1	1	0	112	212	1.96 ± 0.14
100	3	5	1	0	160	260	2.58 ± 0.14
100	3	1	4	0	121	239	2.96 ± 0.14
100	3	5	4	2	229	395	3.92 ± 0.20
100	5	5	4	2	315	525	5.90 ± 0.35

We observe that the computing time evolves linearly with the number of examples that are given as input to the algorithm. For the inference of a UTA-poly model, the higher the degree of the polynomials, the higher the computing time; however the difference is not substantial. Compared to an UTA model, learning a UTA-poly model using polynomials of the 4th degree increases the computing time of a few dozen of milliseconds. The behavior is similar when passing from one to several pieces per marginal. When the number of criteria increases, we observe that the computing time increases too.

Lastly, it should be emphasized that computing times for all instances solved in this section are reasonably short (less than 6 sec.), and compatible with an iterative and interactive use with a DM.

7.5 Experiments with real data sets

In this section we describe the experiments we have performed with 5 real data sets in the sorting context. We studied the impact of the degree of the polynomials and the number of pieces for the learning and testing performances.

To proceed, we used UTADIS-poly and UTADIS-splines which are respectively an adaptation of UTA-poly and UTA-splines for learning an additive value function sorting (AVF-Sort) model. We recall that the objective function and the constraints for learning such a model are given in Equations (2.26) and (2.27). In order to guarantee, the monotonicity of the marginals, the same type of constraints are used as for UTA-poly and UTA-splines.

7.5.1 Data sets and experimental design

The characteristics of the data sets that have been used to test UTADIS-poly and UTADIS-splines are presented in Table 7.4. Note that some of these data sets have been already used in Chapter 3 to test the metaheuristic learning a majority rule sorting (MR-Sort) model.

Table 7.4: Datasets used to test UTADIS-splines.

Dataset	# instance	# attributes	# categories
CEV	1728	6	4
CPU	209	6	4
JRA	172	5	4
LEV	1000	4	5
SWD	1000	10	4

The data sets contain from 172 to 1728 alternatives. The number of attributes varies between 4 and 10 and the number of categories varies between 4 and 5. The values on each attribute have been normalized between 0 and 1. For the experiments, we split the data sets in a twofold partition. The first class of the partition contains the alternatives that are given as input to UTADIS-poly or UTADIS-splines. The second class of the partition contains the alternatives that are used as test set. This test set is used to assess the quality of the model returned by the semidefinite program. As quality index, we use the classification accuracy (see Equation (2.28)).

The following ratios between the size of the learning set and the size of the test set are considered: 30/70, 50/50, 70/30. Similarly to what has been done in Section 3.5, a random drawing of the learning set from the whole data set is repeated 100 times, yielding 100 instances of a partition of the data set in a learning set and test set.

7.5.2 Results

We present the results that we obtained with UTADIS, UTADIS-poly and UTADIS-splines. We varied the degree of the polynomials used for the marginals, the number of pieces per marginal and the continuity degree of the polynomials at the breakpoints.

7.5.3 Variation of the degree of the polynomials

We first tested UTADIS-poly and varied the degree of the polynomials used for the marginals. Tests have been done with polynomials of degree 1, 2 and 3. Note

Table 7.5: Average and standard deviation of the classification accuracy of the learning set which is composed of 30, 50 or 70% of the alternatives of the data set. The first column corresponds to the ratio of alternatives that are used as learning set. The tuple k-D-D_c in first row corresponds to the number of pieces per marginal k, the degree of the polynomials D and the continuity degree D_c at the breakpoints.

Size	Data set	1-1-0	1-2-0	1-3-0
30 %	CEV	0.7558 ± 0.0242	0.7760 ± 0.0245	0.7609 ± 0.0239
	CPU	0.9967 ± 0.0124	0.9990 ± 0.0054	0.9995 ± 0.0048
	JRA	0.7196 ± 0.0720	0.7123 ± 0.0792	0.7452 ± 0.0722
	LEV	0.6128 ± 0.0285	0.6201 ± 0.0260	0.6167 ± 0.0307
	SWD	0.5738 ± 0.0318	0.5870 ± 0.0386	0.5923 ± 0.0342
50 %	CEV	0.7533 ± 0.0181	0.7756 ± 0.0176	0.7597 ± 0.0172
	CPU	0.9918 ± 0.0157	0.9956 ± 0.0121	0.9974 ± 0.0098
	JRA	0.7030 ± 0.0600	0.6963 ± 0.0573	0.7242 ± 0.0574
	LEV	0.6088 ± 0.0252	0.6135 ± 0.0254	0.6081 ± 0.0244
	SWD	0.5681 ± 0.0252	0.5808 ± 0.0288	0.5778 ± 0.0277
70 %	CEV	0.7508 ± 0.0150	0.7735 ± 0.0144	0.7555 ± 0.0161
	CPU	0.9869 ± 0.0178	0.9897 ± 0.0163	0.9950 ± 0.0115
	JRA	0.6941 ± 0.0475	0.6904 ± 0.0500	0.7111 ± 0.0413
	LEV	0.6058 ± 0.0185	0.6108 ± 0.0177	0.6049 ± 0.0198
	SWD	0.5656 ± 0.0228	0.5735 ± 0.0259	0.5742 ± 0.0264

that using polynomials of degree 1 for the marginals amounts to use a weighted sum for determining the score of an alternative. It is also equivalent to the Logistic regression in preference learning (PL).

Table 7.5 shows the classification accuracy of the learning set when the degree of the polynomials of a UTADIS-poly model varies between 1 and 3 with a learning set composed of 30%, 50% and 70% of the data set. For a majority of the data sets, the AVF-Sort model using the polynomials of the highest degree returns the best results. However this is not always the case. Sometimes polynomials of the second degree perform better. It can be due to the objective function which does not maximize the classification accuracy but minimizes slack variables which may lead to compensatory effects. Indeed, the solver may adapt the marginals so that the slack of a majority of wrongly assigned alternatives is reduced at the cost of some other misclassification.

We illustrate the potential compensatory effects with an example. Consider that we want to discriminate a set of alternatives in two categories C^1 and C^2,

with $C^2 \succcurlyeq C^1$, on the basis of their performances on one criterion. In this aim, we try to find a threshold on this criterion so that alternatives that have a performance better than or equal to the threshold are assigned to C^2 and the others are assigned to C^1. As learning set, we have a set of alternatives which are assigned to one of the two categories together with their performances on the criterion. Table 7.6 shows the performances and assignments of these alternatives. Assignments of alternatives $a^{(6)}$ and $a^{(7)}$ are such that it is impossible to restore all the examples because both alternatives are assigned to C^1 while their performances are greater than the one of alternatives assigned in C^2. To maximize the number of correctly assigned alternatives, it is obvious that the threshold delimiting C^1 from C^2 should be set to 0.5 so that 5 of the 7 examples are correctly restored. However, if the constraints and objective function of UTADIS are used (see Equations (2.26) and (2.27)), only 4 of the 7 examples will be correctly assigned. Indeed, with a threshold set to 0.5, the objective function of UTADIS would be equal to 0.8 because the distance of alternatives $a^{(6)}$ and $a^{(7)}$ to the threshold is equal to 0.4. With a threshold fixed to 0.6, the objective function is equal to 0.7: the distance of the alternative $a^{(5)}$ to the threshold is equal to 0.1 and the distance between $a^{(6)}$ and $a^{(7)}$ is equal to 0.3. The best solution is not the one restoring the highest number of examples when the objective function of UTADIS is used.

Table 7.6: Performances and assignments of a set of alternatives incompatible with an AVF-Sort model.

alternative	category	performance
$a^{(1)}$	C^1	0.4
$a^{(2)}$	C^1	0.4
$a^{(3)}$	C^2	0.6
$a^{(4)}$	C^2	0.6
$a^{(5)}$	C^2	0.5
$a^{(6)}$	C^1	0.9
$a^{(7)}$	C^1	0.9

Table 7.7 shows the classification accuracy of the test set for different sizes of the learning set and marginals using polynomials of degree 1 to 3. We don't observe the same trend as for the learning set. The classification accuracy is not better when the degree of the polynomials increases. Indeed, for CPU and LEV the better classification accuracy is obtained with a weighted sum. With CEV and SWD, the better classification accuracy is obtained with polynomials of the second degree. UTADIS-poly achieves better results with higher degree polynomials only for the JRA data set. These results tend to indicate an overfitting

Table 7.7: Average and standard deviation of the classification accuracy of the test set with a learning set composed of 30, 50 or 70 percents of the alternatives of the data set and a test set composed of the disjoint part. The first column corresponds to the ratio of alternatives that are used as learning set. The tuple k-D-D_c in first row corresponds to the number of pieces per marginal k, the degree of the polynomials D and the continuity degree D_c at the breakpoints.

Size	Data set	1-1-0	1-2-0	1-3-0
30 %	CEV	0.7449 ± 0.0110	0.7662 ± 0.0099	0.7487 ± 0.0109
	CPU	0.9195 ± 0.0350	0.9119 ± 0.0319	0.9050 ± 0.0315
	JRA	0.5950 ± 0.0537	0.6084 ± 0.0540	0.6248 ± 0.0399
	LEV	0.5935 ± 0.0134	0.5902 ± 0.0146	0.5768 ± 0.0164
	SWD	0.5408 ± 0.0174	0.5439 ± 0.0201	0.5395 ± 0.0186
50 %	CEV	0.7463 ± 0.0115	0.7668 ± 0.0130	0.7502 ± 0.0133
	CPU	0.9365 ± 0.0261	0.9275 ± 0.0272	0.9166 ± 0.0279
	JRA	0.6134 ± 0.0467	0.6177 ± 0.0487	0.6342 ± 0.0376
	LEV	0.5961 ± 0.0145	0.5933 ± 0.0159	0.5810 ± 0.0169
	SWD	0.5481 ± 0.0164	0.5495 ± 0.0186	0.5436 ± 0.0195
70 %	CEV	0.7444 ± 0.0141	0.7657 ± 0.0136	0.7473 ± 0.0139
	CPU	0.9408 ± 0.0235	0.9308 ± 0.0242	0.9234 ± 0.0285
	JRA	0.6309 ± 0.0476	0.6315 ± 0.0502	0.6405 ± 0.0414
	LEV	0.5982 ± 0.0189	0.5938 ± 0.0188	0.5859 ± 0.0197
	SWD	0.5479 ± 0.0183	0.5506 ± 0.0196	0.5470 ± 0.0162

effect of UTADIS-poly.

7.5.4 Variation of the number of pieces

We study the influence of the number of pieces on the quality of the solution. In this aim, we use polynomials of the third degree. We don't impose the continuity of any derivative at the breakpoints. As for the previous experiment, the data sets are split in a twofold partition: a learning set and a test set.

Table 7.8 shows the average classification accuracy of the learning set and its standard deviation for different sizes of learning set and different numbers of pieces per marginal. We observe that the best results are in general obtained with marginals using more than one piece.

For the data set JRA we observe a benefit of using more than one piece per marginal. With a learning set composed of 70 percents of the data set, the classification is improved by 1 percent when using 2 pieces per marginal instead

Table 7.8: Average and standard deviation of the classification accuracy of the learning set which is composed of 30, 50 or 70% of the alternatives of the data set. The first column corresponds to the ratio of alternatives that are used as learning set. The tuple k-D-D_c in first row corresponds to the number of pieces per marginal k, the degree of the polynomials D and the continuity degree D_c at the breakpoints.

Size	Data set	1-3-0	2-3-0	3-3-0
30 %	CEV	0.7609 ± 0.0239	0.7722 ± 0.0180	0.7729 ± 0.0201
	CPU	0.9995 ± 0.0048	1.0000 ± 0.0000	1.0000 ± 0.0000
	JRA	0.7452 ± 0.0722	0.7608 ± 0.0675	0.7692 ± 0.0765
	LEV	0.6167 ± 0.0307	0.6222 ± 0.0328	0.6211 ± 0.0298
	SWD	0.5923 ± 0.0342	0.5952 ± 0.0351	0.5864 ± 0.0358
50 %	CEV	0.7597 ± 0.0172	0.7686 ± 0.0164	0.7700 ± 0.0157
	CPU	0.9974 ± 0.0098	0.9995 ± 0.0039	1.0000 ± 0.0000
	JRA	0.7242 ± 0.0574	0.7371 ± 0.0577	0.7488 ± 0.0578
	LEV	0.6081 ± 0.0244	0.6126 ± 0.0263	0.6132 ± 0.0252
	SWD	0.5778 ± 0.0277	0.5792 ± 0.0268	0.5771 ± 0.0256
70 %	CEV	0.7555 ± 0.0161	0.7677 ± 0.0159	0.7674 ± 0.0147
	CPU	0.9950 ± 0.0115	0.9978 ± 0.0072	0.9997 ± 0.0027
	JRA	0.7111 ± 0.0413	0.7229 ± 0.0428	0.7368 ± 0.0451
	LEV	0.6049 ± 0.0198	0.6097 ± 0.0197	0.6095 ± 0.0194
	SWD	0.5742 ± 0.0264	0.5733 ± 0.0229	0.5731 ± 0.0242

of one. Using 3 pieces allows to increase the classification accuracy by one more percent.

For the other data sets, the gain is not so important. We also observe a loss in classification accuracy for some data sets. This loss is due to several reason. One of them is the objective function which does not explicitly maximize the classification accuracy as said in the previous section. Numerical errors also degrade the solution. With large data sets, we observe more errors than with small ones. The worst performances are obtained with CEV, LEV and SWD data sets which contain more than 1000 alternatives. Numerical errors are more likely when the degree of the polynomials increases since small numbers are added and subtracted to big ones and the contrary. We did not investigate further numerical issues in this thesis.

Table 7.9 shows the average and standard deviation of the classification accuracy of the test set for different sizes of learning set and numbers of pieces per marginals. The best results are obtained with marginals composed of one piece.

Table 7.9: Average and standard deviation of the classification accuracy of the test set with a learning set composed of 30, 50 or 70% of the alternatives of the data set and a test set composed of the disjoint part. The first column corresponds to the ratio of alternatives that are used as learning set. The tuple k-D-D_c in first row corresponds to the number of pieces per marginal k, the degree of the polynomials D and the continuity degree D_c at the breakpoints.

Size	Data set	1-3-0	2-3-0	3-3-0
30 %	CEV	0.7487 ± 0.0109	0.7586 ± 0.0112	0.7589 ± 0.0113
	CPU	0.9050 ± 0.0315	0.9014 ± 0.0294	0.8955 ± 0.0310
	JRA	0.6248 ± 0.0399	0.6187 ± 0.0438	0.6151 ± 0.0569
	LEV	0.5768 ± 0.0164	0.5731 ± 0.0168	0.5742 ± 0.0177
	SWD	0.5395 ± 0.0186	0.5383 ± 0.0183	0.5364 ± 0.0189
50 %	CEV	0.7502 ± 0.0133	0.7594 ± 0.0095	0.7619 ± 0.0113
	CPU	0.9166 ± 0.0279	0.9157 ± 0.0304	0.9112 ± 0.0318
	JRA	0.6342 ± 0.0376	0.6331 ± 0.0407	0.6351 ± 0.0409
	LEV	0.5810 ± 0.0169	0.5783 ± 0.0166	0.5793 ± 0.0169
	SWD	0.5436 ± 0.0195	0.5435 ± 0.0188	0.5422 ± 0.0184
70 %	CEV	0.7473 ± 0.0139	0.7585 ± 0.0121	0.7592 ± 0.0132
	CPU	0.9234 ± 0.0285	0.9215 ± 0.0294	0.9223 ± 0.0292
	JRA	0.6405 ± 0.0414	0.6425 ± 0.0445	0.6460 ± 0.0433
	LEV	0.5859 ± 0.0197	0.5828 ± 0.0195	0.5818 ± 0.0201
	SWD	0.5470 ± 0.0162	0.5441 ± 0.0184	0.5443 ± 0.0180

It shows a trend to overfitting effects when the number of degrees of freedom increases.

7.5.5 Variation of the degree of continuity

In this subsection we study the impact of the continuity degree on the classification accuracy. Table 7.10 shows the average and standard deviation of the classification accuracy of the learning set for different sizes of learning set and different continuity degrees at the breakpoints. The marginals are composed of 3 polynomials of the third degree.

We observe that the best results are generally obtained when there is no continuity of the derivative at the breakpoints. It is no wonder since the number of degree of freedom is higher when there is no constraint on the derivative at the breakpoints. Nevertheless, there are no big differences between models in which the continuity of the derivative at the breakpoints is not guaranteed. For

Table 7.10: Average and standard deviation of the classification accuracy of the learning set which is composed of 30, 50 or 70% of the alternatives of the data set. The first column corresponds to the ratio of alternatives that are used as learning set. The tuple k-D-D_c in first row corresponds to the number of pieces per marginal k, the degree of the polynomials D and the continuity degree D_c at the breakpoints.

Size	Data set	3-3-0	3-3-1	3-3-2
30 %	CEV	0.7729 ± 0.0201	0.7735 ± 0.0194	0.7744 ± 0.0215
	CPU	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000
	JRA	0.7692 ± 0.0765	0.7662 ± 0.0690	0.7620 ± 0.0677
	LEV	0.6211 ± 0.0298	0.6220 ± 0.0321	0.6198 ± 0.0312
	SWD	0.5864 ± 0.0358	0.5879 ± 0.0345	0.5912 ± 0.0325
50 %	CEV	0.7700 ± 0.0157	0.7706 ± 0.0149	0.7719 ± 0.0160
	CPU	1.0000 ± 0.0000	0.9997 ± 0.0029	0.9990 ± 0.0056
	JRA	0.7488 ± 0.0578	0.7412 ± 0.0555	0.7396 ± 0.0578
	LEV	0.6132 ± 0.0252	0.6119 ± 0.0274	0.6125 ± 0.0251
	SWD	0.5771 ± 0.0256	0.5781 ± 0.0255	0.5773 ± 0.0255
70 %	CEV	0.7674 ± 0.0147	0.7694 ± 0.0134	0.7698 ± 0.0141
	CPU	0.9997 ± 0.0027	0.9993 ± 0.0039	0.9976 ± 0.0071
	JRA	0.7368 ± 0.0451	0.7279 ± 0.0426	0.7258 ± 0.0443
	LEV	0.6095 ± 0.0194	0.6068 ± 0.0192	0.6067 ± 0.0213
	SWD	0.5731 ± 0.0242	0.5733 ± 0.0239	0.5725 ± 0.0248

some data set, we observe that the algorithm performs slightly better when the continuity of the second derivative is guaranteed at the breakpoints (e.g. SWD and CEV). We believe that this behavior is due to numerical errors.

Table 7.11 shows the average classification accuracy and standard deviation of the test set. Adding constraints to ensure the continuity of the first and second derivative increases the classification accuracy of the test set. It reduces the overfitting effect that happens when the continuity of the derivatives is not guaranteed. Moreover it enables to limit discontinuities at the breakpoints.

Table 7.11: Average and standard deviation of the classification accuracy of the test set with a learning set composed of 30, 50 or 70% of the alternatives of the data set and a test set composed of the disjoint part. The first column corresponds to the ratio of alternatives that are used as learning set. The tuple k-D-D_c in first row corresponds to the number of pieces per marginal k, the degree of the polynomials D and the continuity degree D_c at the breakpoints.

Size	Data set	3-3-0	3-3-1	3-3-2
30 %	CEV	0.7589 ± 0.0113	0.7582 ± 0.0111	0.7603 ± 0.0115
	CPU	0.8955 ± 0.0310	0.8962 ± 0.0315	0.8988 ± 0.0296
	JRA	0.6151 ± 0.0569	0.6158 ± 0.0477	0.6197 ± 0.0478
	LEV	0.5742 ± 0.0177	0.5738 ± 0.0191	0.5738 ± 0.0180
	SWD	0.5364 ± 0.0189	0.5375 ± 0.0188	0.5380 ± 0.0191
50 %	CEV	0.7619 ± 0.0113	0.7609 ± 0.0105	0.7626 ± 0.0105
	CPU	0.9112 ± 0.0318	0.9103 ± 0.0325	0.9116 ± 0.0305
	JRA	0.6351 ± 0.0409	0.6361 ± 0.0423	0.6385 ± 0.0461
	LEV	0.5793 ± 0.0169	0.5790 ± 0.0152	0.5780 ± 0.0166
	SWD	0.5422 ± 0.0184	0.5428 ± 0.0189	0.5424 ± 0.0187
70 %	CEV	0.7592 ± 0.0132	0.7604 ± 0.0124	0.7597 ± 0.0139
	CPU	0.9223 ± 0.0292	0.9208 ± 0.0291	0.9208 ± 0.0273
	JRA	0.6460 ± 0.0433	0.6420 ± 0.0428	0.6482 ± 0.0451
	LEV	0.5818 ± 0.0201	0.5820 ± 0.0187	0.5815 ± 0.0178
	SWD	0.5443 ± 0.0180	0.5447 ± 0.0186	0.5445 ± 0.0171

7.6 Conclusion

In this chapter, we proposed a new method to learn an additive value functions model from a set of statements provided by the DM. Learning piecewise linear value functions from preference statements is standard in the literature (UTA methods, e.g. Jacquet-Lagrèze and Siskos (1982), Jacquet-Lagrèze and Siskos (2001)). Instead of piecewise linear marginals, we generalize this standard representation by considering more general forms for marginals. UTA-poly considers marginal value functions which are monotone polynomials, while in UTA-splines

marginals are composed of several pieces of monotone polynomials. UTA-splines generalizes the preference representation used in the standard UTA methods, while UTA-poly is a particular UTA-splines model where a single polynomial is used to represent each marginal.

The inference of such an additive value function with polynomial marginals is performed using a semidefinite programming formulation. From a computational point of view, the resolution of instances corresponding to real data sets is limited to several seconds, and thus compatible with an interactive use with DMs.

We provide an illustrative example showing that the inference program is able to restore value functions that are "close" to the original ones. A specific feature of the methods is that the inferred value function is composed of "smooth" marginals which avoids brutal changes in the slopes of these marginals, thus improving interpretability.

The computational experiments show the ability of the methods to better match the preference statements as the degree of the polynomials involved in the marginals increases.

We extended the method to sorting problems and studied the behavior of the algorithm on real data sets. It showed that the extra degrees of freedom given to the algorithm don't always help to find better solutions. To avoid overfitting effects, the number of pieces and degree of the polynomials have to be chosen carefully.

An innovative aspect of this work is related to the new optimization technique allowing to deal with polynomial and piecewise polynomial marginals instead of piecewise linear marginals. The semidefinite programming approach used here for UTA opens new perspectives for eliciting other preference models based on additive or partly additive value structures, such as additive differences models, e.g. MACBETH (Bana e Costa and Vansnick, 1994; Bana e Costa et al., 2005), and GAI networks (Gonzales et al., 2011).

Similarly as for UTA models, the solution of our new models might not be unique. It would be interesting to try to characterize these situations and pick a solution that is most suited for the DM. Note that, for this work, we used interior-point methods to solve the semidefinite programs. These methods return the so-called analytic center of the set of optimal solutions, that is, it returns a solution 'in the middle' of the set of optimal solutions, similarly as UTASTAR and ACUTA would do for UTA models.

Chapter 8

Conclusion and perspectives

In this thesis we mainly focused on sorting problems, i.e. problems in which each alternative of a set has to be classified in a category selected among a set of predefined and ordered categories. We developed multiple-criteria decision analysis (MCDA) learning methods that are able to deal with large data sets as the ones found in the preference learning (PL) field. We validated these algorithms by using similar validation techniques as in the PL field.

In Chapter 3 we developed a metaheuristic designed to learn a MCDA sorting model called the majority rule sorting (MR-Sort) model. The metaheuristic was validated with artificial and real data sets. Experiments have shown that we were able to compete with PL algorithms and that the approximated solutions obtained with the metaheuristic were close to the optimal ones obtained by a mixed integer program (MIP) inferring a MR-Sort model with the same objective function.

For further research, we advocate an analysis of the complexity of the MR-Sort model. To do so, one can consider analysing the MR-Sort model in terms of the VC dimension (Vapnik, 1998) which measures statistically the capacity of a classification algorithm. It allows to predict a probabilistic upper bound on the test error of a classification model. This approach is often used in the context of PL (see e.g. Tehrani et al., 2012).

The use of metaheuristic is not a common practice in the PL field. During our research, our attention has been drawn to convex relaxation techniques. In recent years, some advances in this field have been done in order to solve problems in which the objective function is not convex (see e.g. Zhang, 2010). Using such techniques with MR-Sort deserves to be studied in further research.

MR-Sort presents the advantage of being easy to interpret. It can be explained to a decision maker (DM) as the application of compact and intuitive rules. This characteristic is useful for some applications. In Chapter 4 we have shown that the

MR-Sort metaheuristic was adapted to deal with a medical application in which doctors prefer to have models that are interpretable and easy to understand. The application consisted in determining the ASA score of patients and then to determine whether they can be accepted for surgery. The metaheuristic has been able to find a MR-Sort model that performs better than other preference learning algorithms in a reasonable amount of time. We described the rules induced by one of the MR-Sort model for the prediction of the ASA score. These rules are easily interpretable by users who are not familiar with MCDA methods such as doctors.

However, when dealing with the application of Chapter 4, we identified a weakness of the MR-Sort model inferred by the metaheuristic: the value of the weights and the majority threshold of the model returned by the inference procedure involve a substantial amount of arbitrariness. The weights may not adequately represent the importance of the criteria. Two criteria having the same decision power in a coalition may have very different weights. Depending on the value of the weights, the model can be more easy or difficult to understand. That's the reason why we proposed a post-treatment process for the weights and the majority threshold. After the metaheuristic found a model, the post-processing consists in running a MIP in order to find a set of weights representing as well as possible the importance of each criterion. We obtained satisfactory results with this post-treatment process. After running it on the inferred model, the MIP returned weights and majority threshold that were better reflecting the importance of each criterion in the model. Nevertheless, the question of finding representative weights and majority threshold deserves further research, in particular, in the aim of defining precisely what can be called "representative weights".

In Chapter 5, we worked with the non-compensatory sorting (NCS) model which is an extension of MR-Sort. Compared to MR-Sort, this model enables to take criteria interactions into account. We proposed two formulations in order to infer the parameters of such a model. The first one is a MIP and the second one is an extension of the metaheuristic presented in Chapter 3. The MIP is not adapted to deal with large problems as the ones found in the PL field because of the required computing time. We were therefore not able to use it on the PL data sets considered in Chapter 5 since the computing time required to obtain a solution is prohibitive. However we were able to use the metaheuristic with these data sets. We observed that the metaheuristic for learning a NCS model does not perform better than the one inferring the parameters of a MR-Sort model. Following these observations, the following question arose: "By how much the expressivity of a NCS model is improved as compared to a MR-Sort model?". We answered this question for models involving up to 6 criteria. Furthermore, we computed the list of minimal sufficient coalitions (MSCs) for n criteria, with $1 \leq n \leq 6$. Among these coalitions we verified successively which

ones were representable with 1-additive, 2-additive or 3-additive capacities and a threshold. We observed that all the sufficient coalitions are representable with 2-additive capacities for $n \leq 5$. For $n = 6$ only 2% of the sufficient coalitions are not representable by 2-additive capacities but all of them are representable with 3-additive capacities. Approximating the proportion of sufficient coalitions representable with k-additive weights and a threshold for $n = 7$ is one direction that deserves to be studied. We also wonder how a single additive set of weights can approximate separating functions that are not 1-additive for $n = 7$.

In Chapter 6, we proposed a new type of veto rule for the MR-Sort model. This new rule enriches the expressivity of the model. The rule enables the veto effect when the alternative goes below a threshold on a subset of criteria. It is a more general form of the standard binary veto. A mixed integer program designed for learning the parameters of a MR-Sort model with coalitional veto (MR-Sort model with coalitional veto (MR-Sort-CV) model) has been proposed and tested on a fictive application. This MIP is not suitable for large data sets since the computing time becomes quickly prohibitive even for small data sets. That's why we outlined a strategy based on the metaheuristic presented in Chapter 3 in order to learn a MR-Sort model with coalitional veto. As for MR-Sort, one can consider using relaxation techniques in order to learn the parameters of a MR-Sort-CV model. This direction deserves further research. The axiomatic characterization of the coalitional veto rule also deserves further research.

In Chapter 7, we addressed the problem of learning additive value functions. Additive value function models can be used to deal with ranking and sorting problems. UTA and UTADIS are two linear programming methods that are designed for learning the parameters of an additive value function model respectively for ranking and sorting problems. The additive value functions learned by these procedures are piecewise linear. We proposed in Chapter 7 the use of semidefinite programming (SDP) in order to learn polynomial functions instead of piecewise linear ones for the marginal value functions. Four new algorithms have been developed. UTA-poly and UTADIS-poly learn additive value functions models using one polynomial per marginal. UTA-splines and UTADIS-splines learn additive value functions models using multiple polynomials per marginal, i.e. splines. We made experiments on artificial and real data sets. The results of these experiments have shown that learning the parameters of a UTA-splines model does not require more computing time than learning the parameters of a UTA model. We have seen that models built with UTA-poly and UTA-splines are more easily interpretable than models built with UTA since there is no discontinuity of the marginal value function at the breakpoints[1]. However, experimental results on real data sets have shown that using UTA-splines may lead to overfitting since

[1] Whereas the continuity at the breakpoints in ensured up to the second derivative for UTA-splines.

it adds more freedom degrees to the model. It is therefore important to choose accordingly the parameters of the model. This direction deserves to be studied more in depth. Another weakness of UTA and UTADIS formulations is their objective function which consists in minimizing a slack. This objective function can lead to compensatory effects which means that the solution found by the solver may not maximize the compatibility of the learning set with the inferred model if there is noise in the data set. Using an objective function as in ACUTA is also a direction that deserves to be exploited. Finally, the semidefinite programming approach used in Chapter 7 opens new perspectives for eliciting other preference models based on additive or partly additive value structures, e.g. MACBETH (Bana e Costa and Vansnick, 1994; Bana e Costa et al., 2005), and GAI networks (Gonzales et al., 2011).

In this thesis, we showed some links between PL and MCDA. The different chapters showed that multiple criteria decision analysis can benefit from preference learning elicitation techniques. We demonstrated this in the context of sorting and ranking problems. Our research can be extended to other MCDA methods. Recently multiple initiatives have emerged in order to use MCDA methods in the context of PL (see e.g. Gurrieri, 2015; Liu, 2016). In this thesis we also opened up several research avenues: the use of semi-definite programming in the context of MCDA and PL, the improvement of the interpretability of the MR-Sort model and the concept of coalitional veto in outranking models.

The list of our contributions is available in Appendix G.

Appendix A

MR-Sort metaheuristic: additional confusion matrices

A.1 Confusion matrices of binary data sets

In this section, we give the confusion matrices for all the binarized data sets that have been used in Chapter 3.

Table A.1: Confusion matrices of the test set for the (binarized) DBS data set. Actual class in rows, predicted class in columns.

(a) META - DBS 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	41.64 ±4.52	8.77 ±4.57
C^2	10.20 ±5.72	39.40 ±5.63

(b) MIP - DBS 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	42.21 ±4.56	8.18 ±4.48
C^2	11.59 ±6.28	38.02 ±5.75

(c) UTADIS - DBS 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	41.94 ±5.23	8.45 ±5.28
C^2	11.64 ±5.67	37.98 ±5.40

(d) META - DBS 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	42.07 ±4.58	8.57 ±4.29
C^2	7.67 ±5.04	41.70 ±4.59

(e) MIP - DBS 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	43.73 ±4.71	6.67 ±3.69
C^2	10.20 ±5.21	39.40 ±5.50

(f) UTADIS - DBS 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	44.35 ±4.70	6.05 ±3.53
C^2	8.77 ±4.42	40.83 ±5.19

(g) META - DBS 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	42.21 ±9.13	8.25 ±5.30
C^2	7.67 ±6.06	41.88 ±9.35

(h) MIP - DBS 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	41.75 ±8.37	6.88 ±5.69
C^2	7.92 ±6.31	43.46 ±8.51

(i) UTADIS - DBS 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	42.92 ±8.30	5.71 ±4.13
C^2	7.08 ±4.79	44.29 ±7.74

Table A.2: Confusion matrices of the test set for the (binarized) CPU data set. Actual class in rows, predicted class in columns.

(a) META - CPU 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	44.77 ±3.71	4.15 ±3.19
C^2	5.74 ±3.46	44.96 ±3.68

(b) MIP - CPU 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	45.65 ±3.35	3.55 ±3.10
C^2	5.45 ±2.88	45.35 ±3.04

(c) UTADIS - CPU 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	47.09 ±2.85	2.11 ±2.39
C^2	4.40 ±3.54	46.39 ±3.69

(d) META - CPU 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	46.47 ±4.26	2.50 ±2.21
C^2	4.12 ±2.17	46.30 ±4.52

(e) MIP - CPU 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	46.65 ±3.75	2.54 ±2.14
C^2	3.86 ±2.07	46.95 ±3.56

(f) UTADIS - CPU 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	48.31 ±3.14	0.88 ±1.25
C^2	1.42 ±1.76	49.39 ±3.61

(g) META - CPU 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	47.81 ±7.06	2.02 ±2.69
C^2	4.43 ±2.83	45.74 ±7.02

(h) MIP - CPU 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	46.33 ±6.86	2.45 ±2.92
C^2	3.52 ±2.86	47.69 ±6.93

(i) UTADIS - CPU 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	48.29 ±7.06	0.50 ±1.03
C^2	1.02 ±1.98	50.19 ±6.77

Table A.3: Confusion matrices of the test set for the (binarized) BCC data set. Actual class in rows, predicted class in columns.

(a) META - BCC 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	59.91 ±4.47	10.55 ±4.93
C^2	17.69 ±4.32	11.85 ±3.80

(b) MIP - BCC 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	61.10 ±4.32	9.36 ±4.65
C^2	17.43 ±3.74	12.11 ±3.07

(c) UTADIS - BCC 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	59.48 ±5.44	11.18 ±6.01
C^2	17.97 ±4.96	11.37 ±4.26

(d) META - BCC 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	60.53 ±4.52	10.28 ±4.86
C^2	17.22 ±4.27	11.96 ±3.19

(e) MIP - BCC 50 %

Not available

(f) UTADIS - BCC 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	60.01 ±4.12	10.72 ±4.32
C^2	17.82 ±4.71	11.45 ±3.89

(g) META - BCC 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	45.05 ±4.96	8.47 ±2.84
C^2	8.39 ±3.21	38.09 ±4.71

(h) MIP - BCC 80 %

Not available

(i) UTADIS - BCC 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	58.79 ±6.64	11.77 ±5.39
C^2	17.36 ±5.74	12.09 ±4.99

Table A.4: Confusion matrices of the test set for the (binarized) MPG data set. Actual class in rows, predicted class in columns.

(a) META - MPG 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	43.91 ±3.08	9.86 ±3.34
C^2	10.39 ±4.50	35.84 ±4.13

(b) MIP - MPG 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	44.79 ±2.86	8.93 ±3.27
C^2	11.87 ±5.00	34.40 ±4.49

(c) UTADIS - MPG 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	42.79 ±3.26	10.93 ±3.47
C^2	11.30 ±3.85	34.97 ±3.52

(d) META - MPG 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	44.28 ±2.64	9.18 ±2.31
C^2	8.63 ±2.29	37.91 ±2.8

(e) MIP - MPG 50 %

Not available

(f) UTADIS - MPG 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	42.46 ±2.65	11.18 ±2.49
C^2	9.70 ±2.45	36.66 ±2.85

(g) META - MPG 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	61.48 ±6.49	9.98 ±6.13
C^2	16.79 ±6.02	11.75 ±4.38

(h) MIP - MPG 80 %

Not available

(i) UTADIS - MPG 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	42.34 ±4.70	11.27 ±3.49
C^2	9.53 ±3.62	36.86 ±5.24

Table A.5: Confusion matrices of the test set for the (binarized) ESL data set. Actual class in rows, predicted class in columns.

(a) META - ESL 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	49.28 ±2.16	5.49 ±2.21
C^2	4.93 ±2.72	40.30 ±2.58

(b) MIP - ESL 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	49.35 ±2.14	5.61 ±2.33
C^2	5.14 ±2.54	39.91 ±2.27

(c) UTADIS - ESL 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	50.30 ±2.02	4.66 ±1.99
C^2	4.23 ±1.93	40.82 ±1.81

(d) META - ESL 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	48.61 ±3.02	5.97 ±2.38
C^2	4.07 ±2.40	41.35 ±2.74

(e) MIP - ESL 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	48.82 ±2.97	5.82 ±2.44
C^2	4.35 ±2.54	41.01 ±2.80

(f) UTADIS - ESL 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	50.31 ±2.32	4.34 ±1.86
C^2	3.50 ±2.02	41.86 ±2.39

(g) META - ESL 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	48.96 ±5.51	6.09 ±3.20
C^2	3.92 ±2.57	41.03 ±5.02

(h) MIP - ESL 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	48.89 ±4.37	5.89 ±2.73
C^2	4.18 ±2.44	41.04 ±4.02

(i) UTADIS - ESL 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	50.59 ±4.46	4.18 ±2.52
C^2	3.26 ±2.30	41.97 ±3.83

Table A.6: Confusion matrices of the test set for the (binarized) MMG data set. Actual class in rows, predicted class in columns.

(a) META - MMG 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	49.28 ±2.16	5.49 ±2.21
C^2	4.93 ±2.72	40.30 ±2.58

(b) MIP - MMG 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	43.32 ±2.68	8.03 ±2.86
C^2	9.13 ±2.26	39.52 ±2.11

(c) UTADIS - MMG 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	43.14 ±3.39	8.21 ±3.64
C^2	10.19 ±2.90	38.46 ±2.64

(d) META - MMG 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	48.61 ±3.02	5.97 ±2.38
C^2	4.07 ±2.40	41.35 ±2.74

(e) MIP - MMG 50 %

Not available

(f) UTADIS - MMG 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	44.14 ±2.76	7.32 ±2.98
C^2	10.26 ±3.14	38.28 ±2.83

(g) META - MMG 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	48.96 ±5.51	6.09 ±3.20
C^2	3.92 ±2.57	41.03 ±5.02

(h) MIP - MMG 80 %

Not available

(i) UTADIS - MMG 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	44.34 ±3.70	7.43 ±2.98
C^2	9.92 ±3.96	38.32 ±4.07

Table A.7: Confusion matrices of the test set for the (binarized) LEV data set. Actual class in rows, predicted class in columns.

(a) META - LEV 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	70.45 ±2.94	7.21 ±3.07
C^2	9.53 ±2.65	12.81 ±2.49

(b) MIP - LEV 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	71.65 ±2.36	6.01 ±2.53
C^2	10.07 ±2.49	12.27 ±2.20

(c) UTADIS - LEV 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	69.47 ±2.11	8.01 ±2.16
C^2	8.53 ±1.56	13.99 ±1.40

(d) META - LEV 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	71.14 ±2.34	6.57 ±2.51
C^2	9.35 ±2.26	12.94 ±1.90

(e) MIP - LEV 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	72.84 ±1.62	4.87 ±1.31
C^2	9.35 ±1.48	12.93 ±1.41

(f) UTADIS - LEV 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	70.06 ±1.71	7.25 ±1.47
C^2	8.30 ±1.23	14.39 ±1.3

(g) META - LEV 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	71.79 ±3.15	6.40 ±2.46
C^2	9.76 ±2.37	12.06 ±2.41

(h) MIP - LEV 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	73.52 ±2.65	4.66 ±1.49
C^2	8.93 ±1.72	12.90 ±2.25

(i) UTADIS - LEV 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	70.11 ±2.98	7.19 ±2.10
C^2	8.53 ±2.05	14.18 ±2.35

Table A.8: Confusion matrices of the test set for the (binarized) CEV data set. Actual class in rows, predicted class in columns.

(a) META - CEV 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	89.24 ±3.95	2.75 ±1.69
C^2	6.61 ±0.81	1.02 ±0.62

(b) MIP - CEV 20 %

Not available

(c) UTADIS - CEV 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	91.96 ±0.88	0.25 ±0.81
C^2	7.69 ±0.43	0.10 ±0.33

(d) META - CEV 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	89.13 ±5.59	2.64 ±1.68
C^2	6.64 ±0.94	1.00 ±0.62

(e) MIP - CEV 50 %

Not available

(f) UTADIS - CEV 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	91.91 ±1.15	0.26 ±0.90
C^2	7.72 ±0.72	0.11 ±0.32

(g) META - CEV 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	89.2 ±2.20	3.28 ±1.92
C^2	6.39 ±1.32	1.14 ±0.79

(h) MIP - CEV 80 %

Not available

(i) UTADIS - CEV 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	91.98 ±1.34	0.08 ±0.26
C^2	7.92 ±1.30	0.03 ±0.12

Table A.9: Confusion matrices of the test set for the (binarized) ASA data set. Actual class in rows, predicted class in columns.

(a) META - ASA 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	30.99 ±0.98	1.43 ±0.92
C^2	0.86 ±0.69	66.72 ±0.92

(b) MIP - ASA 20 %

Not available

(c) UTADIS - ASA 20 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	30.22 ±1.07	2.20 ±1.05
C^2	1.40 ±0.76	66.18 ±0.90

(d) META - ASA 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	31.63 ±1.59	0.82 ±0.55
C^2	0.56 ±0.48	67.00 ±1.61

(e) MIP - ASA 50 %

Not available

(f) UTADIS - ASA 50 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	30.96 ±1.61	1.48 ±0.70
C^2	1.17 ±0.59	66.38 ±1.62

(g) META - ASA 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	31.62 ±3.20	0.69 ±0.72
C^2	0.47 ±0.49	67.22 ±3.30

(h) MIP - ASA 80 %

Not available

(i) UTADIS - ASA 80 %

	$\hat{C}^1$	$\hat{C}^2$
C^1	31.09 ±3.25	1.22 ±0.86
C^2	1.22 ±0.79	66.47 ±3.44

A.2 Confusion matrices of data sets with more than 2 categories

In this section, we give the confusion matrices for all the multi-class data sets that have been used in Chapter 3.

Table A.10: Confusion matrices for the test set of the CPU data set.

(a) META - CPU 20 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	18.95 ±3.43	5.04 ±3.55	0.12 ±0.41	0.00 ±0.00
C^2	4.04 ±2.78	17.71 ±3.76	3.23 ±2.58	0.03 ±0.16
C^3	0.24 ±0.51	5.48 ±3.50	16.93 ±3.54	2.70 ±2.23
C^4	0.08 ±0.31	0.49 ±0.67	3.12 ±2.54	21.85 ±2.69

(b) UTADIS - CPU 20 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	21.95 ±2.14	1.77 ±1.92	0.00 ±0.00	0.30 ±0.96
C^2	2.49 ±2.29	21.10 ±3.23	1.46 ±2.12	0.12 ±0.56
C^3	0.00 ±0.00	2.43 ±2.30	21.59 ±2.63	1.29 ±1.87
C^4	0.04 ±0.20	0.14 ±0.40	3.11 ±2.83	22.20 ±3.02

(c) META - CPU 50 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	21.06 ±3.33	3.27 ±2.32	0.00 ±0.00	0.00 ±0.00
C^2	4.15 ±2.49	17.69 ±3.35	3.07 ±2.47	0.00 ±0.00
C^3	0.07 ±0.24	4.08 ±2.42	19.05 ±3.29	1.98 ±1.75
C^4	0.01 ±0.10	0.52 ±0.89	2.47 ±1.85	22.6 ±3.12

(d) UTADIS - CPU 50 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	22.62 ±2.99	1.32 ±1.30	0.00 ±0.00	0.08 ±0.54
C^2	1.22 ±1.46	23.28 ±2.69	0.60 ±0.95	0.08 ±0.26
C^3	0.00 ±0.00	0.81 ±1.18	23.10 ±3.08	0.90 ±1.27
C^4	0.00 ±0.00	0.02 ±0.19	1.58 ±1.76	24.40 ±3.20

(e) META - CPU 80 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	20.50 ±6.27	3.40 ±3.41	0.00 ±0.00	0.00 ±0.00
C^2	5.00 ±4.07	18.19 ±5.76	2.79 ±3.40	0.00 ±0.00
C^3	0.21 ±0.68	4.45 ±3.28	18.88 ±4.97	1.83 ±2.16
C^4	0.00 ±0.00	0.93 ±1.55	2.14 ±2.31	21.67 ±5.38

(f) UTADIS - CPU 80 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	22.48 ±5.35	1.19 ±1.84	0.00 ±0.00	0.00 ±0.00
C^2	0.62 ±1.29	24.00 ±5.72	0.50 ±1.23	0.00 ±0.00
C^3	0.00 ±0.00	0.57 ±1.13	23.38 ±6.39	0.83 ±1.67
C^4	0.00 ±0.00	0.00 ±0.00	1.17 ±1.90	25.26 ±5.74

Table A.11: Confusion matrices of the test set of the CEV data set.

(a) META - CEV 20 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	61.15 ±3.10	7.30 ±3.28	0.46 ±0.65	1.24 ±0.57
C^2	4.72 ±2.85	12.78 ±3.35	1.30 ±1.75	3.39 ±1.60
C^3	0.90 ±0.57	2.63 ±0.76	0.32 ±0.44	0.09 ±0.18
C^4	0.09 ±0.24	1.78 ±0.81	0.03 ±0.14	1.83 ±0.75

(b) UTADIS - CEV 20 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	63.06 ±1.55	5.73 ±1.69	1.15 ±0.56	0.12 ±0.38
C^2	4.84 ±1.80	13.81 ±2.21	3.22 ±1.55	0.32 ±1.02
C^3	0.70 ±0.46	3.24 ±0.50	0.07 ±0.13	0.00 ±0.02
C^4	0.00 ±0.04	1.77 ±0.80	1.77 ±0.79	0.18 ±0.57

(c) META - CEV 50 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	61.91 ±2.85	6.67 ±2.87	0.33 ±0.61	1.15 ±0.60
C^2	5.33 ±2.04	12.75 ±2.70	0.95 ±1.67	3.23 ±1.55
C^3	1.04 ±0.46	2.66 ±0.71	0.22 ±0.43	0.03 ±0.10
C^4	0.03 ±0.14	1.89 ±0.92	0.07 ±0.34	1.76 ±0.82

(d) UTADIS - CEV 50 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	63.32 ±1.96	5.40 ±1.56	1.17 ±0.58	0.09 ±0.33
C^2	5.18 ±1.64	13.55 ±1.95	3.25 ±1.58	0.28 ±0.86
C^3	0.78 ±0.45	3.18 ±0.56	0.06 ±0.12	0.00 ±0.02
C^4	0.00 ±0.00	1.78 ±0.90	1.79 ±0.83	0.14 ±0.43

(e) META - CEV 80 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	62.74 ±2.89	5.92 ±1.88	0.26 ±0.55	1.14 ±0.75
C^2	5.83 ±1.58	12.71 ±2.77	0.76 ±1.60	3.11 ±1.85
C^3	1.17 ±0.56	2.57 ±0.88	0.17 ±0.39	0.02 ±0.12
C^4	0.00 ±0.03	1.94 ±1.04	0.04 ±0.24	1.62 ±0.99

(f) UTADIS - CEV 80 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	63.04 ±3.00	5.59 ±1.85	1.01 ±0.71	0.03 ±0.15
C^2	5.31 ±2.10	14.03 ±2.65	2.94 ±1.90	0.07 ±0.27
C^3	0.89 ±0.61	3.08 ±0.90	0.10 ±0.23	0.00 ±0.00
C^4	0.00 ±0.00	2.16 ±1.08	1.73 ±1.10	0.02 ±0.10

Table A.12: Confusion matrices of the test set of the LEV data set.

(a) META - LEV 20 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$	$\hat{C}^5$
C^1	4.26 ±1.26	3.71 ±1.40	1.07 ±0.46	0.29 ±0.18	0.02 ±0.06
C^2	3.54 ±2.02	15.31 ±3.06	8.16 ±2.82	0.79 ±0.57	0.25 ±0.24
C^3	1.07 ±0.95	8.71 ±3.01	24.62 ±3.44	4.93 ±2.69	0.93 ±1.01
C^4	0.26 ±0.34	1.22 ±0.76	7.07 ±2.22	8.40 ±2.76	2.73 ±1.80
C^5	0.06 ±0.12	0.13 ±0.14	0.36 ±0.24	1.22 ±0.58	0.88 ±0.43

(b) UTADIS - LEV 20 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$	$\hat{C}^5$
C^1	5.42 ±0.80	2.67 ±0.88	1.08 ±0.27	0.15 ±0.11	0.00 ±0.02
C^2	3.29 ±1.40	16.69 ±1.93	6.94 ±1.67	0.99 ±0.36	0.04 ±0.07
C^3	0.22 ±0.20	7.64 ±1.49	24.94 ±1.85	7.27 ±1.78	0.21 ±0.17
C^4	0.00 ±0.00	0.56 ±0.26	7.43 ±1.54	10.05 ±1.54	1.71 ±1.12
C^5	0.00 ±0.00	0.15 ±0.11	0.44 ±0.17	1.41 ±0.40	0.73 ±0.29

(c) META - LEV 50 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$	$\hat{C}^5$
C^1	4.28 ±1.34	3.80 ±1.63	1.04 ±0.38	0.26 ±0.20	0.01 ±0.04
C^2	3.54 ±1.98	15.74 ±2.69	8.03 ±2.17	0.65 ±0.40	0.21 ±0.19
C^3	0.94 ±0.80	7.81 ±1.90	26.38 ±2.56	4.49 ±2.04	0.53 ±0.71
C^4	0.20 ±0.29	1.12 ±0.54	6.97 ±1.68	9.28 ±1.86	2.05 ±1.33
C^5	0.06 ±0.13	0.13 ±0.14	0.31 ±0.22	1.33 ±0.56	0.83 ±0.39

(d) UTADIS - LEV 50 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$	$\hat{C}^5$
C^1	5.51 ±0.77	2.60 ±0.79	1.00 ±0.35	0.12 ±0.12	0.00 ±0.00
C^2	3.25 ±1.14	16.98 ±1.44	6.68 ±1.30	1.01 ±0.35	0.01 ±0.04
C^3	0.17 ±0.19	7.45 ±1.37	25.02 ±1.60	7.43 ±1.44	0.15 ±0.11
C^4	0.00 ±0.00	0.45 ±0.26	7.44 ±1.49	10.44 ±1.43	1.57 ±1.04
C^5	0.00 ±0.00	0.20 ±0.14	0.44 ±0.22	1.41 ±0.50	0.67 ±0.27

(e) META - LEV 80 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$	$\hat{C}^5$
C^1	3.98 ±1.46	4.01 ±1.82	0.95 ±0.60	0.26 ±0.33	0.00 ±0.00
C^2	3.53 ±2.17	16.65 ±3.42	7.48 ±2.24	0.68 ±0.59	0.26 ±0.35
C^3	0.95 ±0.81	7.78 ±2.05	26.85 ±3.57	4.21 ±2.14	0.63 ±0.88
C^4	0.18 ±0.30	1.12 ±0.69	7.34 ±2.87	8.52 ±2.43	2.00 ±1.56
C^5	0.06 ±0.19	0.11 ±0.22	0.30 ±0.34	1.38 ±0.76	0.84 ±0.68

(f) UTADIS - LEV 80 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$	$\hat{C}^5$
C^1	5.56 ±1.63	2.40 ±1.11	1.00 ±0.64	0.09 ±0.20	0.00 ±0.00
C^2	3.28 ±1.27	16.82 ±2.46	6.73 ±1.83	1.09 ±0.66	0.00 ±0.00
C^3	0.14 ±0.25	7.21 ±1.86	25.40 ±2.85	7.57 ±1.94	0.16 ±0.23
C^4	0.00 ±0.00	0.37 ±0.41	7.19 ±1.80	10.80 ±2.03	1.46 ±0.97
C^5	0.00 ±0.00	0.21 ±0.28	0.40 ±0.44	1.44 ±0.75	0.74 ±0.50

Table A.13: Confusion matrices of the test set of the ASA data set.

(a) META - ASA 20 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	4.93 ±0.73	0.86 ±0.74	0.04 ±0.09	0.00 ±0.00
C^2	0.32 ±0.53	24.67 ±1.30	1.27 ±1.12	0.34 ±0.32
C^3	0.10 ±0.24	0.98 ±0.85	41.11 ±1.57	1.83 ±1.07
C^4	0.00 ±0.01	0.03 ±0.06	1.35 ±0.79	22.18 ±0.89

(b) UTADIS - ASA 20 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	5.02 ±0.78	0.81 ±0.74	0.00 ±0.03	0.00 ±0.00
C^2	0.57 ±0.69	23.57 ±1.18	2.39 ±0.94	0.07 ±0.13
C^3	0.03 ±0.11	1.86 ±0.93	40.06 ±1.26	2.06 ±1.05
C^4	0.03 ±0.20	0.06 ±0.07	1.37 ±0.83	22.1 ±0.85

(c) META - ASA 50 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	5.33 ±0.88	0.49 ±0.48	0.05 ±0.13	0.00 ±0.00
C^2	0.12 ±0.23	25.41 ±1.57	0.76 ±0.77	0.30 ±0.37
C^3	0.03 ±0.09	0.66 ±0.62	41.97 ±1.78	1.30 ±0.84
C^4	0.00 ±0.00	0.05 ±0.10	1.06 ±0.50	22.48 ±1.48

(d) UTADIS - ASA 50 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	5.40 ±0.92	0.46 ±0.44	0.00 ±0.00	0.00 ±0.00
C^2	0.27 ±0.33	24.4 ±1.47	1.89 ±0.70	0.02 ±0.06
C^3	0.00 ±0.00	1.45 ±0.62	40.64 ±1.52	1.89 ±0.79
C^4	0.00 ±0.00	0.05 ±0.09	1.06 ±0.58	22.48 ±1.45

(e) META - ASA 80 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	5.63 ±1.58	0.34 ±0.50	0.03 ±0.12	0.00 ±0.00
C^2	0.17 ±0.38	25.33 ±2.96	0.50 ±0.72	0.31 ±0.50
C^3	0.02 ±0.10	0.67 ±0.67	42.33 ±3.38	1.07 ±1.02
C^4	0.00 ±0.00	0.09 ±0.22	1.13 ±0.72	22.39 ±2.97

(f) UTADIS - ASA 80 %

	$\hat{C}^1$	$\hat{C}^2$	$\hat{C}^3$	$\hat{C}^4$
C^1	5.64 ±1.55	0.36 ±0.48	0.00 ±0.00	0.00 ±0.00
C^2	0.19 ±0.30	24.4 ±2.89	1.71 ±1.10	0.01 ±0.08
C^3	0.00 ±0.00	1.48 ±0.89	40.39 ±3.47	2.21 ±1.16
C^4	0.00 ±0.00	0.01 ±0.06	1.03 ±0.69	22.57 ±2.91

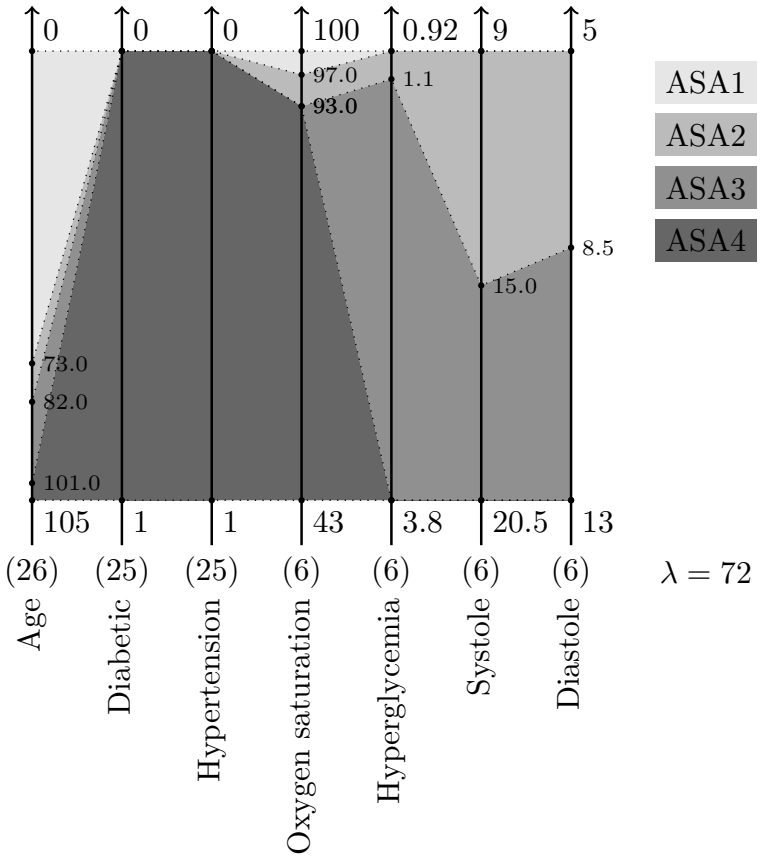
Appendix B

Additional MR-Sort models for ASA classification

In this appendix, we present 10 majority rule sorting (MR-Sort) models learned with the metaheuristic based on the ASA dataset. Each model allows to determine the ASA score of a patient based on his/her performances on the criteria which have an influence on the ASA score.

B.1 MR-Sort model #1

B.1.1 Model parameters



B.1.2 Performances

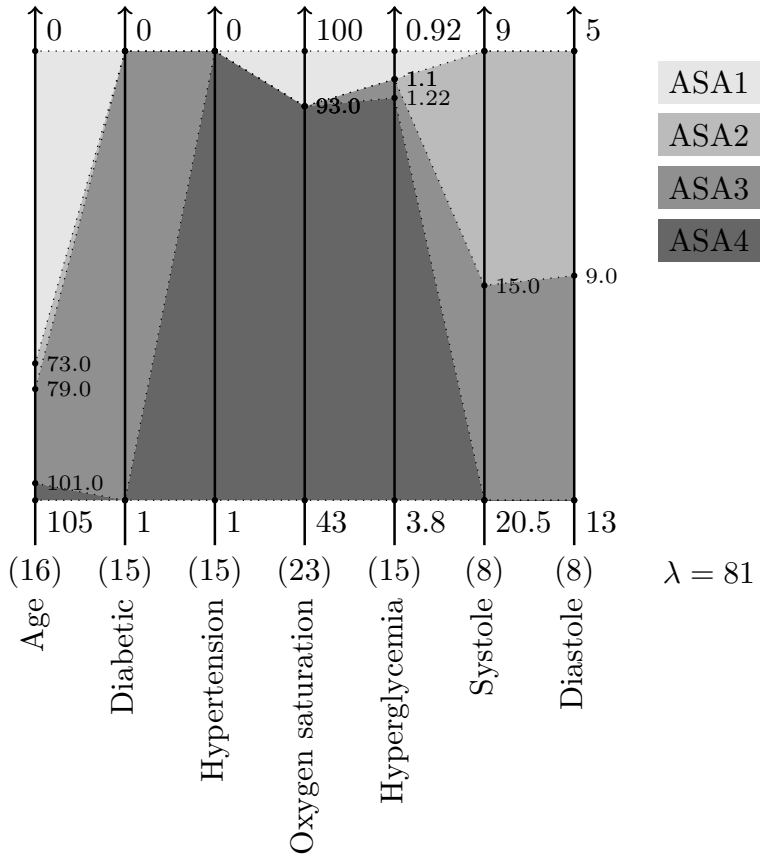
Classification accuracy: 0.9621
Area under the curve: 0.9843

B.1.3 Minimal winning coalitions

```
1 [Age, Diabetic, Oxygen saturation, Hyperglycemia, Systole, Diastole]
2 [Diabetic, Hypertension, Oxygen saturation, Hyperglycemia, Systole, Diastole]
3 [Age, Diabetic, Hypertension]
4 [Age, Hypertension, Oxygen saturation, Hyperglycemia, Systole, Diastole]
```

B.2 MR-Sort model #2

B.2.1 Model parameters



B.2.2 Performances

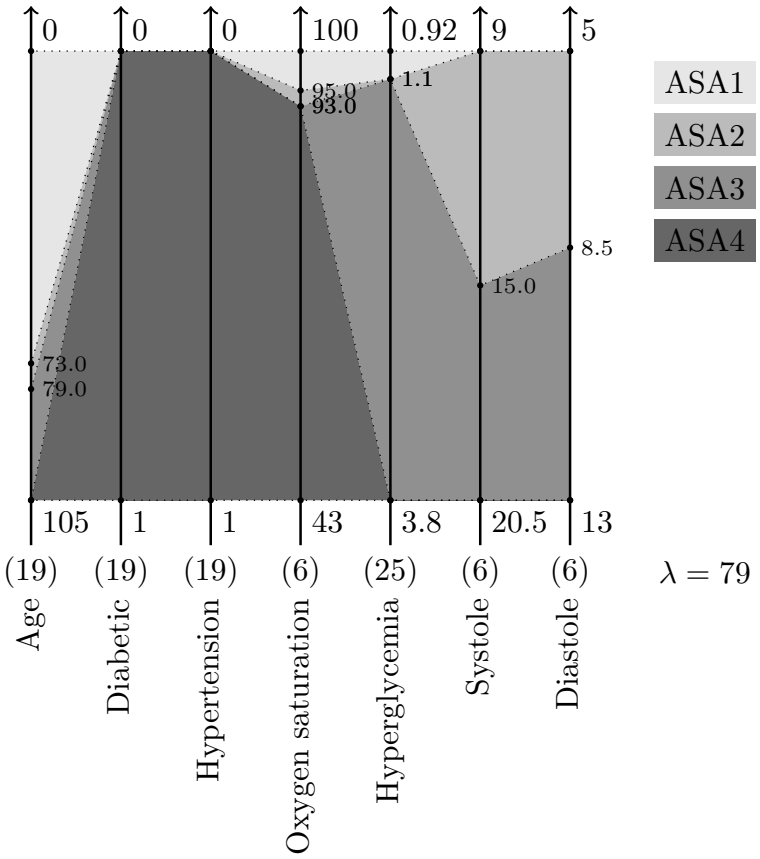
Classification accuracy: 0.9610
Area under the curve: 0.9847

B.2.3 Minimal winning coalitions

```
1 [Age, Diabetic, Hypertension, Oxygen saturation, Systole, Diastole]
2 [Age, Diabetic, Hypertension, Oxygen saturation, Hyperglycemia]
3 [Age, Diabetic, Oxygen saturation, Hyperglycemia, Systole, Diastole]
4 [Diabetic, Hypertension, Oxygen saturation, Hyperglycemia, Systole, Diastole]
5 [Age, Hypertension, Oxygen saturation, Hyperglycemia, Systole, Diastole]
```

B.3 MR-Sort model #3

B.3.1 Model parameters



B.3.2 Performances

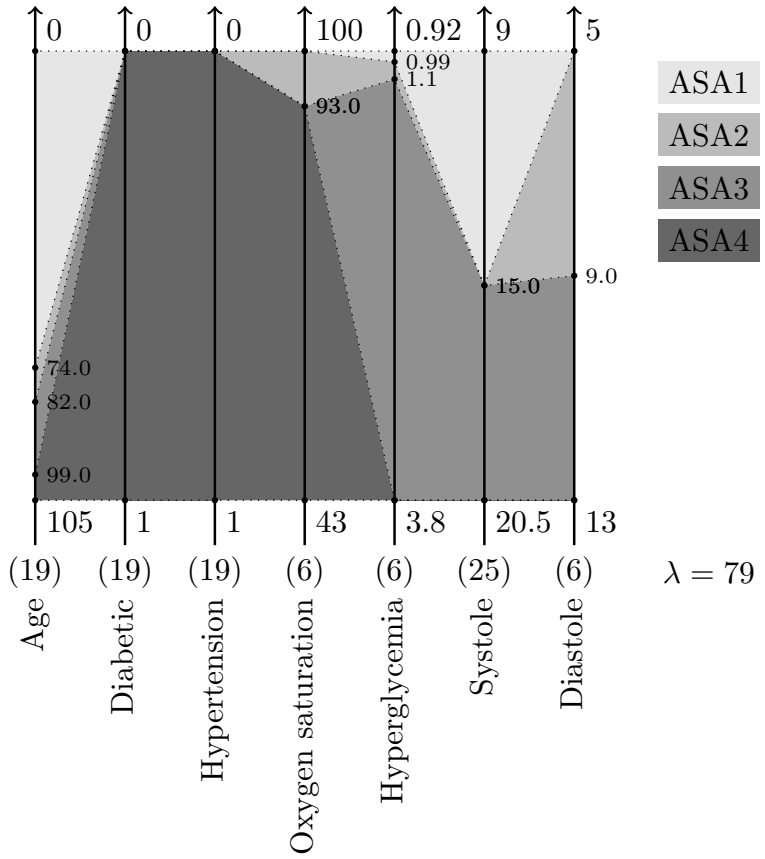
Classification accuracy: 0.9621
Area under the curve: 0.9870

B.3.3 Minimal winning coalitions

```
1 [Age, Diabetic, Hypertension, Hyperglycemia]
2 [Age, Diabetic, Oxygen saturation, Hyperglycemia, Systole, Diastole]
3 [Diabetic, Hypertension, Oxygen saturation, Hyperglycemia, Systole, Diastole]
4 [Age, Hypertension, Oxygen saturation, Hyperglycemia, Systole, Diastole]
```

B.4 MR-Sort model #4

B.4.1 Model parameters



B.4.2 Performances

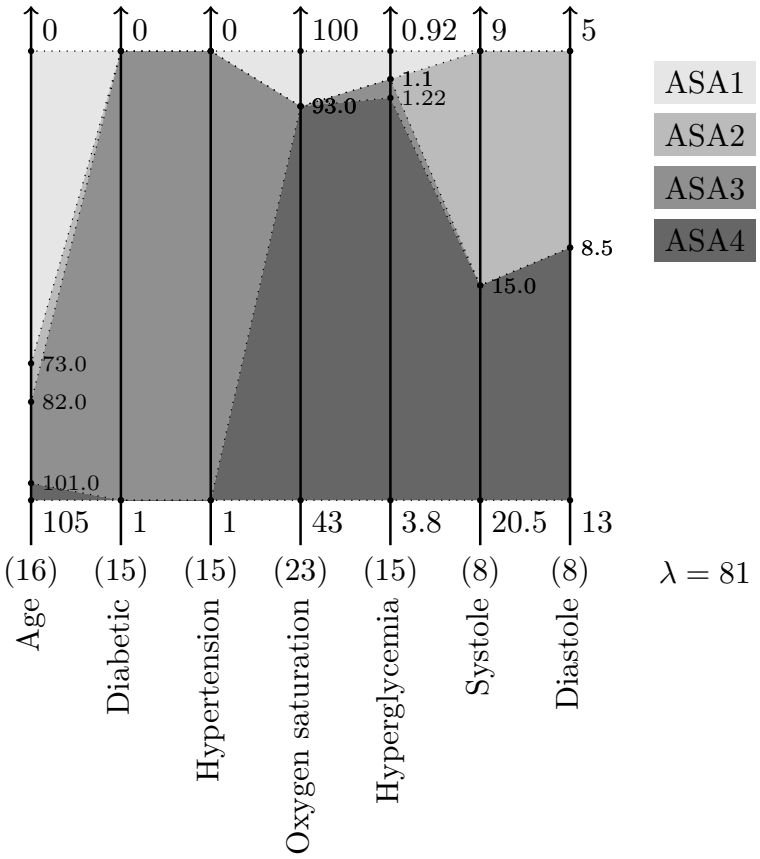
Classification accuracy: 0.9644
Area under the curve: 0.9882

B.4.3 Minimal winning coalitions

```
1 [Age, Diabetic, Hypertension, Systole]
2 [Age, Diabetic, Oxygen saturation, Hyperglycemia, Systole, Diastole]
3 [Diabetic, Hypertension, Oxygen saturation, Hyperglycemia, Systole, Diastole]
4 [Age, Hypertension, Oxygen saturation, Hyperglycemia, Systole, Diastole]
```

B.5 MR-Sort model #5

B.5.1 Model parameters



B.5.2 Performances

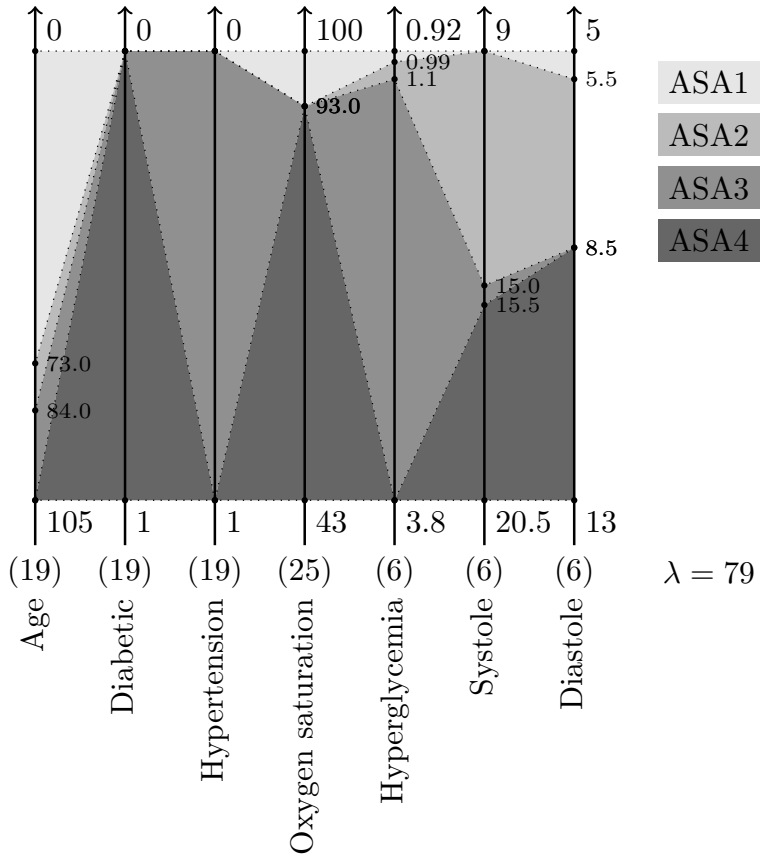
Classification accuracy: 0.9610
Area under the curve: 0.9878

B.5.3 Minimal winning coalitions

```
1 [Age, Diabetic, Hypertension, Oxygen saturation, Systole, Diastole]
2 [Age, Diabetic, Hypertension, Oxygen saturation, Hyperglycemia]
3 [Age, Diabetic, Oxygen saturation, Hyperglycemia, Systole, Diastole]
4 [Diabetic, Hypertension, Oxygen saturation, Hyperglycemia, Systole, Diastole]
5 [Age, Hypertension, Oxygen saturation, Hyperglycemia, Systole, Diastole]
```

B.6 MR-Sort model #6

B.6.1 Model parameters



B.6.2 Performances

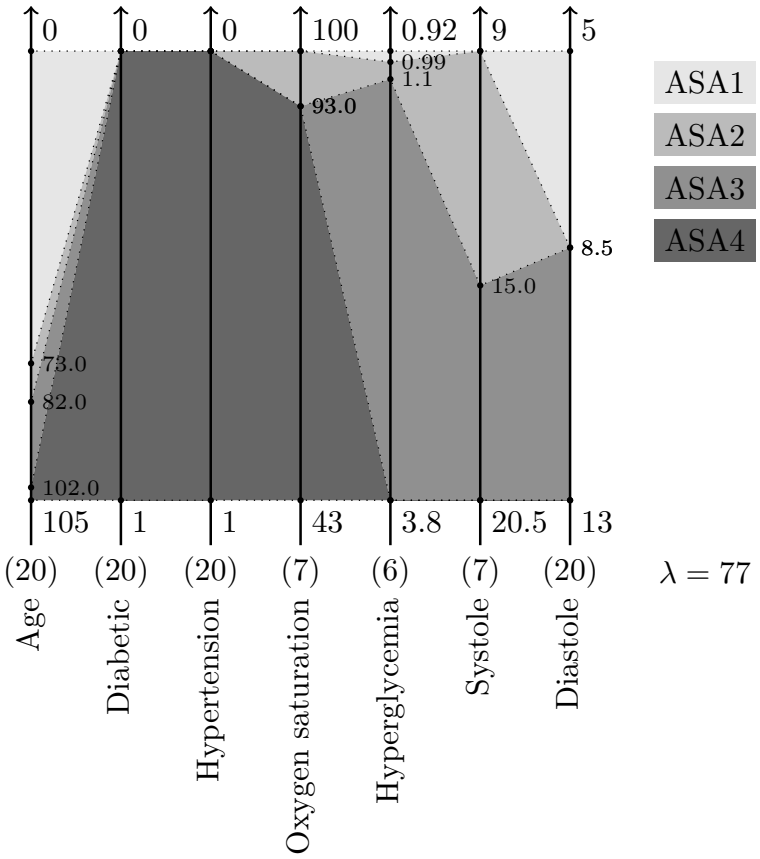
Classification accuracy: 0.9610
Area under the curve: 0.9855

B.6.3 Minimal winning coalitions

```
1  [Age, Diabetic, Hypertension, Oxygen saturation]
2  [Age, Hypertension, Oxygen saturation, Hyperglycemia, Systole, Diastole]
3  [Age, Diabetic, Oxygen saturation, Hyperglycemia, Systole, Diastole]
4  [Diabetic, Hypertension, Oxygen saturation, Hyperglycemia, Systole, Diastole]
```

B.7 MR-Sort model #7

B.7.1 Model parameters



B.7.2 Performances

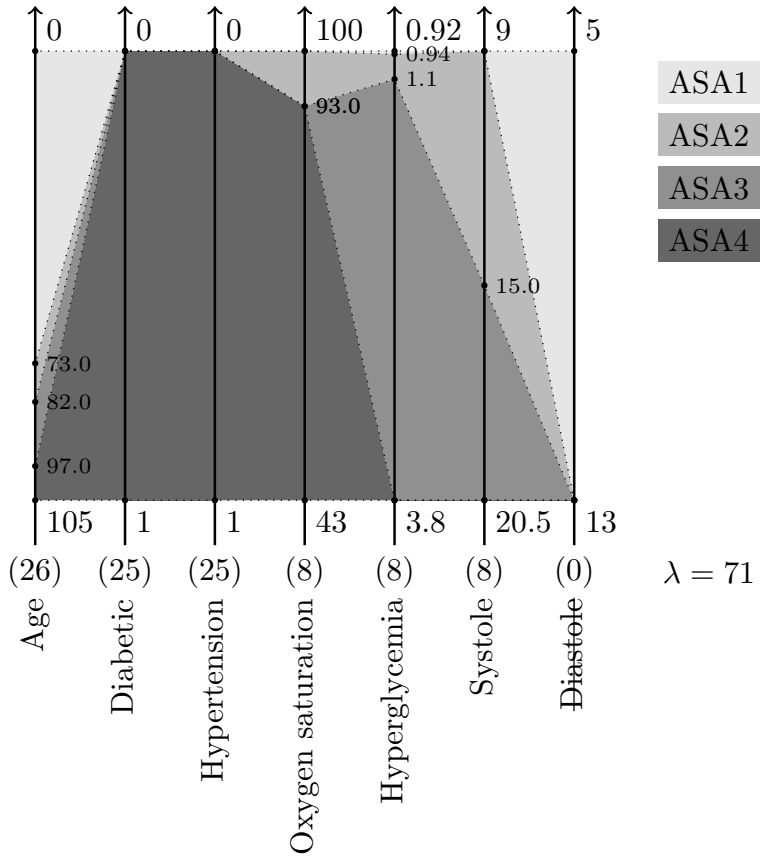
Classification accuracy: 0.9621
Area under the curve: 0.9884

B.7.3 Minimal winning coalitions

```
1 [Diabetic, Hypertension, Oxygen saturation, Hyperglycemia, Systole, Diastole]
2 [Age, Diabetic, Oxygen saturation, Hyperglycemia, Systole, Diastole]
3 [Age, Diabetic, Hypertension, Diastole]
4 [Age, Diabetic, Hypertension, Oxygen saturation, Hyperglycemia, Systole]
5 [Age, Hypertension, Oxygen saturation, Hyperglycemia, Systole, Diastole]
```

B.8 MR-Sort model #8

B.8.1 Model parameters



B.8.2 Performances

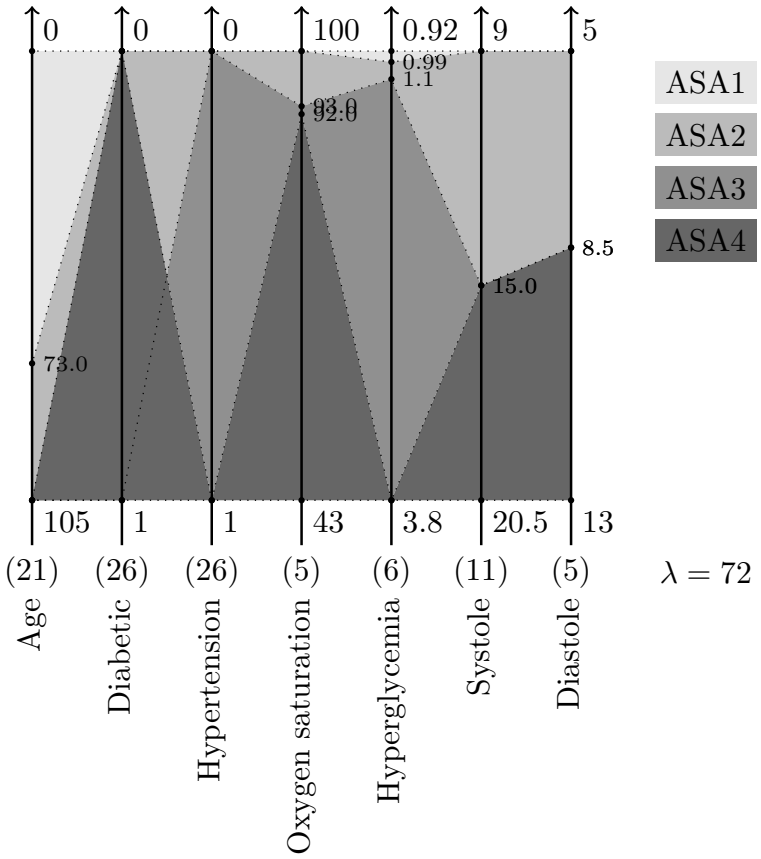
Classification accuracy: 0.9477
Area under the curve: 0.9791

B.8.3 Minimal winning coalitions

```
1  [Age, Hypertension, Oxygen saturation, Hyperglycemia, Systole]
2  [Diabetic, Hypertension, Oxygen saturation, Hyperglycemia, Systole]
3  [Age, Diabetic, Hypertension]
4  [Age, Diabetic, Oxygen saturation, Hyperglycemia, Systole]
```

B.9 MR-Sort model #9

B.9.1 Model parameters



B.9.2 Performances

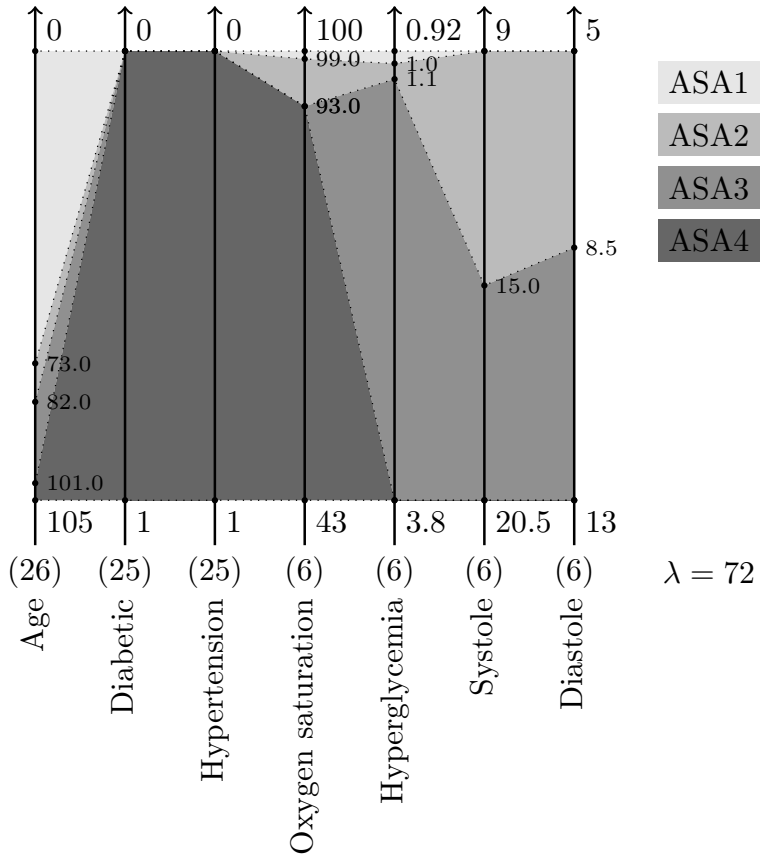
Classification accuracy: 0.9465
Area under the curve: 0.9671

B.9.3 Minimal winning coalitions

```
1 [Diabetic, Hypertension, Oxygen saturation, Hyperglycemia, Systole]
2 [Age, Diabetic, Hypertension]
3 [Diabetic, Hypertension, Oxygen saturation, Systole, Diastole]
4 [Age, Diabetic, Oxygen saturation, Hyperglycemia, Systole, Diastole]
5 [Diabetic, Hypertension, Hyperglycemia, Systole, Diastole]
6 [Age, Hypertension, Oxygen saturation, Hyperglycemia, Systole, Diastole]
```

B.10 MR-Sort model #10

B.10.1 Model parameters



B.10.2 Performances

Classification accuracy: 0.9621
Area under the curve: 0.9851

B.10.3 Minimal winning coalitions

```
1  [Age, Diabetic, Oxygen saturation, Hyperglycemia, Systole, Diastole]
2  [Diabetic, Hypertension, Oxygen saturation, Hyperglycemia, Systole, Diastole]
3  [Age, Diabetic, Hypertension]
4  [Age, Hypertension, Oxygen saturation, Hyperglycemia, Systole, Diastole]
```

Appendix C

List of inequivalent families of monotone sufficient coalitions

C.1 List of inequivalent families of monotone sufficient coalitions for $n = 4$

The families are grouped by type. There are 25 possible types, 29 inequivalent families of minimal sufficient coalition (MSC) (plus the trivial case in which all coalitions are sufficient) and 167 families of MSC (plus the same trivial case). Each inequivalent family in the list is associated the size of its equivalence class. All inequivalent families, except three of them, can be represented by a 1-additive capacity. The three other families can be represented by a 2-additive capacity. They are marked in the last column by $\mathcal{C}_2$.

Table C.1: List of inequivalent MSC of $n = 4$ and number of equivalence for each class. Inequivalent MSC that are not representable by a 1-additive capacity (weighted sum) are pointed out in the last column.

Type	Family of MSC	# eq.	$\mathcal{C}_k$
(0,0,0,0)	{}	1	
(0,0,0,1)	{1234}	1	
(0,0,1,0)	{124}	4	
(0,0,2,0)	{234, 124}	6	
(0,0,3,0)	{134, 123, 124}	4	
(0,0,4,0)	{134, 123, 234, 124}	1	
(0,1,0,0)	{24}	6	
(0,1,1,0)	{14, 123}	12	
(0,1,2,0)	{24, 134, 123}	6	
(0,2,0,0)	{12, 23}	12	
	{23, 14}	3	$\mathcal{C}_2$
(0,2,1,0)	{24, 134, 23}	12	
(0,3,0,0)	{13, 12, 34}	12	$\mathcal{C}_2$
	{24, 12, 14}	4	
	{24, 34, 14}	4	
(0,3,1,0)	{13, 34, 23, 124}	4	
(0,4,0,0)	{24, 12, 13, 34}	3	$\mathcal{C}_2$
	{24, 12, 14, 23}	12	
(0,5,0,0)	{24, 12, 14, 13, 34}	6	
(0,6,0,0)	{24, 12, 14, 34, 23, 13}	1	
(1,0,0,0)	{1}	4	
(1,0,1,0)	{234, 1}	4	
(1,1,0,0)	{14, 2}	12	
(1,2,0,0)	{13, 34, 2}	12	
(1,3,0,0)	{24, 34, 23, 1}	4	
(2,0,0,0)	{4, 3}	6	
(2,1,0,0)	{4, 23, 1}	6	
(3,0,0,0)	{4, 2, 1}	4	
(4,0,0,0)	{4, 2, 3, 1}	1	

C.2 List of inequivalent families of monotone sufficient coalitions for class $\mathcal{C}_2$ for $n = 5$

We list below the 91 inequivalent families of MSC that cannot be represented by a 1-additive capacity. They can all be represented using a 2-additive capacity. The families are grouped by type. Each inequivalent family in the list is associated the size of its equivalence class.

Table C.2: List of inequivalent families of MSC of class $\mathcal{C}_2$ for $n = 5$ and number of equivalence for each class.

Type	Family of MSC	# eq.
(0,0,2,0,0)	{135, 234}	15
(0,0,2,1,0)	{234, 125, 1345}	15
(0,0,3,0,0)	{145, 123, 345}	30
	{235, 234, 125}	60
(0,0,3,1,0)	{134, 135, 2345, 124}	60
(0,0,4,0,0)	{145, 234, 345, 124}	15
	{135, 245, 234, 125}	60
	{235, 145, 135, 123}	60
	{134, 345, 234, 125}	10
(0,0,4,1,0)	{245, 123, 234, 125, 1345}	15
(0,0,5,0,0)	{235, 134, 135, 345, 125}	60
	{235, 134, 135, 245, 124}	12
	{235, 145, 134, 245, 124}	60
	{145, 134, 123, 234, 125}	60
(0,0,6,0,0)	{135, 235, 234, 125, 145, 123}	15
	{135, 345, 234, 125, 245, 123}	10
	{345, 235, 234, 125, 124, 134}	60
	{135, 345, 235, 125, 124, 145}	60
(0,0,7,0,0)	{345, 234, 125, 145, 134, 245, 123}	30
	{135, 235, 125, 124, 145, 134, 245}	60
(0,0,8,0,0)	{135, 345, 234, 125, 124, 145, 245, 123}	15
(0,1,1,0,0)	{123, 45}	10
(0,1,2,0,0)	{15, 123, 345}	60
	{12, 134, 345}	60
(0,1,3,0,0)	{235, 14, 123, 125}	60
	{13, 235, 145, 124}	60
	{235, 14, 123, 245}	60
	{24, 134, 135, 123}	30
(0,1,4,0,0)	{235, 15, 245, 123, 234}	120

Continued on next page

Table C.2 – *Continued from previous page*

Type	Family of MSC	# eq.
	{135, 123, 25, 345, 124}	60
	{235, 34, 145, 125, 124}	60
	{24, 235, 135, 123, 125}	20
(0,1,5,0,0)	{345, 235, 15, 234, 134, 123}	30
	{235, 125, 124, 145, 34, 123}	60
	{24, 135, 345, 235, 125, 123}	60
(0,1,6,0,0)	{24, 135, 345, 235, 145, 134, 123}	60
(0,2,0,0,0)	{34, 15}	15
(0,2,1,0,0)	{12, 35, 234}	60
	{145, 23, 25}	60
(0,2,2,0,0)	{24, 13, 125, 345}	30
	{24, 12, 135, 345}	30
	{134, 23, 35, 124}	60
	{13, 12, 245, 234}	120
	{12, 245, 35, 234}	60
(0,2,3,0,0)	{15, 23, 134, 345, 124}	60
	{45, 134, 135, 234, 25}	120
	{135, 123, 45, 125, 14}	60
	{24, 235, 14, 345, 135}	30
	{24, 34, 135, 123, 125}	60
(0,2,4,0,0)	{135, 235, 14, 234, 123, 45}	60
	{14, 35, 234, 125, 245, 123}	15
	{24, 135, 235, 125, 34, 123}	30
(0,3,0,0,0)	{12, 14, 45}	60
	{12, 34, 45}	30
(0,3,1,0,0)	{24, 145, 23, 25}	60
	{34, 14, 35, 125}	60
	{34, 245, 23, 14}	120
	{34, 14, 123, 25}	60
(0,3,2,0,0)	{15, 14, 123, 25, 345}	60
	{24, 12, 134, 35, 145}	30
	{13, 23, 245, 125, 14}	120
	{15, 45, 123, 234, 25}	60
(0,3,3,0,0)	{24, 135, 145, 134, 23, 25}	20
	{12, 35, 234, 145, 13, 245}	60
(0,4,0,0,0)	{34, 15, 14, 35}	15
	{24, 15, 23, 25}	60
	{24, 34, 15, 23}	10

Continued on next page

Table C.2 – *Continued from previous page*

Type	Family of MSC	# eq.
	{24, 34, 15, 35}	60
(0,4,1,0,0)	{13, 34, 35, 25, 145}	60
	{24, 13, 15, 25, 345}	60
	{13, 15, 23, 25, 345}	30
	{34, 14, 45, 125, 23}	60
(0,4,2,0,0)	{24, 12, 35, 145, 134, 23}	60
	{24, 35, 145, 34, 25, 123}	15
(0,5,0,0,0)	{24, 13, 15, 23, 14}	60
	{24, 12, 15, 35, 25}	60
	{24, 12, 15, 35, 34}	12
	{12, 15, 34, 25, 45}	60
(0,5,1,0,0)	{135, 12, 14, 34, 23, 25}	60
	{15, 35, 124, 23, 13, 45}	60
(0,6,0,0,0)	{24, 12, 23, 25, 13, 45}	15
	{24, 12, 35, 34, 25, 13}	10
	{24, 12, 34, 23, 13, 45}	60
	{15, 14, 34, 23, 25, 45}	60
(0,6,1,0,0)	{24, 12, 35, 145, 34, 25, 13}	10
(0,7,0,0,0)	{12, 14, 34, 23, 25, 13, 45}	30
	{24, 12, 15, 14, 35, 34, 45}	60
(0,8,0,0,0)	{24, 12, 15, 34, 23, 25, 13, 45}	15
(1,2,0,0,0)	{34, 15, 2}	15
(1,3,0,0,0)	{24, 15, 3, 25}	60
(1,4,0,0,0)	{13, 2, 14, 35, 45}	15

Appendix D

Example of a semi-definite program

We consider a ranking problem involving 2 criteria x and y and three alternatives, a^1, a^2 and a^3. The performances of these alternatives are given in Table D.1. The criterion values vary between 0 and 10.

Table D.1: Performances of alternatives a^1, a^2 and a^3 on criteria x and y.

	x	y
a^1	10	7
a^2	6	8
a^3	7	5

A decision maker states that the following ranking holds: $a^1 \succ a^2 \succ a^3$. We use the objective and the set of constraints given in Equation (2.27) in order to find a model restoring this ranking. We use semi-definite programming to learn polynomial marginal value functions. We denote by u_1^* and u_2^* the polynomial functions associated respectively with criteria 1 and 2. The degree of the polynomials of the marginal value functions is fixed to 3.

To ensure the monotonicity of functions u_1^* and u_2^*, we impose the non-negativity of their derivative. Formally, we define u_1^* and u_2^* as follows:

$$u_1^*(x) = p_{x,0} + p_{x,1} \cdot x + p_{x,2} \cdot x^2 + p_{x,3} \cdot x^3,$$
$$u_2^*(y) = p_{y,0} + p_{y,1} \cdot y + p_{y,2} \cdot y^2 + p_{y,3} \cdot y^3.$$

The derivative of $u_1^*(x)$ and $u_2^*(y)$ are respectively equal to:

$$\frac{du_1^*}{dx} = p_{x,1} + 2p_{x,2} \cdot x + 3p_{x,3} \cdot x^2 \qquad \text{and} \qquad \frac{du_2^*}{dy} = p_{y,1} + 2p_{y,2} \cdot y + 3p_{y,3} \cdot y^2.$$

The monotonicity of a polynomial marginal is ensured if its derivative is a sum of squares. Formally, it reads:

$$\begin{aligned}
\frac{du_1^*}{dx} &= \overline{x}^\mathsf{T} Q \overline{x} \\
&= \begin{pmatrix} 1 \\ x \end{pmatrix}^\mathsf{T} \begin{pmatrix} q_{0,0} & q_{0,1} \\ q_{1,0} & q_{1,1} \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} \\
&= q_{0,0} + (q_{0,1} + q_{0,1})\, x + q_{1,1} x^2, \\
\frac{du_2^*}{dy} &= \overline{y}^\mathsf{T} R \overline{y} \\
&= r_{0,0} + (r_{0,1} + r_{1,0})\, y + r_{1,1} y^2.
\end{aligned}$$

To ensure the non-negativity of the derivative, we impose the matrices Q and R to be semi-definite positive in conjunction with the following constraints:

$$\begin{cases} p_{x,1} &= q_{0,0}, \\ 2p_{x,2} &= q_{0,1} + q_{1,0}, \\ 3p_{x,3} &= q_{1,1}, \end{cases} \quad \text{and} \quad \begin{cases} p_{y,1} &= r_{0,0}, \\ 2p_{y,2} &= r_{0,1} + r_{1,0}, \\ 3p_{y,3} &= r_{1,1}. \end{cases}$$

The values of a^1, a^2 and a^3 read:

$$\begin{aligned}
U(a^1) &= p_{x,0} + 10p_{x,1} + 100p_{x,2} + 1000p_{x,3} + p_{y,0} + 7p_{y,1} + 49p_{y,2} \\
&\quad + 343p_{y,3}, \\
U(a^2) &= p_{x,0} + 6p_{x,1} + 36p_{x,2} + 324p_{x,3} + p_{y,0} + 8p_{y,1} + 64p_{y,2} \\
&\quad + 512p_{y,3}, \\
U(a^3) &= p_{x,0} + 7p_{x,1} + 49p_{x,2} + 343p_{x,3} + p_{y,0} + 5p_{y,1} + 25p_{y,2} \\
&\quad + 125p_{y,3}.
\end{aligned}$$

To find a model reflecting the ranking given as input, i.e. $a^1 \succ a^2 \succ a^3$, we have to fulfil two conditions: $a^1 \succ a^2$ and $a^2 \succ a^3$. This is done by adding the following constraints:

$$\begin{cases} U(a^1) - U(a^2) + \sigma^+(a^1) - \sigma^-(a^1) - \sigma^+(a^2) + \sigma^-(a^2) &> 0, \\ U(a^2) - U(a^3) + \sigma^+(a^2) - \sigma^-(a^2) - \sigma^+(a^1) + \sigma^-(a^1) &> 0. \end{cases}$$

After substituting $U(a^1)$, $U(a^2)$ and $U(a^3)$ by their value we obtain the two following constraints:

$$\begin{cases} 4p_{x,1} + 64p_{x,2} + 776p_{x,3} - p_{y,1} - 15p_{y,2} - 231p_{y,3} \\ \qquad + \sigma^+(a^1) - \sigma^-(a^1) - \sigma^+(a^2) + \sigma^-(a^2) &> 0, \\ -p_{x,1} - 13p_{x,2} - 19p_{x,3} + 3p_{y,1} + 39p_{y,2} + 387p_{y,3} \\ \qquad + \sigma^+(a^2) - \sigma^-(a^2) - \sigma^+(a^3) + \sigma^-(a^3) &> 0. \end{cases}$$

Given that criteria domains are comprised between 0 and 10, the following constraints hold:

$$\begin{cases} p_{x,0} &= 0, \\ p_{y,0} &= 0, \\ 10p_{x,1} + 100p_{x,2} + 1000p_{x,3} + 10p_{y,1} + 100p_{y,2} + 1000p_{y,3} &= 1. \end{cases}$$

Finally, by assembling the objective function and the constraints, we obtain the following semi-definite program:

$$\min \sigma^+(a^1) + \sigma^-(a^1) + \sigma^+(a^2) + \sigma^-(a^2) + \sigma^+(a^3) - \sigma^-(a^3)$$

such that:

$$\begin{cases} 4p_{x,1} + 64p_{x,2} + 776p_{x,3} - p_{y,1} - 15p_{y,2} - 231p_{y,3} & \\ \quad +\sigma^+(a^1) - \sigma^-(a^1) - \sigma^+(a^2) + \sigma^-(a^2) &> 0, \\ -p_{x,1} - 13p_{x,2} - 19p_{x,3} + 3p_{y,1} + 39p_{y,2} + 387p_{y,3} & \\ \quad +\sigma^+(a^2) - \sigma^-(a^2) - \sigma^+(a^3) + \sigma^-(a^3) &> 0, \\ p_{x,0} &= 0, \\ p_{y,0} &= 0, \\ 10p_{x,1} + 100p_{x,2} + 1000p_{x,3} + 10p_{y,1} + 100p_{y,2} + 1000p_{y,3} &= 1, \\ p_{x,1} &= q_{0,0}, \\ 2p_{x,2} &= q_{0,1} + q_{1,0}, \\ 3p_{x,3} &= q_{1,1}, \\ p_{y,1} &= r_{0,0}, \\ 2p_{y,2} &= r_{0,1} + r_{1,0}, \\ 3p_{y,3} &= r_{1,1}, \end{cases}$$

with:

$$\begin{cases} Q, R & PSD, \\ \sigma^+(a^1), \sigma^-(a^1), \sigma^+(a^2), \sigma^-(a^2), \sigma^+(a^3), \sigma^-(a^3), & \geq 0. \end{cases}$$

Appendix E

Cholesky factorization

The factorization of Cholesky consists in decomposing a positive semi-definite matrix M into the product of a lower triangular matrix L and its transpose L^T. Formally it reads:

$$M = LL^\mathsf{T}. \tag{E.1}$$

The decomposition works as follows. For a matrix a of size $d \times d$, Equation (E.1) reads:

$$\begin{aligned} M &= \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} & \cdots & m_{1,d} \\ m_{2,1} & m_{2,2} & m_{2,3} & \cdots & m_{2,d} \\ m_{3,1} & m_{3,2} & m_{3,3} & \cdots & m_{3,d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{d,1} & m_{d,2} & m_{d,3} & \cdots & m_{d,d} \end{pmatrix} \\ &= \begin{pmatrix} l_{1,1} & 0 & 0 & \cdots & 0 \\ l_{2,1} & l_{2,2} & 0 & \cdots & 0 \\ l_{3,1} & l_{3,2} & l_{3,3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{d,1} & l_{d,2} & l_{d,3} & \vdots & l_{d,d} \end{pmatrix} \cdot \begin{pmatrix} l_{1,1} & l_{2,1} & l_{3,1} & \cdots & l_{d,1} \\ 0 & l_{2,2} & l_{3,2} & \cdots & l_{d,2} \\ 0 & 0 & l_{3,3} & \cdots & l_{d,3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & l_{d,d} \end{pmatrix}. \end{aligned}$$

The multiplication of the two matrices leads to:

$$M = \begin{pmatrix} l_{1,1}^2 & & & \text{(symmetric)} \\ l_{2,1}l_{1,1} & l_{2,1}^2 + l_{2,2}^2 & & \\ \vdots & \vdots & \ddots & \\ l_{1,1}l_{d,1} & l_{2,1}l_{d,1} + l_{2,2}l_{d,2} & \cdots & \sum_{i=1}^{d} l_{d,i}^2 \end{pmatrix}.$$

The value $m_{i,i}$ and $m_{i,j}$ can be expressed as follows:

$$m_{i,i} = \sum_{k=1}^{i} l_{i,k}^2 \qquad \text{and} \qquad m_{i,j} = \sum_{k=1}^{j} l_{i,k} l_{j,k}$$

The value of the variables $l_{i,i}$ and $l_{i,j}$ are then given by

$$l_{i,i} = \sqrt{m_{i,i} - \sum_{k=1}^{i-1} l_{i,k}^2} \qquad \text{and} \qquad l_{i,j} = \frac{1}{m_{i,i}} \left(m_{i,j} - \sum_{k=1}^{j-1} l_{i,k} l_{j,k} \right)$$

Appendix F

UTA-Splines: detailed results of the experiments

Figure F.1 and F.2 show the average Spearman distance and Kendall Tau of the test set after running the experiment described in Section 7.4 with UTA-poly.

Figure F.3 shows the average Spearman distance and Kendall Tau obtained with UTA-splines for the four models composed of 3 criteria presented in Figure 7.7. The learned models are composed of polynomials of the third degree which are continuous up to the second derivative at the connection points. The number of pieces per value function varies between 1 and 5. Similarly, Figure F.4 shows the average Spearman distance and Kendall Tau obtained with the four models composed of 5 criteria (Figure 7.8).

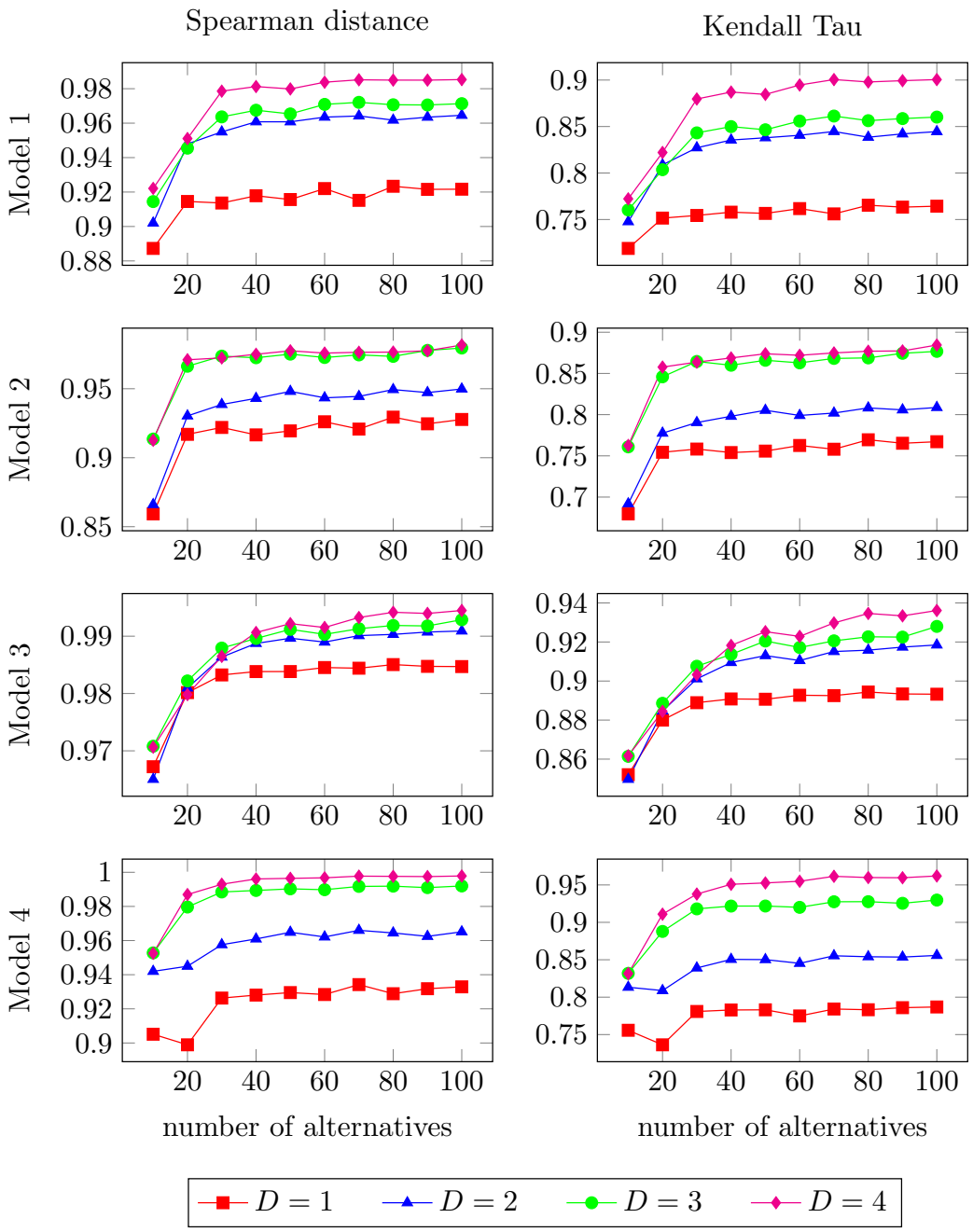


Figure F.1: Average Spearman distance and Kendall Tau of the test set of models 1 to 4 learned by UTA-poly when the degree of the marginals varies between 1 and 4.

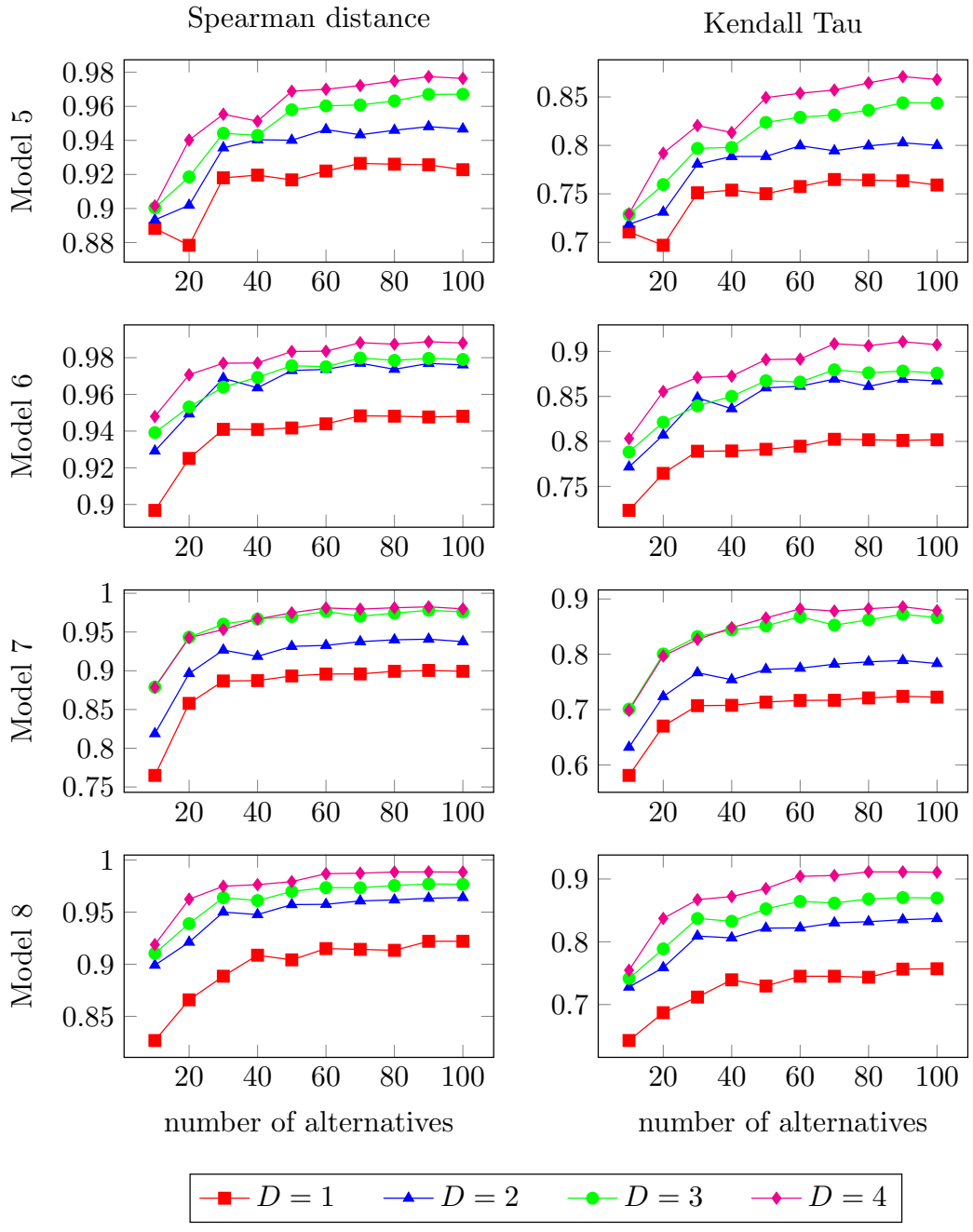


Figure F.2: Average Spearman distance and Kendall Tau of the test set of models 5 to 8 learned by UTA-poly when the degree of the marginals varies between 1 and 4.

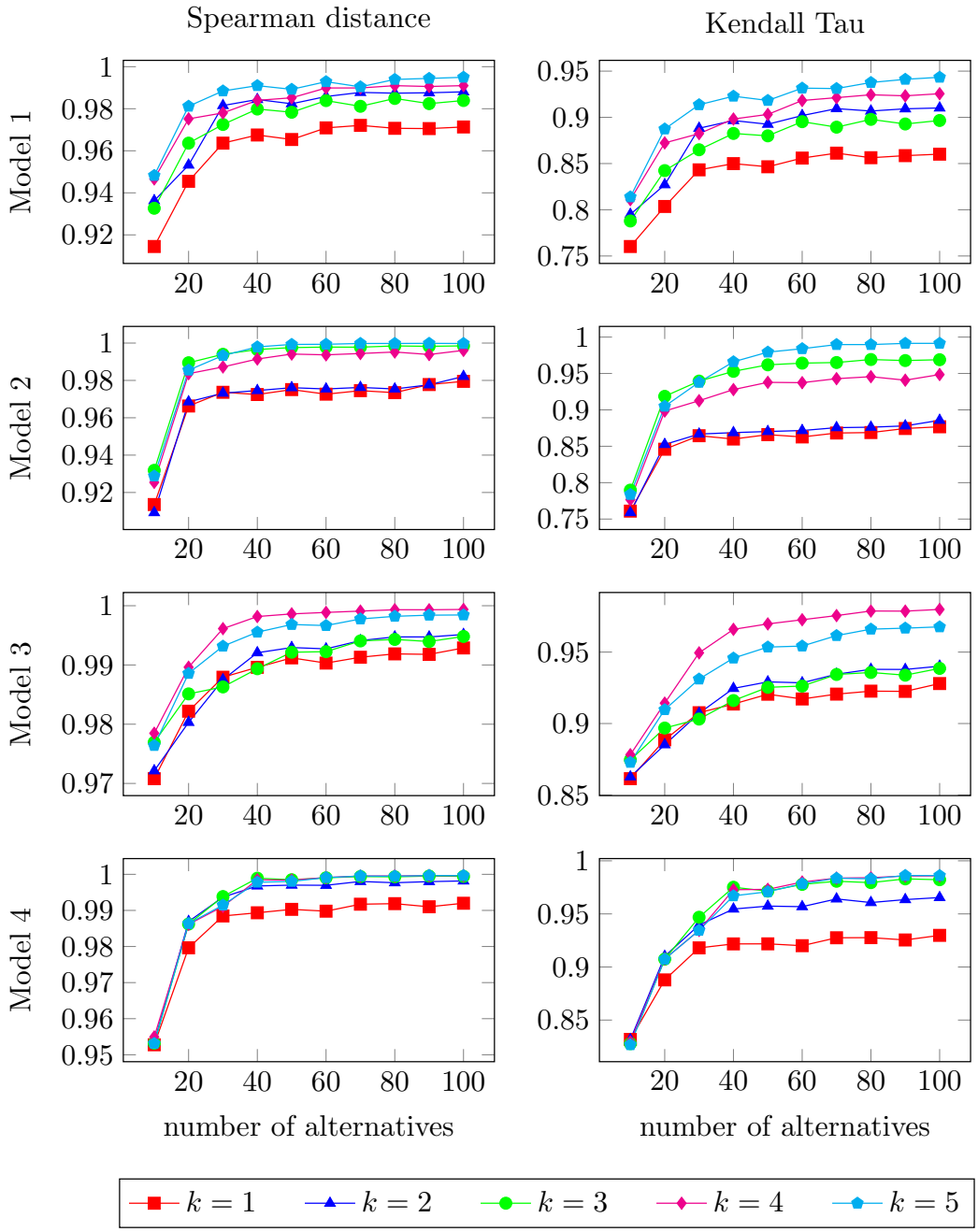


Figure F.3: Average Spearman distance and Kendall Tau of the test set of models 1 to 4 learned by UTA-splines with marginals composed of polynomials of the third degree. The continuity at the breakpoints is ensured up to the second derivative.

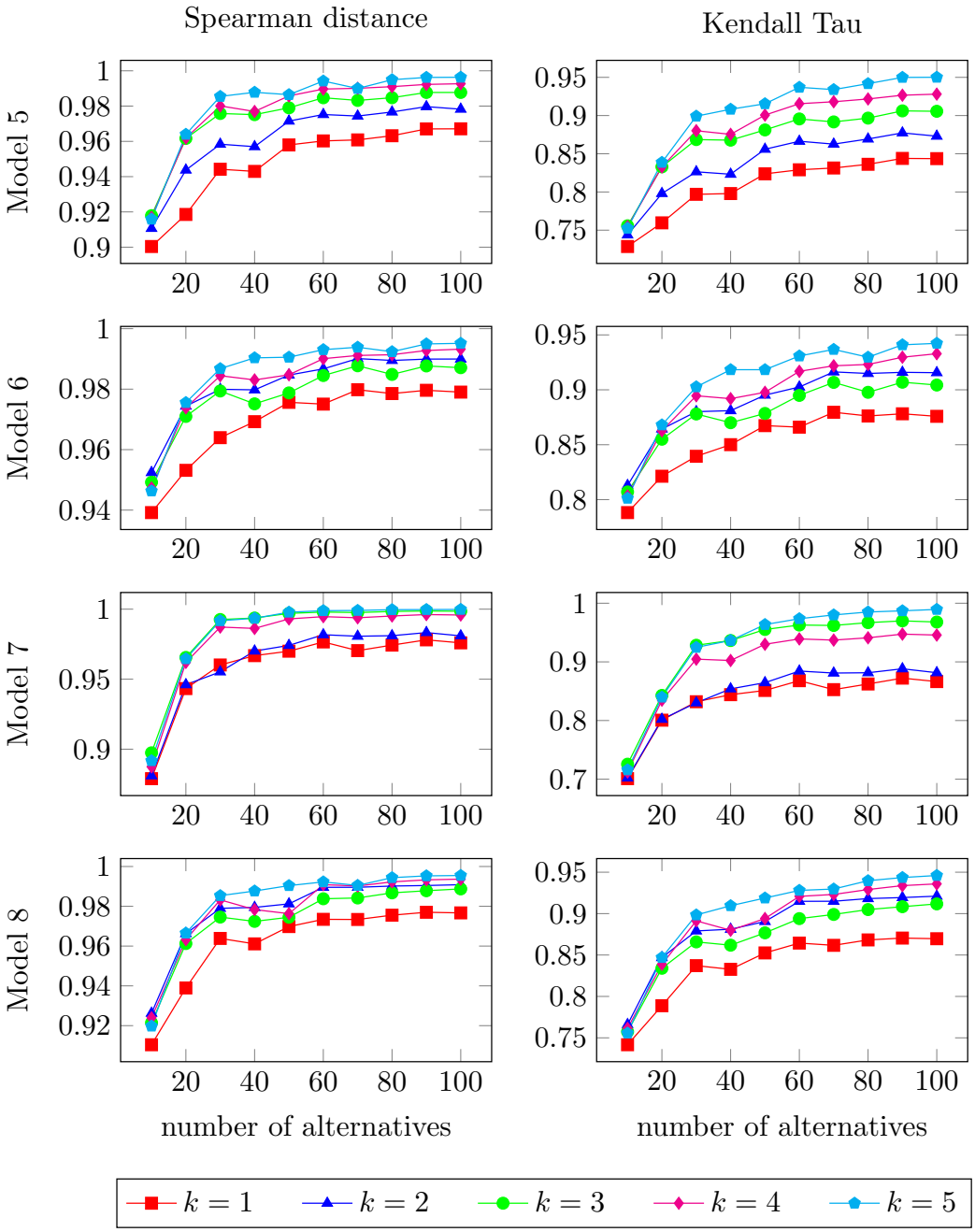


Figure F.4: Average Spearman distance and Kendall Tau of the test set of models 5 to 8 learned by UTA-splines with marginals composed of polynomials of the third degree. The continuity at the breakpoints is ensured up to the second derivative.

Appendix G

List of contributions

We list in this appendix the list of our contributions. It includes articles in scientific journals, conference proceedings and talks.

G.1 Articles

G.1.1 Published

- O. Sobrie, M. Pirlot, and F. Joerin. Intégration de la méthode d'aide à la décision ELECTRE TRI dans un système d'Information Géographique Open Source. *Revue Internationale de Géomatique*, 23(1):13–38, 2013e

- O. Sobrie and M. Pirlot. Implementation of ELECTRE TRI in an Open Source GIS. *EWG/MCDA Newsletter*, pages 15–18, 2012

G.1.2 Submitted

- O. Sobrie, N. Gillis, V. Mousseau, and M. Pirlot. UTA-poly and UTA-splines: additive value functions with polynomial marginals. Submitted, 2016a

- O. Sobrie, M. E. A. Lazouni, S. Mahmoudi, V. Mousseau, and M. Pirlot. A new decision support model for preanesthetic evaluation. Submitted, 2016b

- E. Ersek Uyanık, O. Sobrie, V. Mousseau, and M. Pirlot. Families of sufficient coalitions of criteria involved in ordered classification procedures. Submitted, 2016

G.2 Refereed conference proceedings

- O. Sobrie, V. Mousseau, and M. Pirlot. Learning the parameters of a non compensatory sorting model. In T. Walsh, editor, *Algorithmic Decision Theory*, Lecture Notes in Artificial Intelligence, pages 153–170, Lexington, KY, USA, 2015b. Springer

- E. Ersek Uyanık, O. Sobrie, V. Mousseau, and M. Pirlot. Listing the families of sufficient coalitions of criteria involved in sorting procedures. In *DA2PL 2014 Workshop From Multiple Criteria Decision Aid to Preference Learning*, pages 60–70, 2014. Paris, France

- O. Sobrie, V. Mousseau, and M. Pirlot. Learning the parameters of a majority rule sorting model taking attribute interactions into account. In *DA2PL 2014 Workshop From Multiple Criteria Decision Aid to Preference Learning*, pages 22–30, 2014d. Paris, France

- O. Sobrie, V. Mousseau, and M. Pirlot. Learning a majority rule model from large sets of assignment examples. In P. Perny, M. Pirlot, and A. Tsoukiás, editors, *Algorithmic Decision Theory*, Lecture Notes in Artificial Intelligence, pages 336–350, Brussels, Belgium, 2013d. Springer

- O. Sobrie, V. Mousseau, and M. Pirlot. Learning the parameters of a multiple criteria sorting method from large sets of assignment examples. In *DA2PL 2012 Workshop From Multiple Criteria Decision Aid to Preference Learning*, pages 21–31, 2012. Mons, Belgique

G.3 Talks

- O. Sobrie, V. Mousseau, and M. Pirlot. Using polynomial marginal utility functions in UTADIS. In *27th European Conference on Operational Research*, Glasgow, Scotland, July 2015a

- O. Sobrie, V. Mousseau, and M. Pirlot. New veto rules for sorting models. In *20th Conference of the International Federation of Operational Research Societies*, Barcelona, Spain, July 2014c

- O. Sobrie, V. Mousseau, and M. Pirlot. Learning the parameters of a multiple criteria sorting method from large sets of assignment examples. In *Séminaire "Modélisation des préférences et aide multicritère à la décision", Lamsade*, Paris, France, April 2014b

- O. Sobrie, V. Mousseau, and M. Pirlot. Learning the parameters of a multiple criteria sorting method from large sets of assignment examples. In

28th annual conference of the Belgian Operational Research Society, Mons, Belgium, January 2014a

- O. Sobrie, V. Mousseau, and M. Pirlot. Learning the parameters of a multiple criteria sorting method from large sets of assignment examples. In *26th European Conference on Operational Research*, Roma, Italy, July 2013c

- O. Sobrie, V. Mousseau, and M. Pirlot. Learning the parameters of a multiple criteria sorting method from large sets of assignment examples. In *22nd International Conference on Multiple Criteria Decision Making*, Malaga, Spain, June 2013b

- O. Sobrie, V. Mousseau, and M. Pirlot. Learning the parameters of a multiple criteria sorting method from large sets of assignment examples. In *77th meeting of the EWG on MCDA*, Rouen, France, April 2013a

Bibliography

J. Almeida-Dias, J. R. Figueira, and Bernard Roy. ELECTRE TRI-C: A multiple criteria sorting method based on characteristic reference actions. *European Journal of Operational Research*, 204(3):565–580, 2011.

J. Almeida-Dias, J. R. Figueira, and B. Roy. A multiple criteria sorting method where each category is characterized by several reference actions: The ELECTRE TRI-nC method. *European Journal of Operational Research*, 217(3):567–579, 2012.

S. Angilella, S. Greco, and B. Matarazzo. Non-additive robust ordinal regression: A multiple criteria decision model based on the choquet integral. *European Journal of Operational Research*, 201(1):277–288, 2010.

K. Bache and M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

C. A. Bana e Costa and J.-C. Vansnick. MACBETH, An interactive path towards the construction of cardinal value functions. *International Transactions in Operational Research*, 1(4):489–500, 1994.

C. A. Bana e Costa, J.-M. De Corte, and J.-C. Vansnick. On the mathematical foundations of MACBETH. In S. Greco, M. Ehrgott, and J. R. Figueira, editors, *Multiple Criteria Decision Analysis: state of the art surveys*, International Series in Operations Research and Management Science, pages 409–437. Springer, 2005.

N. Belacel. Multicriteria assignment method PROAFTN: methodology and medical application. *European Journal of Operational Research*, 125(1):175–183, 2000.

N. D. Belnap. *Modern Uses of Multiple-Valued Logic*, volume 2, chapter A Useful Four-Valued Logic, pages 5–37. Springer, Dordrecht, 1977.

G. Blekherman. There are significantly more nonegative polynomials than sums of squares. *Israel Journal of Mathematics*, 153(1):355–380, 2006.

G. Bous, Ph. Fortemps, F. Glineur, and M. Pirlot. ACUTA: A novel method for eliciting additive value functions on the basis of holistic preference statements. *European Journal of Operational Research*, 206(2):435–444, 2010.

D. Bouyssou. Some remarks on the notion of compensation in MCDM. *European Journal of Operational Research*, 26:150–160, 1986.

D. Bouyssou. Outranking methods. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*, pages 2887–2893. Springer, 2009.

D. Bouyssou and 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, 2007a.

D. Bouyssou and 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, 2007b.

D. Bouyssou and T. Marchant. On the relations between ELECTRE TRI-B and ELECTRE TRI-C and on a new variant of ELECTRE TRI-B. *European Journal of Operational Research*, 242(1):201–211, 2015.

D. Bouyssou and M. Pirlot. A characterization of concordance relations. *European Journal of Operational Research*, 167(2):427–443, 2005.

D. Bouyssou and M. Pirlot. Further results on concordance relations. *European Journal of Operational Research*, 181:505–514, 2007.

D. Bouyssou and M. Pirlot. An axiomatic analysis of concordance-discordance relations. *European Journal of Operational Research*, 199:468–477, 2009.

J. P. Brans and Ph. Vincke. A preference ranking organization method. *Management Science*, 31(6):647–656, 1985.

J.-P. Brans, B. Mareschal, and Ph. Vincke. PROMETHEE: a new family of outranking methods in multicriteria analysis. In J.-P. Brans, editor, *Operational Research*, pages 477–490. North Holland, Amsterdam, 1984.

J. Butler, J. Jia, and J. S. Dyer. Simulation techniques for the sensitivity analysis of multi-criteria decision models. *European Journal of Operational Research*, 103(3):531–546, December 1997.

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

K. J. Carter, N. P. Ritchey, F. Castro, L. P. Caccamo, E. Kessler, and B. A. Erickson. Analysis of three decision-making methods: a breast cancer patient as a model. *Medical Decision Making*, 19(1):49–57, 1999.

N. Caspard, B. Leclerc, and B. Monjardet. *Finite Ordered Sets : Concepts, results and uses*. Encyclopedia of Mathematics and its applications. Cambridge University Press, 2012.

R. Chandrasekaran, Young U. Ryu, Varghese S. Jacob, and S. Hong. Isotonic separation. *INFORMS J. on Computing*, 17(4):462–474, October 2005.

R. L. Chatburn and F. P. Primiano. Decision analysis for large capital purchases: how to buy a ventilator. *Respiratory Care*, 46(10):1038–1053, 2001.

A. Chateauneuf and J.-Y. Jaffray. Derivation of some results on monotone capacities by Möbius inversion. In B. Bouchon-Meunier and R. R. Yager, editors, *Uncertainty in Knowledge-Based Systems, International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU '86, Paris, France, June 30 - July 4, 1986, Selected and Extended Contributions*, volume 286 of *Lecture Notes in Computer Science*, pages 95–102. Springer, 1986.

W. Cheng, J. Hühn, and E. Hüllermeier. Decision tree and instance-based learning for label ranking. In L. Bottou and M. Littman, editors, *Proceedings of the 26th International Conference on Machine Learning (ICML-09)*, pages 161–168, Montreal, Canada, June 2009. Omnipress.

W. Cheng, K. Dembczyński, and E. Hüllermeier. Label ranking methods based on the Plackett-Luce model. In Fürnkranz and Joachims (2010), pages 215–222.

W. Cheng, K. Dembczyński, and E. Hüllermeier. Graded multilabel classification: The ordinal case. In Fürnkranz and Joachims (2010), pages 223–230.

M. Cinelli, S. R. Coles, and K. Kirwan. Analysis of the potentials of multi criteria decision analysis methods to conduct sustainability assessment. *Ecological Indicators*, 46(0):138–148, 2014.

S. Corrente, S. Greco, M. Kadziński, and R. Słowiński. Robust ordinal regression in preference learning and ranking. *Machine Learning*, 93(2–3):381–422, 2013.

D. R. Cox. The regression analysis of binary sequences (with discussion). *J Roy Stat Soc B*, 20:215–242, 1958.

Y. Crama and P. L. Hammer. *Boolean Functions - Theory, Algorithms, and Applications*, volume 142 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2011.

S. Damart, L. C. Dias, and V. Mousseau. Supporting groups in sorting decisions: Methodology and use of a multi-criteria aggregation/disaggregation DSS. *Decision Support Systems*, 43(4):1464–1475, 2007.

H. Daniels and B. Kamp. Application of MLP networks to bond rating and house pricing. *Neural Computing & Applications*, 8(3):226–234, 1999.

A. De Montis, P. De Toro, B. Droste-Franke, I. Omann, and S. Stagl. Assessing the quality of different MCDA methods. In M. Getzner, C. Spash, and S. Stagl, editors, *Alternatives for Environmental Valuation*, pages 99–133. Routledge, 2005.

Y. De Smet, P. Nemery, and R. Selvaraj. An exact algorithm for the multicriteria ordered clustering problem. *Omega*, 40(6):861 – 869, 2012. Special Issue on Forecasting in Management Science.

J. M. Devaud, G. Groussaud, and E. Jacquet-Lagrèze. UTADIS: Une méthode de construction de fonctions d'utilité additives rendant compte de jugements globaux. In *European working group on MCDA, Bochum, Germany*, 1980.

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

L. Dias, V. Mousseau, J. R. Figueira, and J. Clímaco. An aggregation/disaggregation approach to obtain robust conclusions with ELECTRE TRI. *European Journal of Operational Research*, 138(1):332–348, 2002.

A. Donati, M. Ruzzi, E. Adrario, P. Pelaia, F. Coluzzi, V. Gabbanelli, and P. Pietropaoli. A new and feasible model for predicting operative risk. *British Journal of Anaesthesia*, 93(3):393–399, 2004.

M. Doumpos and C. Zopounidis. *Multicriteria Decision Aid Classification Methods*. Kluwer Academic Publishers, 2002.

M. Doumpos, Y. Marinakis, M. Marinaki, and C. Zopounidis. 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.

W. Duivesteijn and A. Feelders. Nearest neighbour classification with monotonicity constraints. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, ECML PKDD '08, pages 301–316, Berlin, Heidelberg, 2008. Springer.

E. Ersek Uyanık, O. Sobrie, V. Mousseau, and M. Pirlot. Listing the families of sufficient coalitions of criteria involved in sorting procedures. In *DA2PL 2014 Workshop From Multiple Criteria Decision Aid to Preference Learning*, pages 60–70, 2014. Paris, France.

E. Ersek Uyanık, O. Sobrie, V. Mousseau, and M. Pirlot. Families of sufficient coalitions of criteria involved in ordered classification procedures. Submitted, 2016.

D. Faraggi and B. Reiser. Estimation of the area under the roc curve. *Statistics in Medicine*, 21(20):3093–3106, 2002.

T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27 (8):861–874, 2006.

J. Figueira and B. Roy. Determining the weights of criteria in the ELECTRE type methods with a revised Simos' procedure. *European Journal of Operational Research*, 139(2):317–326, 2002. EURO XVI: O.R. for Innovation and Quality of Life.

J. R. Figueira, S. Greco, and R. Słowiński. Building a set of additive value functions representing a reference preorder and intensities of preference: GRIP method. *European Journal of Operational Research*, 195(2):460–486, 2009.

J. R. Figueira, J. A. Dias, S. Matias, B. Roy, M.J. Carvalho, and C. E. Plancha. ELECTRE TRI-C, a multiple criteria decision aiding sorting model applied to assisted reproduction. *International Journal of Medical Informatics*, 80(4): 262–273, 2011.

P. C. Fishburn. Noncompensatory preferences. *Synthese*, 33(1):393–403, 1976.

Ph. Fortemps and R. Słowiński. A graded quadrivalent logic for ordinal preference modelling: Loyola–like approach. *Fuzzy Optimization and Decision Making*, 1 (1):93–111, 2002.

J. Fürnkranz and E. Hüllermeier. Preference learning: An introduction. In J. Fürnkranz and E. Hüllermeier, editors, *Preference Learning*, pages 1–17. Springer, 2010.

J. Fürnkranz and T. Joachims, editors. *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, 2010. Omnipress.

S. Giove, S. Greco, B. Matarazzo, and R. Słowiński. Variable consistency monotonic decision trees. In J. J. Alpigini, J. F. Peters, J. Skowronek, and N. Zhong, editors, *Rough Sets and Current Trends in Computing, Third International Conference, RSCTC 2002, Malvern, PA, USA, October 14-16, 2002, Proceedings*, volume 2475 of *Lecture Notes in Computer Science*, pages 247–254. Springer, 2002.

L. G. Glance, S. J. Lustik, E. L. Hannan, T. M. Osler, D. B. Mukamel, F. Qian, and A. W. Dick. The surgical mortality probability model: derivation and validation of a simple risk prediction rule for noncardiac surgery. *Annals of surgery*, 255(4):696–702, 2012.

D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, december 1992.

C. Gonzales, P. Perny, and J. P. Dubus. Decision making with multiple objectives using GAI networks. *Artificial Intelligence*, 175(7–8):1153–1179, 2011.

C. S. Goodman and R. Ahn. Methodological approaches of health technology assessment. *International Journal of Medical Informatics*, 56(1–3):97–105, 1999.

M. Grabisch. The application of fuzzy integrals in multicriteria decision making. *European Journal of Operational Research*, 89(3):445–456, 1996.

M. Grabisch and C. Labreuche. A decade of application of the choquet and Sugeno integrals in multi-criteria decision aid. *Annals of Operations Research*, 175(1):247–286, 2010.

M. Grabisch and M. Roubens. Application of the Choquet integral in multicriteria decision making. In *Fuzzy measures and integrals*, pages 348–374. Physica Verlag, 2000.

M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, march 2014.

S. Greco, B. Matarazzo, and R. Słowiński. Rough sets theory for multicriteria decision analysis. *European Journal of Operational Research*, 129:1–47, 2001.

S. Greco, V. Mousseau, and R. Słowiński. Ordinal regression revisited: Multiple criteria ranking using a set of additive value functions. *European Journal of Operational Research*, 191(2):416–436, 2008.

S. Greco, V. Mousseau, and R. Słowiński. Multiple criteria sorting with a set of additive value functions. *European Journal of Operational Research*, 207(3): 1455–1470, 2010a.

S. Greco, R. Słowiński, J.R. Figueira, and V. Mousseau. Robust ordinal regression. In M. Ehrgott, J. R. Figueira, and S. Greco, editors, *Trends in Multiple Criteria Decision Analysis*, volume 142 of *International Series in Operations Research and Management Science*, pages 241–283. Springer, 2010b.

S. Greco, M. Kadziński, V. Mousseau, and R. Słowiński. ELECTRE-GKMS: Robust ordinal regression for outranking methods. *European Journal of Operational Research*, 214(1):118–135, 2011.

M. Gurrieri. *Benchmarks, Reduction Techniques and Rule-based Learning for Label Ranking*. PhD thesis, Université de Mons, 2015.

D. Henrion and J.-B. Lasserre. GloptiPoly: Global optimization over polynomials with Matlab and SeDuMi. *ACM Transactions on Mathematical Software*, 29 (2):165–194, 2003.

D. Henrion, J.-B. Lasserre, and J. Löfberg. GloptiPoly 3: moments, optimization and semidefinite programming. *Optimization Methods & Software*, 24(4–5): 761–779, 2009.

E. Hüllermeier. Preference learning: Machine learning meets MCDA. In *DA2PL 2014 Workshop From Multiple Criteria Decision Aid to Preference Learning*, pages 1–2, 2014. Paris, France.

E. Hüllermeier and A. F. Tehrani. Efficient learning of classifiers based on the 2-additive Choquet integral. In C. Moewes and A. Nürnberger, editors, *Computational Intelligence in Intelligent Data Analysis*, volume 445 of *Studies in Computational Intelligence*, pages 17–29. Springer, 2013.

E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker. Label ranking by learning pairwise preferences. *Aritificial Intelligence*, 172(16–17):1897–1916, 2008.

J. Hussmann and R. C. Russel. *Memorix: Surgery*. Chapman & Hall, 1997.

E. Jacquet-Lagrèze and Y. Siskos. Assessing a set of additive utility functions for multicriteria decision making: the UTA method. *European Journal of Operational Research*, 10:151–164, 1982.

E. Jacquet-Lagrèze and Y. Siskos. Preference disaggregation: 20 years of MCDA experience. *European Journal of Operational Research*, 130(2):233–245, 2001.

S. Karpagavalli, K. Jamuna, and M. Vijaya. Machine learning approach for preoperative anaesthetic risk prediction. *International Journal of Recent Trends in Engineering*, 1(2), 2009.

R. L. Keeney and H. Raiffa. *Decisions with multiple objectives: Preferences and value tradeoffs*. John Wiley & Sons, 1976.

M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

M. Köksalan, V. Mousseau, Ö. Özpeynirci, and S. B. Özpeynirci. A new outranking-based approach for assigning alternatives to ordered classes. *Naval Research Logistics*, pages 74–85, 2009.

S. Kurz and N. Tautenhahn. On Dedekind´s problem for complete simple games. *International Journal of Game Theory*, 42(2):411–437, 2013.

O. I. Larichev and H. M. Moshkovich. The method ORCLASS for ordinal classification of multiattribute alternatives. In *Verbal Decision Analysis for Unstructured Problems*, volume 17 of *Theory and Decision Library*, pages 189–237. Springer, 1997.

J.-B. Lasserre. *Moments, positive polynomials and their applications*, volume 1. World Scientific, 2009.

M. E. A. Lazouni, M. A. Chikh, and S. Mahmoudi. A new computer aided diagnosis system for pre-anesthesia consultation. *Journal of Medical Imaging and Health Informatics*, 3(4):471–479, 2013a.

M. E. A. Lazouni, M. El Habib Daho, N. Settouti, M. A. Chikh, and S. Mahmoudi. Machine learning tool for automatic ASA detection. In A. Amine, O. Aït Mohamed, and L. Bellatreche, editors, *Modeling Approaches and Algorithms for Advanced Computer Applications*, volume 488 of *Studies in Computational Intelligence*, pages 9–16. Springer, 2013b.

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

M. J. Liberatore and R. L. Nydick. The analytic hierarchy process in medical and health care decision making: A literature review. *European Journal of Operational Research*, 189(1):194–207, 2008.

J. Liu. *Preference elicitation for multi-criteria ranking with multiple reference points*. PhD thesis, Université Paris-Saclay, CentraleSupélec, 2016.

P. Lutz. The medical algorithms project, ch31. anaesthesiology, section: Preoperative patient classification and preparation, 2008.

N. Maciej. Preference and veto thresholds in multicriteria analysis based on stochastic dominance. *European Journal of Operational Research*, 158(2):339–350, 2004.

J.-L. Marichal, P. Meyer, and M. Roubens. Sorting multi-attribute alternatives: The TOMASO method. *Computers & OR*, 32(4):861–877, 2005.

R. Massaglia and A. Ostanello. N-tomic: a support system for multicriteria segmentation problems. In P. Korhonen, A. Lewandowski, and J. Wallenius, editors, *Multiple Criteria Decision Support*, pages 167–174. Springer, 1991.

A. K. Menon and C. Elkan. Dyadic prediction using a latent feature log-linear model. *Computing Research Repository*, abs/1006.2156, 2010.

S. A. Metchebon Takougang. *Contributions à l'aide à la décision en matière de gestion spatialisée. Étude de cas en management environnemental et développement de nouveaux outils*. PhD thesis, Université de Mons - Faculté Polytechnique, 2010.

P. Meyer and A.-L. Olteanu. Formalizing and solving the problem of clustering in MCDA. *European Journal of Operational Research*, 227(3):494–502, 2013.

M. Minoux. *Programmation mathématique: théorie et algorithmes*. Collection technique et scientifique des télécommunications. Dunod, 2008. 2nd edition.

J. Moscarola and B. Roy. Procédure automatique d'examen de dossiers fondée sur une segmentation trichotomique en présence de critères multiples. *RAIRO Recherche Opérationnelle*, 11(2):145–173, 1977.

H. Moshkovich, A. Mechitov, and D. Olson. Verbal decision analysis. In *Multiple Criteria Decision Analysis: State of the Art Surveys*, volume 78 of *International Series in Operations Research and Management Science*, pages 609–633. Springer, 2005.

V. Mousseau and L. Dias. Valued outranking relations in ELECTRE providing manageable disaggregation procedures. *European Journal of Operational Research*, 156(1):467–482, 2003.

V. Mousseau and R. Słowiński. Inferring an ELECTRE TRI model from assignment examples. *Journal of Global Optimization*, 12(1):157–174, 1998.

V. Mousseau, J. R. Figueira, and J.-Ph. Naux. Using assignment examples to infer weights for ELECTRE TRI method: Some experimental results. *European Journal of Operational Research*, 130(1):263–275, 2001.

V. Mousseau, J. R. Figueira, J. Dias, C. G. da Silva, and J. Clímaco. Resolving inconsistencies among constraints on the parameters of an MCDA model. *European Journal of Operational Research*, 147(1):72–93, 2003.

V. Mousseau, L. C. Dias, and J. R. Figueira. Dealing with inconsistent judgments in multiple criteria sorting models. *4OR*, 4(2):145–158, 2006.

K. G. Murty and S. N. Kabadi. Some NP-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39(2):117–129, 1987.

P. Nemery and C. Lamboray. FlowSort: a flow-based sorting method with limiting or central profiles. *TOP*, 16(1):90–113, 2008.

A. Ngo The and V. Mousseau. Using assignment examples to infer category limits for the ELECTRE TRI method. *Journal of Multi-criteria Decision Analysis*, 11(1):29–43, 2002.

M. F. Norese and V. Carbone. An application of ELECTRE TRI to support innovation. *Multicriteria Decision Analysis*, 21:77–93, 2014.

W. Ouerdane. Multiple criteria decision aiding: a dialectival perspective. *4OR*, 9(4):429–432, 2011.

V. M. Ozernoy. Choosing the best multiple criteria decision-making method. *INFOR*, 30(2):159–171, 1992.

V.M. Ozernoy. A framework for choosing the most appropriate discrete alternative multiple criteria decision-making method in decision support systems and expert systems. In Y. Sawaragi, K. Inoue, and H. Nakayama, editors, *Toward interactive and intelligent decision support systems: proceedings of the seventh international conference on multiple criteria decision making*, volume 2 of *Lecture notes in economics and mathematical systems*, pages 56–64, Kyoto, Japan, 1987. Springer.

P. A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 96(2):293–320, 2003.

Z. Pawlak. Rough sets. *International Journal of Computer and Information Sciences*, 11(5):341–356, 1982.

P. Perny. Multicriteria filtering methods based on concordance/non-discordance principles. *Annals of Operations Research*, 80:137–167, 1998.

P. Perny and B. Roy. The use of fuzzy outranking relations in preference modelling. *Fuzzy Sets and Systems*, 49(1):33–53, July 1992.

M. Pirlot. General local search methods. *European Journal of Operational Research*, 92(3):493–511, 1996.

R. Potharst and J. C. Bioch. A decision tree algorithm for ordinal classification. In D. J. Hand, J. N. Kok, and M. R. Berthold, editors, *Advances in Intelligent Data Analysis*, volume 1642 of *Lecture Notes in Computer Science*, pages 187–198. Springer, 1999.

R. Potharst and A. J. Feelders. Classification trees for problems with monotonicity constraints. *SIGKDD Explor. Newsl.*, 4(1):1–10, June 2002.

M. Rogers and M. Bruen. Choosing realistic values of indifference, preference and veto thresholds for use with environmental criteria within ELECTRE. *European Journal of Operational Research*, 107(3):542–551, 1998.

B. Roy. Classement et choix en présence de points de vue multiples: La méthode ELECTRE. *Revue Francaise d'Informatique et de Recherche Opérationnelle*, 8:57–75, 1968.

B. Roy. *Méthodologie multicritère d'aide à la décision*. Economica, 1985.

B. Roy. The outranking approach and the foundations of ELECTRE methods. *Theory and Decision*, 31:49–73, 1991.

B. Roy. *Multicriteria methodology for decision aiding*. Springer, 1996.

B. Roy. A missing link in OR-DA: Robustness analysis. *Foundations of Computing and Decision Sciences*, 23(3):141–160, 1998.

B. Roy. Robustness in operational research and decision aiding: A multi-faceted issue. *European Journal of Operational Research*, 200(3):629–638, 2010.

B. Roy and P. Bertier. La méthode ELECTRE II - une application au média-planning. In Ross M., editor, *OR'72*, pages 291–302. North-Holland Publishing Company, 1973.

B. Roy and D. Bouyssou. *Aide multicritère à la décision: méthodes et cas*. Economica Paris, 1993.

B. Roy and R. Słowiński. Handling effects of reinforced preference and counter-veto in credibility of outranking. *European Journal of Operational Research*, 188(1):185–190, 2008.

E. Silberberg and W. C. Suen. *The structure of economics*. McGraw-Hill, Boston, Mass., 3rd edition, 2001.

J. Simos. L'évaluation environnementale: Un processus cognitif négocié. Master's thesis, DGF-EPFL, Lausanne, 1990.

J. Simos and L. Y. Maystre. *Évaluer l'impact sur l'environnement. Une approche originale par l'analyse multicritère et la négotiation*. Presses Polytechniques et Universitaires Romandes, 1990.

E. Siskos and N. Tsotsolas. Elicitation of criteria importance weights through the Simos method: a robustness concern. *European Journal of Operational Research*, 246(2):543–553, 2015.

Y. Siskos and D. Yanacopoulos. UTASTAR - an ordinal regression method for building additive value functions. *Investigação Operacional*, 5:39–53, 1985.

R. Słowiński, S. Greco, and V. Mousseau. Multi-criteria ranking of a finite set of alternatives using ordinal regression and additive utility functions - a new UTA-GMS method. In *Practical Approaches to Multi-Objective Optimization*, 2005.

R. Słowiński, M. Kadziński, and S. Greco. Robust ordinal regression for dominance-based rough set approach under uncertainty. In M. Kryszkiewicz, C. Cornelis, D. Ciucci, J. Medina-Moreno, H. Motoda, and Z. Raś, editors, *Rough Sets and Intelligent Systems Paradigms*, volume 8537 of *Lecture Notes in Computer Science*, pages 77–87. Springer, 2014.

O. Sobrie and M. Pirlot. Implementation of ELECTRE TRI in an Open Source GIS. *EWG/MCDA Newsletter*, pages 15–18, 2012.

O. Sobrie, V. Mousseau, and M. Pirlot. Learning the parameters of a multiple criteria sorting method from large sets of assignment examples. In *DA2PL 2012 Workshop From Multiple Criteria Decision Aid to Preference Learning*, pages 21–31, 2012. Mons, Belgique.

O. Sobrie, V. Mousseau, and M. Pirlot. Learning the parameters of a multiple criteria sorting method from large sets of assignment examples. In *77th meeting of the EWG on MCDA*, Rouen, France, April 2013a.

O. Sobrie, V. Mousseau, and M. Pirlot. Learning the parameters of a multiple criteria sorting method from large sets of assignment examples. In *22nd International Conference on Multiple Criteria Decision Making*, Malaga, Spain, June 2013b.

O. Sobrie, V. Mousseau, and M. Pirlot. Learning the parameters of a multiple criteria sorting method from large sets of assignment examples. In *26th European Conference on Operational Research*, Roma, Italy, July 2013c.

O. Sobrie, V. Mousseau, and M. Pirlot. Learning a majority rule model from large sets of assignment examples. In P. Perny, M. Pirlot, and A. Tsoukiás, editors, *Algorithmic Decision Theory*, Lecture Notes in Artificial Intelligence, pages 336–350, Brussels, Belgium, 2013d. Springer.

O. Sobrie, M. Pirlot, and F. Joerin. Intégration de la méthode d'aide à la décision ELECTRE TRI dans un système d'Information Géographique Open Source. *Revue Internationale de Géomatique*, 23(1):13–38, 2013e.

O. Sobrie, V. Mousseau, and M. Pirlot. Learning the parameters of a multiple criteria sorting method from large sets of assignment examples. In *28th annual conference of the Belgian Operational Research Society*, Mons, Belgium, January 2014a.

O. Sobrie, V. Mousseau, and M. Pirlot. Learning the parameters of a multiple criteria sorting method from large sets of assignment examples. In *Séminaire "Modélisation des préférences et aide multicritère à la décision", Lamsade*, Paris, France, April 2014b.

O. Sobrie, V. Mousseau, and M. Pirlot. New veto rules for sorting models. In *20th Conference of the International Federation of Operational Research Societies*, Barcelona, Spain, July 2014c.

O. Sobrie, V. Mousseau, and M. Pirlot. Learning the parameters of a majority rule sorting model taking attribute interactions into account. In *DA2PL 2014 Workshop From Multiple Criteria Decision Aid to Preference Learning*, pages 22–30, 2014d. Paris, France.

O. Sobrie, V. Mousseau, and M. Pirlot. Using polynomial marginal utility functions in UTADIS. In *27th European Conference on Operational Research*, Glasgow, Scotland, July 2015a.

O. Sobrie, V. Mousseau, and M. Pirlot. Learning the parameters of a non compensatory sorting model. In T. Walsh, editor, *Algorithmic Decision Theory*, Lecture Notes in Artificial Intelligence, pages 153–170, Lexington, KY, USA, 2015b. Springer.

O. Sobrie, N. Gillis, V. Mousseau, and M. Pirlot. UTA-poly and UTA-splines: additive value functions with polynomial marginals. Submitted, 2016a.

O. Sobrie, M. E. A. Lazouni, S. Mahmoudi, V. Mousseau, and M. Pirlot. A new decision support model for preanesthetic evaluation. Submitted, 2016b.

C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904.

T. Stephen and T. Yusun. Counting inequivalent monotone boolean functions. *Discrete Applied Mathematics*, 167(0):15–24, 2014.

T. J. Stewart. Dealing with uncertainties in MCDA. In *Multiple Criteria Decision Analysis: State of the Art Surveys*, volume 78 of *International Series in Operations Research & Management Science*, pages 445–466. Springer, 2005.

A. F. Tehrani, W. Cheng, and E. Hüllermeier. Choquistic regression: Generalizing logistic regression using the choquet integral. In S. Galichet, J. Montero, and G. Mauris, editors, *Proceedings of the 7th conference of the European Society for Fuzzy Logic and Technology, EUSFLAT 2011, Aix-Les-Bains, France, July 18-22, 2011*, pages 868–875. Atlantis Press, 2011.

A. F. Tehrani, W. Cheng, K. Dembczyński, and E. Hüllermeier. Learning monotone nonlinear models using the Choquet integral. *Machine Learning*, 89(1–2): 183–211, 2012.

The OEIS Foundation Inc. On-line encyclopedia of integer sequences. `https://oeis.org`, 2009. [Online; accessed September 13, 2014].

G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

C. Ulu and M. Köksalan. An interactive procedure for selecting acceptable alternatives in the presence of multiple criteria. *Naval Research Logistics*, pages 592–606, 2001.

R. van de Kamp, A. Feelders, and N. Barile. Isotonic classification trees. In N. M. Adams, C. Robardet, A. Siebes, and J.-F. Boulicaut, editors, *Advances in Intelligent Data Analysis VIII, 8th International Symposium on Intelligent Data Analysis, IDA 2009, Lyon, France, August 31 - September 2, 2009. Proceedings*, volume 5772 of *Lecture Notes in Computer Science*, pages 405–416. Springer, 2009.

L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM review*, 38(1): 49–95, 1996.

V. N. Vapnik. *Statistical learning theory*. Wiley, 1998.

Ph. Vincke. Robust solutions and methods in decision-aid. *Journal of Multi-Criteria Decision Analysis*, 8(3):181–187, 1999.

J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, NJ, USA, 3rd edition, 1972.

W. Waegeman, B. De Baets, and L. Boullart. ROC analysis in ordinal regression learning. *Pattern Recognition Letters*, 29(1):1–9, 2008.

D. West, P. Mangiameli, R. Rampal, and V. West. Ensemble strategies for a medical diagnostic decision support system: A breast cancer diagnosis application. *European Journal of Operational Research*, 162(2):532–551, 2005.

L.A. Wolsey. *Integer Programming*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1998.

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

T. Zhang. Analysis of multi-stage convex relaxation for sparse regularization. *Journal of Machine Learning Research*, 11:1081–1107, 2010.

C. Zopounidis and M. Doumpos. Building additive utilities for multi-group hierarchical discrimination: The M.H.DIS method. *Optimization Methods & Software*, 14(3):219–240, 2000.

C. Zopounidis and M. Doumpos. Multicriteria classification and sorting methods: A literature review. *European Journal of Operational Research*, 138(2):229–246, 2002.