


École doctorale n° 130 : Informatique, Télécommunications et Electronique

THÈSE DE DOCTORAT

pour obtenir le grade de docteur délivré par

l'Université Pierre et Marie Curie

Spécialité doctorale : Informatique

présentée et soutenue publiquement par

Hugo GILBERT

le 11 décembre 2017

Oracle-based Algorithms for Optimizing Sophisticated Decision Criteria in Sequential, Robust and Fair Decision Problems

Directeur de thèse : **Olivier SPANJAARD**
Co-encadrants de thèse : **Paolo VIAPPIANI** et **Paul WENG**

Rapporteurs

M. Włodzimierz OGRYCZAK,	Professeur à l'Université Technique de Varsovie
M. Régis SABBADIN,	Directeur de Recherche INRA, INRA (Toulouse)

Examinateurs

M. Yann CHEVALEYRE,	Professeur à l'Université Paris Dauphine (Paris IX)
M. Bruno ESCOFFIER,	Professeur à l'Université Pierre et Marie Curie (Paris VI)
Mme Hélène FARGIER,	Directeur de Recherche CNRS à l'Université Paul Sabatier (Toulouse III)

Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6
UMR 7606, 4 place Jussieu, 75005 Paris, France

Remerciements

En premier lieu, je tiens à remercier mon directeur de thèse, Olivier Spanjaard, Maître de conférence à l'Université Pierre et Marie Curie. J'ai trouvé en lui un directeur de thèse patient, compréhensif et clairvoyant. Je peux témoigner de sa disponibilité, de sa patience et de sa gentillesse, mais aussi de ses grandes aptitudes pédagogiques et d'analyse. Ses qualités humaines et scientifiques ont rendu mon travail de thèse aisé et agréable. Je tiens aussi à remercier Paolo Viappiani, Chargé de Recherche CNRS à l'Université Pierre et Marie Curie, et Paul Weng, Maître de conférence à l'Université Sun Yat-Sen, qui ont co-encadré ma thèse. Je tiens à leur exprimer ma gratitude pour leur sympathie, leur grande disponibilité et pour leurs nombreux conseils qui ont toujours été précieux.

Je remercie vivement Włodzimierz Ogryczak, Professeur à l'Université Technique de Varsovie, et Régis Sabbadin, Directeur de Recherche à l'INRA de Toulouse, pour l'intérêt qu'ils ont porté à mes travaux de recherche en me faisant l'honneur de rapporter ma thèse. Je remercie également Yann Chevaleyre, Professeur à l'Université Paris Dauphine, Bruno Escoffier, Professeur à l'Université Pierre et Marie Curie et Hélène Fargier, Directeur de Recherche CNRS à l'Université Paul Sabatier pour avoir accepté d'évaluer ma thèse en tant qu'examinateurs. Enfin, je remercie Aurélie Beynier, Maître de conférence à l'Université Pierre et Marie Curie, qui était dans le jury d'évaluation à mi-parcours de ma thèse.

Durant les trois dernières années, j'ai eu la chance de collaborer avec de nombreux co-auteurs qui m'ont beaucoup appris. Parmi les personnes non encore citées, je remercie Nawal Benabbou, Esther Nicart, Patrice Perny et Bruno Zanuttini qui m'ont aidé à devenir un meilleur chercheur.

Mes années au LIP6 ont été des plus agréables grâce à la bonne humeur des membres des équipes Décision et Recherche Opérationnelle. Je remercie Spyros Angelopoulos, Evripidis Bampis, Christoph Dürr, Bruno Escoffier, Pierre Fouilloux, Christophe Gonzales, Emmanuel Hyon, Safia Kedad-Sidhoum, Thibaut Lust, Viet Hung Nguyen, Fanny Pascual, Patrice Perny et Pierre-Henry Wuillemin. Cette partie des remerciements serait bien incomplète sans évoquer les doctorants et chercheurs post-doctoraux du couloir 26-00. J'ai eu le plaisir de côtoyer Hamza Agli, Lamia Aoudia, Khaled Belahcene, Nawal Benabbou, Margot Calbrix, Morgan Chopin, Santiago-José Cortijo Aragon, Kimberly Garcia, Shendan Jin, Siao-Leu Phouratsamay, Cécile Rottner, Yasmina Sedik et Angelina Vidali. Enfin, je souhaite tout le meilleur à la sympathique et dynamique relève composée de Nadjet Bourdache, Anne-Elisabeth Falq, Adèle Pass-Lanneau et Alexandre Teillier.

Je tiens aussi à remercier le personnel administratif qui m'a accompagné, et notamment Ghislaine Mary, Noura El Habchi, Belaid Nait Sidnas et Samiha Taba. Je leur suis reconnaissant de leur patience et de leur bienveillance.

Mes remerciements les plus affectueux vont à ceux qui m'ont soutenu et supporté pendant mon doctorat. Parmi les personnes non encore citées, il y a mes amis : Alejendra, Amandine, Christelle, Corentin, David, Edouard, Florian, Luc, Neda et Tristan et ma famille : ma mère Sabine, mon père Laurent, mon frère Adrien et ma sœur Agathe. Enfin des remerciements particuliers vont à Julien qui est un véritable ami et à Léa qui embellit

chaque jour mon quotidien.

Contents

List of Figures

List of Tables

Introduction

> *“ Le sage regarde, en toutes choses, non le résultat, mais la décision qu'il a prise. ”*
>
> Lettres à Lucillius
> Sénèque

The process of decision making is crucial as the choices we face every day are numerous and some can have major consequences. For this reason, researchers from economy, philosophy, psychology or management science have gathered in a research field named *decision theory*. Decision theory investigates *decision problems*; a decision problem involves a *decision maker* who has at her disposal several options (e.g., actions, plans, objects) and aims to make a choice according to her preferences, her beliefs and all pieces of relevant information available to her. Methods from decision theory can be used, for instance, to create autonomous machines that automatically make decisions according to the preferences of the users (e.g., autonomous cars, music flow system); they can also be used to advise someone (decision aid). The system will need to explore the set of solutions and present to the user the ones that seem most compatible with her preferences (e.g., recommendation systems). Lastly, autonomous systems using tools from decision theory can help understanding and predicting the behavior of a user (e.g., profiling a customer on a website).

To formalize how people make decisions and reason about their preferences, decision theory investigates mathematical models called *decision models*. A decision model is a system of representation that encompass the preferences, the beliefs and the criteria of the decision maker. It also includes a *decision criterion*, i.e., a mathematical function whose optimization enables to identify the best options. This modeling problem is hard for several reasons:

- The number of possible choices or alternatives can be huge and even exponential in the size of the problem considered; in the latter case, we say that the problem is of *combinatorial nature* [Papadimitriou and Steiglitz, 1982].

- The outcome of each option might be imprecisely known as in *robust optimization* [Ben-Tal *et al.*, 2009; Kouvelis and Yu, 1997], or subject to exogenous events as in *decision problems under risk* [von Neumann and Morgenstern, 1947].

- The decision problem considered may involve several agents (*multi-agent optimization problem* [Brandt *et al.*, 2016; Moulin, 1991]) and there may not exist a solution which satisfies everyone.

- The problem may be *dynamic* involving different time steps in which decisions should be made [Raiffa, 1993]. The different choices available might lead to more or less favorable situations according to the time step. Moreover, the impact of a decision should be evaluated in the long term.

- The information available on the decision maker or on the decision problem may be poor, imprecise [Regan and Boutilier, 2009], incomplete or only of ordinal [Filar, 1983; Fürnkranz *et al.*, 2012] or qualitative nature [Dubois and Prade, 1995; Sabbadin, 2001].

Researchers from decision theory and combinatorial optimization have developed numerous decision models and decision criteria to integrate and overcome those difficulties. Some examples, for multicriteria decision making, are the Weighted Average (WA), the Ordered Weighted Average (OWA) operator [Yager, 1988], the Weighted OWA (WOWA) operator [Torra, 1997], the Choquet integral [Grabisch and Labreuche, 2010], and for decision under uncertainty, the Expected Utility (EU) model [von Neumann and Morgenstern, 1947], the Rank Dependent Utility (RDU) model [Quiggin, 1993], the Weighted Expected Utility (WEU) model [Chew, 1983], the Skew-Symmetric Bilinear (SSB) utility model [Fishburn and LaValle, 1987], the Choquet Expected Utility (CEU) model [Schmeidler, 1986]. Their mathematical properties have been thoroughly studied to prove their efficiency to represent a wide scope of preferences as well as their ability to tackle important concerns as fairness, robustness or risk aversion.

These models can help at (1) making decisions, (2) advising a decision maker or (3) understanding and predicting the behavior of someone. However, these three tasks can be costly to perform even for a computer. Thus, efficient and operational computational methods must be developed to perform these tasks. This is the topic of *algorithmic decision theory* which is at the crossroads between decision theory and computer science. In algorithmic decision theory, we aim to combine powerful tools from computer science with models from decision theory. Some of these powerful tools are called *oracle* methods (e.g., double oracle methods [McMahan *et al.*, 2003], cutting plane methods [Grötschel *et al.*, 1981]). In short, oracle methods are master-slave methods where the slave algorithm is called "oracle" and is used to generate new constraints on the optimization problem that is solved by the master algorithm. Oracle methods enable to solve efficiently problems that involve a large number of constraints by dynamically generating the ones that are necessary for the resolution. The idea motivating these methods is that even if the number of constraints defining a problem is huge, the number of constraints required to solve it may be much smaller.

However, even with the most powerful computational tools, the difficulty of optimizing, advising or making predictions is highly dependent on the decision model considered. Therefore, decision models can be both evaluated on the basis of their mathematical properties and on the basis of their computational properties. Each model provides a compromise on these two objectives, which is worth investigating.

Goal of the thesis. In this thesis, we investigate the use of several decision models to solve sequential decision problems under risk, robust combinatorial optimization problems and fair multi-agent optimization problems. We will make an extensive use of oracle methods to efficiently solve those problems. A particular attention is given to the skew-symmetric bilinear utility model, the weighted expected utility model and their counter-

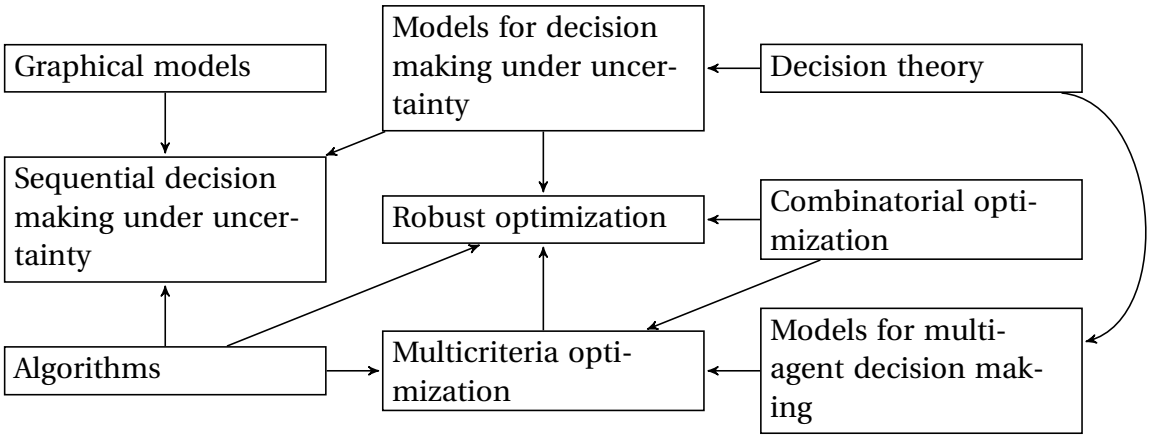


Figure 1: Keywords graph.

parts in multicriteria decision making. Indeed, we show that these models are interesting for multiple reasons. They extend the standard models (e.g., the expected utility model) and allow to represent a broader class of preferences while retaining their good theoretical and algorithmic properties. The thesis will focus both on theoretic (e.g., complexity results) and operational (e.g., design of practically efficient solution methods) aspects of these problems. A graph of the different keywords related to this thesis is given in Figure 1. An arc from keyword x to keyword y means that theme x is useful for the study of theme y.

Organization of the manuscript. The manuscript starts with three introductory chapters that present the decision models, the optimization problems and the algorithmic methods that are used and investigated in this thesis.

• Chapter 1 presents the different types of decision problems and the associated decision models that will be investigated in this thesis. After a general presentation of decision problems, we introduce the different concepts related to multicriteria decision problems, multi-agent decision problems, robust decision problems and decision problems under risk.

• Chapter 2 presents the combinatorial optimization problems that are investigated in the thesis. A particular emphasis is given to sequential decision problems under risk which are particular combinatorial decision problems in which the decision maker has to make decisions under risk repeatedly in different situations and at different time steps. In this chapter, we will present two representations to model these problems, namely Markov decision processes and decision trees.

• Lastly, Chapter 3 presents the oracle methods that are used in this thesis. To present these methods, two particular domains are developed, namely game theory and fractional programming. Indeed, we will resort to these oracle methods by identifying the optimization problems encountered in this thesis as specific problems from game theory and fractional programming. The oracle methods that are used in this thesis are explained and illustrated with examples.

After these three introductory chapters, we explore how to solve the combinatorial problems introduced in Chapter 2 according to the decision models presented in Chapter

1 with the oracle methods presented in Chapter 3. This is done in the next 4 chapters.

• Chapter 4 investigates the resolution of sequential decision problems under risk with the Skew-Symmetric Bilinear (SSB) utility model and the Weighted Expected Utility (WEU) model in decision trees and in Markov decision processes. For the SSB utility model, we prove that while finding an optimal mixed strategy in decision trees is a problem of polynomial complexity, finding an optimal deterministic strategy is an NP-hard problem. Interestingly, for the WEU model, which is a special case of SSB utility, both problems become polynomial. In Markov decision processes with finite horizon, we prove that finding an optimal mixed policy for SSB or WEU is a pseudo-polynomial problem. In each case, we provide oracle methods to compute an optimal strategy.

• Chapter 5 is devoted to decision making under risk with ordinal/preference-based approaches or imprecisely known cost parameters. With this type of information, we can either work with decision criteria adapted to this lack of information or we can precise the values of these parameters by using an elicitation procedure. Chapter 5 presents three works related to these problems. 1) We discuss an ordinal approach based on a quantile criterion to solve finite horizon Markov decision processes where only a preference order over possible episodes is known; 2) assuming the rewards of an infinite horizon Markov decision process are imprecisely known, we refine an interactive version of the value iteration procedure [Weng and Zanuttini, 2013] to reduce the number of preference queries required by the procedure; 3) we develop elicitation methods to assist a decision maker consistent with a weighted expected utility model for which the parameters are imprecisely known.

• Chapter 6 presents a game-theoretic view of robust combinatorial optimization problems with interval data. This game-theoretic view makes it possible to design an efficient anytime procedure to compute a lower bound on the value of an optimal minmax regret solution. We show how this lower bound compares to other lower bounds proposed in the literature and explain how this lower bound can be efficiently integrated in a branch and bound procedure to find an optimal minmax regret solution. This approach is then instantiated on the robust shortest path problem with interval data. Numerical tests are provided that prove the efficiency of the approach.

• Chapter 7 studies multi-agent decision problems with two class of aggregation operators, namely ordered weighted averages and mixture operators. These operators make it possible to find fair solutions (w.r.t. the Pigou-Dalton principle). In the first part of Chapter 7, we focus on ordered weighted averaging operators with decreasing weights and argue that randomized strategies enable us to enhance the fairness of an optimal solution. Then we develop a game-theoretic view of randomized fair multi-agent optimization and derive complexity results and solution methods. In the second part of Chapter 7, we come back to deterministic solutions and consider the optimization of mixture operators in allocation problems and proportional representation problems. For both problems, we are able to determine complexity results and solution methods.

These results are based on several publications (given in Table 1). Furthermore, these chapters have been enriched with some unpublished results.

Chapter 4	[Gilbert and Spanjaard, 2017b; Gilbert *et al.*, 2015b, 2016]
Chapter 5	[Gilbert *et al.*, 2015a, 2017a,b]
Chapter 6	[Gilbert and Spanjaard, 2017a]
Chapter 7	[Gilbert, 2017]

Table 1: Connections between the different chapters and the different papers.

References

A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*. Princeton University Press, 2009. 1

F. Brandt, V. Conitzer, U. Endriss, A.D. Procaccia, and J. Lang. *Handbook of computational social choice*. Cambridge University Press, 2016. 1

S.H. Chew. A generalization of the quasilinear mean with applications to the measurement of income inequality and decision theory resolving the Allais paradox. *Econometrica: Journal of the Econometric Society*, pages 1065–1092, 1983. 2

D. Dubois and H. Prade. Possibility theory as a basis for qualitative decision theory. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 95, pages 1924–1930, 1995. 2

J.E. Filar. Percentiles and markov decision processes. *Operations Research Letters*, 2:13–15, 1983. 2

P.C. Fishburn and I.H. LaValle. A nonlinear, nontransitive and additive-probability model for decisions under uncertainty. *Ann. Statist.*, 15(2):830–844, 06 1987. 2

J. Fürnkranz, E. Hüllermeier, W. Cheng, and S.H. Park. Preference-based reinforcement learning: A formal framework and a policy iteration algorithm. *Machine Learning*, 89(1):123–156, 2012. 2

H. Gilbert and O. Spanjaard. A game theoretic bound for minmax regret optimization problems with interval data. In *European Journal of Operational Reasearch*, 2017. 5

H. Gilbert and O. Spanjaard. Complexity of Solving Decision Trees with Skew-Symmetric Bilinear Utility. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence 2017*, 2017. 5

H. Gilbert, O. Spanjaard, P. Viappiani, and P. Weng. Reducing the Number of Queries in Interactive Value Iteration. In *Proceedings of the International Conference on Algorithmic Decision Theory 2015*, 2015. 5

H. Gilbert, O. Spanjaard, P. Viappiani, and P. Weng. Solving MDPs with Skew Symmetric Bilinear Utility Functions. In *Proceedings of the International Joint Conference on Artificial Intelligence 2015*, pages 1989–1995, 2015. 5

H. Gilbert, B. Zanuttini, P. Viappiani, P. Weng, and E. Nicart. Model-free reinforcement learning with skew-symmetric bilinear utilities. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence 2016*, 2016. 5

H. Gilbert, N. Benabbou, P. Perny, O. Spanjaard, and P. Viappiani. Incremental Decision Making Under Risk with the Weighted Expected Utility Model. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2017. 5

H. Gilbert, P. Weng, and Y. Xu. Optimizing Quantiles in Preference-based Markov Decision Processes. In *Proceedings of AAAI 2017*, 2017. 5

H. Gilbert. Fair Proportional Representation Problems with Mixture Operators. In *Proceedings of the International Conference on Algorithmic Decision Theory 2017*, 2017. 5

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. 2

M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981. 2

P. Kouvelis and G. Yu. *Robust discrete optimization and its applications*. Kluwer Academic Publisher, 1997. 1

H.B. McMahan, G.J. Gordon, and A. Blum. Planning in the presence of cost functions controlled by an adversary. In *International Conference on Machine Learning*, pages 536–543, 2003. 2

H. Moulin. *Axioms of cooperative decision making*. Number 15. Cambridge University Press, 1991. 1

C.H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1982. 1

J. Quiggin. *Generalized Expected Utility Theory: The Rank-Dependent Model*. Kluwer, 1993. 2

H. Raiffa. Decision analysis: introductory lectures on choices under uncertainty. 1968. *MD computing: computers in medical practice*, 10(5):312, 1993. 2

K. Regan and C. Boutilier. Regret-based reward elicitation for markov decision processes. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 444–451, Arlington, Virginia, United States, 2009. AUAI Press. 2

R. Sabbadin. Possibilistic markov decision processes. *Engineering Applications of Artificial Intelligence*, 14(3):287 – 300, 2001. Soft Computing for Planning and Scheduling. 2

D. Schmeidler. Integral representation without additivity. *Proceedings of the American mathematical society*, 97(2):255–261, 1986. 2

V. Torra. The weighted OWA operator. *International Journal of Intelligent Systems*, 12(2):153–166, 1997. 2

J. von Neumann and O. Morgenstern. *Theory of games and economic behaviour*. Princeton University Press, 1947. 1, 2

P. Weng and B. Zanuttini. Interactive value iteration for markov decision processes with unknown rewards. In Francesca Rossi, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*. IJCAI/AAAI, 2013. 4

R.R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on systems, Man, and Cybernetics*, 1988. 2

Chapter 1

Decision Problems and Decision Models

“ A theory is no more like a fact than a photograph is like a person. ”

E. W. Howe

Section Contents

Summary of the chapter

In this chapter, we present and define the decision problems and the associated decision models that are investigated in this thesis. We start from a very general view of decision problems and then instantiate the different concepts introduced to multicriteria decision problems, multi-agent decision problems, robust decision problems and decision problems under risk. In each case, we precise the contributions of this thesis and indicate the corresponding chapters. A particular emphasis is given to the decision criteria that are investigated in this thesis: mixture operators, skew-symmetric bilinear utility models and weighted expected utility models.

1.1 Decision Problems

Decision problems are numerous in everyone's day-to-day life. Choosing what to eat for breakfast, what clothes to wear for the day, or the transportation means that we shall use to go to work are decision problems that affect everyone. Some of this decision problems may be of great importance as choosing the next strategic investment of a company. It is therefore paramount that decision analysts work with rigorous definitions and models of decision problems to be able to help a decision maker, whatever the decision problem she faces.

Decision problems are specified through some specific characteristics [Perny, 2000]:

- A set $\mathscr{A}$ of actions or alternatives that are available to the decision maker(s).
- A set $\mathscr{N}$ of different aspects on which the elements of $\mathscr{A}$ can be compared. These different aspects can be criteria that may be conflicting, several scenarios that may be more or less likely or the preferences of different individuals.
- A problem $\mathscr{P}$ which can be of four types:
  1. Determining a *choice* of an optimal or a "sufficiently good" element in $\mathscr{A}$.
  2. Determining a complete *ranking* of the elements in $\mathscr{A}$ from best to worst.
  3. Determining an *affectation* of the elements in $\mathscr{A}$ to predefined categories.
  4. Determining a *clustering* of the elements in $\mathscr{A}$ in homogeneous subsets.

Thus, without loss of generality, we will denote a decision problem as a tuple $(\mathscr{P},\mathscr{A},\mathscr{N})$. In this thesis, we will mostly be interested in the first kind (choice) and third kind (affectation) of problems.

As illustrated in Figure 1.1, the resolution of decision problems follows two important steps, namely *formalization* and *resolution*. In the formalization step, the problem $(\mathscr{P},\mathscr{A},\mathscr{N})$ is modeled in a given language (using tools from logic theory, probability theory, utility theory, ...) and a *decision criterion* is chosen. A decision criterion is a mathematical function mapping solutions to real values such that solving the problem reduces to optimizing the decision criterion. The *decision model* consists of the combination of the language in which is translated the problem and of the decision criterion used. In the resolution step, we determine a solution of the problem w.r.t. the given decision model. The formalization phase is crucial as a slight change in the decision model (either in the language chosen to describe the model or in the decision criterion) can highly impact both the quality of the solution found in the resolution step and the efficiency of the resolution step. It is therefore paramount that the decision model chosen is endorsed by strong theoretical evidences that it is the "right" model or at least a "good" model.

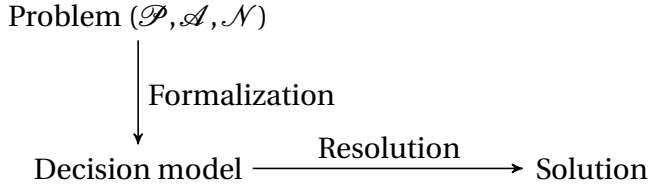


Figure 1.1: Decision process.

To compare different decision models, one can investigate their *prescriptive*, *descriptive* and *normative* abilities. The prescriptive abilities of a model denote the capacity with which the model can be used to make recommendations. The descriptive abilities of a model denote its capacity to accurately represent the preferences of the decision maker(s). Lastly, the normative abilities of a model denote the adequacy between the preferences that can be represented by the model and the ones that are theoretically desirable. To choose a model based on these three aspects, the decision analyst must carefully study and understand the preferences of the decision maker(s). These preferences are mathematically represented by a binary relation.

1.1.1 Relational Representation

In a decision problem, the preferences of the decision maker can generally be described by a binary relation on $\mathscr{A}$.

Definition 1. *A binary relation on a set $\mathscr{A}$ is a subset of the Cartesian product $\mathscr{A} \times \mathscr{A}$.*

We will use the symbol R to denote a binary relation. We now recall some definitions on binary relations.

Definition 2. *A binary relation* R *on a set $\mathscr{A}$ is said:*

- *reflexive iff:* $\forall \mathbf{x} \in \mathscr{A}, \mathbf{x}\mathrm{R}\mathbf{x}$[1].
- *symmetric iff:* $\forall \mathbf{x}, \mathbf{y} \in \mathscr{A}, \mathbf{x}\mathrm{R}\mathbf{y} \Rightarrow \mathbf{y}\mathrm{R}\mathbf{x}$.
- *asymmetric iff:* $\forall \mathbf{x}, \mathbf{y} \in \mathscr{A}, \mathbf{x}\mathrm{R}\mathbf{y} \Rightarrow \neg(\mathbf{y}\mathrm{R}\mathbf{x})$.
- *antisymmetric iff:* $\forall \mathbf{x}, \mathbf{y} \in \mathscr{A}, \mathbf{x}\mathrm{R}\mathbf{y}$ *and* $\mathbf{y}\mathrm{R}\mathbf{x} \Rightarrow \mathbf{y} = \mathbf{x}$.
- *transitive iff:* $\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathscr{A}, \mathbf{x}\mathrm{R}\mathbf{y}$ *and* $\mathbf{y}\mathrm{R}\mathbf{z} \Rightarrow \mathbf{x}\mathrm{R}\mathbf{z}$.
- *complete iff:* $\forall \mathbf{x}, \mathbf{y} \in \mathscr{A}$, *either* $\mathbf{x}\mathrm{R}\mathbf{y}$ *or* $\mathbf{y}\mathrm{R}\mathbf{x}$.

Definition 3. *A preorder is a reflexive and transitive binary relation.*

Definition 4. *An order is an antisymmetric, reflexive and transitive binary relation, i.e., it is an antisymmetric preorder.*

Definition 5. *A preference relation is a binary relation representing the preferences of the decision maker that we denote by $\succsim$.*

It is often implicitly assumed that $\succsim$ is at least a preorder. However, we will encounter non transitive preference relations in this thesis.

Definition 6. *For a preference relation $\succsim$ on a set $\mathscr{A}$, the symmetric and asymmetric parts of $\succsim$ are denoted by $\sim$ and $\succ$. They are defined by:*

$$\forall \mathbf{x}, \mathbf{y} \in \mathscr{A}, \mathbf{x} \sim \mathbf{y} \Leftrightarrow \mathbf{x} \succsim \mathbf{y} \text{ and } \mathbf{y} \succsim \mathbf{x}$$
$$\forall \mathbf{x}, \mathbf{y} \in \mathscr{A}, \mathbf{x} \succ \mathbf{y} \Leftrightarrow \mathbf{x} \succsim \mathbf{y} \text{ and } \mathbf{y} \not\succsim \mathbf{x}$$

[1] Note that bold letters are used to denote vectors. Solutions are here written with bold characters as they will usually be given by vectors in the sequel.

The symmetric and the asymmetric parts of a preference relation have the following meaning. If $\mathbf{x} \succsim \mathbf{y}$, then $\mathbf{x}$ is at least as preferred as $\mathbf{y}$. If $\mathbf{x} \succ \mathbf{y}$, then $\mathbf{x}$ is strictly preferred to $\mathbf{y}$. If $\mathbf{x} \sim \mathbf{y}$, the decision maker is indifferent between $\mathbf{x}$ and $\mathbf{y}$. Lastly, two elements $\mathbf{x}$ and $\mathbf{y}$ are incomparable if neither $\mathbf{x} \succsim \mathbf{y}$ nor $\mathbf{y} \succsim \mathbf{x}$.

Definition 7. *Given a preference relation $\succsim$ on $\mathscr{A}$, an element $\mathbf{x} \in \mathscr{A}$ is said $\succsim$-dominated if there exists an element $\mathbf{y} \in \mathscr{A}$ such that $\mathbf{y} \succ \mathbf{x}$. Otherwise, $\mathbf{x}$ is said to be non $\succsim$-dominated or $\succsim$-optimal.*

At the end of the formalization phase, the decision problem $(\mathscr{P}, \mathscr{A}, \mathscr{N})$ is transformed into a structure $(\mathscr{P}, \mathscr{A}, \succsim)$ and the goal of the resolution is to find the $\succsim$-optimal solutions. If the preference relation is complete, the resolution consists in finding an optimal solution w.r.t. $\succsim$. Otherwise, several equivalence classes of optimal solutions exist and we may return one optimal solution per equivalence class.

The preference relation $\succsim$ reflects how the different solutions are ranked according to the different elements in $\mathscr{N}$. Thus, the structure of $\mathscr{N}$ and the nature of its elements have an impact on $\succsim$ and plays an important role for both the formalization and the resolution steps. In the following sections, we further present different kinds of decision problems that differ from one another by the nature of the elements in $\mathscr{N}$.

1.2 Multicriteria Decision Making

In multicriteria decision problems [Roy, 2013], the set $\mathscr{N}$ contains the different criteria that impact the quality of the solutions in $\mathscr{A}$. For instance, if the decision problem is to find a new flat, $\mathscr{N}$ may contain the cost of the flat, its size or its neighborhood. If, on the contrary, the decision problem consists in choosing a new car, $\mathscr{N}$ may contain the color or the fuel consumption of the possible vehicles. Without loss of generality, we will assume that $\mathscr{N}$ is composed of n different criteria and we will denote by i ($i \in \{1, \ldots, n\}$) the i^{th} criterion in $\mathscr{N}$. For each criterion i, each solution $\mathbf{x} \in \mathscr{A}$ receives a score (also called utility) x_i in a totally ordered set $\mathscr{X}_i$. For simplicity, we will generally assume that $\mathscr{X}_i \subseteq \mathbb{R}$, for all $i \in \{1, \ldots, n\}$. We call *criteria space* the cardinal product $\mathscr{X} = \mathscr{X}_1 \times \ldots \times \mathscr{X}_n$. The value of x_i represents the extent to which solution $\mathbf{x}$ is satisfactory regarding criterion i (the higher the better[2]). Therefore, each solution $\mathbf{x} \in \mathscr{A}$ can be associated to a vector of n elements, $(x_1, \ldots, x_n)$ which is the image of solution $\mathbf{x} \in \mathscr{A}$ in the criteria space $\mathscr{X}$. By abuse of notation, we may use the notation $\mathbf{x}$ to directly denote the vector $(x_1, \ldots, x_n)$.

The utilities x_i induce n complete preference relations $\succsim_i$ w.r.t. each criterion. More formally, $\succsim_i$ for $i \in \{1, \ldots, n\}$ is defined by:

$$\forall \mathbf{x}, \mathbf{y} \in \mathscr{A}, \mathbf{x} \succsim_i \mathbf{y} \Leftrightarrow x_i \geq y_i$$

These preference relations enable us to identify dominated solutions when they all agree on a pair of solutions.

Definition 8. *A solution $\mathbf{x} \in \mathscr{A}$* Pareto-dominates *a solution $\mathbf{y} \in \mathscr{A}$ if:*

$$\forall i \in \{1, \ldots, n\}, \mathbf{x} \succsim_i \mathbf{y}$$
$$\exists i \in \{1, \ldots, n\}, \mathbf{x} \succ_i \mathbf{y}$$

A solution $\mathbf{x}$ is Pareto-optimal *iff there is no $\mathbf{y} \in \mathscr{A}$ such that $\mathbf{y}$ Pareto-dominates $\mathbf{x}$.*

[2] We take the convention here to study maximization problems.

Pareto optimality ensures that the solution is efficient in the sense that the utility of a criterion cannot be increased without decreasing the utility of another criterion.

Example 1. *We consider the decision problem of buying a new car. Several cars have been selected and they are now compared on the basis of two conflicting criteria, their size and their fuel consumption efficiency (ignoring all other attributes). The different options that have been selected are graphically represented by dots in Figure 1.2 where the x-axis (resp. y-axis) gives the utility of the car w.r.t. the criterion "consumption efficiency" (resp. "size"). The zones that are hatched in red represent the dominance cones associated to each solution. These zones are defined such that any solution in the dominance cone of a solution* $\mathbf{x}$ *is Pareto dominated by* $\mathbf{x}$*. The three Pareto-optimal solutions are represented in red. The car chosen should be one of these three cars.*

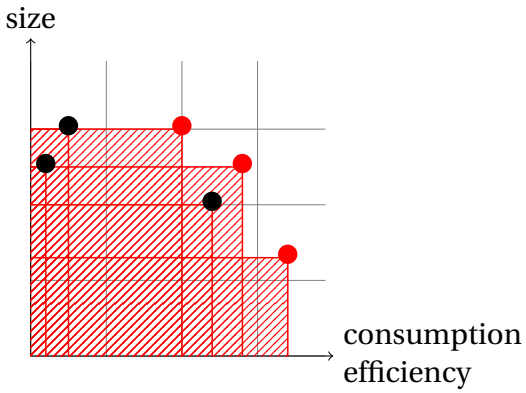


Figure 1.2: Consumption efficiency and size of the different cars available.

Unfortunately, because the different criteria in $\mathcal{N}$ are often conflicting, the Pareto-optimal solutions are often numerous. Hence, the identification of all Pareto optimal solutions can be intractable and may not directly lead to a satisfying prescription. One way to address this problem is to aggregate the preferences over the different criteria via an aggregation operator [Ehrgott, 2006; Greco *et al.*, 2005].

As illustrated by Figure 1.3, two different approaches are possible to aggregate the point of views of the different criteria [Perny, 2000]:

- First aggregate and then compare. More formally, this approach maps each solution $\mathbf{x} \in \mathcal{A}$ to a real value $F(\mathbf{x})$ by using an aggregation operator $F : \mathcal{X} \rightarrow \mathbb{R}$. The comparison of two solutions $\mathbf{x}$ and $\mathbf{y}$ is then based on the comparison of the two real values $F(\mathbf{x})$ and $F(\mathbf{y})$ such that $\mathbf{x} \succsim \mathbf{y}$ iff $F(\mathbf{x}) \geq F(\mathbf{y})$. As the different utilities w.r.t the different criteria of a solution are aggregated, one disadvantage of this approach is that it assumes the commensurability of the different utility values.

- First compare and then aggregate. More formally, given two solutions $\mathbf{x}$ and $\mathbf{y}$ in $\mathcal{A}$, this approach starts by comparing $\mathbf{x}$ and $\mathbf{y}$ on the basis of each criterion i by using a function $\phi_i : \mathcal{X}_i^2 \rightarrow \mathbb{R}$. Put another way, function ϕ_i returns a signed intensity of preference between $\mathbf{x}$ and $\mathbf{y}$ when compared w.r.t. criterion i. Once this first step is done, the different values $\phi_i(x_i, y_i)$ are then aggregated by using an aggregation operator $F : \mathbb{R}^n \rightarrow \mathbb{R}$ and the preference between $\mathbf{x}$ and $\mathbf{y}$ is determined by the sign of $F(\phi_1(x_1, y_1), \ldots, \phi_n(x_n, y_n))$.

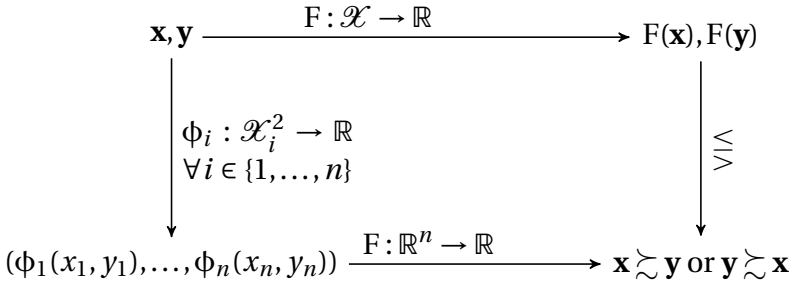


Figure 1.3: Two different ways of aggregating preferences.

Many different aggregation operators are used and investigated in the literature. To choose one operator instead of another, one can investigate their mathematical properties to prove that they can be used to find solutions in $\mathscr{A}$ that have desirable properties regarding the specific preferences of the decision maker. We now give some of these mathematical properties and start by defining formally *aggregation* functions.

Definition 9. *We will call a function* $\mathrm{F} : [a,b]^n \to \mathbb{R}$ *an* aggregation *function if and only if* F *is monotone nondecreasing and* $\mathrm{F}(\underbrace{a,\ldots,a}_{n}) = a$, $\mathrm{F}(\underbrace{b,\ldots,b}_{n}) = b$.

Where monotone nondecreasingness is defined as follows.

Definition 10. *A function* $\mathrm{F} : \mathbb{R}^n \to \mathbb{R}$ *is monotone nondecreasing if and only if,*

$$\forall \mathbf{x},\mathbf{y} \in \mathbb{R}^n, \forall i \in \{1,\ldots,n\}, x_i \geq y_i \Rightarrow \mathrm{F}(\mathbf{x}) \geq \mathrm{F}(\mathbf{y})$$

Moreover, F *is compatible with Pareto dominance if and only if,*

$$\forall \mathbf{x},\mathbf{y} \in \mathbb{R}^n, \forall i \in \{1,\ldots,n\}, x_i \geq y_i \text{ and } \exists i \in \{1,\ldots,n\}, x_i > y_i \Rightarrow \mathrm{F}(\mathbf{x}) > \mathrm{F}(\mathbf{y})$$

Compatibility with Pareto-dominance is necessarily imposed to insure that maximizing the aggregation operator F will yield a Pareto-optimal solution. To insure compatibility with Pareto-dominance, an aggregation operator F should be increasing in each of its variables. Therefore, this condition is also called increasingness.

The condition applied to operator F on vectors $(a,\ldots,a)$ and $(b,\ldots,b)$ in Definition 9 is very natural and guarantees that the overall evaluation of a vector corresponds to the evaluation w.r.t. each of the criteria when there is unanimity. It is often desirable to impose this condition for all possible values. This property is called idempotency or unanimity.

Definition 11. *A function* $\mathrm{F} : \mathbb{R}^n \to \mathbb{R}$ *is* idempotent *if:*

$$\forall x \in \mathbb{R}, \mathrm{F}(\underbrace{x,\ldots,x}_{n}) = x$$

Definition 12. *A function* $\mathrm{F} : \mathbb{R}^n \to \mathbb{R}$ *is symmetric if:*

$$\forall \mathbf{x} \in \mathbb{R}^n, \mathrm{F}(\mathbf{x}) = \mathrm{F}(\mathbf{x}^{\boldsymbol{\sigma}})$$

where $\boldsymbol{\sigma}$ *is any permutation of* $\{1,\ldots,n\}$ *and* $\mathbf{x}^{\boldsymbol{\sigma}}$ *is the vector such that* $\mathbf{x}_i^{\boldsymbol{\sigma}} = \mathbf{x}_{\boldsymbol{\sigma}(i)}$.

A symmetric function considers that the n criteria are equally important. While this property may not be desirable in multicriteria decision problems, it is generally imposed in multi-agent decision problems (see Section 1.3) under the name *anonymity*.

Definition 13. *A function* $\mathrm{F} : \mathbb{R}^n \to \mathbb{R}$ *has an averaging behaviour if:*

$$\forall \mathbf{x} \in \mathbb{R}^n, \min(\mathbf{x}) \leq \mathrm{F}(\mathbf{x}) \leq \max(\mathbf{x})$$

The following definitions formalize two notions of invariance:

Definition 14. *A function* $\mathrm{F} : \mathbb{R}^n \to \mathbb{R}$ *is shift-invariant if* $\mathrm{F}(\mathbf{x} + y\mathbb{1}_n) = \mathrm{F}(\mathbf{x}) + y$ *where* $y \in \mathbb{R}$ *and* $\mathbb{1}_n = (\underbrace{1, \ldots, 1}_{n})$.

Definition 15. *A function* $\mathrm{F} : \mathbb{R}^n \to \mathbb{R}$ *is homogeneous if* $\mathrm{F}(y\mathbf{x}) = y\mathrm{F}(\mathbf{x})$ *for all* $y \in \mathbb{R}$ *and* $\mathbf{x} \in \mathbb{R}^n$. *Furthermore,* F *is positive-homogeneous if* $\mathrm{F}(y\mathbf{x}) = y\mathrm{F}(\mathbf{x})$ *for all* $y \in \mathbb{R}^+$ *and* $\mathbf{x} \in \mathbb{R}^n$.

If an aggregation operator is homogeneous and shift-invariant, then one can safely change the scale of the utility values by using a positive affine transformation without changing the preference relation induced by the operator.

1.2.1 Some Aggregation Criteria

We now present some standard classes of aggregation criteria by starting with the one of Weighted Averages (WAs).

The Weighted Average Operator. WAs compose the most simple class of operators with an averaging behavior that we can think of. A WA consists in doing a weighted sum of the utility values of a solution where for each $i \in \{1, \ldots, n\}$ the weight w_i corresponds to the relative importance of the criterion i (the higher the more important).

Definition 16. *Let* $\mathbf{w} = (w_1, \ldots, w_n)$ *be a vector of positive weights summing up to one. The* $\mathtt{WA}_{\mathbf{w}}(\cdot)$ *operator induced by* $\mathbf{w}$ *is defined by:*

$$\forall \mathbf{x} \in \mathbb{R}^n, \mathtt{WA}_{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i$$

A WA operator is obviously idempotent, averaging, monotone nondecreasing, shift invariant and homogeneous. It is symmetric if all the weights w_i are identical and equal to $1/n$ and it is compatible with Pareto dominance if the weights w_i are strictly positive. These good properties as well as the simplicity of this aggregation operator makes the WA operator an attractive decision criterion. However, WAs reveal quite poor from a descriptive point of view, as illustrated by the following example.

Example 2. *The employee of a youth hostel have identified 3 television subscription offers that seem interesting to them. These three offers are denoted by* $\mathbf{x}$, $\mathbf{y}$ *and* $\mathbf{z}$. *The two criteria considered for this choice are the respective quality of these offers w.r.t. sports and arts. While* $\mathbf{x}$ *is specialized in sports,* $\mathbf{z}$ *is only devoted to arts. The offer* $\mathbf{y}$ *is more balanced and contains TV programs on both sports and arts. Thus,* $\mathscr{A}$ *is here composed of three Pareto optimal solutions:* $\mathbf{x} = (10, 2)$, $\mathbf{y} = (5, 5)$ *and* $\mathbf{z} = (2, 10)$. *If the employee of the youth hostel use a WA to choose they will never consider solution* $\mathbf{y}$. *Indeed, if* $w_1 > w_2$, $\mathbf{x}$ *will be the preferred solution whereas if* $w_1 < w_2$, $\mathbf{z}$ *will be the preferred solution. Lastly, if* $w_1 = w_2$, *both* $\mathbf{x}$ *and* $\mathbf{z}$ *will be optimal solutions but not* $\mathbf{y}$. *This is because* $\mathbf{x}$ *and* $\mathbf{z}$ *are supported solutions, i.e., extreme points of the convex hull of the set of solution vectors, whereas* $\mathbf{y}$ *is not. An optimal solution according to a WA is always a supported solution. And yet* $\mathbf{y}$ *seems to be the best compromise. This example shows the poverty of the WA operator w.r.t. the descriptive viewpoint.*

Therefore, other operators with better descriptive abilities are required.

The Ordered Weighted Average Operator. To represent preferences that cannot be accommodated by WAs, one can use weights that are applied to different criterion according to the ranking of the corresponding utility values. This is the idea of Ordered Weighted Averages (OWAs) [Yager, 1988].

Definition 17. *Let* $\mathbf{w} = (w_1, \ldots, w_n)$ *be a vector of positive weights summing up to one. The* $\mathtt{OWA}_{\mathbf{w}}(\cdot)$ *operator induced by* $\mathbf{w}$ *is defined by:*

$$\forall \mathbf{x} \in \mathbb{R}^n, \mathtt{OWA}_{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{n} w_i x_{\boldsymbol{\sigma}(i)}$$

where $\boldsymbol{\sigma}$ *is a permutation of* $\{1, \ldots, n\}$ *such that* $x_{\boldsymbol{\sigma}(1)} \leq x_{\boldsymbol{\sigma}(2)} \leq \ldots \leq x_{\boldsymbol{\sigma}(n)}$.

The definition of the OWA operator can be extended to the domain $\cup_{m\in\mathbb{N}}\mathbb{R}^m$ *in the following way. Consider a sequence* $\mathbf{w}^* = (\mathbf{w}^{(m)})_{m\in\mathbb{N}}$ *of weight vectors* $\mathbf{w}^{(m)} \in [0,1]^m$*, such that:*

$$\forall\, m \in \mathbb{N},\ \sum_{i=1}^{m} w_i^{(m)} = 1.$$

Then, the ordered weighted averaging operator $\mathtt{OWA}_{\mathbf{w}^*}$ *is the operator defined on* $\cup_{m\in\mathbb{N}}\mathbb{R}^m$ *such that, for all* $m \in \mathbb{N}$*, the restriction of* $\mathtt{OWA}_{\mathbf{w}^*}$ *to* $\mathbb{R}^m$ *is* $\mathtt{OWA}_{\mathbf{w}^{(m)}}$.

The OWA operator has been characterized in several ways [Fodor *et al.*, 1995; Marichal and Mathonet, 1999]. We give here a characterization by Marichal and Mathonet [1999].

Theorem 1 (Marichal and Mathonet [1999])**.** *The aggregation operator* M *defined on* $\cup_{m\in\mathbb{N}}\mathbb{R}^m$ *is an ordered weigthed averaging operator if and only if it fulfils the following conditions, where* $\mathrm{M}^{(m)}$ *is the restriction of* M *to* $\mathbb{R}^m$*:*

- *For all* $m \in \mathbb{N}$, $\mathrm{M}^{(m)}$ *is a symmetric function on* $\mathbb{R}^m$.
- *For all* $m \in \mathbb{N}$, $\mathrm{M}^{(m)}$ *is nondecreasing in each of its arguments:*

  $$\forall\, i \in \{1, \ldots, m\}, \forall\, x_1, \ldots, x_m, x_i' \in \mathbb{R},$$
  $$x_i < x_i' \Rightarrow \mathrm{M}^{(m)}(x_1, \ldots, x_i, \ldots, x_m) \leq \mathrm{M}^{(m)}(x_1, \ldots, x_i', \ldots, x_m).$$

- *For all* $m \in \mathbb{N}$, $\mathrm{M}^{(m)}$ *is stable by positive linear transformations:*

  $$\mathrm{M}^{(m)}(r\,x_1 + t, \ldots, r\,x_m + t) = r\mathrm{M}^{(m)}(x_1, \ldots, x_m) + t$$

  for all $(x_1, \ldots x_m) \in \mathbb{R}^m$*, all* $r > 0$*, and all* $t \in \mathbb{R}$.

- $\mathrm{M}^{(1)}(x) = x$, $\forall\, x \in \mathbb{R}$ *and for all* $m, p \in \mathbb{N}$*:*

  $$\mathrm{M}^{(p)}(\mathrm{M}^{(m)}(x_{11}, \ldots, x_{1m}), \ldots, \mathrm{M}^{(m)}(x_{p1}, \ldots, x_{pm})) = \mathrm{M}^{(m)}(\mathrm{M}^{(p)}(x_{11}, \ldots, x_{p1}), \ldots, \mathrm{M}^{(p)}(x_{1m}, \ldots, x_{pm}))$$

  for all ordered matrices:

  $$\mathrm{X} = \begin{pmatrix} x_{11} & \ldots & x_{1m} \\ \vdots & \vdots & \vdots \\ x_{p1} & \ldots & x_{pm} \end{pmatrix} \in \mathbb{R}^{p\times m}$$

  where X *is ordered if its elements satisfy* $x_{ij} \leq x_{kl}$ *whenever* $i \leq k$ *and* $j \leq l$.

The last condition in Theorem 1 is called bisymmetry and has been justified by Marichal and Mathonet [1999].

An OWA operator is obviously idempotent, symmetric, averaging, monotone non-decreasing, shift invariant and positive-homogeneous. Moreover, it is compatible with Pareto dominance if the weights w_i are strictly positive. OWA operators are very general operators and encompass the average, the minimum and the maximum operators by using the weight vectors $(1/n,\ldots,1/n)$, $(1,0\ldots,0)$ and $(0\ldots,0,1)$. Moreover, when $w_i \gg w_{i+1}$ for all i, the OWA operator behaves as the leximin operator [Dubois *et al.*, 1997] which entails the preference relation $\succsim_{leximin}$ defined as follows:

Definition 18. *Let* $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$ *and let* $\boldsymbol{\sigma}_x$ *and* $\boldsymbol{\sigma}_y$ *be the permutations of* $\{1,\ldots,n\}$ *such that:*

$$x_{\boldsymbol{\sigma}_x(1)} \leq x_{\boldsymbol{\sigma}_x(2)} \leq \ldots \leq x_{\boldsymbol{\sigma}_x(n)} \text{ and } y_{\boldsymbol{\sigma}_y(1)} \leq y_{\boldsymbol{\sigma}_y(2)} \leq \ldots \leq y_{\boldsymbol{\sigma}_y(n)}$$

then, $x \succsim_{leximin} y$ *if and only if:*

$$\exists i \in \{1,\ldots,n\} \text{ such that, } x_{\boldsymbol{\sigma}_x(i)} > y_{\boldsymbol{\sigma}_y(i)} \text{ and } \forall j \in \{1,\ldots,i-1\}, x_{\boldsymbol{\sigma}_x(j)} = y_{\boldsymbol{\sigma}_y(j)}$$

Note that as the OWA is a symmetric operator it may not be suited for some multi-criteria decision problems. However, on the positive side, OWAs make it possible to express preferences that cannot be accommodated by a WA.

Example 3. *[Example 2 continued] We recall that with a WA operator, solution* $\mathbf{y}$ *could never be considered as an optimal solution. And yet, the television offer* $\mathbf{y}$ *seemed to be the best compromise as solutions* $\mathbf{x}$ *or* $\mathbf{z}$ *were completely devoted either to sports or arts but not both. Interestingly, by using an OWA operator with* $w_1 > 5/8$*, we obtain* $\mathbf{y}$ *as only optimal solution.*

OWA operators have been studied on an algorithmic point of view for fair allocation problems [Goldsmith and Sloan, 2007; Lesca and Perny, 2010; Ogryczak, 2000] or for voting procedures [Elkind and Ismaili, 2015]. However, they often lead to hard optimization problems because of the reordering induced by this operator.

We now present *mixture operators*. Mixture operators resemble OWA operators but their weights depend on the *values* that are at stake in the evaluated vector and not on the *ranks* of the components. This class of criteria is more detailed than the others as the mixture operator criterion is much studied in this thesis.

The Mixture Operator. In a nutshell, Mixture Operators (MOs) are like weighted averages where the numerical weights are replaced by weighting functions.

Definition 19. *Let* $w : \mathbb{R} \to (0,\infty]$ *be a positive weighting function. The* mixture operator $\mathrm{M}_w(\cdot)$ *induced by function* w *is defined as follows:*

$$\forall \mathbf{x} \in \mathbb{R}^n, \mathrm{M}_w(\mathbf{x}) = \sum_{i=1}^{n} \frac{w(x_i)}{\sum_{j=1}^{n} w(x_j)} x_i$$

MOs are special instances of Losonczi means [Losonczi, 1973], Bajraktarević means and generalized mixture functions [Ribeiro and Pereira, 2003]. We make this point clearer in the following definition:

Definition 20. *Let* $\mathtt{D} \subset \mathbb{R}$ *be an interval of* $\mathbb{R}$*. Let* $f : \mathtt{D} \to \mathbb{R}$ *be a continuous and strictly monotonic function and let* $w : \mathtt{D} \to]0, \infty[$ *be a strictly positive-valued function. The* n*-ary quasi-arithmetic mean generated by* f *with weight function* w *(also called Bajraktarević mean) is the function* $\mathrm{M}_{f,w} : \mathtt{D}^n \to \mathtt{D}$ *defined as:*

$$\mathrm{M}_{f,w}(\mathbf{x}) = f^{-1}\left(\frac{\sum_{i=1}^{n} w(x_i) f(x_i)}{\sum_{i=1}^{n} w(x_i)}\right).$$

While the Losonczi means are obtained by allowing function w *to change according to the index* i*, the mixture operators are obtained when* f *is the identity function, i.e.,* $f(x) = x$*. The definition of the Bajraktarević means can be extended to the domain* $\cup_{n\in\mathbb{N}}\mathtt{D}^n$ *in the following way. The extended quasi-arithmetic mean generated by* f *with weight function* w *is the function* $\mathrm{M}_{f,w} : \cup_{n\in\mathbb{N}}\mathtt{D}^n \to \mathtt{D}$ *whose restriction to* $\mathtt{D}^n$ *is the* n*-ary quasi-arithmetic mean generated by* f *with weight function* w*.*

The extended quasi-arithmetic means with weight function have been axiomatized in the following way by Páles [1987].

Theorem 2 (Páles [1987])**.** *Assume* $\mathtt{D}$ *is open. The function* $\mathrm{F} : \cup_{n\in\mathbb{N}}\mathtt{D}^n \to \mathbb{R}$ *is an extended quasi-arithmetic mean with weight function if and only if the following properties are satisfied:*

- $\mathrm{F}(x) = x,\ \forall x \in \mathtt{D}$.
- F *is symmetric.*
- *For any* $x < u < v < y$ *in* $\mathtt{D}$*, there can be found* $n, m \in \mathbb{N}$ *such that:*

$$u < \mathrm{F}(n \cdot x, m \cdot y) < v.$$

- *For any* $n, m \in \mathbb{N} \setminus \{0\}$ *and any* $\mathbf{x} \in \mathtt{D}^n$ *and* $\mathbf{y} \in \mathtt{D}^m$*,*

$$\lim_{k\to\infty} \mathrm{F}(k \cdot x_1, \ldots, k \cdot x_n, \mathbf{y}) = \mathrm{F}(\mathbf{x}).$$

- *For any* $n, m \in \mathbb{N} \setminus \{0\}$ *and any* $k, l \in \mathbb{N}$*, and any* $\mathbf{x} \in \mathtt{D}^n$*,* $\mathbf{y} \in \mathtt{D}^m$*,* $\mathbf{u} \in \mathtt{D}^k$ *and* $\mathbf{v} \in \mathtt{D}^l$*, the inequalities:*

$$\mathrm{F}(\mathbf{x}, \mathbf{u}) \leq \mathrm{F}(\mathbf{x}, \mathbf{v}) \text{ and } \mathrm{F}(\mathbf{y}, \mathbf{u}) \leq \mathrm{F}(\mathbf{y}, \mathbf{v})$$

  imply:

$$\mathrm{F}(\mathbf{x}, \mathbf{u}, \mathbf{y}, \mathbf{u}) \leq \mathrm{F}(\mathbf{x}, \mathbf{v}, \mathbf{y}, \mathbf{v}).$$

On the other hand, MOs extend several averaging operators as the Gini mean or the Lehmer mean [Beliakov *et al.*, 2015]. Note that if function w is constant then the MO boils down to the average operator. Furthermore, the special case of the Lehmer mean is defined by $w(x) = x^{p-1}$ where p is a parameter. Put another way, the Lehmer mean L_p is defined by:

$$\forall \mathbf{x} \in \mathbb{R}^n,\ \mathrm{L}_p(\mathbf{x}) = \frac{\sum_{i=1}^{n} x_i^p}{\sum_{i=1}^{n} x_i^{p-1}}$$

If p tends towards $-\infty$ (resp. $+\infty$), then the Lehmer mean tends towards the min (resp. max) operator.

The following example illustrates the way an MO distorts the weights associated to each criteria.

Example 4. *We assume that the utilities take values in an interval* $(0, \mathrm{U})$ *and we consider the weight function* w *defined by* $w(x) = 2\mathrm{U} - x$*. Then given a solution* $\mathbf{x} \in \mathscr{A}$*, the weight* w_i *associated to criterion* i *is given by:*

$$w_i = \frac{w(x_i)}{\sum_{j=1}^{n} w(x_j)} = \frac{2\mathrm{U} - x_i}{\sum_{j=1}^{n} 2\mathrm{U} - x_j} = \frac{2\mathrm{U} - x_i}{n(2\mathrm{U} - \mathrm{E})}$$

where $\mathrm{E} = \sum_{j=1}^{n} x_j / n$ *is the arithmetic mean of vector* $\mathbf{x}$*. Thus,* $w_i \geq$ *(resp.* $\leq$*)* $1/n$ *if* $x_i \leq$ *(resp.* $\geq$*)* E*. Stated differently, if the utility value of a criterion* i *is less than the average utility of the criteria, then criterion* i *is given a greater weight in operator* M_w*. More generally, the more* w *is decreasing, the more operator* M_w *will focus on the least satisfied criteria.*

An MO is obviously idempotent, symmetric and averaging. Note that as the MO is a symmetric operator it may not be suited for some multi-criteria decision problems. The more general class of Losonczi means [Losonczi, 1973] can be investigated to obtain non symmetric operators. The study of the mathematical properties of MOs (e.g., monotonicity, orness, ...) is an active research topic [Liu, 2010; Mesiar *et al.*, 2008; Ribeiro and Pereira, 2003]. Many works have been particularly focused on the monotonicity of MOs [Beliakov and Špirková, 2016; Beliakov and Wilkin, 2014]. Indeed, MOs are not monotone nondecreasing and compatible with Pareto dominance in general. For this reason, sufficient conditions to ensure the monotonicity of this operator have been studied. For instance, if the utilities take values in an interval $(0, \mathrm{U})$ and if $w(x) \geq \frac{dw}{dx}(x)(\mathrm{U} - x)$ for all $x \in (0, \mathrm{U})$ with w increasing and piecewise differentiable, then M_w is increasing [Mesiar and Špirková, 2006]. A simpler condition to impose monotonicity, when utilities are positive, is to have function $x \to w(x)$ decreasing and function $x \to w(x)x$ increasing. A more general result is given by Chew [1983].

Proposition 1. *(Chew [1983]) We assume that the utility values are defined on an open interval* (a, b)*. Assume that functions* $x \to w(x)$ *and* $x \to w(x)x$ *are continuous and bounded, then operator* $\mathrm{M}_w(\cdot)$ *is monotone nondecreasing (resp. compatible with Pareto dominance) iff function* $x \to w(x)(x - y)$ *is increasing (resp. strictly increasing) on* (a, b) *for all* y *in* (a, b)*.*

Proof. We give the sketch of the proof for compatibility with Pareto dominance. Adapting the proof for monotone nondecreasingness is trivial.
Sufficiency: Note that given vectors $\mathbf{x}, \mathbf{y} \in (a, b)^n$:

$$\begin{aligned}
\mathrm{M}_w(\mathbf{x}) \geq \mathrm{M}_w(\mathbf{y}) &\Leftrightarrow \sum_{i=1}^{n} \frac{w(x_i)}{\sum_{j=1}^{n} w(x_j)} x_i \geq \sum_{i=1}^{n} \frac{w(y_i)}{\sum_{j=1}^{n} w(y_j)} y_i \\
&\Leftrightarrow \sum_{i=1}^{n} \sum_{j=1}^{n} w(y_j) w(x_i) x_i \geq \sum_{i=1}^{n} \sum_{j=1}^{n} w(y_i) w(x_j) y_i \\
&\Leftrightarrow \sum_{i=1}^{n} \sum_{j=1}^{n} w(y_j) w(x_i)(x_i - y_j) \geq 0
\end{aligned}$$

In particular, for any vector $\mathbf{x} \in (a, b)^n$,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} w(x_j) w(x_i)(x_i - x_j) = 0.$$

Assume now that function $x \to w(x)(x - y)$ is strictly increasing on (a, b) for all y in (a, b), then function $\phi_{\mathbf{y}} : x \to \sum_{j=1}^{n} w(y_j) w(x)(x - y_j)$ is also strictly increasing on (a, b) for all $\mathbf{y}$ in

$(a,b)^n$ as $w(y_j)>0$ for all $j\in\{1,\dots,n\}$. Hence, if $\mathbf{x}^\epsilon$ is obtained from $\mathbf{x}\in(a,b)^n$ by adding ϵ to one of the utility values, then:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w(x_j)\,w(x_i^\epsilon)(x_i^\epsilon-x_j) > \sum_{i=1}^{n}\sum_{j=1}^{n} w(x_j)\,w(x_i)(x_i-x_j)=0$$

Thus, $\mathrm{M}_w(\mathbf{x}^\epsilon)>\mathrm{M}_w(\mathbf{x})$ and, by repeating this argument for all improved utility values, the MO is compatible with Pareto dominance.
Necessity: By contradiction, assume there exists $s\in(a,b)$ such that $x\to w(x)(x-s)$ is not strictly increasing. Thus, there exists $\hat{x}$ and $\epsilon>0$ such that $[\hat{x}-\epsilon,\hat{x}+\epsilon]\subset(a,b)$ and $\phi_s: x\to w(s)w(x)(x-s)$ is decreasing on $[\hat{x}-\epsilon,\hat{x}+\epsilon]$. Assume w.l.o.g. that $s>\hat{x}$ and consider $t\in(a,b)$ such that $t>s$. As $\mathbb{Q}$ is dense in $\mathbb{R}$ and as w is bounded and continuous on (a,b), t can be chosen such that there exists $k,l\in\mathbb{N}^*$ with $\phi_s(\hat{x})+(k/l)\phi_s(t)=0$. Set $n=(l+k)$ and consider vectors $\mathbf{x}=(\underbrace{\hat{x},\hat{x},\dots,\hat{x}}_{l},\underbrace{t,\dots,t}_{k})$ and $\mathbf{s}=(\underbrace{s,\dots,s}_{(l+k)})$. Then, by construction, $\mathrm{M}_w(\mathbf{x})=\mathrm{M}_w(\mathbf{s})$. Consider $\mathbf{x}^\epsilon$ obtained from $\mathbf{x}$ by adding ϵ to one of the utility values equal to $\hat{x}$. As ϕ_s is decreasing on $[\hat{x}-\epsilon,\hat{x}+\epsilon]$, we have $\mathrm{M}_w(\mathbf{x}^\epsilon)\le\mathrm{M}_w(\mathbf{s})=\mathrm{M}_w(\mathbf{x})$ which violates Pareto dominance and concludes the proof. □

We now discuss the descriptive abilities of MOs.

Example 5 (Example 3 continued)**.** *As OWAs, MOs can represent preferences that WAs cannot accommodate and have optimal solutions that are not supported. For instance, consider the decision problem presented in Example 2 and the MO defined by the weighting function $w: x\to 20-x$. We obtain $\mathrm{M}_w(\mathbf{x})=\mathrm{M}_w(\mathbf{z})\approx 4.86<\mathrm{M}_w(\mathbf{y})=5$ and $\mathbf{y}$ is the only optimal solution.*

Now, one can investigate if one class of aggregation operator between MOs and OWAs has more descriptive abilities than the other.

Comparison of OWAs and MOs on the descriptive point of view. We have no theoretical argument to support more one operator than the other from a descriptive point of view. In fact, examples of preferences that can be accounted with one class of operator and not with the other can be found in both cases.

We start by giving an example where MOs are able to account for preferences that cannot be represented by OWA or WA operators.

Example 6. *We consider a problem with two criteria (i.e., $n=2$) and we consider four feasible solutions $\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_1'$ and $\mathbf{x}_2'$ such that:*

$$\mathbf{x}_1=(4,8)\qquad \mathbf{x}_2=(2,12)$$
$$\mathbf{x}_1'=(8,8)\qquad \mathbf{x}_2'=(6,12).$$

Note that $\mathbf{x}_1'$ and $\mathbf{x}_2'$ are obtained from $\mathbf{x}_1$ and $\mathbf{x}_2$ by adding 4 to the utility value of the first criterion. In solution $\mathbf{x}_2$, criterion 1 obtains a very low score, so one could prefer the more balanced solution $\mathbf{x}_1$ even if the average utility value of the solution is lower with $\mathbf{x}_1$ than with $\mathbf{x}_2$. Conversely, in both solutions $\mathbf{x}_1'$ and $\mathbf{x}_2'$, criteria 1 and 2 obtain scores that are reasonably high. Hence, one could prefer $\mathbf{x}_2'$ to $\mathbf{x}_1'$ as it yields the highest average utility value. It is easy to see that accounting for both of these preferences is not possible with a WA operator nor an OWA operator. Indeed, whatever the values of the weights $\mathbf{w}=(w_1,w_2)$:

$$\mathtt{WA}_{\mathbf{w}}(\mathbf{x}_1')=\mathtt{WA}_{\mathbf{w}}(\mathbf{x}_1)+4w_1\quad \mathtt{WA}_{\mathbf{w}}(\mathbf{x}_2')=\mathtt{WA}_{\mathbf{w}}(\mathbf{x}_2)+4w_1$$
$$\mathtt{OWA}_{\mathbf{w}}(\mathbf{x}_1')=\mathtt{OWA}_{\mathbf{w}}(\mathbf{x}_1)+4w_1\quad \mathtt{OWA}_{\mathbf{w}}(\mathbf{x}_2')=\mathtt{OWA}_{\mathbf{w}}(\mathbf{x}_2)+4w_1$$

Hence, if preferences are represented by a WA or an OWA, the preference holding between $\mathbf{x}_1$ *and* $\mathbf{x}_2$ *should be the same as the one holding between* $\mathbf{x}'_1$ *and* $\mathbf{x}'_2$*. However, by considering* $w(x) = 24 - x$*, one obtains:*

$$\begin{aligned} \mathrm{M}_w(\mathbf{x}_1) &\approx 5.78 \qquad \mathrm{M}_w(\mathbf{x}_2) \approx 5.53 \\ \mathrm{M}_w(\mathbf{x}'_1) &= 8 \qquad \mathrm{M}_w(\mathbf{x}'_2) \approx 8.4 \end{aligned}$$

which is consistent with the desired preferences.

We now give an example where OWAs are able to account for preferences that cannot be represented by MOs.

Example 7. *We consider a problem with four criteria (i.e.,* $n = 4$*) and we consider three solutions,* $\mathbf{x}_1 = (4,4,8,8)$*,* $\mathbf{x}_2 = (2,2,10,10)$ *and* $\mathbf{x}_3 = (2,4,8,10)$*. If one wants to be the best as possible on three criteria out of four, one might strictly prefer solution* $\mathbf{x}_3$ *to* $\mathbf{x}_1$ *and* $\mathbf{x}_2$*. Representing such a preference is not possible with an MO. Indeed, whatever the weighting function* w*:*

$$\mathrm{M}_w(\mathbf{x}_3) = \frac{2w(2) + 4w(4) + 8w(8) + 10w(10)}{w(2) + w(4) + w(8) + w(10)}$$

$$\mathrm{M}_w(\mathbf{x}_3) = \frac{w(4) + w(8)}{w(2) + w(4) + w(8) + w(10)}\mathrm{M}_w(\mathbf{x}_1) + \frac{w(2) + w(10)}{w(2) + w(4) + w(8) + w(10)}\mathrm{M}_w(\mathbf{x}_2)$$

Thus, $\mathrm{M}_w(\mathbf{x}_3)$ *is a convex combination of* $\mathrm{M}_w(\mathbf{x}_1)$ *and* $\mathrm{M}_w(\mathbf{x}_2)$*. Therefore, if* $\mathrm{M}_w(\mathbf{x}_1) > \mathrm{M}_w(\mathbf{x}_2)$ *(resp.* $\mathrm{M}_w(\mathbf{x}_2) > \mathrm{M}_w(\mathbf{x}_1)$*), then* $\mathrm{M}_w(\mathbf{x}_1) > \mathrm{M}_w(\mathbf{x}_3)$ *(resp.* $\mathrm{M}_w(\mathbf{x}_2) > \mathrm{M}_w(\mathbf{x}_3)$*). However, by considering the OWA operator with weight vector* $\mathbf{w} = (1,4,3,2)$*, one obtains the desired preferences:*

$$\mathtt{OWA}_{\mathbf{w}}(\mathbf{x}_1) = 60 \quad \mathtt{OWA}_{\mathbf{w}}(\mathbf{x}_2) = 60 \quad \mathtt{OWA}_{\mathbf{w}}(\mathbf{x}_3) = 62.$$

1.3 Multi-Agent Decision Making

Multi-agent decision problems are decision problems where several agents are involved. They can be seen as multicriteria decision problems where criterion i corresponds to the utility of the solution w.r.t. the i^{th} agent.

Example 8. *Laurent is going to the super-market to buy food for his family composed of 5 persons (i.e.,* $n = 5$*). He has enough room in his bag to buy five products. Stated differently, the set of solutions* $\mathscr{A}$ *is composed of all possible combinations of 5 products available at the super-market. However, the tastes of the 5 family members differ and therefore, the utility of a product is not the same depending on the person considered. Consequently, the same can be said about the utilities of the solutions. To avoid unwanted tensions at home, Laurent will have to choose carefully a solution that satisfies everyone.*

Example 9. *Agathe receives the visit of her two brothers Adrien and Hugo in Lyon. She decides to benefit from the way back from the train station to her flat to show them the city. Unfortunately, Adrien and Hugo do not want to visit the same places. Agathe knows three paths* p_1*,* p_2 *and* p_3 *to go back to her flat with following utilities for her brothers (Table 1.1).*

Agathe will probably be tempted to choose path p_3 *as it is the most equitable solution.*

	p_1	p_2	p_3
Utilities for Adrien and Hugo	(8,2)	(1,9)	(4,5)

Table 1.1: Utilities of the different paths for Adrien and Hugo.

This specificity implies some requirements on the aggregation operator used. Above all, the aggregation operator used should be anonymous (i.e., symmetric) as it should treat equally all the agents[3]. Moreover, it is often desirable that the aggregation operator favors solutions that are fair. Several mathematical concepts have been proposed to define fairness as envy-freeness [Arnsperger, 1994], proportionality [Brams and Taylor, 1996], minmax fairness [Golovin, 2005] or compatibility with the Pigou-Dalton transfer principle [Moulin, 1991]. In this thesis, we will consider that an operator is fair if it favors solutions whose utility vector is well balanced. This idea is formalized by the Pigou-Dalton transfer principle:

Definition 21 (Moulin [1991])**.** Pigou-Dalton Transfer Principle*: let* $\mathbf{x} \in \mathbb{R}^n$ *such that* $x_i > x_j$ *for some* i, j*. Then, for all* ϵ *such that* $0 < \epsilon < x_i - x_j$*,* $\mathbf{x} - \epsilon\mathbf{b}_i + \epsilon\mathbf{b}_j$ *should be strictly preferred to* $\mathbf{x}$ *where* $\mathbf{b}_i$ *and* $\mathbf{b}_j$ *are the vector whose* i^{th} *(resp.* j^{th}*) component equals 1, all others being null.*

The transfer principle states that a transfer from a "more satisfied" agent to a "less satisfied" agent should improve a solution. Indeed, such a transfer reduces inequality while keeping constant the arithmetic mean of the utility vector. **We will say that an operator is *fair* if it satisfies the Pigou-Dalton transfer principle.**

We now recall the definition and some properties of *Lorenz vectors*, which is another key concept in inequality measurement, related to the Pigou-Dalton transfer principle (see Theorem 3).

Definition 22 (Marshall *et al.* [1979])**.** *Given a vector* $\mathbf{y} = (y_1, \ldots, y_n)$*, the* Lorenz vector *of* $\mathbf{y}$ *is defined by* $\mathscr{L}(\mathbf{y}) = (l_1(\mathbf{y}), \ldots, l_n(\mathbf{y}))$*, where* $l_i(\mathbf{y})$ *is the sum of the* i *smallest elements of* $\mathbf{y}$*. More formally, let* $\boldsymbol{\sigma}$ *be the permutation of* $\{1, \ldots, n\}$ *such that* $y_{\boldsymbol{\sigma}(1)} \leq y_{\boldsymbol{\sigma}(2)} \leq \ldots \leq y_{\boldsymbol{\sigma}(n)}$*, then* $l_i(\mathbf{y}) = \sum_{j=1}^{i} y_{\boldsymbol{\sigma}(j)}$*.*

A vector $\mathbf{y}$ is said to *Lorenz-dominate* a vector $\mathbf{y}'$ if $\mathscr{L}(\mathbf{y})$ Pareto-dominates $\mathscr{L}(\mathbf{y}')$. The *Lorenz curve* associated with a vector $\mathbf{y}$ is the piecewise-linear curve joining points $(0,0), (1, l_1(\mathbf{y})), \ldots, (n, l_n(\mathbf{y}))$ in the bi-dimensional Cartesian coordinate system. Graphically, a vector $\mathbf{y}$ Lorenz-dominates a vector $\mathbf{y}'$ iff the Lorenz curve of $\mathbf{y}$ is above the one of $\mathbf{y}'$. Lorenz dominance is illustrated in Example 10. Lorenz dominance is a key concept in inequality measurement due to the following result:

Theorem 3. *[Chong, 1976] For any pair of distinct positive vectors* $\mathbf{y}, \mathbf{y}'$*, if* $\mathbf{y}$ *Pareto-dominates* $\mathbf{y}'$*, or if* $\mathbf{y}'$ *is obtained from* $\mathbf{y}$ *by a Pigou-Dalton transfer, then* $\mathbf{y}$ *Lorenz-dominates* $\mathbf{y}'$*. Conversely, if* $\mathbf{y}$ *Lorenz-dominates* $\mathbf{y}'$*, then there exists a sequence of Pigou-Dalton transfers and/or Pareto-improvements to transform* $\mathbf{y}'$ *into* $\mathbf{y}$*.*

In other words, given two solutions $\mathbf{x}$ and $\mathbf{x}'$, if $\mathbf{x}$ Lorenz-dominates $\mathbf{x}'$, then $\mathbf{x}$ should be preferred to $\mathbf{x}'$ from the viewpoint of efficiency and fairness. The previous result also implies that any operator which is compatible with Pareto dominance and compatible with the Pigou-Dalton transfer principle is compatible with Lorenz dominance.

[3]However, note that there exists collective decision making situations in which agents may have exogenous rights [Bouveret and Lemaître, 2007].

Example 10. *Consider a multi-agent decision problem with 3 agents and three solutions* $\mathbf{x} = (2,4,1)$, $\mathbf{y} = (3,1,3)$ *and* $\mathbf{z} = (2,2,2)$. *The Lorenz curves associated to each solution are given in Figure 1.4.*

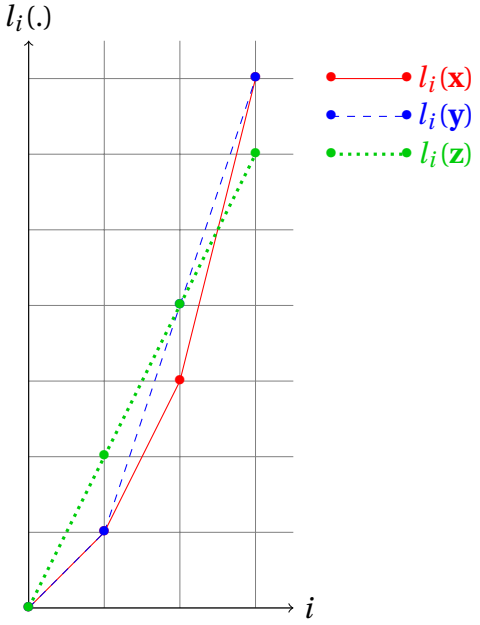


Figure 1.4: The three Lorenz curves associated to solutions $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$.

On this Figure, it is easy to see that solution $\mathbf{y}$ *Lorenz dominates solution* $\mathbf{x}$ *as the lorenz curve associated to* $\mathbf{y}$ *(in blue) is above the one of* $\mathbf{x}$ *(in red).*

Fairness and OWA operators. A well known class of fair operators whose optimization yields a Pareto-optimal solution is the class of OWA operators with strictly decreasing weights [Yager, 1988] (i.e., OWA operators such that $w_i > w_{i+1}$ for $i \in \{1,\ldots,n-1\}$).

OWA operators have an elegant connection with Lorenz curves. Indeed, an OWA operator can be rewritten as:

$$\forall \mathbf{x} \in \mathscr{A}, \mathtt{OWA}_{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{n} \omega_i l_i(\mathbf{x})$$

where $\omega = (w_1 - w_2, w_2 - w_3, \ldots, w_{n-1} - w_n, w_n)$. Thus, provided $w_i > w_{i+1}$ for $i \in \{1,\ldots,n-1\}$, a solution optimizing an OWA operator is always Lorenz optimal.

The weights initially proposed for a famous fair OWA operator, namely the *Gini social-evaluation function*, are [Moulin, 1991]:

$$w_i = (2(n-i)+1)/n^2 \tag{1.1}$$

With these weights, $\mathtt{OWA}_{\mathbf{w}}(\mathbf{x})$ has a nice graphical interpretation: it can indeed be shown that $1 - \mathtt{OWA}_{\mathbf{w}}(\mathbf{x})/\mu$, where $\mu = \sum_{i=1}^{n} x_i/n$, equals two times the area between the Lorenz curve of $\mathbf{x}$ and the diagonal representing the egalitarian distribution of utilities (where $x_i = \mu$ for every agent). Obviously, for a given μ, the narrower this area the better, which is consistent with the maximization of OWA.

Fairness and MOs. Similarly to OWA operators, a mathematical condition can be imposed on mixture operators to ensure their compatibility with the Pigou-Dalton transfer principle. This condition was first given in the context of decision making under risk (which will be presented in Section 1.5). Indeed, mixture operators are instances of both the Weighted Expected Utility (WEU) model [Chew, 1983] and the decomposable Skew-Symmetric Bilinear (SSB) functions axiomatized and investigated by Nakamura [1989]. The properties of WEU functions and SSB functions w.r.t. risk-sensitivity and stochastic dominance have been thoroughly investigated [Chew, 1983; Fishburn, 1984a; Nakamura, 1989] and those results can directly be used to entail results on the fairness of MOs (the SSB and WEU models are further presented in subsections 1.5.3 and 1.5.4). Indeed, the Pigou-Dalton transfer principle in inequality measurement coincides with consistency with second order stochastic dominance in decision making under risk.

Proposition 2. *(Chew [1983]) We assume that utilities take values in an open interval (a,b). Assume that functions $x \to w(x)$ and $x \to w(x)x$ as well as their first derivatives are continuous and bounded on (a,b), then operator $\mathrm{M}_w(\cdot)$ is fair iff function $x \to w(x)(x-y)$ is strictly concave on (a,b) for all y in (a,b).*

Proof. We give the sketch of the proof.
Sufficiency: Note that given vectors $\mathbf{x},\mathbf{y} \in (a,b)^n$:

$$\begin{aligned}
\mathrm{M}_w(\mathbf{x}) \geq \mathrm{M}_w(\mathbf{y}) &\Leftrightarrow \sum_{i=1}^{n} \frac{w(x_i)}{\sum_{j=1}^{n} w(x_j)} x_i \geq \sum_{i=1}^{n} \frac{w(y_i)}{\sum_{j=1}^{n} w(y_j)} y_i \\
&\Leftrightarrow \sum_{i=1}^{n}\sum_{j=1}^{n} w(y_j)w(x_i)x_i \geq \sum_{i=1}^{n}\sum_{j=1}^{n} w(y_i)w(x_j)y_i \\
&\Leftrightarrow \sum_{i=1}^{n}\sum_{j=1}^{n} w(y_j)w(x_i)(x_i - y_j) \geq 0
\end{aligned}$$

In particular, for any vector $\mathbf{x} \in (a,b)^n$,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w(x_j)w(x_i)(x_i - x_j) = 0.$$

Assume that function $x \to w(x)(x-y)$ is strictly concave on (a,b) for all y in (a,b), then function $\phi_{\mathbf{y}} : x \to \sum_{j=1}^{n} w(y_j)w(x)(x-y_j)$ is also strictly concave on (a,b) for all $\mathbf{y}$ in $(a,b)^n$ as $w(y) > 0$ for all y in (a,b). Hence, by using Lemma 2 from the work of Dasgupta *et al.* [1973], if $\mathbf{x}^\epsilon$ is obtained from $\mathbf{x} \in (a,b)^n$ by an ϵ-transfer (i.e., $\mathbf{x}^\epsilon = \mathbf{x} + \epsilon(\mathbf{b}_j - \mathbf{b}_i)$ with $0 < \epsilon < x_i - x_j$ and where $\mathbf{b}_i$ is the i^{th} canonical vector) then:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w(x_j)w(x_i^\epsilon)(x_i^\epsilon - x_j) > \sum_{i=1}^{n}\sum_{j=1}^{n} w(x_j)w(x_i)(x_i - x_j) = 0$$

Thus, $\mathrm{M}_w(\mathbf{x}^\epsilon) > \mathrm{M}_w(\mathbf{x})$ and the MO satisfies the Pigou-Dalton transfer principle.
Necessity: By contradiction, assume there exists $s \in (a,b)$ such that $x \to w(x)(x-s)$ is not strictly concave. Thus, there exists $\hat{x}$ and $\epsilon > 0$ such that $[\hat{x}-\epsilon, \hat{x}+\epsilon] \subset (a,b)$ and $\phi_s : x \to w(s)w(x)(x-s)$ is convex on $[\hat{x}-\epsilon, \hat{x}+\epsilon]$. Assume w.l.o.g. that $s > \hat{x}$ and consider $t \in (a,b)$ such that $t > s$. As $\mathbb{Q}$ is dense in $\mathbb{R}$ and as w is bounded and continuous on (a,b), t can be chosen such that there exists $k,l \in \mathbb{N}^*$ with $\phi_s(\hat{x}) + (k/l)\phi_s(t) = 0$. Set $n = 2(l+k)$ and consider vectors $\mathbf{x} = (\underbrace{\hat{x},\hat{x},\ldots,\hat{x}}_{2l},\underbrace{t,\ldots,t}_{2k})$ and $\mathbf{s} = (\underbrace{s,\ldots,s}_{2(l+k)})$. Then, by construction, $\mathrm{M}_w(\mathbf{x}) = \mathrm{M}_w(\mathbf{s})$. Consider $\mathbf{x}^\epsilon$ obtained by transferring ϵ from one of the $\hat{x}$ terms

to another (increasing inequality). As in the sufficiency part, as ϕ_s is convex on $[\hat{x}-\epsilon, \hat{x}+\epsilon]$, we have $M_w(\mathbf{x}^\epsilon) \geq M_w(\mathbf{s}) = M_w(\mathbf{x})$ which violates the Pigou-Dalton transfer principle and concludes the proof. □

For instance, if $(a,b) \subset \mathbb{R}^+$, a sufficient condition is to have function $x \to w(x)x$ concave and function $x \to w(x)$ convex (with at least one property being strict). Under this sufficient condition, it is easy to see that the MO will be fair, as a Pigou-Dalton transfer will increase (resp. decrease) $\sum_{i=1}^n w(x_i)x_i$ (resp. $\sum_{i=1}^n w(x_i)$).

If $(a,b) = (0, U)$, examples of MOs that are increasing and fair on (a,b) can be defined by using $w(x) = 1/(1+x)$ or $w(x) = (\alpha+1)U^\alpha - x^\alpha$ with $0 < \alpha \leq 1$. Indeed, in these cases, function $x \to w(x)$ is convex and decreasing on (a,b) and function $x \to w(x)x$ is strictly concave and increasing on (a,b).

In Chapter 7, we investigate fair multi-agent optimization problems. Our contributions concern both the OWA operators and the MOs. For OWA operators, we investigate randomized solutions of fair multi-agent optimization problems. We prove that the determination of a randomized optimal OWA solution is of polynomial complexity if the OWA has decreasing weights and if the sum of utilities of the agents can be optimized in polynomial time. Moreover, we give two generic methods to compute a randomized OWA optimal solution. For MOs, we investigate fair allocation problems and fair multi-winner voting procedures. We give methods to compute an optimal MO solution and show that determining an optimal MO solution is of polynomial complexity in the following two cases: (1) the multi-agent optimization problem is the assignment problem and (2) the multi-agent problem is a proportional representation problem with the Chamberlin-Courant's multi-winner voting scheme and the preferences of the agents abide to specific structures that we precise.

1.4 Robust Decision Making with Imprecise Parameters

In various decision problems, it is unreasonable to assume that all the parameters are precisely known. This imprecision can result from the fact that the parameters cannot be measured exactly. For instance, it may be possible that their values drift around some "standard" values. This uncertainty can affect different types of parameters as cost values [Montemanni, 2006; Regan and Boutilier, 2009] or probability values [Bagnell *et al.*, 2001]. Furthermore, we may allow all the parameters to vary independently or we may assume that the number of parameters that may vary is bounded by a constant [Bertsimas and Sim, 2003]. This imprecision over the parameters of the optimization problem induces a set of possible scenarios called *uncertainty set*.

Any kind of uncertainty set can be considered. However, we introduce the two most well known types of uncertainty sets: the *discrete scenario model* and the *interval model*.

For instance, instead of having one vector giving the values of each parameter, several vectors of parameter values may be possible. Each possible vector of parameter values is then called a *scenario*. This is known as the *discrete scenario model* where the possible scenarios consist in a finite set of parameter vectors.

Alternatively, instead of specifying scalar values, there are situations where only an interval of possible values is known for each parameter. It is then assumed that the different parameters may take any value of the given interval independently of the other parameters. As in the previous case, the assignment of a scalar value in each interval is called

a *scenario*. This is known as the *interval model* where each parameter p_i takes value in an interval $[\underline{p}_i, \overline{p}_i]$ and where the set of scenarios is defined implicitly as the Cartesian product $\mathscr{U} = \times_{i=1}^{m} [\underline{p}_i, \overline{p}_i]$, with m the number of parameters. We illustrate the discrete scenario model and the interval model in the following example.

Example 11. *Adrien needs to go from his house to the hospital. Three different paths p_1, p_2 and p_3 of different length can be used by car. We denote by c_1, c_2 and c_3 the time (in hours) required by paths p_1, p_2 and p_3 respectively and by* $\mathbf{c}$ *the vector* (c_1, c_2, c_3).

Discrete scenario model: Adrien knows that one of the three paths is a highway while the two others are standard national roads. Unfortunately, he does not remember which one. This leads to the following three discrete scenarios: $\mathbf{c} = (1, 0.2, 0.8)$, $\mathbf{c} = (0.4, 0.5, 0.8)$ *and* $\mathbf{c} = (1, 0.5, 0.32)$.

Interval scenario model: Adrien now remembers that the highway is path p_2. However, while p_3 is a quiet path, paths p_1 and p_2 can be very crowded which can highly impact c_1 and c_2. Unaware of the current traffic conditions, Adrien only knows the minimal and maximal values that can take c_1, c_2 and c_3: $c_1 \in [0.35, 0.5]$, $c_2 \in [0.45, 0.7]$ and $c_3 \in [0.78, 0.82]$. If we make the hypothesis that parameters c_1, c_2 and c_3 are independent, $\mathbf{c}$ *can be any vector in* $[0.35, 0.5] \times [0.45, 0.7] \times [0.78, 0.82]$.

One can then adopt a multicriteria point of view and consider that each possible scenario defines a criterion. The idea is then to find a solution whose value is good whatever the scenario that finally occurs. Such a solution is called a *robust* solution. This may lead to favor aggregation criteria that make it possible to find solutions whose utility vectors are well-balanced. This can be related to the concept of *ambiguity aversion* which is similar to the concept of fairness in multi-agent decision problems (see previous subsection). The link between robust optimization and ambiguity aversion is illustrated in Example 12.

Example 12. *Consider two urns, urn A and urn B. Urn A contains 50 blue balls and 50 red balls and urn B contains 100 balls of unknown color that can be either blue or red. The game consists in choosing an urn from which a ball is withdrawn. If this ball is blue, then the player wins* 10$. *At first glance, none of the two urns seems more desirable. Choosing urn A leads to an expected value of* 5$. *The expected value of choosing urn B is unknown and can be any value in* $\{0\$, 0.1\$, \ldots, 9.9\$, 10\$\}$. *However, if we assume that all scenarios are equiprobable, the expected value of choosing urn B would also be* 5$.

Ambiguity aversion is a preference for known risks over unknown risks. A decision maker which is averse to ambiguity would here always prefer to choose urn A. In a sense, choosing urn A is a more robust solution than choosing urn B as there is no uncertainty over the expected value of this choice whereas urn B could be filled with red balls leading to an expected value of 0$.

One possible approach to find a robust solution consists in optimizing the worst case performance (i.e., performance of the considered solution in the worst possible scenario). This approach leads to the *maxmin* criterion (or *minmax* criterion in a minimization problem) which consists of evaluating a solution on the basis of its worst value over all scenarios. However, this approach often leads to an overly conservative solution. A less conservative approach, which is known as minmax regret optimization, minimizes the maximum difference in the objective value of a solution over all scenarios, compared to the best possible objective value attainable in this scenario. Put another way, the *minmax regret* criterion consists of evaluating a solution on the basis of its maximal deviation from the optimal value over all scenarios.

More formally, we denote by $\mathscr{U}$ the set of possible scenarios (which may contain an infinite number of scenarios in the interval model). The maxmin (or minmax in a minimization problem) optimal solution $\mathbf{x}^*_{\text{MM}} \in \mathscr{A}$ and the minmax regret optimal solution $\mathbf{x}^*_{\text{MMR}} \in \mathscr{A}$ are defined by:

$$\mathbf{x}^*_{\text{MM}} = \max_{\mathbf{x}\in\mathscr{A}} \min_{i\in\mathscr{U}} x_i \qquad \mathbf{x}^*_{\text{MMR}} = \min_{\mathbf{x}\in\mathscr{A}} \max_{\mathbf{y}\in\mathscr{A}, i\in\mathscr{U}} y_i - x_i$$

in a maximization problem and by[4]:

$$\mathbf{x}^*_{\text{MM}} = \min_{\mathbf{x}\in\mathscr{A}} \max_{i\in\mathscr{U}} x_i \qquad \mathbf{x}^*_{\text{MMR}} = \min_{\mathbf{x}\in\mathscr{A}} \max_{\mathbf{y}\in\mathscr{A}, i\in\mathscr{U}} x_i - y_i$$

in a minimization problem. We illustrate the maxmin and the minmax regret criteria in the two following examples.

Example 13. *A company will soon receive a visit from important investors. To please them, the company decides to plan an activity during their stay. Unfortunately, the activities are very popular and it is therefore necessary to book them in advance. Furthermore, the interest of the different activities depends heavily on the weather conditions. The company has identified three possible activities: going to the cinema, going to a music festival and doing canoe-kayak. After studying the profiles of the investors, they have determined the following utility values for doing these activities depending on the weather conditions:*

	rainy	sunny	hot
cinema	4	4	4
music festival	1	6	5
canoe-kayak	3	6	7

Table 1.2: Utilities of the different activities according to the weather conditions.

For this problem, the maxmin optimal solution is choosing the cinema event as this solution has an utility value of 4 whatever the weather conditions whereas the two other activities have utilities strictly lower than 4 if the weather is rainy. However, the minmax regret optimal solution differs and consists in booking a canoe-kayak activity as this solution has a max regret of 1 (obtained if the weather is rainy) against a max regret of 3 for the two other solutions: for example, if we decide for the cinema (utility of 4) we may regret, if the weather is hot, not having chosen canoe-kayak (utility of 7).

Example 14. *Agathe wakes up the day of an important work meeting. Therefore, she wishes to arrive as soon as possible at work. She can take three different paths denoted by p_1, p_2 and p_3 to arrive at work. The travel time of each path depends on the traffic condition but the uncertainty is limited to an interval of possible travel time for each path. We assume that the traffic conditions on the three different paths are independent.*

	p_1	p_2	p_3
travel time interval	[25,30]	[20,60]	[10,35]

Table 1.3: Travel time intervals of the different paths in minutes.

[4]We here distinguish the minimization and maximization cases as the chapter related to robust optimisation adopts the minimization convention unstead of the maximization one which is used everywhere else in the thesis.

The optimal minmax solution of this problem is path p_1 with a maximum value of 30. The optimal minmax regret solution of this problem is path p_3 with a max regret value of 15 *minutes: in the worst case, she may regret not having taken path p_2 (20 minutes in the best case) instead of p_3 that may take up to 35 minutes.*

In Chapter 6, we investigate robust combinatorial optimization problems. Our contributions concern robust optimization problems with interval data and the minmax regret criterion. We investigate a lower bound on the optimal minmax regret, obtained by computing the value of a Nash equilibrium of a particular game that we specify. We present an anytime procedure to compute this lower bound and show how to efficiently use it in a branch and bound procedure. This approach is tested on the robust shortest path problem, for which a significant gain in time performances is obtained on some classes of instances.

1.5 Decision Making under Risk

As in robust decision making, we consider problems where the consequences of our choices are uncertain. However, we now consider a framework with more information as we assume that for each possible solution, we know a probability distribution over the possible consequences that can result from this solution. This defines the framework of decision making under risk, which is more specific than decision making under uncertainty. Indeed, in decision making under uncertainty, other tools than probabilities can be used, as possibilities and necessities [Dubois *et al.*, 1996; Sabbadin, 2001], belief functions [Dempster, 1967; Shafer and others, 1976], and/or subjective probabilities [Savage, 1972; Schmeidler, 1986].

Example 15. *Sabine has just bought a new mobile phone worth 700 €. The mobile phone agency suggests that she should take an insurance for one year. Sabine hesitates, she has read recently that in her neighborhood:*

- *the probability of breaking a mobile phone the first year is of* 20% *and,*
- *the probability of a mobile phone being stoled the first year is of* 10%.

Sabine has three choices (i.e., $\mathscr{A} = \{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}\}$). She can take no insurance (solution $\mathbf{x_1}$), take an insurance that covers material damages and which costs 240 € (solution $\mathbf{x_2}$) or take an insurance covering both material damages and thieveries which costs 360 € (solution $\mathbf{x_3}$). If the mobile phone is broken or stolen Sabine will have to buy it again (at the orignal price of 700 €) unless she took an insurance. The costs of each solution according to the possible events are given in Table 1.4.

	mobile not being stolen or broken probability 70%	mobile breaks probability 20%	mobile stolen probability 10%
$\mathbf{x_1}$	0	700	700
$\mathbf{x_2}$	240	240	700
$\mathbf{x_3}$	360	360	360

Table 1.4: Possible costs in euros of the three solutions.

The possible costs in euros are 0, 240, 360 and 700. The three solutions $\mathbf{x_1}$, $\mathbf{x_2}$ and $\mathbf{x_3}$ induce probability distributions l_1, l_2 and l_3 over these possible outcomes that are given in Table 1.5.

		Loss incurred			
	x	0 €	240 €	360 €	700 €
Lotteries associated	$l_1(x)$	0.7	0	0	0.3
with choices $\mathbf{x}_1$, $\mathbf{x}_2$	$l_2(x)$	0	0.9	0	0.1
and $\mathbf{x}_3$	$l_3(x)$	0	0	1	0

Table 1.5: Probability distributions over possible costs in euros induced by the three possible solutions.

Thus, the decision problem amounts to choose one of these probability distributions. According to the amount of risk that Sabine is ready to take she will favor different solutions (choosing l_3 being the less risky solution and choosing l_1 being the most risky one).

Probability distributions over gains/losses are known as "lotteries" in the literature of decision theory under risk. As illustrated by the previous example, in decision problems under risk, the objects that are considered are lotteries and mixture of lotteries and one important component of a lottery is the level of uncertainty that it induces.

Definition 23. *A lottery is a distribution of probability over a finite number of consequences. We will denote by $l = (x_1, p_1; \ldots; x_k, p_k)$ the lottery under which the probability of outcome x_i is p_i, with $\sum_{i=1}^{k} p_i = 1$ and $p_i \geq 0$ for all i in $\{1, \ldots, k\}$. We will also use the notation $l(x)$ to denote the probability with which outcome x is obtained with lottery l.*

Definition 24. *A mixture of lotteries (also called compound lottery) is a probability distribution over a finite number of lotteries. We will denote by $m = (l_1, p_1; \ldots; l_s, p_s)$ the mixture of lotteries under which the probability of lottery l_i is p_i, with $\sum_{i=1}^{s} p_i = 1$ and $p_i \geq 0$ for all i in $\{1, \ldots, s\}$. Note that each mixture of lotteries $m = (l_1, p_1; \ldots; l_s, p_s)$ induces a lottery l defined by $l(x) = \sum_{i=1}^{s} p_i l_i(x)$.*

As illustrated by example 15, the notion of risk is important in decision problems under risk. This notion of risk can be formalized by the notions of weak risk aversion and risk loving.

Definition 25. *Given a lottery $l = (x_1, p_1; \ldots; x_k, p_k)$ we denote by $\mathrm{EV}(l)$ the expected value of l (i.e., $\mathrm{EV}(l) = \sum_{i=1}^{k} x_i p_i$). A decision maker is said to be weakly risk-averse iff for any lottery l, the lottery that obtains $\mathrm{EV}(l)$ with certainty is preferred to l. On the opposite, she is said to be weakly risk-seeking iff for any lottery l, l is preferred to the lottery that obtains $\mathrm{EV}(l)$ with certainty.*

We now discuss how decision problems under risk are related to multicriteria decision problems. For this purpose, the concept of *decumulative functions* needs to be introduced.

Definition 26. *The decumulative function $\mathrm{G}_l : \mathbb{R} \to [0, 1]$ associated to a lottery $l = (x_1, p_1; \ldots; x_q, p_q)$ is defined by:*

$$\mathrm{G}_l(x) = \sum_{j \in \{1,\ldots,q\}: x_j \geq x} p_j.$$

In other words, the decumulative function G_l maps each value x to the probability of obtaining a value greater than or equal to x with lottery l. For each possible value of x, maximizing the probability of having something at least as good as x can be seen as a potential objective. Therefore, decision problems under risk can be seen as multicriteria

decision problems when the solutions (i.e., lotteries) are compared on the basis of their decumulative functions.

We make this connection more concrete by showing how the concepts of Pareto dominance and fairness are expressed in decision problems under risk.

As in multicriteria decision problems with the concept of Pareto dominance, in decision problems under risk some lotteries can seem better than other on all points of view. This is formalized by the concept of *first-order stochastic dominance*.

Definition 27. *The lottery l_1 first-order stochastically dominates the lottery l_2, denoted by $l_1 \gg_1 l_2$, iff:*

$$\forall x \in \mathbb{R}, G_{l_1}(x) \geq G_{l_2}(x) \quad \textit{and} \quad \exists x \in \mathbb{R}, G_{l_1}(x) > G_{l_2}(x)$$

If $l_1 \gg_1 l_2$, then for each possible value x, the probability of obtaining an outcome whose value is greater than x is always greater with l_1 than with l_2 and this property is strict for at least one value of x. Thus, it seems rational to prefer l_1 to l_2.

First-order stochastic dominance induces a simple partial preference relation over solutions. One way of refining this preference relation consists in considering the "quantity" of uncertainty induced by the lotteries. The underlying idea is that the solutions that are less risky should be preferred. This idea is formalized by the concept of *second-order stochastic dominance*.

Definition 28. *The lottery l_1 second-order stochastically dominates the lottery l_2 denoted by $l_1 \gg_2 l_2$ iff:*

$$\forall x \in \mathbb{R}, \int_{-\infty}^{x} G_{l_1}(y)dy \geq \int_{-\infty}^{x} G_{l_2}(y)dy \quad \textit{and} \quad \exists x \in \mathbb{R}, \int_{-\infty}^{x} G_{l_1}(y)dy > \int_{-\infty}^{x} G_{l_2}(y)dy$$

Second-order stochastic dominance is related to the concept of Lorenz dominance in multi-agent decision problems. Indeed, by interpreting a vector $(v_1, \ldots, v_m)$ of utilities of m agents as a lottery where each value v_i is obtained with probability $1/m$, Lorenz dominance reduces to second-order stochastic dominance. More formally, $(v_1, \ldots, v_m)$ Lorenz-dominates $(v'_1, \ldots, v'_m)$ iff $(v_1, 1/m; \ldots; v_m, 1/m) \gg_2 (v'_1, 1/m; \ldots; v_m, 1/m')$.

If $l_1 \gg_2 l_2$ and if additionally lotteries l_1 and l_2 have the same expected value, then we say that l_2 is a mean preserving spread of l_1. This notion can be related to Pigou-Dalton transfers in multi-agent decision problems and is at the basis of the notion of strong aversion to risk.

Definition 29. *A decision maker is strongly risk-averse iff she always prefers a lottery l_1 to a mean preserving spread of l_1. Contrarily, a decision maker is strongly risk-seeking iff she always prefers a mean preserving spread of a lottery l_1 to the lottery l_1 itself. Note that strong risk-aversion (resp. risk-loving) implies weak risk-aversion (resp. risk-loving).*

Example 16. *Lea enters in a shop to buy scratch games. Three possible games are for sale. The first game costs 2 € and makes it possible to win 4 € half of the time leading to a profit of 2 €. The second game costs 1 € and yields a gain of 2 € half of the time leading to a profit of 1 €. Lastly, the third game costs 2 € and makes it possible to win 4 € with a probability of 0.25 and 3 € with a probability of 0.25. The lotteries l_1, l_2 and l_3 over the possible profit values induced by the three games are given in Table 1.6.*

In this example, lottery l_1 first order stochastically dominates lottery l_3 and therefore Lea should not buy game 3. Furthermore, lottery l_1 is a mean preserving spread of l_2. Therefore, the choice of the game to buy between l_1 and l_2 should only be based on Lea's relation towards risk.

x	-2	-1	1	2
$l_1(x)$	0.5	0	0	0.5
$l_2(x)$	0	0.5	0.5	0
$l_3(x)$	0.5	0	0.25	0.25

x	-2	-1	1	2
$G_{l_1}(x)$	1	0.5	0.5	0.5
$G_{l_2}(x)$	1	1	0.5	0
$G_{l_3}(x)$	1	0.5	0.5	0.25

Table 1.6: Lotteries induced by the three possible games over possible profits in euro and the corresponding decumulatives.

First-order and second-order stochastic dominance induce natural preference relations over the solutions of a decision problem under risk. However, these preference relations are not very discriminating which limits there prescriptive abilities. Thus, to make decisions, a decision model with more prescriptive abilities should be used.

1.5.1 The Expected Value Model

The easiest criterion that we can think of to compare lotteries (on numerical outcomes) consists in comparing their respective expected values. Thus, in the expected value model, a lottery $l = (x_1, p_1; \ldots; x_q, p_q)$ is evaluated on the basis of its expected value $\mathrm{EV}(l) = \sum_{i=1}^{q} x_i p_i$ and a lottery l_1 is preferred (resp. strictly preferred) to a lottery l_2 iff $\mathrm{EV}(l_1) \geq$ (resp $>$) $\mathrm{EV}(l_2)$. At first glance, the expected value model is appealing due to its simplicity. However, it suffers from serious limitations on descriptive and normative points of view. In particular, expectation is not a *risk sensitive* criterion, in the sense that it assumes risk neutrality (e.g., a sure \$500 gain is equivalent to having a probability 1/2 of a \$1000 gain or nothing). And yet, most people take risk into account when they make decisions (descriptive limitation). Furthermore, the expected value model can lead to surprising conclusions as illustrated by the Saint-Petersburg paradox (normative limitation).

Example 17. *The Saint-Petersburg paradox relies on the following game. The player recursively tosses a coin. Every time heads appears, the earning of the player doubles. If tails appears on the i^{th} toss, the game ends and the player earns an amount of 2^{i-1} €. A casino wonders how much his clients would be willing to pay to play this game. To answer this question, the casino computes the expected gain of this game, which is worth $\sum_{i\in\mathbb{N}} 1/2^i \times 2^{i-1} = \sum_{i\in\mathbb{N}} 1/2 = +\infty$. Therefore, the casino concludes that if a decision maker follows the expected value criterion, she would be willing to pay an infinitely high price to play this game. However, most people would not be willing to bet an important amount of money to play this game.*

For these reasons, the use of the expected value model can be unadapted, to say the least, and other decision models should be investigated.

1.5.2 The Expected Utility Model

The most popular model in decision theory is the Expected Utility (EU) model [von Neumann and Morgenstern, 1947]. In this model, the decision maker's preferences are modeled using a utility function u that assigns a numerical value $u(x)$ to each possible consequence x. Then, a lottery l_1 is preferred to another lottery l_2 iff $\mathrm{EU}(l_1) \geq \mathrm{EU}(l_2)$, where $\mathrm{EU}(l) = \sum_{x\in\mathbb{R}} l(x)u(x)$ for any lottery l and where we recall that $l(x)$ denotes the probability of x in lottery l.

The EU model solves the two problems that were raised previously about the expected value model. Firstly, it is a risk-sensitive criterion.

Proposition 3. *A decision maker consistent with the EU model is both weakly and strongly risk-averse (resp. risk-seeking) iff the utility function that represent her preferences is concave (resp. convexe).*

This property gives a very simple way of modeling risk-averse and risk-seeking behaviors with the EU model. However, it shows that it is not possible with the EU model to represent a decision maker which would be weakly risk-averse but not strongly risk-averse.

Secondly, the EU model is consistent with the fact that most people would not bet important amounts of money in the Saint-Petersburg game.

Example 18. *(Example 17 continued) We come back to the Saint-Petersburg paradox and we consider the utility function* $u : x \mapsto \ln(x)$. *Now, the expected utility value of playing the game is equal to:*

$$\sum_{i\in\mathbb{N}^*} 1/2^i \times \ln(2^{i-1}) = \ln(2) \sum_{i\in\mathbb{N}^*} 1/2^i \times (i-1) = \ln(2).$$

This corresponds to a certainty equivalent of 2 *euros since* $u(2) = \ln(2)$. *Thus, the price of the game will be given by a finite value.*

Furthermore, the EU model benefits from a "good" axiomatic justification. Stated differently, there is a set of simple and intuitive rules called *axioms* that justify the EU model in the sense that anyone that agrees with the axioms should agree with the use of the EU model and anyone that agrees with the EU model should agree with the axioms. The axioms supporting the EU model are the following ones:

Axiom 1. ***Complete preorder****: the preference relation* $\succsim$ *which is defined on lotteries is complete, reflexive and transitive.*

Axiom 2. ***Independence****: given any triple of lotteries* l_1, l_2, l_3 *and for all* p *in* $[0,1]$,

$$l_1 \succsim l_2 \Leftrightarrow (l_1, p; l_3, 1-p) \succsim (l_2, p; l_3, 1-p)$$

The independence axiom states that a preference between two lotteries l_1 and l_2 is not modified by mixing them with a third lottery l_3 in similar proportion. Indeed, as the lottery l_3 is present in both mixtures with the same probability $(1-p)$, the preference should only be based on the preference of the decision maker between lottery l_1 and lottery l_2.

Axiom 3. ***Continuity****: given any triple of lotteries* l_1, l_2, l_3,

$$l_1 \succ l_2 \succ l_3 \Rightarrow \exists p \in [0,1], (l_1, p; l_3, 1-p) \sim l_2$$

The continuity axiom states that if lottery l_2 lies between lottery l_1 and lottery l_3 in terms of preference, then there should be a probability value p such that the decision maker is indifferent between mixture $(l_1, p; l_3, 1-p)$ and l_2. This axiom resembles the intermediate value theorem which requires continuity, hence the name of the axiom.

We now give the result stating that the combination of the complete preorder axiom, the independence axiom and the continuity axiom both imply and is implied by the EU model. Such a result is called a *representation theorem.*

Theorem 4 (Machina [1990])**.** *The preference relation* $\succsim$ *satisfies axioms* complete preorder, independence *and* continuity *iff there exists a utility function* $u : \mathbb{R} \to \mathbb{R}$ *unique up to a strictly positive affine transformation such that:*

$$l_1 \succsim l_2 \Leftrightarrow \sum_{x\in\mathbb{R}} l_1(x)u(x) \geq \sum_{x\in\mathbb{R}} l_2(x)u(x)$$

Nevertheless, despite its intuitive appeal and its axiomatic justification, the EU model does not make it possible to account for some rational decision behaviors frequently observed experimentally. We present three situations in which the EU model is not adapted.

For instance, it is unable to explain the paradox of nontransitive dice as designed by statistician Efron and reported by Gardner [1970]. The specific variant presented here is due to Rowett.

Example 19 (Rowett Dice)**.** *Consider a two-player game involving the following set of six-sided dice: die* A *with sides* $(1,4,4,4,4,4)$*, die* B *with sides* $(3,3,3,3,3,6)$ *and die* C *with sides* $(2,2,2,5,5,5)$*. The players, each equipped with a personal set of Rowett dice, simultaneously choose a die to throw; the winner is the player who rolls the highest number. It is easy to realize that die* A *rolls higher than* B *most of the time, so die* A *should be preferred to* B*. Similarly die* B *rolls higher than* C *most of the time, and the same can be said about* C *against* A*. In other words, the relation "more likely to win" is not transitive, and in fact it is even cyclic.*

This example can be formalized by characterizing dice A, B, C by lotteries l_A, l_B, l_C over the set $X = \{1,2,\ldots,6\}$ of possible outcomes. The expected utility model is obviously unable to accommodate the above preferences $l_A \succ l_B \succ l_C \succ l_A$ because it is impossible to have $EU(l_A) > EU(l_B) > EU(l_C) > EU(l_A)$. Actually, every binary preference relation solely based on a unary functional u over distributions would fail to explain the paradox.

A second example of situations that cannot be accommodated by the EU model is the preference reversal phenomenon that was notably observed experimentally by Lichtenstein and Slovic [1971, 1973] and Lindman [1971].

Example 20. *The preference reversal phenomenon occurs in the monetary context when lottery l_1 is strictly preferred to lottery l_2, $l_1 \sim x$, $l_2 \sim y$ and $x < y$ (see Figure 1.5).*

$$\begin{array}{ccc} l_1 & \succ & l_2 \\ \wr & & \wr \\ x & < & y \end{array}$$

Figure 1.5: Illustration of the preference reversal phenomenon.

That is, the person would agree to trade l_1 for x, l_2 for y, yet strictly prefers l_1 to l_2 and y to x. The EU model can of course not accommodate this phenomenon as we cannot have $EU(l_1) > EU(l_2) = u(y) > u(x) = EU(l_1)$.

A last example of a situation that cannot be accommodated by the EU model is the so-called Allais' paradox [1953]. We recall in Example 21 a simpler version of this paradox due to Kahneman and Tversky [1979]:

Example 21. *Consider the following lotteries l_1, l_2, l'_1, l'_2:*

x	0\$	3000\$	4000\$
$l_1(x)$	0	1	0
$l_2(x)$	0.2	0	0.8
$l'_1(x)$	0.75	0.25	0
$l'_2(x)$	0.8	0	0.2

In this example Kahneman and Tversky observed that most people prefer lottery l_1 to lottery l_2 but prefer lottery l'_2 to lottery l'_1. One possible explanation for these preferences is that l_1 is more desirable than l_2 because it leads to a sure outcome. However, both l'_1 and l'_2 are risky prospects and one might prefer the one that leads to the highest expected value, i.e., l'_2. However, lottery l'_1 (resp. l'_2) is just the mixture of lottery l_1 (resp. l_2) and a sure amount of 0$ *with probabilities* 0.25 *and* 0.75 *respectively. Thus, those preferences violate the independence axiom which holds in EU theory. This makes it impossible to account for such preferences with the EU model.*

Those descriptive limitations of the EU model have prompted researchers in decision theory to develop more general models that extend EU. Famous examples of such models are: the generalized expected utility model [Machina, 1983], the Choquet expected utility model [Schmeidler, 1986] or the rank-dependent utility model [Quiggin, 1993]. In the rest of the chapter we present the skew-symmetric bilinear utility model and the weighted expected utility model as they are the models investigated in this thesis.

1.5.3 The Skew-Symmetric Bilinear Utility Model

The Skew-Symmetric Bilinear (SSB) utility theory developed and axiomatized by Fishburn [1982, 1984b] extends expected utility theory and enables to accommodate the descriptive limitations that we have mentioned.

In this model, an agent is endowed with a binary functional φ over pairs (x, y), with $x \succ y \Leftrightarrow \varphi(x, y) > 0$. The value $\varphi(x, y)$ is commonly interpreted as the intensity with which the agent prefers x to y. Functional φ is assumed to be skew-symmetric, i.e., $\varphi(x, y) = -\varphi(y, x)$ and bilinear w.r.t. the usual mixture operation on lotteries, i.e., the SSB criterion for comparing lotteries l_1 and l_2 is written:

$$\varphi(l_1, l_2) = \sum_{x,y} l_1(x)\, l_2(y)\varphi(x, y)$$

We have $l_1 \succ$ (resp. $\prec$) l_2 if $\varphi(l_1, l_2) >$ (resp. $<$) 0 (strict preference), and $l_1 \sim l_2$ if $\varphi(l_1, l_2) = 0$ (indifference).

SSB utility theory is appealing due to its strong descriptive abilities. Indeed, as will be seen in Examples 22, 23 and 24, it can account for the preferences observed in the Gardner dice paradox, in the weak preference reversal paradox and in Allais' Paradox.

Example 22. *(Example 19 cont'd) Lotteries l_A, l_B, l_C are defined in Figure 1.6.*

x	1	2	3	4	5	6
$l_A(x)$	1/6	0	0	5/6	0	0
$l_B(x)$	0	0	5/6	0	0	1/6
$l_C(x)$	0	1/2	0	0	1/2	0

Figure 1.6: Distributions l_A, l_B, l_C.

By setting $\varphi(x, y) = 1$ if $x > y$, and $\varphi(x, y) = -1$ if $x < y$, $\varphi(l_1, l_2)$ corresponds then to the probability that l_1 beats l_2 minus the probability that l_2 beats l_1. The obtained SSB utilities in the example are:

$$\begin{aligned} \varphi(l_A, l_B) &= 25/36 - 11/36 = 14/36, \\ \varphi(l_B, l_C) &= 21/36 - 15/36 = 6/36, \\ \varphi(l_C, l_A) &= 21/36 - 15/36 = 6/36. \end{aligned}$$

Therefore, we have $\varphi(l_A, l_B) > 0$, $\varphi(l_B, l_C) > 0$ *and* $\varphi(l_C, l_A) > 0$, *which is consistent with the relation "more likely to win" between dice (i.e.,* $l_A \succ l_B \succ l_C \succ l_A$*).*

As the SSB model is compatible with intransitive preferences, the SSB model makes it possible to accommodate preference reversal phenomenons [Fishburn, 1984b; Loomes and Sugden, 1983].

Example 23. *(Example 20 cont'd) Assume* $l_1 = (30, 0.9; 0, 0.1)$, $l_2 = (100, 0.3; 0, 0.7)$ *with* $l_1 \sim 25$, $l_2 \sim 27$, *and* $l_1 \succ l_2$. *As* $27 > 25$, *these judgments violate every model that assumes full transitivity. However, they are consistent with the SSB model with* $\varphi(100, 0) = 24$, $\varphi(100, 27) = 70/3$, $\varphi(100, 30) = 23$, $\varphi(30, 0) = 12$, $\varphi(30, 25) = 1$, $\varphi(27, 0) = 10$ *and* $\varphi(25, 0) = 9$. *Indeed in that case:*

$$\varphi(l_1, 25) = 0.9\varphi(30, 25) + 0.1\varphi(0, 25) = 0$$
$$\varphi(l_2, 27) = 0.3\varphi(100, 27) + 0.7\varphi(0, 27) = 0$$
$$\varphi(l_1, l_2) = 0.27\varphi(30, 100) + 0.03\varphi(0, 100) + 0.63\varphi(30, 0) + 0.07\varphi(0, 0) \approx 0.63 > 0$$

To list the axioms that support the SSB model, we introduce two new axioms:

Axiom 4. ***Dominance****: We consider three lotteries* l_1, l_2 *and* l_3. *If* $l_1 \succ l_2$ *and* $l_1 \succsim l_3$ *then* $\forall p \in [0, 1]$, $l_1 \succ (l_2, p; l_3, 1 - p)$. *If* $l_2 \succ l_1$ *and* $l_3 \succsim l_1$ *then* $\forall p \in [0, 1]$, $(l_2, p; l_3, 1 - p) \succ l_1$. *Lastly, if* $l_1 \sim l_2$ *and* $l_1 \sim l_3$ *then* $\forall p \in [0, 1]$, $l_1 \sim (l_2, p; l_3, 1 - p)$.

Axiom 5. ***Symmetry****: We consider three lotteries* l_1, l_2 *and* l_3. *If* $l_1 \succ l_2 \succ l_3$, $l_1 \succ l_3$ *and* $l_2 \sim (l_1, 0.5; l_3, 0.5)$, *then* $(l_1, p; l_3, 1 - p) \sim (l_1, 0.5; l_2, 0.5)$ *iff* $(l_1, 1 - p; l_3, p) \sim (l_2, 0.5; l_3, 0.5)$.

While the dominance axiom is quite natural and easy to understand, the symmetry axiom is more original and deserves more explanations. This axiom is illustrated in a Marschak-Machina triangle in Figure 1.7.

A Marschak-Machina triangle is a graphical representation of the iso-preference curves (i.e., lines connecting lotteries that the decision maker deems as equally "good") of all possible lotteries between three elements [Marschak, 1950]. More formally, the three extreme points represent three lotteries l_1, l_2 and l_3 and each point in the triangle represents a lottery over these three elements with probabilities corresponding to the weights of the corresponding convex combination of l_1, l_2 and l_3.

The Marschak-Machina triangle given below is constructed for three lotteries such that $l_1 \succ l_2 \succ l_3$, $l_1 \succ l_3$ and $l_2 \sim (l_1, 0.5; l_3, 0.5)$. This last point is illustrated by a vertical iso-preference curve in the triangle connecting l_2 to $(l_1, 0.5; l_3, 0.5)$. Note that the dominance axiom implies that the iso-preference curves are necessary straight lines in the triangle. The symmetry axiom induces an axial symmetry of the iso-preference curves of axis the vertical bisector. Hence the name of the axiom.

Note that this property is verified by the EU model where the iso-preference curves are all parallel straight lines. However, iso-preference curves are not parallel in general in the SSB model and might intersect.

If for all triples (l_1, l_2, l_3) the iso-preference curves intersect out of the triangle or does not intersect, then we will obtain a transitive SSB model. However, if there exists a triple (l_1, l_2, l_3) such that the iso-preference curves intersect in the triangle, then we obtain a preference cycle as in Example 19. These different cases are illustrated in Figures 1.8, 1.9 and 1.10.

Now that the dominance and symmetry axioms have been introduced, we can state the representation theorem of the SSB model.

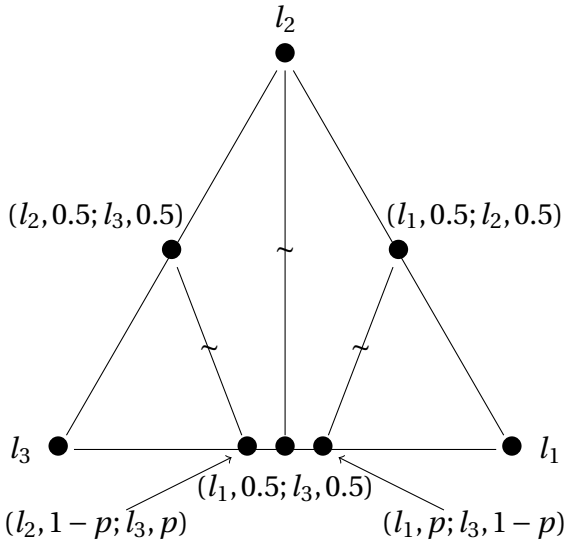


Figure 1.7: Illustration of the symmetry axiom.

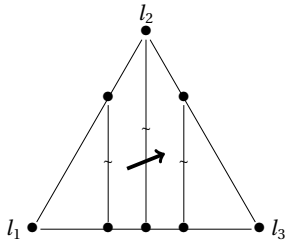


Figure 1.8: Expected Utility.

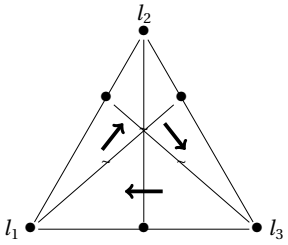


Figure 1.9: SSB: Cyclic.

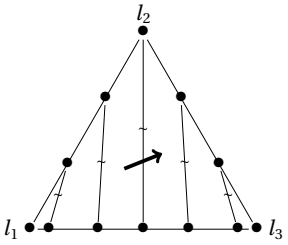


Figure 1.10: SSB: Transitive.

Patterns of $\sim$ curves. $l_1 > l_2$, $l_2 > l_3$, and $l_2 \sim (l_1, 0.5; l_3, 0.5)$ in all cases. Arrows show decreasing preference directions.

Theorem 5 (Fishburn [1982])**.** *The preference relation $\succsim$ satisfies axioms* continuity, dominance *and* symmetry *iff there is a skew-symmetric bilinear function $\varphi : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ such that:*

$$l_1 \succsim l_2 \Leftrightarrow \sum_{x \in \mathbb{R}} \sum_{y \in \mathbb{R}} l_1(x) l_2(y) \varphi(x, y) \geq 0$$

The SSB utility function φ is unique up to a similarity transformation (i.e., a multiplication by a positive constant).

The SSB model encompasses many decision criteria, e.g.:

- $\varphi(x, y) = x - y$ yields the expectation criterion;
- $\varphi(x, y) = u(x) - u(y)$ yields the EU model;
- $\varphi(x, y) = \mathbb{1}_{x \geq \theta} - \mathbb{1}_{y \geq \theta}$, where $\mathbb{1}_{x \geq \theta} = 1\,(0)$ if $x \geq (<)\ \theta$, yields the probability threshold criterion [Yu *et al.*, 1998], which states that $l_1 \succ l_2$ if $\sum_{x \geq \theta} l_1(x) > \sum_{x \geq \theta} l_2(x)$;
- $\varphi(x, y) = 1$ (resp. $0, -1$) if $x > y$ (resp. $x = y$, $x < y$) yields the dominance relation of Example 19, which states that $l_1 \succ l_2$ if $\sum_{x > y} l_1(x) l_2(y) > \sum_{y > x} l_1(x) l_2(y)$; this is called *probabilistic dominance* (PD) in the sequel [Blavatskyy, 2006].

Moreover, the SSB model can represent different risk attitudes via an adequate choice of functional φ.

Proposition 4 (Nakamura [1989])**.** *Let* $\varphi_1(x,y) = \partial\varphi(x,y)/\partial x$*. If* $\varphi_1(x,x) \neq 0$ *exists for all* x*,* φ *is weakly risk-averse (resp. risk-seeking) if and only if* $\varphi_1(y,y)/\varphi(x,y) \geq (resp.\ \leq)\ 1/(x-y)$ *for* $x \neq y$*.*

We now address, two points that could at first be seen as potential disadvantages for the SSB model but are in fact not problematic.

• Firstly, the possibility of cyclic preferences in SSB utility theory could be seen as a significant barrier to its use in automated decision as the existence of an *optimal* lottery is not obvious. Given a set $\mathscr{L} = \{l_1, \ldots, l_n\}$ of lotteries, the SSB criterion induces a *weighted tournament* on $\mathscr{L}$, i.e., for each pair l_1, l_2 of lotteries such that $\varphi(l_1,l_2) > 0$, $\varphi(l_1,l_2)$ represents the intensity with which l_1 is preferred to l_2. Multiple rules exist for determining the winner(s) of a weighted tournament [Fischer *et al.*, 2016]. We adopt in this thesis the minimax rule [Young, 1977], also known as Condorcet's rule or the Simpson-Kramer method, where each lottery l is evaluated by the highest intensity with which another lottery is preferred to l, and one selects a lottery with minimal evaluation. More formally, one seeks a lottery l in $\arg\min_{l\in\mathscr{L}} \max_{l'\in\mathscr{L}} \varphi(l',l)$.

Interestingly, if one enlarges the set of possible lotteries to the convex hull $\mathrm{CH}(\mathscr{L})$ of $\mathscr{L}$ where $\mathrm{CH}(\mathscr{L}) = \{l : l{=}\sum_{i=1}^{n}\lambda_i l_i$ with $\sum_{i=1}^{n}\lambda_i{=}1$ and $\lambda_i{\geq}0,\ \forall i\}$, Fishburn [1984a,b] showed that an optimal lottery $l{\in}\mathrm{CH}(\mathscr{L})$ w.r.t. the minimax rule has the desirable property that $\varphi(l,l'){\geq}0$ for all l'. For instance, coming back to our dice example (Example 22), playing die A (resp. B, C) with probability $\frac{3}{13}$ (resp. $\frac{3}{13}$, $\frac{7}{13}$) is an optimal strategy for the relation "more likely to win". However, a decision maker may not accept to use a mixture of lotteries (i.e., a lottery in $\mathrm{CH}(\mathscr{L})\backslash\mathscr{L}$). Thus, we will study both optimization in $\mathscr{L}$ and in $\mathrm{CH}(\mathscr{L})$.

• Secondly, the specification of an SSB utility function requires the evaluation of $\varphi(x,y)$ for all pairs of value (x,y) given in the problem. This may be prohibitive for some applications. Interestingly, Nakamura [1989] and Chew [1983] axiomatized subclasses of SSB functions that have more compact representations. We first present the subclass investigated by Nakamura, and then present the one developed by Chew in the next subsection.

Axiom 6. ***Multiplicative improvements:*** *Consider 5 numerical values* x_1*,* x_2*,* x_3*,* x_4 *and* x_5 *such that* $x_1 < x_2 < x_3 < x_4 < x_5$*. If* $x_3 = (x_2+x_4)/2 = (x_1+x_5)/2$*, and if* $(x_1,p_1;x_3,1-p_1) \sim x_2$*,* $(x_3,p_2,x_5,1-p_2) \sim x_4$*,* $(x_1,p_3,x_5,1-p_3) \sim x_3$ *then* $(1-p_3)/p_3 = (1-p_1)/p_1 \times (1-p_2)/p_2$*.*

In words, this axiom states that if x_3 is the middle of $[x_2,x_4]$ and $[x_1,x_5]$, then the ratio between the intensity of preference between x_5 and x_3 and the intensity of preference between x_3 and x_1 is the product of two ratios: (1) the one between the intensity of preference between x_5 and x_4 and the intensity of preference between x_4 and x_3 and (2) the one between the intensity of preference between x_3 and x_2 and the intensity of preference between x_2 and x_1.

Theorem 6 (Nakamura [1989])**.** *The preference relation* $\succsim$ *satisfies axioms* continuity, dominance, symmetry *and* multiplicative improvements *iff there is a skew-symmetric bilinear function* $\varphi : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ *such that:*

$$l_1 \succsim l_2 \Leftrightarrow \sum_{x\in\mathbb{R}}\sum_{y\in\mathbb{R}} l_1(x)\,l_2(y)\,\varphi(x,y) \geq 0$$

$$\varphi(x,y) = w(x)\,w(y)\,f(x-y),$$

where $w > 0$ *is continuous, and* f *is continuous and strictly increasing.*

In this model, the intensity of preference between two values x and y depends on two functions w and f. While w acts as a multiplicative factor which increases or decreases the intensity of preference according to the values of x and y, the function f measures the intensity of preference on the basis of the difference between x and y.

We recall that Proposition 4 (on page 37) characterizes the attitude towards risk of a decision maker consistent with the SSB model. Proposition 4 applied to this particular class of SSB functions gives the following result.

Proposition 5 (Nakamura [1989])**.** *Consider a decision maker having preferences $\succsim$ such that there is a skew-symmetric bilinear function $\varphi : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ with:*

$$l_1 \succsim l_2 \Leftrightarrow \sum_{x\in\mathbb{R}} \sum_{y\in\mathbb{R}} l_1(x) l_2(y) \varphi(x,y) \geq 0$$
$$\varphi(x,y) = w(x) w(y) f(x-y),$$

where $w > 0$ is continuous, and f is continuous and strictly increasing. Then the decision maker is weakly risk-averse (resp. risk seeking) iff for all $x > y$,

$$\frac{w(x)}{w(y)} \leq (resp.\ \geq) \frac{f(x-y)}{f'(0)(x-y)} \leq (resp.\ \geq) \frac{w(y)}{w(x)},$$

where $f'(0) > 0$ and $w' \leq (\ resp.\ \geq) 0$.

We now turn to the WEU criterion which is a special case of SSB utilities developed by Chew [1983]. This model yields a more compact representation of the preferences of the decision maker and enforces transitivity while still being able to cope with Allais' paradox.

1.5.4 The Weighted Expected Utility Model

The Weighted Expected Utility (WEU) theory, developed by Chew [1983], is obtained from the SSB utility theory by adding the following transitivity axiom [Fishburn, 1983].

Axiom 7. *$\sim$-**Transitivity**: If $l_1 \sim l_2$ and $l_2 \sim l_3$, then $l_1 \sim l_3$.*

Theorem 7. *The preference relation $\succsim$ satisfies axioms* continuity, dominance, symmetry *and* $\sim$-transitivity *iff there are linear functionals $u : \mathbb{R} \to \mathbb{R}$ and $w : \mathbb{R} \to \mathbb{R}$ with w non-negative (and strictly positive if the set of lotteries considered has both maximally preferred and minimally preferred elements, or if it has neither) such that:*

$$l_1 \succsim l_2 \Leftrightarrow \sum_{x\in\mathbb{R}} \sum_{y\in\mathbb{R}} l_1(x) l_2(y) (u(x) w(y) - u(y) w(x)) \geq 0 \tag{1.2}$$

If, in addition, $\succsim$ is nonempty and (u, w) is any pair of linear functionals (w.r.t. mixtures of probabilities) that satisfies Equation 1.2, then a pair (u', w') of linear functionals (w.r.t. mixtures of probabilities) satisfies Equation 1.2 in place of (u, w) iff, there are numbers a, b, c and d such that:

$$u' = au + bw$$
$$w' = cu + dw$$
$$ad - bc > 0$$

The WEU criterion relies on two functions u and w (with $w \geq 0$) defined on $\mathbb{R}$ such that $\varphi(x,y) = u(x)w(y) - u(y)w(x)$. These two functions are lifted to lotteries by linearity in probabilities: $u(l) = \sum_{x\in\mathbb{R}} l(x)u(x)$ and $w(l) = \sum_{x\in\mathbb{R}} l(x)w(x)$. Consequently: $l_1 \succ l_2 \Leftrightarrow u(l_1)w(l_2) - u(l_2)w(l_1) > 0$ and $u(l_1)w(l_2) - u(l_2)w(l_1)$ is interpreted as a signed intensity of preference between lotteries l_1 and l_2 as in the SSB model. This last equation can be rewritten (if $w > 0$) as:

$$l_1 \succ l_2 \Leftrightarrow v(l_1) = u(l_1)/w(l_1) > v(l_2) = u(l_2)/w(l_2) \tag{1.3}$$

In the sequel, we will continue to use v to denote the ratio u/w. It is worth noting that if w is constant then WEU boils down to EU. As WEU assigns a score $v(p)$ to each lottery p, it is unable to accommodate intransitive preferences which can be seen as a strength from a normative point of view. Thus, the Marschak-Machina triangles corresponding to the WEU model are similar to the one represented in Figure 1.10[5] (on page 36). However, the WEU model can still accommodate Allais' paradox:

Example 24. *(Example 21 cont'd) We rescale the possible gains from interval* $[0,4000]$ *to interval* $[0,1]$. *Consider the pair of functionals* (u,w) *defined by* $u(x) = x^2$ *and* $w(x) = 1 - \sqrt{x}$*, leading to the following values:*

	l_1	l_2	l'_1	l'_2
u	0.5625	0.8	0.1406	0.2
w	0.1340	0.2	0.7835	0.8

We can easily check that the WEU model using these two functions is compatible with Allais' paradox:

- $l_1 \succ l_2$ *since* $u(l_1)w(l_2) - u(l_2)w(l_1) \approx 0.005 > 0$.
- $l'_2 \succ l'_1$ *since* $u(l'_2)w(l'_1) - u(l'_1)w(l'_2) \approx 0.044 > 0$.

Attitude towards risk. The components of the WEU model can be chosen to account for risk-averse or risk-seeking behaviors. By specifying Proposition 4 (on page 37) to the case of WEU, we obtain that a decision maker consistent with WEU is risk-averse (resp. risk-seeking) iff $r(x,y)$ is positive (resp. negative) for all x, y with $x \neq y$ where:

$$\begin{aligned} r(x,y) = w(y)[\frac{du}{dx}(y)(x-y) + u(y) - u(x)] \\ + u(y)[w(x) - \frac{dw}{dx}(y)(x-y) - w(y)] \end{aligned}$$

Those conditions can easily be met. For instance, if functions u and w are positive, it is sufficient to choose a concave (resp. convex) u function and a convex (resp. concave) w function to obtain a risk-averse (resp. risk-seeking) behavior. We illustrate this point in the following example.

Example 25. *Consider the following lotteries* l_1 *and* l_2*:*

x	0\$	25\$	100\$
$l_1(x)$	0	1	0
$l_2(x)$	0.75	0	0.25

[5] More can be found about Marschak-Machina triangles with the WEU theory in the work by Chew and Waller [1986].

Note that l_1 is the sure lottery where one obtains the expectancy of lottery l_2. Let $u_1(x) = (x/100)^2$, $u_2(x) = \sqrt{x/100}$ and $w_1(x) = w_2(x) = 1 - x/100$. Let $\succsim_1$ (resp. $\succsim_2$) denote the preference relation induced by (u_1, w_1) (resp. (u_2, w_2)). Then $\succsim_1$ represents a risk-seeking behavior, whereas $\succsim_2$ represents a risk-averse behavior as illustrated by the preferences expressed over lotteries l_1 and l_2:

$$l_2 \succ_1 l_1 \text{ since } u_1(l_1)w_1(l_2) - u_1(l_2)w_1(l_1) \approx -0.14$$
$$l_1 \succ_2 l_2 \text{ since } u_2(l_1)w_2(l_2) - u_2(l_2)w_2(l_1) \approx 0.19$$

Interpretation of WEU. To give another insight on WEU, let us rewrite Equation 1.3 in the following ways (if $w > 0$):

$$u(l_1)/w(l_1) \geq u(l_2)/w(l_2)$$
$$\sum_{x\in\mathbb{R}} \frac{w(x)l_1(x)}{\sum_{y\in\mathbb{R}} w(y)l_1(y)} v(x) \geq \sum_{x\in\mathbb{R}} \frac{w(x)l_2(x)}{\sum_{y\in\mathbb{R}} w(y)l_2(y)} v(x)$$

where $v(x)$ is defined as the ratio $u(x)/w(x)$. Hence, lottery l_1 is preferred to lottery l_2 with the WEU model iff $\mathrm{V}(l_1) \geq \mathrm{V}(l_2)$, where for any lottery l:

$$\mathrm{V}(l) = \frac{u(l)}{w(l)} = \sum_{x\in\mathbb{R}} \frac{w(x)l(x)}{\sum_{y\in\mathbb{R}} w(y)l(y)} v(x)$$

This last formulation shows that the mixture operator defined in subsection 1.2.1 in the setting of multicriteria decision making is a WEU type criterion where function v is the identity function. This formulation makes it also possible to interpret functions w and v. While function v can be interpreted as a utility function representing the extent to which an outcome is satisfactory, function w can be interpreted as a probability distortion function where the distortion depends on the amounts that are at stake in the lottery. This distortion of probabilities enables the violation of independence exposed by Allais' paradox. Note that, in this respect, the WEU model is close to the Rank-Dependent Utility (RDU) model [Quiggin, 1993] (that also makes it possible to account for Allais' paradox by introducing a distortion of probabilities). However, an important difference between RDU and WEU can be made regarding their relative position w.r.t. the *betweenness* axiom (discussed below).

Axiom 8. ***Betweenness:*** *Given any pair of lotteries l_1, l_2:*

$$\forall \lambda \in (0,1),\ \textit{if } l_1 \succsim l_2,\ \textit{then } l_1 \succsim (l_1, \lambda; l_2, (1-\lambda)) \succsim l_2.$$

The betweenness axiom [Chew, 1989] states that an agent has no strict preference or aversion for randomization. If betweenness holds, then for any pair of lotteries (l_1, l_2), any mixture of l_1 and l_2 is ranked between l_1 and l_2 w.r.t $\succsim$.

Verifying betweenness, which is a relaxation of the independence axiom, ensures several desirable properties. In particular, it ensures that the preferred element in a finite set $\mathscr{L}$ is also the preferred element in the convex hull of $\mathscr{L}$. This implies that we don't need to investigate mixtures of alternatives to find an optimal decision. Besides, betweenness has also some nice consequences in game theory , concerning e.g., the existence of Nash equilibria (see [Chew, 1989]). While the WEU model verifies the betweenness axiom, RDU does not.

In Chapter 4, we investigate sequential decision making problems with the SSB utility model and the WEU model. We show that while the determination of a randomized optimal SSB strategy is of polynomial complexity in the setting of decision trees, it becomes NP-hard if randomized strategies are not allowed. Interestingly, we prove that both problems are of polynomial complexity in the WEU case. We present several solution methods either based on mathematical programming tools or on combinatorial methods. We then extend our results to the setting of Markov decision processes by using the concept of wealth augmented Markov decision processes. Lastly, in Chapter 5, we investigate the elicitation of the WEU model. We present an elicitation protocol to elicit the two functions u and w required by the model and we show how to use it in an incremental elicitation procedure.

1.6 Conclusion

In this chapter, we have formally defined the notions of decision problem and decision model and we have formalized the notion of preferences of a decision maker by using binary relations and aggregation operators. We have explored several aggregation operators and decision models and their properties in the following domains: multicriteria decision making, multi-agent decision making, robust decision making and decision making under risk. We have seen that these frameworks consider elements of different natures and yield different concerns as fairness in multi-agent decision making or risk aversion in decision making under risk. However, we also pointed out that there exists many connections between them. These connections are summarized in Table 1.7.

	Multicriteria	Multi-agent	Robust	Risk
Objective	Criterion	Agent	Scenario	Reward value
Number of objectives	Constant	Variable	Finite (discrete model) or infinite (interval model)	Continuum
First order dominance	Pareto dominance	Pareto dominance		First-order stochastic dominance
Second order dominance	Lorenz dominance	Lorenz dominance		Second-order stochastic dominance
Quantification of objectives	Importance coefficients	Exogenous rights		Probabilities

Table 1.7: Connections between the different domains considered in this chapter.

In this thesis, we will use the decision models that we have investigated in this chapter to solve many decision problems of combinatorial nature. These problems are presented in the next chapter.

1.7 References

M. Allais. Le comportement de l'homme rationnel devant le risque: Critique des postulats et axiomes de l'ecole americaine. *Econometrica*, 21(4):pp. 503–546, 1953. 33

C. Arnsperger. Envy-freeness and distributive justice. *Journal of Economic Surveys*, 8(2):155–186, 1994. 22

J.A. Bagnell, A.Y. Ng, and J.G. Schneider. Solving uncertain markov decision processes. Technical report, Carnegie Mellon University, 2001. 25

G. Beliakov and J. Špirková. Weak monotonicity of Lehmer and Gini means. *Fuzzy sets and systems*, 299:26–40, 2016. 19

G. Beliakov and T. Wilkin. On some properties of weighted averaging with variable weights. *Information Sciences*, 281:1–7, 2014. 19

G. Beliakov, T. Calvo, and T. Wilkin. On the weak monotonicity of gini means and other mixture functions. *Information Sciences*, 300:70–84, 2015. 18

D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98(1):49–71, 2003. 25

P. Blavatskyy. Axiomatization of a Preference for Most Probable Winner. *Theory and Decision*, 60(1):17–33, 02 2006. 36

S. Bouveret and M. Lemaître. Fonctions d'utilité collective avec droits exogènes inégaux. *Actes des cinquièmes journées francophones Modèles Formels de l'Interaction (MFI'07)*, 2007. 22

S.J. Brams and A.D. Taylor. *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press, 1996. 22

S.H. Chew and W.S. Waller. Empirical tests of weighted utility theory. *Journal of Mathematical Psychology*, 30(1):55–72, 1986. 39

S.H. Chew. A generalization of the quasilinear mean with applications to the measurement of income inequality and decision theory resolving the Allais paradox. *Econometrica: Journal of the Econometric Society*, pages 1065–1092, 1983. 19, 24, 37, 38

S.H. Chew. Axiomatic utility theories with the betweenness property. *Annals of operations Research*, 19(1):273–298, 1989. 40

K.M. Chong. An induction theorem for rearrangements. *Canadian J. of Math.*, 28:154–160, 1976. 22

P. Dasgupta, A. Sen, and D. Starrett. Notes on the measurement of inequality. *Journal of economic theory*, 6(2):180–187, 1973. 24

A.P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *The annals of mathematical statistics*, pages 325–339, 1967. 28

D. Dubois, H. Fargier, J. Lang, H. Prade, and R. Sabbadin. Qualitative decision theory and multistage decision making: A possibilistic approach. In *In Proceedings of the European Workshop on Fuzzy Decision Analysis for Management, Planning and Optimization (EFDAN'96*. Citeseer, 1996. 28

D. Dubois, H. Fargier, and H. Prade. Beyond min aggregation in multicriteria decision:(ordered) weighted min, discri-min, leximin. In *The ordered weighted averaging operators*, pages 181–192. Springer, 1997. 17

M. Ehrgott. *Multicriteria optimization*. Springer Science & Business Media, 2006. 13

E. Elkind and A. Ismaili. OWA-based extensions of the chamberlin–courant rule. In *Proceedings of the International Conference on Algorithmic Decision Theory 2015, (ADT)*, pages 486–502. Springer, 2015. 17

F. Fischer, O. Hudry, and R. Niedermeier. Weighted tournament solutions. In Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, editors, *Handbook of Computational Social Choice*, pages 85–102. Cambridge University Press, 2016. 37

P.C. Fishburn. Nontransitive measurable utility. *Journal of Mathematical Psychology*, 26(1):31–67, 1982. 34, 36

P.C. Fishburn. Transitive measurable utility. *Journal of Economic Theory*, 31(2):293–317, 1983. 38

P.C. Fishburn. Dominance in SSB utility theory. *Journal of Economic Theory*, 34(1):130–148, 1984. 24, 37

P.C. Fishburn. SSB utility theory: an economic perspective. *Mathematical Social Sciences*, 8(1):63 – 94, 1984. 34, 35, 37

J. Fodor, J.L. Marichal, and M. Roubens. Characterization of the ordered weighted averaging operators. *IEEE Trans. on Fuzzy Systems*, 3(2):236–240, 1995. 16

M. Gardner. Mathematical games: The paradox of nontransitive dice and the elusive principle of indifference. *Sci. Amer.*, 223:110–114, Dec. 1970. 33

J. Goldsmith and R. H. Sloan. The ai conference paper assignment problem. In *2007 AAAI Workshop*, 2007. 17

D. Golovin. Max-min fair allocation of indivisible goods. 2005. 22

S. Greco, J. Figueira, and M. Ehrgott. Multiple criteria decision analysis. *Springer's International series*, 2005. 13

D. Kahneman and A. Tversky. Prospect theory: An analysis of decisions under risk. *Econometrica*, pages 263–291, 1979. 33

J. Lesca and P. Perny. LP solvable models for multiagent fair allocation problems. In *Proceedings of the European Conference on Artificial Intelligence*, pages 393–398, 2010. 17

S. Lichtenstein and P. Slovic. Reversals of preference between bids and choices in gambling decisions. *Journal of experimental psychology*, 89(1):46, 1971. 33

S. Lichtenstein and P. Slovic. Response-induced reversals of preference in gambling: An extended replication in las vegas. *Journal of Experimental Psychology*, 101(1):16, 1973. 33

H.R. Lindman. Inconsistent preferences among gambles. *Journal of Experimental Psychology*, 89(2):390, 1971. 33

X. Liu. The orness measures for two compound quasi-arithmetic mean aggregation operators. *International Journal of Approximate Reasoning*, 51(3):305–334, 2010. 19

G. Loomes and R. Sugden. A rationale for preference reversal. *The American Economic Review*, 73(3):428–432, 1983. 35

L. Losonczi. General inequalities for nonsymmetric means. *aequationes mathematicae*, 9(2-3):221–235, 1973. 17, 19

M.J Machina. Generalized expected utility analysis and the nature of observed violations of the independence axiom. In *Foundations of utility and risk theory with applications*, pages 263–293. Springer, 1983. 34

M.J. Machina. Expected utility hypothesis. In *Utility and Probability*, pages 79–95. Springer, 1990. 32

J.L. Marichal and P. Mathonet. A characterization of the ordered weighted averaging functions based on the ordered bisymmetry property. *IEEE Transactions on Fuzzy Systems*, 7(1):93–96, 1999. 16, 17

J. Marschak. Rational behavior, uncertain prospects, and measurable utility. *Econometrica: Journal of the Econometric Society*, pages 111–141, 1950. 35

A.W. Marshall, I. Olkin, and B.C. Arnold. *Inequalities: theory of majorization and its applications*, volume 143. Springer, 1979. 22

R. Mesiar and J. Špirková. Weighted means and weighting functions. *Kybernetika*, 42(2):151–160, 2006. 19

R. Mesiar, J. Špirková, and L. Vavríková. Weighted aggregation operators based on minimization. *Information Sciences*, 178(4):1133–1140, 2008. 19

R. Montemanni. A benders decomposition approach for the robust spanning tree problem with interval data. *European Journal of Operational Research*, 174(3):1479–1490, 2006. 25

H. Moulin. *Axioms of cooperative decision making*. Number 15. Cambridge University Press, 1991. 22, 23

Y. Nakamura. Risk attitudes for nonlinear measurable utility. *Annals of Operations Research*, 19(1):311–333, 1989. 24, 37, 38

W. Ogryczak. Inequality measures and equitable approaches to location problems. *European Journal of Operational Research*, 122(2):374–391, 2000. 17

Z. Páles. On the characterization of quasiarithmetic means with weight function. *aequationes mathematicae*, 32(1):171–194, 1987. 18

P. Perny. Modélisation des préférences, agrégation multicritere et systemes d'aide à la décision. *These d'habilitation, Université Pierre et Marie Curie, Paris*, 2000. 10, 13

J. Quiggin. *Generalized Expected Utility Theory: The Rank-Dependent Model*. Kluwer, 1993. 34, 40

K. Regan and C. Boutilier. Regret-based reward elicitation for markov decision processes. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 444–451, Arlington, Virginia, United States, 2009. AUAI Press. 25

R.A. Ribeiro and R.A.M. Pereira. Generalized mixture operators using weighting functions: A comparative study with WA and OWA. *European Journal of Operational Research*, 145(2):329–342, 2003. 17, 19

B. Roy. *Multicriteria methodology for decision aiding*, volume 12. Springer Science & Business Media, 2013. 12

R. Sabbadin. Possibilistic markov decision processes. *Engineering Applications of Artificial Intelligence*, 14(3):287 – 300, 2001. Soft Computing for Planning and Scheduling. 28

L.J. Savage. *The foundations of statistics*. Courier Corporation, 1972. 28

D. Schmeidler. Integral representation without additivity. *Proceedings of the American mathematical society*, 97(2):255–261, 1986. 28, 34

G. Shafer et al. *A mathematical theory of evidence*, volume 1. Princeton university press Princeton, 1976. 28

J. von Neumann and O. Morgenstern. *Theory of games and economic behaviour*. Princeton University Press, 1947. 31

R.R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on systems, Man, and Cybernetics*, 1988. 16, 23

H.P. Young. Extending Condorcet's rule. *Journal of Economic Theory*, 16(2):335–353, 1977. 37

S.X. Yu, Y. Lin, and P. Yan. Optimization models for the first arrival target distribution function in discrete time. *Journal of Mathematical Analysis and Applications*, 225(1):193–223, 1998. 36

Chapter 2

Combinatorial Optimization Problems and Sequential Decision Problems Under Risk

" I am always fascinated by the structure of things; why do things work this way and not that way. "

U. Wehrli

Section Contents

Summary of the chapter

In the previous chapter, we assumed that the different solutions of the problem could be listed explicitly. Such hypothesis does not always hold true as the number of possible solutions may be huge and even exponential in the size of the problem. These problems are said to be combinatorial as they are defined by several elements and the set of solutions can be composed of a (possibly huge) number of possible combinations of these elements. In this chapter, we present combinatorial optimization problems in general and succinctly describe the different combinatorial optimization problems that will be considered in this thesis. Then, we discuss a particular kind of combinatorial optimization problem, sequential decision problems under risk in which the decision maker has to make decisions under risk repeatedly in different situations and at different time steps.

2.1 Combinatorial Optimization Problems

This section gives a brief overview of the combinatorial optimization problems studied in this thesis [Papadimitriou and Steiglitz, 1982]. Its purpose is to introduce the different concepts from combinatorial optimization theory that are required for the rest of the thesis.

2.1.1 A General Presentation

In combinatorial optimization problems, the number of possible solutions is usually huge and often exponential in the size of the problem. Thus, it is impractical to explore all possible solutions. Fortunately, in these optimization problems, there exists a structure underlying the set of possible solutions. This structure can be exploited to perform efficiently the optimization of the problem and find, either an optimal solution or a "sufficiently good" solution.

More formally in a combinatorial optimization problem, a set $\mathrm{E} = \{1,\ldots,k\}$ of *elements* is given and a *solution* is defined as a subset of E. The set S of *admissible solutions* is then defined as the set of solutions satisfying a given set of mathematical constraints. The difficulty of combinatorial optimization problems is that the number of feasible solutions makes an exhaustive search impractical as k grows. Solving an optimization problem consists in finding an optimal feasible solution (i.e., a most preferred feasible solution). The preference relation on elements is defined by a cost function c which gives the cost $c(e)$ of including an element $e \in \{1,\ldots,k\}$ in a solution. This preference relation is then extended to feasible solutions by using an aggregation operator which is often the sum of the costs of the different elements of the solution.

Many well-known combinatorial optimization problems can be expressed in the context of *graph theory* [Bondy and Murty, 1976]. A graph $\mathscr{G} = (\mathscr{V},\mathscr{E})$ is an ordered pair consisting of a set $\mathscr{V}$ of vertices (also called nodes) together with a set $\mathscr{E}$ of edges or arcs between pairs of elements of $\mathscr{V}$. A graph is undirected if its edges have no orientation (i.e., edge (x,y) is identical to edge (y,x)) and directed otherwise. A graph is weighted if a weight is assigned to each edge. Such weights usually represent cost values but they may also represent lengths or capacities. Numerous concepts can be defined on graphs. One of the most intuitive one is the concept of *path*. Two nodes are adjacent when they are both incident to a common edge. More formally, a node v_i is adjacent to a node v_j if (v_i, v_j) belongs to $\mathscr{E}$. A path in a graph is a sequence of vertices $\mathrm{P} = (v_1, v_2, \ldots, v_n) \in \mathscr{V}^n$ such that v_i is adjacent to v_{i+1} for $1 \leq i \leq n-1$. Such a path is called a path of length $n-1$ from v_1 to v_n. In this thesis, we will only consider *simple paths* in which all nodes are distinct from one another (i.e., there is no cycles in a simple path). We denote by $\mathscr{P}_i^j$ the set of (simple) paths starting at node i and ending at node j.

Example 26. *We illustrate this notions on a simple graph represented in Figure 2.1. This directed graph has 6 nodes (i.e.,* $|\mathscr{V}| = 6$*) with* $\mathscr{V} = \{1,\ldots,6\}$ *and 9 edges (i.e.,* $|\mathscr{E}| = 9$*) with* $\mathscr{E} = \{(1,2),(1,3),(2,3),(2,4),(4,2),(3,4),(3,5),(4,6),(5,6)\}$*. The set of paths from node 1 to node 6 is composed of 5 paths,* $\mathscr{P}_1^6 = \{(1,2,4,6),(1,2,3,4,6),(1,2,3,5,6),(1,3,4,6),(1,3,5,6)\}$*. An exemple of path in* $\mathscr{P}_1^2$ *from node* 1 *to node* 2 *is* $(1,3,4,2)$*. The edges are here valued. For instance, the value assigned to edge* $(2,4)$ *is 3.*

To make the general framework more concrete, we now succinctly describe several combinatorial optimization problems that will be tackled in this thesis.

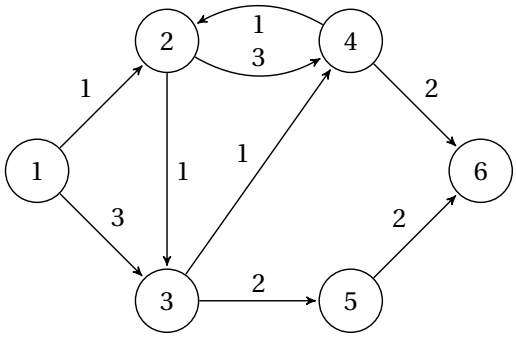


Figure 2.1: The graph of Example 26.

2.1.2 Various Combinatorial Optimization Problems

There exists numerous combinatorial optimization problems, too many to be listed here. We succinctly present here three of them that will be used as applications in this thesis.

The Shortest Path Problem. In the shortest path problem, a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is given where $\mathcal{V}$ is a set of vertices numbered from 1 to $|\mathcal{V}|$, and $\mathcal{E}$ is a set of edges. A starting node $s \in \mathcal{V}$ and a destination node $t \in \mathcal{V}$ are given and a cost value $c_{ij} \in \mathbb{R}^+$ is associated to each edge $(i, j) \in \mathcal{E}$, where c_{ij} is the cost induced by edge (i, j). In the shortest path problem, one wishes to determine a path $p \in \mathcal{P}_s^t$ that minimizes $\sum_{(i,j)\in p} c_{ij}$. We illustrate the problem with the two following examples:

Example 27. *In a first example, consider the graph represented in Figure 2.2 with 6 nodes (i.e., $|\mathcal{V}| = 6$) and 8 edges (i.e., $|\mathcal{E}| = 8$). We choose node 1 as the starting node and node 6 as the destination node, i.e., $s = 1$ and $t = 6$. The set of paths from node 1 to node 6 is composed of 5 paths, $\mathcal{P}_s^t = \{(1,2,4,6), (1,2,3,4,6), (1,2,3,5,6), (1,3,4,6), (1,3,5,6)\}$.*

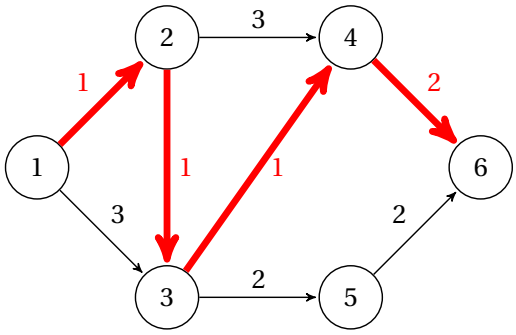


Figure 2.2: Shortest path problem of Example 27.

Here the optimal path is $(1,2,3,4,6)$, drawn with a bold red line on the graph, and has an optimal total cost of 5.

Example 28. *In a second example, consider the graph given in Figure 2.3 composed of 14 nodes and 33 edges and where the edge costs are not represented for readability reasons. This graph belongs to a particular class of graphs named* layered graphs. *A layered graph is composed of layers such that each layer is completely connected to the next layer. The graph represented in Figure 2.3 is composed of four layers, colored respectively in red, blue, green and yellow. We choose node 1 as the starting node and node 14 as the destination node, i.e., $s = 1$ and $t = 14$. This type of graph illustrates the combinatorial nature of the shortest path problem. Indeed, in this example, there are 81 paths going from node 1 to node 14. Moreover*

each additional layer would multiply this number by 3 (as there are 3 nodes in each layer). Therefore, the shortest path problem requires combinatorial optimization algorithms to be solved efficiently.

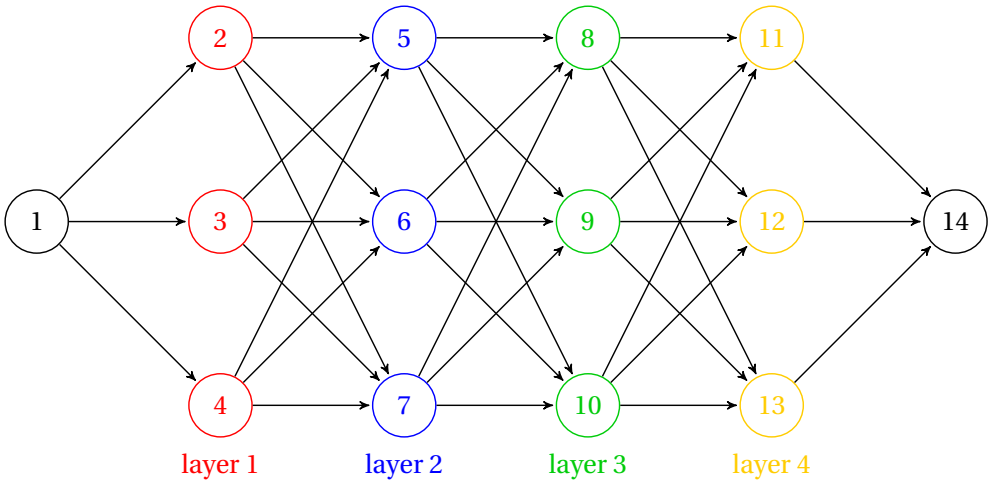


Figure 2.3: Shortest path problem of Example 28.

The most well known algorithms for solving this problem are: Dijkstra's algorithm [Dijkstra, 1971], Bellman-Ford algorithm [Bellman, 1958] or the A* search algorithm [Hart *et al.*, 1968].

These methods require to know the precise cost values of the edges of the graph. Unfortunately, for several reasons, these values may be known imprecisely. In this case, other solution methods must be used to find a robust solution (a presentation of robust optimization is given in Section 1.4 on page 25).

In Chapter 6, we study robust combinatorial optimization problems with interval data, i.e., each cost parameter c is only known to be in an interval $[\underline{c}, \overline{c}]$. We will present a game-theoretic view of these problems which makes it possible to design an efficient procedure to compute a lower bound on the value of an optimal minmax regret solution. We then describe how this lower-bounding procedure can be used in a branch and bound to find an optimal minmax regret solution. This approach is then instantiated and tested on the robust shortest path problem with interval data. Our numerical tests prove the efficiency of our approach for the robust shortest path problem with interval data.

Allocation of Indivisible Goods Problems. We now present allocation problems [Bouveret *et al.*, 2005; Lang and Rothe, 2016] in which a given number of goods must be allocated to a given number of agents such that all agents are satisfied with the allocation. More formally, the problem is defined by:

- a number n of agents;
- a number m of objects;
- $n \times m$ values v_{ij} such that v_{ij} is the utility value that agent i attributes to object j;
- for each agent i, a lower bound l_i and an upper bound u_i over the number of objects that agent i can receive.

If for all $i \in \{1,\dots,n\}$, $l_i = u_i = 1$ and $n = m$, then this problem is known as the assignment problem where the number of agents is the same as the number of objects and where each agent should receive exactly one object (no more, no less). Usually, in allocation problems, the goal is to find the optimal allocation (repartition of the goods to each agent) in order to maximize the sum of the utility values. These problems are illustrated by the following example:

Example 29. *Three books can be loaned at the local library, a thriller, a science fiction novel and a book on nutrition. Agathe, Lea and Sabine want to borrow a book and they agree that each of them will borrow one. They decide that each one of them should give their preferences with grades from 1 to 10 showing their interest for each book. Therefore, in this example $n = m = 3$ and for all $i \in \{1,2,3\}$, $l_i = u_i = 1$ which means that this is an assignment problem. The grades given to each book are the v_{ij} values (v_{ij} being the value at the i^{th} row and the j^{th} column) given below.*

	thriller	*science fiction novel*	*book on nutrition*
Agathe	6	***10***	2
Lea	6	4	***8***
Sabine	***8***	3	9

In this example, there are as many possible assignments as there are permutations of three elements, i.e., there are 6 feasible solutions. A similar assignment problem with n agents would have n! solutions which shows the combinatorial nature of the problem. In this example, the assignment that maximizes the sum of utilities consists in giving the science fiction novel to Agathe, the thriller to Sabine, and lastly the book on nutrition to Lea. This assignment (represented in bold and in red in the table above) has a total utility value of 26.

This problem can be solved efficiently by using the Hungarian method [Kuhn, 1955] for the assignment problem or by using the stepping-stone method [Charnes and Cooper, 1954] for more general allocation problems.

These methods find a feasible solution that maximizes the sum of utilities of the agents. However, this decision criterion does not guarantee anything about the fairness of the optimal solutions as a high utility for an agent can compensate for low utilities of other agents (a presentation of fair multi-agent optimization is given in Section 1.3 on page 21).

In Chapter 7, we study multi-agent optimization problems with two classes of aggregation operators, namely ordered weighted averages and mixture operators. A presentation of these two classes of operators in the setting of multi-agent optimization is given in Section 1.3. These operators are, under known conditions on their parameters, compatible with the Pigou-Dalton transfer principle. Thus, their optimization makes it possible to find fair solutions.
In the first part of the chapter, we develop a game-theoretic view of the determination of an optimal randomized strategy with ordered weighted average operators with decreasing weights and derive complexity results and solution methods from it. This approach is tested on the assignment problem.
In the second part of the chapter, we investigate the optimization of a mixture operator criterion in allocation problems and determine complexity results and solution methods.

The Proportional Representation Problem. We now present multi-winner voting problems, also known as proportional representation problems [Betzler *et al.*, 2013; Pro-

caccia *et al.*, 2008]. In proportional representation problems n voters express their preferences over m candidates with a view to an election. These preferences are expressed by a *preference profile* P of size $m \times n$, where the i^{th} column of P is the preference order of voter i. From profile P, nm utility values v_{ij} are derived that represent the level of satisfaction of voter i if she is represented by candidate j.

At the end of the election, k candidates are elected and each voter is affected to one of the winning candidates that will represent her. More formally, a solution of the proportional representation problem is a set of k winning candidates $\{c_1, \ldots, c_k\} \subseteq C$, and k sets S_j ($j \in \{c_1, \ldots, c_k\}$), where S_j is the subset of voters represented by candidate j. Therefore, a solution is denoted by a set $\{c_1, S_{c_1}; \ldots; c_k, S_{c_k}\}$ containing the k winning candidates c_i and the k sets S_{c_i}. Given a feasible solution, the utility v_i of voter i is then given by v_{ij} if i belongs to S_j in the solution. The initial formulation of the proportional representation problem aims to determine a solution that maximizes the sum of the voters' utilities.

Two main multi-winner voting schemes have been designed for proportional representation problems, namely *Monroe*'s Voting Scheme [Monroe, 1995] (abbreviated by MVS) and *Chamberlin-Courant*'s Voting Scheme [Chamberlin and Courant, 1983] (abbreviated by CCVS). While CCVS does not constraint the possible assignments from winning candidates to voters, MVS imposes that the k sets consisting of the voters represented by the same candidate should be equally sized.

We illustrate proportional representation problems in Example 30.

Example 30. *Florian is launching a new pasta restaurant. He knows how to cook 4 dishes: lasagna, pasta carbonara, pasta bolognese, pasta with pesto. However, for financial reasons, he can only choose 2 dishes for the menu. Therefore, in this problem $n = 5$, $m = 4$ and $k = 2$. He decides to ask 5 of his friends and probably best future clients their preferences, given below:*

Julien	*lasagna*	$\succ$	*carbonara*	$\succ$	*bolognese*	$\succ$	*pesto*
Edouard	*lasagna*	$\succ$	*bolognese*	$\succ$	*carbonara*	$\succ$	*pesto*
Amandine	*lasagna*	$\succ$	*bolognese*	$\succ$	*carbonara*	$\succ$	*pesto*
Hugo	*pesto*	$\succ$	*carbonara*	$\succ$	*lasagna*	$\succ$	*bolognese*
Fabien	*lasagna*	$\succ$	*carbonara*	$\succ$	*bolognese*	$\succ$	*pesto*

From this preference profile, Florian computes the utility values v_{ij} by using the formula $v_{ij} = 5 - \mathtt{rk}(j)$ where $\mathtt{rk}(j)$ is the rank of dish j in the preference order of voter i. These values are given below

	lasagna	*carbonara*	*bolognese*	*pesto*
Julien	**4**	3	2	1
Edouard	**4**	2	3	1
Amandine	**4**	2	3	1
Hugo	2	**3**	1	4
Fabien	4	**3**	2	1

A solution maximizing the sum of utilities of the agents under the CCVS is {lasagna,{Julien, Edouard, Amandine, Fabien}; pesto,{Hugo}} with a total utility value of 20. However, for logistic reasons, Florian may not accept to have a dish on the menu for only one customer. Therefore, it is probably more interesting for him to use the MVS for which a solution maximizing the sum of utilities of the agents is {lasagna,{Julien, Edouard, Amandine}; carbonara ,{Hugo,Fabien}} with a total utility value of 18. These optimal solutions are shown in blue and framed in an oval box for CCVS and in yellow and in a larger font for MVS (if an element is selected in both voting schemes then it is colored in green).

General proportional representation problems can be solved by integer programming [Potthoff and Brams, 1998]. Moreover, under the CCVS and if the preferences of the agents abide to specific structures, then dynamic programming methods can be used to optimize the sum of utilities of the agents [Betzler *et al.*, 2013; Skowron *et al.*, 2013].

In Example 30, we searched for a feasible solution maximizing the sum of utilities of the agents. This is known as the *utilitarian* version of proportional representation problems. However, as already pointed out, this decision criterion does not guarantee anything about the fairness of the optimal solutions.

In the second part of Chapter 7, we study proportional representation problems with a mixture operator criterion. A presentation of this operator in the setting of multi-agent optimization is given in Section 1.3. The optimization of this operator makes it possible (under known conditions on its parameters) to find fair solutions w.r.t. the Pigou-Dalton principle. Complexity results and solution methods are presented for the proportional representation problem with a mixture operator criterion.

The three combinatorial optimization problems that have just been presented are not equally difficult. More precisely, the proportional representation problem is known to be more difficult than the allocation problem and the shortest path problem, and no efficient algorithm (in a sense that will be made precise in the next subsection) is known to solve this problem under general preferences [Procaccia *et al.*, 2008]. The topic of studying the computational complexity of combinatorial optimization problems is called *computational complexity theory*.

2.1.3 Computational Complexity Theory

At this point of the chapter, it is important to make clear the distinction between a combinatorial optimization problem and an instance of a combinatorial optimization problem. In examples 27, 29 and 30, we have described instances of combinatorial optimization problems by specifying all the parameters of the associated problems. A combinatorial optimization problem is formally defined by precising the nature of its components and the objective that is sought; an infinity of instances are possible for a given combinatorial optimization problem. Put another way, coming back to example 27, what is presented in the example is an instance of the shortest path problem whereas the shortest path problem is the general problem of finding a path of minimal cost between two nodes of a valued graph.

To solve a combinatorial optimization problem, one must design an efficient algorithm which takes advantage of the structure of the problem to find an optimal solution given any possible instance of the problem. By efficient, we mean an algorithm which runs in a polynomial number of steps w.r.t. the size of the input string describing the instance of the problem. If such an algorithm exists, we say that it is polynomial time and we say that the combinatorial optimization problem is of polynomial complexity. The class of problems which are of polynomial complexity is denoted by $\mathtt{P}$. Note that other concepts related to efficiency could be used such as the amount of communication (used in communication complexity), the number of gates in a circuit (used in circuit complexity) or the number of processors (used in parallel computing).

If one does not succeed in designing such an algorithm, one may try to explain this

failure by showing that the problem tackled is "difficult". Computational complexity theory [Garey and Johnson, 2002] is a branch of computer science that classifies computational problems according to their inherent difficulty and relates those classes to each other. A problem is regarded as inherently difficult if its resolution requires significant resources (i.e., the number of steps cannot be polynomially bounded by the size of the instance) whatever the algorithm used.

To better formalize the notion of hard problems, we first need to explain the difference between an optimization problem and a decision problem[1]. Whereas an optimization problem searches for a solution that maximizes or minimizes an objective function, in computational complexity theory, a decision problem is a question that only accepts two possible answers, *yes* or *no*. Note that any maximization (resp. minimization) problem can always be associated to the decision problem that asks whether there exists a solution whose cost is superior (resp. inferior) to a given threshold θ. Thereby, an optimization problem can be solved by solving several instance of the associated decision problem with changing values of θ (with a binary search for instance).

To make the concept of decision problem more concrete, let us consider the subset sum problem. In the subset sum problem, a set S of integers is given and we wish to know if there exists a subset of this set whose elements sum to 0. This is a decision problem as there are only two possible answers to this problem, either yes or no. For instance, assume that we are given the set $S = \{-4,-1,2,5,8\}$. In this example, the answer is "yes", since the subset $\{-4,-1,5\} \subset S$ yields the sum $(-4)+(-1)+5=0$. Given an instance $\mathscr{I}$ of a decision problem, one can try to guess a solution which satisfies the decision problem to show that the answer is yes for instance $\mathscr{I}$. Such a guess is called a *certificate*. If the certificate indeed shows that the answer to $\mathscr{I}$ is "yes", then it is called a *witness*. An algorithm that verifies whether a given certificate for an instance $\mathscr{I}$ is a witness (in which case it returns *true*) or not (in which case it returns *false*) is called a *verifier*. For instance, in the subset sum problem, a certificate is a subset of S and a verifier consists in summing the elements of the certificate and comparing the result with 0.

To prove that a problem is "difficult", one can try to prove that it is as hard as a collection of problems that are known to be difficult because no one has found an efficient algorithm to solve them yet[2]. This is the case for the class of NP-*complete problems* where NP stands for "Non-deterministic Polynomial time".

Definition 30. *A decision problem belongs to* NP *if there exists a polynomial time verifier for this problem.*

For instance, the subset sum problem belongs to NP as the verifier which consists in summing the elements of the certificate and comparing the result with 0 is polynomial time.

Definition 31. *A problem is* NP*-hard if it is at least as hard as any problem in* NP *in the sense that any problem in* NP *can be reduced to this problem via a polynomial time transformation. Stated differently, if there was a polynomial time algorithm to solve this problem, there would be a polynomial time algorithm to solve any problem in* NP.

Therefore, to prove that a problem P is NP-hard it is enough to find a problem P' which is already known to be NP-hard and such that any instance $\mathscr{I}$ of P' can be reduced in polynomial time to an instance of P.

[1] Note that in computational complexity theory, the notion of decision problem as a specific meaning which is different from the one acknowledged so far in this thesis.

[2] and computer scientists have been searching for a long time...

Definition 32. *A decision problem is* NP*-complete if it is* NP*-hard and if it belongs to* NP.

It is an open question to know if NP-complete problems can be solved in polynomial time (P= NPhypothesis). If someone proves that her problem is NP-hard and then finds a polynomial time method to solve it, then she would answer positively this question. However, as researchers in computational complexity theory have been trying to answer this question for decades unsuccessfully, it is believed that there is few chances that someone will be able to do so: therefore, many believes that P⊂NP even if it has not been proved.

Thus, if a problem is NP-hard, other leads than the one of designing a polynomial time exact solving method may be investigated. We now specify some of these leads that are investigated in this thesis.

• 1. First, one can try to design efficient solution procedures even if they are not polynomial time. An example of such solution procedures are *branch and bound* methods [Lawler and Wood, 1966] which ingeniously explore the set of feasible solutions to quickly identify an optimal solution in most instances of the problem.

• 2. Second, to obtain a polynomial solution method, one can try to impose additional constraints on the problem studied as imposing a specific structure on the problem or bounding some parameters. For instance, the proportional representation problem is NP-hard but becomes polynomial if the preferences of the voters respect some specific constraints [Betzler *et al.*, 2013; Procaccia *et al.*, 2008; Skowron *et al.*, 2013]. As to bounding the numerical values of the problem, this idea leads to the notion of *pseudo-polynomial time methods*.

Definition 33. *An algorithm runs in pseudo-polynomial time if its running time is polynomial in the size of the input string describing the instance of the problem and in the largest numerical value of the input.*

The simplest example to illustrate this notion is the following. We wonder whether a number n is prime. To answer this question, we consider the algorithm which checks if any number in the set $\{2,3,\ldots,\sqrt{n}\}$ divides n. This approach can in the worst case take up to $\sqrt{n}-1$ divisions, which is polynomial in the value of n. Therefore, this is a pseudo-polynomial time algorithm. However, this algorithm is not polynomial time as $\sqrt{n}-1$ is exponential in the length of n, which is represented by $\log(n)$ bits.

• 3. Last, one can design a polynomial algorithm which may not be an exact solution method but which yields a feasible solution whose value is known to be close to the optimal one. Such an algorithm is called an *approximation algorithm* [Vazirani, 2013].

These different options will be investigated several times in this thesis.

In Chapter 4, we investigate the complexity of solving sequential decision problems under risk with the SSB and WEU models. These decision models are presented in Section 1.5 on page 34. We allow two different settings, according to the nature of the feasible solutions (deterministic or randomized). According to the setting considered and the representation used to model the problem (these representations are presented in the next section), we find problems that are either of polynomial complexity, or NP-hard. In some cases, we also present pseudo-polynomial solution methods.
In Chapter 6, we investigate robust combinatorial optimization problems with interval data and the minmax regret criterion. These problems are often NP-hard. Thus, we investigate a lower bounding procedure based on game-theoretic arguments. We show how to use this method efficiently in a branch and bound procedure. The lower bound returned by this solution can also be used to obtain an instance dependent approximation ratio for specific feasible solutions.
Lastly, in Chapter 7, we investigate the complexity of several problems: the optimization (in a randomized setting) of multi-agent decision problems with an ordered weighted average criterion and the optimization of allocation problems and proportional representation problems with a mixture operator criterion. According to the optimization problem considered and the fact that the preferences of the agents abide to specific constraints or not, we find complexities that are either polynomial or NP-hard.

We now turn in the next section to the presentation of the type of combinatorial optimization problems that has received most of our attention, namely sequential decision problems under risk.

2.2 Sequential Decision Making Under Risk

A sequential decision problem under risk is a decision problem in which a decision maker repeatedly makes choices in different situations and at different time steps, the consequences of which are subject to probabilities. The resolution of the problem consists in finding an optimal plan[3], i.e., a sequence of choices conditioned by events, optimal in the sense of the decision criterion used (see Section 1.5 for a description of various possible decision criteria).

Thus, the difficulty of finding an optimal plan is at least threefold:

- Firstly, the nature of the problem is of course combinatorial and the number of combinations of possible choices can be huge. This point will be illustrated in Example 31.
- Secondly, as the choices' consequences are uncertain, a plan results in a probability distribution over possible outcomes (i.e., a lottery) and comparing plans amounts to comparing their respective lotteries which is more difficult than comparing outcomes. These lotteries can be compared using models from decision theory, which leads to the domain of decision-theoretic planning [Blythe, 1999; Boutilier *et al.*, 1999].

[3] According to the context, we may also use the words *policy* or *strategy* to denote a plan.

- Lastly, the problem involves several time steps. To face this problem, the decision maker will have to project herself to anticipate future possible events and choices. This point will be particularly developed in subsection 2.2.3.

We now illustrate the concept of sequential decision problems under risk with an example.

Example 31. *Julien lives at the university residence. He has an urgent business to solve with his friend Edouard who is unfortunately not in his room. Julien concludes that Edouard can be in three different rooms. He can be in Amandine's room with probability* 0.6*, in Florian's room with probability* 0.3 *or in Christelle's room with probability* 0.1*. Amandine's (resp. Florian's) room is 10 meters on the right (resp. left) of Edouard's room and Christelle's room is 20 meters on the right of Edouard's room. Furthermore, if Julien unsuccessfully searches Edouard in Amandine's room there is a risk of* 50% *that she will ask him to go fetch an object in his room and bring it back to her which represents a trip of 30 meters. The arrangement of the rooms in the university residence is summarized in Figure 2.4. The goal of Julien is to minimize the distance that he will have to travel before finding Edouard.*

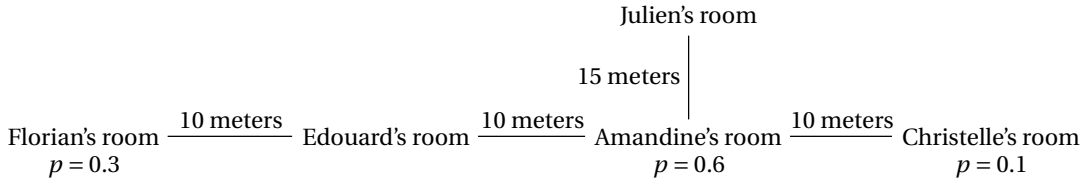


Figure 2.4: The arrangement of the rooms in the university residence.

The problem is sequential as Julien will have to make choices at different time steps (before visiting the first room, after visiting the first room, the second...) and in different physical positions. Note that his decisions may depend on the result of uncertain events as the fact that Amandine does or does not make a request. For instance a possible plan could be:

- *1. Look for Edouard in Amandine's room. If Edouard is found the problem ends. Otherwise, if Amandine makes a request, bring back the object and then:*
  - *2. Look for Edouard in Christelle's room.*
  - *3. If Edouard is not found in Christelle's room, look for Edouard in Florian's room.*

  If Amandine does not make a request:
  - *2. Look for Edouard in Florian's room.*
  - *3. If Edouard is not found in Florian's room, look for Edouard in Christelle's room.*

and the lottery over possible distances in meters associated to this plan is $(10m, 0.6; 30m, 0.15; 50m, 0.05; 60m, 0.05; 80m, 0.15)$*. This example illustrates the explosion of the number of solutions with the size of the problem. Here, there are three rooms in which we can look for Edouard. Even if we restrict the plans to the possible orders with which we can investigate the rooms, there would be as many plans as there are permutations of 3 elements (i.e., 6 plans). With 4 rooms, there would be 24 possible plans. With 5 rooms, there would be 120 possible plans...*

To represent and solve sequential decision problems under risk, different representations can be used. Those representations are based on tools from graph theory to represent the different time steps of the decision problems, the different choices available in each situation, and the probabilistic components of each choice. According to the representation used, it may be more or less easy to design a solution method and the problem may be more or less compactly represented (i.e., the memory space required to represent the problem can be more or less substantial). The tendency is that, the more a representation is simple, the easier it is to design a solving procedure and the less compact is the representation. In the rest of this section, we present two well known representations of sequential decision problems, namely, *decision trees* and *Markov decision processes*. Note that other compact representations to represent sequential decision problems under risk, such as influence diagrams [Howard and Matheson, 1981] or factored Markov decision processes [Boutilier *et al.*, 2000], are possible.

2.2.1 Decision Trees

A decision tree is a graphical model which enables to represent sequential decision problems with a particular type of graphs which are named *trees*.

Definition 34. *A tree of root* N_r *is an oriented graph where the following conditions hold:*

- *for all nodes* N, (N,N_r) *is not an arc of the graph.*
- *there exists a path from* N_r *to any other node.*
- *two different paths cannot have identical nodes at their extremities.*

In a tree, the parent of a node N_c *is the only node* N_p *such that* $(\mathrm{N}_p, \mathrm{N}_c)$ *is an arc of the tree. The successors of a node* N_p *(also called children) are all the nodes* N_c *such that* N_p *is the parent of* N_c*. In a tree, a leaf is a node* N_t *which has no children. Given a node* N *of the tree, the subtree rooted at* N *is the tree composed of* N *and all the subtrees rooted at children of* N*. The depth of a node* N *is defined as the length of the path between the root* N_r *and* N*. Lastly, the height of a tree is the length of the longest path between the root and a leaf of the tree.*

A decision tree [Raiffa, 1993] is a particular type of tree which enables to represent a sequential decision problem with three types of nodes:

- The *decision nodes* that we will graphically represent by squares. They correspond to situations in which the decision maker must make decisions. The branches starting from a decision node correspond to the different possible decisions that can be made.
- The *chance nodes* that we will graphically represent by circles. They correspond to situations that are uncertain. The branches starting from a chance node correspond to different possible events. Each branch is weighted by the probability (which is known) of the corresponding possible event.
- The *terminal nodes* (the leaves of the tree). The values indicated at the leaves correspond to the resulting outcomes.

In this thesis, we will make the assumption that successors of decision nodes (resp. chance nodes) are always either chance nodes (resp. decision nodes) or terminal nodes.

Generally, a decision tree is represented horizontally with the root at the leftmost part of the tree such that the chronological order of the possible events can be read from left to right.

Such a decision tree is given in Figure 2.5. This decision tree involves 2 decision nodes D_1 and D_2, 2 chance nodes C_1 and C_2 and 4 terminal nodes T_1, T_2, T_3 and T_4. Note that this decision tree is related to Allais' paradox which was presented in Example 21 on page 33. Indeed in node D_2, the agent needs to choose between lotteries $l_1 = (3000\$, 1)$ and $l_2 = (0\$, 0.2; 4000\$, 0.8)$ while in node D_1, the two possible plans yield lotteries $l'_1 = (0\$, 0.75; 3000\$, 0.25)$ and $l'_2 = (0\$, 0.8; 4000\$, 0.2)$. These lotteries are exactly the same as the lotteries l_1, l_2, l'_1 and l'_2 defined in Example 21.

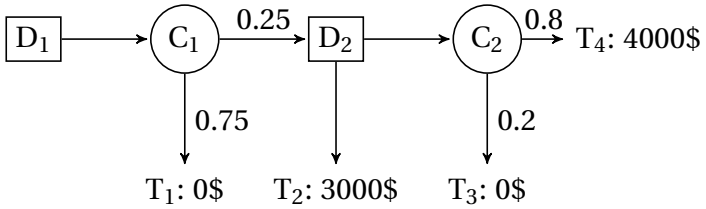


Figure 2.5: Allais' paradox as a decision tree problem.

We now make more formal the definition of a decision tree. A decision tree $\mathscr{T}$ is composed of a set of nodes $\mathscr{N}$ and a set of edges $\mathscr{E}$. The root node is denoted by N_r and the set of decision nodes, chance nodes and terminal nodes are respectively denoted by $\mathscr{N}_D$, $\mathscr{N}_C$ and $\mathscr{N}_T$. For instance, in the decision tree represented in Figure 2.5, $N_r = D_1$, $\mathscr{N}_D = \{D_1, D_2\}$, $\mathscr{N}_C = \{C_1, C_2\}$ and $\mathscr{N}_T = \{T_1, T_2, T_3, T_4\}$. Additionally, we define $\mathscr{E}_D := \{(D, N) \in \mathscr{E} : D \in \mathscr{N}_D\}$, the set of edges starting from a decision node. Every edge $E = (C, N) \in \mathscr{E}$ such that $C \in \mathscr{N}_C$ is associated with the probability p_E of the corresponding event; every terminal node $T \in \mathscr{N}_T$ is labeled by the resulting outcome $o(T)$. For instance, in the decision tree represented in Figure 2.5, edge (C_2, T_4) is associated with the probability $p_{(C_2,T_4)} = 0.8$ and node T_2 is labelled by the outcome $o(T_2) = 3000\$$. Lastly, $\mathscr{S}(N)$ denotes the set of successors of N. For instance, in the decision tree represented in Figure 2.5, $\mathscr{S}(C_1) = \{D_2, T_1\}$.

We now formalize the notion of plan in the context of decision trees. A plan in a decision tree is called a *strategy*. We can distinguish three types of strategies: *deterministic strategies, randomized strategies* and *mixed strategies*.

Deterministic strategies. A deterministic strategy δ corresponds to a set $\mathscr{E}_\delta \subseteq \mathscr{E}_D$. The set of all feasible deterministic strategies is denoted by Δ. Let $\chi^\delta := (\chi_E)_{E \in \mathscr{E}_D}$ denote the incidence vector of a deterministic strategy δ (i.e., $\chi_E = 1$ if $E \in \mathscr{E}_\delta$ and $\chi_E = 0$ otherwise). The set of feasible deterministic strategies can be characterized as the incidence vectors satisfying the following constraints:

$$\sum_{N \in \mathscr{S}(N_r)} \chi_{(N_r,N)} = 1$$

$$\forall D \in \mathscr{N}_D :$$

$$\text{if } \chi_{(D,C)} = 0 \text{ then } \sum_{N \in \mathscr{S}(D')} \chi_{(D',N)} = 0 \quad \forall D' \in \mathscr{S}(C) \cap \mathscr{N}_D$$

$$\text{otherwise } \sum_{N \in \mathscr{S}(D')} \chi_{(D',N)} = 1 \quad \forall D' \in \mathscr{S}(C) \cap \mathscr{N}_D.$$

For instance, in the decision tree of Figure 2.5, $\mathscr{E}_{\mathrm{D}}$ includes the three edges E_1=(D_1,C_1), E_2=(D_2,C_2) and E_3=(D_2,T_2), and there are two possible deterministic strategies characterized by incidence vectors (χ_{E_1}=1,χ_{E_2}=1,χ_{E_3}=0) and (χ_{E_1}=1,χ_{E_2}=0,χ_{E_3}=1).

By abuse of notation, we will denote a strategy indifferently by δ or χ^{δ}. Following the same convention, we will denote by Δ both the set of feasible deterministic strategies and the set of corresponding incidence vectors.

Randomized strategies. *Randomized strategies* are obtained by relaxing the domain of variables χ_{E} from $\{0,1\}$ to $[0,1]$ while keeping the same constraints. The value of $\chi_{(\mathrm{N}_1\mathrm{N}_2)}$ is then the probability of taking edge $(\mathrm{N}_1\mathrm{N}_2)$ given the fact that we are in node N_1. For instance, in the decision tree represented in Figure 2.5, an example of a randomized strategy is given by vector ($\chi_{\mathrm{E}_1}=1,\chi_{\mathrm{E}_2}=0.5,\chi_{\mathrm{E}_3}=0.5$). This corresponds to the situation where the decision maker in decision node D_2 will choose with equal probabilities to go either in T_2 or in C_2. We denote by Δ_r the set of feasible randomized strategies (or the corresponding incidence vectors).

Mixed strategies. Instead of using a randomized strategy, a decision maker can rely on a *mixed strategy*. A mixed strategy is simply a lottery over deterministic strategies. Put another way, using the mixed strategy $\delta^m=(\delta_1,p_1;\ldots;\delta_k,p_k)$ consists in sampling a deterministic strategy δ_i according to δ^m and following δ_i thereafter. We denote by $\widetilde{\Delta}$ the set of feasible mixed strategies.

As usual in sequential decision under risk, a strategy δ (whatever its type) induces a lottery l_δ over outcomes. Let p^{δ}_{N} denote the probability to reach node N from node N_r when following strategy δ. Values p^{δ}_{N} can be recursively computed as follows for deterministic and randomized strategies:

$$
\begin{aligned}
p^{\delta}_{\mathrm{N}_r} &= 1\\
\forall \mathrm{D}\in\mathscr{N}_{\mathrm{D}},\forall \mathrm{N}\in\mathscr{S}(\mathrm{D}),\ p^{\delta}_{\mathrm{N}} &= p^{\delta}_{\mathrm{D}} * \chi^{\delta}_{(\mathrm{D},\mathrm{N})}\\
\forall \mathrm{C}\in\mathscr{N}_{\mathrm{C}},\forall \mathrm{N}\in\mathscr{S}(\mathrm{C}),\ p^{\delta}_{\mathrm{N}} &= p^{\delta}_{\mathrm{C}} * p_{(\mathrm{C},\mathrm{N})}.
\end{aligned}
$$

For a mixed strategy $\delta^m=(\delta_1,p_1;\ldots;\delta_k,p_k)$, values $p^{\delta^m}_{\mathrm{N}}$ can be computed with the following formula:

$$
\forall \mathrm{N}\in\mathscr{N},\ p^{\delta^m}_{\mathrm{N}}=\sum_{i=1}^{k} p_i\, p^{\delta_i}_{\mathrm{N}}.
$$

The lottery over outcomes l_δ induced by a strategy δ is then defined by $l_\delta(x)=\sum_{\mathrm{T}\in\mathscr{N}_{\mathrm{T}}:o(\mathrm{T})=x} p^{\delta}_{\mathrm{T}}$ for all $x\in\mathbb{R}$.

Let $\mathscr{L}=\{l_\delta:\delta\in\Delta\}$ (resp. $\mathscr{L}_r=\{l_\delta:\delta\in\Delta_r\}$) denote the image of deterministic (resp. randomized) strategies in the space of lotteries, and $\widetilde{\mathscr{L}}=\{l_\delta:\delta\in\widetilde{\Delta}\}$ denote the image of mixed strategies in the space of lotteries. It is worth noting that $\widetilde{\mathscr{L}}=\mathscr{L}_r$. This is due to the fact that given a strategy in $\widetilde{\Delta}$ it is possible to determine a corresponding strategy in Δ_r which induces the same lottery and *vice-versa*. We will return to this point in the following chapters. As a consequence of this observation, optimizing over $\mathscr{L}_r$ (with randomized strategies) is equivalent to optimizing over $\widetilde{\mathscr{L}}$ (with mixed strategies).

Note that the number of deterministic strategies can possibly grow exponentially with the size of the decision tree, i.e., the number of decision nodes n_d (this number has indeed the same order of magnitude as the number of nodes in $\mathscr{T}$).

For instance consider a complete binary decision tree $\mathscr{T}$ of height $2h$. There is one decision node of depth 0, 4 decision nodes of depth 2, 16 decision nodes of depth 4 and so on. Therefore the number n_d of decision nodes equals $\sum_{i=0}^{h-1} 4^i = \frac{4^h-1}{3}$. For instance, in the complete binary decision tree in Figure 2.6, the length of the tree is equal to 4 (i.e., $h = 2$) and there are $(4^2-1)/3 = 5$ decision nodes.

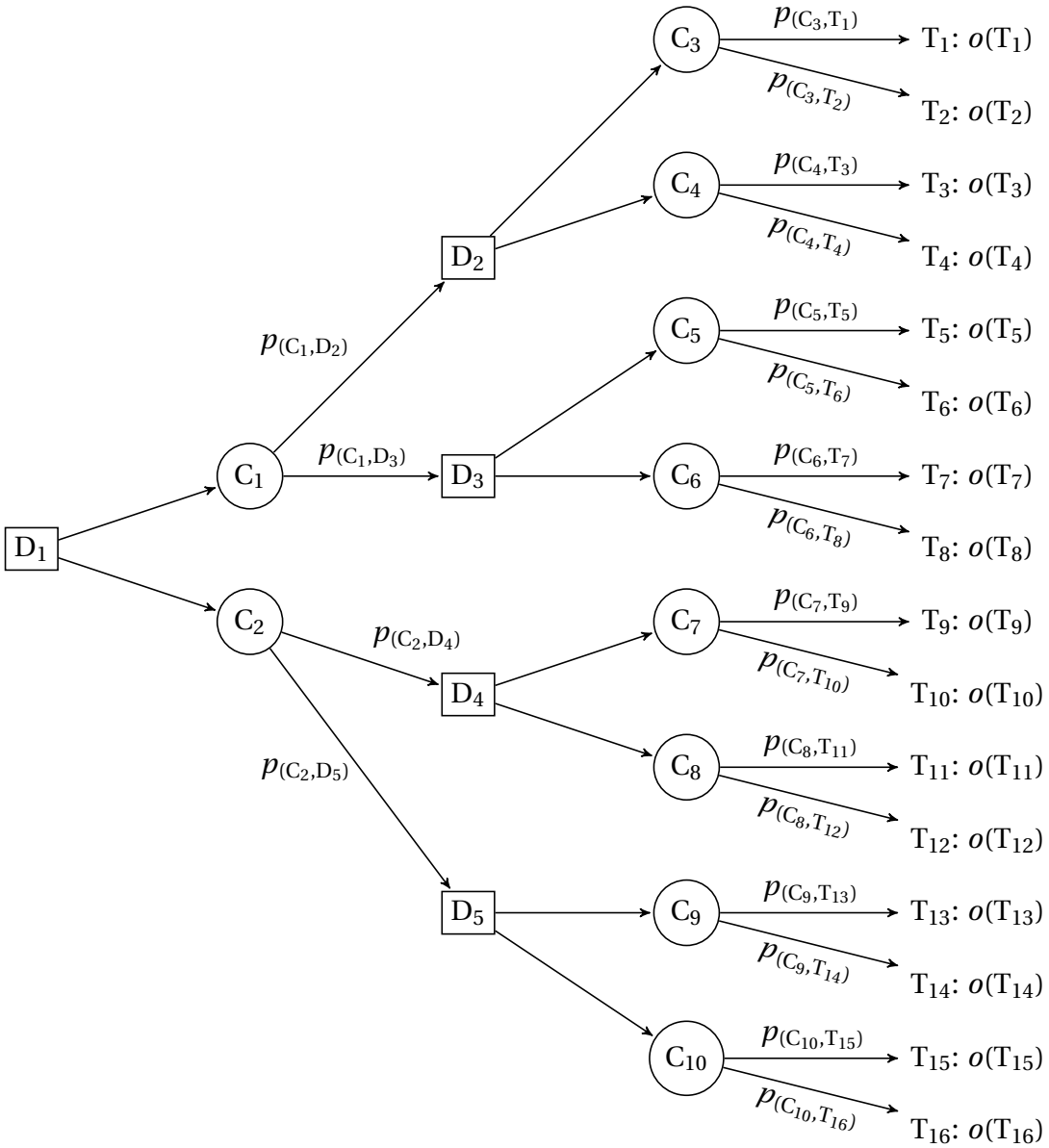


Figure 2.6: Complete binary decision tree of height 4.

Let me count the number of deterministic strategies as a function of the height of $\mathscr{T}$. We proceed by induction. There are 2 possible deterministic strategies in a complete binary decision tree with only 1 decision node (i.e., a tree of height 2). We use the following recursive rules. The number of deterministic strategies from a chance node is the product of the numbers of possible strategies from its two children. The number of deterministic strategies from a decision node is the sum of the numbers of possible strategies from its two children. Therefore, the number of possible deterministic strategies from a decision node can be computed via the recursive second order sequence $u_0 = 2$, $u_k = 2u_{k-1}^2$, where k denotes the number of decision nodes on a path from the considered node (which is not counted) to a leaf of the tree. The general term of this sequence is $2^{2^{k+1}-1}$. It is easy to

see that $k = h - 1$ at the root. Thus the total number of deterministic strategies from the root is $2^{2^h-1} \in \Theta(2^{\sqrt{n}})$ as $n = \frac{4^h-1}{3}$. For this reason, even in the deterministic setting, it is necessary to develop an optimization algorithm to determine an optimal strategy.

For EU, such an optimal strategy can be determined by a dynamic programming method as the rolling back method. The rolling back method consists in starting from the leaves and rolling back the tree until reaching the root. At each new node the optimal substrategy is computed and the corresponding actions are irrevocably included in the optimal strategy being built, where a substrategy is the restriction of a strategy in $\mathscr{T}$ to a subtree.

This method can be used to compute in linear time an optimal strategy for EU because the optimality principle holds: any substrategy of an optimal strategy is itself optimal. Starting from the leaves, one computes recursively for each node the EU of an optimal substrategy: the optimal EU for a chance node equals the expectation of the optimal utilities of its successors; the optimal EU for a decision node equals the maximum EU of it's successors.

For instance, consider the decision tree represented in Figure 2.5 and assume we would like to compute the optimal strategy according to the EU model with utility function $u : x \mapsto \sqrt{x}$. The rolling back procedure would compute the utilities in the following order:

1. The expected utility of node C_2 is: $\mathrm{EU}(C_2) = 0.8u(4000) + 0.2u(0) \approx 50.6$.

2. The expected utility of node D_2 is: $\mathrm{EU}(D_2) = \max(\mathrm{EU}(C_2), u(3000)) = u(3000) \approx 54.8 \Rightarrow (D_2, T_2)$ is added to the optimal strategy.

3. The expected utility of node C_1 is: $\mathrm{EU}(C_1) = 0.75u(0) + 0.25\mathrm{EU}(D_2) \approx 13.7$.

4. The expected utility of node D_1 is: $\mathrm{EU}(D_1) = \max(\mathrm{EU}(C_1)) = \mathrm{EU}(C_1) \approx 13.7 \Rightarrow (D_1, C_1)$ is added to the optimal strategy.

and the optimal strategy is $\{(D_1, C_1), (D_2, T_2)\}$.

However as argued in Chapter 1 (in Section 1.5) the EU model is not able to account for behaviors that have been observed in various experiments. This limitation is highlighted in several paradoxes as the intransitive dice paradox (described in Example 19 on page 33), the preference reversal paradox (described in Example 20 on page 33) and Allais' paradox (described in Example 21 on page 33). Therefore decision models with more descriptive abilities should be investigated to solve decision trees.

Unfortunately, note that criteria able to account for intransitive and/or non-independent preferences, violate the Bellman optimality principle. This is illustrated by the decision tree related to Allais' paradox (represented in Figure 2.5) in which a criterion compatible with Allais' paradox would have $\{(\mathbf{D_2}, \mathbf{T_2})\}$ as optimal substrategy from node D_2 and $\{(D_1, C_1), (\mathbf{D_2}, \mathbf{C_2})\}$ as optimal strategy from node D_1. Put another way, the optimal strategy from node D_1 would include a suboptimal strategy from node D_2. Thus, the rolling back method cannot be used to solve decision trees with a decision model able to account for intransitive and/or non-independent preferences. This models are therefore difficult to optimize in decision trees.

In Chapter 4, we investigate the optimization of the SSB and WEU criteria in decision trees. These decision models extend EU and have stronger descriptive abilities (see subsection 1.5.3 on page 34). We allow two different settings, according to the nature of the feasible solutions, one in which only deterministic strategies are considered and one in which randomized strategies are also allowed. For the SSB model, we design solution methods for the randomized setting. These methods are either based on game-theoretic arguments or on linear programming. For the WEU model, we design solution methods for both settings. These methods are based on fractional programming (which is a domain presented in the next chapter).

The framework of decision trees is one of the most intuitive and easy representation for sequential decision making. While this representation is not compact, designing a solving procedure is relatively easy in this model. We now discuss another representation which is more compact, the one of Markov decision processes.

2.2.2 Markov Decision Processes

We now present, in few words, the setting of Markov Decision Processes[4] (MDP) which offers a general and powerful formalism to model compactly and solve sequential decision problems under risk [Derman, 1970; Puterman, 2014]. Note that, the thesis especially focusses on finite MDPs, that is, MDPs whose state and action spaces are finite.

Definition 35. *An MDP $\mathscr{M}$ is formally described by a tuple* $(\mathrm{T}, \mathscr{S}, \mathscr{A}, \mathscr{P}, c, \gamma)$ *where:*

- T*, a positive integer, is the* time horizon*, i.e., the number of time steps, also called decision epochs, for which the decision maker is making a plan. If* T *is finite (resp. infinite) we say that the problem is a finite (resp. infinite) horizon problem. In the infinite horizon case, the agent acts forever unless a* goal *state is reached (i.e., a state where there is no possible actions and that leads to no other states);*

- *$\mathscr{S}$ is a finite collection of states, one of which is designated as the initial state and denoted by s_0[5]. These states correspond to the different situations in which the decision maker can be. The state in which the decision maker is at time step t is denoted by s_t.*

- *$\mathscr{A} = \{\mathscr{A}_s | s \in \mathscr{S}\}$ is a collection of finite sets of possible actions, one set for each state. The set $\mathscr{A}_s$ contains the possible choices that the decision maker can make in state s. By abuse of notation, we may also denote by $\mathscr{A}$ the union $\cup_{s\in\mathscr{S}}\mathscr{A}_s$;*

- *$\mathscr{P} = \{\mathscr{P}_t | t = 0, \ldots, \mathrm{T}-1\}$ is a collection of functions where for each $t \in \{0, \ldots, \mathrm{T}-1\}$, $\mathscr{P}_t : \mathscr{S} \times \mathscr{A}_{s_t} \times \mathscr{S} \to \mathbb{R}$ is a transition function where $\mathscr{P}_t(s'|s,a)$ is the probability of reaching state s' at time step $t+1$ when action a is performed in state s at time step t;*

- *$c = \{c_t | t = 0, \ldots \mathrm{T}-1\}$ is a collection of functions where for each $t \in \{0, \ldots, \mathrm{T}-1\}$, $c_t : \mathscr{S} \times \mathscr{A}_{s_t} \times \mathscr{S} \to \mathbb{R}$ is a reward function where $c_t(s,a,s')$ is the reward obtained if the*

[4] This presentation is obviously partial and an interested reader is redirected towards the book by Puterman [2014] for a more complete presentation.

[5] Note that this assumption is not restrictive. If instead there should be a probability distribution P_0 over initial states, one can create a dummy initial state with only one action leading to all other states with probability P_0.

state at time step $t+1$ is s' given that the state at time step t was s and that we have performed action a;

- *and $\gamma \in [0,1)$ is a discount factor that may be used in the decision criterion and which induces a time preference. More formally, when $\gamma < 1$, the same amount of reward has more value if received sooner than later.*

Note that in the setting of MDPs, we assume that all this information is known. This differs from the setting of reinforcement learning [Sutton and Barto, 1998] in which the transition and reward functions are unknown and must be learned. Furthermore, note that the MDP literature usually considers rewards that do not depend on the next state. This is because, with the standard decision criteria of the MDP literature, it is harmless to replace values $c_t(s,a,s')$ by $c_t(s,a) = \sum_{s' \in \mathcal{S}} \mathcal{P}_t(s'|s,a) c_t(s,a,s')$, i.e., the expectation over all possible next states of reward values obtained when performing action a in state s at time step t. However, with the sophisticated decision criteria that we will use in this thesis, this property does not hold.

In a nutshell, at each time step t, the agent knows her current state s_t. According to this state, she decides to perform an action a_t. This action results in a new state $s_{t+1} \in \mathcal{S}$ according to probability distribution $\mathcal{P}_t(.|s_t,a_t)$, and a reward $c_t(s_t,a_t,s_{t+1})$ which penalizes or reinforces the choice of this action. This process is illustrated on Figure 2.7. The qualifier "Markov" is used because the transition probabilities and rewards depend on the past only through the current state s_t and action a_t. At time step $t=0$, the agent is in the initial state s_0. We will call *t-history* h_t a succession of t state-action pairs starting from state s_0, e.g., $h_t = (s_0, a_0, s_1, \ldots, s_{t-1}, a_{t-1}, s_t)$. We call *episode* a T-history and denote $\mathcal{E}$ the set of episodes.

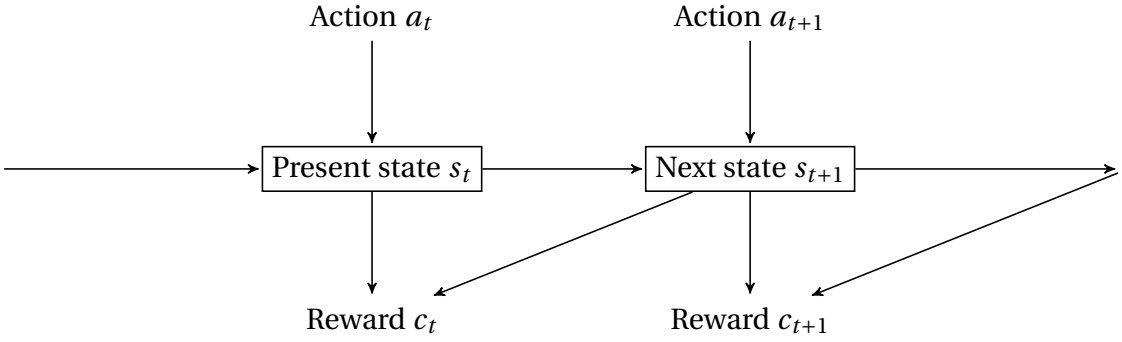


Figure 2.7: Symbolic representation of sequential decision problems in MDPs.

We illustrate the notations on two simple sequential decision problems.

Example 32. *A robotic agent tries to solve a maze. The maze is composed of a 3×3 grid leading to 9 possible physical positions. More formally, $\mathcal{S} = \{s_{xy} : x,y \in \{1,2,3\}\}$. The initial state is s_{11} and the end of the maze is located in state s_{33}. In each state the robot can try to go left, up, right or down leading to 4 possible actions (i.e., $\mathcal{A}_s = \{l,u,r,d\}, \forall s \in \mathcal{S}$). If the action is not possible because of a wall, the action fails and the robot stays in the same cell (i.e., $\mathcal{P}_t(s|s,a) = 1$). If the action is possible, then the action succeeds with probability 0.8 (for instance $\mathcal{P}_t(s_{(x+1)y}|s_{xy},r) = 0.8$). However, with probability 0.2, an error of transmission occurs and the robot goes in a random direction. The planning horizon T is here infinite and the state s_{33} is terminal (i.e., the robot stops acting once state s_{33} has been reached). Lastly, the cost function is defined as follows: $c_t(s_t,a_t,s_{t+1}) = 0$ if $s_{t+1} = s_{33}$ and $c_t(s_t,a_t,s_{t+1}) = -1$ otherwise.*

Example 33. *A decision maker is faced with the following game, for three consecutive time steps she can decide to throw a coin. If heads appears, the decision maker wins* 4$*. Otherwise, if tails appears, she losses all the money she earned so far. The MDP is here composed of two states: s_{in} meaning that the agent is still playing the game and s_{out} in which either the agent has lost the game or has decided to stop playing. In state s_{in}, two actions are possible, either play (denoted by p) or quit (denoted by q). If she decides to play in state s_{in}, she has equal chances of ending up in state s_{out} or in state s_{in}. If she quits, the agent moves to state s_{out} with probability one. The planning horizon* T *is here equal to 3. The reward function is defined in the following way: $c_t(s_t, a_t, s_{t+1}) = 0$ if $a_t = q$; $c_t(s_t, p, s_{in}) = 4$; $c_t(s_t, p, s_{out}) = 0$ (resp. $-4, -8$) if t = 0 (resp. 1,2). The corresponding MDP is represented in Figure 2.8. Note that this example illustrates the fact that the MDP representation is more compact than the one of decision trees which requires 3 decision nodes, 3 chance nodes and 7 terminal nodes.*

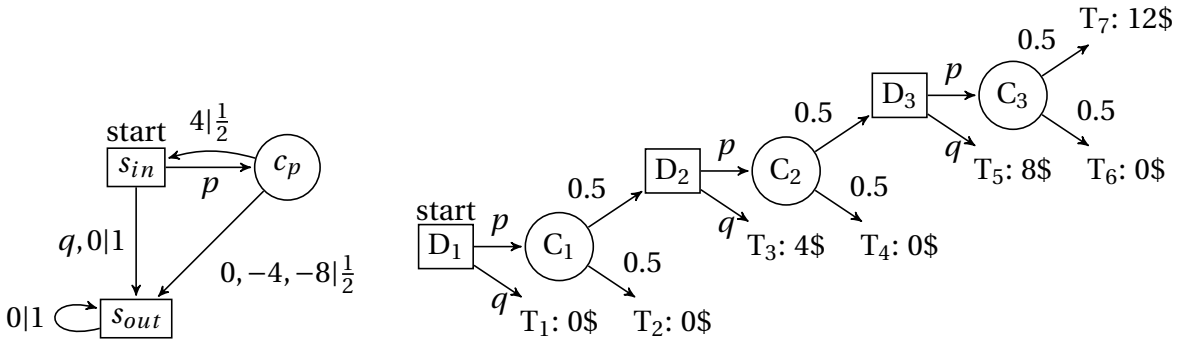


Figure 2.8: An MDP representing the decision problem in Example 33 (left) and a decision tree representing the same problem (right).

The goal of the agent is to determine a policy, i.e., a procedure to select an action in each possible situation, that is optimal for a given criterion. More formally, a *policy* π at an horizon T is a sequence of T decision rules $(\delta_0, \ldots, \delta_{T-1})$. *Decision rules* are functions which prescribe the actions that the agent should perform. A decision rule can be *history-dependent* meaning that it takes as argument the entire history generated so far or *Markovian* if it only takes as argument the current state. Moreover, a decision rule is either *deterministic* if, given an argument, it always selects the same action, or *randomized* if it prescribes a probability distribution over possible actions. A policy can be *history-dependent*, *Markovian*, *deterministic* or *randomized* according to the type of its decision rules. Lastly, a Markovian policy is *stationary* if it uses the same decision rule at every time step, i.e., $\pi = (\delta, \delta, \ldots)$, and non-stationary if the decision rules can be different, depending on the time step. By abuse of notation, if the policy is stationary, we may omit the notation of decision rules δ and use the one of policies π instead.

We use the notations of Table 2.1 for the sets of the different types of policies and we illustrate the relations between these different classes in the lattice of Figure 2.9, where an arrow goes from a more general class of policies to a more specific class of policies. Importantly, given a set $\Pi = \{\pi_1, \pi_2, \ldots\}$ of policies, we define an enlarged set $\widetilde{\Pi}$ of policies, that denotes the set consisting of mixtures of policies, i.e., $\widetilde{\Pi} = \{\tilde{\pi} = (\pi_1, \alpha_1; \pi_2, \alpha_2; \ldots) : \sum_i \alpha_i = 1, \alpha_i \geq 0\}$, where $\tilde{\pi}$ is the *mixed policy*[6] that randomly selects policy π_i with probability α_i and follows it thereafter.

[6]Not to be confused with the notion of randomized policies.

		Markovian	history-dependent
deterministic	non-stationary	Π_s^t	Π_h
	stationary	Π_s	-
randomized	non-stationary	$\Pi_{s,r}^t$	$\Pi_{h,r}$
	stationary	$\Pi_{s,r}$	-

Table 2.1: Policy notations

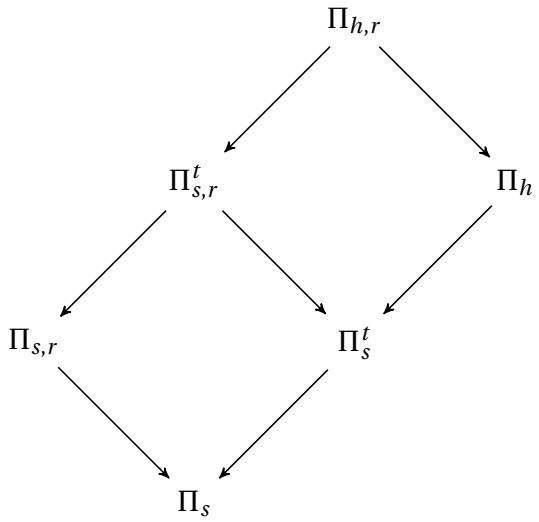


Figure 2.9: The relationships among different classes of policies.

Given a decision criterion, an important question in MDPs consists in determining the most specific class of policies which surely contains an optimal policy. Indeed, focusing on the most simple types of policies is desirable not only from a normative point of view but also from a computational point of view.

To make the relation clear with the terminology of decision making under risk, for MDPs, the set of outcomes is the set of all possible values of cumulated rewards along an episode, and every policy induced a lottery over cumulated rewards.

We now detail basic results on MDPs. Note that the finite horizon case is distinguished from the infinite horizon case which is more difficult.

Finite horizon MDPs. In a finite horizon MDP, the decision maker makes T decisions at the first T decision epochs of the problem and then stops making decisions. The episode is considered to be over. The planning horizon T can be suggested by the problem itself or set by the decision maker.

In finite horizon MDPs, different criteria can be defined in order to compare policies. One standard criterion is the *expected total cumulated reward*, for which it is known that an optimal deterministic Markovian policy exists at any horizon T [Puterman, 2014]. This criterion is defined as follows. First, the value $\rho(h_t)$ of a history $h_t = (s_0, a_0, s_1, \ldots, s_{t-1}, a_{t-1}, s_t)$ is defined as the sum of rewards/costs obtained along it, i.e., $\rho(h_t) = \sum_{i=0}^{t-1} c_i(s_i, a_i, s_{i+1})$. Then, the value of a policy $\pi = (\delta_0, \ldots, \delta_{T-1})$ in a state s is set to be the expected value of the episodes that can be generated by π from s if s is the initial state. This value, given by

the *value function* $v_0^\pi : \mathscr{S} \to \mathbb{R}$ can be computed iteratively as follows:

$$\begin{aligned} \forall s \in \mathscr{S}, \quad & v_T^\pi(s) = 0 \\ \forall s \in \mathscr{S}, \forall t \in \{0,\dots,T-1\}, \quad & v_t^\pi(s) = \sum_{s' \in \mathscr{S}} \mathscr{P}_t(s'|s,\delta_t(s))(c_t(s,\delta_t(s),s') + v_{t+1}^\pi(s')) \end{aligned} \tag{2.1}$$

The value $v_t^\pi(s)$ is the expectation of cumulated rewards obtained by the agent from time step t if she performs action $\delta_t(s)$ in state s at time step t and continues to follow policy π thereafter. Note that, alternatively, the decision maker can investigate the use of another action than $\delta_t(s)$ by considering the value $Q_t^\pi(s,a)$ which gives the expectation of cumulated rewards obtained by the agent if she performs action a in state s at time step t and continues to follow policy π thereafter. This function, named Q-function (which indicates the value of doing an action a in a state s), can be computed as follows:

$$\forall s \in \mathscr{S}, \forall a \in \mathscr{A}_s, \forall t \in \{0,\dots,T-1\}, \quad Q_t^\pi(s,a) = \sum_{s' \in \mathscr{S}} \mathscr{P}_t(s'|s,a)(c_t(s,a,s') + v_{t+1}^\pi(s')) \tag{2.2}$$

The higher the values of $v_t^\pi(s)$ and $Q_t^\pi(s,a)$ are, the better. Interestingly, the value function induces a preference relation $\succsim_\pi$ over policies in the following way:

$$\pi \succsim_\pi \pi' \Leftrightarrow \forall s \in \mathscr{S}, \forall t = 1,\dots,T, v_t^\pi(s) \geq v_t^{\pi'}(s)$$

A solution to an MDP is a policy, called *optimal policy*, that ranks the highest with respect to $\succsim_\pi$. Such a policy can be found by solving the *Bellman optimality equations*.

$$\forall s \in \mathscr{S}, \quad v_T^*(s) = 0 \tag{2.3}$$

$$\forall s \in \mathscr{S}, \forall t \in \{0,\dots,T-1\}, \quad v_t^*(s) = \max_{a \in \mathscr{A}_s} \sum_{s' \in \mathscr{S}} \mathscr{P}_t(s'|s,a)(c_t(s,a,s') + v_{t+1}^*(s')) \tag{2.4}$$

Indeed, an optimal deterministic Markovian policy consists in choosing in each state one of the actions realizing the maximum value in right-hand side of Equation 2.4. This shows the importance of the value function which makes it possible to determine an optimal policy.

Note that the total reward criterion does not induce preferences over time. Stated differently, with this criterion, the agent is indifferent between receiving a reward at time step t and receiving the same reward at time step $t' > t$. However, a decision maker could prefer to receive a reward as soon as it is possible. To account for such preferences, one can use a discount factor γ which penalizes rewards exponentially w.r.t. the time step at which they are received. This criterion is called the *expected total discounted reward* criterion. With this criterion, the value $\rho(h_t)$ of a history $h_t = (s_0, a_0, s_1, \dots, s_{t-1}, a_{t-1}, s_t)$ is defined as the sum of discounted rewards obtained along it, i.e., $\rho(h_t) = \sum_{i=0}^{t-1} \gamma^i c_i(s_i, a_i, s_{i+1})$. Then, the value of a policy $\pi = (\delta_0, \dots, \delta_{T-1})$ in a state s is set to be the expected value of the histories that can be generated by π from s and can be computed iteratively as follows:

$$\begin{aligned} \forall s \in \mathscr{S}, \quad & v_T^\pi(s) = 0 \\ \forall s \in \mathscr{S}, \forall t \in \{0,\dots,T-1\}, \quad & v_t^\pi(s) = \sum_{s' \in \mathscr{S}} \mathscr{P}_t(s'|s,\delta_t(s))(c_t(s,\delta_t(s),s') + \gamma v_{t+1}^\pi(s')) \end{aligned} \tag{2.5}$$

The Q-function and the preferences over policies are then easily adapted from the total reward criterion case.

In both settings, the Bellman optimality equations can be solved by using the *backward induction* procedure (presented in Algorithm 1 for the expected total cumulated reward criterion), which simply follows from the optimality equations. This procedure is a dynamic programming method similar to the rolling back method presented for decision trees. The backward induction procedure returns a Markovian deterministic optimal policy.

Algorithm 1: Backward Induction

Data: An MDP $\mathscr{M} = (\mathrm{T}, \mathscr{S}, \mathscr{A}, \mathscr{P}, c, \gamma)$.
Result: The optimal values v_t^* and an optimal deterministic Markovian policy π^*.

$t \leftarrow \mathrm{T}$
for $s \in \mathscr{S}$ **do**
  $v_t^*(s) = 0$
for $t \leftarrow \mathrm{T}-1$ *downto* 0 **do**
  for $s \in \mathscr{S}$ **do**
    $v_t^*(s) \leftarrow \max_{a \in \mathscr{A}_s} \sum_{s' \in \mathscr{S}} \mathscr{P}_t(s'|s,a)(c_t(s,a,s') + v_{t+1}^*(s'))$
    choose $d_t^*(s) \in arg\max_{a \in \mathscr{A}_s} \sum_{s' \in \mathscr{S}} \mathscr{P}_t(s'|s,a)(c_t(s,a,s') + v_{t+1}^*(s'))$
$\pi^* = (d_0^*, \ldots, d_{\mathrm{T}-1}^*)$

The backward induction method enables to find an optimal policy in finite horizon MDP with the total (discounted or not) reward criterion. However, as argued in Section 1.5 (on page 31), this criterion suffers from serious limitations as illustrated in the Saint-Petersburg paradox. Furthermore, this criterion is not risk sensitive. To address these issues, one can change the decision criterion by using the expected utility model.

The most popular type of utility functions for MDPs is the exponential one. Indeed, it is well known that the standard methods to solve MDPs can easily be adapted to this setting [Koenig and Simmons, 1994]. If another type of utility function is used, the problem becomes more difficult and an augmentation of the state space is required [Iwamoto, 2004; Spanjaard and Weng, 2013]. This was first noticed by White [1987] who considered EU of the total cumulated rewards as decision criterion for MDPs. Recently, this problem was thoroughly investigated in the thesis of Liu [Liu and Koenig, 2006, 2008; Liu, 2005].

However, as argued in Chapter 1 (in Section 1.5), the EU model is not able to account for observed behaviors as stressed by the intransitive dice paradox (see Example 19 on page 33), the preference reversal paradox (see Example 20 on page 33) and Allais' paradox (see Example 21 on page 33). Therefore decision models with more descriptive abilities should be investigated for MDPs even if they are more difficult to optimize.

In the second part of Chapter 4, we investigate the optimization of the SSB and WEU criterion in MDPs. These decision models extend EU and have stronger descriptive abilities (see subsection 1.5.3 on page 34). We design solution methods to find the optimal randomized SSB policy in an augmented version of a finite horizon MDP. These methods are either based on game-theoretic arguments or on linear programming. For the WEU model, we design solution methods to find an optimal policy in an augmented version of a finite horizon MDP. These methods are based on fractional programming (which is a domain presented in the next chapter).

Moreover, the backward induction method requires to know the exact values of the reward function. This hypothesis may be unrealistic in some problems as it may be hard to determine the values of some parameters. Furthermore, the number of parameters involved in an MDP can be huge (several thousand parameters) and determining the value of each parameter may simply be impossible. Unfortunately, the optimal policy can be very sensitive to these numerical values.

In Chapter 5, we address this problem by developing an ordinal approach based on a quantile criterion to solve finite horizon MDPs where only an order over possible episodes is known. We present a solving procedure which uses a sequence of backward induction methods to find an optimal policy in this ordinal setting.

Infinite horizon MDP. Infinite horizon problems occur either because the horizon is very large and can be considered infinite or because the problem is everlasting. In infinite horizon problems, we will consider that the rewards and transition functions do not depend on the time step and we will therefore omit the corresponding index. The expected outcome over an infinite horizon is defined as the limit of a series of finite-horizon expected costs as the horizon tends towards infinity. More formally, the value of state s for policy π is defined as $v^\pi(s) = \lim_{T\to\infty} v^\pi_T(s)$. The expected value exists if and only if the limit converges towards either a finite value, or positive infinity, or negative infinity. However, to make the comparison of policies "easy", not only is it desirable for these values to exist, but they should also be finite. Indeed, it is not possible to compare policies with infinite expected values.

For this reason, it is popular to use the expected total discounted reward criterion in the infinite horizon case. The expected total discounted reward that can be obtained from a state s given a policy π is defined as:

$$\forall s \in \mathscr{S}, \quad v^\pi(s) = \mathbb{E}[\sum_{i=0}^{\infty} \gamma^i c_i(s_t, a_t, s_{t+1}) | s_0 = s, \pi]$$

and the optimal expected total discounted cost of state s is $v^*(s) = \sup_\pi v^\pi(s)$. Discounting is a common technique that insures, if the state and action spaces are finite, that for all possible episodes, the total discounted rewards are finite, and therefore the states values always exist and are finite. Furthermore, it is well known that for infinite horizon MDPs with the expected total cumulated discounted reward criterion, there exists an optimal policy which is stationary and deterministic [Puterman, 2014].

A stationary deterministic Markovian policy $\pi : \mathscr{S} \to \mathscr{A}$ associates an action to each state. As in the finite horizon case, such a policy is evaluated by a *value function* $v^\pi : \mathscr{S} \to \mathbb{R}$ and a Q-*function* $Q^\pi : \mathscr{S} \times \mathscr{A} \to \mathbb{R}$, defined as follows:

$$\forall s \in \mathscr{S}, \quad v^\pi(s) = \sum_{s' \in \mathscr{S}} \mathscr{P}(s'|s, \pi(s))(c(s, \pi(s), s') + \gamma v^\pi(s')) \tag{2.6}$$

$$\forall s \in \mathscr{S}, \forall a \in \mathscr{A}_s, \quad Q^\pi(s, a) = \sum_{s' \in \mathscr{S}} \mathscr{P}(s'|s, a)(c(s, a, s') + \gamma v^\pi(s')) \tag{2.7}$$

Then a preference relation is defined over policies by: $\pi \succsim \pi' \Leftrightarrow \forall s \in \mathscr{S}, v^\pi(s) \geq v^{\pi'}(s)$. A solution to an MDP is a policy, called *optimal policy*, that ranks the highest with respect to

$\succsim$. Once again, such a policy can be found by solving the *Bellman optimality equations.*

$$\forall s \in \mathscr{S}, \quad v^*(s) = \max_{a \in \mathscr{A}_s} \sum_{s' \in \mathscr{S}} \mathscr{P}(s'|s,a)(c(s,a,s') + \gamma v^*(s')).$$

An optimal stationary deterministic policy π^* is then determined in the following way:

$$\forall s \in \mathscr{S}, \quad \pi^*(s) \in \arg\max_{a \in \mathscr{A}_s} \sum_{s' \in \mathscr{S}} \mathscr{P}(s'|s,a)(c(s,a,s') + \gamma v^*(s')).$$

To solve an infinite horizon MDP, one must determine an optimal policy or an ϵ-optimal policy π_ϵ for which $|v^{\pi_\epsilon} - v^*| \leq \epsilon$. The most well known solution procedures for infinite MDPs are dynamic programming methods such as the value iteration and policy iteration procedures. In this section, we only present these two methods, but note that other solution methods exist such as those based on linear programming [Altman, 1995; Puterman, 2014].

Algorithm 2: Value Iteration

Data: An MDP $\mathscr{M} = (T, \mathscr{S}, \mathscr{A}, \mathscr{P}, c, \gamma)$, where $T = \infty$ and $0 < \gamma < 1$ and an accuracy parameter, $\epsilon > 0$.

Result: An ϵ-optimal value function $v^{*,\epsilon}$ and an ϵ-optimal stationary deterministic Markovian policy $\pi^{*,\epsilon}$

1 $t \leftarrow 0$
2 **do**
3 **for** $s \in \mathscr{S}$ **do**
4 $v^{t+1}(s) \leftarrow \max_{a \in \mathscr{A}_s} \sum_{s' \in \mathscr{S}} \mathscr{P}(s'|s,a)(c(s,a,s') + \gamma v^t(s'))$
5 $t \leftarrow t + 1$
6 **while** $||v^t - v^{t-1}|| > \frac{1-\gamma}{2\gamma}\epsilon$
7 $v^{*,\epsilon} \leftarrow v^t$
8 **for** $s \in \mathscr{S}$ **do**
9 choose $\pi^{*,\epsilon}(s) \in \arg\max_{a \in \mathscr{A}_s} \sum_{s' \in \mathscr{S}} \mathscr{P}(s'|s,a)(c(s,a,s') + \gamma v^{*,\epsilon})$

The value iteration procedure given in Algorithm 2 is a generalization of the backward induction procedure to infinite horizon problems. Value iteration uses a value function to approximate the optimal values. Starting from any value function, the method iteratively improves the approximation of the optimal value function by using the update equation in Line 4, until the approximation error is sufficiently small. When the approximation is close enough, an ϵ-optimal policy is determined in Line 9.

The policy iteration procedure given in Algorithm 3, iteratively improves a policy until an optimal policy is found. Starting from an arbitrary policy, the method alternates between a policy evaluation step (Line 4) and an improvement step (Line 6) until the policy is optimal.

The value iteration procedure and policy iteration procedure enable to find (near-)optimal policies in infinite horizon MDPs. However, this methods require to know the exact values of the reward function. This hypothesis may be unrealistic in some problems as it may be hard to determine the values of some parameters. Unfortunately, the optimal policy is extremely sensitive to these numerical values.

Algorithm 3: Policy Iteration

Data: An MDP $\mathcal{M} = (T, \mathcal{S}, \mathcal{A}, \mathcal{P}, c, \gamma)$, where $T = \infty$ and $0 < \gamma < 1$.
Result: An optimal value function v^* and an optimal stationary deterministic Markovian policy π^*.

$t \leftarrow 0$
do
  solve the system of equations
  $v^t(s) = \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, \pi^t(s))(c(s, \pi^t(s), s') + \gamma v^t(s')), \forall s \in \mathcal{S}$
  for $s \in \mathcal{S}$ **do**
    choose $\pi^{t+1}(s) \in \arg\max_{a \in \mathcal{A}_s} \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, a)(c(s, a, s') + \gamma v^t(s'))$, setting $\pi^{t+1}(s) = \pi^t(s)$ if possible.
  $t \leftarrow t + 1$
while $\pi^{t+1} \neq \pi^t$
$v^* \leftarrow v^t$
$\pi^* \leftarrow \pi^t$

In Chapter 5, we address this problem by improving a procedure that interleaves the elicitation of the reward function and the resolution of the MDP [Weng and Zanuttini, 2013]. We assume that the rewards of an infinite horizon MDP are imprecisely known and we refine an interactive version of the value iteration procedure which searches for an optimal policy while refining its knowledge about the possible reward functions, by querying an *expert* when needed. Our goal is to reduce the number of preference queries issued by this procedure.

We have now presented the two representations of sequential decision making under risk that we will use in this thesis, namely decision trees and MDPs. For each representation, we illustrated how an optimal plan could be found for a given decision criterion (EU for decision trees and the expected total cumulated discounted reward for MDPs). We presented some limits of the usual approaches used for these models and mentioned how these limits will be addressed in the thesis. We now conclude this chapter by giving a broader picture of the different types of possible behaviors in sequential decision making under risk.

2.2.3 Different Types of Behaviors in Sequential Decision Making

One difficulty in sequential decision problems under risk is that the decision maker must make decisions at different time steps. For this reason, the decision maker must be able to anticipate the possible future events and her possible future choices. However, the choices that seem optimal to her now may seem more questionable in a near future. If the decision maker is aware of these possible changes of perceptions, she may adapt herself in different manners leading to different types of behaviors that we detail now.

A *consequentialist* decision maker [Hammond, 1988] selects a plan looking only at the possible futures (regardless of the past or counterfactual history); among consequentialist decision makers, the *sophisticated* ones select a plan starting from the anticipated future decisions and rolling back to the present (in a recursive manner); on the contrary, a *resolute* decision maker [McClennen, 1990] considers all possible plans from the present time, chooses one and commits to it without deviating thereafter. The behavior of a res-

olute decision maker is therefore non-consequentialist because her choices in the future will depend on past and counterfactual events.

Note that an EU optimizer is both sophisticated and resolute. This is related to the fact that the Bellman optimality principle holds for the EU criterion (i.e., a subplan of an optimal plan is always optimal). However, it is not the case for criteria able to account for intransitive and/or non-independent preferences because, as already pointed out, they violate the Bellman optimality principle. Hence, a decision maker whose behavior is consistent with a model able to account for intransitive and/or non-independent preferences is thus either sophisticated or resolute. If the decision maker has intransitive or non-independent preferences, the drawback of acting as a sophisticated decision maker is the possibility to end up with a stochastically dominated plan [Hammond, 1988]. We illustrate this point in Example 34.

Example 34. *As in the Gardner's dice example (page 33), assume a decision maker has three different options* A, B *and* C *and prefers* A *to* B, B *to* C, *and* C *to* A. *It is reasonable to assume that she refuses a small amount* ε *of money to replace* A *(resp.* B, C*) by* B *(resp.* C, A*). Consider now a sequential decision problem where the decision maker initially receives die* A *for free, then is proposed to switch for* B. *If she accepts to change for* B, *then she is proposed to switch for* C. *Again, if she accepts the change, then she is proposed to switch for* A. *Each exchange is rewarded by* ε. *This corresponds to the decision tree of Figure 2.10.*

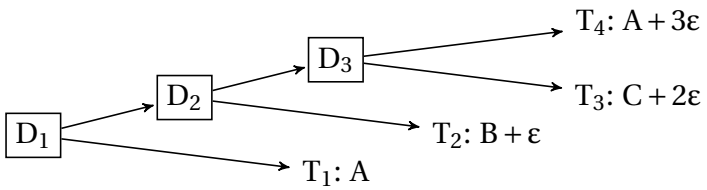


Figure 2.10: Decision tree in Example 34.

A sophisticated decision maker anticipates that she will prefer $C+2\varepsilon$ *to* $A+3\varepsilon$ *in decision node* D_3 *and* $B+\varepsilon$ *to* $C+2\varepsilon$ *in decision node* D_2. *Hence, in node* D_1 *she decides to keep* A *as she prefers* A *to* $B+\varepsilon$. *However, a better strategy is obviously to accept all changes.*

Interestingly, a resolute decision maker whose preferences are consistent with stochastic dominance is subject to none of these drawbacks. For a resolute decision maker, there are several manners to define an optimal strategy:

- *Dictatorship of the root*: the "root" refers to the situation at the start of the decision process; with this criterion, the decision maker chooses a plan that maximizes her decision criterion at the root. The use of this criterion has been advocated by McClennen [1990].

- *Resolute choice with selves*: it consists in considering the present decision situation as well as each possible future decision situations as a self who represents the decision maker at the time and state when the decision is made [Jaffray, 1998; Nielsen and Jaffray, 2006]; one then aims to determine a plan achieving a compromise between the different selves of a resolute decision maker, i.e., a strategy that remains suitable for all selves.

We illustrate these different types of behaviors in Example 35.

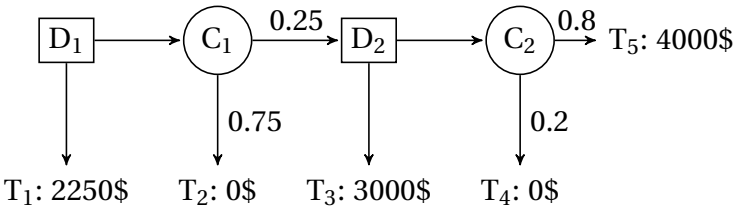


Figure 2.11: Decision tree in Example 35.

Example 35. *We consider the decision tree in Figure 2.11 and we restrict our attention to deterministic strategies. In this decision tree, there are three deterministic strategies from the root node (i.e., node* D_1*). These strategies are denoted by* δ_1*,* δ_2 *and* δ_3 *and are characterized by incidence vectors:*

$$\xi^{\delta_1} = \left(\xi^{\delta_1}_{(\mathrm{D}_1,\mathrm{T}_1)} = 0, \xi^{\delta_1}_{(\mathrm{D}_1,\mathrm{C}_1)} = 1, \xi^{\delta_1}_{(\mathrm{D}_2,\mathrm{C}_2)} = 1, \xi^{\delta_1}_{(\mathrm{D}_2,\mathrm{T}_3)} = 0\right);$$
$$\xi^{\delta_2} = \left(\xi^{\delta_2}_{(\mathrm{D}_1,\mathrm{T}_1)} = 0, \xi^{\delta_2}_{(\mathrm{D}_1,\mathrm{C}_1)} = 1, \xi^{\delta_2}_{(\mathrm{D}_2,\mathrm{C}_2)} = 0, \xi^{\delta_2}_{(\mathrm{D}_2,\mathrm{T}_3)} = 1\right);$$
$$\xi^{\delta_3} = \left(\xi^{\delta_3}_{(\mathrm{D}_1,\mathrm{T}_1)} = 1, \xi^{\delta_3}_{(\mathrm{D}_1,\mathrm{C}_1)} = 0, \xi^{\delta_3}_{(\mathrm{D}_2,\mathrm{C}_2)} = 0, \xi^{\delta_3}_{(\mathrm{D}_2,\mathrm{T}_3)} = 0\right).$$

Moreover, in the subtree rooted at node D_2*, there are two deterministic substrategies that can be used. These strategies are denoted by* δ'_1 *and* δ'_2 *and are characterized by incidence vectors:*

$$\xi^{\delta'_1} = \left(\xi^{\delta'_1}_{(\mathrm{D}_2,\mathrm{C}_2)} = 1, \xi^{\delta'_1}_{(\mathrm{D}_2,\mathrm{T}_3)} = 0\right) \text{ and } \xi^{\delta'_2} = \left(\xi^{\delta'_2}_{(\mathrm{D}_2,\mathrm{C}_2)} = 0, \xi^{\delta'_2}_{(\mathrm{D}_2,\mathrm{T}_3)} = 1\right).$$

Note that strategy δ'_1 *(resp.* δ'_2*) is the substrategy of* δ_1 *(resp.* δ_2*) rooted at node* D_2*. Strategies* δ_1*,* δ_2*,* δ_3*,* δ'_1 *and* δ'_2 *induce the following lotteries on the set of outcomes of the decision tree:*

- *From node* D_1*:* $l_{\delta_1} = (0\$, 0.8; 4000\$, 0.2)$*;* $l_{\delta_2} = (0\$, 0.75; 3000\$, 0.25)$*;* $l_{\delta_3} = (2250\$, 1)$*;*
- *From node* D_2*:* $l_{\delta'_1} = (0\$, 0.2; 4000\$, 0.8)$*;* $l_{\delta_{2'}} = (3000\$, 1)$*.*

Assume that we have a preference relation $\succsim$ *over the set of lotteries (and a fortiori over policies) which yields the following preferences:*

$$l_{\delta_1} \succ l_{\delta_3} \succ l_{\delta_2} \text{ and } l_{\delta'_2} \succ l_{\delta'_1}.$$

Thus, in node D_2*, the decision maker prefers* δ'_2 *to strategy* δ'_1*. However, in node* D_1*, the decision maker prefers* δ_1 *(which includes* δ'_1 *as a substrategy) to strategy* δ_3 *and* δ_2 *(which includes* δ'_2 *as a substrategy). Note that these preferences are reasonable as they are almost identical to the ones observed in Allais paradox (Example 21 on page 33).*

How would a sophisticated/resolute decision maker act in this decision tree?

- *A sophisticated decision maker solves a decision tree by rolling it back from the leaves. Notably, she anticipates that in node* D_2*, she will prefer strategy* δ'_2 *to strategy* δ'_1*. Therefore, she discards strategy* δ_1 *by imposing that the only acceptable substragy in node* D_2 *is* δ'_2*. Hence, in node* D_1 *she finally decides to use strategy* δ_3 *as* $l_{\delta_3} \succ l_{\delta_2}$*.*
- *A resolute decision maker with "dictatorship of the root" seeks an optimal strategy from the root node (i.e., node* D_1*). Thus, it will select strategy* δ_1 *which is the preferred strategy in node* D_1 *(as* $l_{\delta_1} \succ l_{\delta_3} \succ l_{\delta_2}$*) and will not deviate from it.*

- *A resolute decision maker with "selves" considers two selves: one for node* D_1 *and one for node* D_2*. These selves represent a projection of the decision maker in the different decision nodes of the decision tree. The idea is to find a compromise between this two selves. This idea can be implemented in several ways. We succinctly describe what such an implementation could look like (in the spirit of [Jeantet* et al.*, 2012]).*

  Each self is then associated to a loss function. The loss function associated to the self of node D_1 *(resp.* D_2*) is denoted by* $\mathtt{Loss}_1$ *(resp.* $\mathtt{Loss}_2$*). Function* $\mathtt{Loss}_1$ *(resp.* $\mathtt{Loss}_2$*) maps each strategy* $\delta \in \Delta$ *that reaches node* D_1 *(resp.* D_2*) with non-zero probability to the loss value obtained for playing* δ *instead of the optimal strategy w.r.t. the self of node* D_1 *(resp.* D_2*). For instance, in this example, possible loss functions (i.e., compatible with the preferences of each self) could be:*

$$\mathtt{Loss}_1(\delta_1) = 0, \quad \mathtt{Loss}_1(\delta_2) = 2, \quad \mathtt{Loss}_1(\delta_3) = 1$$
$$\mathtt{Loss}_2(\delta_1) = 2, \quad \mathtt{Loss}_2(\delta_2) = 0$$

  Then, an optimal strategy is determined by minimizing a function of these loss values. For instance, if we minimize the maximum loss over reachable selves, then an optimal strategy is δ_3 *with an optimal loss value of* 1*.*

We would like to emphasize the dichotomy between the resolute approach with root dictatorship and the sophisticated approach. While the first one is defined by a *dictatorship of the root*, the second one can be seen as a *dictatorship of the leaves*. The resolute choice with selves approach seems to be one way to make a compromise between these two extreme approaches.

In this thesis, when the decision model is not EU or the expected total (discounted) reward, we study the resolute approach with dictatorship of the root. This will be the case for the SSB and WEU models in Chapter 4 as well as for a quantile criterion in Chapter 5.

2.3 Conclusion

In this section, we have presented the setting of combinatorial optimization problems. In the first section, we made a general presentation of this setting, including the description of several combinatorial optimization problems and the presentation of the main notions of computational complexity theory. Then, in a second section, we have particularly developed sequential decision problems under risk which are the combinatorial optimization problems that have received the most attention in this thesis. In each section, we have mentioned where the notions and problems introduced were used and investigated in the thesis.

Solving sequential decision making problems under risk in decision trees and MDPs will be explored in Chapter 4 and 5. While Chapter 4 focuses on the SSB and the WEU model, Chapter 5 focuses on imprecise and ordinal pieces of information and requires elicitation procedures or adapted criteria. The other combinatorial optimization problems presented at the beginning of this section (shortest path problem, affectation problem, proportional representation problem) are investigated in Chapter 6 and 7. While Chapter 6 addresses robust combinatorial optimization problems with the minmax regret criterion, Chapter 7 focuses on fair multi-agent problems with the OWA and MO criteria. This information is summarized in Table 2.2.

To solve this problems efficiently, we will often rely on powerful algorithmic tools named *oracle* methods. The goal of the next chapter is to present these methods.

Problem \ Criterion	Minmax regret	OWA	MO	SSB	WEU	Quantile criterion	Expected total discounted reward
Shortest path problem	Chapter 6						
Allocation problem		Chapter 7 in Section 7.2	Chapter 7 in Section 7.3				
Proportional representation problem			Chapter 7 in Section 7.3				
Decision Tree				Chapter 4 in Section 4.2	Chapter 4 in Section 4.2		
Finite horizon MDP				Chapter 4 in Section 4.3	Chapter 4 in Section 4.3	Chapter 5 in Section 5.2	
Infinite horizon MDP							Chapter 5 in Section 5.3

Table 2.2: Summary table answering the questions: "Which problems are studied? With which criteria? Where?"

2.4 References

E. Altman. Constrained markov decision processes, 1995. 70

R. Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958. 50

N. Betzler, A. Slinko, and J. Uhlmann. On the computation of fully proportional representation. *Journal of Artificial Intelligence Research*, 47:475–519, 2013. 51, 53, 55

J. Blythe. Decision-theoretic planning. *AI Magazine*, 20(2):37–54, 1999. 56

J.A. Bondy and U.S.R. Murty. *Graph theory with applications*, volume 290. Macmillan London, 1976. 48

C. Boutilier, T. Dean, and S. Hanks. Decision theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999. 56

C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial intelligence*, 121(1):49–107, 2000. 58

S. Bouveret, M. Lemaître, H. Fargier, and J. Lang. Allocation of indivisible goods: a general model and some complexity results. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1309–1310. ACM, 2005. 50

J.R. Chamberlin and P.N. Courant. Representative deliberations and representative decisions: Proportional representation and the borda rule. *American Political Science Review*, 77(03):718–733, 1983. 52

A. Charnes and W.W. Cooper. The stepping stone method for explaining linear programming calculations in transportation problems. *Management Science*, 1(1):49–69, 1954. 51

C. Derman. *Finite state Markovian decision processes*. Mathematics in science and engineering. Academic Press, 1970. 63

E.W. Dijkstra. *A short introduction to the art of programming*, volume 4. Technische Hogeschool Eindhoven Eindhoven, 1971. 50

M.R. Garey and D.S. Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002. 54

P.J. Hammond. Consequentialist foundations for expected utility. *Theory and decision*, 25(1):25–78, 1988. 71, 72

P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. 50

R.A. Howard and J.E. Matheson. *Influence diagrams*. Stanford Research Institute, 1981. 58

S. Iwamoto. Stochastic optimization of forward recursive functions. *Journal of Mathematical Analysis and Applications*, 292(1):73 – 83, 2004. 68

J.-Y. Jaffray. Implementing resolute choice under uncertainty. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence 1998*, pages 282–288. Morgan Kaufmann Publishers Inc., 1998. 72

G. Jeantet, P. Perny, and O. Spanjaard. Sequential Decision Making with Rank Dependent Utility: a Minimax Regret Approach. In *Proc. of AAAI 2012*, pages 1931–1937, 2012. 74

S. Koenig and R.G. Simmons. How to make reactive planners risk-sensitive. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, volume 293298, 1994. 68

H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. 51

J. Lang and J. Rothe. Fair division of indivisible goods. In *Economics and Computation*, pages 493–550. Springer, 2016. 50

E.L. Lawler and D.E. Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966. 55

Y. Liu and S. Koenig. Functional value iteration for decision-theoretic planning with general utility functions. In *Proceedings of AAAI 2006*, volume 21, page 1186, 2006. 68

Y. Liu and S. Koenig. An exact algorithm for solving MDPs under risk-sensitive planning objectives with one-switch utility functions. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '08, pages 453–460, 2008. 68

Y. Liu. Risk-sensitive planning with one-switch utility functions: Value iteration. In *In AAAI*, pages 993–999, 2005. 68

E.F. McClennen. *Rationality and dynamic choice: Foundational explorations*. Cambridge university press, 1990. 71, 72

B.L. Monroe. Fully proportional representation. *American Political Science Review*, 89(04):925–940, 1995. 52

T.D. Nielsen and J.Y. Jaffray. Dynamic decision making without expected utility: An operational approach. *European Journal of Operational Research*, 169(1):226–246, 2006. 72

C.H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1982. 48

R.F. Potthoff and S.J. Brams. Proportional representation: Broadening the options. *Journal of Theoretical Politics*, 10(2):147–178, 1998. 53

A.D. Procaccia, J.S. Rosenschein, and A. Zohar. On the complexity of achieving proportional representation. *Social Choice and Welfare*, 30(3):353–362, 2008. 51, 53, 55

M.L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014. 63, 66, 69, 70

H. Raiffa. Decision analysis: introductory lectures on choices under uncertainty. 1968. *MD computing: computers in medical practice*, 10(5):312, 1993. 58

P. Skowron, L. Yu, P. Faliszewski, and E. Elkind. The complexity of fully proportional representation for single-crossing electorates. In *International Symposium on Algorithmic Game Theory*, pages 1–12. Springer, 2013. 53, 55

O. Spanjaard and P. Weng. Markov decision processes with functional rewards. In *International Workshop on Multi-disciplinary Trends in Artificial Intelligence*, pages 269–280. Springer, 2013. 68

R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998. 64

V.V. Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013. 55

P. Weng and B. Zanuttini. Interactive value iteration for markov decision processes with unknown rewards. In Francesca Rossi, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*. IJCAI/AAAI, 2013. 71

D. J. White. Utility, probabilistic constraints, mean and variance of discounted rewards in Markov decision processes. *OR Spektrum*, 9:13–22, 1987. 68

Chapter 3

Oracle Methods

“ Le portrait d’un être qu’on aime doit pouvoir être non seulement une image à laquelle on sourit mais encore un oracle qu’on interroge. ”

A. Breton

Section Contents

Summary of the chapter

In this chapter, we present the *oracle methods* that are used in this thesis in two particular domains, game theory and fractional programming. In short, we call oracle method, a master-slave method where the slave algorithm is called “oracle” and is used to generate new constraints on the optimization problem that is solved by the master algorithm. Oracle methods are powerful computational methods which makes it possible to solve problems that are defined through a large number of constraints by dynamically generating the ones that are actually required to solve the problem (which we hope are much fewer).

In this chapter, we present the *oracle methods* that are used in this thesis. In short, we call oracle method, a master-slave method where the slave algorithm is called "oracle" and is used to generate new constraints on the optimization problem that is solved by the master algorithm. Oracle methods are powerful computational tools. The idea underlying oracle methods is that, even if a computational problem is defined through a large number of constraints, the number of constraints required to solve the problem is often much lower. Thus, even if it might be impossible to directly solve the problem, the resolution might be possible by dynamically generating the constraints that are required to solve the problem.

To present oracle methods, we investigate two particular domains, the theory of zero-sum two-player games in normal form and fractional programming. Indeed, many problems motivated in the previous chapter can be reformulated either as the search for a mixed Nash equilibrium in a zero-sum two-player game or as a fractional programming problem. As we will see in Chapters 4, 6 and 7, optimizing SSB utility functions, the minmax regret criterion and the OWA criterion can be seen as solving specific zero-sum two-player games. Moreover, as we will see in Chapters 4 and 7, optimizing the WEU model and the MO criterion can be done by using fractional programming methods.

Game theory is important both for decision theory and combinatorial optimization problems. Firstly, game theory extends decision theory by studying how a decision maker behaves when acting with other persons whose actions also affect the decision maker. Secondly, while the optimization problems studied in game theory can be of combinatorial nature, combinatorial optimization problems can often be reformulated in the setting of game theory. For this reason, theoretical results and algorithmic tools from game theory reveal precious allies to study and solve many combinatorial optimization problems.

Fractional programming is a subfield of optimization theory studying the optimization of ratios of two linear functions. Fractional programming has often been investigated in bi-objective optimization problems where the first (resp. second) objective must be maximized (resp. minimized). Optimizing the ratio of the first objective over the second one can then be a possible objective function to obtain a Pareto optimal solution. In this thesis, fractional programming will provide efficient algorithmic tools to solve combinatorial optimization problems with the WEU model or the MO criterion.

3.1 Zero-Sum Two-Player Games

In this section, we explain the bases of zero-sum two-player games. These games involve two players with opposed goals, meaning that one player's gains are balanced by the other player's losses and vice-versa. The actions performed by a player during the game form her *strategy* and solving the game consists in finding an optimal strategy for each player. In this thesis, our interest for these games results from the fact that many combinatorial optimization problems can be reformulated as the search for a mixed Nash equilibrium in a zero-sum two-player game. Indeed, as we will see in Chapters 4, 6 and 7, optimizing SSB utility functions, the minmax regret criterion and the OWA criterion can be seen as solving specific zero-sum two-player games.

Let me give some examples of these games.

Example 36. *In this section, we will use the well-known game "rock, paper, scissors" as running example. This game is a zero-sum two-player game as it involves two completely adversarial players. The sets of possible strategies of the two players are here the same:*

{*"play rock", "play paper", "play scissors"*}. *We will also use the notations r , p and s to denote these three strategies.*

The game "rock, paper, scissors" is a particular type of game as the two players have the same set of possible strategies. In this case, the game is said to be *symmetric*. Not all games are symmetric, as shown by this second example.

Example 37. *France and England, are at war. France's military force is composed of 4 armies whereas England can only rely on 3 armies. There are two fronts, one at Orléans and one at London, and it is not conceivable to let one undefended. France can make three choices: send 3 armies at Orléans and 1 at London, send 1 army at Orléan and 3 at London, or send 2 armies on each front. England only has two possible strategies: send 2 armies at Orléans and 1 at London or vice-versa.*

In case of numerical equality on a front, neither one of the two countries wins anything. Otherwise, the victory is earned by the country that has the most armies. Both fronts do not have the same importance: a victory (resp. defeat) at London gives a reward (resp. penalty) of 10 points whereas a victory (resp. defeat) at Orléans gives a reward (resp. penalty) of 6 points. Once again, this game is a zero-sum two-player game as it involves two completely adversarial players. However this game is clearly not symmetric.

More formally, a zero-sum two-player game is represented by a matrix M with n rows and m columns[1]. In this case, the first player (or player 1) has n strategies while the second player (or player 2) has m strategies. The value $\mathrm{M}[i,j]$, a real number, at the intersection of the i^{th} row and the j^{th} column is then called the "payoff" (which can be negative) that receives player 1 if she plays her i^{th} strategy while player 2 plays her j^{th} strategy. At the same time, player 2 receives a loss of $\mathrm{M}[i,j]$ or equivalently a payoff of $-\mathrm{M}[i,j]$. The sum of payoffs of the two players is therefore always equal to zero, which justifies the name of this type of two-player games.

In the game "rock, paper, scissors" (Example 36), the strategy "play rock" beats the strategy "play scissors", the strategy "play scissors" beats the strategy "play paper" and lastly, the strategy "play paper" beats the strategy "play rock". If one attributes a payoff of 1 to winning the game, then the matrix of the game is given by:

$$\mathrm{M}_{rps} = \begin{array}{c} \\ r \\ p \\ s \end{array} \begin{array}{c} \begin{array}{ccc} r & p & s \end{array} \\ \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} \end{array}$$

Note that matrix M_{rps} is of a particular type. Indeed, for all pair (i,j), $\mathrm{M}_{rps}[i,j]$ is equal to $-\mathrm{M}_{rps}[j,i]$. The matrix M_{rps} is said to be *antisymmetric*. In fact, this property applies to all matrices representing zero-sum symmetric games.

Proposition 6. *If a two-player zero-sum game is symmetric, then the matrix associated to this game is antisymmetric.*

Each player has at her disposal two types of strategies, *pure* strategies and *mixed* strategies. A mixed strategy consists in choosing a probability distribution over the available strategies. Then the strategy to play is sampled according to this probability distribution. Instead, a pure strategy consists in choosing a specific strategy to play. Therefore, pure strategies correspond to the mixed strategies that are associated with degenerate probability distributions. We will denote by Δ_1 (resp. Δ_2) the set of mixed strategies of player

[1] By abuse of language, we will indifferently talk of a game or of the matrix representing it.

1 (resp. player 2) and $S_1 \subseteq \Delta_1$ (resp. $S_2 \subseteq \Delta_2$) the set of pure strategies of player 1 (resp. player 2).

For the game "rock, paper, scissors", an example of pure strategy is to play "paper" and an example of mixed strategy is to play "paper" with probability 0.5 and to play "rock" with probability 0.5. A pure strategy is said to be in the *support* of a mixed strategy if it is played with a strictly positive probability according to the mixed strategy.

More formally, for player 1 (resp. player 2) a pure strategy corresponds to a row (resp. a column) of the matrix M whereas a mixed strategy corresponds to a probability distribution over rows (resp. columns) of matrix M. A strategy belonging to S_1 (resp. S_2) can then be denoted by the index $i \in \{1,\dots,n\}$ (resp. $j \in \{1,\dots,m\}$) of the corresponding row (resp. column). Differently, a strategy belonging to Δ_1 (resp. Δ_2) can be denoted either:

- by a lottery $\tilde{s} = (s_1, p_1; \dots; s_k, p_k)$ where for all i, s_i belongs to S_1 (resp. S_2) and p_i is the propability of playing pure strategy s_i;
- or by a positive vector $\mathbf{p}$ of size n (resp. m) with elements summing up to 1. The i^{th} element of $\mathbf{p}$ is the probability of playing the i^{th} pure strategy with the mixed strategy defined by vector $\mathbf{p}$.

As already noticed, a pure strategy can also be written as a mixed strategy by using the canonical vectors (i.e., vectors with one element equal to 1 all others being null).

For all pair $\mathbf{p}, \mathbf{q} \in \Delta_1 \times \Delta_2$ the expected payoff of player 1 is given by the matrix product ${}^t\mathbf{p}\mathrm{M}\mathbf{q}$ and the expected payoff of player 2 is given by $-{}^t\mathbf{p}\mathrm{M}\mathbf{q}$. Both players seek to maximize their payoff. Thus, while player 1 tries to maximize ${}^t\mathbf{p}\mathrm{M}\mathbf{q}$, player 2 tries to minimize this value. To do so, each player will try to protect herself against the strategies of the other player. To clarify this idea, we now define the concept of *best response*.

Definition 36. *Given a strategy $\mathbf{p} \in \Delta_1$ (resp. $\mathbf{q} \in \Delta_2$), a best response of player 2 (resp. player 1) in game* M *is defined as a strategy $\mathbf{q} \in \Delta_2$ (resp. $\mathbf{p} \in \Delta_1$) minimizing (resp. maximizing) the matrix product ${}^t\mathbf{p}\mathrm{M}\mathbf{q}$. This leads us to define the two sets $\mathrm{BR}_\mathrm{M}(\mathbf{p}) \subseteq \Delta_2$ and $\mathrm{BR}_\mathrm{M}(\mathbf{q}) \subseteq \Delta_1$ as follows:*

$$\mathrm{BR}_\mathrm{M}(\mathbf{p}) = \arg\min_{\mathbf{q}\in\Delta_2} {}^t\mathbf{p}\mathrm{M}\mathbf{q} \tag{3.1}$$

$$\mathrm{BR}_\mathrm{M}(\mathbf{q}) = \arg\max_{\mathbf{p}\in\Delta_1} {}^t\mathbf{p}\mathrm{M}\mathbf{q} \tag{3.2}$$

An important property of best responses is that a best response can always be found among pure strategies. This is proven by contradiction: if a best response is a mixed strategy, then each element of its support should be a best response. Otherwise, removing this pure strategy would improve the mixed strategy.

Proposition 7. *Given a strategy $\mathbf{p} \in \Delta_1$ (resp. $\mathbf{q} \in \Delta_2$), a best response for player 2 (resp. player 1) can always be found among pure strategies. Stated differently, $\mathrm{BR}_\mathrm{M}(\mathbf{p}) \cap S_2 \neq \emptyset$ and $\mathrm{BR}_\mathrm{M}(\mathbf{q}) \cap S_1 \neq \emptyset$.*

If one player is pessimistic, she will consider that the adversary facing her is going to play with a best response whatever the strategy that she chooses. Therefore, if player 1 and player 2 are both pessimistic, they will try to find strategies $\mathbf{p}^*$ and $\mathbf{q}^*$ such that:

$$\mathbf{p}^* \in \arg\max_{\mathbf{p}\in\Delta_1} \min_{\mathbf{q}\in\Delta_2} {}^t\mathbf{p}\mathrm{M}\mathbf{q} = \arg\max_{\mathbf{p}\in\Delta_1} \min_{\mathbf{q}\in S_2} {}^t\mathbf{p}\mathrm{M}\mathbf{q} \tag{3.3}$$

$$\mathbf{q}^* \in \arg\min_{\mathbf{q}\in\Delta_2} \max_{\mathbf{p}\in\Delta_1} {}^t\mathbf{p}\mathrm{M}\mathbf{q} = \arg\min_{\mathbf{q}\in\Delta_2} \max_{\mathbf{p}\in S_1} {}^t\mathbf{p}\mathrm{M}\mathbf{q} \tag{3.4}$$

Such a strategy $\mathbf{p}^*$ (resp. $\mathbf{q}^*$) is called a *maxmin* (resp. *minmax*) strategy. One of the important results of these section is a consequence from the minimax theorem by Von Neumann which states that the values guaranteed by strategies p^* and q^* are equal to one another:

Theorem 8. *In a finite zero-sum two-player game:*

$$v = \max_{\mathbf{p}\in\Delta_1} \min_{\mathbf{q}\in\Delta_2} {}^t\mathbf{p}\mathrm{M}\mathbf{q} = \min_{\mathbf{q}\in\Delta_2} \max_{\mathbf{p}\in\Delta_1} {}^t\mathbf{p}\mathrm{M}\mathbf{q} \tag{3.5}$$

The value v is called the value of the game and a pair of strategies $(\mathbf{p}^*, \mathbf{q}^*)$ such that $\mathbf{p}^*$ and $\mathbf{q}^*$ satisfy equations 3.3 and 3.4 simultaneously is called a *Nash equilibrium*. Note that the minimax theorem insures the existence of a Nash equilibrium in finite zero-sum two-player games. If players 1 and 2 play according to a Nash equilibrium, then neither one of them can improve her payoff by changing her strategy. Note that if the two-player zero-sum game is symmetric, then by symmetry, the value of the game v has to be 0.

For instance, in the game "rock, paper, scissors", a Nash equilibrium is given by the pair $(\mathbf{p}^*, \mathbf{p}^*)$, where $\mathbf{p}^*$ plays with probability 1/3 each pure strategy and the value of the game is $v = 0$.

The concept of Nash equilibrium is central in game theory and, in most of the literature, solving the game amounts to determining a Nash equilibrium. However note that this concept is not the only one worth investigating and that in some situations it can seem quite unoptimal, as illustrated by the prisoner dilemma for instance. However, in this thesis, we will only be interested in the concept of Nash equilibrium. We now present several methods to determine a Nash equilibrium in a zero-sum two-player game.

3.1.1 Linear Programming

A first method to solve zero-sum two-player games is to use one of the two following linear programs [Chvatal, 1983; Raghavan, 1994]:

$$\max_{v,p_1,\ldots,p_n} v \tag{3.6}$$

$$v \leq \sum_{i=1}^{n} p_i \mathrm{M}_{ij} \qquad \forall j \in \{1,\ldots,m\} \tag{3.7}$$

$$\sum_{i=1}^{n} p_i = 1$$

$$p_i \geq 0 \qquad \forall i \in \{1,\ldots,n\}$$

$$\min_{v,q_1,\ldots,q_m} v \tag{3.8}$$

$$v \geq \sum_{j=1}^{m} q_j \mathrm{M}_{ij} \qquad \forall i \in \{1,\ldots,n\} \tag{3.9}$$

$$\sum_{j=1}^{m} q_j = 1$$

$$q_j \geq 0 \qquad \forall j \in \{1,\ldots,m\}$$

Variables p_i (resp. q_j) describe all possible mixed strategies of player 1 (resp. player 2). Constraints 3.7 (resp. 3.9) ensure that whatever the mixed strategy described by variables p_i (resp. q_j), v will take the value of the best response of player 2 (resp. player 1). Consequently, at optimality, v takes the value of the game and the optimal strategy for player 1 (resp. player 2) is given by the values of the variables p_i (resp. q_j). Note that it is only necessary to solve one program to have both optimal strategies as programs 3.6 and 3.8 are duals. Thus, variables q_j (resp. p_i) are the dual variables associated to constraints 3.7 (resp. 3.9).

Several algorithms are known to solve linear programs. The simplex algorithm is quite efficient in practice but can require an exponential time (in the size of the instance) in the worst case. The ellipsoid algorithm and the interior point methods are often less efficient in practice but are known to be polynomial time methods [Grötschel *et al.*, 1981;

Khachiyan, 1980].

Unfortunately, if n and/or m are too large (e.g., if there is a combinatorial set of strategies), solving the game by directly using these linear programs may be impractical as the number of variables and constraints of the linear program will be too important. To address this problem, several oracle methods can be used which we detail now.

3.1.2 Fictitious Play

The first oracle method that we present, the most well-known one, is called Fictitious Play [Brown, 1951]. To solve a game, this method relies on two oracles $\mathscr{O}_1 : \Delta_2 \to \mathrm{S}_1$ and $\mathscr{O}_2 : \Delta_1 \to \mathrm{S}_2$ (one for each player) which given any strategy of the opposite player returns a pure best response to it.

The algorithm maintains for each player her mixed policy so far denoted by $\tilde{s}_1$ (resp. $\tilde{s}_2$) for player 1 (resp. player 2). These mixed policies are defined as $\tilde{s}_1 = (s_1^1, p_1; \ldots; s_1^k, p_k)$ and $\tilde{s}_2 = (s_2^1, q_1; \ldots; s_2^l, q_l)$, where for all i, s_1^i (resp. s_2^i) belongs to S_1 (resp. S_2) and p_i (resp. q_i) is the fraction of episodes so far in which the pure strategy s_1^i (resp. s_2^i) has been played by player 1 (resp. player 2). At each time step, player 1 (resp. player 2) considers that $\tilde{s}_2$ (resp. $\tilde{s}_1$) perfectly represents the mixed strategy that is used by the adversary and they simultaneously play a best response to it. The algorithm converges to a Nash equilibrium of the game for a zero-sum two-player game [Fudenberg and Levine, 1998; Robinson, 1951]. The pseudocode of fictitious play is shown in Algorithm 4. Note that if the game is symmetric, this procedure can be simplified and we can only consider a single mixed strategy $\tilde{s}$ of a player playing against herself.

Algorithm 4: Fictitious Play

Data: Game $\mathscr{G}$, arbitrary pure strategies s_1^0 and s_2^0
$\tilde{s}_1$ and $\tilde{s}_2$ are initially set to s_1^0 and s_2^0
Current iteration of the method $t = 0$
while *True* **do**
  Player 1 plays $s_1^t = \mathscr{O}_1(\tilde{s}_2)$ and Player 2 plays $s_2^t = \mathscr{O}_2(\tilde{s}_1)$
  # update current mixed strategy for player 1
  $\tilde{s}_1 = (s_1^1, p_1; \ldots; s_1^n, p_n)$ with
  $\begin{cases} p_i = t \cdot p_i/(t+1) + 1/(t+1) & \text{for } s_1^i = s_1^t \\ p_i = t \cdot p_i/(t+1) & \text{for } s_1^i \neq s_1^t \end{cases}$
  # update current mixed strategy for player 2
  $\tilde{s}_2 = (s_2^1, q_1; \ldots; s_2^m, q_m)$ with
  $\begin{cases} q_i = t \cdot q_i/(t+1) + 1/(t+1) & \text{for } s_2^i = s_2^t \\ q_i = t \cdot q_i/(t+1) & \text{for } s_2^i \neq s_2^t \end{cases}$
  $t \leftarrow t + 1$

Example 38. *We come back to the game "rock, paper, scissors" to show the beginning of one run of this method. Let us assume that the game starts with $s_1^0 = r$ and $s_2^0 = p$.*

Iteration 0: Player 1 plays r and player 2 plays p. The mixed strategies of the players are updated, $\tilde{s}_1 = (r, 1)$ and $\tilde{s}_2 = (p, 1)$.

Iteration 1: The two players use their oracles, $\mathscr{O}_1(\tilde{s}_2) = s$ and $\mathscr{O}_2(\tilde{s}_1) = p$. Player 1 plays s and player 2 plays p. The mixed strategies of the players are updated, $\tilde{s}_1 = (r, 0.5; s, 0.5)$ and $\tilde{s}_2 = (p, 1)$.

Iteration 2: The two players use their oracles, $\mathcal{O}_1(\tilde{s}_2) = s$ and $\mathcal{O}_2(\tilde{s}_1) = r$. Player 1 plays s and player 2 plays r. The mixed strategies of the players are updated, $\tilde{s}_1 = (r, 1/3; s, 2/3)$ and $\tilde{s}_2 = (p, 2/3; r, 1/3)$.

Iteration 3: The two players use their oracles, $\mathcal{O}_1(\tilde{s}_2) = p$ and $\mathcal{O}_2(\tilde{s}_1) = r$. Player 1 plays p and player 2 plays r. The mixed strategies of the players are updated, $\tilde{s}_1 = (r, 0.25; s, 0.5; p, 0.25)$ and $\tilde{s}_2 = (p, 0.5; r, 0.5)$.

And so on.

Fictitious play has many advantages. In particular, it can be used on huge games without requiring an explicit definition of the entire game. Moreover, it may only store and focus on a small portion of the game as only the strategies in the support of $\tilde{s}_1$ and $\tilde{s}_2$ need to be stored. However, fictitious play is known to converge very slowly. This empirical evidence is clarified with the following result for two-player constant-sum games (which obviously include two-player zero-sum games):

Theorem 9 (Brandt *et al.* [2010])**.** *In symmetric two-player constant-sum games, fictitious play may require exponentially many rounds (in the size of the representation of the game) before an equilibrium action is eventually played. This holds even for games solvable via iterated strict dominance.*

This negative results show the necessity for other oracle methods to solve large games.

3.1.3 The Double Oracle Algorithm

The double oracle algorithm applies to any zero-sum two-player game [McMahan *et al.*, 2003]. It has been used in many domains recently: by Chen *et al.* [2016] to solve inverse reinforcement learning problems; by Jain *et al.* [2011] to solve security games; lastly, it was adapted by Bosansky *et al.* [2013] to solve extensive form games. The double oracle algorithm will be used repeatedly in this thesis. The idea underlying this method is that for many games, while the number of pure strategies can be large, the number of relevant ones to solve the game is much lower. Thus if one can efficiently identify these strategies, one can solve the game using a restricted game with much fewer strategies.

Definition 37. *Consider a zero-sum two-player game defined by a payoff matrix* M *where players 1 and 2 have the set of pure strategies* S_1 *and* S_2*. Let* $\mathrm{S}'_1 \subseteq \mathrm{S}_1$ *and* $\mathrm{S}'_2 \subseteq \mathrm{S}_2$*, the game* M *restricted to* S'_1 *and* S'_2 *is the game associated to the submatrix of* M *containing the rows and columns corresponding to the pure strategies in* S'_1 *and* S'_2 *and where the available strategies of player 1 and player 2 are limited to* S'_1 *and* S'_2*. Thus, a mixed strategy of player 1 (resp. 2) can only assign positive probabilities to the strategies in* S'_1 *(resp.* S'_2*). We will denote by* $\mathrm{M}(\mathrm{S}'_1, \mathrm{S}'_2)$ *the game* M *restricted to* S'_1 *and* S'_2*.*

We illustrate the concept of restricted game on the "rock, paper, scissors" game.

Example 39. *Denote by r, p and s the three possible pure strategies corresponding to "play rock", "play paper" and "play scissors". Let* $\mathrm{S}'_1 = \{r, p\} \subset \mathrm{S}_1$ *and* $\mathrm{S}'_2 = \{s\} \subset \mathrm{S}_2$*, the matrix associated to the game* $\mathrm{M}(\mathrm{S}'_1, \mathrm{S}'_2)$ *restricted to* S'_1 *and* S'_2 *is given by:*

$$\begin{array}{c} \\ r \\ p \end{array} \begin{array}{c} s \\ \begin{pmatrix} 1 \\ -1 \end{pmatrix} \end{array}$$

The idea of the double oracle algorithm is to find a "small" restricted game such that solving this restricted game amounts to solving the entire game. The following proposition enables to identify such a restricted game.

Proposition 8. *Consider a zero-sum two-player game* M *and let* $\mathrm{M}(\mathrm{S}'_1, \mathrm{S}'_2)$ *denote the game* M *restricted to strategy sets* S'_1 *and* S'_2*. Lastly, let* $(\tilde{s}_1^*, \tilde{s}_2^*)$ *be a Nash equilibrium of* $\mathrm{M}(\mathrm{S}'_1, \mathrm{S}'_2)$*. If* $\mathrm{S}'_1 \cap \mathrm{BR}_{\mathrm{M}}(\tilde{s}_2^*) \neq \emptyset$ *and* $\mathrm{S}'_2 \cap \mathrm{BR}_{\mathrm{M}}(\tilde{s}_1^*) \neq \emptyset$*, then* $(\tilde{s}_1^*, \tilde{s}_2^*)$ *is a Nash equilibrium of* M*.*

Looking for the smallest possible game verifying Proposition 8 could enable to solve the game M much more efficiently than solving M directly. The double oracle algorithm intents to incrementally build such a restricted game by using two oracles $\mathscr{O}_1 : \Delta_2 \to \mathrm{S}_1$ and $\mathscr{O}_2 : \Delta_1 \to \mathrm{S}_2$ (one for each player) which given any strategy of the opposite player returns a pure best response to it.

The algorithm starts with small sets S'_1 and S'_2 of pure strategies (singletons in Algorithm 5), and then grows these set in every iteration by applying the best-response oracles $\mathscr{O}_1$ and $\mathscr{O}_2$ to the optimal strategies (given by an NE) the players can play in the restricted game $\mathrm{M}(\mathrm{S}'_1, \mathrm{S}'_2)$. An NE in a zero-sum two-player game can be computed by linear programming as detailed in subsection 3.1.1. Execution continues until convergence is detected. Convergence is achieved when the oracles both generate pure strategies that are already contained in S'_1 and S'_2 (see Proposition 8). The double oracle approach is implemented by the procedure described in Algorithm 5.

Algorithm 5: Double Oracle Algorithm

Data: Finite zero-sum two-player game M, singletons $\mathrm{S}'_1 = \{s_1\}$ and $\mathrm{S}'_2 = \{s_2\}$ including arbitrary pure strategies $s_1 \in \mathrm{S}_1$ and $s_2 \in \mathrm{S}_2$
Result: a nash equilibrium $(\tilde{s}_1^*, \tilde{s}_2^*)$ of the game M

```
converge = False
while converge is False do
    Find a Nash equilibrium (s̃₁*, s̃₂*) of the game M(S'₁, S'₂)
    Determine s₁ = O₁(s̃₂) and s₂ = O₂(s̃₁)
    if s₁ ∉ S'₁ or s₂ ∉ S'₂ then
        add s₁ to S'₁ and/or s₂ to S'₂
    else
        converge =True
return (s̃₁*, s̃₂*)
```

The correctness of double oracle algorithms for two-player zero-sum games has been established by McMahan *et al.* [2003]; this correctness directly follows from Proposition 8. The intuition behind this correctness is the following. Once best responses for both players are already in the restricted game, the current solution of the restricted game must be an equilibrium of the game, because each player's current strategy is a best response to the other player's current strategy. This stems from the fact that the oracles, which search over all possible strategies, cannot find anything better.

Furthermore, the algorithm must converge, because at worst, it will generate all pure strategies. In practice, we hope that the restricted game will stay relatively small and that many pure strategies will never enter the restricted strategy sets S'_1 and S'_2.

Example 40. *We come back to the game "rock, paper, scissors" to show one run of this method. We start with* $\mathrm{S}'_1 = \{r, p\} \subset \mathrm{S}_1$ *and* $\mathrm{S}'_2 = \{s\} \subset \mathrm{S}_2$*.*

Iteration 0: A Nash equilibrium of $\mathrm{M}(\mathrm{S}'_1, \mathrm{S}'_2)$ *is* (r, s)*. The two players use their oracles,* $\mathcal{O}_1(s) = r$ *and* $\mathcal{O}_2(r) = p$*. The strategy* r *is already in* S'_1*. However,* p *is not in* S'_2 *and is added to this set.*

Iteration 1: A Nash equilibrium of $\mathrm{M}(\mathrm{S}'_1, \mathrm{S}'_2)$ *is now* $(\tilde{s}_1, \tilde{s}_2)$ *with* $\tilde{s}_1 = (r, 1/3; p, 2/3)$ *and* $\tilde{s}_2 = (s, 1/3; p, 2/3)$*. The two players use their oracles,* $\mathcal{O}_1(\tilde{s}_2) = s$ *and* $\mathcal{O}_2(\tilde{s}_1) = s$*. The strategy* s *is added to* S_1*.*

Iteration 2: A Nash equilibrium of $\mathrm{M}(\mathrm{S}'_1, \mathrm{S}'_2)$ *is now* $(\tilde{s}_1, \tilde{s}_2)$ *with* $\tilde{s}_1 = (r, 1/3; p, 0/3; s, 2/3)$ *and* $\tilde{s}_2 = (s, 2/3; p, 1/3)$*. The two players use their oracles and* $\mathcal{O}_2(\tilde{s}_1) = r$ *is added to* S'_2*.*

Iteration 3: A Nash equilibrium of the game is now $(\tilde{s}^*, \tilde{s}^*)$ *with* $\tilde{s}^* = (r, 1/3; p, 1/3; s, 1/3)$*. Both oracles now generate strategies that are already in* S'_1 *and* S'_2 *and the algorithm terminates.*

This example illustrates that unfortunately, in the worst case, the algorithm might generate all strategies of the game M (note however that the game is here very small). It would be interesting to have positive results under some conditions on the number of iterations that are required by the double oracle algorithm to converge. Zinkevich *et al.* [2007] designed an algorithm inspired from the double oracle algorithm whose number of iterations until convergence is bounded by a theoretical measure that they named *range of skill*. Unfortunately, Hansen *et al.* [2008] provided negative results on the range of skill measure. For instance, they showed that the range of skill of a perfectly balanced game tree is almost exponential in its size and that the range of skill of Tic-Tac-Toe is more than 100000.

3.1.4 Cutting Plane Methods

The last oracle method that we present in this section is related to cutting plane methods. These methods make it possible to solve linear programs that involve a large number of constraints.

More formally, consider the following *OPTIMIZATION* problem: $\max\{{}^{\mathrm{t}}cx | x \in \mathrm{X} \subset \mathbb{R}^n\}$ where X is the set of feasible solutions. The associated *SEPARATION* problem reads as follows: given $\hat{x} \in \mathbb{R}^n$, is $\hat{x} \in \mathrm{CH(X)}$? (where CH(X) denotes the convex hull of X)and if not, give a *certificate*, i.e., an inequality ${}^{\mathrm{t}}ax \leq b$ satisfied by all points in X but violated by $\hat{x}$ (i.e., ${}^{\mathrm{t}}a\hat{x} > b$).

In cutting plane methods (e.g., Benders decomposition [Benders, 2005]), the oracle $\mathcal{O}$ used should solve the *SEPARATION* problem and is therefore called a *separation oracle*. The idea of these methods is that even if the set S of constraints describing X is huge, these constraints can be handled efficiently by dynamically generating them. More precisely, the cutting plane methods start by solving the *OPTIMIZATION* problem by only trying to enforce a small subset $\mathrm{S}' \subseteq \mathrm{S}$ of constraints. Given the solution found $\hat{x}$, the oracle $\mathcal{O}$ indicates a constraint in $\mathrm{S} \setminus \mathrm{S}'$ that is not satisfied by $\hat{x}$ and adds this constraint to S', or states that there is no violated constraints in which case the current solution is optimal. In practice, the oracle method tries to find the most violated constraint although finding one that is violated is enough. The algorithm continues this process until $\mathcal{O}$ states that there is no violated constraints.

For zero-sum two-player games, cutting plane methods can be used to solve linear program 3.6 (resp. 3.8) in the case where the number of strategies for player 2 (resp. player 1) is too large to solve the entire game directly. In this case, the strategies of player 2 (resp. player 1) would be dynamically generated. Given a Nash equilibrium $(\tilde{s}_1, \tilde{s}_2)$ in the restricted game, the oracle $\mathcal{O}$ would try to find a best response to $\tilde{s}_1$ (resp. $\tilde{s}_2$) and see if the corresponding constraint in 3.7 (resp. 3.9) is satisfied.

We illustrate how cutting plane methods work in Example 41.

Example 41. *We come back to the game "rock, paper, scissors". The linear program solving this game as described in subsection 3.1.1 is:*

$$\max_{v,p_1,p_2,p_3} v \tag{3.10}$$
$$v \leq p_2 - p_3 \qquad /* \text{ constrained associated to "player 2 plays rock"} */ \tag{3.11}$$
$$v \leq -p_1 + p_3 \qquad /* \text{ constrained associated to "player 2 plays paper"} */ \tag{3.12}$$
$$v \leq p_1 - p_2 \qquad /* \text{ constrained associated to "player 2 plays scissors"} */ \tag{3.13}$$
$$1 = p_1 + p_2 + p_3 \tag{3.14}$$
$$p_i \geq 0 \qquad \forall i \in \{1,\dots,3\}$$

We denote by S *the set of constraints of Program 3.10. This program can be solved by using a cutting plane approach. In this approach, at each iteration, we solve Program 3.10, while only considering a subset* S′ *of constraints. Then, given the solution* (v, p_1, p_2, p_3) *that is determined, an oracle* $\mathcal{O}$ *checks if one constraint of* S *is violated. Checking if* $p_1 + p_2 + p_3 = 1$ *is easy. If* $p_1 + p_2 + p_3 \neq 1$*, constraint 3.14 is added to* S′*. Otherwise,* $\mathcal{O}$ *checks if one of constraints 3.11, 3.12 or 3.13 is violated. This can be done by computing the best response to the current strategy of player 1 given by variable* p_1*,* p_2 *and* p_3*. Indeed, this will compute the most violated constraint if any. If one of constraints 3.11, 3.12 or 3.13 is violated, then it is added to* S′*. Otherwise, the method terminates and states that the strategy defined by variables* p_1*,* p_2 *and* p_3 *is an optimal strategy for player 1. Let us see one run of this method.*

Iteration 0: We start with $S' = \{v \leq p_2 - p_3, 1 = p_1 + p_2 + p_3\}$*. We solve the linear program 3.15. The optimal value of this program is* $v = 1$ *and is obtained for* $p_1 = 0$*,* $p_2 = 1$ *and* $p_3 = 0$ *which is the strategy "play paper". We assume that we have an oracle* $\mathcal{O}$ *that can compute a best response to this strategy. It obviously returns strategy "play scissors". The corresponding constraint is violated as* $v = 1 > p_1 - p_2 = -1$*. Thus constraint* $v \leq p_1 - p_2$ *is added to* S′*.*

$$\max_{v,p_1,p_2,p_3} v \tag{3.15}$$
$$v \leq p_2 - p_3$$
$$1 = p_1 + p_2 + p_3$$
$$p_i \geq 0 \qquad \forall i \in \{1,\dots,3\}$$

Iteration 1: We solve the linear program 3.16. The optimal value of this program is $v = 1/3$ *and is obtained for* $p_1 = 2/3$*,* $p_2 = 1/3$ *and* $p_3 = 0$ *which is the strategy "play rock"* $2/3$ *of the time and "play paper"* $1/3$ *of the time. The oracle* $\mathcal{O}$ *returns strategy "play paper" as a best response. The corresponding constraint is violated as* $v = 1/3 > -p_1 + p_3 = -2/3$*. Thus constraint* $v \leq -p_1 + p_3$ *is added to* S′*.*

$$\max_{v,p_1,p_2,p_3} v \tag{3.16}$$
$$v \leq p_2 - p_3$$
$$v \leq p_1 - p_2$$
$$1 = p_1 + p_2 + p_3$$
$$p_i \geq 0 \qquad \forall i \in \{1,\dots,3\}$$

Iteration 2: We solve the linear program 3.17. The optimal value of this program is $v = 0$ *and is obtained for* $p_1 = 1/3$, $p_2 = 1/3$ *and* $p_3 = 1/3$ *which is the strategy "play rock"* $1/3$ *of the time, "play paper"* $1/3$ *of the time and "play scissors"* $1/3$ *of the time. The oracle* $\mathcal{O}$ *returns strategy "play paper" as a best response. The corresponding constraint is not violated as* $v = 0 = -p_1 + p_3 = 0$. *Thus, the program terminates and states that an optimal strategy for player 1 is given by* $p_1 = p_2 = p_3 = 1/3$.

$$\begin{aligned} \max_{v,p_1,p_2,p_3} \quad & v \\ & v \leq p_2 - p_3 \\ & v \leq -p_1 + p_3 \\ & v \leq p_1 - p_2 \\ & 1 = p_1 + p_2 + p_3 \\ & p_i \geq 0 \qquad \forall i \in \{1,\dots,3\} \end{aligned} \tag{3.17}$$

In this example, all constraints are needed to find the optimal solution, but this (luckily) is not always true.

Formulating a problem in a way that makes it solvable via a cutting plane method can lead to theoretical results on the complexity of the problem studied due to the following important result.

Theorem 10 (Grötschel *et al.* [1981])**.** *There is a polynomial time equivalence between the* SEPARATION *and the* OPTIMIZATION *problems. Stated differently, if one can optimize over* X *in polynomial time, then one can separate over* CH(X) *in polynomial time, and if one can separate over* CH(X) *in polynomial time, then one can optimize over* X *in polynomial time.*

This theorem gives us a motivation to try to formulate our optimization problems as problems solvable by cutting plane methods. Indeed, in this case, studying the complexity of the separation oracle method may enable us to find the complexity of the optimization problem.

In this section, we have briefly presented the theory of zero-sum two-player games and have presented several oracle methods from this literature. These oracle methods will reveal useful in this thesis as several of the problems that we will encounter can be analyzed in a game-theoretic setting.

In Chapter 4, we will show that optimizing SSB utility functions in sequential decision problems under risk can be seen as solving a specific zero-sum two-player symmetric game. This analysis will make it possible to develop a linear programming approach, a double oracle approach as well as a cutting plane approach.
In Chapter 6, we will see that finding a lower bound on the optimal minmax regret value in robust combinatorial optimization problems with interval data reduces to solving a specific zero-sum two-player game. This analysis will make it possible to develop a double oracle approach that we will integrate in a branch and bound to solve efficiently robust combinatorial optimization problems with interval data.
In Chapter 7, we will see that finding an optimal mixed solution in multi-agent problems with an OWA criterion can be seen as solving a specific zero-sum two-player. This analysis will make it possible to develop a double oracle approach as well as a cutting plane approach.

In the next section, we will briefly present the theory of fractional programming and detail two oracle methods used to solve fractional programming problems.

3.2 Fractional Programming

Fractional programming is a subfield of optimization theory studying the optimization of ratios of two linear functions. More precisely, in fractional programming the optimization problem studied is stated as follows:

$$\max_{x_1,\ldots,x_n} \frac{a_0 + a_1 x_1 + \ldots + a_n x_n}{b_0 + b_1 x_1 + \ldots + b_n x_n} \tag{3.18}$$

$$\text{s.t. } \mathbf{x} = (x_1, \ldots, x_n) \in \mathrm{D} \tag{3.19}$$

where D is a finite set of possible solutions that is defined by some mathematical constraints and where we assume that the denominator $b_0 + b_1 x_1 + \ldots + b_n x_n$ is always strictly positive. Note that, if one uses an artificial term x_0 equal to 1, then the numerator (resp. denominator) can be written as ${}^t\mathbf{ax}$ (resp. ${}^t\mathbf{bx}$) where $\mathbf{a} = (a_0, \ldots, a_n)$ (resp. $\mathbf{b} = (b_0, \ldots, b_n)$). One example of a practical application of fractional programming is that of maximizing cost-to-time ratio. For instance, Dantzig *et al.* [1966] and Lawler [1966] investigated the problem of finding a minimum cost-to-time ratio cycle in a graph. This applies to finding optimal ship routing. Another example of applications is the assignment problem if one tries to maximize the value received per unit of time expended on the assignments [Kabadi and Punnen, 2008; Shigeno *et al.*, 1995].

A large literature has been developed on fractional programming [Nagih and Plateau, 1999; Schaible and Ibaraki, 1983; Stancu-Minasian, 2012], leading to the development of several solution methods and optimization techniques. In this section, we will present two oracle methods, namely Megiddo's and Dinkelbach's methods, that were developed in the setting of fractional programming. Note that several other solution methods exist, e.g., Charnes-Cooper's method to solve fractional programs by linear programs [Horst and Pardalos, 2013].

In both of these methods, we assume that there exists an algorithm denoted by $\mathscr{O}$ that solves the *standard* problem:

$$\max_{x_1,\ldots,x_n} c_0 + c_1 x_1 + \ldots + c_n x_n \tag{3.20}$$

$$\text{s.t. } \mathbf{x} = (x_1, \ldots, x_n) \in \mathrm{D} \tag{3.21}$$

The algorithm $\mathscr{O}$ will play the role of the oracle in these methods.

3.2.1 Megiddo's Method

The first oracle method that we present was developed by Megiddo [1979]. This method is appealing due to the following result:

Theorem 11 (Megiddo [1979])**.** *If the oracle $\mathscr{O}$ solves the standard problem with* $\mathrm{O}(p(n))$ *comparisons and* $\mathrm{O}(q(n))$ *additions then the associated fractional problem is solvable in time* $\mathrm{O}(p(n)(q(n) + p(n))$ *via Megiddo's method.*

This method is based on the following observation:

Observation 1. *Let $\lambda \in \mathbb{R}$ and consider the vector $\mathbf{c}^\lambda$ defined by $c_i^\lambda = a_i - \lambda b_i$ for all $i \in \{0,\dots,n\}$. Let $\mathbf{x}^\lambda$ and $v^\lambda = {}^t\mathbf{c}^\lambda\mathbf{x}^\lambda$ be an optimal solution and the optimal value for the standard problem with objective function ${}^t\mathbf{c}^\lambda\mathbf{x}$. Then, the sign of v^λ is determined by the way λ and λ^* compare to one another, where λ^* is defined as the optimal value of the fractional problem (i.e., $\lambda^* = \max_{(x_1,\dots,x_n)\in D} \frac{a_0+a_1x_1+\dots+a_nx_n}{b_0+b_1x_1+\dots+b_nx_n}$).*

- *1. If $\lambda = \lambda^*$, then $v^\lambda = 0$ and $\mathbf{x}^\lambda$ is an optimal solution of the fractional problem.*
- *2. If $\lambda > \lambda^*$, then v^λ will be strictly negative. Indeed, there exists no feasible solution $\mathbf{x} \in D$ such that $\frac{a_0+a_1x_1+\dots+a_nx_n}{b_0+b_1x_1+\dots+b_nx_n} \geq \lambda$.*
- *3. On the contrary, if $\lambda < \lambda^*$, then v^λ will be strictly positive.*

Thus, the problem of solving the fractional problem reduces to solving the standard problem with objective function ${}^t\mathbf{c}^{\lambda^*}\mathbf{x}$ (case 1 of Observation 1), where we recall that $c_i^{\lambda^*} = a_i - \lambda^* b_i$. However, while values a_i and b_i are known, the value of λ^* is not. Thus, the values $c_i^{\lambda^*}$ are incompletely specified. In short, Megiddo's method mimics algorithm $\mathcal{O}$ to solve the standard problem (Equation 3.20) with objective function ${}^t\mathbf{c}^{\lambda^*}\mathbf{x}$. However, the method redefines the addition and the comparison operations to handle the fact that λ^* is unknown.

Management of the imprecisely known value of λ^*.
Instead of having precise values for $c_i^{\lambda^*}$, the algorithm works with pairs of values (a_i, b_i) and maintains a lower bound λ^l and an upper bound λ^u over λ^* (originally $-\infty$ and $+\infty$). Thus $c_i^{\lambda^*}$ is only known to be in the interval $[l(c_i^{\lambda^*}), u(c_i^{\lambda^*})]$ where $l(c_i^{\lambda^*}) = \min\{a_i - \lambda^l b_i, a_i - \lambda^u b_i\}$ and $u(c_i^{\lambda^*}) = \max\{a_i - \lambda^l b_i, a_i - \lambda^u b_i\}$.

Redefinition of the addition operation.
The additions and subtractions required by algorithm $\mathcal{O}$ are simply replaced by componentwise additions and subtractions of pairs. Stated differently, the sum of (a_i, b_i) and (a_k, b_k) is $(a_i + a_k, b_i + b_k)$. Indeed, whatever the value of λ^*:

$$a_i - \lambda^* b_i + a_k - \lambda^* b_k = a_i + a_k - \lambda^*(b_i + b_k).$$

Redefinition of the comparison operation.
To compare two pairs (a_i, b_i) and (a_k, b_k), Megiddo's method uses the following routine.

A fist step consists in checking if $a_i - \lambda b_i$ is less (resp. greater) than $a_k - \lambda b_k$, regardless of the value of $\lambda \in [\lambda^l, \lambda^u]$. Indeed, in that case (illustrated on the left side of Figure 3.1 below), the algorithm can conclude that $a_i - \lambda^* b_i \leq$ (resp. $\geq$) $a_k - \lambda^* b_k$. Note that $a_i - \lambda b_i$ and $a_k - \lambda b_k$ are linear functions of λ. Therefore, the method needs only to check the inequality for values λ^l and λ^u:

$$\begin{aligned} a_i - \lambda b_i \leq a_k - \lambda b_k, \forall \lambda \in \{\lambda^l, \lambda^u\} &\Rightarrow a_i - \lambda b_i \leq a_k - \lambda b_k, \forall \lambda \in [\lambda^l, \lambda^u] \\ &\Rightarrow a_i - \lambda^* b_i \leq a_k - \lambda^* b_k \end{aligned}$$

If this first step does not conclude which pair is the minimum (case illustrated on the right side of Figure 3.1), then the algorithm considers the value $\hat{\lambda}$ such that $a_i - \hat{\lambda} b_i = a_k - \hat{\lambda} b_k$. Then, if we assume w.l.o.g.[2] that $a_i - \lambda^l b_i < a_k - \lambda^l b_k$, we have:

$$\forall \lambda \in [\lambda^l, \hat{\lambda}], a_i - \lambda b_i \leq a_k - \lambda b_k \tag{3.22}$$

$$\forall \lambda \in [\hat{\lambda}, \lambda^u], a_i - \lambda b_i \geq a_k - \lambda b_k. \tag{3.23}$$

[2] if $a_i - \lambda^l b_i > a_k - \lambda^l b_k$, just reverse inequalities 3.22 and 3.23.

Now, for the algorithm to conclude, it needs only to check if $\lambda^* \in [\lambda^l, \hat{\lambda}]$ (in which case, λ^u is set to $\hat{\lambda}$) or $\lambda^* \in [\hat{\lambda}, \lambda^u]$ (in which case, λ^l is set to $\hat{\lambda}$). This is done by testing the sign of the optimal value of the standard problem with objective function ${}^{\mathrm{t}}\mathbf{c}^{\hat{\lambda}}\mathbf{x}$ (see Observation 1), which is computed by the oracle $\mathscr{O}$. Note that this operation updates either λ^l or λ^u, which precises the value of λ^* and enables us to perform more comparisons without rerunning the oracle $\mathscr{O}$.

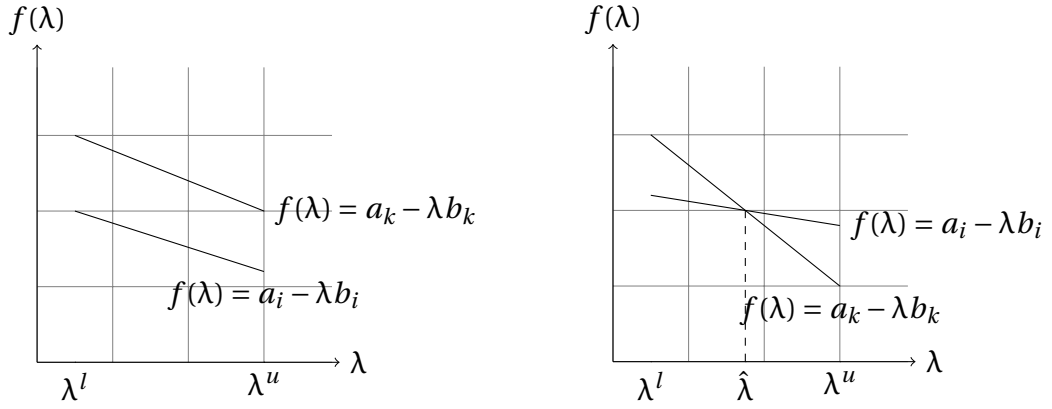


Figure 3.1: Illustration of the two possible cases that can occur in the comparison routine.

Polynomial time complexity of the method.
We assume that algorithm $\mathscr{O}$ relies on a polynomial number of additions and comparisons. As Megiddo's method mimics algorithm $\mathscr{O}$, it relies on a polynomial number of redefined additions and comparisons. As algorithm $\mathscr{O}$ (which is used in the comparison operator) is of polynomial complexity, these two operations can be performed in polynomial time. This proves the polynomial complexity of the method.

Additions and comparisons are the two types of operations that Megiddo [1979] adapted to his method. However, note that other operations may be adapted similarly. We will use two other operations in this thesis.

Multiplication by a scalar.
The scalar multiplication operations that may be required by algorithm $\mathscr{O}$ are simply replaced by multiplying both components of the pair by the scalar value. Stated differently, the pair (a, b) multiplied by a value α is the pair $(\alpha \times a, \alpha \times b)$. Indeed, whatever the value of λ^*:

$$\alpha \times (a - \lambda^* b) = \alpha \times a - \lambda^* (\alpha \times b).$$

We will require the *multiplication by a scalar* operation for solving decision trees according to the weighted expected utility model with Megiddo's method (in Chapter 4).

Identification of a zero.
To identify if a pair (a, b) represents a zero (i.e., if $a - \lambda^* b = 0$) we proceed as follows. First, if b is equal to 0, then the pair represents a 0 iff a equals 0. If b is different than 0, then (a, b) represents a zero iff $\lambda^* = a/b$. Stated differently,

$$a - \lambda^* b = 0 \Leftrightarrow a = 0 \quad \text{if } b = 0$$
$$a - \lambda^* b = 0 \Leftrightarrow \lambda^* = \frac{a}{b} \quad \text{otherwise.}$$

If $a/b \in [\lambda_l, \lambda_u]$ (otherwise we already know that $\lambda^* \neq a/b$), then testing if $\lambda^* = a/b$ is done by testing if the optimal value of the standard problem with objective function ${}^{\mathrm{t}}\mathbf{c}^{\lambda}\mathbf{x}$ is equal to 0 with $\lambda = a/b$ (see case 1 of Observation 1).

We will require the *identification of a zero* operation for solving assignment problems according to a mixture operator with Megiddo's method (in Chapter 7).

We illustrate Megiddo's method in the following example.

Example 42. *We consider the following directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$:*

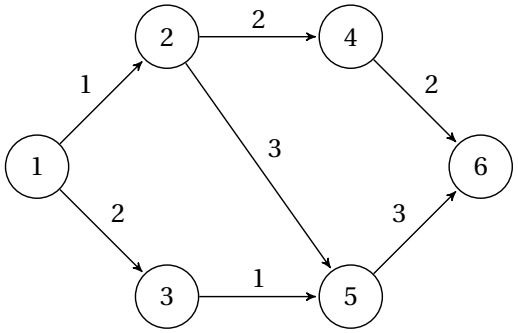


Figure 3.2: Directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each edge is valued by its utility.

where the value u_{ij} assigned to each edge $(i, j) \in \mathcal{E}$ is the utility of taking edge (i, j). We wish to find the path from vertex 1 to vertex 6 with highest utility (where the utility of a path is defined by the sum of the utilities of the edges that constitute it). This is an instance of the longest path problem. This problem can be formally written as a mathematical program in the following way. For each edge $(i, j) \in \mathcal{E}$, we consider a binary variable $x_{ij} \in \{0, 1\}$ that indicates if edge (i, j) is in the path from vertex 1 to vertex 6 that we are considering. By using variables x_{ij}, the longest path problem can be rewritten as:

$$\begin{aligned} \max_{x_{ij}:(i,j)\in\mathcal{E}} & \sum_{x_{ij}:(i,j)\in\mathcal{E}} x_{ij} u_{ij} \\ b_j = & -\sum_{(i,j)\in\mathcal{E}} x_{ij} + \sum_{(j,k)\in\mathcal{E}} x_{jk} \\ x_{ij} \in & \{0, 1\} \end{aligned} \tag{3.24}$$

where variable $x_{ij} = 1$ if edge (i, j) is in the path considered and $b_j = 1$ if $j = 1$, $b_j = -1$ if $j = 6$, and $b_j = 0$ otherwise. Constraints 3.24 are the flow conservation constraints that ensure that variables x_{ij} describe a valid path from vertex 1 to vertex 6.

Longest path problems can be solved in linear time for directed acyclic graphs by using the following procedure:

1. *Initialize the distances to all vertices as $-\infty$ and initialize the distance to the source as 0. Initialize the predecessors of all vertices as -1. Then, find a topological sorting of the graph.*

2. *Once we have a topological order of the graph, process all vertices, one by one, in topological order. For every vertex being processed, update the distances of its adjacent vertices and their predecessors by using the distance of the current vertex.*

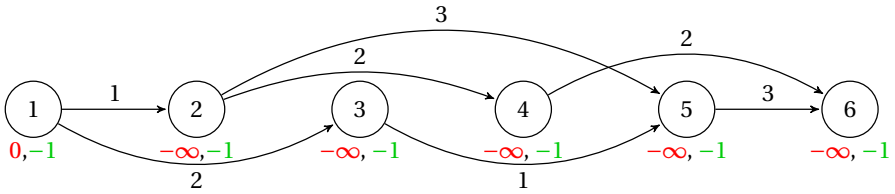


Figure 3.3: A topological order of graph 3.2.

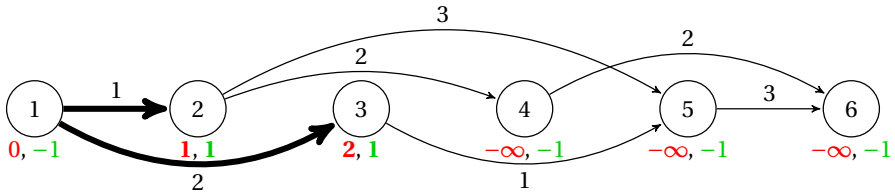


Figure 3.4: Vertex 1 is processed. The distance to vertex 2 is updated to 1 as $1 = 0 + 1 > -\infty$. The distance to vertex 3 is updated to 2 as $2 = 0 + 2 > -\infty$. The predecessors of vertices 2 and 3 are updated to 1.

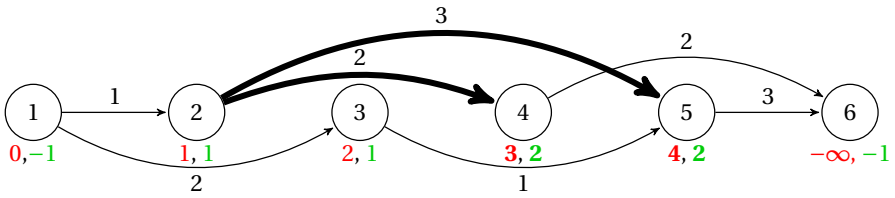


Figure 3.5: Vertex 2 is processed. The distance to vertex 4 is updated to 3 as $3 = 1 + 2 > -\infty$. The distance to vertex 5 is updated to 4 as $4 = 1 + 3 > -\infty$. The predecessors of vertices 4 and 5 are updated to 2.

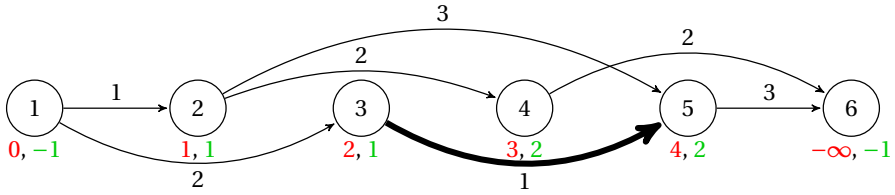


Figure 3.6: Vertex 3 is processed. The distance to node 5 is not updated as $3 = 2 + 1 < 4$. The predecessor of node 5 is not updated.

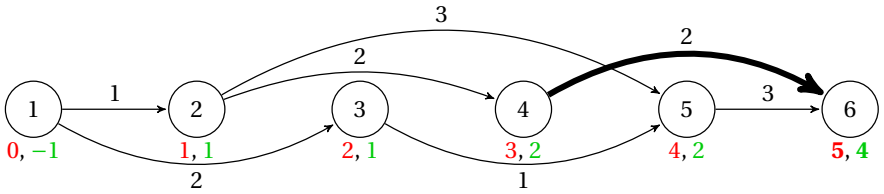


Figure 3.7: Vertex 4 is processed. The distance to vertex 6 is updated to 5 as $5 = 3 + 2 > -\infty$. The predecessor of vertex 6 is updated to 4.

Figures 3.3 to 3.8 show, step by step, the process of finding a longest path for the graph represented in Figure 3.2. The distances to each vertex from vertex 0 are given in red. The predecessor of each vertex is given in green.

We find that a path of maximum length from vertex 1 to vertex 6 is $(1,2,5,6)$.

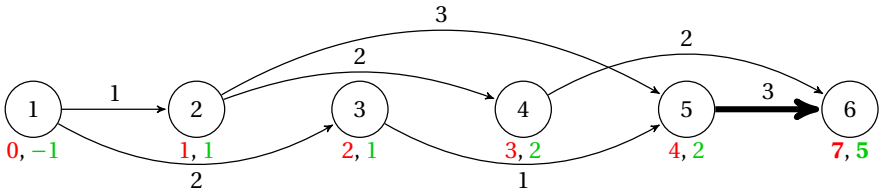


Figure 3.8: Vertex 5 is processed. The distance to vertex 6 is updated to 7 as $7 = 4+3 > 5$. The predecessor of vertex 6 is updated to 5.

We now add to each edge the time that is required to travel this edge. This gives us a new graph (represented in Figure 3.9) that we will also denote by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

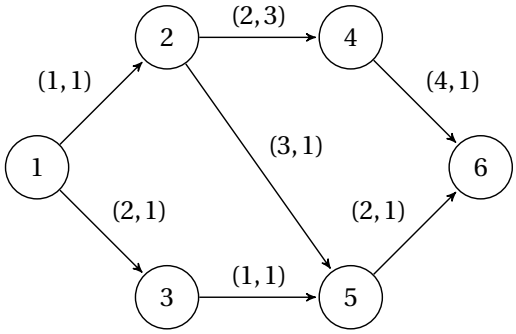


Figure 3.9: Directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each edge is valued by its utility and the time required to travel it.

For each edge $(i,j) \in \mathcal{E}$*, a pair of values* (u_{ij}, t_{ij}) *is given such that* u_{ij} *is the utility of taking edge* (i,j) *and* t_{ij} *is the time required to take edge* (i,j)[3].

We now wish to find the path from vertex 1 to vertex 6 with highest utility per time where the utility of a path is defined as the sum of the utilities of the edges that constitute it and the time required to travel a path is defined as the sum of the times required by each of its edges. This is an instance of the fractional programming problem where the standard problem is the longest path problem. Indeed, by using variables x_{ij}*, this problem can be rewritten as the following mathematical program:*

$$\max_{x_{ij}:(i,j)\in\mathcal{E}} \frac{\sum_{x_{ij}:(i,j)\in\mathcal{E}} x_{ij}u_{ij}}{\sum_{x_{ij}:(i,j)\in\mathcal{E}} x_{ij}t_{ij}}$$

$$b_j = -\sum_{(i,j)\in\mathcal{E}} x_{ij} + \sum_{(j,k)\in\mathcal{E}} x_{jk}$$

$$x_{ij} \in \{0,1\}$$

where $b_j = 1$ *if* $j = 1$, $b_j = -1$ *if* $j = 6$, *and* $b_j = 0$ *otherwise. Thus, in this case, the oracle* $\mathcal{O}$ *solves a longest path problem in linear time (as the graph is directed and acyclic) by using topological sorting as illustrated by Figures 3.3 to 3.8. We now illustrate how Megiddo's method would solve the fractional programming problem in Figures 3.10 to 3.15. Instead of initializing the distance to each vertex (resp. vertex 1) to* $-\infty$ *(resp. 0), we initialize these values with the pair* $(-\infty, 0)$ *(resp.* $(0,0)$*). The value* λ^l *can be initialized to 0 as all values of the problem are positive. The value* λ^u *is initialized to* $+\infty$.

In Figure 3.11, the distance to vertex 2 is updated to $(1,1)$ *as* $(0+1)-\lambda\times(0+1) > -\infty-\lambda\times 0$ *whatever the value of* $\lambda \in [\lambda^l, \lambda^u]$ *(i.e.,* $[0, +\infty]$*). The distance to vertex 3 is updated to* $(2,1)$

[3]Note that some utility values have changed compared to Figure 3.2.

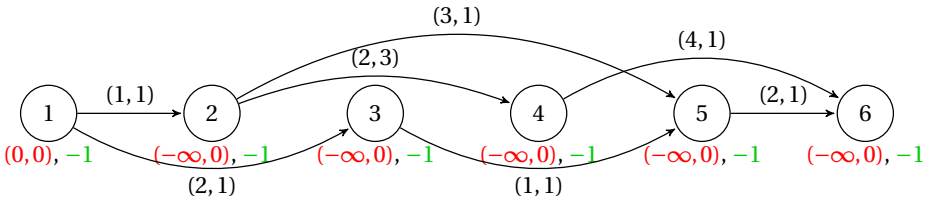


Figure 3.10: A topological order of graph 3.9.

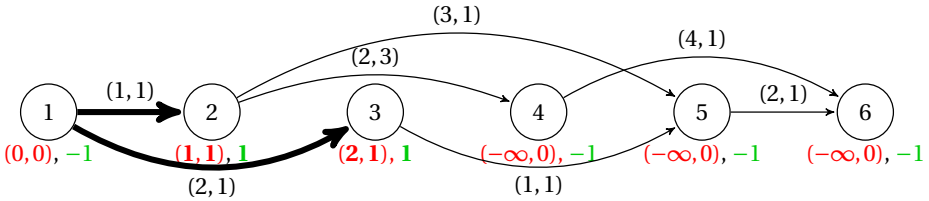


Figure 3.11: Vertex 1 is processed.

as $(0+2)-\lambda\times(0+1) > -\infty-\lambda\times 0$ *whatever the value of* $\lambda \in [\lambda^l, \lambda^u]$ *(i.e.,* $[0,+\infty]$*). The predecessors of vertices 2 and 3 are updated to 1.*

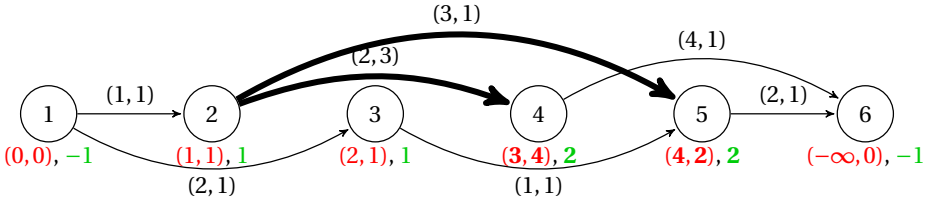


Figure 3.12: Vertex 2 is processed.

In Figure 3.12, the distance to vertex 4 is updated to $(3,4)$ *as* $(1+2)-\lambda\times(1+3) > -\infty-\lambda\times 0$ *whatever the value of* $\lambda \in [\lambda^l, \lambda^u]$ *(i.e.,* $[0,+\infty]$*). The distance to vertex 5 is updated to* $(4,2)$ *as* $(1+3)-\lambda\times(1+1) > -\infty-\lambda\times 0$ *whatever the value of* $\lambda \in [\lambda^l, \lambda^u]$ *(i.e.,* $[0,+\infty]$*). The predecessors of vertices 4 and 5 are updated to 2.*

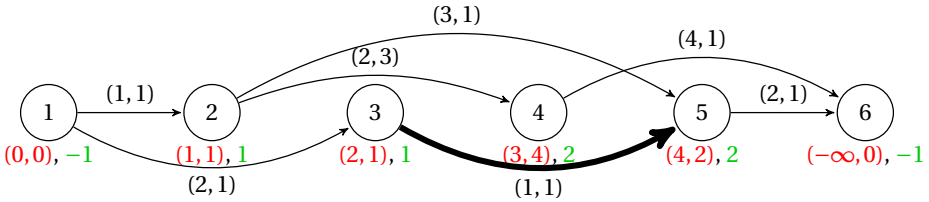


Figure 3.13: Vertex 3 is processed.

In Figure 3.13, the distance to vertex 5 is not updated as $(2+1)-\lambda\times(1+1) < 4-\lambda\times 2$ *whatever the value of* $\lambda \in [\lambda^l, \lambda^u]$ *(i.e.,* $[0,+\infty]$*). The predecessor of vertex 5 is not updated.*

In Figure 3.14, the distance to vertex 6 is updated to $(7,5)$ *as* $(3+4)-\lambda\times(4+1) > -\infty-\lambda\times 0$ *whatever the value of* $\lambda \in [\lambda^l, \lambda^u]$ *(i.e.,* $[0,+\infty]$*). The predecessor of vertex 6 is updated to 4.*

In Figure 3.15, vertex 5 is processed. In this step, we need to get the maximum of the two pairs $(7,5)$ *and* $(4+2, 2+1) = (6,3)$*. We first try to see if* $7-5\lambda \geq 6-3\lambda$, $\forall \lambda \in [\lambda^l, \lambda^u]$ *or if* $7-5\lambda \leq 6-3\lambda$, $\forall \lambda \in [\lambda^l, \lambda^u]$*. We have:*

$$7-5\lambda^l-(6-3\lambda^l) = 1-2\times 0 > 0 \Rightarrow 7-5\lambda^l > 6-3\lambda^l$$
$$7-5\lambda^u-(6-3\lambda^u) = 1-2\times\infty < 0 \Rightarrow 7-5\lambda^u < 6-3\lambda^u$$

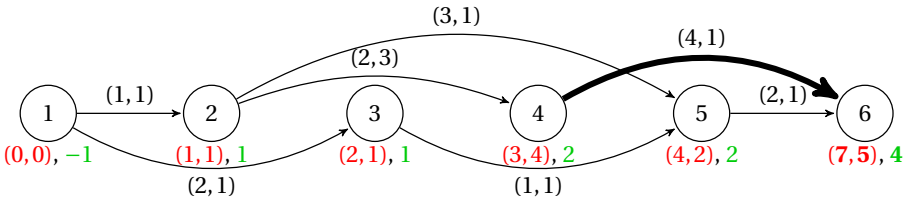


Figure 3.14: Vertex 4 is processed.

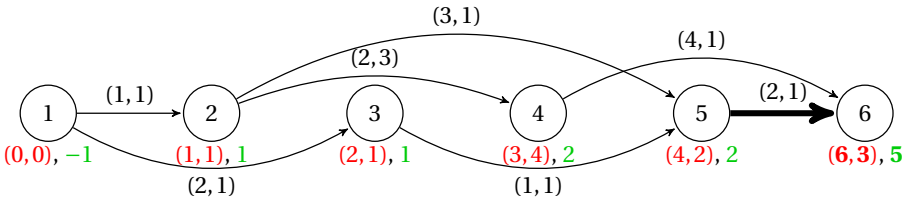


Figure 3.15: Vertex 5 is processed.

Therefore, we cannot directly conclude which pair should be kept. Thus, we compute $\hat{\lambda} = (7-6)/(5-3) = 0.5$ *such that* $7-5\hat{\lambda} = 6-3\hat{\lambda}$ *and we solve the standard problem where the utility of taking edge* (i,j) *is defined by* $u_{ij} - \hat{\lambda} t_{ij}$*. Stated differently, we try to find the value of the longest path from node 1 to node 6 in the graph represented in Figure 3.16.*

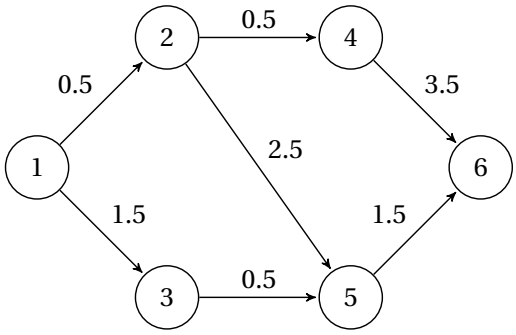


Figure 3.16: Directed acyclic graph with utilities defined by $u_{ij} - \hat{\lambda} t_{ij}$.

We omit the details but it can be easily checked that a longest path in this graph is $(1,2,5,6)$ *with value* $4.5 > 0$*. By Observation 1 (case 3), we conclude that* $\lambda^* > \hat{\lambda}$*. Thus* λ^l *is updated to* $\hat{\lambda} = 0.5$*. Now:*

$$7-5\lambda^l-(6-3\lambda^l) = 1-2\times 0.5 = 0 \Rightarrow 7-5\lambda^l = 6-3\lambda^l$$
$$7-5\lambda^u-(6-3\lambda^u) = 1-2\times\infty < 0 \Rightarrow 7-5\lambda^u < 6-3\lambda^u$$
$$\textit{Therefore, } 7-5\lambda \le 6-3\lambda, \quad \forall \lambda \in [\lambda^l,\lambda^u] \Rightarrow 7-5\lambda^* \le 6-3\lambda^*$$

Hence, the distance to vertex 6 is updated to $(6,3)$ *and its predecessor is updated to 5. Hence, path* $(1,2,5,6)$ *is a longest path for all utility functions defined by* $u_{ij} - \lambda t_{ij}$ *with* $\lambda \in [\lambda^l,\lambda^u]$ *including* $\lambda = \lambda^*$*. Thus, by Observation 1 (case 1), we can conclude that a path that maximizes the utility per time is* $(1,2,5,6)$ *with a value of* $\lambda^* = 6/3 = 2$*.*

We now detail in the next subsection another oracle method from the fractional programming literature, namely Dinkelbach's method.

3.2.2 Dinkelbach's Method

The second oracle method that we present is called Newton's method [Radzik, 1992] or Dinkelbach's Method [Dinkelbach, 1967; Ródenas *et al.*, 1999]. Interestingly, this method is related to the cutting plane approaches presented in subsection 3.1.4. We introduce this method in a way that highlights this point.

For a solution $\mathbf{x}^*$ maximizing the fractional problem 3.18, we have

$$\frac{{}^t\mathbf{a}\mathbf{x}^*}{{}^t\mathbf{b}\mathbf{x}^*} \geq \frac{{}^t\mathbf{a}\mathbf{x}}{{}^t\mathbf{b}\mathbf{x}}, \quad \forall \mathbf{x} \in \mathrm{D}.$$

Replacing ${}^t\mathbf{a}\mathbf{x}^*$ / ${}^t\mathbf{b}\mathbf{x}^*$ by λ^*, these inequalities can be rewritten as follows:

$${}^t\mathbf{a}\mathbf{x} - \lambda^* {}^t\mathbf{b}\mathbf{x} \leq 0, \quad \forall \mathbf{x} \in \mathrm{D} \tag{3.25}$$

and the constraint is tight for $\mathbf{x} = \mathbf{x}^*$. Therefore, λ^* is the minimal value such that all inequalities in 3.25 are satisfied. This analysis yields the following Linear Program (LP) $\mathrm{LP}^{\mathrm{FP}}$ (FP for fractional programming):

$$\mathrm{LP}^{\mathrm{FP}} \begin{cases} \min\limits_{\lambda} \lambda \\ {}^t\mathbf{a}\mathbf{x} - \lambda^* {}^t\mathbf{b}\mathbf{x} \leq 0, \qquad \forall \mathbf{x} \in \mathrm{D} \\ \lambda \in \mathbb{R} \end{cases} \tag{3.26}$$

Even if the number of constraints in 3.26 may be huge (as there is one constraint per feasible solution), these constraints can be handled efficiently by resorting to a cutting plane algorithm (see subsection 3.1.4). The separation oracle is here played by the algorithm $\mathscr{O}$. Given the current optimal value $\tilde{\lambda}$ of the linear program (solved with a subset of constraints), $\mathscr{O}$ solves the standard problem with objective function ${}^t\mathbf{c}^{\tilde{\lambda}}\mathbf{x}$. If the optimal value $v^{\tilde{\lambda}}$ is less than or equal to 0 (cases 1 and 2 of Observation 1), $\mathscr{O}$ says that no constraints is violated. Otherwise (case 3 of Observation 1), it returns the constraint associated to the optimal solution $\mathbf{x}^{\tilde{\lambda}}$ found. If the complexity of $\mathscr{O}$ is polynomial, the complexity of solving $\mathrm{LP}^{\mathrm{FP}}$ is polynomial by the polynomial time equivalence of *OPTIMIZATION* and *SEPARATION* (see Theorem 10 on page 89).

In practice, one can resort to Dinkelbach's method [Dinkelbach, 1967] for solving program $\mathrm{LP}^{\mathrm{FP}}$. Indeed, Dinkelbach's method reveals very efficient in practice and is also a polynomial method if the complexity of $\mathscr{O}$ is polynomial [Radzik, 1992]. We recall that $\mathbf{x}^{\lambda}$ denotes an optimal solution of the standard problem with objective function ${}^t\mathbf{c}^{\lambda}\mathbf{x}$. Program $\mathrm{LP}^{\mathrm{FP}}$ can be solved by computing a sequence of solutions in D through the following recursive equation:

$$\mathbf{x}^{t+1} = \mathbf{x}^{\lambda_t} \tag{3.27}$$

where:

$$\lambda_t = \frac{{}^t\mathbf{a}\mathbf{x}^t}{{}^t\mathbf{b}\mathbf{x}^t} \tag{3.28}$$

The key point for the validity of this approach is that the solutions generated in this way are of increasing values w.r.t. the fractional problem. Indeed, a direct corollary from Observation 1 is that while $\lambda_t < \lambda^*$, $\lambda_{t+1} > \lambda_t$. Therefore, the sequence $(\lambda_t)_{t\in\mathbb{N}}$ is strictly increasing until reaching λ^*. Value λ^* is always reached after a finite number of iterations as there is a finite number of solutions in D. In fact, Radzik [1992] showed that the method converges afer a polynomial number of iterations. Thus, after a polynomial number of iterations, we will have $\lambda_t = \lambda_{t+1}$ which means that an optimal solution of the fractional problem has been found.

We illustrate Dinkelbach's method in example 43.

Example 43 (Example 42 continued)**.** *We consider the same problem as in Example 42. Stated differently, we wish to find the path from vertex 1 to vertex 6 that maximizes utility per time in the graph represented in Figure 3.17 where each edge* $(i,j) \in \mathscr{E}$ *of the graph is valued by a pair* (u_{ij}, t_{ij}) *such that* u_{ij} *is the utility of taking edge* (i,j) *and* t_{ij} *is the time required to travel by edge* (i,j)*. This problem has been solved in Example 42 by using Megiddo's method. Let us now solve this problem by using Dinkelbach's method.*

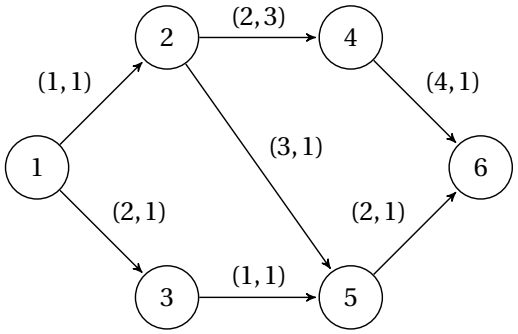


Figure 3.17: Directed acyclic graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$. Each edge is valued by its utility and the time required to travel it.

Iteration 0: We start with $\lambda_0 = 0$ *and we compute a solution* $\mathbf{x}^1$ *which is a longest path in the graph represented in Figure 3.17 where the utilities of each edge* (i,j) *is defined by* $u_{ij} - \lambda_0 t_{ij} = u_{ij}$*. Stated differently, we seek a longest path in the graph represented in Figure 3.18. This problem can be solved as illustrated in Example 42 by using a topological sorting of the graph. We obtain a longest path* $(1,2,4,6)$ *with a length of 7. The utility of this path is worth* $(1+2+4) = 7$ *and the time required to travel this path is worth* $(1+3+1) = 5$*.*

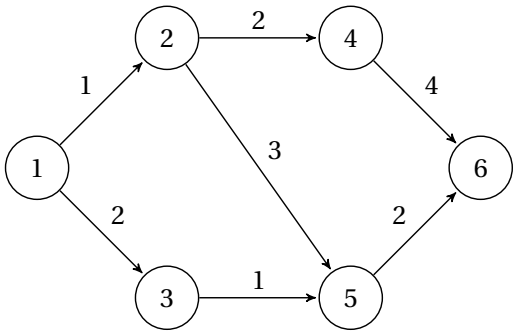


Figure 3.18: Directed acyclic graph with utilities defined by $u_{ij} - \lambda_0 t_{ij}$.

Iteration 1: We set λ_1 *to* $7/5$*. We now compute a solution* $\mathbf{x}^2$ *which is a longest path in graph 3.17 where the utilities of each edge* (i,j) *is given by* $u_{ij} - \lambda_1 t_{ij} = u_{ij} - (7/5)t_{ij}$*. Stated differently, we seek a longest path in the graph represented in Figure 3.19. We obtain a longest path* $(1,2,5,6)$ *with a length of* $9/5$*. The utility of this path is worth* $(1+3+2) = 6$ *and the time required to travel this path is worth* $(1+1+1) = 3$*.*

Iteration 2: We set λ_2 *to* $6/3 = 2$*. We now compute a solution* $\mathbf{x}^3$ *which is a longest path in graph 3.17 where the utilities of each edge* (i,j) *is given by* $u_{ij} - \lambda_2 t_{ij} = u_{ij} - 2t_{ij}$*. Stated differently, we seek a longest path in the graph represented in Figure 3.20. We obtain a longest path* $(1,2,5,6)$ *with a length of* 0*. The utility of this path is worth* $(1+3+2) = 6$ *and the time required to travel this path is worth* $(1+1+1) = 3$*.*

Iteration 3: We set λ_3 *to* $6/3 = 2$*. As* $\lambda_3 = \lambda_2$ *the method terminates and states that a path from vertex 1 to vertex 6 maximizing the utility per time is given by* $(1,2,5,6)$ *with an optimal value of* $\lambda^* = 2$*.*

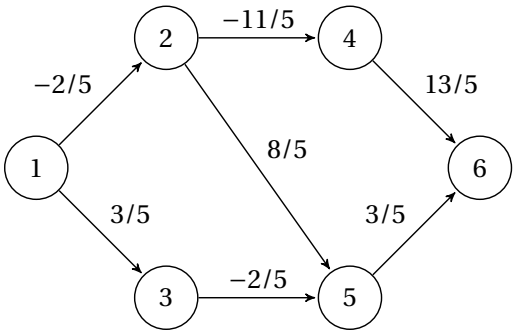


Figure 3.19: Directed acyclic graph with utilities defined by $u_{ij} - \lambda_1 t_{ij}$.

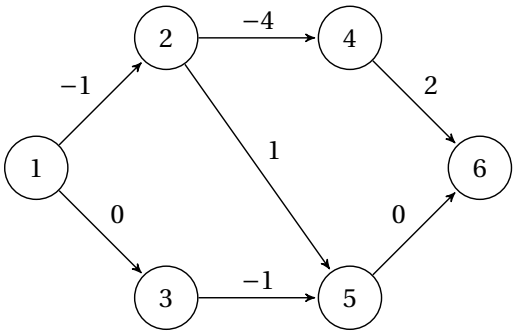


Figure 3.20: Directed acyclic graph with utilities defined by $u_{ij} - \lambda_2 t_{ij}$.

We have now seen and described the two main fractional programming methods that we will use in this thesis. These methods will often be useful when studying decision problems with the WEU model or an MO criterion.

In Chapter 4, we will see that optimizing the WEU model in sequential decision problems under risk can be performed with fractional programming methods. Similarly, in Chapter 7, we will see that optimizing an MO criterion in some multi-agent problems as the assignment problem or the proportional representation problem, can be performed with fractional programming methods.

3.3 Conclusion

We have presented in this chapter the oracle methods that are used in this thesis. For sake of illustration and to present these methods in the domains in which they where originally developed, we presented them in the setting of game theory and fractional programming. These tools will reveal very efficient to solve the different problems that will arise in the following chapters and that constitute the contributions of this thesis. The techniques used in the thesis are summarized in Table 3.1.

Decision model	Optimization problem	Algorithmic methods
SSB utility model	Sequential decision problems under risk	Linear programming, Double oracle approach, cutting plane approach
WEU model	Sequential decision problems under risk	Megiddo's method, Dinkelbach's method
Minmax regret criterion	Robust combinatorial optimization problems with interval data	Double oracle approach
OWA criterion	Randomized version of multi-agent decision problems	Double oracle approach, cutting plane approach
MO criterion	Some multi-agent decision problems: allocation problems, proportional representation problems	Megiddo's method, Dinkelbach's method

Table 3.1: Summary table answering the question: "Which methods are used for which problems?"

3.4 References

J.F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Computational Management Science, re-publication of: Numerische Mathematik 4: 238-252, 1962*, 2(1):3–19, 2005. 87

B. Bosansky, C. Kiekintveld, V. Lisy, J. Cermak, and M. Pechoucek. Double-oracle algorithm for computing an exact nash equilibrium in zero-sum extensive-form games. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 335–342. International Foundation for Autonomous Agents and Multiagent Systems, 2013. 85

F. Brandt, F. Fischer, and P. Harrenstein. On the rate of convergence of fictitious play. In *Proceedings of the Third International Conference on Algorithmic Game Theory*, SAGT'10, pages 102–113, Berlin, Heidelberg, 2010. Springer-Verlag. 85

G. W. Brown. Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, 1951. 84

X. Chen, M. Monfort, B.D. Ziebart, and P. Carr. Adversarial inverse optimal control for general imitation learning losses and embodiment transfer. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 2016. 85

V. Chvatal. *Linear programming.* Macmillan, 1983. 83

G.B. Dantzig, W.O. Blattner, and M.R. Rao. Finding a cycle in a graph with minimum cost to time ratio with application to a ship routing problem. Technical report, STANFORD UNIV CA OPERATIONS RESEARCH HOUSE, 1966. 90

W. Dinkelbach. On nonlinear fractional programming. *Management science*, 13(7):492–498, 1967. 98

D. Fudenberg and D.K. Levine. *The theory of learning in games*, volume 2. MIT press, 1998. 84

M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981. 83, 89

T.D. Hansen, P.B. Miltersen, and T.B. Sørensen. On range of skill. In *AAAI*, pages 277–282, 2008. 87

R. Horst and P.M. Pardalos. *Handbook of global optimization*, volume 2. Springer Science & Business Media, 2013. 90

M. Jain, D. Korzhyk, O. Vaněk, V. Conitzer, M. Pěchouček, and M. Tambe. A double oracle algorithm for zero-sum security games on graphs. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 327–334. International Foundation for Autonomous Agents and Multiagent Systems, 2011. 85

S.N. Kabadi and A.P. Punnen. A strongly polynomial simplex method for the linear fractional assignment problem. *Operations Research Letters*, 36(4):402–407, 2008. 90

L.G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980. 84

E.L. Lawler. Optimal cycles in doubly weighted directed linear graphs. In *Proceedings of the International Symposium on the Theory of Graphs*, pages 209–232, 1966. 90

H.B. McMahan, G.J. Gordon, and A. Blum. Planning in the presence of cost functions controlled by an adversary. In *International Conference on Machine Learning*, pages 536–543, 2003. 85, 86

N. Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4(4):414–424, 1979. 90, 92

A. Nagih and G. Plateau. Problèmes fractionnaires: tour d'horizon sur les applications et méthodes de résolution. *RAIRO-Operations Research*, 33(4):383–419, 1999. 90

T. Radzik. Newton's method for fractional combinatorial optimization. In *Foundations of Computer Science, 1992. Proceedings., 33rd Annual Symposium on*, pages 659–669. IEEE, 1992. 98

T.E.S Raghavan. Zero-sum two-person games. *Handbook of game theory*, 2:735–768, 1994. 83

J. Robinson. An iterative method of solving a game. *Annals of mathematics*, pages 296–301, 1951. 84

R.G. Ródenas, M.L. López, and D. Verastegui. Extensions of dinkelbach's algorithm for solving non-linear fractional programming problems. *Top*, 7(1):33–70, 1999. 98

S. Schaible and T. Ibaraki. Fractional programming. *European Journal of Operational Research*, 12(4):325–338, 1983. 90

M. Shigeno, Y. Saruwatari, and T. Matsui. An algorithm for fractional assignment problems. *Discrete Applied Mathematics*, 56(2-3):333–343, 1995. 90

I.M. Stancu-Minasian. *Fractional programming: theory, methods and applications*, volume 409. Springer Science & Business Media, 2012. 90

M. Zinkevich, M.H. Bowling, and N. Burch. A new algorithm for generating equilibria in massive zero-sum games. In *AAAI*, pages 788–794. AAAI Press, 2007. 87

Chapter 4

Sequential Decision Problems Under Risk with Skew-Symmetric Bilinear Utilities

" While the SSB theory is necessarily a better descriptive theory than the von Neumann-Morgenstern theory since it is a generalization of the latter, it has obvious shortcomings as a theory of actual behavior. On the other hand, I feel that it is an appealing normative theory of reasonable (rational) behavior under risk "

P.C. Fishburn

Section Contents

Summary of the chapter

In this chapter, we investigate the resolution of sequential decision problems under risk with the SSB and WEU models (see subsections 1.5.3 and 1.5.4) in decision trees and in finite horizon MDPs (see Section 2.2). We prove that, while finding an SSB optimal randomized strategy in decision trees is a problem of polynomial complexity, finding a deterministic strategy which is SSB optimal within the set of deterministic strategies is an NP-hard problem. Interestingly, with the WEU model, both problems become of polynomial complexity. Then, we extend these results to finite horizon MDPs. In finite horizon MDPs, we prove that determining an SSB optimal randomized policy and a WEU optimal deterministic policy can be performed in pseudo-polynomial time. In each framework, we formulate the resolution problem either as the search for a Nash equilibrium in a specific two-player zero-sum game or as a fractional programming optimization problem. This enables us to resort to the oracle methods introduced in Chapter 3. Lastly, we provide computational results showing the operationality of our approaches.

This chapter is based on several publications [Gilbert and Spanjaard, 2017; Gilbert *et al.*, 2015].

4.1 Introduction

In a sequential decision problem under risk, one aims to determine a preferred plan (i.e., a functions that maps possible decision situations to choices or probability distributions over choices) according to the decision model of the decision maker (also called agent). More precisely, both in decision trees and in MDPs, each plan (strategy/policy) induces a lottery over the possible outcomes of the sequential decision problem and the goal of the agent is to find a plan that yields an optimal lottery with respect to her decision model.

We recall that the SSB and WEU models have appealing descriptive abilities: the SSB utility model [Fishburn, 1982, 1984] is able to account for intransitive and non-independent preferences (i.e., preferences that do not satisfy the independence axiom), and the WEU model [Chew, 1983; Fishburn, 1983] (a subclass of the SSB model) can represent non-independent preferences. However, being able to account for such types of preferences has an algorithmic cost and the Bellman optimality principle does not hold for these two models. This makes the optimization of sequential decision problems under risk with the SSB and WEU models tricky. Nevertheless, for each representation and for both decision models, we will try to answer the following questions:

1. What are the properties of the SSB/WEU optimal plans? For instance in finite horizon MDPs, can we always find a deterministic Markovian policy which is at least as good as all other policies (when considering all types of policies)?
2. What are the computational complexities of the induced optimization problems?
3. What efficient algorithms can we use to find an SSB/WEU optimal plan?

Let us answer some preliminary questions to precise the setting of this chapter.

What types of behaviors are studied (resolute/sophisticated decision makers)?

We have seen that different types of behaviors could be defined in sequential decision problems under risk (see subsection 2.2.3): sophisticated, resolute choice with root dictatorship, resolute choice with selves. While these behaviors coincide when the Bellman optimality principle holds as with the traditional decision criteria (e.g., EU), they can differ

when this principle is violated. As the SSB/WEU model can represent non-independent preferences, the preferences induced with the SSB/WEU model can violate the Bellman optimality principle. Hence, we need to precise the type of behavior we are considering.

In this chapter, we study the dictatorship of the root approach. While studying the sophisticated approach would be quite straightforward from an algorithmic point of view (as this would lead to a simple backward induction method), this type of behavior can lead to non-optimal (and even stochastically dominated) plans at the root. The resolute approach with multiple selves is an interesting but more complicated paradigm that we wish to investigate in future works.

What is the decision criterion used to define an SSB optimal plan? As the SSB model can lead to preference cycles, defining an SSB optimal plan is not obvious. In fact, the SSB model induces a tournament over solutions and several decision criteria can be used to define an SSB optimal plan, i.e., a winner of the tournament. In this thesis, we use the minmax rule [Young, 1977], also known as Condorcet's rule or the Simpson-Kramer method, where each lottery l is evaluated by the highest intensity with which another lottery is preferred to l, and one selects a lottery with minimal evaluation. More formally, a lottery l is evaluated by $\max_{l' \in \mathscr{L}} \varphi(l', l)$ (to be minimized) where $\mathscr{L}$ is the set of lotteries induced by the possible plans (of all types) and φ is the SSB utility function.

Note that the minmax rule induces another preference relation than the one represented by φ. We use two different vocabularies to dissociate them from each other:

- A lottery l_1 is *preferred* (resp. *strictly preferred*) to a lottery l_2 if $\varphi(l_1, l_2) \geq$ (resp $>$) 0. A lottery l is *preferred to all other lotteries* if $\forall\, l' \in \mathscr{L}, \varphi(l, l') \geq 0$.

- Differently, we will say that a lottery l_1 is *at least as good as* (resp. *better than*) a lottery l_2 if $\max_{l' \in \mathscr{L}} \varphi(l', l_1) \leq$ (resp. $<$) $\max_{l' \in \mathscr{L}} \varphi(l', l_2)$. A lottery is *SSB optimal* if it belongs to $\arg\min_{l \in \mathscr{L}} \max_{l' \in \mathscr{L}} \varphi(l', l)$.

We wish to find an SSB optimal plan (i.e., a plan that induces an SSB optimal lottery). Interestingly, we will see that an SSB optimal plan is also preferred to all other plans.

What types of feasible plans are investigated? With the SSB model, we will see that an optimal deterministic plan may not exist and that optimal plans are in general randomized or mixed. Stated differently, there are sequential decision problems for which the set $\arg\min_{l \in \mathscr{L}} \max_{l' \in \mathscr{L}} \varphi(l', l)$ contains no lottery associated to a deterministic strategy. Our intuition is that a decision maker may refuse to use a randomized plan. For this reason, we distinguish between two settings according to the nature of the plans considered (deterministic/randomized). In the *deterministic setting*, we seek a deterministic plan which is SSB optimal within the set of deterministic plans. In the *randomized setting*, we seek an SSB optimal plan among *all* plans, including randomized and mixed ones.

With the WEU model, we will see that there always exists an optimal plan among the deterministic ones. Hence, it is sufficient to focus on deterministic plans when optimizing a weighted expected utility.

Summary of the results. In decision trees, we prove that an SSB (resp. WEU) optimal plan can always be found as a mixed or randomized (resp. deterministic) strategy. Similarly, in finite horizon MDPs, we prove that an SSB (resp. WEU) optimal plan can always be found as a mixed or randomized (resp. deterministic) wealth-Markovian policy (to be defined later in this chapter).

Regarding the complexity of the different optimization problems, we prove that while finding an SSB optimal randomized strategy in a decision tree is a problem of polynomial complexity, finding a deterministic strategy which is SSB optimal within the set of deterministic strategies is an NP-hard problem. Interestingly, with the WEU model, both problems become of polynomial complexity. To the best of our knowledge, WEU is the first decision model compatible with Allais' paradox and for which an optimal plan can be determined in polynomial time in the setting of decision trees. In finite horizon MDPs, we design pseudo-polynomial methods to determine an SSB optimal mixed policy and a WEU optimal deterministic policy.

Regarding the solution methods in themselves, we show that optimizing an SSB utility function in a sequential decision problem under risk can be seen as solving a specific zero-sum two-player symmetric game. This analysis will make it possible to design a linear programming approach, a double oracle approach as well as a cutting plane approach. Besides, we show that optimizing the WEU model in sequential decision problems under risk can be performed with fractional programming methods.

Table 4.1 summarizes our contributions.

		Decision Trees		Finite horizon MDPs	
	setting	SSB	WEU	SSB	WEU
Complexity	*deterministic*	NP-hard	P	NP-hard	pseudo-polynomial
	randomized	P	P	pseudo-polynomial	pseudo-polynomial
Type of optimal plan sought		randomized or mixed	deterministic	randomized or mixed wealth-Markovian	deterministic wealth-Markovian

Table 4.1: Synthesis of the contributions.

We start in the next section with the setting of decision trees.

4.2 Optimizing the SSB and WEU Models in Decision Trees

In this section, we study the optimization of the SSB and WEU models in decision trees. In the literature, several other non-EU models have also been investigated with this representation of sequential decision problems under risk in both the resolute and sophisticated frameworks. We mention some of them below.

4.2.1 Related Work

Imprecise Probabilities. When probabilities are imprecise (i.e., a set of probability measures should be taken into account in the decisions instead of a single measure), several decision criteria can be used to evaluate a plan. In this setting, a pessimistic DM will make the decision that maximizes the worst possible expected utility. This is known as the Γ-maximin criterion. Conversely, an optimistic agent will make the decision that maximizes the best possible expected utility. This is known as the Γ-maximax criterion. Kikuti *et al.* [2011] proposed algorithms enabling a sophisticated DM to determine her preferred plan with respect to both criteria. Their algorithms rely on linear/multilinear programming. Subsequently, Fargier *et al.* [2011] studied the optimization of the Γ-maximin criterion under the dictatorship of the root paradigm. They showed that even the evaluation of a plan

according to the Γ-maximin criterion is NP-hard, and proposed a procedure to determine an optimal plan derived from preliminary results by Huntley and Troffaes [2008]. The Hurwicz criterion for imprecise probabilities is a convex combination of the Γ-maximin and the Γ-maximax criteria, that can model intermediate attitudes w.r.t. ambiguity [Jaffray and Jeleva, 2007]. The optimization of the Hurwicz criterion has been studied under the dictatorship of the root paradigm [Jeantet and Spanjaard, 2009]. Once again, the determination of an optimal plan according to the Hurwicz criterion is NP-hard.

Qualitative Decision Trees with Possibilities. When the information about uncertainty cannot be quantified in a probabilistic way, a possible alternative is to use possibilities. Possibility theory [Zadeh, 1999] is a qualitative counterpart of probability theory which measures uncertainty by stating if events are possible. Total certainty arises when one outcome is totally possible, all others being impossible, and total uncertainty arises when everything is totally possible. In this qualitative setting, optimistic and pessimistic counterparts of EU have been proposed [Dubois and Prade, 1995], and a compromise between these two criteria is defined by the binary possibilistic utility criterion which evaluates each possibilistic lottery with a pair giving both its best and worst possible consequences [Giang and Shenoy, 2001]. These criteria can be optimized in decision trees in polynomial time by using a backward induction method [Garcia and Sabbadin, 2006]. Other possibilistic criteria exist, as the possibilistic likely dominance criterion [Dubois *et al.*, 2003] and the possibilistic Choquet integral [Rébillé, 2006], which are possibilistic counterparts of the probabilistic dominance criterion and the Choquet integral. While a standard dynamic programming method cannot be used for possibilistic likely dominance as this criterion does not induce a transitive preference relation, Ben Amor *et al.* [2014] showed how to adapt the backward induction method to solve a decision tree in polynomial time. In each node, their algorithm keeps a subset of strategies whose size can be polynomially bounded. They also showed that solving a decision tree with the possibilistic Choquet integral is NP-hard.

Rank-Dependent Utility (RDU) model. The RDU model proposed by Quiggin [1993], where one specifies a probability distortion function, is compatible with Allais' paradox. Nielsen and Jaffray [2006] have studied the solution of sequential decision problems with RDU, under the resolute choice with selves paradigm. They proposed an operational approach eliminating any plan that appears to be largely suboptimal for RDU in some decision situation. Their approach returns an RDU-optimal plan (viewed from the root) among the remaining ones, such that no other plan stochastically dominates it. Jeantet *et al.* [2012] proposed another implementation of RDU theory in decision trees. The difference with the previous approach is that the cooperation between selves is formalized through the use of a weighted max regret criterion, where regrets measure RDU losses. Lastly, Jeantet and Spanjaard [2008] investigated the solution of sequential decision problems with the RDU model, under the dictatorship of the root paradigm. They showed that the problem is NP-hard and designed a branch and bound procedure to solve it.

The work of the following subsection falls in the same line of research. We study the algorithmic properties of the SSB and WEU models in the setting of decision trees.

4.2.2 Optimizing the SSB Criterion in the Randomized Setting

In this section, we wish to solve decision trees according to the SSB model. Let us briefly recall how this model works.

In the SSB model, an agent is endowed with a binary functional φ over pairs (x, y), with $x \succ y \Leftrightarrow \varphi(x, y) > 0$. The value $\varphi(x, y)$ is the signed intensity with which the decision maker prefers x to y. Functional φ is skew symmetric, i.e., $\varphi(x, y) = -\varphi(y, x)$ and bilinear with respect to the usual mixture operation on lotteries. Put another way, the SSB criterion for comparing two lotteries l_1 and l_2 is written:

$$\varphi(l_1, l_2) = \sum_{x,y} l_1(x) l_2(y) \varphi(x, y)$$

We have $l_1 \succ$ (resp. $\prec$) l_2 if $\varphi(l_1, l_2) >$ (resp. $<$) 0 (strict preference), and $l_1 \sim l_2$ if $\varphi(l_1, l_2) = 0$ (indifference). This model will be used to compare the different strategies that can be defined in a decision tree on the basis of the lotteries that they induce on the possible outcomes of the tree.

In a decision tree, a strategy is a rule which indicates what choices the decision maker should make in each decision node. We recall that there are three different types of strategies.

- *Deterministic strategies.* A deterministic strategy δ is a function which assigns to each decision node (reachable by the strategy) the choice that should be made in this node, i.e., an edge coming out from this node. The set of deterministic strategies is denoted by Δ.

- *Randomized strategies.* A randomized strategy δ_r is a function which assigns to each decision node (reachable by the strategy) a probability distribution over the possible choices that can be made in this node, i.e., a probability distribution over the edges coming out from this node. The set of randomized strategies is denoted by Δ_r.

- *Mixed strategies.* Instead of using a randomized strategy, a decision maker can rely on a *mixed strategy*. A mixed strategy $\tilde{\delta}$ is simply a lottery over deterministic strategies. Put another way, using the mixed strategy $\tilde{\delta} = (\delta_1, p_1; \ldots; \delta_k, p_k)$ consists in sampling a deterministic strategy δ_i according to $\tilde{\delta}$ and following δ_i thereafter. The set of mixed strategies is denoted by $\widetilde{\Delta}$.

Each strategy δ, whatever its type, induces a lottery l_δ over the possible outcomes of the decision trees $\mathscr{T}$ (the outcomes are received by the decision maker at the leaves of the tree). We will denote by $\mathscr{X}$ the set of possible outcomes of $\mathscr{T}$. Comparing strategies according to the SSB model reduces then to comparing their induced lotteries over $\mathscr{X}$.

Let $\mathscr{L} = \{l_\delta : \delta \in \Delta\}$ (resp. $\mathscr{L}_r = \{l_\delta : \delta \in \Delta_r\}$) denote the image of deterministic (resp. randomized) strategies in the space of lotteries. Additionally, we denote by $\widetilde{\mathscr{L}} = \{l_\delta : \delta \in \widetilde{\Delta}\} = \mathrm{CH}(\mathscr{L})$ the image of mixed strategies in the space of lotteries.

Obviously, $\mathscr{L} \subseteq \widetilde{\mathscr{L}}$ and $\mathscr{L} \subseteq \mathscr{L}_r$. However, it is not obvious to see how the two sets $\widetilde{\mathscr{L}}$ and $\mathscr{L}_r$ compare to one another. This question is answered by the following theorem.

Theorem 12 (Collins and McNamara [1998])**.** *The image of randomized strategies in the space of lotteries is equal to the image of mixed strategies in the space of lotteries. More formally,* $\mathscr{L}_r = \widetilde{\mathscr{L}}$.

Proof. This result has been proved by Collins and McNamara [1998] in the setting of finite horizon MDPs. As a decision tree can be seen as a specific finite horizon MDP, their result also holds for decision trees.

We give the sketch of their proof. let $\tilde{\delta} = (\delta_1, \lambda_1; \ldots; \delta_k, \lambda_k)$ be a mixed strategy. Given a decision node D and an edge (D,N), each strategy δ_i induces probabilities p^i_{D} and $p^i_{(\mathrm{D},\mathrm{N})}$ of respectively reaching node D and taking edge (D,N). Let δ_r be the randomized strategy that in each decision node D (such that $\sum_{i=1}^k \lambda_i p^i_{\mathrm{D}} > 0$) chooses edge (D,N) with probability $\frac{\sum_{i=1}^k \lambda_i p^i_{(\mathrm{D},\mathrm{N})}}{\sum_{i=1}^k \lambda_i p^i_{\mathrm{D}}}$. Then, it can be shown that $l_{\delta_r} = l_{\tilde{\delta}}$. Note that this provides an efficient procedure to compute from a mixed strategy a randomized strategy that yields the same lottery over the outcomes of the decision tree.

On the other hand, let δ_r be a randomized strategy which advocates to take edge (D,N) with probability $\alpha(\mathrm{D},\mathrm{N})$ when the decision maker is in decision node D ($\alpha(\mathrm{D},\mathrm{N})$ is considered to be 0 if D is not reachable with strategy δ_r). Let $\tilde{\delta}$ be the mixed strategy which samples each deterministic strategy δ_i with probability $\lambda_i = \Pi_{(\mathrm{D},\mathrm{N})\in\delta_i} \alpha(\mathrm{D},\mathrm{N})$. Then, it can be shown that $l_{\delta_r} = l_{\tilde{\delta}}$. Note that this is an existence result, and that it does not provide an efficient procedure to compute from a randomized strategy a mixed strategy that yields the same lottery over outcomes (as the support of $\tilde{\delta}$ can be of combinatorial size). □

Note that the proof of this theorem is constructive. In particular from any mixed strategy $\tilde{\delta}$, a randomized strategy δ_r such that $l_{\tilde{\delta}} = l_{\delta_r}$ can be efficiently computed. *As a consequence of this result, optimizing over $\mathscr{L}_r$ is equivalent to optimizing over $\widetilde{\mathscr{L}}$ and is therefore equivalent to optimizing over $\mathscr{L}_r \cup \widetilde{\mathscr{L}} \cup \mathscr{L}$*. Hence, there always exists an SSB optimal lottery in $\mathscr{L}_r$ and $\widetilde{\mathscr{L}}$. However, an SSB optimal lottery (i.e., a lottery which is at least as good as any other lottery in $\mathscr{L}_r$ with respect to the minmax rule) does not always exists in set $\mathscr{L}$ as illustrated by the following example:

Example 44. *We recall the Rowett dice paradox (first presented in Example 19). Consider a two-player game involving three six-sided dice: die* A *with sides* $(1,4,4,4,4,4)$*, die* B *with sides* $(3,3,3,3,3,6)$ *and die* C *with sides* $(2,2,2,5,5,5)$*. The players, each equipped with a personal set of dice, simultaneously choose a die to throw; the winner is the player who rolls the highest number.*

This decision problem is represented as a decision tree in Figure 4.1. In this decision tree, there are three possible deterministic strategies (corresponding to the three dice that can be chosen) δ_{A}, δ_{B} *and* δ_{C} *such that* $l_{\delta_{\mathrm{A}}} = (4,5/6;1,1/6)$, $l_{\delta_{\mathrm{B}}} = (6,1/6;3,5/6)$ *and* $l_{\delta_{\mathrm{C}}} = (5,1/2;2,1/2)$*. Finding a strategy which maximizes the probability of winning the two-player game induced by the two sets of Rowett dice reduces to solving this decision tree with respect to the SSB utility function* φ *where* $\varphi(x,y)$ *is defined by 1 if* $x > y$*,* -1 *if* $y > x$ *and 0 otherwise.*

With this SSB utility function, the intensities of preference between the three possible deterministic strategies have the following values:

$$\varphi(l_{\delta_{\mathrm{A}}}, l_{\delta_{\mathrm{B}}}) = 25/36 - 11/36 = 14/36,$$
$$\varphi(l_{\delta_{\mathrm{B}}}, l_{\delta_{\mathrm{C}}}) = 21/36 - 15/36 = 6/36,$$
$$\varphi(l_{\delta_{\mathrm{C}}}, l_{\delta_{\mathrm{A}}}) = 21/36 - 15/36 = 6/36.$$

Therefore, die A *(resp.* B, C*) wins most of the time against die* B *(resp.* C, A*). Thus, we have a preference cycle between strategies* δ_{A}, δ_{B} *and* δ_{C}*. According to the minmax rule, the optimal value of a deterministic strategy is:*

$$\min_{\delta\in\Delta} \max_{\delta'\in\tilde{\Delta}} \varphi(l_{\delta'}, l_{\delta}) = 6/36.$$

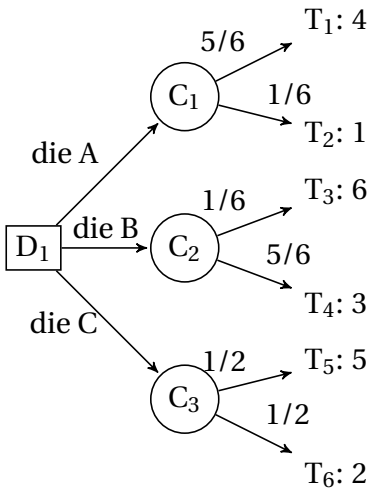


Figure 4.1: Rowett dice paradox as a decision tree problem.

Now consider the mixed strategy $\tilde{\delta} = (\delta_A, 3/13; \delta_B, 3/13, \delta_C, 7/13)$*, one can easily check that:*

$$\varphi(l_{\tilde{\delta}}, l_{\delta_A}) = \frac{3}{13} \times 0 - \frac{3}{13} \times \frac{14}{36} + \frac{7}{13} \times \frac{6}{36} = 0,$$
$$\varphi(l_{\tilde{\delta}}, l_{\delta_B}) = \frac{3}{13} \times \frac{14}{36} + \frac{3}{13} \times 0 - \frac{7}{13} \times \frac{6}{36} = 0,$$
$$\varphi(l_{\tilde{\delta}}, l_{\delta_C}) = -\frac{3}{13} \times \frac{6}{36} + \frac{3}{13} \times \frac{6}{36} + \frac{7}{13} \times 0 = 0.$$

Thus, the minmax SSB value of strategy $\tilde{\delta}$ *is:*

$$\max_{\delta' \in \widetilde{\Delta}} \varphi(l_{\delta'}, l_{\tilde{\delta}}) = 0 < 6/36.$$

This simple example proves that an SSB optimal deterministic strategy may not exist.

We now show how to compute an optimal randomized or mixed strategy. This corresponds to solving the following optimization problem:

$$(\mathscr{R}) : \min_{\delta \in \widetilde{\Delta}} \max_{\delta' \in \widetilde{\Delta}} \varphi(l_{\delta'}, l_{\delta}) \tag{4.1}$$

$$\text{(or equivalently) } (\mathscr{R}) : \min_{\delta \in \Delta_r} \max_{\delta' \in \Delta_r} \varphi(l_{\delta'}, l_{\delta}). \tag{4.2}$$

To solve problem ($\mathscr{R}$) ($\mathscr{R}$ for randomized), we resort to a game-theoretic analysis of this optimization problem.

A Game Theoretic Analysis. The problem of computing an SSB optimal strategy in a decision tree $\mathscr{T}$ can be modeled as the search for a Nash equilibrium in a zero-sum two-player symmetric game induced by the decision tree $\mathscr{T}$ and the SSB utility function φ. For both players, the set of pure strategies in this game is the set of deterministic strategies, and the payoff function is inferred from the SSB utility function. More precisely, in this game, each player plays simultaneously a deterministic strategy in Δ and the payoff of playing strategy δ against strategy δ' yields a payoff of $\varphi(l_{\delta'}, l_{\delta})$. Determining a Nash equilibrium in this game is equivalent to solving problem ($\mathscr{R}$). It is well-known that there always exists a symmetric Nash equilibrium in a finite zero-sum symmetric game. The following holds for such a Nash equilibrium (δ^*, δ^*):

$$\forall \delta \in \widetilde{\Delta}, \varphi(l_{\delta^*}, l_{\delta}) \geq \varphi(l_{\delta^*}, l_{\delta^*}) = 0.$$

This property actually characterizes the optimal solutions of problem $(\mathscr{R})$.

Proposition 9. *A strategy δ is an optimal solution of problem $(\mathscr{R})$ iff:*

$$\forall \delta' \in \widetilde{\Delta},\ \varphi(l_\delta, l_{\delta'}) \geq 0.$$

Proof. Note that as for any strategy δ, $\varphi(l_\delta, l_\delta) = 0$, the optimal value of problem $(\mathscr{R})$ is lower bounded by 0. We have just seen that this value was reached by any (possibly mixed) strategy $\delta^* \in \widetilde{\Delta}$ such that (δ^*, δ^*) is a symmetric Nash equilibrium of the game induced by the decision tree and the SSB utility function φ. Therefore, the optimal value of $(\mathscr{R})$ is 0 and a strategy δ is an optimal solution of problem $(\mathscr{R})$ iff:

$$\begin{aligned} &\max_{\delta' \in \widetilde{\Delta}} \varphi(l_{\delta'}, l_\delta) = 0, \\ \Leftrightarrow\ &\forall \delta' \in \widetilde{\Delta},\ \varphi(l_\delta, l_{\delta'}) \geq \varphi(l_\delta, l_\delta) = 0. \end{aligned}$$

□

Thus any SSB optimal (possibly mixed or randomized) strategy has the desirable property of being preferred to all mixed strategies and a fortiori to all strategies. Once realized that a Nash equilibrium of the game on deterministic strategies provides us an SSB optimal strategy, our aim becomes to solve this game. Unfortunately, the set of deterministic strategies is combinatorial in nature, which makes impractical the generation of the whole payoff matrix of the game. However, the oracle methods described in Section 3.1 (on page 80) can be used to solve this game efficiently.

To use these oracle methods, a best response oracle $\mathscr{O}$ needs to be instantiated. For this purpose, we start by noting that:

$$\max_{\delta' \in \widetilde{\Delta}} \varphi(l_{\delta'}, l_\delta) = \max_{\delta' \in \Delta} \varphi(l_{\delta'}, l_\delta) \tag{4.3}$$

for any strategy $\delta \in \widetilde{\Delta}$. Indeed, this is a direct consequence of Proposition 7 (on page 82) which states that a best response can always be found as a pure strategy of the game (a deterministic strategy in Δ in our case). Thus given any strategy δ a strategy that is most preferred to δ can always be found as a deterministic strategy.

We now rewrite the objective function in a way highlighting the fact that determining a strategy δ' that is most preferred to strategy δ can be done by rolling back the decision tree $\mathscr{T}$ with a particular utility function V_δ.
By linearity of φ, we have:

$$\max_{\delta' \in \Delta} \varphi(l_{\delta'}, l_\delta) = \max_{\delta' \in \Delta} \sum_{x \in \mathscr{X}} l_{\delta'}(x) \varphi(x, l_\delta) \tag{4.4}$$

This latter expression can be rewritten as follows:

$$\max_{\delta' \in \Delta} \sum_{x \in \mathscr{X}} l_{\delta'}(x) \varphi(x, l_\delta) = \max_{\delta' \in \Delta} \sum_{x \in \mathscr{X}} l_{\delta'}(x) V_\delta(x) \tag{4.5}$$

where utilities $V_\delta(x)$ are defined by $V_\delta(x) = \varphi(x, l_\delta)$. Thus, a best response oracle $\mathscr{O}$ consists here in rolling back the decision tree with utility function $V_\delta(x)$ and returning the optimal deterministic strategy found. Based on this oracle procedure, several oracle methods can be used. We first propose a double oracle procedure.

A Double Oracle Approach. The adaptation of the double oracle approach (explained in subsection 3.1.3 on page 85) for solving problem $(\mathscr{R})$ is described in Algorithm 6. Note that some simplifications arise due to the fact that the game is symmetric:

- Firstly the game only maintains *one* set of strategies $S \subseteq \Delta$ (for both players) and runs the oracle only *once* per iteration. The oracle $\mathscr{O}$ finds a best response δ to the strategy $\tilde{\delta}$, given by a symmetric Nash equilibrium $(\tilde{\delta}, \tilde{\delta})$ of the restricted game $M(S,S)$, where M denotes the payoff matrix of the game induced by $\mathscr{T}$ and by φ.

- Secondly the stopping condition is here simply $\varphi(l_\delta, l_{\tilde{\delta}}) \leq 0$. Indeed, by Proposition 9, this condition (which occurs if the oracle computes a strategy which is already in S) is sufficient to ensure that $(\tilde{\delta}, \tilde{\delta})$ is a Nash equilibrium of the whole game.

At the end of the algorithm, an optimal mixed strategy $\tilde{\delta}^*$ is found. From this strategy, an optimal randomized strategy can be determined (using Theorem 12).

Algorithm 6: Double oracle algorithm to solve problem $(\mathscr{R})$

Data: Decision tree $\mathscr{T}$, singleton $S = \{\delta\}$ including an arbitrary pure strategy $\delta \in \Delta$
Result: an SSB optimal randomized strategy $\delta_r \in \Delta_r$

```
converge = False
while converge is False do
    Find a Nash equilibrium (δ̃, δ̃) of the game M(S, S)
    Find δ = O(δ̃) ∈ Δ
    if φ(l_δ, l_δ̃) ≤ 0 then
        converge =True
    else
        add δ to S
Compute randomized strategy δ_r s.t. l_{δ_r} = l_δ̃.
return δ_r
```

This double oracle approach is a first method which enables to solve decision trees with the SSB model efficiently. However, this method is not polynomial time. Thus, it does not tell us if finding an SSB optimal strategy is a problem of polynomial complexity. To answer this question, we turn to a linear programming approach.

Linear Programming Approach. We now present a Linear Program (LP) for solving optimization problem $(\mathscr{R})$.

The LP we propose is composed of two classes of constraints. The first class of constraints involves a set of variables p_N, describing all feasible randomized strategies $\delta_r \in \Delta_r$, where p_N is the probability that node N is reached with strategy δ_r:

$$\begin{aligned} p_{N_r} &= 1 \\ p_D &= \sum_{N \in \mathscr{S}(D)} p_N, \forall D \in \mathscr{N}_D \\ p_{(C,N)}\, p_C &= p_N, \ \forall C \in \mathscr{N}_C, \ \forall N \in \mathscr{S}(C) \\ p_N &\geq 0, \forall N \in \mathscr{N} \end{aligned}$$

The strategy δ_r (more precisely, its incidence vector χ^{δ_r}) can be recovered from variables p_N by using equation $\chi^{\delta_r}_{(D,N)} = p_N / p_D$ (or 0 if $p_D = 0$) for each edge $(D,N) \in \mathscr{E}_D$. The second class of constraints aims to determine $\max_{\delta' \in \Delta} \sum_{x \in \mathscr{X}} l_{\delta'}(x) V_{\delta_r}(x)$ (i.e., determining the intensity of preference induced by a best response to δ_r). As already noted, this value can be

computed by rolling back the decision tree, which amounts to satisfy the following Bellman's equations, where q_{N} denotes the EU value in node N according to utility function V_{δ_r} defined as above by $\mathrm{V}_{\delta_r}(x) = \varphi(x, l_{\delta_r})$:

$$\begin{aligned}
&\min\ q_{\mathrm{N}_r}\\
&q_{\mathrm{T}} = \mathrm{V}_{\delta_r}(o(\mathrm{T})), \forall \mathrm{T} \in \mathscr{N}_{\mathrm{T}}\\
&q_{\mathrm{C}} = \sum_{\mathrm{N} \in \mathscr{S}(\mathrm{C})} p_{(\mathrm{C},\mathrm{N})}\, q_{\mathrm{N}}, \forall \mathrm{C} \in \mathscr{N}_{\mathrm{C}}\\
&q_{\mathrm{D}} \geq q_{\mathrm{N}}, \forall \mathrm{N} \in \mathscr{S}(\mathrm{D}),\ \forall \mathrm{D} \in \mathscr{N}_{\mathrm{D}}\\
&q_{\mathrm{N}} \in \mathbb{R}, \forall \mathrm{N} \in \mathscr{N}
\end{aligned}$$

Therefore, we have $q_{\mathrm{N}_r} = \max_{\delta' \in \Delta} \sum_{x \in \mathscr{X}} l_{\delta'}(x) \mathrm{V}_{\delta_r}(x)$. Putting together the objective function $\min q_{\mathrm{N}_r}$ and both classes of constraints, and replacing values $\mathrm{V}_{\delta_r}(o(\mathrm{T}))$ by the expressions $\sum_{\mathrm{T}' \in \mathscr{N}_{\mathrm{T}}} p_{\mathrm{T}'} \varphi(o(\mathrm{T}), o(\mathrm{T}'))$, we obtain the final program $\mathscr{P}_{\mathrm{SSB}}$ given below, that enables to determine an SSB optimal randomized strategy.

$$\mathscr{P}_{\mathrm{SSB}} \begin{cases}
\min q_{\mathrm{N}_r}\\
p_{\mathrm{N}_r} = 1\\
p_{\mathrm{D}} = \sum_{\mathrm{N} \in \mathscr{S}(\mathrm{D})} p_{\mathrm{N}}, \forall \mathrm{D} \in \mathscr{N}_{\mathrm{D}}\\
p_{(\mathrm{C},\mathrm{N})}\, p_{\mathrm{C}} = p_{\mathrm{N}},\ \forall \mathrm{C} \in \mathscr{N}_{\mathrm{C}},\ \forall \mathrm{N} \in \mathscr{S}(\mathrm{C})\\
q_{\mathrm{T}} = \sum_{\mathrm{T}' \in \mathscr{N}_{\mathrm{T}}} p_{\mathrm{T}'} \varphi(o(\mathrm{T}), o(\mathrm{T}')), \forall \mathrm{T} \in \mathscr{N}_{\mathrm{T}}\\
q_{\mathrm{C}} = \sum_{\forall \mathrm{N} \in \mathscr{S}(\mathrm{C})} p_{(\mathrm{C},\mathrm{N})}\, q_{\mathrm{N}}, \forall \mathrm{C} \in \mathscr{N}_{\mathrm{C}}\\
q_{\mathrm{D}} \geq q_{\mathrm{N}}, \forall \mathrm{N} \in \mathscr{S}(\mathrm{D}),\ \forall \mathrm{D} \in \mathscr{N}_{\mathrm{D}}\\
p_{\mathrm{N}} \geq 0, \forall \mathrm{N} \in \mathscr{N}\\
q_{\mathrm{N}} \in \mathbb{R}, \forall \mathrm{N} \in \mathscr{N}
\end{cases}$$

The linear program $\mathscr{P}_{\mathrm{SSB}}$ involves a polynomial number of variables and constraints (in the size of the decision tree). Thus, by polynomial complexity of linear programming [Khachiyan, 1980], determining an SSB optimal randomized strategy in a decision tree (i.e., solving optimization problem ($\mathscr{R}$)) is a polynomial time problem.

Theorem 13. *Finding an SSB optimal strategy in a decision tree, i.e., solving optimization problem* ($\mathscr{R}$)*, is a problem of polynomial complexity.*

Interestingly, linear program $\mathscr{P}_{\mathrm{SSB}}$ can be used to build a polynomial variant of the double oracle algorithm presented in Algorithm 6.

Polynomial Variant of the Double Oracle Approach. A standard double oracle algorithm builds a restricted game in normal form such that a Nash equilibrium of the restricted game is also a Nash equilibrium of the whole game. This algorithm does not take advantage of the structure underlying the game (i.e., the decision tree in our case). The variant we propose (Algorithm 7) tries to take advantage of this structure.

Instead of building a restricted game in normal form, this variant iteratively builds a subtree $\mathscr{T}'$ of $\mathscr{T}$ (which we hope stays much smaller) such that an optimal randomized SSB strategy in $\mathscr{T}'$ is also optimal in $\mathscr{T}$. To build $\mathscr{T}'$, this algorithm uses the same best response oracle as Algorithm 6.

The algorithm starts with a subtree $\mathscr{T}'$ of $\mathscr{T}$ which is only composed of the nodes and edges that can be reached by a deterministic strategy chosen arbitrarily. At each iteration, the algorithm computes an SSB-optimal randomized strategy δ_r in $\mathscr{T}'$ (Line 4). This strategy is also a valid randomized strategy in $\mathscr{T}$. However, it may not be optimal in $\mathscr{T}$. To test optimality, we determine a strategy δ which is a best response to δ_r in $\mathscr{T}$ (Line 5). If $\varphi(\delta, \delta_r) \leq 0$, then δ_r is also an optimal strategy in $\mathscr{T}$ and the algorithm terminates (Line 9). Otherwise, all nodes and edges that can be reached by δ are added to $\mathscr{T}'$ and δ becomes a new valid strategy of $\mathscr{T}'$ (Line 7).

Algorithm 7: Polynomial variant of the double oracle method to solve problem $(\mathscr{R})$

Data: Decision tree $\mathscr{T}$, empty decision tree $\mathscr{T}'$, arbitrary strategy $\delta \in \Delta$
Result: an SSB optimal randomized strategy $\delta_r \in \Delta_r$

1 Add to $\mathscr{T}'$ all nodes and edges of $\mathscr{T}$ that can be reached by δ.
2 converge = False
3 **while** *converge is False* **do**
4 Find δ_r, an SSB optimal randomized strategy in $\mathscr{T}'$ by using linear program $\mathscr{P}_{\text{SSB}}$ applied to $\mathscr{T}'$
5 Find $\delta = \mathscr{O}(\delta_r) \in \Delta$ in $\mathscr{T}$
6 **if** $\varphi(\delta, \delta_r) > 0$ **then**
7 Add to $\mathscr{T}'$ all nodes and edges of $\mathscr{T}$ that can be reached by δ.
8 **else**
9 converge =True
10 **return** δ_r

This method seems to combine the advantages of the two previous approaches. First it can solve the decision tree while only focussing on the relevant parts of the decision tree, which may be very small in real life applications. Secondly, it is a polynomial time method. Indeed, note that each iteration can be performed in polynomial time, as solving $\mathscr{P}_{\text{SSB}}$ applied to $\mathscr{T}'$ and determining $\delta = \mathscr{O}(\delta_r)$ can be done in polynomial time. Moreover, note that the number of iterations of the method is bounded by the number of nodes in $\mathscr{T}$. Indeed, at each iteration, at least one new node is added to $\mathscr{T}'$. In the worst case, the method adds all nodes and edges of $\mathscr{T}$ to $\mathscr{T}'$ and the method terminates by solving $\mathscr{P}_{\text{SSB}}$ applied to $\mathscr{T}$.

A Cutting Plane Approach. The last oracle method we propose is a cutting plane method (this type of method is presented in subsection 3.1.4 on page 87). Similarly to program $\mathscr{P}_{\text{SSB}}$, the induced linear program is composed of two sets of constraints. Unlike problem $\mathscr{P}_{\text{SSB}}$, one of these sets of constraints is of combinatorial size.

The first set of constraints (constraints 4.6 to 4.8) is the same as in LP $\mathscr{P}_{\text{SSB}}$. It describes all feasible randomized strategies $\delta_r \in \Delta_r$, with variables p_{N}, where p_{N} is the probability that node N is reached with strategy δ_r.

The second set of constraints aims to represent all possible deterministic strategies that can be played by the fictitious adversary. The value $\max_{\delta' \in \Delta} \varphi(l_{\delta'}, l_{\delta_r})$ is attained by variable λ under the following constraints:

$$\min \lambda$$
$$\lambda \geq \sum_{\text{T} \in \mathscr{N}_{\text{T}}} \sum_{\text{T}' \in \mathscr{N}_{\text{T}}} p^{\delta}_{\text{T}} p_{\text{T}'} \varphi(o(\text{T}), o(\text{T}')), \forall \delta \in \Delta$$

where p_{T}^{δ} is the probability with which leaf T is reached with strategy δ.

Putting together the objective function $\min \lambda$ and both classes of constraints, we obtain the final program $\mathscr{P}_{\mathrm{SSB}}^{\mathrm{CP}}$ given below.

$$\mathscr{P}_{\mathrm{SSB}}^{\mathrm{CP}} \begin{cases} \min \lambda & \\ p_{\mathrm{N}_r} = 1 & (4.6) \\ p_{\mathrm{D}} = \sum_{\mathrm{N} \in \mathscr{S}(\mathrm{D})} p_{\mathrm{N}}, \forall \mathrm{D} \in \mathscr{N}_{\mathrm{D}} & (4.7) \\ p_{(\mathrm{C},\mathrm{N})}\, p_{\mathrm{C}} = p_{\mathrm{N}},\ \forall \mathrm{C} \in \mathscr{N}_{\mathrm{C}},\ \forall \mathrm{N} \in \mathscr{S}(\mathrm{C}) & (4.8) \\ \lambda \geq \sum_{\mathrm{T} \in \mathscr{N}_{\mathrm{T}}} \sum_{\mathrm{T}' \in \mathscr{N}_{\mathrm{T}}} p_{\mathrm{T}}^{\delta} p_{\mathrm{T}'} \varphi(o(\mathrm{T}), o(\mathrm{T}')), \forall \delta \in \Delta & (4.9) \\ p_{\mathrm{N}} \geq 0, \forall \mathrm{N} \in \mathscr{N} & \\ \lambda \in \mathbb{R} & \end{cases}$$

We wish to solve program $\mathscr{P}_{\mathrm{SSB}}^{\mathrm{CP}}$ by using a cutting plane approach. A cutting plane algorithm makes it possible to solve a linear program involving an exponential number of constraints to define a polyhedron P, provided there exists a separation oracle. In program $\mathscr{P}_{\mathrm{SSB}}^{\mathrm{CP}}$, polyhedron P is defined by constraints 4.6 to 4.8 and constraints 4.9. Given $(\lambda, \mathbf{p}) \in \mathbb{R}^{|\mathscr{N}|+1}$, a separation oracle should determine whether $(\lambda, \mathbf{p})$ belongs to P or not, and finds a separating hyperplane in the latter case. The proposed separation oracle consists of a separation oracle for the polyhedron P_1 defined by constraints 4.6 to 4.8 and a separation oracle for the polyhedron P_2 defined by constraints 4.9:

- The set of constraints defining polyhedron P_1 is composed of a polynomial (in the size of $\mathscr{T}$) number of constraints. Therefore, given values for variables $p_{\mathrm{N}}, \mathrm{N} \in \mathscr{N}$, checking if one of these constraints is violated can be done in polynomial time.
- Regarding polyhedron P_2, we use the separation oracle which computes a best response δ to the randomized strategy δ_r described by variables $p_{\mathrm{N}}, \mathrm{N} \in \mathscr{N}$ (by rolling back the decision tree with utility values $\mathrm{V}_{\delta_r}(o(\mathrm{T}))$) and checks if the corresponding constraint is satisfied, i.e., it verifies that the following inequality holds:

$$\lambda \geq \sum_{\mathrm{T} \in \mathscr{N}_{\mathrm{T}}} \sum_{\mathrm{T}' \in \mathscr{N}_{\mathrm{T}}} p_{\mathrm{T}}^{\delta} p_{\mathrm{T}'} \varphi(o(\mathrm{T}), o(\mathrm{T}')).$$

  If it does hold, then constraints 4.9 are all satisfied.

Combining both oracles yields a separation oracle for polyhedron $\mathrm{P} = \mathrm{P}_1 \cap \mathrm{P}_2$, which is polynomial time as the separation oracles for P_1 and P_2 are both polynomial time. Therefore the complexity of solving $\mathscr{P}_{\mathrm{SSB}}^{\mathrm{CP}}$ is polynomial (separation → optimization) by the polynomial time equivalence of optimization and separation (Theorem 10 on page 89). This is another proof that solving problem ($\mathscr{R}$) is of polynomial complexity.

In this subsection, we have seen that solving problem ($\mathscr{R}$) is of polynomial complexity and we have explored four different methods to compute an SSB optimal mixed or randomized policy. However, our intuition is that a decision maker may not accept to use a randomized strategy.

Therefore, we now turn in the next subsection to the deterministic setting, in which we are interested in determining a deterministic strategy which is SSB optimal within the set of deterministic strategies. Stated differently, we will seek a deterministic strategy which is at least as good as all other deterministic strategies with respect to the minmax rule.

4.2.3 Optimizing the SSB Criterion in the Deterministic Setting

We now wish to compute an optimal strategy among the set of deterministic strategies. This corresponds to solving the following optimization problem:

$$(\mathscr{D}): \min_{\delta\in\Delta} \max_{\delta'\in\Delta} \varphi(l_{\delta'}, l_{\delta}) \tag{4.10}$$

In this deterministic setting ($\mathscr{D}$ for deterministic), the problem of determining an SSB optimal strategy becomes much harder. In fact, the determination of an SSB optimal *deterministic* strategy in a decision tree is an NP-hard problem, where the size of the instance is the number of involved decision nodes.

Theorem 14. *Finding an SSB optimal deterministic strategy in a decision tree (solving* $(\mathscr{D})$*) is an* NP*-hard problem.*

Proof. To ease the presentation, the proof is divided into two parts. While the first part contains all the important ideas of the proof, it proves the result with an SSB utility function which expresses intransitive preferences over the (certain) outcomes of the tree. As this property can seem unnatural or restrictive, the second part of the proof extends the result to an SSB utility function with transitive preferences over the outcomes of the tree.

Part 1: The proof relies on a polynomial reduction from 3-SAT, which can be stated as follows:

INSTANCE: a set X of boolean variables, a collection $\mathscr{C}$ of clauses on X such that $|c| = 3$ for every clause $c \in \mathscr{C}$.
QUESTION: does there exist an assignment of truth values to the boolean variables of X that satisfies simultaneously all the clauses in $\mathscr{C}$?

Let $X = \{x_1, \ldots, x_n\}$ and $\mathscr{C} = \{c_1, \ldots, c_m\}$. The polynomial generation of a decision tree from an instance of 3-SAT is performed as follows. An example is provided in Figure 4.2. One defines a decision node X_i for every variable $x_i \in X$. Each node X_i has two children: the first one (chance node denoted by T_i) corresponds to the statement "x_i is true" while the second one (chance node denoted by F_i) corresponds to the statement "x_i is false". The subset of clauses which includes the positive (resp. negative) literal x_i (resp. $\overline{x_i}$) is denoted by $\{c_{i_1}, \ldots, c_{i_j}\} \subset \mathscr{C}$ (resp. $\{c_{i'_1}, \ldots, c_{i'_k}\} \subset \mathscr{C}$). For every clause c_{i_h} (resp. $c_{i'_h}$) one generates a child of T_i (resp. F_i) denoted by c_{i_h} (resp. $c_{i'_h}$). These children are terminal nodes. Moreover, one generates an additional child of T_i (resp. F_i) denoted by c_0, corresponding to a fictive clause. Node T_i (resp. F_i) has therefore $j+1$ (resp. $k+1$) children. One adds a chance node C predecessor of all decision nodes x_i and a decision node D as root with C as child. Lastly, for every clause c_i in $\mathscr{C}$ one adds a child $\overline{c}_i$ to the root, a terminal node. The obtained decision tree involves $n+1$ decision nodes, $2n+1$ chance nodes and at most $2n(m+1)+m$ terminal nodes. Note that there is a bijection between the assignments of truth values and the subset of deterministic strategies that choose chance node C at the root. To map assignments to strategies and conversely, one sets $x_i = 1$ in problem 3-SAT iff edge (x_i, T_i) is included in the strategy, and $x_i = 0$ otherwise. An assignment such that the entire expression is true in 3-SAT corresponds to a strategy such that every clause c_i, $i = 1 \ldots m$ is a possible outcome. To complete the reduction, we need to specify the SSB function φ and the probabilities in the tree such that property $(\mathscr{P})$ holds:

$(\mathscr{P})$ If the 3-SAT formula is (resp. is not) satisfiable, then any SSB optimal deterministic strategy reaches (resp. does not reach) every clause c_i, for $i \in \{1, \ldots, m\}$, with non null

probability.

We set the probabilities in the following way. The edges starting from C have probabilities $1/n$. The edges leading to leaves $c_{i=1...m}$ have probabilities $1/m$. Thus, if c_i ($i \in \{1,\dots,m\}$) is a possible outcome of a strategy then the probability of obtaining c_i is greater than $1/nm$. The SSB function φ is defined to have the three following properties. i) If c_i ($i \in \{1,\dots,m\}$) is reached with a non null probability with strategy δ then $\varphi(l_\delta, \overline{c}_i) > 0$, ii) otherwise $\varphi(l_\delta, \overline{c}_i) < 0$. iii) Given two strategies δ and δ' both containing edge (D,C), $\varphi(l_\delta, l_{\delta'}) = 0$. Indeed, it is easy to see that i), ii) and iii) imply ($\mathscr{P}$). Those conditions are satisfied by the SSB function φ defined by: $\forall i, j \in \{0,\dots,m\}$, $\varphi(c_i, c_j) = 0$; $\forall i \in \{1,\dots,m\}$, $\forall j \neq i \in \{0,\dots,m\}$, $\varphi(\overline{c}_i, c_j) = 1$; $\forall i \in \{1,\dots,m\}$, $\varphi(c_i, \overline{c}_i) = nm$; $\forall i, j \in \{1,\dots,m\}$, $\varphi(\overline{c}_i, \overline{c}_j) = 0$.

Indeed, in this case, properties ii) and iii) obviously hold. To see that i) is also satisfied, note that:

- the probability of obtaining c_i is greater than $1/nm$ if c_i is reached with a non null probability with strategy δ;
- $\varphi(c_j, \overline{c}_i) = -1$, $\forall j \neq i$ by skew symmetry.

Thus, we have:

$$\varphi(l_\delta, \overline{c}_i) \geq \varphi(c_i, \overline{c}_i)/nm - (1 - 1/nm) = 1/nm > 0$$

where $1 - 1/nm$ is an upper bound on the probability to reach another consequence than c_i with δ.

Part 2: Part 1 of the proof uses an SSB utility function with intransitive preferences on the outcomes of the tree (as $\overline{c}_i \succ c_j \succ \overline{c}_j \succ c_i \succ \overline{c}_i$) which can seem unnatural. Thus, we now consider another SSB utility function defined on $\mathbb{R}^2$ by $\varphi(x, y)$ equal to 1 (resp. 0, -1) if $x > y$ (resp. $x = y$, $x < y$). The decision tree considered is the same as in the first part of the proof, but now, terminal nodes are replaced by the following lotteries:

$$\forall i \in \{0,\dots,m\} : c_i = (i, 0.5; -i, 0.5)$$
$$\forall i \in \{1,\dots,m\} : \overline{c}_i = (i - 0.5, p; -i - 0.5, p; m + 1, 1 - 2p)$$

with $p \in (nm/(1+2nm), 0.5)$. The number of terminal nodes is now upper bounded by $6n(m+1)+3m$ and the preferences on the clauses are now transitive. One can check that with those changes, properties i), ii) and iii) of part 1 still hold, and therefore property ($\mathscr{P}$) holds. Indeed, let δ be a deterministic strategy choosing node C at the root, then if c_i is not reached by δ, $\varphi(l_\delta, \overline{c}_i) = 2p - 1 < 0$ (as $p < 0.5$). Hence, property ii) holds. On the contrary, if c_i is reached with non null probability by δ then:

$$\begin{aligned}
\varphi(l_\delta, \overline{c}_i) &\geq \frac{1}{nm}\varphi(c_i, \overline{c}_i) - (1 - \frac{1}{nm})(1 - 2p) \\
&= \frac{1}{nm}(p - 1 + 2p) - (1 - \frac{1}{nm})(1 - 2p) \\
&= p\frac{1 + 2nm}{nm} - 1 > 0 \text{ (as } p > \frac{nm}{1 + 2nm}.)
\end{aligned}$$

Hence, property i) holds. Lastly, for all $i, j \in \{0,\dots,m\}$, we have $\varphi(c_i, c_j) = 0$. Therefore, for any pair δ, δ' of deterministic strategies choosing node C at the root, $\varphi(l_\delta, l_{\delta'}) = 0$ and condition iii) holds. □

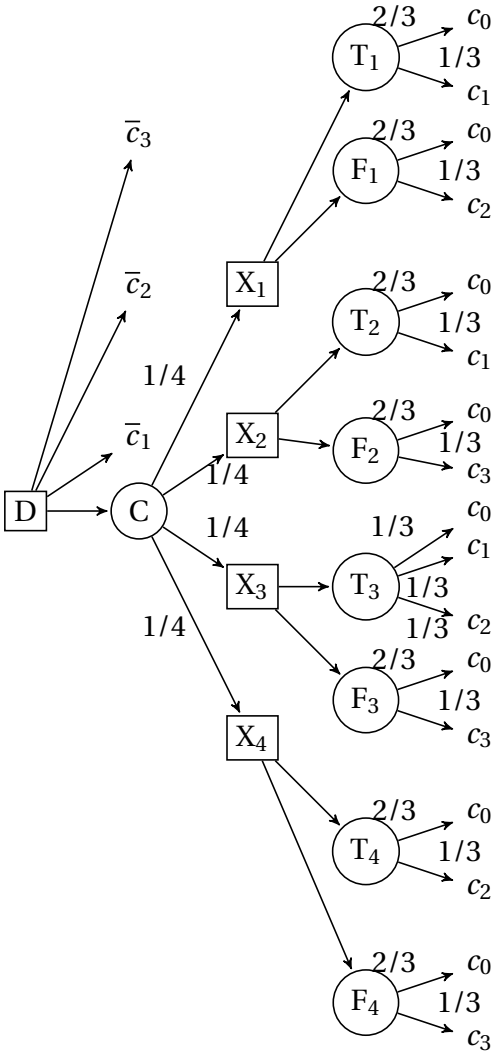


Figure 4.2: The decision tree obtained for 3-SAT formula: $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee x_3 \vee x_4) \wedge (\overline{x_2} \vee \overline{x_3} \vee \overline{x_4})$.

Moreover, a stronger result is that the decision problem associated to optimization problem $(\mathscr{D})$ is NP-complete which is not straightforward as it is not obvious that this problem is in NP.

Theorem 15. *The decision problem "Given* α*, does there exist a deterministic strategy* δ *such that* $\max_{\delta' \in \Delta} \varphi(l_{\delta'}, l_\delta) \leq \alpha$*?" is* NP*-complete.*

Proof. It belongs to NP because, if an oracle provides a strategy δ, one can obtain in polynomial time the value $\max_{\delta' \in \Delta} \varphi(l_{\delta'}, l_\delta)$ by rolling back the decision tree with utility function V_δ.

It can be proved NP-hard by using similar arguments as in the proof of Theorem 14 with $\alpha = 0$. Indeed, note that in the decision tree built in this proof, we have the property: $(\mathscr{P}')$ If the 3-SAT formula is (resp. is not) satisfiable, then there exists (resp. does not exist) a deterministic strategy δ such that $\max_{\delta' \in \Delta} \varphi(p_{\delta'}, p_\delta) \leq 0$. □

Even in the light of these negative results, problem $(\mathscr{D})$ can be solved in reasonable times for small decision tree problems. We now present a solution method to solve problem $(\mathscr{D})$.

A Mixed Integer Linear Program. To solve problem $(\mathscr{D})$, we consider a Mixed Integer Linear Program (MILP) built similarly as $\mathscr{P}_{\text{SSB}}$. This program consists of two classes of constraints. The first class of constraints involves a set of binary variables χ_{N} describing all deterministic strategies $\delta \in \Delta$.

$$\chi_{\text{N}_r} = 1 \tag{4.11}$$
$$\chi_{\text{D}} = \sum_{\text{N} \in \mathscr{S}(\text{D})} \chi_{\text{N}}, \forall \text{D} \in \mathscr{N}_{\text{D}} \tag{4.12}$$
$$\chi_{\text{C}} = \chi_{\text{N}}, \ \forall \text{N} \in \mathscr{S}(\text{C}), \ \forall \text{C} \in \mathscr{N}_{\text{C}} \tag{4.13}$$
$$\chi_{\text{N}} \in \{0,1\}, \forall \text{N} \in \mathscr{N} \tag{4.14}$$

The strategy δ (more precisely, its incidence vector χ^{δ}) can be recovered from variables χ_{N} by using equation $\chi^{\delta}_{(\text{D,N})} = \chi_{\text{N}}$ for each edge $(\text{D,N}) \in \mathscr{E}_{\text{D}}$. Note that variables χ_{N} do not represent probabilities. They are binary values indicating if the deterministic strategy δ described by variables χ_{N} reaches node N with non-zero probability. The probability with which node N is reached with strategy δ can be obtained by computing the product $\chi_{\text{N}}\text{P(N)}$ where P(N) is the product of all probabilities on the path from the root node N_r to node N.

Similarly as in the linear program $\mathscr{P}_{\text{SSB}}$, the second class of constraints aims to determine $\max_{\delta' \in \Delta} \sum_{x \in \mathscr{X}} l_{\delta'}(x)\text{V}_{\delta}(x)$ (i.e., determining the intensity of preference induced by a best response to δ). As already noted, this value can be computed by rolling back the decision tree, which amounts to satisfy the following Bellman's equations, where q_{N} denotes the EU value in node N according to utility function V_{δ} defined by $\text{V}_{\delta}(x) = \varphi(x, l_{\delta})$:

$$\begin{aligned}
&\min q_{\text{N}_r} \\
&q_{\text{T}} = \text{V}_{\delta}(o(\text{T})), \forall \text{T} \in \mathscr{N}_{\text{T}} \\
&q_{\text{C}} = \sum_{\text{N} \in \mathscr{S}(\text{C})} p_{(\text{C,N})}\, q_{\text{N}}, \forall \text{C} \in \mathscr{N}_{\text{C}} \\
&q_{\text{D}} \geq q_{\text{N}}, \forall \text{N} \in \mathscr{S}(\text{D}), \ \forall \text{D} \in \mathscr{N}_{\text{D}} \\
&q_{\text{N}} \in \mathbb{R}, \forall \text{N} \in \mathscr{N}
\end{aligned}$$

We have therefore $q_{\text{N}_r} = \max_{\delta' \in \Delta} \sum_{x \in \mathscr{X}} l_{\delta'}(x)\text{V}_{\delta}(x)$. Note that as lottery l_{δ} returns $o(\text{T}')$ with probability $\chi_{\text{T}'}\text{P(T}')$, we have $\varphi(o(\text{T}), l_{\delta}) = \sum_{\text{T}' \in \mathscr{N}_{\text{T}}} \chi_{\text{T}'}\text{P(T}')\varphi(o(\text{T}), o(\text{T}'))$. Putting together the objective function $\min q_{\text{N}_r}$ and both classes of constraints, and replacing values $\text{V}_{\delta}(o(\text{T}))$ by the expressions $\sum_{\text{T}' \in \mathscr{N}_{\text{T}}} \chi_{\text{T}'}\text{P(T}')\varphi(o(\text{T}), o(\text{T}'))$, we obtain the final program MILP_{SSB}.

$$\text{MILP}_{\text{SSB}} \begin{cases}
\min q_{\text{N}_r} \\
\chi_{\text{N}_r} = 1 \\
\chi_{\text{D}} = \sum_{\text{N} \in \mathscr{S}(\text{D})} \chi_{\text{N}}, \forall \text{D} \in \mathscr{N}_{\text{D}} \\
\chi_{\text{C}} = \chi_{\text{N}}, \ \forall \text{N} \in \mathscr{S}(\text{C}), \ \forall \text{C} \in \mathscr{N}_{\text{C}} \\
q_{\text{T}} = \sum_{\text{T}' \in \mathscr{N}_{\text{T}}} \chi_{\text{T}'}\text{P(T}')\varphi(o(\text{T}), o(\text{T}')), \forall \text{T} \in \mathscr{N}_{\text{T}} \\
q_{\text{C}} = \sum_{\forall \text{N} \in \mathscr{S}(\text{C})} p_{(\text{C,N})}\, q_{\text{N}}, \forall \text{C} \in \mathscr{N}_{\text{C}} \\
q_{\text{D}} \geq q_{\text{N}}, \forall \text{N} \in \mathscr{S}(\text{D}), \ \forall \text{D} \in \mathscr{N}_{\text{D}} \\
\chi_{\text{N}} \in \{0,1\}, \forall \text{N} \in \mathscr{N}, \\
q_{\text{N}} \in \mathbb{R}, \forall \text{N} \in \mathscr{N}, \\
\end{cases}$$

Note that the relaxed version of this MILP also solves $(\mathscr{R})$ but is less elegant than $\mathscr{P}_{\text{SSB}}$ due to the values $\{\text{P(T)} : \text{T} \in \mathscr{N}_{\text{T}}\}$ that have to be computed.

In this subsection, we have seen that solving optimization problem $(\mathscr{D})$ is NP-hard for general SSB utility functions. A natural question is then: "does there exist a subclass of SSB (wider than EU) for which $(\mathscr{D})$ is of polynomial complexity?". In the next subsection, we answer positively this question by considering the WEU model.

4.2.4 Optimizing the WEU Criterion

We now turn to the special case of the WEU model. We recall that the WEU model is a subclass of the SSB model in which function φ takes the form $\varphi(x,y) = u(x)w(y) - u(y)w(x)$, where w is a strictly positive function[1]. Let δ and δ' be two strategies, then according to the WEU model, we have:

$$\begin{aligned} l_\delta \succeq l_{\delta'} &\Leftrightarrow \sum_{x\in\mathscr{X}}\sum_{y\in\mathscr{X}} l_\delta(x)\,l_{\delta'}(y)(u(x)w(y)-u(y)w(x)) \geq 0 \\ &\Leftrightarrow \left(\sum_{x\in\mathscr{X}} l_\delta(x)u(x)\right)\left(\sum_{y\in\mathscr{X}} l_{\delta'}(y)w(y)\right) \geq \left(\sum_{x\in\mathscr{X}} l_\delta(x)w(x)\right)\left(\sum_{y\in\mathscr{X}} l_{\delta'}(y)u(y)\right) \\ &\Leftrightarrow \frac{\sum_{x\in\mathscr{X}} l_\delta(x)u(x)}{\sum_{x\in\mathscr{X}} l_\delta(x)w(x)} \geq \frac{\sum_{y\in\mathscr{X}} l_{\delta'}(y)u(y)}{\sum_{y\in\mathscr{X}} l_{\delta'}(y)w(y)} \qquad (\text{as } w>0) \end{aligned} \tag{4.15}$$

Thus, as shown by Equation 4.15, with the WEU model, each strategy δ can be valued by the ratio $u(l_\delta)/w(l_\delta)$ and solving a decision tree reduces to the optimization problem $\max_{\delta\in\widetilde{\Delta}} u(l_\delta)/w(l_\delta)$. Indeed, a strategy realizing this maximum will be preferred to all other strategies and will therefore be a WEU optimal strategy (by Proposition 9). Note that (differently from the SSB model) the WEU model always entails transitive preferences.

We first show that, with the WEU criterion, solving optimization problem $(\mathscr{D})$ is equivalent to solving optimization problem $(\mathscr{R})$.

Let $\Delta = \{\delta_1,\ldots,\delta_n\}$. For any strategy $\delta \in \Delta_r$, (as $\widetilde{\mathscr{L}} = \mathscr{L}_r$) there exists positive λ_i's summing up to 1 such that $l_\delta = \sum_{i=1}^n \lambda_i l_{\delta_i}$. By bilinearity of φ, we have:

$$\varphi(l_{\delta_j}, \sum_i \lambda_i l_{\delta_i}) = \sum_{i=1}^n \lambda_i \varphi(l_{\delta_j}, l_{\delta_i}) \quad \forall\, j \in \{1,\ldots,n\} \tag{4.16}$$

Let δ^* be a deterministic strategy which is WEU optimal within the set of deterministic strategy. By transitivity of preferences, we have $\varphi(l_{\delta^*}, l_{\delta_i}) \geq 0$ for all i. Hence, by positivity of λ_i's, we can deduce that $\varphi(l_{\delta^*}, \sum_i \lambda_i l_{\delta_i}) \geq 0$ and therefore δ^* is preferred to any randomized or mixed strategy. Thus, by Proposition 9, δ^* is an optimal strategy for problem $(\mathscr{R})$.

Conversely, consider an optimal strategy $\delta_r \in \Delta_r$ for problem $(\mathscr{R})$. Let $l_{\delta_r} = \sum_i \lambda_i l_{\delta_i}$. In Equation 4.16, by optimality of δ_r, we should have $\sum_{i=1}^n \lambda_i \varphi(l_\delta, l_{\delta_i}) \leq 0$ for all $\delta \in \Delta$. Let δ^* be an optimal deterministic strategy. By transitivity of preferences, we have that $\varphi(l_{\delta^*}, l_{\delta_i}) \geq 0$ for all i, and therefore $\lambda_i \varphi(l_{\delta^*}, l_{\delta_i}) = 0$ for all i. Consequently, $\lambda_i > 0 \Rightarrow \varphi(l_{\delta_j}, l_{\delta_i}) = 0$. Stated differently, it means that all deterministic strategies δ_i such that $\lambda_i > 0$ are optimal for problem $(\mathscr{D})$. These results are summarized in Theorem 16.

[1] We suppose here that the set of lotteries that can be defined over $\mathscr{X}$ has both maximally preferred and minimally preferred elements, or has neither, which is not a very restrictive assumption.

Theorem 16. *For any sequential decision problem under risk represented as a decision tree $\mathscr{T}$, a WEU-optimal strategy exists as a deterministic strategy. Stated differently:*

$$\min_{\delta\in\Delta_r}\max_{\delta'\in\Delta}\varphi(l_{\delta'},l_\delta)=\min_{\delta\in\Delta}\max_{\delta'\in\Delta}\varphi(l_{\delta'},l_\delta)$$

if φ is consistent with the WEU model. As a consequence, in that case, solving optimization problem $(\mathscr{D})$ solves optimization problem $(\mathscr{R})$. Furthermore, given an optimal mixed strategy, its support consists only of optimal deterministic strategies.

We now show how to compute a WEU optimal deterministic strategy. As shown by Equation 4.15, solving $(\mathscr{D})$ with the WEU model reduces to the optimization problem $\max_{\delta\in\Delta} u(l_\delta)/w(l_\delta)$. This problem can be solved by using program $\mathscr{P}_{\text{SSB}}$, without any supplementary calculation. Indeed, note that at optimum of program $\mathscr{P}_{\text{SSB}}$, variables $p_{\text{N}}, \text{N}\in\mathscr{N}$ describe an optimal randomized strategy δ_r (which therefore maximizes the ratio $u(l_\delta)/w(l_\delta)$) and variables q_{N} describe a deterministic strategy δ which is a best response to δ_r. It is then easy to realize that the deterministic strategy δ must be a WEU optimal strategy. Indeed, by definition of best response:

$$\varphi(l_\delta,l_{\delta_r})\geq\varphi(l_{\delta_r},l_{\delta_r})=0.$$

By replacing φ by its expression (i.e., $\varphi(x,y)=u(x)w(y)-u(y)w(x)$), we obtain:

$$\frac{u(l_\delta)}{w(l_\delta)}\geq\frac{u(l_{\delta_r})}{w(l_{\delta_r})}=\max_{\delta'\in\Delta_r}\frac{u(l_{\delta'})}{w(l_{\delta'})}.$$

Thus, strategy δ is a WEU optimal deterministic strategy.

As solving the linear program $\mathscr{P}_{\text{SSB}}$ is a polynomial time problem, we can conclude that solving $(\mathscr{D})$ with the WEU model is also a polynomial time problem.

Theorem 17. *Finding a WEU optimal deterministic strategy (solving both $(\mathscr{D})$ and $(\mathscr{R})$) is a problem of polynomial complexity.*

However, more efficient solution methods can be designed by resorting to fractional programming. Indeed, as functions u and w are linear with respect to mixtures of probabilities, the determination of a WEU optimal deterministic strategy can be identified as a fractional version of a standard decision tree problem. This fractional programming problem can be written as the following mathematical program:

$$\max_{p_{\text{N}}:\text{N}\in\mathscr{N}}\ \frac{\sum_{\text{T}\in\mathscr{N}_{\text{T}}}\chi_{\text{T}}\text{P(T)}u(o(\text{T}))}{\sum_{\text{T}\in\mathscr{N}_{\text{T}}}\chi_{\text{T}}\text{P(T)}w(o(\text{T}))} \tag{4.17}$$

$$\chi_{\text{N}_r}=1 \tag{4.18}$$

$$\chi_{\text{D}}=\sum_{\text{N}\in\mathscr{S}(\text{D})}\chi_{\text{N}},\forall\text{D}\in\mathscr{N}_{\text{D}} \tag{4.19}$$

$$\chi_{\text{C}}=\chi_{\text{N}},\ \forall\text{N}\in\mathscr{S}(\text{C}),\ \forall\text{C}\in\mathscr{N}_{\text{C}} \tag{4.20}$$

$$\chi_{\text{N}}\in\{0,1\},\forall\text{N}\in\mathscr{N} \tag{4.21}$$

where variables $\chi_{\text{N}},\forall\text{N}\in\mathscr{N}$ describe all deterministic strategies and where the value P(T) is the product of all probabilities on the path from the root node N_r to the leaf T.

As a consequence, solution methods from fractional programming can be used (these methods are described in section 3.2 on page 90), as Megiddo's method [Megiddo, 1979] and Dinkelbach's method [Dinkelbach, 1967]. Both methods return a WEU optimal strategy which is deterministic. In both methods, the oracle $\mathscr{O}$ consists in solving the decision tree with a utility function V^λ of the form $\text{V}^\lambda(x)\rightarrow u(x)-\lambda w(x)$. A run of the oracle

can be performed in linear time by rolling back the decision tree with utility function V^{λ}. Megiddo's method for the rolling back algorithm relies on the redefined *sum*, *comparison*, and *multiplication by a scalar* operations described on page 90.

Algorithmic complexity of the methods.

As the oracle $\mathcal{O}$ is linear time, both Megiddo's method and Dinkelbach's method are polynomial in the number of decision nodes of the decision tree. This is another proof that solving a decision tree w.r.t. to the WEU model is a polynomial time problem. Let us precise the complexity of Megiddo's method for solving decision trees with the WEU model. As the number of comparisons of lotteries in the rolling back method is upper bounded by $|\mathcal{N}_C| + |\mathcal{N}_T|$, the number of standard rolling back methods (in $O(|\mathcal{N}|)$) launched by Megiddo's method to find a WEU optimal strategy is also upper bounded by $|\mathcal{N}_C| + |\mathcal{N}_T|$. The complexity of the method is therefore $O(|\mathcal{N}^2|)$.

Initialization of the methods.

Megiddo's method maintains two parameters λ_l and λ_u that represent a lower and an upper bound on the optimal value of the fractional optimization problem. The initial values of variables λ_l and λ_u may impact the performance of the method. A "good" initialization can be obtained by computing an optimal deterministic strategy δ_u (resp. δ_w) maximizing (resp. minimizing) EU with utility function u (resp. w). Variable λ_l can then be set to $u(\delta_u)/w(\delta_u)$ and variable λ_u to $\max\{0, u(\delta_u)/w(\delta_w)\}$.

Dinkelbach's method works by generating a sequence of strategies $(\delta_t)_{t\in\mathbb{N}}$ such that $\delta_{t+1} \succ \delta_t$. The initial strategy δ_0 can be any feasible strategy in Δ. A "good" choice of δ_0 may increase the efficiency of the approach. For instance, δ_0 can be chosen as an optimal EU strategy according to utility function $V_0 = u$.

4.2.5 Numerical Tests

We now present the results of our numerical test[2].

Random instances. Our tests were performed on complete binary decision trees. The height of these decision trees varies from 5 to 20 (21 to 349525 decision nodes), with an alternation of decision nodes and chance nodes. At every terminal node T, outcome $o(T)$ is uniformly drawn in interval $[0,500]$. The mapping u (resp. w) from $[0,500]$ to $[0,100]$ (resp. $[1,100]$) is randomly generated while enforcing nondecreasingness (resp. nonincreasingness). Imposing these monotonicity conditions on u and w ensures that $v = u/w$ is a nondecreasing function of outcomes. Table 4.2 presents the performances of Dinkelbach's method and Megiddo's method (denoted by DIN and MEG from now on) as the height of the decision tree increases. For each height level, we give the average computation time over 50 instances in seconds as well as the average number of times the subroutine rolling back the decision tree for a given utility function V_λ was used. Initialization of both methods where performed as described in the previous section.

The results show that both methods are very efficient, solving trees of height 20 in less than 1 sec (resp. 4 sec) for DIN (resp. MEG). Method DIN seems to perform best as it requires to launch less sub-routine algorithms.

To measure the impact of the quality of the initialization, we computed the same results for these two methods launched with weak initializations. They are denoted by DIN^{WI} and MEG^{WI} in the table (WI for weak initialization). While DIN^{WI} is initialized with

[2]All methods were implemented in C++ using Gurobi version 5.6.3 to solve the LPs. All times are wall-clocked on a 2.4 GHz Intel Core i5 machine with 8G main memory.

	height	5	10	15	20
DIN	time (sec)	<0.001	<0.001	0.019	0.91
	nbsr	2.2	2.98	3.1	3.66
DIN^{WI}	time (sec)	<0.001	<0.001	0.019	0.789
	nbsr	3.04	3.72	4.06	4.24
MEG	time (sec)	<0.001	<0.001	0.039	3.572
	nbsr	2.02	2.74	5.52	14.38
MEG^{WI}	time (sec)	<0.001	0.001	0.074	4.77
	nbsr	0.52	6.02	11.02	19.16

Table 4.2: Computation times for WEU, and number of sub-routines launched (nbsr in the table).

a random initial strategy δ_0, MEG^{WI} starts with the loose lower and upper bounds $\lambda_l = 0$ and $\lambda_u = u(500)/w(0)$.

We observe that a weak initialization does not seem to highly impact the performance of the two methods (DIN^{WI} even achieves better performances than DIN on decision trees of height 20).

We also tested our algorithms determining an SSB optimal randomized strategy on the same trees, with SSB utility function φ defined by $\varphi(x,y) = 1$ (resp. 0,-1) if $x > y$ (resp. $x = y$, $x < y$). Note that (as illustrated by Example 44), this function may model intransitive preferences. The results are given in Table 4.3 for our LP $\mathscr{P}_{\text{SSB}}$, our double oracle algorithm (denoted by DO) and our polynomial variant of the double oracle algorithm (denoted by DO$_{\text{P}}$). The computation times are averaged over 50 instances. The computation times of $\mathscr{P}_{\text{SSB}}$ exponentially increase with the height of the tree, raising from 2.239 seconds for height 10 to 53.741 seconds for height 12. This is not surprising as $|\mathscr{N}|$ (and thus the number of variables in $\mathscr{P}_{\text{SSB}}$) exponentially increases with the height of the tree. For this reason, $\mathscr{P}_{\text{SSB}}$ was not run for decision trees with height superior to 12. Methods DO and DO$_{\text{P}}$ seem much more scalable, solving decision trees with height going up to 16 in less than 24 (resp. 3) seconds. On these decision trees, DO$_{\text{P}}$ outperformed the two other methods.

height	6	8	10	12	14	16
$\mathscr{P}_{\text{SSB}}$	0.007	0.115	2.239	53.741	-	-
DO	0.002	0.004	0.020	0.160	1.742	23.053
DO$_{\text{P}}$	0.001	0.004	0.012	0.059	0.260	2.003

Table 4.3: Computation times for SSB.

Application to Who wants to be a millionaire? *Who wants to be a millionaire?* is a popular game show, were a contestant must answer a sequence of 15 questions with 4 possible answers. The questions enable the contestant to earn increasing sums of money but are also of increasing difficulty. If the answer given is wrong then the contestant leaves the game with no money except what was earned at the last guarantee point (questions

		model 1	model 2
DIN	time (sec)	0.017	73.307
	nbsr	2.5	2.4
MEG	time (sec)	0.036	104.1
	nbsr	4.1	2.6

Table 4.4: Computation times and number of sub-routines (nbsr in the table) launched for methods DIN and MEG .

5 and 10). At each question, the contestant can also decide to stop. She then leaves with the money won so far. We used the Spanish version of the game modeled by Perea and Puerto [2007], where the monetary values of the questions range from 150€ (question 1) to 300000€ (question 15). Lastly, the contestant has three lifelines that can be used once during the game: Phone a friend, 50:50, and Ask the audience. We tested our solution methods for two models of this game. While model 1 (which contains around 70000 nodes) is the original one proposed by Perea and Puerto, model 2 (which contains around 80 million nodes) is a refinement proposed by Jeantet and Spanjaard [2008] to take into account the fact that the candidate may or may not (with a certain probability) know the answer to a given question. The results are averaged over 20 instances where functions of the form $u = x^{\alpha}$ and $w = 300001^{\beta} - x^{\beta}$ were used. For each instance, parameters α and β were uniformly sampled in (0,1). Our numerical results are given in Table 4.4. We observe that both methods DIN and MEG are efficient and that they require very few calls to their sub-routine.

Let us summarize the results of this section. We have seen that an SSB optimal strategy can always be found as a randomized strategy which can be determined in polynomial time. However, we also proved that finding an optimal strategy among deterministic strategies is an NP-hard problem. We have formulated the optimization of an SSB utility function in decision trees as the search for a Nash equilibrium in a specific zero-sum two-player symmetric game. This analysis enabled us to design a linear programming approach, a double oracle approach as well as a cutting plane approach. Regarding the WEU model, we have seen that a WEU optimal strategy can always be found as a deterministic strategy, which can be determined in polynomial time with fractional programming methods. Lastly, we gave some numerical results showing the practicality of the proposed approaches.

We now turn to another representation of sequential decision problems, namely finite horizon MDPs, and show how our results can be extended to this more compact representation.

4.3 Optimizing the SSB and WEU Models in Finite Horizon MDPs

In this section, we study the optimization of the SSB and WEU models in finite horizon MDPs. In the literature, several other non-standard decision criteria have also been investigated with this representation of sequential decision problems under risk. We begin by briefly describing these previous works.

4.3.1 Related Work

In the MDP literature, the three criteria *expected discounted sum of rewards*, *expected total rewards* or *expected average rewards* are prominent. However, many other models have been considered [Boussard *et al.*, 2010]. One important reason to investigate other decision criteria is that the prominent criteria are not *risk sensitive*. Indeed, all three of them are based on expected values. Thus, they can compensate very low values by very high ones and favor solutions for which the variance is very high. To address this problem, numerous solutions have been investigated. We mention some of them below.

Variance-penalized criteria. Firstly, one can add a term in the decision criterion or a constraint which penalizes policies with high variance. This solution was investigated by White [1987] and more recently by Prashanth and Ghavamzadeh [2013] and Mannor and Tsitsiklis [2011], who provided algorithms for mean-variance formulations of MDPs (i.e., maximizing the expected cumulated reward while ensuring an upper bound on the variance). Additionally, Filar *et al.* [1989] investigated decision criteria that are variance-penalized versions of the standard ones. They formulated the obtained optimization problem as a non-linear program.

Threshold objective or quantile criteria. Other criteria that favor low risk policies can be found. For instance, one can try to optimize the probability that the total (discounted) reward exceeds a given threshold. This optimization problem was investigated by numerous researchers [Bouakiz and Kebir, 1995; Fan *et al.*, 2005; Hou *et al.*, 2014; Ohtsubo and Toyonaga, 2002; White, 1993; Wu and Lin, 1999; Yu *et al.*, 1998]. Alternatively, recent works on MDPs and reinforcement learning have considered conditional Value-at-risk (CVaR), a criterion related to quantiles, as a risk measure. Bäuerle and Ott [2011] proved the existence of deterministic wealth-Markovian[3] policies optimal with respect to CVaR. Chow and Ghavamzadeh [2014] proposed gradient-based algorithms for CVaR optimization. In contrast, Borkar and Jain [2014] used CVaR in inequality constraints instead of as objective function.

Decision theoretic criteria. Lastly, researchers tried to use risk-sensitive models from decision theory as EU. The most popular type of utility functions for MDPs is the exponential one. Indeed, it is well known that the standard methods to solve MDPs can easily be adapted to this setting [Koenig and Simmons, 1994]. If another type of utility function is used, the problem becomes more difficult and a state space augmentation is required [Iwamoto, 2004; Spanjaard and Weng, 2013]. This was first noticed by White [1987], who considered EU of the total cumulated rewards as decision criterion for MDPs. This problem was thoroughly investigated in the thesis of Liu [Liu and Koenig, 2006, 2008; Liu, 2005]. Researchers extended this problem to the case of constrained-MDP [Ermon *et al.*, 2012; Kadota *et al.*, 2006].

The work we present in this section is closest to this last line of research about decision theoretic criteria. As the SSB model extends the EU model, this work extends the work on risk-sensitive MDPs with EU while allowing for more general preferences on the set of policies.

[3] a wealth-Markovian policy is a Markovian policy in the MDP where the state space is augmented by the cumulated rewards obtained so far. This notion will be further explained in this chapter.

4.3.2 Optimizing the SSB Criterion in the Randomized Setting

We now show how to extend the results of section 4.2 when the sequential decision problem under risk is represented as a finite horizon MDP (this representation of sequential decision problems is detailed in subsection 2.2.2). Decision trees can be seen as specific finite horizon MDPs. As a consequence, determining an optimal policy among the deterministic policies is an NP-hard problem (Theorem 14). Thus, in this section, we will focus on the randomized setting.

In this section, we model the problem of computing an SSB optimal policy in a finite horizon MDP as the search for a Nash equilibrium in a zero-sum two-player symmetric game defined by the MDP and the SSB utility function φ. The set of pure strategies in this game is the set of deterministic policies after transforming the given MDP in an "augmented" MDP [Liu, 2005], and the payoff function is inferred from the SSB utility function. The set of deterministic policies is combinatorial in nature, which makes impractical the generation of the whole payoff matrix of the game. For this reason, to solve the game, we resort to the double oracle method presented in subsection 3.1.3 (on page 85).

We now specify the setting of MDPs that we will study in this section.

Background. In this section, we study finite horizon MDPs with finite state and action spaces. An MDP is formally defined by $\mathscr{M} = (\mathrm{T}, \mathscr{S}, \mathscr{A}, \mathscr{P}, c)$ where:

- T, a finite positive integer, is the time horizon;
- $\mathscr{S}$ is a finite collection of states, one of which is designated as the initial state and is denoted by s_0;
- $\mathscr{A} = \{\mathscr{A}_s | s \in \mathscr{S}\}$ is a collection of finite sets of possible actions, one set for each state;
- $\mathscr{P} = \{\mathscr{P}_t | t = 0, \ldots, \mathrm{T}-1\}$ is a collection of transition probabilities where $\mathscr{P}_t(s'|s,a)$ is the probability that the state at time step $t+1$ is s' given that the state at time step t was s and that we have performed action a;
- $c = \{c_t | t = 0, \ldots \mathrm{T}-1\}$ is a collection of reward functions where $c_t(s,a,s')$ is the reward obtained if the state at time step $t+1$ is s' given that the state at time step t was s and that we have performed action a.

To illustrate our notations on a voluntarily simple sequential decision problem, we (artificially) modify the Rowett dice paradox recalled in Example 44.

Example 45 (One-agent sequential variant of Rowett dice)**.** *An agent has first to choose whether she wants to throw (action a_1) or not (action a'_1) die* A*; if she does not throw die* A*, she needs to choose between die* B *(action a_2) or* C *(action a_3). Whatever die is chosen, we distinguish two cases: success (state s_3) if one of the advantageous faces of the die is rolled (e.g., a face 4 for die* A*), or failure (state s_2) otherwise.*

The decision problem can be modeled by the MDP represented in Figure 4.3 with $\mathrm{T} = 2, \mathscr{S} = \{s_1, s'_1, s_2, s_3\}, \mathscr{A} = \{\{a_1, a'_1\}, \{a_2, a_3\}, \{a_4\}, \{a_5\}\}$ *and where c_1, c_2 and c_3 are the chance nodes induced by $\mathscr{P}$. The values $c_t(s,a,s') | \mathscr{P}_t(s'|s,a)$ (that do not depend on t in this example) are shown along the edges. In this MDP, the reward functions $c_t(s,a,s')$ take value in* $\{0, \cdots, 6\}$.

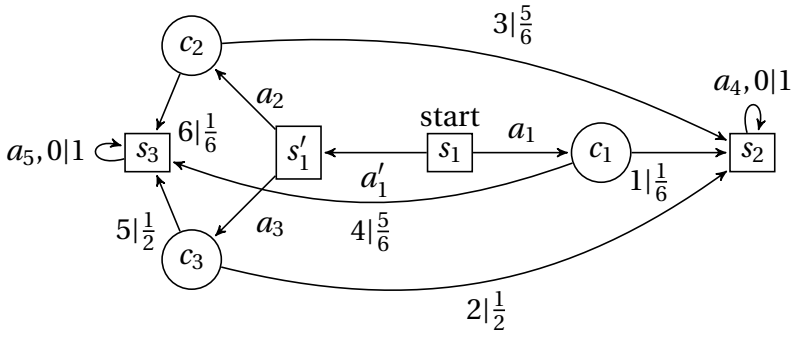


Figure 4.3: The MDP in Example 45.

During one episode of the sequential decision problem, the agent realizes a T-history, i.e., a sequence of T state-action pairs, $h_T = (s_0, a_0, s_1, \ldots, s_{T-1}, a_{T-1}, s_T)$. Each episode is valued by the wealth level (i.e., sum of cumulated rewards) of the agent at the end of the episode $\rho_T = \sum_{i=0}^{T-1} c_i(s_i, a_i, s_{i+1})$. We will denote by $\mathscr{W}$ the finite set of possible wealth levels that can be obtained in the MDP and by $\mathscr{W}_T \subseteq \mathscr{W}$ the set of possible final wealth levels. The values in $\mathscr{W}_T$ are the possible outcomes of the finite horizon MDP.

The goal of the agent is to determine a policy, i.e., a procedure to select an action in each possible situation. We recall that a policy π at an horizon T is a sequence of T decision rules $(\delta_0, \ldots, \delta_{T-1})$. A decision rule δ_t is a function which prescribes the action that the agent should perform at time step t according to the current situation. A decision rule can be *history-dependent*, meaning that it takes as argument the entire history generated so far, or *Markovian* if it only takes as argument the current state. Moreover, a decision rule is either *deterministic* if, given an argument, it always selects the same action, or *randomized* if it prescribes a probability distribution over possible actions. A policy can be *history-dependent*, *Markovian*, *deterministic* or *randomized* according to the type of its decision rules.

We use the notations of Table 4.5 for the sets of the different types of policies. Importantly, given a set $\Pi = \{\pi_1, \pi_2, \ldots\}$ of policies, we define an enlarged set $\widetilde{\Pi}$ of policies, that denotes the set consisting of mixtures of policies, i.e., $\widetilde{\Pi} = \{\tilde{\pi} = (\pi_1, \alpha_1; \pi_2, \alpha_2; \ldots) : \sum_i \alpha_i = 1, \alpha_i \geq 0\}$, where $\tilde{\pi}$ is the *mixed policy* that randomly selects policy π_i with probability α_i and follows it thereafter.

	Markovian	history-dependent
deterministic	Π_s^t	Π_h
randomized	$\Pi_{s,r}^t$	$\Pi_{h,r}$

Table 4.5: Policy Notations

Each policy π induces a lottery l_π over possible final wealth levels and the SSB utility function φ of the agent defines a preference relation on lotteries over $\mathscr{W}_T$. Hence, comparing policies according to the SSB model reduces to comparing their induced lotteries over $\mathscr{W}_T$:

$$\pi \succsim \pi' \Leftrightarrow \varphi(l_\pi, l_{\pi'}) \geq 0 \tag{4.22}$$

$$\varphi(l_\pi, l_{\pi'}) = \sum_{\rho \in \mathscr{W}_T} \sum_{\rho' \in \mathscr{W}_T} l_\pi(\rho) l_{\pi'}(\rho') \varphi(\rho, \rho') \tag{4.23}$$

where $l_\pi(x)$ denotes the probability of obtaining a final wealth level of value x when applying policy π. In our setting, the preference relation $\succsim$ induces the following optimization

problem:

$$(\mathscr{R}): \min_{\pi \in \widetilde{\Pi}_{h,r}} \max_{\pi' \in \widetilde{\Pi}_{h,r}} \varphi(l_{\pi'}, l_{\pi}). \tag{4.24}$$

Put another way, a policy is evaluated by the maximal intensity with which another policy is preferred to it, and we look for the policy with minimal evaluation among all policies in $\widetilde{\Pi}_{h,r}$.

We now investigate if an optimal policy can always be found in a more specific set of policies. For instance, with the total cumulated reward criterion an optimal policy can always be found in Π^t_s. Could this property also hold for the SSB criterion? The answer is no and in fact an optimal SSB policy does not always exists in the sets of policies $\widetilde{\Pi}^t_s$ and $\Pi^t_{s,r}$. One reason for this negative result is that, as the preference relation over policies depends in a complex manner on wealth levels, optimal policies will also depend on them. We support this claim in the following example:

Example 46. *Consider the finite horizon MDP represented in Figure 4.4 with* $\mathrm{T} = 3$, $\mathscr{S} = \{s_1, \ldots, s_8\}$, $\mathscr{A} = \{\{a_1\}, \{a_2\}, \{a_3\}, \{a_4, a_5\}, \{a_6\}, \{a_7\}, \{a_8\}, \{a_9\}\}$ *and where* c_1, c_2 *and* c_3 *are the chance nodes induced by* $\mathscr{P}$. *The values* $c_t(s,a,s')|\mathscr{P}_t(s'|s,a)$ *(that do not depend on* t *in this example) are shown along the edges. The set of possible rewards that can be obtained in this MDP is* $\{0,2,4,5,7,9\}$ *and the set of possible final wealth levels* $\mathscr{W}_\mathrm{T}$ *is* $\{2,5,6,9,11,12,14\}$.

For any deterministic Markovian policy, the induced lottery over final wealth levels is here completely determined by the choice of the action to realize in state s_4 *(i.e.,* a_4 *or* a_5*) at time step 2. We denote by* π_1 *(resp.* π_2*) a policy in* Π^t_s *choosing* a_4 *(resp.* a_5*) in such situation. In this example, for any randomized or mixed Markovian policy, there exists a mixture of* π_1 *and* π_2 *that yields the same lottery over* $\mathscr{W}_\mathrm{T}$. *The lotteries over* $\mathscr{W}_\mathrm{T}$ *induced by these two policies are* $l_{\pi_1} = (6, 0.2; 9, 0.5; 12, 0.3)$ *and* $l_{\pi_2} = (2, 0.25; 5, 0.25; 11, 0.25; 14, 0.25)$. *Now consider the wealth-dependent policy* π_ρ *which chooses* a_4 *in* s_4 *when the wealth level in this state is* 5, *and* a_5 *otherwise. The lottery over* $\mathscr{W}_\mathrm{T}$ *induced by* π_ρ *is* $l_{\pi_\rho} = (2, 0.25; 9, 0.2; 11, 0.25; 12, 0.3)$.

We wish to determine a policy which maximizes the probability $\mathrm{G}_\pi(11)$ *of yielding a final wealth level greater than or equal to 11. We recall that this decision criterion is a special transitive case of SSB with* φ *defined by* $\varphi(x,y) = \mathbb{1}_{x \geq 11} - \mathbb{1}_{y \geq 11}$ *where* $\mathbb{1}_{x \geq 11}$ *equals 1 if* $x \geq 11$ *and 0 otherwise. Therefore, with this* φ *function, determining an SSB optimal policy reduces to determining a policy* π *maximizing* $\mathrm{G}_\pi(11)$.

We have, $\mathrm{G}_{\pi_1}(11) = 0.3$, $\mathrm{G}_{\pi_2}(11) = 0.5$ *and* $\mathrm{G}_{\pi_\rho}(11) = 0.55$. *Hence, by linearity of* $\mathrm{G}_\pi(11)$ *with respect to mixtures of policies,* π_ρ *is strictly better than all policies in* $\widetilde{\Pi}^t_s$ *and* $\Pi^t_{s,r}$.

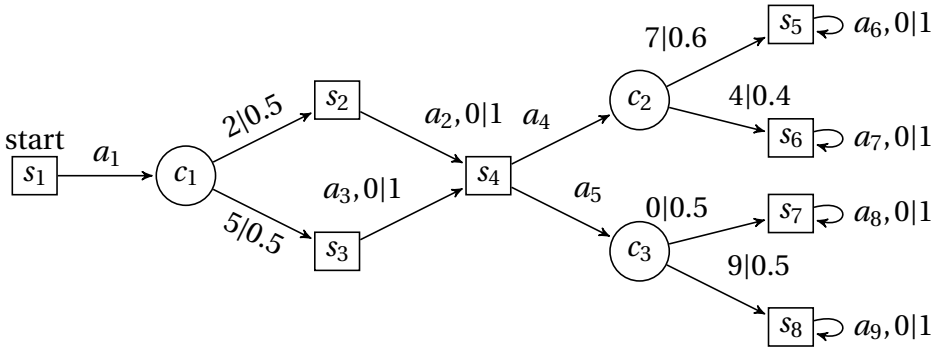


Figure 4.4: The MDP in Example 46.

This example proves that, in general the optimal policies will depend on the history generated so far, at least through the wealth levels. For this reason, we incorporate the wealth levels in the state space. Following Liu and Koenig [2008], we transform the given

MDP into an augmented MDP whose states are pairs (s,ρ) where s is a state of the original MDP and $\rho \in \mathscr{W}$ a wealth level attainable by executing actions in the given MDP.

Example 47. *We illustrate the notion of augmented MDP on our modified Rowett dice problem, represented in Figure 4.3. The augmented MDP is represented in Figure 4.5.*

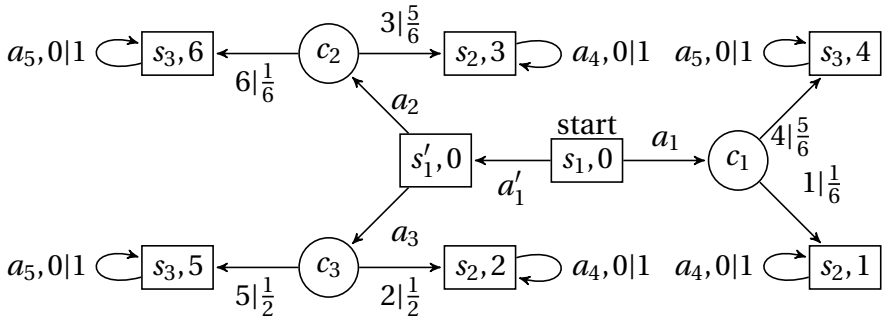


Figure 4.5: The augmented MDP in Example 47.

To avoid any confusion, we denote by $\overline{s} = (s,\rho)$ a state in the augmented MDP. For instance, $\Pi^t_{\overline{s}}$ (resp. $\Pi^t_{\overline{s},r}$) denotes the set of deterministic (resp. randomized) Markovian policies in the augmented MDP. We will say that these policies are wealth-Markovian.

Definition 38. *A policy is said to be* wealth-Markovian *if its decision rules are functions of both the current state and the current wealth level. Stated differently, these policies are Markovian in the augmented MDP.*

Wealth-Markovian policies make it possible to find a compromise between history-dependent policies and Markovian policies as illustrated in the lattice of Figure 4.6, where an arrow goes from a more general class of policies to a more specific class of policies.

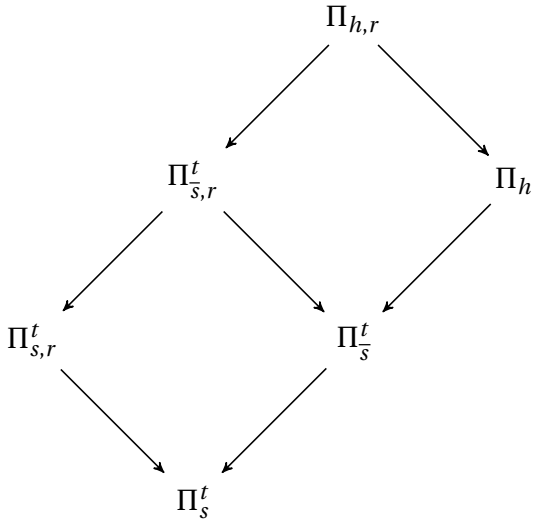


Figure 4.6: The relationships between wealth-Markovian policies and the other classes of policies.

Let $\overline{\mathscr{S}}_{\mathrm{T}} \subseteq \mathscr{S} \times \mathscr{W}_{\mathrm{T}}$ be the set of final states in the augmented MDP. We recall that the preference relation over policies only depends on the lotteries induced by policies over final wealth levels. Now, in the augmented MDP, these lotteries only depend on the lotteries induced over final states. These lotteries can be computed in the following way. Recall

that in any MDP, a policy induces a lottery over episodes and a fortiori over final states. The lottery over final states induced by a policy can be easily computed by a standard dynamic programming procedure. In the augmented MDP, we denote by $l_\pi^{\mathtt{fs}}$ the lottery over final states induced by policy π, such that $l_\pi^{\mathtt{fs}}(\overline{s})$ is the probability of being in the state $\overline{s}$ at the end of an episode with policy π. Assuming a lottery over final states $l_\pi^{\mathtt{fs}}$ has been computed, then the associated probability distribution l_π over final wealth levels can simply be obtained by marginalization:

$$l_\pi(\rho) = \sum_{\overline{s}=(s,\rho')\in\overline{\mathscr{S}}_T:\rho'=\rho} l_\pi^{\mathtt{fs}}(\overline{s}).$$

Collins and McNamara [1998] showed that, provided we are only interested in the probabilities of the final states (in the augmented MDP), it is not a restrictive assumption to focus on mixed policies in $\widetilde{\Pi}^t_{\overline{s}}$ or on randomized policies in $\Pi^t_{\overline{s},r}$ (this is similar to Theorem 12). Indeed, they showed that for any mixed policy $\tilde{\pi}$ in $\widetilde{\Pi}_{h,r}$ there exist a mixed policy $\tilde{\pi}'$ in $\widetilde{\Pi}^t_{\overline{s}}$ and a randomized policy π_r in $\Pi^t_{\overline{s},r}$ such that $l_{\tilde{\pi}}^{\mathtt{fs}} = l_{\tilde{\pi}'}^{\mathtt{fs}}$ and $l_{\tilde{\pi}}^{\mathtt{fs}} = l_{\pi_r}^{\mathtt{fs}}$.

In particular, a randomized policy π_r corresponding to any mixed policy $\tilde{\pi} = (\pi_1, \alpha_1; \pi_2, \alpha_2; \ldots) \in \widetilde{\Pi}_{h,r}$ is obtained by using the following equation [Strauch and Veinott, 1966]:

$$\mathbb{P}(a_t = a | s_t = s, \pi) = \frac{\sum_i \alpha_i \mathbb{P}(s_t = s, a_t = a | \pi_i)}{\sum_{a'\in\mathscr{A}_s}(\sum_k \alpha_k \mathbb{P}(s_t = s, a_t = a' | \pi_k))}$$

Consequently, in the remainder of the section, we focus either on policies in $\widetilde{\Pi}^t_{\overline{s}}$ or on policies in $\Pi^t_{\overline{s},r}$ in the augmented MDP as:

$$\begin{aligned}\min_{\pi\in\widetilde{\Pi}_{h,r}} \max_{\pi'\in\widetilde{\Pi}_{h,r}} \varphi(l_{\pi'}, l_\pi) &= \min_{\pi\in\Pi^t_{\overline{s},r}} \max_{\pi'\in\Pi^t_{\overline{s},r}} \varphi(l_{\pi'}, l_\pi) \\ &= \min_{\pi\in\widetilde{\Pi}^t_{\overline{s}}} \max_{\pi'\in\widetilde{\Pi}^t_{\overline{s}}} \varphi(l_{\pi'}, l_\pi). \end{aligned} \tag{4.25}$$

Determining an optimal policy for the SSB criterion is not straightforward as the SSB criterion does not respect Bellman's principle of optimality. Therefore a dynamic programming procedure cannot be used directly and we turn to a game-theoretic analysis of the problem of identifying an SSB-optimal policy from the initial state.

A Game Theoretic Analysis. When an MDP and a time step T are fixed, Equations 4.22 and 4.23 induce a zero-sum two-player symmetric game where the set of pure strategies is the set of wealth-Markovian deterministic policies (i.e., $\Pi^t_{\overline{s}}$). Finding a Nash equilibrium of this game solves the optimization problem ($\mathscr{R}$) recalled in Equation 4.25. In this game, each player $i \in \{1,2\}$ chooses simultaneously a strategy π_i which can be deterministic (i.e., $\pi_i \in \Pi^t_{\overline{s}}$) or mixed (i.e., $\pi_i \in \widetilde{\Pi}^t_{\overline{s}}$). The resulting payoff is then given by $\varphi(l_{\pi_1}, l_{\pi_2})$.

In a zero-sum two-player symmetric game, it is well-known that there exists a symmetric Nash equilibrium. The following holds for such a Nash equilibrium (π^*, π^*):

$$\begin{aligned}\forall \pi \in \widetilde{\Pi}^t_{\overline{s}},\ \varphi(l_{\pi^*}, l_\pi) \geq \varphi(l_{\pi^*}, l_{\pi^*}) = 0,\\ \text{which implies } \forall \pi \in \widetilde{\Pi}_{h,r},\ \varphi(l_{\pi^*}, l_\pi) \geq 0.\end{aligned}$$

In fact, this property will hold for *any* optimal policy in $\widetilde{\Pi}_{h,r}$ for problem ($\mathscr{R}$).

Proposition 10. *A policy $\pi \in \widetilde{\Pi}_{h,r}$ is an optimal solution of problem ($\mathscr{R}$) iff:*

$$\forall \pi' \in \widetilde{\Pi}_{h,r},\ \varphi(l_\pi, l_{\pi'}) \geq 0.$$

Proof. Note that as $\forall \pi \in \widetilde{\Pi}_{h,r}, \varphi(l_\pi, l_\pi) = 0$, the optimal value of problem $(\mathscr{R})$ is lower bounded by 0. We have just seen that this value was reached by any (possibly mixed) policy $\tilde{\pi} \in \widetilde{\Pi}^t_s$ such that $(\tilde{\pi}, \tilde{\pi})$ is a symmetric Nash equilibrium of the game induced by the finite horizon MDP and φ. Therefore, the optimal value of $(\mathscr{R})$ is 0 and a policy $\pi \in \widetilde{\Pi}_{h,r}$ is an optimal solution of problem $(\mathscr{R})$ iff:

$$\begin{aligned} &\max_{\pi' \in \widetilde{\Pi}_{h,r}} \varphi(l_{\pi'}, l_\pi) = 0, \\ \Leftrightarrow\ &\forall \pi' \in \widetilde{\Pi}_{h,r},\ \varphi(l_\pi, l_{\pi'}) \geq \varphi(l_\pi, l_\pi) = 0. \end{aligned}$$

□

Hence, an SSB optimal policy has the desirable property of being preferred to all other policies. We aim to compute a Nash equilibrium (π^*, π^*) of the game on policies characterized by payoff function φ since strategy π^* will be an SSB optimal policy in the MDP. However, note that the size of Π^t_s is combinatorial in nature, which prohibits solving the game directly. We address this issue by resorting to the double oracle method [McMahan *et al.*, 2003] described in subsection 3.1.3 (on page 85).

To use this double oracle method, a best response oracle $\mathscr{O}$ needs to be instantiated. This procedure should find a policy which is a best response to a fixed policy $\tilde{\pi}$. This problem can be seen as the problem of solving the MDP according to EU with utility function $V_{\tilde{\pi}}$ defined by $V_{\tilde{\pi}}(x) = \varphi(x, l_{\tilde{\pi}})$ and amounts to taking $\tilde{\pi}$ as a reference point.

Solving this problem can be performed by using the following reward function, denoted by $c^{\tilde{\pi}}_t$, in the augmented MDP:

$$\begin{aligned} c^{\tilde{\pi}}_t((s,\rho), a, (s',\rho')) &= 0 \text{ for } t < T-1 \\ c^{\tilde{\pi}}_{T-1}((s,\rho), a, (s',\rho')) &= \varphi(\rho', l_{\tilde{\pi}}) \end{aligned}$$

Indeed, using this reward function, maximizing the standard expectation of total reward criterion is equivalent to maximizing $\varphi(l_\pi, l_{\tilde{\pi}})$. A policy maximizing $\varphi(l_\pi, l_{\tilde{\pi}})$ can therefore be found by a standard backward induction in the augmented MDP endowed with the reward function $c^{\tilde{\pi}}_t$. It returns a deterministic wealth-Markovian policy (i.e., a policy in Π^t_s).

Example 48. *Coming back to the sequential Rowett dice example, we show how to find a best response to the policy* $\tilde{\pi}$ *that chooses die* A *with probability* 1. *Assume that one uses the probabilistic dominance criterion, i.e.,* $\varphi(x,y) = 1$ *(resp.* $0, -1$*) if* $x > y$*, (resp.* $x = y$*,* $x < y$*). In this example,* $\mathscr{W} = \mathscr{W}_T = \{1,2,\ldots,6\}$ *and* $l_{\tilde{\pi}} = (1, \frac{1}{6}; 2,0; 3,0; 4, \frac{5}{6}; 5,0; 6,0)$*. To determine the reward function, we compute the values* $\{\varphi(\rho, \tilde{\pi}) : \rho \in \mathscr{W}_T\}$*:*

$$\begin{aligned} &\varphi(1,\tilde{\pi}) = -\frac{5}{6}, \quad \varphi(2,\tilde{\pi}) = -\frac{4}{6}, \quad \varphi(3,\tilde{\pi}) = -\frac{4}{6}, \\ &\varphi(4,\tilde{\pi}) = \frac{1}{6}, \qquad \varphi(5,\tilde{\pi}) = 1, \qquad \varphi(6,\tilde{\pi}) = 1 \end{aligned}$$

Determining $\mathscr{O}(\tilde{\pi})$ *amounts then to computing the policy maximizing the expectation of total reward in the augmented MDP represented in Figure 4.5, in which the reward function is replaced by:*

$$\begin{aligned} c^{\tilde{\pi}}_0((s,\rho), a, (s',\rho')) &= 0 \\ c^{\tilde{\pi}}_1((s,\rho), a, (s',\rho_i)) &= (-\frac{5}{6}, -\frac{4}{6}, -\frac{4}{6}, \frac{1}{6}, 1, 1).e_i \end{aligned}$$

where we denote by ρ_i *the* i^{th} *smallest value in* $\mathcal{W}_T$ *and by* e_i *the* i^{th} *canonical vector. Unsurprisingly, the policy obtained prescribes to play die* C.

A double oracle approach. In our setting, the double oracle approach is implemented by the procedure described in Algorithm 8. Note that some simplifications arise due to the fact that the game is symmetric:

- Firstly the game only maintains *one* set of strategies $\Pi' \subseteq \Pi^t_{\overline{s}}$ (for both players) and runs the oracle only *once* per iteration. The oracle $\mathscr{O}$ finds a best response π to the policy $\tilde{\pi}$ given by a symmetric Nash equilibrium $(\tilde{\pi}, \tilde{\pi})$ in the restricted game $M(\Pi', \Pi')$, where M denotes the payoff matrix of the game induced by the finite horizon MDP and by φ.

- Secondly the stopping condition is simplified to test if $\varphi(l_\pi, l_{\tilde{\pi}}) \leq 0$. Indeed, by Proposition 10, this condition (which occurs if the oracle computes a strategy which is already in Π') is sufficient to ensure that $\tilde{\pi}$ is an optimal SSB policy.

Algorithm 8: Double oracle algorithm to solve problem $(\mathscr{R})$ in finite horizon MDP

Data: Finite horizon MDP $(\mathscr{M}, T)$, singleton $\Pi' = \{\pi\}$ including an arbitrary policy $\pi \in \Pi^t_{\overline{s}}$

Result: an SSB optimal mixed policy $\tilde{\pi} \in \widetilde{\Pi}^t_{\overline{s}}$

converge = False
while *converge is False* **do**
    Find Nash equilibrium $(\tilde{\pi}, \tilde{\pi}) \in M(\Pi', \Pi')$
    Find $\pi = \mathscr{O}(\tilde{\pi}) \in \Pi^t_{\overline{s}}$
    if $\varphi(l_\pi, l_{\tilde{\pi}}) \leq 0$ **then**
        converge =True
    else
        add π to Π'
return $\tilde{\pi}$

This double oracle approach is a first method which enables to solve finite horizon MDPs with the SSB model. We now turn to a linear programming approach to give a pseudo polynomial method to solve finite horizon MDPs with SSB utility functions.

Linear Programming Approach. We now present a Linear Program (LP) to find an SSB optimal policy in $\Pi^t_{\overline{s},r}$.

The LP is compounded of two classes of constraints. The first class of constraints involves two sets of variables $p_{(s,\rho),t}$ and $p_{(s,\rho),a,t}$, describing all feasible randomized strategies $\pi_r \in \Pi^t_{\overline{s},r}$, where:

- $p_{(s,\rho),t}$ is the probability that the agent occupies the augmented state (s,ρ) at time step t with policy π_r;

- and $p_{(s,\rho),a,t}$ is the probability that the agent performs action a in the augmented state (s,ρ) at time step t with policy π_r.

These constraints are the following:

$$
\begin{aligned}
p_{(s_0,0),0} &= 1 \\
p_{(s,\rho),0} &= 0, \quad \forall (s,\rho) \in \overline{\mathscr{S}} \text{ such that } (s,\rho) \neq (s_0,0) \\
p_{(s,\rho),t} &= \sum_{a \in \mathscr{A}_s} p_{(s,\rho),a,t}, \quad \forall (s,\rho) \in \overline{\mathscr{S}}, \forall\, t \in \{0,\dots,\mathrm{T}-1\} \\
p_{(s',\rho'),t+1} &= \sum_{(s,\rho) \in \overline{\mathscr{S}}} \sum_{a \in \mathscr{A}_s} p_{(s,\rho),a,t} \overline{\mathscr{P}}_t(s',\rho'|s,\rho,a), \quad \forall (s',\rho') \in \overline{\mathscr{S}}, \forall\, t \in \{0,\dots,\mathrm{T}-1\} \\
p_{(s,\rho),t} &\geq 0, \quad \forall (s,\rho) \in \overline{\mathscr{S}}, \forall\, t \in \{0,\dots,\mathrm{T}\} \\
p_{(s,\rho),a,t} &\geq 0, \quad \forall (s,\rho) \in \overline{\mathscr{S}}, \forall\, a \in \mathscr{A}_s, \forall\, t \in \{0,\dots,\mathrm{T}-1\}
\end{aligned}
$$

where[4]:

$$
\overline{\mathscr{P}}_t(s',\rho'|s,\rho,a) = \mathscr{P}_t(s'|s,a) \mathbb{1}_{\rho' = \rho + c_t(s,a,s')}
$$

The corresponding randomized policy $\pi_r = (\delta_1,\dots,\delta_{\mathrm{T}})$ can be obtained by using the formula:

$$
\mathbb{P}(a_t = a | \overline{s}_t = (s,\rho), \pi_r) = \frac{p_{(s,\rho),a,t}}{p_{(s,\rho),t}}.
$$

The second class of constraints aims to determine $\max_{\pi \in \Pi^t_{\overline{s}}} \varphi(\pi,\pi_r)$. As indicated above, this value can be computed by backward induction with a reward function induced by policy π_r and function φ (note that the rewards are here only obtained at the final time step). This amounts to satisfy the following Bellman's equations, where $q_{(s,\rho),t}$ denotes the optimal value (i.e., maximal expected cumulated reward that can be obtained from this state) of the augmented state (s,ρ) at time step t:

$$
\begin{aligned}
&\min q_{(s_0,0),0} \\
q_{(s,\rho),\mathrm{T}} &= \sum_{(s',\rho') \in \overline{\mathscr{S}}_{\mathrm{T}}} p_{(s',\rho'),\mathrm{T}} \varphi(\rho,\rho'), \forall (s,\rho) \in \overline{\mathscr{S}}_{\mathrm{T}} \\
q_{(s,\rho),t} &\geq \sum_{(s',\rho') \in \overline{\mathscr{S}}} \overline{\mathscr{P}}_t(s',\rho'|s,\rho,a) q_{(s',\rho'),t+1}, \forall\, a \in \mathscr{A}_s, \forall (s,\rho) \in \overline{\mathscr{S}}, \forall\, t \in \{0,\dots,\mathrm{T}-1\} \\
q_{(s,\rho),t} &\in \mathbb{R}, \forall (s,\rho) \in \overline{\mathscr{S}}, \forall\, t \in \{0,\dots,\mathrm{T}\}
\end{aligned}
$$

We have therefore $q_{(s_0,0),0} = \max_{\pi \in \Pi^t_{\overline{s}}} \varphi(\pi,\pi_r)$. Putting together the objective function $\min q_{(s_0,0),0}$ and both classes of constraints, we obtain the final program $\mathscr{P}^{\mathrm{MDP}}_{\mathrm{SSB}}$ given be-

[4]The notation $\mathbb{1}_{x=y}$ denotes a binary value equal to 1 if $x = y$ and 0 otherwise.

low, that enables to determine an SSB optimal randomized policy.

$$\mathscr{P}_{\text{SSB}}^{\text{MDP}} \begin{cases} \min q_{(s_0,0),0} \\ p_{(s_0,0),0} = 1 \\ p_{(s,\rho),0} = 0, \quad \forall (s,\rho) \in \overline{\mathscr{S}} \text{ such that } (s,\rho) \neq (s_0,0) \\ p_{(s,\rho),t} = \sum\limits_{a \in \mathscr{A}_s} p_{(s,\rho),a,t}, \quad \forall (s,\rho) \in \overline{\mathscr{S}}, \forall t \in \{0,\dots,\mathrm{T}-1\} \\ p_{(s',\rho'),t+1} = \sum\limits_{(s,\rho)\in\overline{\mathscr{S}}} \sum\limits_{a \in \mathscr{A}_s} p_{(s,\rho),a,t} \overline{\mathscr{P}}_t(s',\rho'|s,\rho,a), \quad \forall (s',\rho') \in \overline{\mathscr{S}}, \forall t \in \{0,\dots,\mathrm{T}-1\} \\ q_{(s,\rho),\mathrm{T}} = \sum\limits_{(s',\rho')\in\overline{\mathscr{S}}_{\mathrm{T}}} p_{(s',\rho'),\mathrm{T}} \varphi(\rho,\rho'), \forall (s,\rho) \in \overline{\mathscr{S}}_{\mathrm{T}} \\ q_{(s,\rho),t} \geq \sum\limits_{(s',\rho')\in\overline{\mathscr{S}}} \overline{\mathscr{P}}_t(s',\rho'|s,\rho,a) q_{(s',\rho'),t+1}, \forall a \in \mathscr{A}_s, \forall (s,\rho) \in \overline{\mathscr{S}}, \forall t \in \{0,\dots,\mathrm{T}-1\} \\ p_{(s,\rho),t} \geq 0, \quad \forall (s,\rho) \in \overline{\mathscr{S}}, \forall t \in \{0,\dots,\mathrm{T}\} \\ p_{(s,\rho),a,t} \geq 0, \quad \forall (s,\rho) \in \overline{\mathscr{S}}, \forall a \in \mathscr{A}_s, \forall t \in \{0,\dots,\mathrm{T}-1\} \\ q_{(s,\rho),t} \in \mathbb{R}, \forall (s,\rho) \in \overline{\mathscr{S}}, \forall t \in \{0,\dots,\mathrm{T}\} \end{cases}$$

As the size of $\overline{\mathscr{S}}$ may be exponential in the size of $\mathscr{S}$, this LP might involve a number of variables exponential as a function of the size of the original problem. However, if we assume that the rewards are integer values and that the cost function is bounded by $-\mathrm{U}$ and U (i.e., $-\mathrm{U} \leq c_t(s,a,s') \leq \mathrm{U}$), then the number of states in $\overline{\mathscr{S}}$ is bounded by $\mathrm{T} \times |\mathscr{S}| \times 2\mathrm{U}$. Indeed, in this case, the wealth level of each state (s,ρ) is an integer value belonging to the interval $[-\mathrm{TU}, \mathrm{TU}]$. As a consequence, in that case, the number of variables and constraints in program $\mathscr{P}_{\text{SSB}}^{\text{MDP}}$ is polynomial in T, $|\mathscr{S}|$, $|\mathscr{A}|$ and U. Thus, by polynomial complexity of linear programming [Khachiyan, 1980], determining an SSB optimal randomized policy in a finite horizon MDP can be done in pseudo-polynomial time.

Theorem 18. *Finding an SSB optimal policy in a finite horizon MDP, i.e., solving optimization problem* $(\mathscr{R})$*, can be done with a pseudo-polynomial algorithm if all reward values are integer values.*

In the next subsection, we turn to the special case of the WEU model to see if this result can be refined in this more specific setting.

4.3.3 Optimizing the WEU Criterion

If function φ takes the form $\varphi(x,y) = u(x)w(y) - u(y)w(x)$ where w is a strictly positive function, then the SSB model becomes the WEU model. Let π and π' be two strategies, then according to the WEU model, we have:

$$\begin{aligned} l_\pi \succeq l_{\pi'} &\Leftrightarrow \sum_{\rho\in\mathscr{W}_{\mathrm{T}}} \sum_{\rho'\in\mathscr{W}_{\mathrm{T}}} l_\pi(\rho) l_{\pi'}(\rho') (u(\rho)w(\rho') - u(\rho')w(\rho)) \geq 0 \\ &\Leftrightarrow \left(\sum_{\rho\in\mathscr{W}_{\mathrm{T}}} l_\delta(\rho)u(\rho)\right)\left(\sum_{\rho'\in\mathscr{W}_{\mathrm{T}}} l_{\delta'}(\rho')w(\rho')\right) \geq \left(\sum_{\rho\in\mathscr{W}_{\mathrm{T}}} l_\delta(\rho)w(\rho)\right)\left(\sum_{\rho'\in\mathscr{W}_{\mathrm{T}}} l_{\delta'}(\rho')u(\rho')\right) \\ &\Leftrightarrow \frac{\sum_{\rho\in\mathscr{W}_{\mathrm{T}}} l_\delta(\rho)u(\rho)}{\sum_{\rho\in\mathscr{W}_{\mathrm{T}}} l_\delta(\rho)w(\rho)} \geq \frac{\sum_{\rho'\in\mathscr{W}_{\mathrm{T}}} l_{\delta'}(\rho')u(\rho')}{\sum_{\rho'\in\mathscr{W}_{\mathrm{T}}} l_{\delta'}(\rho')w(\rho')} \qquad (\text{as } w>0) \end{aligned} \tag{4.26}$$

Thus, as shown by Equation 4.26, solving a finite horizon MDP with the WEU model reduces to the optimization problem $\max_{\pi \in \widetilde{\Pi}^t_s} u(l_\pi)/w(l_\pi)$.

We now show that, with the WEU criterion, an optimal policy always exists in Π^t_s.

Let $\Pi^t_s = \{\pi_1, \ldots, \pi_n\}$. For any policy $\pi \in \widetilde{\Pi}^t_s$, there exists positive λ_i's summing up to 1 such that $l_\pi = \sum_{i=1}^n \lambda_i l_{\pi_i}$. By bilinearity of φ, we have:

$$\varphi(l_{\pi_j}, \sum_i \lambda_i l_{\pi_i}) = \sum_{i=1}^n \lambda_i \varphi(l_{\pi_j}, l_{\pi_i}) \quad \forall j \in \{1, \ldots, n\} \tag{4.27}$$

Let $\pi^* \in \Pi^t_s$ be an optimal policy within the set of deterministic wealth Markovian policies. As the preferences entailed by the WEU model are transitive, we have $\varphi(l_{\pi^*}, l_{\pi_i}) \geq 0$ for all i. Hence, by positivity of λ_i's, we can deduce that $\varphi(l_{\pi^*}, \sum_i \lambda_i l_{\pi_i}) \geq 0$. Put another way, π^* is preferred to any policy in $\widetilde{\Pi}^t_s$ and is therefore also preferred to any policy in $\widetilde{\Pi}_{h,r}$. By Proposition 10, π^* is therefore an SSB optimal policy.

Theorem 19. *For any sequential decision problem under risk represented as a finite horizon MDP, a WEU-optimal policy always exists as a deterministic wealth-Markovian policy.*

We now discuss the computation of a WEU optimal policy in Π^t_s. As shown by Equation 4.26, optimizing the WEU model in a finite horizon MDP reduces to the optimization problem $\max_{\pi \in \Pi^t_s} u(l_\pi)/w(l_\pi)$. This problem can be solved by using program $\mathscr{P}^{\text{MDP}}_{\text{SSB}}$. Indeed, note that at the optimum of program $\mathscr{P}^{\text{MDP}}_{\text{SSB}}$, variables $p_{(s,\rho),t}$ and $p_{(s,\rho),a,t}$ describe an optimal randomized policy π_r in $\Pi^t_{s,r}$ (which therefore maximizes the ratio $u(l_\pi)/w(l_\pi)$) and variables $q_{(s,\rho),t}$ describe a deterministic wealth-Markovian policy $\pi \in \Pi^t_s$ which is a best response to π_r. It is then easy to realize that the policy π must be a WEU optimal strategy. Indeed, by definition of best response:

$$\varphi(l_\pi, l_{\pi_r}) \geq \varphi(l_{\pi_r}, l_{\pi_r}) = 0.$$

By replacing φ by its expression (i.e., $\varphi(x, y) = u(x)w(y) - u(y)w(x)$), we obtain:

$$\frac{u(l_\pi)}{w(l_\pi)} \geq \frac{u(l_{\pi_r})}{w(l_{\pi_r})} = \max_{\pi \in \Pi^t_{s,r}} \frac{u(l_\pi)}{w(l_\pi)}.$$

Thus, policy $\pi \in \Pi^t_s$ is a WEU optimal deterministic policy.

As solving the linear program $\mathscr{P}^{\text{MDP}}_{\text{SSB}}$ is a pseudo-polynomial time problem if all reward functions are integer-valued, we can conclude that finding a WEU optimal policy which is in Π^t_s is also a pseudo-polynomial time problem in that case.

Theorem 20. *Finding a WEU optimal deterministic policy in a finite horizon MDP can be done in pseudo-polynomial time if all reward functions are integer-valued.*

Note that more efficient solution methods than $\mathscr{P}^{\text{MDP}}_{\text{SSB}}$ could be designed in the WEU case, by resorting to fractional programming methods as in the previous section. Indeed, as functions u and w are linear with respect to mixtures of probabilities, the determination of a WEU optimal policy can be identified as a fractional version of a standard finite horizon MDP with EU as decision criterion. Megiddo's method [Megiddo, 1979] and Dinkelbach's method [Dinkelbach, 1967] could be used. In both methods, the oracle $\mathscr{O}$ would consist in solving the finite horizon MDP with a utility function V^λ of the form $\mathrm{V}^\lambda(x) \rightarrow u(x) - \lambda w(x)$. This problem can be solved by using a backward induction method in the augmented MDP where the reward function would be induced by function V^λ.

4.3.4 Numerical Tests

We provide here experimental results in order to demonstrate the operationally of the double oracle method and to provide a deeper insight on the interest of using an SSB utility function.

Who Wants to Be a Millionaire? In this popular television game show, a contestant tries to answer a sequence of 15 multiple-choice questions of increasing difficulty. Questions (four possible answers are given) are played for increasingly large sums, roughly doubling the pot. At each time step, the contestant may decide to walk away with the money currently won. If she answers incorrectly then all winnings are lost besides what has been earned at a "guarantee point" (questions 5 and 10). The player is given the possibility of using 3 lifelines (*50:50*, removing two of the possible choices, *ask the audience* and *call a friend* for suggestions); each can only be used once in the whole game.

We used the two models of the Spanish 2003 version of the game presented by Perea and Puerto [2007].[5] In the first model the probability of answering correctly is a function of the question's number and the lifelines (if any) used; lifelines increase the probability of answering correctly (the model is fitted using real data). This first model is overly simplistic as it does not actually take into account whether the player does in fact know the answer or not. The second model represents the hesitation of the contestant by distinguishing four epistemic cases, corresponding to the number of answers (among the four given) that are believed possible correct answers for the current question. At each step of the game, when a new question is asked, a categorical distribution dictates the probability of each of the epistemic cases. As the game proceeds, questions are more difficult, and the distribution is then skewed towards hesitating between larger sets of answers.

[5]The possible wealth values are 0, 150, 300, 450, 900, 1800, 2100, 2700, 3600, 4500, 9k, 18k, 36k, 72k, 144k, 330k.

	Exp	PD	PD	PD	RA	Th
p	1	0.07	0.39	0.54	1	1
t=1	NL	NL	NL	NL	NL	NL
t=2	NL	NL	NL	NL	NL	NL
t=3	NL	NL	NL	NL	NL	NL
t=4	NL	50:50	50:50	50:50	50:50	NL
t=5	NL	P	P	P & A	P&A	A
t=6	NL	NL	NL	NL	NL	50:50
t=7	NL	A	NL	NL	NL	P
t=8	NL	S	A	S	NL	S
t=9	50:50		S		NL	
t=10	P				NL	
t=11	NL				NL	
t=12	A				S	
t=13	S					

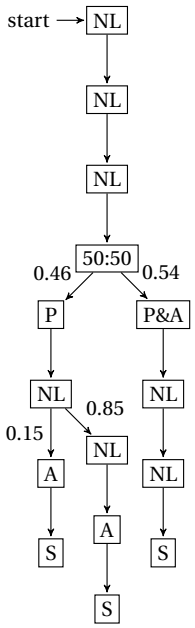


Figure 4.7: First model of the game. (Left) Optimal mixed policies according to the following criteria : Exp=Expectation, PD=Probabilistic Dominance, RA=Risk-averse, Th=Threshold. (Right) Optimal randomized policy according to the PD criterion (NL=No Lifelines, P=Call a friend, A=Audience, S=Stop)

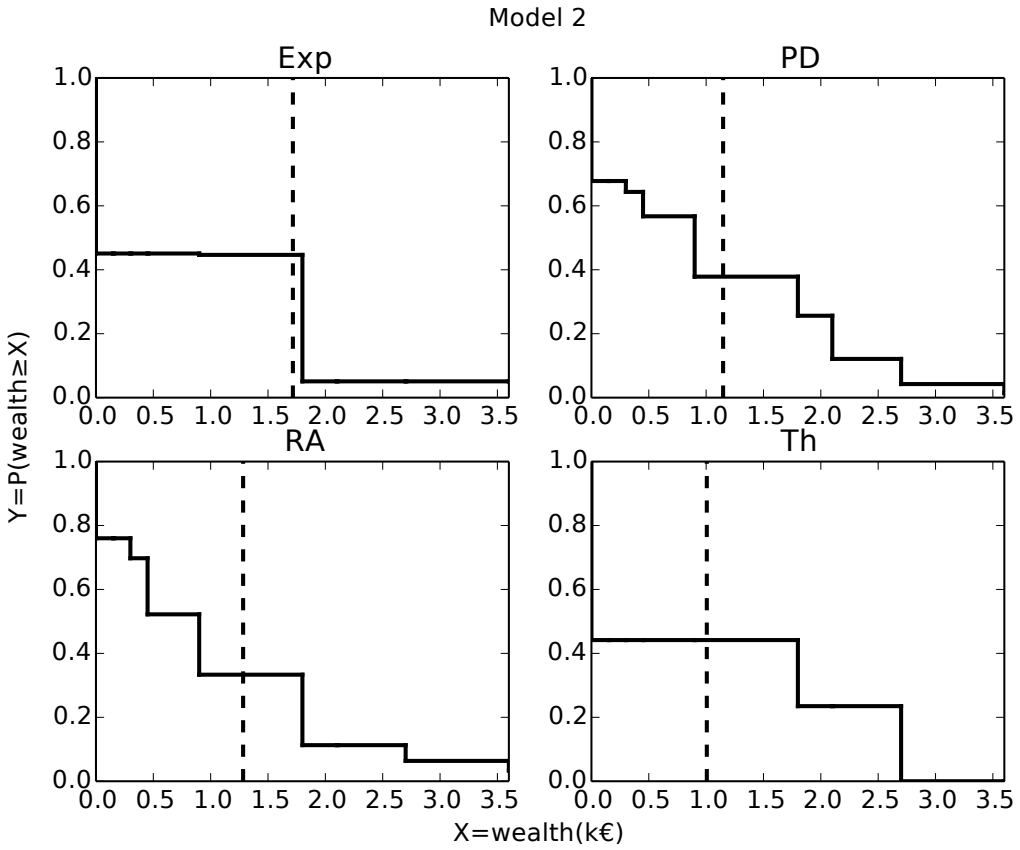


Figure 4.8: Second model of the game. Decumulative probability distribution of wealth according to the different SSB utility functions. The dashed vertical line indicates the expected wealth.

We computed the optimal policies for the two models according to several instantiations of the SSB utility function: the expectation (Exp), probabilistic dominance (PD), threshold probability (Th) criteria (threshold set to 2700) and a risk averse SSB utility function (RA) defined by $\varphi_{\mathrm{RA}}(x,y) = (x-y)/(x+y+1)^{\frac{2}{3}}$, which is indeed a risk averse SSB function on $(\mathbb{R}^+)^2$ since Nakamura's condition holds (see Proposition 4 on page 37 in Section 1.5.3).

Given the simplicity of the first model of the game, policies can be compactly displayed; the optimal policies are reported on the left of Figure 4.7. The Exp, RA and Th criteria are associated with deterministic policies, while the PD-optimal policy can be seen either as a mixed policy or as a randomized policy (see right of Figure 4.7). According to intuition, the preferred policies for criteria PD and RA make use of lifelines much earlier in the game, in order to secure a significant gain; the preferred policy for criterion Th uses the lifelines to reach 2700 with high probability and stops playing thereafter. Finally, with criterion Exp, one keeps aside the lifelines for later (more difficult) questions, even at a cost of a premature end of the game.

When using the second model, the state space is much larger and policies are too complex to be represented compactly. As we are interested in comparing their overall performance, we plot in Figure 4.8 the decumulative distribution of wealth (as the pot skyrockets if the contestant reaches the very last questions, but this happens for all policies with low probabilities, we plot only the 9 first wealth levels for emphasizing the differences between the wealth distributions).

Unsurprisingly, the expectation-optimal policy yields the highest wealth expectancy

(2387 and 1717 in the two models of the game). While the optimal policy according to PD achieves a lower expected value, it scores better very often: it achieves a wealth level at least as good as 75% (resp. 70%) of the time and strictly better 44% (resp. 48%) of the time, when considering the first (resp. second) model. The policy obtained with the RA criterion is safer than the expectation-optimal policy. Regarding the threshold-optimal policy, it obtains with higher probability (23%) the threshold objective as can be seen in Figure 4.8 (second model).

Regarding the computational aspect, the initial MDP in this problem consists of 9 (resp. 136) states for the first model (resp. second model) and the corresponding augmented MDP has 33 (resp. 496) states. The computation time (resp. number of iterations of the double oracle algorithm) for finding each optimal policy for the second model of the game was respectively of 7.6s (resp. 1) for the Exp criterion, 9.0s (resp. 8) for PD criterion, 7.8s (resp 3) for the RA criterion and 7.6s (resp 1) for the Th criterion.[6]

Other domains. To have a deeper insight into the operationality of our method, we have carried out some preliminary experiments on other domains, notably on a simple grid world (GW) domain (with randomly generated instances) and a cancer clinical trials (CCT) domain (with a model proposed by Cheng *et al.* [2011]). In both domains, we used the probabilistic dominance criterion. In the GW domain, the initial state space includes from 100 to 400 states, and the augmented state space from thousands to tens of thousands states. The computation times vary from a few seconds to half an hour. Regarding the CCT domain, the initial state space includes 5078 states, and the augmented state space 5129 states. The method takes about two minutes to compute an optimal policy. Generally speaking, provided the structure of the MDP prevents the augmentation of the state space to be too costly (as can bee seen in the two previous examples, there is an important variability in the increase of the number of states after augmentation of the MDP), we believe the method is rather scalable.

4.4 Conclusion

In this chapter, we have investigated the resolution of sequential decision problems under risk with the SSB and WEU models in decision trees and in finite horizon MDPs. SSB utility theory and WEU theory generalize von Neumann and Morgenstern's EU theory to encompass rational decision behaviors that EU cannot accommodate. Thus, they make it possible to represent a wider scope of preferences in sequential decision problems.

In a first section, we proved that an SSB optimal strategy always exists in decision trees as a randomized strategy or (equivalently) as a mixed strategy. We showed that determining an SSB optimal strategy among deterministic strategies is an NP-hard problem while it is polynomial when considering randomized strategies. We gave a game theoretic interpretation of the optimization problem induced by the SSB model in a decision tree and derived from it several oracle methods to find an SSB optimal randomized strategy. Regarding the special case of WEU, both the deterministic and the randomized settings collapse as there always exists a WEU optimal strategy which is deterministic. We showed that determining such an optimal strategy is polynomial time by instantiating fractional programming methods. As far as we know, it is the first polynomial time complexity re-

[6] All times are wall-clock times on a 2,4 GHz Intel Core i5 machine with 8G main memory. Our implementation is in Python, with an external call to GUROBI version 5.6.3 in order to solve the linear programs required to find the Nash equilibria.

sult in sequential decision making under risk for a decision model encompassing Allais' paradox.

In a second section, we showed how to extend these results to the setting of finite horizon MDPs. We showed that there always exists an SSB optimal (potentially randomized or mixed) Markovian policy in an augmented MDP where the states are augmented by the sum of rewards accumulated by the agent. We gave a game theoretic interpretation of the optimization problem induced by the SSB model and a finite horizon MDP and derived a double oracle approach from it to find an SSB optimal policy. Lastly, we showed that finding an SSB or WEU optimal strategy can be done with a pseudo-polynomial time algorithm.

Following this work, we investigated the use of SSB utility functions to compare policies in the reinforcement learning setting (a more general setting than MDPs where transition functions are unknown and must be learned). We designed a model-free SSB reinforcement learning algorithm that we called *SSB Q-learning*, to find a policy that is ϵ-optimal according to SSB [Gilbert *et al.*, 2016]. The proposed algorithm is an adaptation of fictitious play [Brown, 1951] combined with techniques from stochastic approximation [Borkar, 1997]. This algorithm has been applied in an industrial context with good results: an automated information extraction (IE) treatment chain modeled as an MDP, and improved using a reward function balancing extraction quality and treatment time. In this setting, SSB utility theory was used to formalize qualitative preferences expressed by human operators on the output of the treatments [Nicart *et al.*, 2016].

4.5 References

N. Bäuerle and J. Ott. Markov decision processes with average value-at-risk criteria. *Mathematical Methods of Operations Research*, 74(3):361–379, 2011. 125

N. Ben Amor, H. Fargier, and W. Guezguez. Possibilistic sequential decision making. *International Journal of Approximate Reasoning*, 55(5):1269–1300, 2014. 107

V. Borkar and R. Jain. Risk-constrained Markov decision processes. *IEEE Trans. on Automatic Control*, 59(9):2574–2579, 2014. 125

V. Borkar. Stochastic approximation with two time scales. *Systems & Control Letters*, 29(5):291 – 294, 1997. 139

M. Bouakiz and Y. Kebir. Target-level criterion in Markov decision processes. *Journal of Optimization Theory and Applications*, 86(1):1–15, 1995. 125

M. Boussard, M. Bouzid, A.-I. Mouaddib, R. Sabbadin, and P. Weng. *Markov Decision Processes in Artificial Intelligence*, chapter Non-Standard Criteria, pages 319–359. Wiley, 2010. 125

G. W. Brown. Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, 1951. 139

W. Cheng, J. Fürnkranz, E. Hüllermeier, and S.-H. Park. Preference-based policy iteration: Leveraging preference learning for reinforcement learning. In *Proc. of European Conference on Machine Learning and Knowledge Discovery in Databases ECML PKDD Part I*, pages 312–327, 2011. 138

S.H. Chew. A generalization of the quasilinear mean with applications to the measurement of income inequality and decision theory resolving the Allais paradox. *Econometrica: Journal of the Econometric Society*, pages 1065–1092, 1983. 104

Y. Chow and M. Ghavamzadeh. Algorithms for CVaR optimization in MDPs. In *Neural Information Processing Systems*, 2014. 125

E.J. Collins and J.M. McNamara. Finite-horizon dynamic optimisation when the terminal reward is a concave functional of the distribution of the final state. *Advances in Applied Probability*, 30(1):122–136, 1998. 108, 109, 130

W. Dinkelbach. On nonlinear fractional programming. *Management science*, 13(7):492–498, 1967. 121, 135

D. Dubois and H. Prade. Possibility theory as a basis for qualitative decision theory. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 95, pages 1924–1930, 1995. 107

D. Dubois, H. Fargier, and P. Perny. Qualitative decision theory with preference relations and comparative uncertainty: An axiomatic approach. *Artificial Intelligence*, 148:219–260, 2003. 107

S. Ermon, C. Gomes, B. Selman, and A. Vladimirsky. Probabilistic planning with non-linear utility functions and worst-case guarantees. In *AAMAS*, pages 965–972, 2012. 125

Y.Y. Fan, R.E. Kalaba, and J.E. Moore II. Arriving on time. *Journal of Optimization Theory and Applications*, 127(3):497–513, 2005. 125

H. Fargier, G. Jeantet, and O. Spanjaard. Resolute choice in sequential decision problems with multiple priors. In *Proceedings of the International Joint Conference on Artificial Intelligence 2011*, pages 2120–2125, 2011. 106

J.A. Filar, L.C.M. Kallenberg, and H-M Lee. Variance-penalized markov decision processes. *Mathematics of Operations Research*, 14(1):147–161, 1989. 125

P.C. Fishburn. Nontransitive measurable utility. *Journal of Mathematical Psychology*, 26(1):31–67, 1982. 104

P.C. Fishburn. Transitive measurable utility. *Journal of Economic Theory*, 31(2):293–317, 1983. 104

P.C. Fishburn. SSB utility theory: an economic perspective. *Mathematical Social Sciences*, 8(1):63 – 94, 1984. 104

L. Garcia and R. Sabbadin. Possibilistic influence diagrams. In *Proceedings of the European Conference on Artificial Intelligence*, volume 6, pages 372–376, 2006. 107

P.H. Giang and P.P. Shenoy. A comparison of axiomatic approaches to qualitative decision making using possibility theory. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 162–170. Morgan Kaufmann Publishers Inc., 2001. 107

H. Gilbert and O. Spanjaard. Complexity of Solving Decision Trees with Skew-Symmetric Bilinear Utility. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence 2017*, 2017. 104

H. Gilbert, O. Spanjaard, P. Viappiani, and P. Weng. Solving MDPs with Skew Symmetric Bilinear Utility Functions. In *Proceedings of the International Joint Conference on Artificial Intelligence 2015*, pages 1989–1995, 2015. 104

H. Gilbert, B. Zanuttini, P. Viappiani, P. Weng, and E. Nicart. Model-free reinforcement learning with skew-symmetric bilinear utilities. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence 2016*, 2016. 139

P. Hou, W. Yeoh, and P. Varakantham. Revisiting risk-sensitive MDPs: New algorithms and results. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 136–144, 2014. 125

N. Huntley and M. Troffaes. An efficient normal form solution to decision trees with lower previsions. In *Soft Methods for Handling Variability and Imprecision*, pages 419–426. Springer, 2008. 107

S. Iwamoto. Stochastic optimization of forward recursive functions. *Journal of Mathematical Analysis and Applications*, 292(1):73 – 83, 2004. 125

J.-Y. Jaffray and M. Jeleva. Information processing under imprecise risk with the Hurwicz criterion. In *5th international symposium on imprecise probability: theories and applications*, pages 233–242, 2007. 107

G. Jeantet and O. Spanjaard. Rank-dependent Probability Weighting in Sequential Decision Problems under Uncertainty. In *Proceedings of the International Conference on Autonomous Planning and Scheduling (ICAPS)*, pages 148–155. AAAI Press, 2008. 107, 124

G. Jeantet and O. Spanjaard. Optimizing the Hurwicz criterion in decision trees with imprecise probabilities. In *International Conference on Algorithmic Decision Theory*, pages 340–352. Springer, 2009. 107

G. Jeantet, P. Perny, and O. Spanjaard. Sequential Decision Making with Rank Dependent Utility: a Minimax Regret Approach. In *Proc. of AAAI 2012*, pages 1931–1937, 2012. 107

Y. Kadota, M. Kurano, and M. Yasuda. Discounted Markov decision processes with utility constraints. *Computers & Mathematics with Applications*, 51(2):279–284, 2006. 125

L.G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980. 113, 134

D. Kikuti, F.G. Cozman, and R. Shirota Filho. Sequential decision making with partially ordered preferences. *Artificial Intelligence*, 175(7-8):1346–1365, 2011. 106

S. Koenig and R.G. Simmons. How to make reactive planners risk-sensitive. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, volume 293298, 1994. 125

Y. Liu and S. Koenig. Functional value iteration for decision-theoretic planning with general utility functions. In *Proceedings of AAAI 2006*, volume 21, page 1186, 2006. 125

Y. Liu and S. Koenig. An exact algorithm for solving MDPs under risk-sensitive planning objectives with one-switch utility functions. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '08, pages 453–460, 2008. 125, 128

Y. Liu. Risk-sensitive planning with one-switch utility functions: Value iteration. In *In AAAI*, pages 993–999, 2005. 125, 126

S. Mannor and J. Tsitsiklis. Mean-variance optimization in Markov decision processes. In *International Conference on Machine Learning*, 2011. 125

H.B. McMahan, G.J. Gordon, and A. Blum. Planning in the presence of cost functions controlled by an adversary. In *International Conference on Machine Learning*, pages 536–543, 2003. 131

N. Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4(4):414–424, 1979. 121, 135

E. Nicart, B. Zanuttini, H. Gilbert, B. Grilhères, and F. Praca. Building document treatment chains using reinforcement learning and intuitive feedback. In *Proceedings of International Conference on Tools with Artificial Intelligence 2016*, pages 635–639, 2016. 139

T.D. Nielsen and J.Y. Jaffray. Dynamic decision making without expected utility: An operational approach. *European Journal of Operational Research*, 169(1):226–246, 2006. 107

Y. Ohtsubo and K. Toyonaga. Optimal policy for minimizing risk models in Markov decision processes. *Journal of mathematical analysis and applications*, 271:66–81, 2002. 125

F. Perea and J. Puerto. Dynamic programming analysis of the TV game "Who wants to be a millionaire?". *European Journal of Operational Research*, 183(2):805–811, 2007. 124, 136

L.A. Prashanth and M. Ghavamzadeh. Actor-critic algorithms for risk-sensitive MDPs. In *Neural Information Processing Systems*, pages 252–260, 2013. 125

J. Quiggin. *Generalized Expected Utility Theory: The Rank-Dependent Model*. Kluwer, 1993. 107

Y. Rébillé. Decision making over necessity measures through the Choquet integral criterion. *Fuzzy sets and systems*, 157(23):3025–3039, 2006. 107

O. Spanjaard and P. Weng. Markov decision processes with functional rewards. In *International Workshop on Multi-disciplinary Trends in Artificial Intelligence*, pages 269–280. Springer, 2013. 125

R. Strauch and Jr. A.F. Veinott. A property of sequential control processes. Technical report, Rand McNally, Chicago, 1966. 130

D. J. White. Utility, probabilistic constraints, mean and variance of discounted rewards in Markov decision processes. *OR Spektrum*, 9:13–22, 1987. 125

D.J. White. Minimising a threshold probability in discounted Markov decision processes. *Journal of mathematical analysis and applications*, 173(634–646), 1993. 125

C. Wu and Y. Lin. Minimizing risk models in Markov decision processes with policies depending on target values. *Journal of mathematical analysis and applications*, 231:41–67, 1999. 125

H.P. Young. Extending Condorcet's rule. *Journal of Economic Theory*, 16(2):335–353, 1977. 105

S.X. Yu, Y. Lin, and P. Yan. Optimization models for the first arrival target distribution function in discrete time. *Journal of mathematical analysis and applications*, 225:193–223, 1998. 125

L.A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy sets and systems*, 100:9–34, 1999. 107

Chapter 5

Decision Making under Risk with Ordinal/Imprecise Values

" When your values are clear to you, making decisions become easier. "

R.E. Disney

Section Contents

Summary of the chapter

This chapter deals with decision making under risk (see section 1.5) and sequential decision making (see section 2.2) with ordinal/preference-based approaches or imprecisely known parameters (e.g., reward parameters, parameters of a decision model). With this type of information, we can either accept that the parameters have imprecisely known values and work with decision criteria adapted to this situation or we can specify the values of these parameters (by interacting with an expert for instance) to elicit the minimum number of pieces of information required to be able to discriminate with a partially specified decision criterion.

In this chapter, we present three works that are related to these issues. In a first section, we discuss an ordinal approach, based on a quantile criterion, to solve finite horizon MDPs where only an order over possible episodes is known. Then, in a second section, we assume the rewards of an infinite horizon MDP are imprecisely known and we refine an interactive version of the usual value iteration procedure to solve the MDP while asking the minimum number of preference queries to an expert. Lastly, in a third section, we assume that we must assist a decision maker consistent with the weighted expected utility model. We develop an elicitation protocol to specify the parameters of the weighted expected utility model consistent with her preferences. Then we show how to integrate this protocol in an incremental elicitation procedure.

This chapter is based on several publications [Gilbert *et al.*, 2015, 2017a,b].

5.1 Introduction

Decision problems under risk (see section 1.5) and sequential decision problems (see section 2.2) usually involve decision criteria that require to know the exact values of some parameters (e.g., the rewards of an MDP, the utility function in an EU model, etc.). These parameters are defined either implicitly by the environment or by a human user. Making the hypothesis that they can be known exactly is often unrealistic:

- In the case where these parameters are defined by the environment, it may be impossible or too costly to determine them with a perfect precision.
- The specification of the parameters by a human represents also a difficulty as this task can be cognitively hard, even if the human is an “expert” user.

Therefore, we may investigate frameworks where this precise information about parameters is not required a priori. This idea has led to much work aiming to reduce the burden of specifying completely each parameter. Two possible approaches in this direction are:

- We can accept that the values of the parameters are not known precisely (e.g., we may only have ordinal information on the parameters) and work with decision criteria that can cope with the resulting lack of information (non standard criteria approach).
- We can interact with the decision maker or an “expert” to elicit the minimum number of pieces of information required to use the a priori chosen decision criterion (elicitation approach).

In this chapter, we present three works that rely either on the non standard criteria approach or on the elicitation approach. These works are presented in three different sections (see Table 5.1).

	Problem Studied	Decision Criterion	Type of Approach/Problem
Section 1	Finite horizon MDP	Quantile criterion	Ordinal/Preference-based approach
Section 2	Infinite horizon MDP	Total discounted expected return	Elicitation and optimization intertwined
Section 3	Decision problems under risk	WEU model	Incremental elicitation strategies

Table 5.1: Summary of Chapter 5.

In a first section, we discuss an ordinal approach for finite horizon MDPs where only an order over possible episodes is known. To solve the MDP, we investigate a quantile criterion and develop a solution method.

Then, in a second section, we assume the rewards of an infinite horizon MDP are imprecisely known but can be precised by asking preference queries to an expert (also called tutor). In this framework, we refine an interactive version of the usual value iteration procedure [Weng and Zanuttini, 2013] to solve the MDP while asking the minimum number of preference queries to the expert.

Lastly, in a third section, we assume that we must assist a decision maker consistent with the WEU model. However, the parameters of her preference model (i.e., the two functions u and w involved in the WEU model) are unknown and must be elicited. We develop an elicitation protocol to precise the values of the two functions required by this decision model and show how to integrate it efficiently in an incremental elicitation procedure.

5.2 An Ordinal Approach for Sequential Decision Making under Risk: Finite Horizon MDPs with a Quantile Criterion

Sequential decision-making in uncertain environments is an important task in artificial intelligence. Such problems can be modeled as MDPs (see subsection 2.2.2). In an MDP, an agent chooses at every time step the action to perform according to the current state of the world in order to optimize a criterion. In standard MDPs, uncertainty is described by probabilities over the possible action outcomes, preferences are represented by numeric rewards and the expectation of future cumulated rewards is used as the decision criterion. And yet, for numerous applications, the expectation of cumulated rewards may not be the most appropriate criterion. For instance, in one-shot decision-making problems, an alternative and well motivated objective for the agent is to ensure a certain level of satisfaction with high probability. Secondly, the expectation of cumulated rewards requires to know the exact values of the rewards parameters which may be unrealistic.

In this section, we focus on the decision criterion that consists in maximizing a quantile. Intuitively, the τ-quantile of a population is the value x such that $100 \cdot \tau$ percent of the population is lower than or equal to x and $100 \cdot (1-\tau)$ percent of the population is greater than or equal to x. Optimizing a quantile criterion offers nice properties:

i) no assumption is made about the commensurability between preferences and uncertainty,

ii) preferences over actions or episodes can be expressed on a purely ordinal scale,

iii) preferences induced over policies are more robust than with the standard criterion of maximizing the expectation of cumulated rewards.

As a result, maximizing a quantile is used in many applications. For instance, the *Value-at-Risk* criterion [Jorion, 2006] widely used in finance is in fact a quantile. Moreover, in the Web industry [DeCandia *et al.*, 2007; Wolski and Brevik, 2014], decisions about performance or Quality-Of-Service are often made based on quantiles. For instance, Amazon reports [DeCandia *et al.*, 2007] that they optimize the 0.999-quantile for their cloud services. More generally, in the service industry, because of skewed distributions [Benoit and Van den Poel, 2009], one generally does not want customers to be satisfied on average, but rather that most customers (e.g., 99% of them) be as satisfied as possible.

In this section, we show that optimizing the quantile criterion in finite horizon MDPs amounts to solving a sequence of MDPs using an expected utility criterion. We provide a binary search algorithm using functional backward induction [Liu and Koenig, 2006] as a subroutine for computing a near-optimal policy. We investigate some properties of the optimal policies. Finally, we provide the results of experiments testing the proposed algorithm.

5.2.1 Related Work

Several works in the MDP literature [Boussard *et al.*, 2010] considered decision criteria different from the standard ones (i.e., expected discounted sum of rewards, expected total rewards or expected average rewards). For instance, in the operations research community, White [1988] considered different non standard criteria for MDPs where the preferences over policies only depend on sums of rewards: expected utility, probabilistic constraints and mean-variance formulations (i.e., maximizing the expected total reward while ensuring an upper bound on the variance or minimizing the variance while ensuring a lower bound on the expected total reward). In this context, he showed the sufficiency of working in a state space augmented with the sum of rewards obtained so far. Recently, Prashanth and Ghavamzadeh [2013] and Mannor and Tsitsiklis [2011] provided algorithms for the mean-variance formulations described by White. Filar *et al.* [1989] investigated decision criteria that are variance-penalized versions of the standard ones. They formulated the obtained optimization problem as a non-linear program. Several researchers [Hou *et al.*, 2014; Wu and Lin, 1999; Yu *et al.*, 1998] worked on the problem of maximizing the probability that the total (discounted) reward exceeds a given threshold.

Recent work in MDP and reinforcement learning considered conditional Value-at-risk (CVaR), a criterion related to quantile, as a risk measure. Bäuerle and Ott [2011] proved the existence of deterministic wealth-Markovian policies optimal with respect to CVaR. Chow and Ghavamzadeh [2014] proposed gradient-based algorithms for CVaR optimization. In contrast, Borkar and Jain [2014] used CVaR in inequality constraints instead of using it as objective function.

Closer to the work of this section, several quantile-based decision models have been investigated in different contexts. In uncertain MDPs where the parameters of the transition and reward functions are imprecisely known, Delage and Mannor [2007] presented and investigated a quantile-like criterion to capture the trade-off between optimistic and pessimistic viewpoints on an uncertain MDP. The quantile criterion they use is different from ours as it takes into account the uncertainty present in the parameters of the MDP. Filar *et al.* [1995] proposed an algorithm for optimizing the quantile criterion when histories are valued by average rewards. In that setting, they showed that an optimal stationary deterministic Markovian policy exists.

Filar [1983] and more recently Weng [2011, 2012] proposed decision criteria that could be used when only the order over rewards, but not the exact values, is known. They both

investigated quantile-based decision models to compute policies that maximize a quantile. While quantiles in those works are defined on distributions over ordinal rewards, we defined them as distributions over histories.

These approaches with ordinal rewards are close to the preference-based setting that only relies on ordinal pieces of information such as "this episode is preferred to this other episode". Such preference-based approaches have received much attention lately [Akrour *et al.*, 2012; Wilson *et al.*, 2012; Wirth and Neumann, 2015]. In this domain, other non standard criteria are used as probabilistic dominance (which consists in maximizing the probability of yielding a preferred outcome when compared to another solution) [Busa-fekete *et al.*, 2014], a special case of SSB utility function.

Lastly, in the machine learning community, quantile-based criteria have been proposed in the multi-armed bandit setting, a special case of reinforcement learning. Yu and Nikolova [2013] proposed an algorithm in the pure exploration setting for different risk measures, including Value-at-Risk. Carpentier and Valko [2014] studied the problem of identifying arms with extreme payoffs, a particular case of quantiles. Finally, Szörenyi *et al.* [2015] investigated multi-armed bandit problems where a quantile is optimized instead of the mean.

We now turn to our contribution; we design in this section a very general procedure to optimize a quantile in a finite horizon MDP. Our solution method can be used with numerical rewards as well as with preference-based information over the episodes.

5.2.2 Background

In this subsection, we provide the background information necessary for the rest of the section and we specify the MDP framework that we will use.

Markov Decision Processes. In this section, an MDP is defined as a tuple $\mathscr{M}=(\mathrm{T},\mathscr{S},\mathscr{A},\mathscr{P},c)$ where T is a finite time horizon, $\mathscr{S}$ is a finite set of states containing an initial state s_0, $\mathscr{A}$ is a finite set of actions, $\mathscr{P}$ is a transition function with $\mathscr{P}(s'|s,a)$ being the probability of reaching state s' when action a is performed in state s, $c:\mathscr{S}\times\mathscr{A}\times\mathscr{S}\rightarrow\mathbb{R}$ is a bounded reward function.

We recall that a t-history h_t is a succession of t state-action pairs starting from state s_0 (e.g., $h_t=(s_0,a_0,s_1,\ldots,s_{t-1},a_{t-1},s_t)$) and that T-histories are called episodes. The set of episodes is denoted by $\mathscr{E}$.

Different criteria can be defined in order to compare policies. The most standard criterion is *expected cumulated reward*, for which it is known that an optimal deterministic Markovian policy exists at any horizon T.

However, the decision criterion based on the expectation of cumulated rewards may not always be suitable. Firstly, unfortunately, in many cases, the reward function c is not known and must be specified by an expert. This task may be too cognitively difficult and it may be easier for the expert to just give ordinal information. Hence, in this section, we only assume that we have a strict weak ordering on episodes.

Secondly, for numerous applications, the expectation of cumulated reward may not be the most appropriate criterion (even when a numeric reward function is defined). For instance, in the Web industry, most decisions about performance are based on the minimal quality of 99% of the possible outcomes. Therefore, in this section we aim to use a quantile as a decision criterion to solve an MDP.

Preferences over Episodes. For generality's sake, contrary to standard MDPs, we define the reward function to take values in a set $\mathscr{R}$. In practical applications, this will often be a subset of $\mathbb{R}$ but we keep it general to allow ordinal types of reward. Moreover, we assume that the values of histories take values in a set $\mathscr{W}$, called the wealth level space, and that the value $\rho(h_t)$ of a history $h_t = (s_0, a_0, s_1, \ldots, s_t)$ is defined by:

$$\rho(h_0) = \rho_0 \qquad \rho(h_t) = \rho(h_{t-1}) \circ c(s_{t-1}, a_{t-1}, s_t)$$

where $h_{t-1} = (s_0, a_0, s_1, \ldots, s_{t-1})$, $\circ$ is a binary operation from $\mathscr{W} \times \mathscr{R}$ to $\mathscr{W}$ and $\rho_0 \in \mathscr{W}$ is the left identity element of $\circ$. Let $\mathscr{W}_T = \{\rho(h_T) : h_T \in \mathscr{E}\} \subset \mathscr{W}$ be the set of wealth levels of episodes (i.e., T-histories). We make three assumptions about $\mathscr{W}_T$:

- It is totally ordered by $\preceq_{\mathscr{W}}$, which induces an order $\preceq$ over the episodes: $h_T \preceq h'_T$ iff $\rho(h_T) \preceq_{\mathscr{W}} \rho(h'_T)$,
- It admits a lowest element, denoted by $\rho_{\min}$, and a greatest element, denoted by $\rho_{\max}$, for order $\preceq_{\mathscr{W}}$.
- A distance consistent with $\preceq_{\mathscr{W}}$ is defined over $\mathscr{W}_T$. It is denoted by $d(\rho, \rho')$ for any pair $(\rho, \rho') \in \mathscr{W}_T \times \mathscr{W}_T$.

Note that when a distance is defined, for any pair (ρ, ρ'), its set of mid-elements is also defined:

$$\texttt{mid}(\rho, \rho') = \arg\min_{\rho'' \in \mathscr{W}_T} \{\max(d(\rho, \rho''), d(\rho', \rho''))\}.$$

In a numerical context, the possible wealth levels of a state are the possible sums (resp. γ-discounted sums) of rewards that can be obtained during an episode. Put another way, in a numerical context, we could set $\rho_{\max} = R_{\max} T$ (resp. $\rho_{\max} = R_{\max} \frac{(1-\gamma)^T}{1-\gamma}$) with $R_{\max}$ being the highest possible reward and $\texttt{mid}(\rho, \rho') = \{(\rho + \rho')/2\}$.

In the most general case, the possible wealth levels at the end of an episode are the possible histories (or more precisely their equivalence classes) that can be obtained. Here, if the equivalence classes are known and denoted by $\rho_1 \prec_{\mathscr{W}} \rho_2 \prec_{\mathscr{W}} \ldots \prec_{\mathscr{W}} \rho_m$ and if $d(\rho_i, \rho_j) = |j - i|$, then $\rho_{\min} = \rho_1$, $\rho_{\max} = \rho_m$ and $\texttt{mid}(\rho_i, \rho_j) = \{\rho_{\lfloor (i+j)/2 \rfloor}, \rho_{\lceil (i+j)/2 \rceil}\}$ (where $\lfloor x \rfloor$ is the greatest integer smaller than x and $\lceil x \rceil$ is the smallest integer greater than x).

The goal of the agent is then to make sure that most of the time, she will follow episodes that have the highest possible wealth levels. This can be implemented by optimizing a quantile criterion.

Quantile Criterion. Intuitively, the τ-quantile of a population of ordered elements, for $\tau \in [0, 1]$, is the value q such that $100 \cdot \tau\%$ of the population is equal or lower than q and $100 \cdot (1 - \tau)\%$ of the population is equal or greater than q. The 0.5-quantile, also known as the median, can be seen as the ordinal counterpart of the mean. More generally, quantiles define decision criteria that have the nice property of not requiring numeric valuations of the episodes, but only an order relation. They have been axiomatically studied as decision criteria by Rostek [2010].

We now give a formal definition of quantiles for a policy. We recall that l_π stands for the lottery over wealth levels induced by policy π, i.e., $l_\pi(\rho)$ is the probability of getting a wealth level $\rho \in \mathscr{W}_T$ when applying policy π from the initial state. The *cumulative distribution* induced by l_π is then defined as F^π where $F^\pi(\rho) = \sum_{\rho' \preceq_{\mathscr{W}} \rho} l_\pi(\rho')$ is the probability of getting a wealth level less or equally preferred to ρ when applying policy π. Similarly, the

decumulative distribution induced by l_π is defined as $G^\pi(\rho) = \sum_{\rho \preceq_\mathcal{W} \rho'} l_\pi(\rho')$ is the probability of getting a wealth level at least as good as ρ.

These two notions of cumulative and decumulative distributions enable us to define two quantile criteria.

Definition 39. *Given a policy π, we define the lower τ-quantile for $\tau \in (0,1]$ as:*

$$\underline{q}^\pi_\tau = \min\{\rho \in \mathcal{W}_T \mid F^\pi(\rho) \geq \tau\} \tag{5.1}$$

where the min *operator is with respect to $\prec_\mathcal{W}$.*

Definition 40. *Given a policy π, we define the upper τ-quantile for $\tau \in [0,1)$ as:*

$$\overline{q}^\pi_\tau = \max\{\rho \in \mathcal{W}_T \mid G^\pi(\rho) \geq 1 - \tau\} \tag{5.2}$$

where the max *operator is with respect to $\prec_\mathcal{W}$.*

By construction, we have $\underline{q}^\pi_\tau \preceq_\mathcal{W} \overline{q}^\pi_\tau$. If those two values are equal, q^π_τ is set to their value. For instance, this is always the case in continuous settings for continuous distributions. However, in our discrete setting, it could happen that those values differ, as shown by Example 49.

Example 49. *Consider an MDP where $\mathcal{W}_T = \{\rho_1 \prec_\mathcal{W} \rho_2 \prec_\mathcal{W} \rho_3\}$. Now assume a policy π attains each wealth level with probabilities 0.5, 0.2 and 0.3 respectively. The cumulative distribution F^π (resp. decumulative distribution G^π) induced by policy π is given by $F^\pi(\rho_1) = 0.5$, $F^\pi(\rho_2) = 0.7$ and $F^\pi(\rho_3) = 1$ (resp. $G^\pi(\rho_1) = 1$, $G^\pi(\rho_2) = 0.5$ and $G^\pi(\rho_3) = 0.3$). Then it is easy to see that $\underline{q}^\pi_{0.5} = \rho_1$ whereas $\overline{q}^\pi_{0.5} = \rho_2$.*

When the lower and upper quantiles differ, one may define the quantile as a function of the lower and upper quantiles [Weng, 2012]. For simplicity, we show in this section how to optimize (approximately) the lower and the upper quantiles.

Definition 41. *A policy π^* is optimal for the lower (resp. upper) τ-quantile criterion if:*

$$\underline{q}^{\pi^*}_\tau = \max_\pi \underline{q}^\pi_\tau \quad (\textit{resp. } \overline{q}^{\pi^*}_\tau = \max_\pi \overline{q}^\pi_\tau) \tag{5.3}$$

where the max *operator is with respect to $\prec_\mathcal{W}$ and taken over all policies π at horizon T. For simplicity, we will denote by $\underline{q}^*_\tau$ and $\overline{q}^*_\tau$ the optimal lower and upper quantile respectively.*

Even in a numerical context where a numerical reward function is given and the quality of an episode is defined as the cumulative of rewards received along the episode, this criterion is difficult to optimize, notably due to the two following related points:

- The criterion is *non-linear* with respect to the mixture operation. Stated differently, the τ-quantile $q^{\tilde{\pi}}_\tau$ of the mixed policy $\tilde{\pi}$ that generates an episode using policy π with probability p and π' with probability $1-p$ is not given by $pq^\pi_\tau + (1-p)q^{\pi'}_\tau$.
- The criterion is *non-dynamically consistent*, meaning that at time step t, an optimal policy computed in s_0 with horizon T might not prescribe in state s_t to follow a policy optimal in s_t for horizon $T - t$. Stated differently, it is non consistent with Bellman's optimality principle.

Three solutions are then possible [McClennen, 1990] (these solutions are detailed in subsection 2.2.3) to tackle this latter point:

1. adopting a *sophisticated* approach. At time step $T-1$, we compute an optimal policy in each state for horizon 1. The policy for previous steps are then computed recursively by proceeding backwards. At time step t, we compute an optimal policy for each state at horizon $T-t$ with the constraint that we cannot change our previous choices. With a quantile criterion, the resulting policy may not be optimal for horizon T and initial state s_0;

2. adopting a *resolute choice with root dictatorship* approach, i.e., at time step $t=0$ we apply an optimal policy (viewed from the initial state) for the problem with horizon T and initial state s_0 and do not deviate from it;

3. adopting a *resolute choice with selves* approach [Jaffray, 1998; Jeantet *et al.*, 2012], i.e., we apply a policy π (chosen at the beginning) that trades off between how much π is optimal for all horizons $T, T-1, \ldots, 1$.

With non-dynamically consistent preferences, it is debatable to adopt a sophisticated approach, as the sequence of decisions may lead to dominated policies. In this section, we adopt a resolute choice with root dictatorship point of view.

As optimizing exactly a (lower or upper) quantile is hard, we aim to find an approximate solution.

Definition 42. *Let $\varepsilon > 0$. A policy π^*_ε is said to be ε-optimal for the lower (resp. upper) τ-quantile criterion if $d(\underline{q}_\tau^{\pi^*_\varepsilon}, \underline{q}^*_\tau) \leq \varepsilon$ (resp. $d(\overline{q}_\tau^{\pi^*_\varepsilon}, \overline{q}^*_\tau) \leq \varepsilon$).*

Now that we have introduced formally the quantile criteria and shown how they evaluate the policies, we can present our solution method.

5.2.3 Solution Method

In this subsection, we present a technique for computing an ε-optimal policy for the quantile criterion. The presented approach amounts to solving a sequence of MDPs optimizing EU with target utility functions (which are particular utility functions defined in this subsection).

Binary Search. In order to justify the algorithm, we introduce two lemmas that characterize the optimal lower and upper quantiles:

Lemma 1. *The optimal lower τ-quantile $\underline{q}^*_\tau$ satisfies:*

$$\underline{q}^*_\tau = \min\{\rho : F^*(\rho) \geq \tau\}, \textit{ where} \tag{5.4}$$

$$F^*(\rho) = \min_\pi F^\pi(\rho) \quad \forall \rho \in \mathscr{W} \tag{5.5}$$

Proof. We recall that for any policy π, F^π is nondecreasing and that consequently F^* is also nondecreasing. Let $\rho_1 = \max_\pi \underline{q}^\pi_\tau = \max_\pi \min_\rho\{\rho \in \mathscr{W}_T | F^\pi(\rho) \geq \tau\}$ and let $\rho_2 = \min\{\rho : F^*(\rho) \geq \tau\}$. By contradiction, assume $\rho_1 \succ_{\mathscr{W}} \rho_2$. Then there exists π such that $F^\pi(\rho_1) \geq \tau$ and $F^\pi(\rho) < \tau$, $\forall \rho \prec_{\mathscr{W}} \rho_1$. Thus $F^\pi(\rho_2) < \tau$ which implies that $F^*(\rho_2) < \tau$. This contradicts the definition of ρ_2. Now, assume $\rho_2 \succ_{\mathscr{W}} \rho_1$. Then $F^*(\rho_1) < \tau$. Thus, there exists π such that $F^\pi(\rho_1) < \tau$. Hence, by definition of the lower quantile, $\underline{q}^\pi_\tau \succ_{\mathscr{W}} \rho_1$ which contradicts the definition of ρ_1. □

Note the last two equations can be equivalently rewritten:

$$\underline{q}^*_\tau = \min\{\rho : G^*_\prec(\rho) \leq 1-\tau\}, \text{ where} \tag{5.6}$$

$$G^*_\prec(\rho) = \max_\pi G^\pi_\prec(\rho) \quad \forall \rho \in \mathscr{W} \tag{5.7}$$

where $G^\pi_\prec(\rho) = 1 - F^\pi(\rho) = \sum_{\rho \prec_\mathscr{W} \rho'} l_\pi(\rho')$.

Lemma 2. *The optimal upper τ-quantile $\overline{q}^*_\tau$ satisfies:*

$$\overline{q}^*_\tau = \max\{\rho : G^*(\rho) \geq 1-\tau\}, \textit{ where} \tag{5.8}$$

$$G^*(\rho) = \max_\pi G^\pi(\rho) \quad \forall \rho \in \mathscr{W} \tag{5.9}$$

Proof. We recall that for any policy π, G^π is nonincreasing and that consequently G^* is also nonincreasing. Let $\rho_1 = \max_\pi \overline{q}^\pi_\tau = \max_\pi \max_\rho\{\rho \in \mathscr{W}_T | G^\pi(\rho) \geq 1-\tau\}$ and let $\rho_2 = \max\{\rho : G^*(\rho) \geq 1-\tau\}$. By definition of ρ_1, there exists a policy π such that $G^\pi(\rho_1) \geq 1-\tau$, thus $G^*(\rho_1) \geq 1-\tau$ and $\rho_2 \succeq_\mathscr{W} \rho_1$. By definition of ρ_2, there exists a policy π such that $G^\pi(\rho_2) \geq 1-\tau$, thus $\max_\rho\{\rho \in \mathscr{W}_T | G^\pi(\rho) \geq 1-\tau\} \succeq_\mathscr{W} \rho_2$ and $\rho_1 \succeq_\mathscr{W} \rho_2$. □

Given Lemmas 1 and 2, the problem now reduces to finding the right value of $\rho \in \mathscr{W}$ that solves the problems defined by Equation 5.6 or 5.8.

The solving method that we present to find these values is based on binary search (see Algorithm 9) and on the function $\mathscr{O}(\mathscr{M}, \rho)$ that returns a pair (π, p), the solution of the problems defined by Equation 5.7 or 5.9 for a fixed ρ, i.e., the optimal value returned by $\mathscr{O}(\mathscr{M}, \rho)$ is p and is attained by policy π.

Let us now discuss why finding the optimal quantile value enables us to find an optimal policy. For the upper quantile criterion, it is easy to see that $\mathscr{O}(\mathscr{M}, \overline{q}^*_\tau)$ returns an optimal policy. This is because $\mathscr{O}(\mathscr{M}, \overline{q}^*_\tau)$ will return a policy π for which $G^\pi(\overline{q}^*_\tau) \geq 1-\tau$. However, for the lower quantile, $\mathscr{O}(\mathscr{M}, \underline{q}^*_\tau)$ may not return an optimal policy if $\underline{q}^*_\tau \succ_\mathscr{W} \rho_{\min}$. This fact is illustrated in Figure 5.1. On the left side of the picture, we have drawn an example of function F^*. In this example, we have set τ to 0.5 and we observe that $F^*(\underline{q}^*_\tau) = 0.6$. The cumulative distribution of a policy realizing this minimum is shown on the right side of the figure. It is easy to see that this policy is not optimal with respect to the lower 0.5-quantile. In fact, the 0.5-quantile of this policy is $\rho_{\min}$.

However, $\mathscr{O}(\mathscr{M}, \texttt{prec}(\underline{q}^*_\tau))$ returns an optimal policy where $\texttt{prec}(\rho)$ is the most preferred element $\rho' \in \mathscr{W}_T$ such that $\rho' \prec_\mathscr{W} \rho$. This is because $\mathscr{O}(\mathscr{M}, \texttt{prec}(\underline{q}^*_\tau))$ will return a policy π for which $F^\pi(\texttt{prec}(\underline{q}^*_\tau)) < \tau$ (as $F^*(\texttt{prec}(\underline{q}^*_\tau)) < \tau$). Therefore, by nondecreasingness of the cumulative, $\underline{q}^\pi_\tau \geq \underline{q}^*_\tau$.

Hence, for both quantile criteria, finding the optimal quantile value enables to find an optimal policy. Furthermore, we will see that finding a close lower bound on the optimal quantile value is enough to find a near-optimal policy. Our binary search method determines such a lower bound.

When $\mathscr{W}_T$ is defined on $\mathbb{R}$, Algorithm 9 needs only $\lceil \log_2 d(\rho_{\max}, \rho_{\min})/\varepsilon \rceil$ iterations to terminate by using $[\rho_{\min}, \rho_{\max}]$ as $\mathscr{W}_T$. In the case where $\mathscr{W}_T$ is finite, the binary search method can of course determine the optimal policy with $\varepsilon = 1$ and needs $\lceil \log_2(|\mathscr{W}_T|) \rceil$ iterations.

Before proving that Algorithm 9 is correct, we introduce a lemma that gives sufficient conditions for a policy to be approximately optimal.

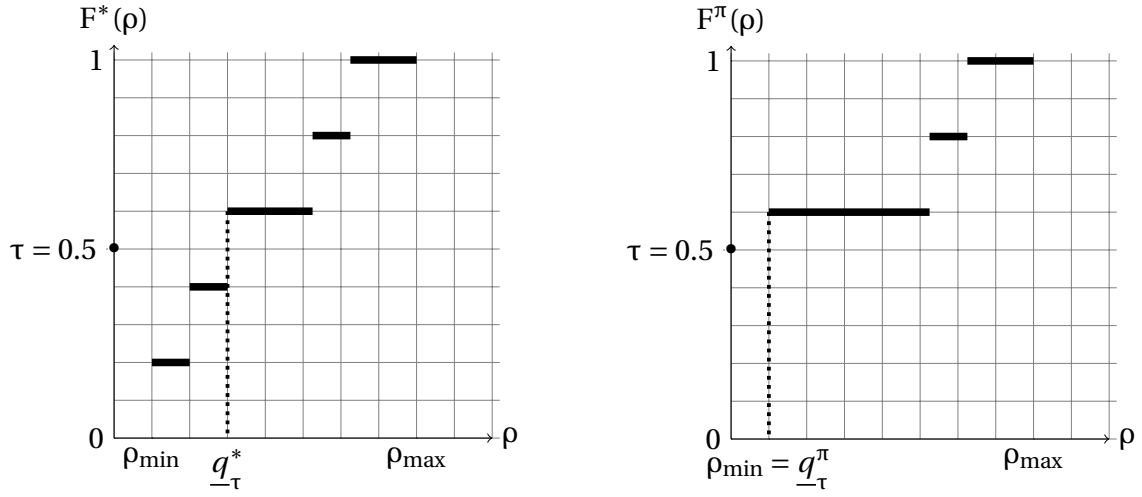


Figure 5.1: Why the subroutine applied to the optimal lower quantile may not return an optimal policy with respect to the lower quantile.

Algorithm 9: Binary Search for the Lower Quantile (resp. Upper Quantile)

Data: MDP $\mathscr{M}$, τ, ε
Result: an ε-optimal policy π

$\overline{\rho} \leftarrow \rho_{\max}$; $\underline{\rho} \leftarrow \rho_{\min}$; $\rho \leftarrow \texttt{mid}(\underline{\rho}, \overline{\rho})$
while $d(\overline{\rho}, \underline{\rho}) > \varepsilon$ **do**
  $(\pi, p) = \mathscr{O}(\mathscr{M}, \rho)$;
  if $p > 1-\tau$ *(resp.* $p \geq 1-\tau$*)* **then**
    $\underline{\rho} \leftarrow \rho$; $\rho \leftarrow \max(\texttt{mid}(\underline{\rho}, \overline{\rho}))$;
    $\pi^* \leftarrow \pi$;
  else
    $\overline{\rho} \leftarrow \rho$; $\rho \leftarrow \min(\texttt{mid}(\underline{\rho}, \overline{\rho}))$;
return π^*

Lemma 3. *Let π be a policy for which there exists ρ such that $d(\rho, \underline{q}^*_\tau) \leq \varepsilon$ (resp. $d(\rho, \overline{q}^*_\tau) \leq \varepsilon$) and:*

$$F^\pi(\rho) < \tau \quad (\textit{resp. } G^\pi(\rho) \geq 1-\tau).$$

Then π is ε-optimal for the lower (resp. upper) τ-quantile criterion.

Proof. Let $[\rho, \rho'']$ denote the set of wealth values ρ' such that $\rho \preceq_{\mathscr{W}} \rho' \preceq_{\mathscr{W}} \rho''$. Let π be a policy that satisfies the condition of the lemma. For such a policy, as $F^\pi(\rho)$ is nondecreasing (resp. $G^\pi(\rho)$ is nonincreasing), we have that $\underline{q}^\pi_\tau \in [\rho, \underline{q}^*_\tau]$ (resp. $\overline{q}^\pi_\tau \in [\rho, \overline{q}^*_\tau]$) and thus $d(\underline{q}^\pi_\tau, \underline{q}^*_\tau) \leq d(\rho, \underline{q}^*_\tau) \leq \varepsilon$ (resp. $d(\overline{q}^\pi_\tau, \overline{q}^*_\tau) \leq d(\rho, \overline{q}^*_\tau) \leq \varepsilon$) . □

The next proposition asserts that Algorithm 9 is correct:

Proposition 11. *Algorithm 9 returns an ε-optimal policy for the lower (resp. upper) quantile criterion.*

Proof. If $\mathscr{W}_T \subset \mathbb{R}$ we have seen that the algorithm terminates in $\left\lceil \log_2 \frac{d(\rho_{\max}, \rho_{\min})}{\varepsilon} \right\rceil$ iterations. In the most general setting, the algorithm terminates, because in the worst case we will

check all m possible final wealth values. Let π be the policy returned by the algorithm. For the lower (resp. upper) quantile, when the algorithm terminates, $d(\underline{q}^*_\tau, \underline{\rho})$ (resp. $d(\overline{q}^*_\tau, \underline{\rho})$) $\leq d(\overline{\rho}, \underline{\rho}) \leq \varepsilon$ and $F^{\pi^*}(\underline{\rho}) < \tau$ (resp. $G^\pi(\underline{\rho}) \geq 1-\tau$). Thus, we can apply Lemma 3 which concludes the proof. □

We now see how step $\mathcal{O}(\mathcal{M}, \rho)$ can be performed.

Dynamic Programming. For $\lhd \in \{\prec_\mathcal{W}, \preceq_\mathcal{W}\}$, we denote by $V^\lhd_\rho : \mathcal{W} \to \mathbb{R}$ the function, called *target utility function*, defined as follows:

$$V^\lhd_\rho(x) = 1 \text{ if } \rho \lhd x \text{ and } 0 \text{ else.} \tag{5.10}$$

When optimizing the lower (resp. upper) quantile, function $\mathcal{O}(\mathcal{M}, \rho)$ can be computed by solving MDP $\mathcal{M}$ using EU as a decision criterion with $V^{\prec_\mathcal{W}}_\rho$ (resp. $V^{\preceq_\mathcal{W}}_\rho$) as a utility function. Indeed, we have:

$$\mathbb{E}_\pi[V^\lhd_\rho\big(\rho(H_T)\big)] = \mathbb{P}[\rho \lhd \rho(H_T) \,|\, \pi]$$

where H_T is a random variable representing a T-history and $\mathbb{P}[\rho \lhd \rho(H_T) \,|\, \pi]$ denotes the probability that π generates a history whose wealth level is strictly better than (resp. at least as good as) ρ when $\lhd = \prec_\mathcal{W}$ (resp. $\lhd = \preceq_\mathcal{W}$).

Following Liu and Koenig [2006], this problem can be solved with a functional backward induction (Algorithm 10). For each state s, it maintains a function $v_t(s,.)$ which associates to each possible wealth level ρ the expected utility obtained by applying an optimal policy in state s for the remaining $T-t$ time steps with ρ as initial wealth level [1]. At each time step ($t = T-1, \ldots, 0$) this function is updated similarly as in backward induction except that operations are not applied to scalars but to functions. The max and $\times$ operations are extended over functions as pointwise operations. As utility functions defined by Equation 5.10 are piecewise-linear, $v_t(s,.)$ is also piecewise-linear because all the operations in Line 5 of Algorithm 10 preserve this property.

Algorithm 10: Functional Backward Induction [Liu and Koenig, 2006]

Data: MDP $\mathcal{M}$, wealth ρ
Result: an optimal policy π and the optimal expected utility value from s_0

1 **for** $s \in \mathcal{S}$ **do**
2 $\quad v_T(s,.) \leftarrow V^\lhd_\rho(.)$
3 **for** $t = T-1$ `downto 0` **do**
4 $\quad$ **for** $s \in \mathcal{S}$ **do**
5 $\quad\quad v_t(s,\cdot) \leftarrow \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, v_{t+1}(s', \cdot \circ c(s,a,s'))$
6 **return** $(\pi_{v_0}, v_0(s_0, \rho_0))$ \\ π_{v_0}= policy corresponding to v_0

The policies returned by Algorithm 10 have a special structure. They are deterministic and wealth-Markovian. Besides, an optimal policy with respect to the quantile criterion can always be found as a deterministic wealth-Markovian policy.

Proposition 12. *In finite horizon MDPs, with finite state and action spaces, an optimal policy for the lower or upper quantile can be found as a deterministic wealth-Markovian policy.*

[1] This is equivalent as using a standard backward induction method in an augmented MDP as defined in subsection 4.3.2 and with a reward function induced by utility function $V^\lhd_\rho$.

Proof. We recall that for the lower (resp. upper) quantile criterion, procedure $\mathcal{O}(\mathcal{M}, \rho)$ returns the policy which minimizes $F^{\pi}(\rho)$ (resp. maximizes $G^{\pi}(\rho)$). Thus, for any policy π, by definition of quantiles, $\mathcal{O}(\mathcal{M}, \texttt{prec}(\underline{q}^{\pi}_{\tau}))$ (resp. $\mathcal{O}(\mathcal{M}, \overline{q}^{\pi}_{\tau})$) returns a deterministic wealth-Markovian policy, which is at least as good as π regarding the lower (resp. upper) quantile criterion. As the set of deterministic wealth-Markovian policies is finite in the finite horizon case, taking the one with highest lower (resp. upper) quantile concludes the proof. □

We now turn to numerical tests where our approach is evaluated.

5.2.4 Numerical Tests

The approach was experimentally evaluated on a server equipped with four Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz and 64Gb of RAM. The algorithms were implemented in Matlab and ran only on one core.

Three sets of experiments where designed to test the approach. The first set of experiments shows the running time of functional backward induction for different varying state sizes on random MDPs. The second set of experiments shows the running time of functional backward induction for different horizons on a data center control problem with various number of servers. Finally, the third set of experiments compares for a fixed MDP the cumulative distributions of an optimal policy for the quantile criterion with an optimal policy for the standard criterion.

The first set of experiments was conducted on Garnets [McKinnon and Thomas, 1995], which designate random MDPs with a constrained branching factor. A Garnet $G(n_S, n_A, b)$ is characterized by n_S a number of states, n_A a number of actions and b the number of successor states for every state and action. For our experiments, $n_S \in \{250, 500, 750, 1000, 1250, 1500, 1750, 2000, 2250\}$ and we set $n_A = 5$ and $b = \lceil \log_2 n_S \rceil$. Rewards are randomly chosen in $[0, 1]$ and the values of histories are simply cumulated rewards. The horizon of the problem was set to 5 and the target of the utility function was set to $R_{max}T/2$, where R_{max} is the maximal reward value of the MDP. The results are presented in Figure 5.2 where the x-axis represents the state size and the y-axis the computation time. Each point is an average over 10 runs. Naturally, computation times increases with state sizes. In this setting, the binary search method would call functional backward induction $\lceil \log_2(1/\varepsilon) \rceil = 10$ times if $\varepsilon = 10^{-3}$.

The second set of experiments was performed on a more realistic domain, which is a data center control problem inspired by the model proposed by Yin and Sinopoli [2014]. In this problem, one needs to decide every time step how many servers to switch on or off, while maximizing Quality-of-Service and minimizing power consumption. In the model proposed by Yin and Sinopoli [2014], the two objectives are simply combined into one cost, which defines our reward function. The state is defined as the number of servers that are currently on and the number of jobs that needs to be processed during a time step. The action represents the number of servers that will be on at the next time step. We assume for simplicity that the maximum number of jobs that can arrive at one time step, denoted by J, is three times the total number of servers. For instance, in a problem with $n = 30$ servers, the total number of states is $J \times n = 30 \times 3 \times 30 = 2700$. Besides, the distribution of the next number of jobs is modeled as a Poisson distribution whose parameter can be $\lceil J/6 \rceil$, $\lceil J/2 \rceil$ or $\lceil 5J/6 \rceil$ (to model different regimes) depending on the current number of jobs. The target of the utility function was set as in the first set of experiments. Figure 5.3 shows the computation times of functional backward induction for $n \in \{20, 30, 40\}$ and

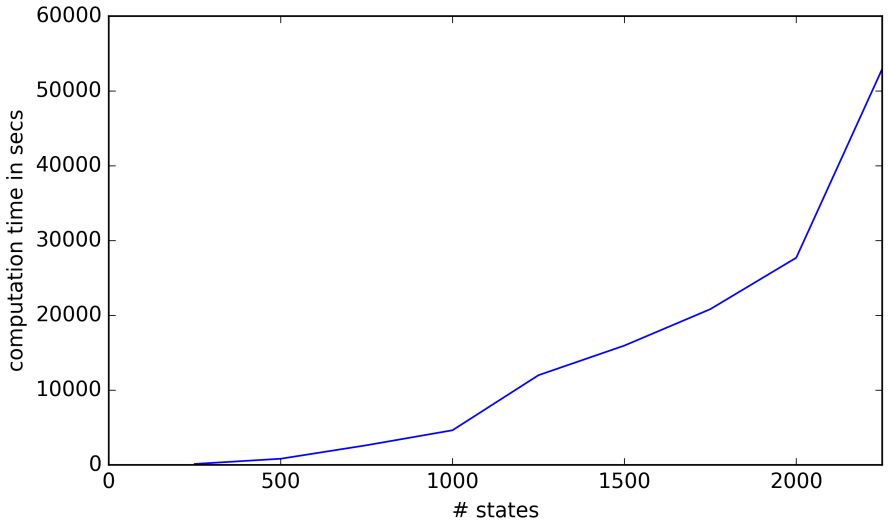


Figure 5.2: Computation times vs state sizes for Functional Backward Induction.

different horizons. We can see that for more structured problems, the computation time is much more reasonable than on random MDPs.

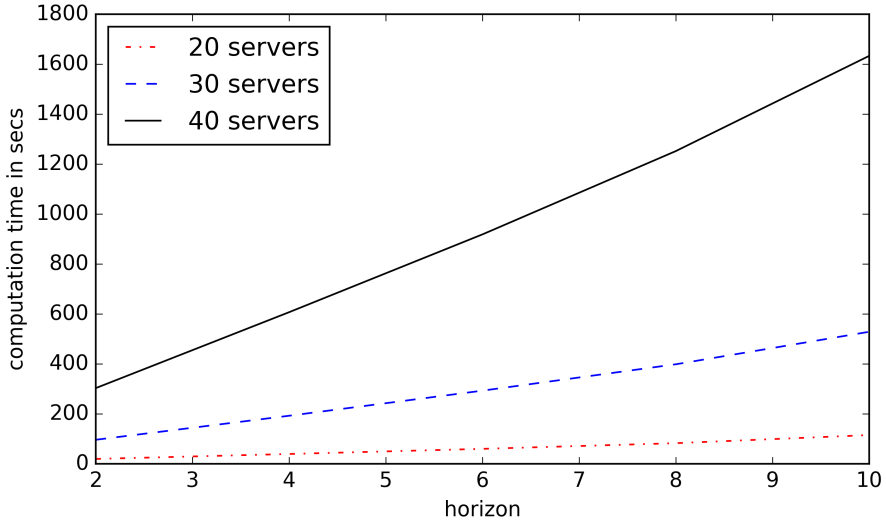


Figure 5.3: Computation times vs horizon for Functional Backward Induction.

In the last set of experiments, to give an intuition of the kind of policy obtained when optimizing a quantile, we compare the cumulative distribution of an optimal policy for the quantile criterion with the cumulative distribution of an optimal policy for the standard criterion. This experiment is performed on an instance of Garnet $G(100, 5, \lceil \log_2 100 \rceil)$ whose rewards are slightly modified to make the distribution of the optimal policy skewed, as it is often the case in some real applications [Benoit and Van den Poel, 2009]. The horizon is set to 5 and we optimize the 0.1-quantile with $\varepsilon = 0.001$ in the binary search method. The two cumulative distributions are plotted in Figure 5.4. We can observe that although the optimal policy for the standard criterion maximizes the expectation, it may be a more risky policy to apply. Indeed, the 10% worst final wealth values obtained with the opti-

mal policy for the 0.1-quantile criterion are better than the 10% worst final wealth values obtained with the optimal policy maximizing expectation.

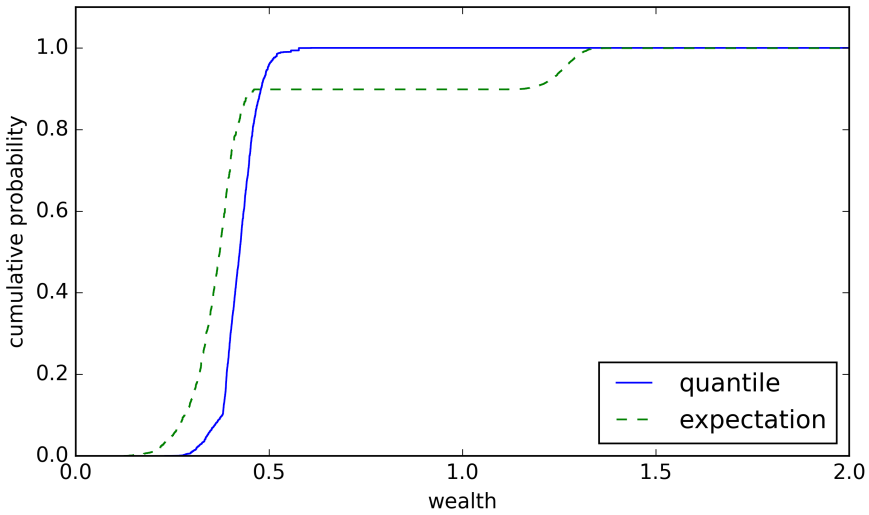


Figure 5.4: Comparison of cumulative distributions under the quantile criterion and the standard criterion

5.2.5 Summary of Section: MDPs with a Quantile Criterion

In this section, we have developed a framework to solve sequential decision problems in a very general setting according to a quantile criterion. Modeling those problems as MDPs, an algorithm was developed in order to compute an ϵ-optimal policy, and we investigated the properties of the optimal policies in the finite horizon case. Lastly, we provided experimental results, testing this approach.

In the next section, we explore another framework that does not require to know the exact values of the reward parameters. We keep the standard expected discounted reward criterion and we assume that only an order relation over the rewards is known. We then require the help of a tutor to learn the minimum amount of information required to find the optimal policy.

5.3 Sequential Decision Making under Risk with Imprecise Reward Values: an Interactive Value Iteration Algorithm for MDPs

In the previous section, we designed a solution method according to a quantile criterion that only assumes a weak order on episodes. This method can be used to alleviate the difficulty of defining the reward function exactly, when this task has to be done by a human expert. We now return to the standard expectation of total discounted reward criterion and wish to help the expert in another way. More precisely, our objective is to solve the MDP while asking the expert the minimum amount of information about the reward function.

Following this line of research, Weng and Zanuttini [2013], revisited a well known algorithm for solving MDPs, namely value iteration, by incorporating the elicitation process in the solving procedure. In this new algorithm called *Interactive Value Iteration* (IVI), a human tutor is queried about multi-sets of rewards when the information acquired so far does not allow to continue solving the MDP. This procedure is appealing as answering comparison queries is much less cognitively demanding than providing the reward function to optimize. However, in the original IVI procedure, one does not try to explicitly minimize the number of queries issued, which may lead to a prohibitive effort for the tutor.

In this section, we address the problem of modifying IVI in order to reduce the number of queries issued. To that aim, we propose a variant of IVI based on the ideas that 1) delaying the queries open the possibility that some of them meanwhile become unnecessary, 2) the order in which the queries are asked matters, and this order should be optimized, 3) queries can be avoided by using heuristic information to guide the iteration process. We show empirically that these combined techniques can greatly reduce the number of queries issued.

After recalling related works (subsection 5.3.1) and the main features of IVI (subsection 5.3.2), we present the proposed optimizations (subsection 5.3.3). Finally, we provide the results of numerical tests that show a significant decrease in the number of queries issued (subsection 5.3.4).

5.3.1 Related Work

The difficulty for a human agent to define precisely the reward function of an MDP has motivated much work. We distinguish four main approaches, although their boundaries may be blurry.

Robust approach. In the first approach, the parameters of the MDP (i.e., rewards and possibly also probabilities) are assumed to be imprecisely known. A natural way [Bagnell *et al.*, 2001; Givan *et al.*, 2000] to handle such situation is to search for robust solutions, i.e., solutions that are as good as possible even in the worst case. However, this method often leads to solutions that are too pessimistic. An alternative approach is based on the optimization of a minmax regret criterion [Xu and Mannor, 2009]. Here, one tries to minimize the gap between the value of the best policy (after the true reward values are revealed) and that of the chosen policy. However, this leads to NP-hard problems.

Preference learning. Another approach, which has been mainly developed for reinforcement learning (i.e., a context more general than MDPs), aims to learn the reward values, either from demonstrations (live [Abbeel and Ng, 2004] or from recorded logs [Piot *et al.*, 2014]) or from interactions with a human tutor [Thomaz *et al.*, 2005]. This domain is also called inverse reinforcement learning and is a hot topic in artificial intelligence [Finn *et al.*, 2016; Jin *et al.*, 2017; Levine *et al.*, 2011; Ramachandran and Amir, 2007; Syed, 2010]. One drawback of this approach is that it generally assumes that demonstrations from human tutors can be easily translated into the state/action representation of the learning agent, which may be difficult as humans and agents evolve in different state/action spaces.

Multiple Policy Learning. If the parameters of the MDP are unknown, one option is to consider several scenarios and to learn all or a set of optimal policies with respect to

these possible scenarios [Barrett and Narayanan, 2008; Lizotte *et al.*, 2012; Van Moffaert *et al.*, 2014; Wiering and De Jong, 2007]. This line of research can also be related to the work of Fard and Pineau [2011] where the goal is to determine a policy which maps states to subsets of "good" actions from which the decision maker can make a choice.

Preference elicitation. A final approach assumes a human tutor is present and the agent may query her to get more precise information about reward values. In a series of papers, Regan and Boutilier [2009, 2010, 2011a,b] show how to compute policies which optimize minmax regret with respect to all candidate reward functions, and discuss how this criterion can be used to generate informative queries to ask the tutor about the true reward function. This work was further developed by Freire da Silva and Reali Costa [2011] and Alizadeh *et al.* [2015]. Iteratively issuing such queries is shown to allow convergence to an optimal policy for this function. However, the problem of computing such robust policy is NP-hard [Xu and Mannor, 2009] and their algorithm issues bound queries which can be cognitively difficult to answer.

The work on Interactive Value Iteration (IVI) [Weng and Zanuttini, 2013] falls in the scope of the preference elicitation approach. IVI resembles the standard value iteration procedure. However, it works with an incompletely specified reward function and queries the decision maker about her preferences when this uncertainty prevents the resolution procedure from continuing. This approach was further extended by Weng *et al.* [2013] to the case where the tutor could make mistakes, and also by Alizadeh *et al.* [2016], who investigated the use of the notion of *advantages* in MDPs and clustering methods to improve IVI.

5.3.2 Background

Markov Decision Process. In this section, we consider an infinite horizon MDP defined as a tuple $\mathcal{M} = (\mathcal{S},\mathcal{A},\mathcal{P},c,\gamma)$ where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{P}$ is a transition function with $\mathcal{P}(s'|s,a)$ being the probability of reaching state s' when action a is performed in state s, $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function and $\gamma \in [0,1[$ is a discount factor.

We wish to find a policy according to the expected total discounted reward criterion. We recall that, with this criterion, a stationary, deterministic policy $\pi : \mathcal{S} \to \mathcal{A}$ is evaluated by a value function $v^\pi : \mathcal{S} \to \mathbb{R}$ and a Q-*function*, $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ defined as follows:

$$v^\pi(s) = c(s,\pi(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,\pi(s))\, v^\pi(s') \tag{5.11}$$

$$Q^\pi(s,a) = c(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, v^\pi(s') \tag{5.12}$$

A solution to the MDP is a policy, with maximal evaluation. Such a policy can be found by solving the *Bellman equations*.

$$v^*(s) = \max_{a \in \mathcal{A}} c(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, v^*(s') \tag{5.13}$$

In a setting where the reward function is not known with certainty, defining the value function and optimizing it as done with the Bellman equations can be problematic. In this section, we will query ordinal information from a tutor to unveil the maximal actions.

Ordinal Reward MDP. In this section, we suppose that there exists a standard numerical reward function. However, while the rewards' numerical values are assumed to be unknown, we suppose that their relative order is given. Such situation can be represented as an *Ordinal Reward MDP* (ORMDP) [Weng, 2011] defined by a tuple $(\mathscr{S}, \mathscr{A}, \mathscr{P}, \hat{c}, \gamma)$ where the reward function $\hat{c} : \mathscr{S} \times \mathscr{A} \to \mathscr{R}$ takes its values in a set $\mathscr{R} = \{c_1 < c_2 \ldots < c_k\}$ of unknown ordered rewards.

In order to count the number of each unknown reward obtained by a policy, an ORMDP can be reformulated as a Vector Reward MDP (VMDP) $(\mathscr{S}, \mathscr{A}, \mathscr{P}, \mathbf{c}, \gamma)$ where $\mathbf{c}(s,a)$ is the vector in $\mathbb{R}^k$ whose i^{th} component is 1 for $\hat{c}(s,a) = c_i$, and 0 on the other components. Like in standard MDPs, in such a VMDP the value function $\mathbf{v}^\pi$ and the Q-function $\mathbf{Q}^\pi$ of a policy π can be defined by:

$$\mathbf{v}^\pi(s) = \mathbf{c}(s,\pi(s)) + \gamma \sum_{s' \in \mathscr{S}} \mathscr{P}(s'|s,\pi(s))\mathbf{v}^\pi(s') \tag{5.14}$$

$$\mathbf{Q}^\pi(s,a) = \mathbf{c}(s,a) + \gamma \sum_{s' \in \mathscr{S}} \mathscr{P}(s'|s,a)\mathbf{v}^\pi(s') \tag{5.15}$$

where additions and multiplications are componentwise. The i-th component of a vector $\mathbf{v} \in \mathbb{R}^k$ can be interpreted as the expected and discounted number of unknown reward c_i.

Therefore, a value function in a state can be interpreted as a multi-set or a bag of elements of $\mathscr{R}$ represented by a vector. Now, comparing policies amounts to comparing vectors. Interactive value iteration [Weng and Zanuttini, 2013] is a procedure inspired from value iteration to find an optimal policy according to the true unknown reward function by querying when needed an expert for comparisons of two bags of elements of $\mathscr{R}$. We now present the original version of the algorithm by Weng and Zanuttini.

Interactive Value Iteration. In order to find an (approximate) optimal policy for an ORMDP with an initially unknown preference relation over vectors, Weng and Zanuttini [2013] developed a variant of value iteration, named Interactive Value Iteration (IVI), where the agent may ask a tutor which of two sequences of rewards should be preferred. An example of query is "$c_1 + c_3 \geq 2c_2$?", meaning "is receiving c_1 and c_3 as good as receiving $2c_2$?". As the tutor answers queries, the set of admissible reward functions shrinks and more vectors can be compared without querying the tutor.

At the beginning of the process, the agent only knows the order over rewards, i.e., $c_1 < c_2 < \ldots < c_k$, and knows that she can set without loss of generality c_1 to 0 and c_k to 1 [Weng, 2012]. This initial knowledge is represented by a set $\mathscr{K}$ of linear constraints. When having to choose the best of two value vectors $\mathbf{v}$ and $\mathbf{v}'$ in a given state (see Algorithm 12), function $StDom(\mathbf{v},\mathbf{v}')$ compares the two vectors with respect to the following dominance relation named st-dominance (which is analogous to stochastic dominance):

$$\forall\, j = 1,\ldots,k \;\; \sum_{i=j}^{k} v_i \geq \sum_{i=j}^{k} v'_i \tag{5.16}$$

Put another way, a vector $\mathbf{v}$ st-dominates a vector $\mathbf{v}'$ if for each reward value c_i the expected and discounted number of rewards that are greater than or equal to c_i is higher in $\mathbf{v}$ than in $\mathbf{v}'$. In this case for any reward values $\mathbf{c}$ consistent with the constraint $c_1 = 0 < c_2 < \ldots < c_k = 1$, we have that $\mathbf{v}\cdot\mathbf{c} \geq \mathbf{v}'\cdot\mathbf{c}$. In the case where no dominance is found, the next step is to check whether one of the two vectors is necessarily preferred to the other given the constraints in $\mathscr{K}$. The function $KDom(\mathbf{v},\mathbf{v}')$ checks whether $\mathbf{v}$ is not less preferred than $\mathbf{v}'$

(denoted by $\mathbf{v} \succeq \mathbf{v}'$) by solving the following linear program:

$$z^* = \min (\mathbf{v} - \mathbf{v}') \cdot \mathbf{c} \tag{5.17}$$

$$\text{s.t.} \quad \mathbf{c} \in \mathscr{C}(\mathscr{K}) \tag{5.18}$$

where the dot in (5.17) denotes the inner product and $\mathscr{C}(\mathscr{K})$ is the set of reward values $\mathbf{c} = (c_1, \ldots, c_k)$ satisfying all constraints in $\mathscr{K}$. We distinguish the following three cases:

1. A non-negative optimal value for the objective function z^* implies that for any possible reward function, $\mathbf{v} \succeq \mathbf{v}'$; $\mathbf{v}$ is then said to KDominate $\mathbf{v}'$.

2. In case of negativity of the optimal value z^*, $KDom(\mathbf{v}', \mathbf{v})$ is called to check if $\mathbf{v}' \succeq \mathbf{v}$ for all reward function satisfying constraints in $\mathscr{K}$.

3. If the two vectors can still not be compared, the tutor is asked which of $\mathbf{v}$ or $\mathbf{v}'$ should be preferred. A query is a 2-subset $\{\mathbf{v}, \mathbf{v}'\}$ from which the tutor picks her preferred element (if indifferent, she arbitrarily picks one of them). If she picks $\mathbf{v}$, then $\mathbf{v} \succeq \mathbf{v}'$, otherwise $\mathbf{v}' \succeq \mathbf{v}$. If $\mathbf{v} \succeq \mathbf{v}'$ (resp. $\mathbf{v}' \succeq \mathbf{v}$), then the new constraint $(\mathbf{v} - \mathbf{v}').\mathbf{c} \geq 0$ (resp. $(\mathbf{v}' - \mathbf{v}).\mathbf{c} \geq 0$) is added to $\mathscr{K}$.

Algorithm 11 summarizes the IVI procedure. It uses the functions *Init*, that returns the initial set $\mathscr{K}$ of linear constraints induced by $c_1 < \ldots < c_k$, and *getBest* (Algorithm 12) that returns the best of two vectors. The function *getBest* calls first function *StDom*, then function *KDom*, and finally function query (Algorithm 13) if no dominance is found. If function query is called than a new constraint is added to $\mathscr{K}$ which reduces the set of admissible reward values. Note that $StDom(\mathbf{v}, \mathbf{v}')$=$KDom(\mathbf{v}, \mathbf{v}')$ for initial set $\mathscr{K}$, and that $StDom(\mathbf{v}, \mathbf{v}') \Rightarrow KDom(\mathbf{v}, \mathbf{v}')$ in all cases. Nevertheless, the IVI procedure takes advantage of the computational efficiency of *StDom* to save some calls to *KDom*.

Weng and Zanuttini [2013] showed that the number of queries used by IVI is polynomial in the size of the MDP. However, as the tutor is queried each time two vectors cannot be compared, it seems that a more sophisticated approach could greatly reduce the number of queries needed by the algorithm. In the next subsection, a modified version of IVI is proposed that integrates several techniques in order to reduce the number of queries.

5.3.3 Modified Interactive Value Iteration

Interactive Value Iteration is appealing as answering comparison queries is significantly less cognitively demanding than directly defining the reward function. And yet, to be a reasonable method, the number of queries required by the procedure needs to be as low as possible. Several ideas are investigated to address this problem: 1) delaying queries in order to avoid asking some unnecessary queries; 2) prioritizing queries in order to ask the most informative ones first; 3) allowing small mistakes in the dominance tests in the early stages in order to anticipate the shrinking of the set of admissible reward functions.

Delaying the queries. In Algorithm 11, the tutor is queried whenever two vectors $\mathbf{v}$ and $\mathbf{v}'$ cannot be compared. The intuition behind the improvement proposed here is that by delaying the query phase after all vectors are generated (each vector refers to a possible action in a given state), we might benefit from the fact that new dominance relations might appear later on. Indeed when trying to assess which of three vectors $\mathbf{v^1}, \mathbf{v^2}$, and $\mathbf{v^3}$ is the best, we may be able to find that $\mathbf{v^3}$ dominates both $\mathbf{v^1}$ and $\mathbf{v^2}$; this is enough for us, even if we ignore which is better between $\mathbf{v^1}$ and $\mathbf{v^2}$. Therefore querying which of $\mathbf{v^1}$ and $\mathbf{v^2}$

Algorithm 11: Interactive Value Iteration [Weng and Zanuttini, 2013]

Data: $\mathscr{S}, \mathscr{A}, \mathscr{P}, \hat{c}, \gamma, \mathscr{R}, \epsilon$
1 $t \leftarrow 0$
2 compute $\mathbf{c}$ from $\hat{c}$
3 $\mathscr{K} \leftarrow Init(\mathscr{R})$
4 **for** $s \in \mathscr{S}$ **do** $\mathbf{v}_0(s) \leftarrow (0,\ldots,0)$
5 **repeat**
6 $\quad t \leftarrow t+1$
7 $\quad$ **for** $s \in \mathscr{S}$ **do**
8 $\quad\quad$ **best** $\leftarrow (0,\ldots,0)$
9 $\quad\quad$ **for** $a \in \mathscr{A}$ **do**
10 $\quad\quad\quad \mathbf{v} \leftarrow \mathbf{c}(s,a) + \gamma \sum_{s' \in \mathscr{S}} \mathscr{P}(s'|s,a)\mathbf{v}_{t-1}(s')$
11 $\quad\quad\quad$ (**best**, $\mathscr{K}$) $\leftarrow getBest(\mathbf{best}, \mathbf{v}, \mathscr{K})$
12 $\quad\quad$ $\mathbf{v}_t(s) \leftarrow$ **best**
13 **until** $||\mathbf{v}_t - \mathbf{v}_{t-1}|| < \epsilon$
14 **return** $\mathbf{v}_t$

Algorithm 12: getBest

Data: $\mathbf{v}, \mathbf{v}', \mathscr{K}$
1 **if** $StDom(\mathbf{v}, \mathbf{v}')$ **then return** $(\mathbf{v}, \mathscr{K})$
2 **if** $StDom(\mathbf{v}', \mathbf{v})$ **then return** $(\mathbf{v}', \mathscr{K})$
3 **if** $KDom(\mathbf{v}, \mathbf{v}', \mathscr{K})$ **then return** $(\mathbf{v}, \mathscr{K})$
4 **if** $KDom(\mathbf{v}', \mathbf{v}, \mathscr{K})$ **then return** $(\mathbf{v}', \mathscr{K})$
5 (**best**, $\mathscr{K}$) $\leftarrow$ query$(\mathbf{v}, \mathbf{v}', \mathscr{K})$
6 **return** (**best**, $\mathscr{K}$)

Algorithm 13: query

Data: $\mathbf{v}, \mathbf{v}', \mathscr{K}$
1 Ask query $\{\mathbf{v}, \mathbf{v}'\}$
2 **if** $\mathbf{v} \geq \mathbf{v}'$ **then**
3 $\quad$ **return** $(\mathbf{v}, \mathscr{K} \cup \{(\mathbf{v}-\mathbf{v}').\mathbf{c} \geq 0\})$
4 **else**
5 $\quad$ **return** $(\mathbf{v}', \mathscr{K} \cup \{(\mathbf{v}'-\mathbf{v}).\mathbf{c} \geq 0\})$

is best, as would do Algorithm 11, is unnecessary. Thus, delaying the querying step from the loop over actions will prevent asking some unnecessary queries. For this purpose, we replace Lines 7 to 12 in Algorithm 11 by the following lines:

1 **for** $s \in \mathscr{S}$ **do**
2 $\quad \mathbf{Q}(s) \leftarrow \emptyset$
3 $\quad$ **for** $a \in \mathscr{A}$ **do**
4 $\quad\quad \mathbf{Q}(s,a) \leftarrow \mathbf{c}(s,a) + \gamma \sum_{s' \in \mathscr{S}} \mathscr{P}(s'|s,a)\mathbf{v}_{t-1}(s')$
5 $\quad\quad \mathbf{Q}(s) \leftarrow \mathbf{Q}(s) \cup \{\mathbf{Q}(s,a)\}$
6 $\quad \mathbf{Q}(s) \leftarrow$ StDFilter$(\mathbf{Q}(s))$
7 $\quad \mathbf{Q}(s) \leftarrow$ KDFilter$(\mathbf{Q}(s), \mathscr{K})$
8 $\quad$ **while** $|\mathbf{Q}(s)| > 1$ **do**
9 $\quad\quad$ Let $\{\mathbf{v}, \mathbf{v}'\} \subseteq \mathbf{Q}(s)$
10 $\quad\quad$ $(\_, \mathscr{K}) \leftarrow$ query$(\mathbf{v}, \mathbf{v}', \mathscr{K})$
11 $\quad\quad \mathbf{Q}(s) \leftarrow$ KDFilter$(\mathbf{Q}(s), \mathscr{K})$
12 $\quad \mathbf{v}_t(s) \leftarrow$unique vector of $\mathbf{Q}(s)$

where primitives *StDFilter* and *KDFilter* are given in Algorithms 14 and 15: the former checks the dominance described by Equation 5.16 for each pair of vectors $\mathbf{v}, \mathbf{v}'$ in $\mathbf{Q}$ and returns the set of undominated vectors; the latter does the same thing for K-dominance.

Basically, $\mathbf{Q}(s)$ is a set of vectors (discounted collections of unknown rewards) associated to a state s while $\mathbf{Q}(s,a)$ is related to the value of taking an action a in state s. As in the original IVI, we filter out st-dominated and K-dominated vectors before starting the querying phase. Finally, in the **while** loop we query the user about pairs of non-dominated vectors in $\mathbf{Q}(s)$ until $\mathbf{Q}(s)$ contains a single element, that is assigned to $\mathbf{v}_t(s)$ in the last line. Note that, after each query, we try to find new K-dominance relations by using method *KDFilter* (Line 11). This idea of delaying the queries can be pushed further

by delaying the querying phase outside of the loop over states or by waiting several time steps before asking queries.

Algorithm 14: StDFilter

Data: $\mathbf{Q}$
1 **for** $\mathbf{v} \in \mathbf{Q}$ **do**
2 **for** $\mathbf{v}' \in \mathbf{Q}$ *with* $\mathbf{v} \neq \mathbf{v}'$ **do**
3 **if** $StDom(\mathbf{v}, \mathbf{v}')$ **then**
  $\mathbf{Q} \leftarrow \mathbf{Q} \setminus \{\mathbf{v}'\}$
4 **return** $\mathbf{Q}$

Algorithm 15: KDFilter

Data: $\mathbf{Q}, \mathscr{K}$
1 **for** $\mathbf{v} \in \mathbf{Q}$ **do**
2 **for** $\mathbf{v}' \in \mathbf{Q}$ *with* $\mathbf{v} \neq \mathbf{v}'$ **do**
3 **if** $KDom(\mathbf{v}, \mathbf{v}', \mathscr{K})$ **then**
  $\mathbf{Q} \leftarrow \mathbf{Q} \setminus \{\mathbf{v}'\}$
4 **return** $\mathbf{Q}$

Prioritizing the queries. By delaying the queries outside of the loop over actions and even outside of the loop over states, one can choose to ask queries in a different order than the sequential one induced by the original IVI procedure. Certainly this order will count as queries are not all equally informative: the queries that we presume might solve many others should be asked first. Defining a relevance score for each query to guide the querying process seems to be a promising technique to curb the number of queries necessary to solve the MDP. For this purpose, we now replace Lines 7 to 12 in Algorithm 11 by the following lines:

1 **for** $s \in \mathscr{S}$ **do**
2 $\mathbf{Q}(s) \leftarrow \emptyset$
3 **for** $a \in \mathscr{A}$ **do**
4 $\mathbf{Q}(s,a) \leftarrow \mathbf{c}(s,a) + \gamma \sum_{s' \in \mathscr{S}} \mathscr{P}(s'|s,a)\mathbf{v}_{t-1}(s')$
5 $\mathbf{Q}(s) \leftarrow \mathbf{Q}(s) \cup \{\mathbf{Q}(s,a)\}$
6 $\mathbf{Q}(s) \leftarrow \text{StDFilter}(\mathbf{Q}(s))$
7 $\mathbf{Q}(s) \leftarrow \text{KDFilter}(\mathbf{Q}(s), \mathscr{K})$
8 $\texttt{Queries} \leftarrow \bigcup_s \left\{\{\mathbf{v}, \mathbf{v}'\} \subseteq \mathbf{Q}(s) : \mathbf{v} \neq \mathbf{v}'\right\}$
9 **while** $\texttt{Queries} \neq \emptyset$ **do**
10 $\{\mathbf{v}, \mathbf{v}'\} \leftarrow \arg\max\{\texttt{X-score}(\{\mathbf{v}, \mathbf{v}'\}) : \{\mathbf{v}, \mathbf{v}'\} \in \texttt{Queries}\}$
11 $(\_, \mathscr{K}) \leftarrow \text{query}(\mathbf{v}, \mathbf{v}', \mathscr{K})$
12 **for** $s \in \mathscr{S}$ **do** $\mathbf{Q}(s) = \text{KDFilter}(\mathbf{Q}(s), \mathscr{K})$
13 $\texttt{Queries} \leftarrow \bigcup_s \left\{\{\mathbf{v}, \mathbf{v}'\} \subseteq \mathbf{Q}(s) : \mathbf{v} \neq \mathbf{v}'\right\}$

where function X-score (with X-score$\in$\{Q,K,S\}) is one of the priority functions described below, the value of which is intended to reflect how informative a query is, and Queries is the set of all unsolved queries. Note that here the best vector returned by query$(\mathbf{v}, \mathbf{v}', \mathscr{K})$ does not need to be saved.

To define priority functions evaluating the informative value of a query, several methods can be considered. Here we propose two types of heuristics. The first type aims to reduce as much as possible the set of remaining unsolved queries (i.e., focusing on the impact on the cardinality of Queries, the set of unsolved queries) while the second type tries to reduce as much as possible the set of admissible reward functions (i.e., focusing on the impact on the polytope $\mathscr{C}(\mathscr{K})$).

Strategies aiming to reduce the cardinality of set Queries. Let $\{\mathbf{v}, \mathbf{v}'\}$ be a query. If we know that, for instance, $\mathbf{v}$ is preferred to $\mathbf{v}'$, this induces an additional constraint that re-

duces the polytope $\mathscr{C}(\mathscr{K})$. This new, smaller, polytope may induce new relations of K-dominance — for instance, we might be able to check that for all $\mathbf{c} \in \mathscr{C}(\mathscr{K} \cup \{\mathbf{v} \succeq \mathbf{v}'\})$ a vector $\mathbf{v^1}$ is preferred to another vector $\mathbf{v^2}$ — in other words the information carried in an answer to a query can generalize to other queries. A natural idea to evaluate the information value of a query is then to count the number of queries that can be resolved if $\mathbf{v} \succeq \mathbf{v}'$, the number of queries resolved if, on the other hand, $\mathbf{v}' \succeq \mathbf{v}$ and to take the minimum between the two values. This relevance score will be called Q-score (Q for *Queries*). Let $\mathrm{Q}_{val}(\mathbf{v},\mathbf{v}')$ be the number of queries decided if $\mathbf{v} \succeq \mathbf{v}'$; then:

$$\texttt{Q}\text{-score}(\{\mathbf{v},\mathbf{v}'\}) = \min\{\mathrm{Q}_{val}(\mathbf{v},\mathbf{v}'), \mathrm{Q}_{val}(\mathbf{v}',\mathbf{v})\}. \tag{5.19}$$

This idea is simple and natural but comes at the cost of solving $4(\mathrm{N}-1)$ linear programs (one for each call to KDom) in the worst case if N is the number of queries.

Algorithm 16: Q-score

Data: $\mathbf{v},\mathbf{v}',\mathscr{K}$, Queries
Result: Q-score($\{\mathbf{v},\mathbf{v}'\}$)

$s_{\mathbf{v}}, s_{\mathbf{v}'} \leftarrow 0,0$
$\mathscr{K}_{\mathbf{v}}, \mathscr{K}_{\mathbf{v}'} \leftarrow \mathscr{K} \cup \{(\mathbf{v}-\mathbf{v}').\mathbf{c} \geq 0\}, \mathscr{K} \cup \{(\mathbf{v}'-\mathbf{v}).\mathbf{c} \geq 0\}$
for $\{\mathbf{v^1},\mathbf{v^2}\} \in$ Queries **do**
  if *KDominates*$(\mathbf{v^2},\mathbf{v^1},\mathscr{K}_{\mathbf{v}})$ *or KDominates*$(\mathbf{v^1},\mathbf{v^2},\mathscr{K}_{\mathbf{v}})$ **then** $s_{\mathbf{v}} \leftarrow s_{\mathbf{v}}+1$
  if *KDominates*$(\mathbf{v^2},\mathbf{v^1},\mathscr{K}_{\mathbf{v}'})$ *or KDominates*$(\mathbf{v^1},\mathbf{v^2},\mathscr{K}_{\mathbf{v}'})$ **then** $s_{\mathbf{v}'} \leftarrow s_{\mathbf{v}'}+1$
return $\min\{s_{\mathbf{v}}, s_{\mathbf{v}'}\}$

Strategies aiming to directly reduce polytope $\mathscr{C}(\mathscr{K})$. Another idea is to use the optimal objective values given by procedure KDom. Consider a query $\{\mathbf{v},\mathbf{v}'\} \in$ *Queries*. Let $\mathscr{K}_{val}(\mathbf{v},\mathbf{v}')$ be the optimal value of the linear program described by Equations 5.17-5.18, normalized by $||\mathbf{v}-\mathbf{v}'||$. Since neither $\mathbf{v}$ nor $\mathbf{v}'$ could be filtered, both $\mathscr{K}_{val}(\mathbf{v},\mathbf{v}')$ and $\mathscr{K}_{val}(\mathbf{v}',\mathbf{v})$ are necessarily negative. We define the priority of query $\{\mathbf{v},\mathbf{v}'\}$ as

$$\texttt{K}\text{-score}(\{\mathbf{v},\mathbf{v}'\}) = \min\{|\mathscr{K}_{val}(\mathbf{v},\mathbf{v}')|, |\mathscr{K}_{val}(\mathbf{v}',\mathbf{v})|\}. \tag{5.20}$$

The idea of K-score is that $\mathscr{K}_{val}(\mathbf{v},\mathbf{v}')$ and $\mathscr{K}_{val}(\mathbf{v}',\mathbf{v})$ give us an approximation of the volume of the polytope on both sides of the constraint defined by the query (see Figure 5.5). Thus, it is likely that a high K-score query will reduce largely the polytope no matter the answer of the tutor. This alternative has the benefits of its algorithmic simplicity. Indeed the computation of $\mathscr{K}_{val}(\mathbf{v},\mathbf{v}')$ and $\mathscr{K}_{val}(\mathbf{v}',\mathbf{v})$ only requires to solve small linear programs.

Alternatively, following the same idea, rewards can be sampled in the admissible reward space (using a Gibbs sampler [Casella and George, 1992]). Let SR be the set of samples generated and consider a query $\{\mathbf{v},\mathbf{v}'\} \in$ *Queries*. Let $\mathrm{S}_{val}(\mathbf{v},\mathbf{v}') = |\{\mathbf{c} \in \texttt{SR} : (\mathbf{v}-\mathbf{v}').\mathbf{c} \geq 0\}|$. The priority of query $\{\mathbf{v},\mathbf{v}'\}$ is then defined as its S-score (S for sampling):

$$\texttt{S}\text{-score}(\{\mathbf{v},\mathbf{v}'\}) = \min\{\mathrm{S}_{val}(\mathbf{v},\mathbf{v}'), \mathrm{S}_{val}(\mathbf{v}',\mathbf{v})\}. \tag{5.21}$$

The numbers of samples on each side of the hyperplane defined by the query $\{\mathbf{v},\mathbf{v}'\}$ gives us an approximation of the volumes of the polytope on each side of the query (see Figure 5.6 where samples labeled by y belong to $\{\mathbf{c} \in \texttt{SR} : (\mathbf{v}-\mathbf{v}').\mathbf{c} \geq 0\}$ and samples labeled by n belong to $\{\mathbf{c} \in \texttt{SR} : (\mathbf{v}'-\mathbf{v}).\mathbf{c} \geq 0\}$). Hence a query which has roughly 50% of samples on

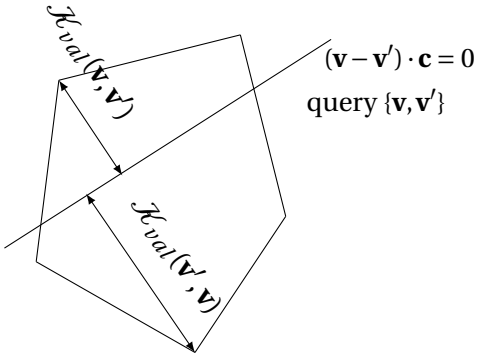


Figure 5.5: Illustration of K-score.

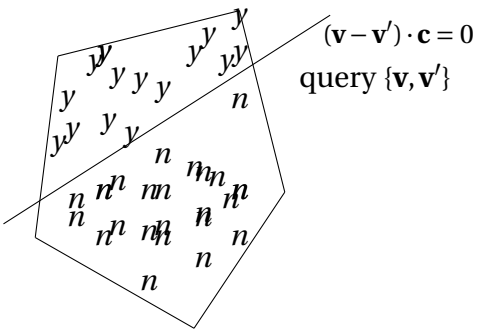


Figure 5.6: Illustration of S-score

both sides will approximately cut the polytope in two equal volumes and it will be deemed a very informative query according to this strategy. A similar idea was used by Rosenthal and Veloso [2012] in a context where rewards are a weighted sum of known subrewards and the elicitation procedure searches for the unknown weights.

Allowing small mistakes in the early stages. The modification of IVI we propose in this subsection aims to make a compromise between the number of iterations of the procedure and the number of queries issued. The idea is to take the "risk" of slowering convergence by avoiding as much as possible to ask queries in the early stages. For this purpose, the condition $z^*_{\mathrm{K}} \geq 0$ in function *KDom* (we recall that z^*_{K} corresponds to the optimal value of program (5.17-5.18)) is loosened: a vector $\mathbf{v}'$ is considered to be dominated if $z^*_{\mathrm{K}} \geq -\mathrm{err}(t)$, where $\mathrm{err}(t) \geq 0$ is a function that decreases to 0 with t. Clearly, this trick has the potentiality to avoid many queries during the early stage with the possible drawback of temporarily driving IVI towards a misleading direction. To ensure that this modification will not prevent the algorithm to converge towards the optimal value function, we modify the main loop of IVI so that the algorithm will keep running until $\mathrm{err}(t)$ reaches 0.

Synthesis. Algorithm 17 synthesizes our modifications to IVI. The initialization of the algorithm (Lines 1 to 4) is unchanged. The main loop (Lines 5 to 22) iterates until the value function converges to the optimal value function and the $\mathrm{err}(t)$ function converges to 0 (i.e., $\mathrm{err}(t) < \delta$ with $\delta << 1$). From Line 7 to Line 13, we fill the sets of possible value vectors for each state by computing and appending the Q-values of the corresponding state-action pairs (Lines 7 to 11) and then filter out the vectors (Lines 12 and 13) that we already know are dominated by using functions *StDFilter* (Algorithm 14) and *KDFilter* (Algorithm 15). This latter algorithm now takes an extra parameter (i.e., $\mathrm{err}(t)$) for implementing the idea presented in the previous subsection. In the second part of the loop, we consider the set `Queries` of all unsolved queries composed of pairs of non-dominated vectors of a same state. While there exists unsolved queries we select and issue the most informative query (Lines 16 and 17) using the priority score, X-score ($\mathscr{X} \in \{\mathtt{Q},\mathtt{K},\mathtt{S}\}$). Once the tutor has answered the query, the acquired information may enable to filter other vectors (Line 18), thus reducing the number of unsolved queries (Line 19). Once all the queries are solved, each set $\mathbf{Q}(s)$ is composed of a single element, corresponding to $\mathbf{v}_t(s)$ (the value vector of s for the next time step). The optimal value function for $\mathscr{M} = (\mathscr{S}, \mathscr{A}, \mathscr{P}, \hat{\mathrm{c}}, \gamma)$ is returned. By using standard bookkeeping techniques, we could return the optimal policy as well.

Algorithm 17: Modified IVI

Data: $\mathscr{S}$, $\mathscr{A}$, $\mathscr{P}$, $\hat{c}$, γ, $\mathscr{R}$, ϵ, err

$t \leftarrow 0$
compute $\mathbf{c}$ from $\hat{c}$
$\mathscr{K} \leftarrow Init(\mathscr{R})$
for $s \in \mathscr{S}$ **do** $\mathbf{v}_0(s) \leftarrow (0,\ldots,0)$
repeat
  $t \leftarrow t+1$
  for $s \in \mathscr{S}$ **do**
    $\mathbf{Q}(s) \leftarrow \emptyset$
    for $a \in \mathscr{A}$ **do**
      $\mathbf{Q}(s,a) \leftarrow \mathbf{c}(s,a) + \gamma \sum_{s' \in \mathscr{S}} \mathscr{P}(s'|s,a)\mathbf{v}_{t-1}(s')$
      $\mathbf{Q}(s) \leftarrow \mathbf{Q}(s) \cup \{\mathbf{Q}(s,a)\}$
    $\mathbf{Q}(s) \leftarrow$ StDFilter($\mathbf{Q}(s)$)
    $\mathbf{Q}(s) \leftarrow$ KDFilter($\mathbf{Q}(s), \mathscr{K}$, err$(t)$)
  Queries $\leftarrow \bigcup_s \left\{\{\mathbf{v},\mathbf{v}'\} \subseteq \mathbf{Q}(s) : \mathbf{v} \neq \mathbf{v}'\right\}$
  while Queries $\neq \emptyset$ **do**
    $\{\mathbf{v},\mathbf{v}'\} \leftarrow \arg\max\{$X-score$(\{\mathbf{v},\mathbf{v}'\}) : \{\mathbf{v},\mathbf{v}'\} \subseteq$ Queries$\}$
    $(\_, \mathscr{K}) \leftarrow$ query$(\mathbf{v},\mathbf{v}', \mathscr{K})$
    for $s \in \mathscr{S}$ **do** $\mathbf{Q}(s) =$ KDFilter$(\mathbf{Q}(s), \mathscr{K}$, err$(t))$
    Queries $\leftarrow \bigcup_s \left\{\{\mathbf{v},\mathbf{v}'\} \subseteq \mathbf{Q}(s) : \mathbf{v} \neq \mathbf{v}'\right\}$
  /*each $\mathbf{Q}(s)$ is now a singleton*/
  for $s \in \mathscr{S}$ **do** $\mathbf{v}_t = \mathbf{Q}(s)$
until $||\mathbf{v}_t - \mathbf{v}_{t-1}|| < \epsilon$ *and* err$(t) < \delta$
return $\mathbf{v}_t$

Discussion. We show below in Example 50 that the reduction of the number of queries entailed by our modifications can be arbitrarily large or null. The example considered is related to the first improvement we have made on IVI about delaying the queries out of the loop over actions but similar examples can be created regarding the other modifications.

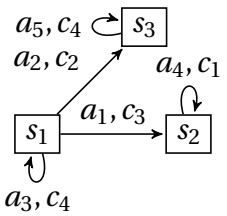


Figure 5.7: The MDP in Example 50.

Example 50. *Consider the MDP represented in Figure 5.7 with* $\gamma = 1/3$ *and true (unknown) reward values* $c_1 = 0$, $c_2 = 0.2$, $c_3 = 0.8$, $c_4 = 1$. *The only state with several possible actions is* s_1. *In state* s_1, *as action* a_3 *dominates all other actions (i.e.,* a_1 *and* a_2*) with respect st-dominance, our modified IVI will not need to issue any queries to solve this MDP. Indeed, method* StDFilter *will always filter out actions* a_1 *and* a_2. *However, if in state* s_1 *the original IVI procedure starts by comparing actions* a_1 *and* a_2, *an important number of queries will*

be issued. Indeed, at iteration t, the quality of actions a_1 and a_2 computed by the value iteration algorithm are:

$$\mathbf{Q}_t(s_1,a_1) = \mathbf{c}_3 + \frac{1}{3}\mathbf{v}_{t-1}(s_2) = \mathbf{c}_3 + \frac{1}{3}\frac{(1-(1/3)^t)}{1-1/3}\mathbf{c}_1 = \mathbf{c}_3 + 0.5(1-(1/3)^t)\mathbf{c}_1$$
$$\mathbf{Q}_t(s_1,a_2) = \mathbf{c}_2 + \frac{1}{3}\mathbf{v}_{t-1}(s_3) = \mathbf{c}_2 + \frac{1}{3}\frac{(1-(1/3)^t)}{1-1/3}\mathbf{c}_4 = \mathbf{c}_2 + 0.5(1-(1/3)^t)\mathbf{c}_4$$

where $\mathbf{c}_i$ is the reward vector whose i^{th} component is 1 all other being 0. There is no dominance relation between $\mathbf{Q}_t(s_1,a_1)$ and $\mathbf{Q}_t(s_1,a_2)$ regarding st-dominance. Hence, the algorithm will need to issue some queries and acquire preference constraints to filter some dominated actions. As $c_3+0.5(1-(1/3)^t)c_1 = 0.8$ and $c_2+0.5(1-(1/3)^t)c_4 = 0.2+0.5(1-(1/3)^t) <$ 0.7, the answer to the query "Is $\mathbf{Q}_t(s_1,a_1) \succeq \mathbf{Q}_t(s_1,a_2)$?" is always yes. Unfortunately, new constraints of this form will not make it possible to know if $\mathbf{Q}_{t'}(s_1,a_1)$ should be preferred or not to $\mathbf{Q}_{t'}(s_1,a_2)$ with $t' > t$ because:

$$\mathrm{Q}_t(s_1,a_2) < \mathrm{Q}_{t+1}(s_1,a_2) < \ldots < \mathrm{Q}_{t+1}(s_1,a_1) = \mathrm{Q}_t(s_1,a_1).$$

Hence, a query will be issued at every iteration of the algorithm.

On the contrary, if we remove action a_3 from state s_1 then IVI and our modified IVI will both yield the same performance (assuming that the errors we enable modified IVI to make at the beginning of the algorithm are too small to save any queries), and will both issue one query per iteration.

This example shows that the gain from our modifications can be arbitrarily large or null. It also shows that the number of queries issued can be very large in the worst case. Indeed at every time step the number of non dominated state-action pairs can be as large as $|\mathscr{S}| \cdot |\mathscr{A}|$. In practice we expect queries to be related so that getting the answer to one query from the tutor might provide the answer to some other queries. Thus, we hope the number of queries issued will not be too large.

5.3.4 Numerical Tests

This approach was tested on three different domains: randomly generated MDPs, autonomic computing [Boutilier *et al.*, 2003] and a simulated setting of personalized assistance to impaired people (Coach domain [Boger *et al.*, 2006])[2]. The original IVI and the improved version described in subsection 5.3.3 were coded in Python using Gurobi 6.0 as an LP solver. The discount factor γ is set to 0.95, ϵ to 10^{-3}, δ to 10^{-7} and $\mathrm{err}(t) = e^{-t}$. The number of samples used by the S-score is 5000. All numeric results are averaged over 20 runs.

Random MDPs. Let us first compare IVI and different variants of the improved IVI on randomly generated MDPs. Given fixed n, m, k (the numbers of states, of actions and of different reward values), we randomly generate the transition function assuming that each pair (s,a) has $\lfloor \log_2(n) \rfloor$ successors (chosen uniformly from the set of states). Transition probabilities are obtained by sampling between 0 and 1 and then normalizing. The type of reward of each pair (s,a) is picked from the uniform categorical distribution $c_1,\ldots,c_k$; the numerical values are randomly generated in interval $[0,1]$ and reordered in order to be consistent.

[2] In both the autonomic computing domain and in Coach, the transition values and the rewards were randomly generated in such a way to satisfy the constraints imposed by the problem domain.

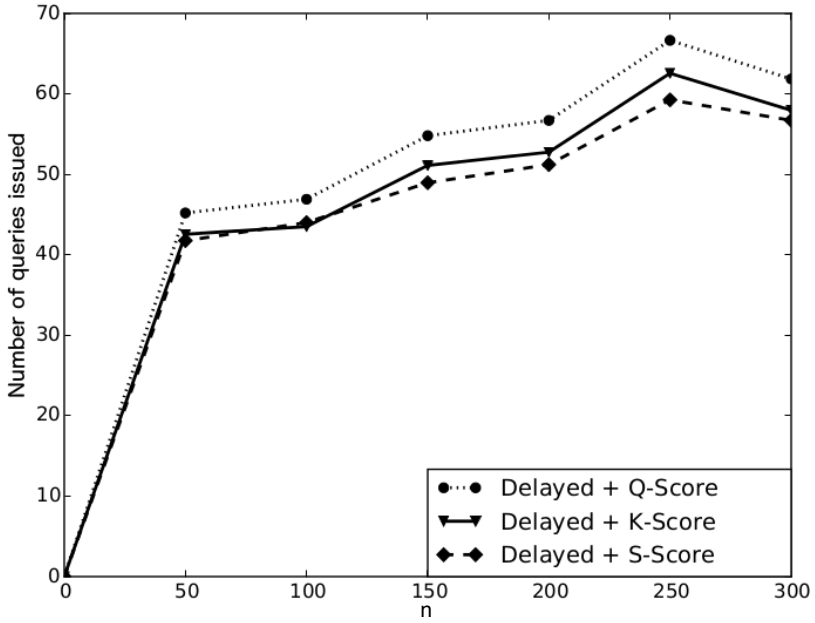


Figure 5.8: Number of queries vs number of states for each priority scores.

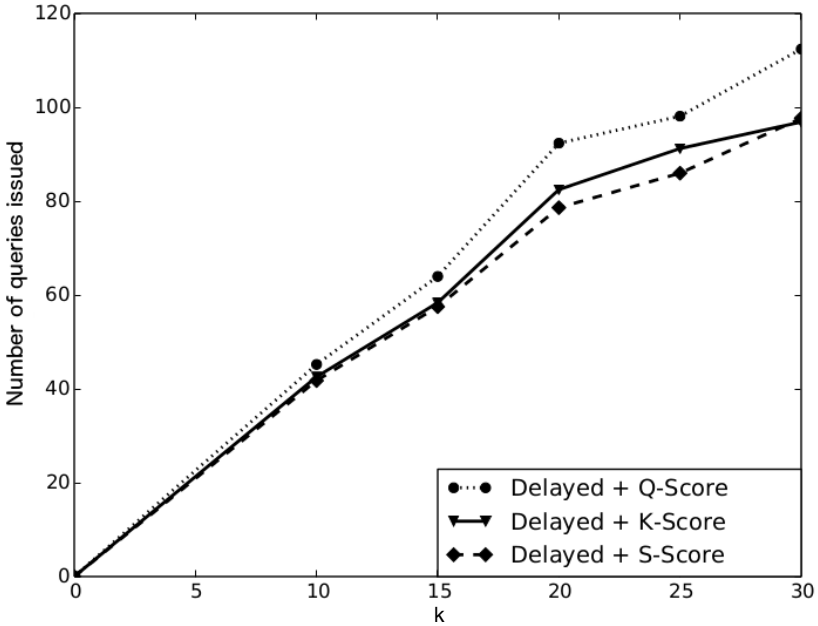


Figure 5.9: Number of queries vs number of rewards for each priority scores.

In Figures 5.8 (the number of queries asked as a function of n; $k = |\mathscr{R}|$ is fixed to 10 and m to 5) and 5.9 (the number of queries asked as a function of the number $|\mathscr{R}|$ of different ordinal rewards; n is fixed to 50 and m to 5) we compare the different priority scores that can be used to choose the next query to ask; the graph shows that all three techniques (`Q`-score, `K`-score and `S`-score) are similarly effective, with `S`-score performing best.

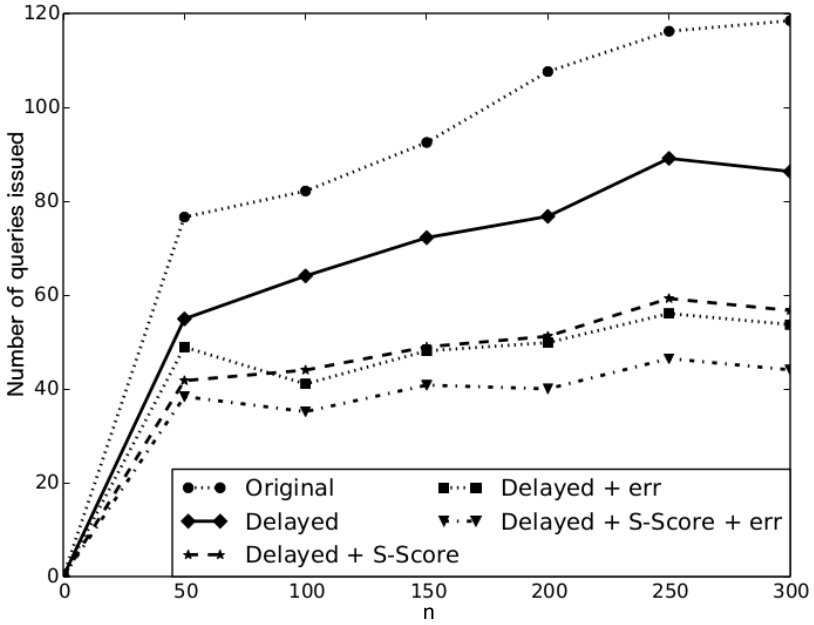


Figure 5.10: Number of queries vs number of states for different query strategies.

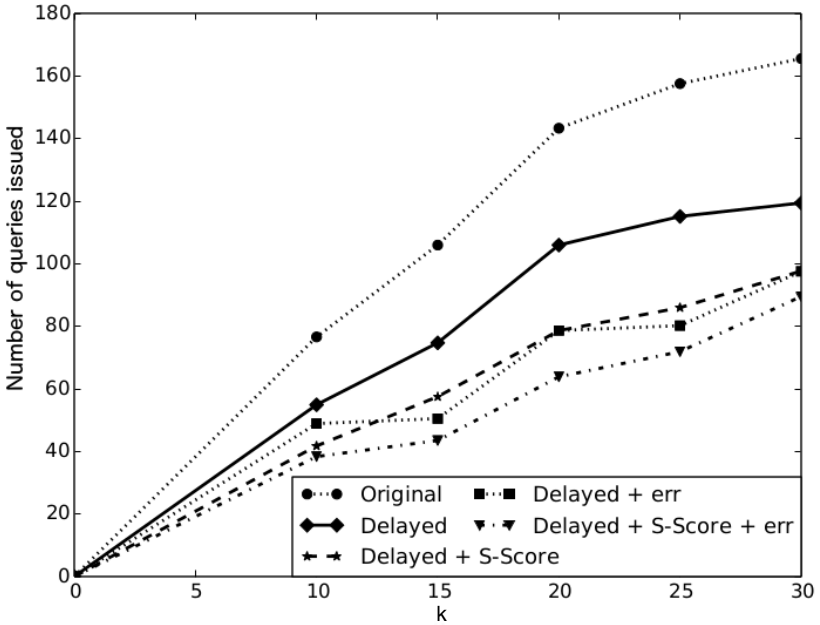


Figure 5.11: Number of queries vs number of rewards for different query strategies.

In Figure 5.10 we compare the impact of the different improvements that we described with the performance of the original IVI ($k = |\mathscr{R}|$ is fixed to 10 and m to 5). For the queries' priority we focus on the S-score since this seemed to be the best performing strategy. As expected, IVI asks the highest number of queries (around 80 with 100 states; 120 with 300 states); delaying the moment of asking queries already gives a very significant advantage

(around 60 and 85 queries for 100, 300 states). If we additionally ask queries according to the priority induced by the S-score, the number of queries reduces even more (around 45 and 55 queries for 100, 300 states); but surprisingly this improvement is less than the improvement obtained by the combination of delaying the queries and the heuristic of allowing small errors in the initial iterations. The best results are obtained by combining the S-score with the "error" heuristic (about 40 queries with 300 states). Interestingly, with our improvements, the number of queries asked by modified IVI grows very slowly with respect to the number of states.

Figure 5.11 compares the same strategies but with different numbers of rewards (i.e., $k = |\mathscr{R}|$), the number of states n being fixed to 50 and m to 5. By comparing Figure 5.10 with Figure 5.11 we can see that parameter k (number of reward values) impacts much more the number of queries than parameter n (number of states); especially when considering the version of IVI including all our improvements (denoted by "Delayed + S-score + err" in the plots).

Autonomic Computing. We also applied our algorithm on the domain of autonomic computing [Boutilier *et al.*, 2003]. In this domain, we assume there are κ application server elements on which N available resources have to be assigned. A feasible allocation is an integer vector $\mathbf{a} = (a_1, \ldots, a_\kappa)$ with $\sum_{i=1}^{\kappa} a_i \leq \mathrm{N}$. The client's demand (changing over time) is an integer vector $\mathbf{d} = (d_1, \ldots, d_\kappa)$ representing κ levels of demand in $\{1, \ldots, \mathrm{D}\}$. A state of the MDP is a vector $(\mathbf{a}, \mathbf{d})$ defining current allocation and demand. An action is the adoption of a new allocation $\mathbf{m} = (m_1, \ldots, m_\kappa)$. The reward of taking action $\mathbf{m}$ in state $(\mathbf{a}, \mathbf{d})$ is $c((\mathbf{a}, \mathbf{d}), \mathbf{m}) = u(\mathbf{a}, \mathbf{d}) - \zeta(\mathbf{a}, \mathbf{d}, \mathbf{m})$ where $u(\mathbf{a}, \mathbf{d}) = \sum_{i=1}^{\kappa} u(n_i, d_i)$ is the sum of non-decreasing utility-functions $u(a_i, d_i)$ and $\zeta(\mathbf{a}, \mathbf{d}, \mathbf{m})$ is the sum of the costs for removing a resource unit from the server. An action deterministically sets the next allocation, while the uncertainty about demands is stochastic and exogenous.

We ran IVI and modified IVIs on instances with $\kappa = 2$, $\mathrm{N} = 3$ and $\mathrm{D} = 3$ (90 states and 10 actions). While the original IVI needs 188.7 queries to converge, by delaying the queries we reduce this number to 108.9. Prioritizing the order in which the queries are asked further reduces the number of queries to 86.9. Finally by using the heuristic that allows small mistakes in the first iterations of the algorithm, we only need to ask 66.3 queries.

Coach. Finally, we present our experimental results on the "Coach" domain [Boger *et al.*, 2006]. In this problem, we provide assistance to a person with dementia accomplishing a daily-life activity (e.g., handwashing) that is decomposed into $\mathrm{T} = \{0, \ldots, l\}$ phases. Different types of aids are available, modeled by actions $\mathscr{A} = \{0, \ldots, m\}$; $a \in \mathscr{A}$ is a form of assistance, each associated with a different level of intrusiveness between 0 and m; 0 represents no prompt (no aid is given), $m-1$ represents the most intrusive prompt and m means that a caregiver has to be called. The goal is to aid the person in completing the task, with enough help but avoiding being too intrusive.

A state in the MDP is described as a tuple (t, d, f) where $t \in \mathrm{T}$ is the current time step, $d \in \mathrm{D} = \{0, \ldots, 5\}$ is the delay (time already spent in the current phase of the task) and $f \in \mathscr{A}$ is the last prompt used. The transitions model the chance of "success", i.e., the probability that the person moves to the next phase. To model the effectiveness of the aid, at each phase $t < l$ of the task, the probability of success is increasing with the level of intrusiveness of action a; however the probability is decreasing with d. Furthermore, reaching the next step after prompting is less likely if a prompt has already failed at the current step. The reward associated to taking action a in state (t, d, f) is defined by:

$$c((t, d, f), a) = c_{goal}(t) + c_{progress}(d) + c_{delay}(d) + c_{prompt}(a)$$

where $c_{goal}(t)$ gives a large reward when the final phase is reached and 0 otherwise, $c_{progress}(d)$ is a small reward when passing to the next phase with no delay and 0 otherwise, $c_{delay}(d)$ and $c_{prompt}(a)$ are increasing cost functions.

We ran IVI and our improved versions of IVI on instances with $l = 14$ and $m = 6$ (630 states and 7 actions). While the original version of IVI needed 169.9 queries to converge, by delaying the queries we reduced this number to 114.7 and prioritizing the order in which the queries were issued curbed this score to 77.9. Lastly by allowing small mistakes at the beginning of the algorithm the number of queries issued decreased to 71.2.

5.3.5 Summary of Section: an Interactive Value Iteration Algorithm

IVI is an appealing procedure that mitigates the burden of defining the reward function of an MDP by interweaving the elicitation and resolution phases. In order to find an optimal policy for an MDP, this procedure queries a tutor about comparisons of multisets of rewards when needed. In this section, we have presented modifications to the original algorithm that are shown to reduce substantially the number of queries issued. The main ideas are that we can avoid unnecessary queries by delaying the querying phase, and reasoning about the order in which we ask the query.

The next section also deals with an elicitation process. While the decision problem considered is much simpler (we suppose that we can explicitly list all the solutions of the problem), the decision model considered is much more complex.

5.4 Decision Making under Risk with an Imprecise WEU Model: an Efficient Incremental Elicitation Procedure

In this section, a decision maker is faced with numerous risky prospects given as lotteries on a set of possible outcomes denoted by $\mathscr{X}$ and must find an alternative that suits her best. In this context, the decision analyst has to fit the parameters of the decision model of the decision maker through interactions with her. This is known as the preference elicitation step, which is paramount to advise the decision maker or to predict her choices. This task is complex, especially due to the fact that uncertainty affects the decision maker's judgements.

We assume that the decision maker is consistent with the WEU model (see subsection 1.5.4), a subclass of the SSB model. The WEU model relies on two functions u and w. Therefore, the preference elicitation step consists in eliciting these functions to unveil the preferences of the decision maker. To the best of our knowledge, no previous work has tackled the elicitation of the WEU model. In this section, we design an elicitation protocol to elicit the WEU model and then show how this protocol can be used in an incremental elicitation procedure. Our numerical tests show the efficiency of the approach.

5.4.1 Related Work

Once we have identified the decision model that describes the preferences of the decision maker, the decision analyst knows on which parameters its recommendation should be made. However, the decision model does not stipulate how the values of these parameters should be chosen to fit the preferences of the decision maker. Unfortunately, a slight change in these values may lead to completely different solutions than the preferred ones

of the decision maker. To set correctly these values, one can rely on an elicitation procedure which specifies the parameters of the model through a sequence of interactions with the decision maker. Preference elicitation is an active research topic in artificial intelligence and has led to different approaches. We classified them as follows but note that some approaches might overlap with others.

The standard approach aims to elicit entirely the model by specifying completely each parameter. For instance, with the Expected Utility (EU) model, the elicitation procedure would completely specify the utility function of the decision maker. Elicitation protocols exist for several decision models as EU (e.g., [Wakker and Deneffe, 1996]), Rank Dependent Utility (RDU) (e.g., [Bleichrodt and Pinto, 2000]) or regret theory (e.g., [Bleichrodt *et al.*, 2010]). The drawback of this approach is that it can require a large number of interactions before being able to fit exactly the preferences of the decision maker. On the other hand, the advantage of this approach is that, at the end of the process, the model has been completely elicited and can be reused repeatedly and in different problems.

Instead of trying to learn completely the model, another elicitation approach aims to elicit the model "just" to solve one specific optimization problem. In that case, one can try to interleave the elicitation procedure with the resolution of the optimization problem. This is called *interactive optimization* [Korhonen and Laakso, 1986; Zionts and Wallenius, 1976]. The idea is to collect feedbacks from the decision maker to orientate the search of an optimal solution within the set of possible optimal solutions (given the uncertainty in the model). This is realized via an interactive loop in which two steps are realized. In the first step, the system evaluates the solutions using a possible set of parameters for the decision model. In a second step, the system presents a subset of solutions to the decision maker on which she can give feedbacks. The process stops once the decision maker has found a "satisfying" solution. The advantage of this approach is that it does not need to elicit entirely the model before finding a satisfying solution. Hence, it requires less interactions with the decision maker than the standard approach. However, the information acquired may not be sufficient enough to solve other problems than the one considered in the interactive process.

Within the setting of interactive optimization, several approaches can be distinguished. For instance, in *Bayesian elicitation* methods, instead of collecting hard constraints which separate admissible from non-admissible sets of parameters, the system uses the information acquired to define a probability distribution over the possible sets of parameters [Braziunas, 2012; Chajewska *et al.*, 2000; Viappiani and Boutilier, 2010]. One advantage of this method is that it makes it possible to model possible errors made by the decision maker. In this framework, each solution is evaluated by an expected value over the possible sets of parameters and a good query is likely to increase the maximal expected value of a solution. The process is stopped once the system has identified a "good" solution with high enough probability.

A last approach is the *incremental elicitation* framework in which the preference queries are selected one by one in order to efficiently reduce the imprecision over the parameters of the model until a near-optimal choice is identified. This approach uses sophisticated decision criteria to identify the next query that will be issued, such as the minmax regret criterion [Benabbou and Perny, 2015; Benabbou *et al.*, 2014; Regan and Boutilier, 2009]. This approach has proved very efficient in various domains, solving complex problems while only requiring few preference queries (see e.g., [Chajewska *et al.*, 2000] and [Wang

and Boutilier, 2003] for EU, [Perny *et al.*, 2016] for Rank Dependent Utility or [Nguyen *et al.*, 2014] for the incremental elicitation of attacker payoffs in security games).

The work that we present in this section is both related to the standard approach and the incremental elicitation approach. Indeed, we give an elicitation procedure for the WEU model that could be used to elicit the model entirely, and then show how to integrate it in an incremental setting.

5.4.2 Background

The WEU model [Chew, 1983] is a risk-sensitive model that extends the EU model with enhanced descriptive abilities while conserving desirable normative properties. Indeed, while it has been shown in Chapter 1 that the WEU model could account for non-independent preferences (as the ones exposed by Allais' paradox), we have seen in Chapter 4 that this model could still be optimized in polynomial time in decision trees. The WEU model relies on two real functions u and w defined on the set of possible outcomes, denoted here by $\mathscr{X}$. These functions are then lifted from outcomes to lotteries by using expectations: $u(l) = \sum_{x\in\mathscr{X}} l(x)u(x)$ and $w(l) = \sum_{x\in\mathscr{X}} l(x)w(x)$. In the WEU model, the preference $\succsim$ is defined by[3]:

$$l_1 \succsim l_2 \Leftrightarrow u(l_1)w(l_2) - u(l_2)w(l_1) \geq 0 \tag{5.22}$$

and $u(l_1)w(l_2) - u(l_2)w(l_1)$ is a signed intensity of preference between lotteries l_1 and l_2.

One key issue in preference modeling with WEU is the elicitation of the two functions u and w to fit the value system of a given decision maker.

Two different elicitation approaches can be distinguished. On the one hand, one can perform a full elicitation using a systematic sequence of queries aiming to completely specify the decision model. However, this precise elicitation of functions u and w would certainly require a prohibitive number of queries. Moreover, the full elicitation of the model is generally not necessary to identify the optimal choice within a given set of lotteries.

For this reason, after we have identified how to elicit the WEU model, we will focus on the incremental approach in which preference queries are selected one by one in order to efficiently reduce the imprecision over the parameters of the model until a near-optimal choice is identified.

5.4.3 Preliminary Results

Let $\mathscr{P}$ be the set of all lotteries defined on the outcome space $\mathscr{X}$. In this work, we assume that $\mathscr{X}$ is a bounded interval of $\mathbb{R}$, so that we can set $\mathscr{X}$ to $[0,1]$ without loss of generality. In our setting, the decision maker has to select a lottery among a finite set $\mathscr{L} \subset \mathscr{P}$.

We now make two natural assumptions on the decision maker's preferences. We notably assume that the decision maker is not always indifferent, imposing that the highest outcome 1 is strictly preferred to the lowest one 0. Furthermore, given two outcomes x and y with $x \geq y$, we assume that the signed intensity of preference between x and any other outcome z is at least as high as the one between y and z. In the WEU model, these hypotheses can be written as follows (see Equation 5.22):

$$h_1\colon u(1)w(0) - u(0)w(1) > 0$$
$$h_2\colon x \geq y \Rightarrow \forall z \in \mathscr{X}, (u(x) - u(y))w(z) \geq u(z)(w(x) - w(y))$$

[3]Note that a strict inequality corresponds to a strict preference

We now derive some properties that hold for u and w in this context. The first one is a consequence of the following uniqueness theorem:

Theorem 21 (Fishburn [1983])**.** *Suppose $\succ$ is a nonempty asymmetric weak order on $\mathscr{P}$ and (u,w) is a pair of linear functionals on $\mathscr{P}$ such that: $\forall l_1, l_2 \in \mathscr{P}$, $l_1 \succ l_2 \Leftrightarrow u(l_1)w(l_2) - u(l_2)w(l_1) > 0$. Then, for any pair (u',w') of linear functionals on $\mathscr{P}$, the following properties are equivalent:*

1. $\forall l_1, l_2 \in \mathscr{P}$, $l_1 \succ l_2 \Leftrightarrow u'(l_1)w'(l_2) - u'(l_2)w'(l_1) > 0$
2. *There exist $a,b,c,d \in \mathbb{R}$ such that:*

$$\begin{aligned} u' &= au + bw, \\ w' &= cu + dw \\ \text{and} \quad ad - bc &> 0. \end{aligned}$$

We can now state the following result:

Proposition 13. *Suppose $\succ$ is a nonempty asymmetric weak order on $\mathscr{P}$ and (u,w) is a pair of linear functionals on $\mathscr{P}$ such that: $\forall l_1, l_2 \in \mathscr{P}$, $l_1 \succ l_2 \Leftrightarrow u(l_1)w(l_2) - u(l_2)w(l_1) > 0$. Then, if h_1 holds, we can build from u and w another pair of functionals (u',w') such that:*

- $\forall l_1, l_2 \in \mathscr{P}$, $l_1 \succ l_2 \Leftrightarrow u'(l_1)w'(l_2) - u'(l_2)w'(l_1) > 0$
- $u'(1) = w'(0) = 1$ *and* $u'(0) = w'(1) = 0$

Proof. Let $\Delta := u(1)w(0) - u(0)w(1)$, which is strictly positive due to h_1. Let (u',w') be the pair of linear functionals defined by $u' = au + bw$ and $w' = cu + dw$, where $a = w(0)/\Delta$, $b = -u(0)/\Delta$, $c = -w(1)/\Delta$ and $d = u(1)/\Delta$. It is easy to check that $u'(0) = w'(1) = 0$ and $u'(1) = w'(0) = 1$, as required. To conclude the proof, we need to prove that we have $l_1 \succ l_2$ iff $u'(l_1)w'(l_2) > u'(l_2)w'(l_1)$ for all $l_1, l_2 \in \mathscr{P}$. Since $ad - bc = 1/\Delta > 0$, we know that this property is verified using Theorem 21. □

This proposition enables us to conclude that the decision maker's preferences can be modeled using a pair (u,w) such that $u(0) = w(1) = 0$ and $u(1) = w(0) = 1$; note that these values somehow reflects the antisymmetric roles that functions u and w play in the WEU model. Interestingly, setting those values in this way combined with hypothesis h_2 imposes monotonicity conditions on u and w:

Proposition 14. *Suppose $\succ$ is a nonempty asymmetric weak order on $\mathscr{P}$. Moreover, suppose (u,w) is a pair of linear functionals on $\mathscr{P}$ such that:*

- $\forall l_1, l_2 \in \mathscr{P}$, $l_1 \succ l_2 \Leftrightarrow u(l_1)w(l_2) - u(l_2)w(l_1) > 0$
- $u(1) = w(0) = 1$ *and* $u(0) = w(1) = 0$

Then h_2 holds iff u is nondecreasing and w is nonincreasing.

Proof. ($\Rightarrow$) Let $x, y \in \mathscr{X}$ be such that $x \succeq y$. By definition of h_2, we have $(u(x) - u(y))w(z) \geq (w(x) - w(y))u(z)$ for all z in $\mathscr{X}$. In particular, by taking $z = 0$ and $z = 1$, we obtain $u(x) - u(y) \geq 0$ and $w(x) - w(y) \leq 0$ respectively.
($\Leftarrow$) Assume that u is nondecreasing and w is nonincreasing. In that case, for all $x, y \in \mathscr{X}$ such that $x \succeq y$, we have $u(x) - u(y) \geq 0$ and $w(x) - w(y) \leq 0$. Moreover, since $u(0) = 0$ and $w(1) = 1$, we know that $u(z)$ and $w(z)$ are positive for all $z \in \mathscr{X}$. Therefore, we necessarily have $(u(x) - u(y))w(z) \geq u(z)(w(x) - w(y))$. □

This proposition also ensures that the decision maker is rational in the sense that any outcome x is necessarily preferred to any outcome y such that $x \geq y$ when assuming h_1 and h_2.

One can note that the monotonicity conditions of Proposition 14 applies to the couple (u, w) introduced in Example 24 (on page 39), which shows that imposing h_1 and h_2 does not prevent to account for Allais' paradox.

5.4.4 The Elicitation Method

Within WEU theory the preferences of the decision maker are completely characterized by the pair (u, w), which is initially unknown. To assist the decision maker in her choice, we propose an incremental elicitation procedure aiming to reduce the indetermination of functions u and w in order to discriminate the elements in $\mathscr{L}$. At any stage of the process, we manage two sets of functions $\mathscr{U}$ and $\mathscr{W}$ that represent all possible functions u and w given the preference information collected so far. We will iteratively generate carefully chosen preference queries in such a way that collected preference statements enable to reduce either the set $\mathscr{U}$ or the set $\mathscr{W}$.

To initiate the elicitation process, we consider two reference outcomes x_+ and x_- such that $x_+ \succ 1 \succ 0 \succ x_-$, i.e., x_+ and x_- must be chosen outside the range of $\mathscr{X}$. The first step of the elicitation process is to elicit the values $u(x_-)$ and $w(x_+)$. The elicitation can be performed using the following lotteries: $l_\alpha = (1, \alpha; x_-, 1-\alpha)$ and $l_\beta = (x_+, \beta; 0, 1-\beta)$. We ask the probabilities α_0 and β_0 for which the following indifference relations hold: $l_{\alpha_0} \sim 0$ and $l_{\beta_0} \sim 1$. Note that *probability equivalence queries* are standard tools used in decision making under uncertainty. The answers to those queries yield the following equalities:

$$u(l_{\alpha_0})\,w(0) - u(0)\,w(l_{\alpha_0}) = 0 \Leftrightarrow u(x_-) = -\alpha_0/(1-\alpha_0)$$
$$u(l_{\beta_0})\,w(1) - u(1)\,w(l_{\beta_0}) = 0 \Leftrightarrow w(x_+) = -(1-\beta_0)/\beta_0$$

Note that both $u(x_-)$ and $w(x_+)$ are strictly negative, which is consistent with the nondecreasingness (resp. nonincreasingness) of u (resp. w). Once values $u(x_-)$ and $w(x_+)$ are known, we consider two types of queries, denoted by $\texttt{uQuery}(\alpha, l)$ and $\texttt{wQuery}(\alpha, l)$, both parametrized by a lottery l and by a probability value α.

$\texttt{uQuery}(\alpha, l)$ asks the decision maker to compare the compound lottery $l_\alpha = (l, \alpha; x_-, 1-\alpha)$ to a sure gain of 0. If $l_\alpha \succsim 0$, then we deduce a new constraint as follows:

$$\begin{aligned} l_\alpha \succsim 0 &\Leftrightarrow u(l_\alpha)\,w(0) - u(0)\,w(l_\alpha) \geq 0 \\ &\Leftrightarrow u(l_\alpha) \geq 0 \text{ since } u(0) = 0 \text{ and } w(0) = 1 \\ &\Leftrightarrow \alpha u(l) - \alpha_0(1-\alpha)/(1-\alpha_0) \geq 0 \\ &\Leftrightarrow \alpha \sum_{x \in \mathscr{X}} l(x)\,u(x) - \alpha_0(1-\alpha)/(1-\alpha_0) \geq 0 \end{aligned} \tag{5.23}$$

If the decision maker prefers the sure gain, then the inequality is reversed and in both cases we obtain a constraint on the value $u(l)$.

$\texttt{wQuery}(\alpha, l)$ asks the decision maker to compare lottery $l_\alpha = (x_+, \alpha; l, 1-\alpha)$ to a sure gain of 1. If $l_\alpha \succsim 1$, then we have to impose:

$$\begin{aligned} l_\alpha \succsim 1 &\Leftrightarrow u(l_\alpha)\,w(1) - u(1)\,w(l_\alpha) \geq 0 \\ &\Leftrightarrow w(l_\alpha) \leq 0 \text{ since } u(1) = 1 \text{ and } w(1) = 0 \\ &\Leftrightarrow \alpha(\beta_0 - 1)/\beta_0 + (1-\alpha)\,w(l) \leq 0 \\ &\Leftrightarrow \alpha(\beta_0 - 1)/\beta_0 + (1-\alpha) \sum_{x \in \mathscr{X}} l(x)\,w(x) \leq 0 \end{aligned} \tag{5.24}$$

Similarly, we have to reverse the inequality if we observe $1 \gtrsim l_\alpha$ instead. Thus, we are able to derive some constraints on the values of functions u and w that are completely decoupled. This point will reveal paramount for the efficiency of the elicitation process. Those constraints are then used to reduce the uncertainty attached to the sets $\mathscr{U}$ and $\mathscr{W}$.

To identify a near-optimal lottery with few queries, our approach has to focus on the relevant parts of functions u and w. This is achieved by using a criterion that indicates on which lotteries we should ask a query in order to generate a highly informative constraint (i.e., that enables to identify as quickly as possible a near-optimal lottery in $\mathscr{L}$). To choose the next lottery on which a query is asked, we use the *maxmin* criterion.

Maxmin Optimization. Under preference uncertainty, one may be interested in the lottery l_* that performs best in the worst-case scenario. Since the value of any lottery $l \in \mathscr{L}$ is equal to $u(l)/w(l)$ with the WEU model, the maxmin lottery l_* is formally defined by:

$$l_* \in \arg\max_{l \in \mathscr{L}} \min_{(u,w) \in \mathscr{U} \times \mathscr{W}} \frac{u(l)}{w(l)}$$

The key observation to determine l_* is that the constraints obtained when asking `uQueries` are completely decoupled from the ones obtained when asking `wQueries`. Therefore, minimizing $u(l)/w(l)$ for a fixed lottery l amounts to minimizing $u(l)$ over $\mathscr{U}$ and maximizing $w(l)$ over $\mathscr{W}$ (note that all u and w values are nonnegative).

In order to obtain a guarantee on the quality of l_* with respect to the other options in $\mathscr{L}$, we now wish to determine an upper bound $\mathrm{UB}(l', l_*)$ on how much another lottery $l' \in \mathscr{L}$ could be preferred to l_* (i.e., how large $u(l')w(l_*) - u(l_*)w(l')$ can be). For any $l \in \mathscr{L}$, we denote by $\underline{u}(l)$ and $\overline{u}(l)$ (resp. $\underline{w}(l)$ and $\overline{w}(l)$) the minimal and maximal values of $u(l)$ for $u \in \mathscr{U}$ (resp. $w(l)$ for $w \in \mathscr{W}$). Given two lotteries $l_1, l_2 \in \mathscr{L}$, it can easily be checked that $\overline{u}(l_1)\overline{w}(l_2) - \underline{u}(l_2)\underline{w}(l_1) \geq u(l_1)w(l_2) - u(l_2)w(l_1)$ for all $(u,w) \in \mathscr{U} \times \mathscr{W}$, and therefore one can set $\mathrm{UB}(l', l_*) = \overline{u}(l')\overline{w}(l_*) - \underline{u}(l_*)\underline{w}(l')$. Thus, an upper bound on how bad the choice of l_* could be is:

$$\mathrm{UB}(l_*) = \max_{l' \in \mathscr{L} \setminus \{l_*\}} \mathrm{UB}(l', l_*)$$

For some given sets $\mathscr{U}$ and $\mathscr{W}$, it might be the case that $\mathrm{UB}(l_*)$ is too large, meaning that recommending l_* is a bad decision in the worst-case scenario. In this situation, we collect new preference queries to reduce $\mathscr{U}$ and $\mathscr{W}$ until $\mathrm{UB}(l_*)$ becomes smaller than a given acceptable threshold; in particular, if $\mathrm{UB}(l_*) = 0$, then l_* is necessarily the best option. Note that the obtained guarantee is looser than the one obtained by minmax regret as:

$$\overline{u}(l_1)\overline{w}(l_2) - \underline{u}(l_2)\underline{w}(l_1) \geq \max_{(u,w) \in \mathscr{U} \times \mathscr{W}} (u(l_1)w(l_2) - u(l_2)w(l_1)), \forall\, l_1, l_2 \in \mathscr{L}$$

However, the minmax regret induces quadratic optimization problems (instead of linear ones) which are relatively difficult to optimize in general.

Query Generation Strategy. At each step of the elicitation process, we compute both a maxmin lottery l_* and a challenger maxmax lottery l^* defined by:

$$l^* \in \arg\max_{l' \in \mathscr{L} \setminus \{l_*\}} \mathrm{UB}(l', l_*).$$

By definition, lottery l^* may induce a large loss when recommending l_*. To decrease $\mathrm{UB}(l_*) = \mathrm{UB}(l^*, l_*) = \overline{u}(l^*)\overline{w}(l_*) - \underline{u}(l_*)\underline{w}(l^*)$ as much as possible, we first compute:

$$\max\{\overline{u}(l_*) - \underline{u}(l_*), \overline{w}(l_*) - \underline{w}(l_*), \overline{u}(l^*) - \underline{u}(l^*), \overline{w}(l^*) - \underline{w}(l^*)\}.$$

Then, depending on which is maximum, we ask either, respectively, $\texttt{uQuery}(\alpha, l_*)$, $\texttt{wQuery}(\alpha, l_*)$, $\texttt{uQuery}(\alpha, l^*)$, or $\texttt{wQuery}(\alpha, l^*)$ with an appropriate value α. We now discuss the choice of α so as to obtain an informative query.

Consider a query of type $\texttt{wQuery}(\alpha, l)$ for a given l. We want to choose α so that the imprecision on $w(l)$ is reduced as much as possible in the worst-case scenario of answers. Therefore, we choose α in order to obtain a query allowing $w(l)$ to be compared to $(\overline{w}(l) + \underline{w}(l))/2$. In this case, the exact expression of α is:

$$\alpha := \frac{\overline{w}(l) + \underline{w}(l)}{\overline{w}(l) + \underline{w}(l) - 2w(x_+)}.$$

This strategy enables to reduce the imprecision on $w(l)$ by 50% regardless of the decision maker's answer. The procedure is similar for choosing α in queries of type $\texttt{uQuery}$.

Compact Representation of u and w. A first approach for representing functions u and w leads to the introduction of two variables representing $u(z)$ and $w(z)$ for any relevant consequence z (i.e., the consequences that appear at least once in the set $\mathscr{L}$ of lotteries to be compared). However, this approach becomes more and more inefficient as the number and/or the size of lotteries increase. It directly impacts the number of variables defining the set of admissible functions u and w but also the number of constraints that must be considered to enforce the desired monotonicity of those two functions. For this reason, we adopt a more compact representation consisting in approximating functions u and w by monotone spline functions [Perny *et al.*, 2016; Ramsay, 1988].

Spline functions are piecewise polynomials whose pieces connect with a high degree of smoothness. They have a large capacity to approximate complex shapes and guarantee that smooth curves will be derived from smooth data (for more details, see e.g., [Beatty and Barsky, 1987]). On a given interval $[a, b]$, a spline function f is defined as a linear combination of m basis spline functions f_i, $i \in \{1, \ldots, m\}$, i.e., $f(x) = \sum_{i=1}^{m} \lambda_i f_i(x)$. The definition of the basis spline functions is based on a subdivision of $[a, b]$ into k parts $[\xi_j, \xi_{j+1}]$, $j \in \{1, \ldots, k-1\}$, with $\xi_1 = a$ and $\xi_k = b$. The basis spline functions are piecewise polynomials on $[a, b]$ as they are polynomials on each interval $[\xi_j, \xi_{j+1})$, $j \in \{1, \ldots, k-1\}$; therefore, f is also piecewise polynomial on $[a, b]$ by construction. Moreover, adjacent polynomials have matching derivatives (insuring the smoothness). In this work, we use cubic splines (defined on $[a, b] = [0, 1]$) as they are known to offer a good compromise between smoothness and flexibility (see e.g., [Beatty and Barsky, 1987]).

When one wants to enforce the monotonicity of function f, a standard approach is to use a basis of monotone spline functions, namely I-splines [Ramsay, 1988] denoted by I_i, $i \in \{1, \ldots, m\}$. For illustrative purpose, Figure 5.12 shows a basis composed of six cubic I-spline functions, together with a spline function generated from this basis.

As function u must be non-decreasing with $u(0) = 0$ and $u(1) = 1$, we define u as a convex combination of I-splines:

$$u(x) = \sum_{i=1}^{m} \lambda_i^u \mathrm{I}_i(x) \tag{5.25}$$

where $\lambda_i^u \geq 0$ for all $i \in \{1, \ldots, m\}$ and $\sum_{i=1}^{m} \lambda_i^u = 1$. Similarly, since w is a non-increasing function with $w(0) = 1$ and $w(1) = 0$, we define w by:

$$w(x) = 1 - \sum_{i=1}^{m} \lambda_i^w \mathrm{I}_i(x) \tag{5.26}$$

where $\lambda_i^w \geq 0$ for all $i \in \{1, \ldots, m\}$ and $\sum_{i=1}^{m} \lambda_i^w = 1$. Equations 5.25 and 5.26 provide a compact definition of functions u and w using only $2m$ parameters (λ_i^u and λ_i^w), a number

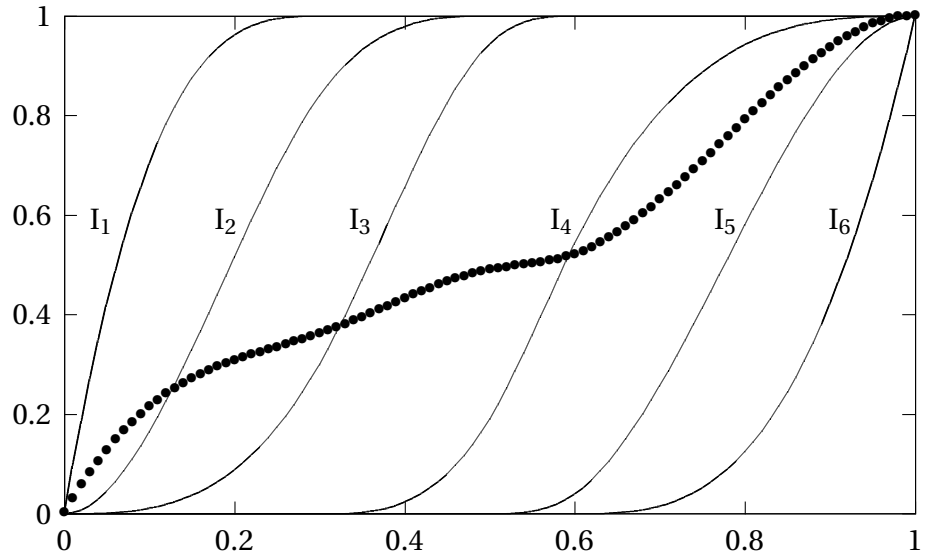


Figure 5.12: The six cubic I-splines associated with the subdivision of $[0,1] = [0,.3] \cup [.3,.5] \cup [.5,.6] \cup [.6,1]$, together with function $f(x) = 0.3\mathrm{I}_1 + 0.2\mathrm{I}_3 + 0.5\mathrm{I}_5$ (dotted line).

which remains constant as the number of lotteries increase. Without any knowledge of the decision maker's preferences, function u can be any function in $\mathscr{U} = \{\sum_{i=1}^{m} \lambda_i^u \mathrm{I}_i,\ (\lambda_1^u, \ldots, \lambda_m^u) \in \mathrm{U}\}$, where:

$$\mathrm{U} = \{(\lambda_1^u, \ldots, \lambda_m^u) \in [0,1]^m,\ \sum_{i=1}^{m} \lambda_i^u = 1\}$$

Similarly, w can initially be any element of the set $\mathscr{W} = \{1 - \sum_{i=1}^{m} \lambda_i^w \mathrm{I}_i,\ (\lambda_1^w, \ldots, \lambda_m^w) \in \mathrm{W}\}$, where:

$$\mathrm{W} = \{(\lambda_1^w, \ldots, \lambda_m^w) \in [0,1]^m,\ \sum_{i=1}^{m} \lambda_i^w = 1\}$$

We now explain how this spline representation is used in our elicitation procedure. To this end, we show how values $\underline{u}(l)$, $\overline{u}(l)$, $\underline{w}(l)$ and $\overline{w}(l)$ are computed for any lottery l. By using our spline representation in Equation 5.23 (resp. 5.24), we observe that any `uQuery` (resp. `wQuery`) yields a linear constraint on parameters λ_i^u (resp. λ_i^w). Those linear constraints are then imposed to elements of U and W which are therefore characterized by linear constraints. Remark that sets U and W are two convex polyhedra that implicitly represent $\mathscr{U}$ and $\mathscr{W}$ at any time of the process.

Moreover, note that, for any lottery l, $u(l)$ (resp. $w(l)$) is linear in parameters λ_i^u (resp. λ_i^w). Thus computing $\underline{u}(l)$ and $\overline{w}(l)$ can be achieved by solving the following linear programs (LPs):

$$\mathscr{P}_u \begin{cases} \min\limits_{\lambda_1^u, \ldots, \lambda_m^u} \sum\limits_{i=1}^{m} \lambda_i^u \sum\limits_{x} l(x)\mathrm{I}_i(x) \\ (\lambda_1^u, \ldots, \lambda_m^u) \in \mathrm{U} \end{cases}$$

$$\mathscr{P}_w \begin{cases} \min\limits_{\lambda_1^w, \ldots, \lambda_m^w} \sum\limits_{i=1}^{m} \lambda_i^w \sum\limits_{x} l(x)\mathrm{I}_i(x) \\ (\lambda_1^w, \ldots, \lambda_m^w) \in \mathrm{W} \end{cases}$$

where LP $\mathscr{P}_u$ (resp. $\mathscr{P}_w$) is used to minimize $u(l)$ (resp. maximize $w(l)$). Note that $\mathscr{P}_w$ is here stated as a minimization problem because $\max\{1-x\} = 1 - \min\{x\}$. Values $\overline{u}(l)$ and

$\underline{w}(l)$ can similarly be obtained by changing the minimization problems into maximization problems.

5.4.5 Numerical Tests

We carried out numerical tests[4] in order to assess both the average number of queries required by our approach and the quality of the returned lottery. Interactions with the decision maker are simulated by generating answers to queries using WEU with two hidden, randomly generated, functions u_h and w_h. To model u_h and w_h, we use splines generated by a basis of $m = 12$ cubic I-spline functions as defined in Equations 5.25, 5.26. All times are wall-clock times on a 2.4 GHz Intel Core i5 machine with 8G main memory.

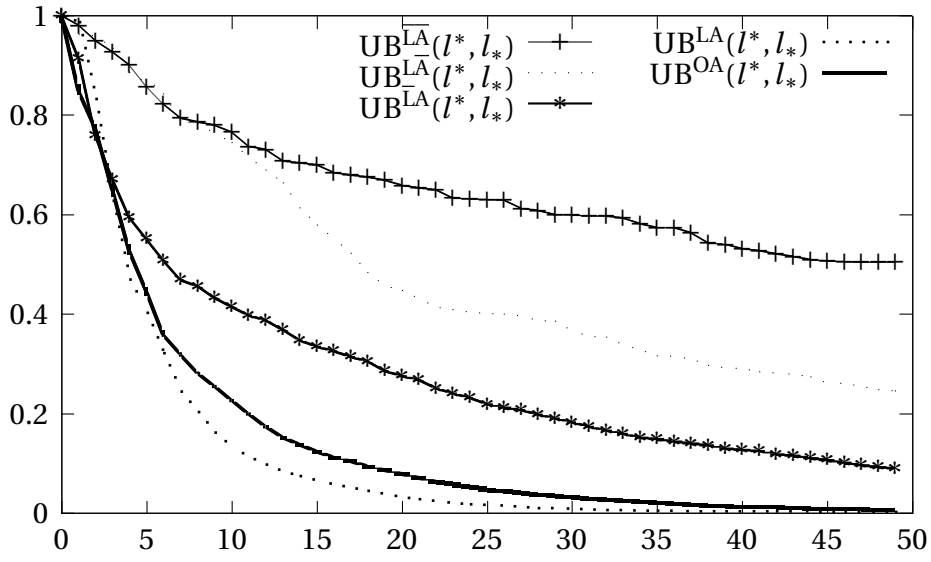


Figure 5.13: Reduction of $UB^{\mathscr{S}}(l^*, l_*)$ as the number of queries increases (results averaged over 50 runs).

Number of queries. Our query generation strategy (described in the previous section and denoted by LA hereafter) is characterized by both the rule for selecting the lottery on which a query is asked and the rule for choosing the probability parameter α. In order to assess the impact of each of these rules, we investigate other strategies relaxing either the former and/or the latter. The different elicitation strategies obtained are denoted by $\overline{\text{L}}$A, L$\overline{\text{A}}$ and $\overline{\text{LA}}$, where L (resp. $\overline{\text{L}}$) means that the queried lottery is chosen as in our strategy (resp. is chosen randomly) and A (resp. $\overline{\text{A}}$) means that parameter α is chosen as in our strategy (resp. is chosen randomly). Finally, we also consider strategy OA that, instead of asking a query on the lotteries l_* and l^*, asks a query on the *outcome* (hence the O) in their support for which the value-imprecision is the highest; those queries are easier to answer for a decision maker as lotteries can be quite complex objects. We denote by $UB^{\mathscr{S}}(l^*, l_*)$ the upper bound obtained with strategy $\mathscr{S}$, $\mathscr{S} \in \{\text{LA}, \overline{\text{L}}\text{A}, \text{L}\overline{\text{A}}, \overline{\text{LA}}\}$. Figure 5.13 plots the decrease of $UB^{\mathscr{S}}(l^*, l_*)$ according to the number of queries averaged over 50 randomly generated sets $\mathscr{L}$ of possible lotteries. Each set $\mathscr{L}$ contains 1000 lotteries such that no stochastic dominance relation exists between them. The support of each lottery has a size generated uniformly in $\{1, \ldots, 10\}$ and consists of values generated uniformly in $(0, 1)$.

[4]Implemented in Java using Gurobi 5.6.3 as solver for the LPs. Times are wall-clock times on a 2.4 GHz Intel Core i5 machine with 8G main memory.

One observes that $\mathrm{UB}^{\mathrm{LA}}(l^*, l_*)$ decreases quickly with the number of queries; for instance, $\mathrm{UB}^{\mathrm{LA}}(l^*, l_*) \leq 0.05$ after only 15 queries. If queries are issued on outcomes (strategy OA), the performance is slightly lower and 22 queries are required to reach $\mathrm{UB}^{\mathrm{OA}}(l^*, l_*) = 0.05$. By contrast, the other strategies $\mathscr{S}$ are not able to reduce effectively $\mathrm{UB}^{\mathscr{S}}(l^*, l_*)$.

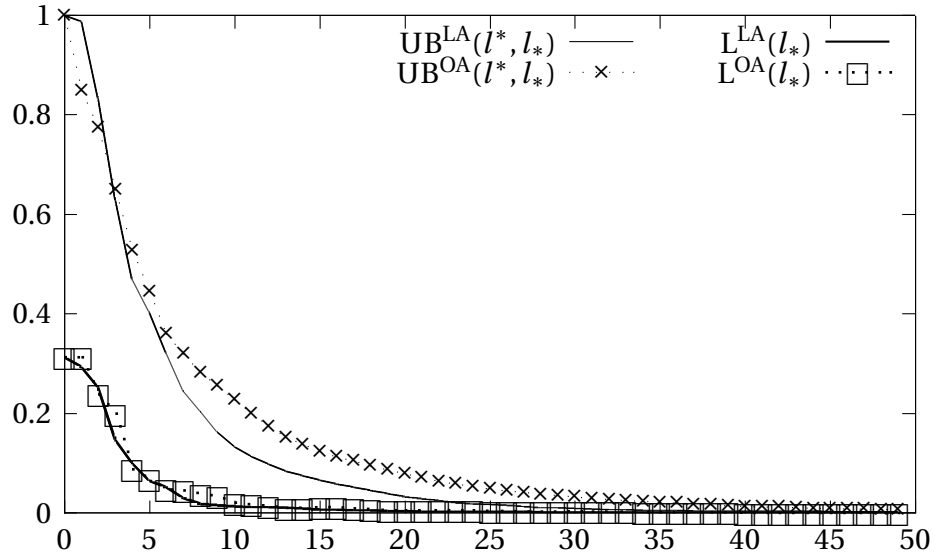


Figure 5.14: Reduction of $\mathrm{UB}^{\mathscr{S}}(l^*, l_*)$ and $\mathrm{L}^{\mathscr{S}}(l_*)$ as the number of queries increases (results averaged over 50 runs).

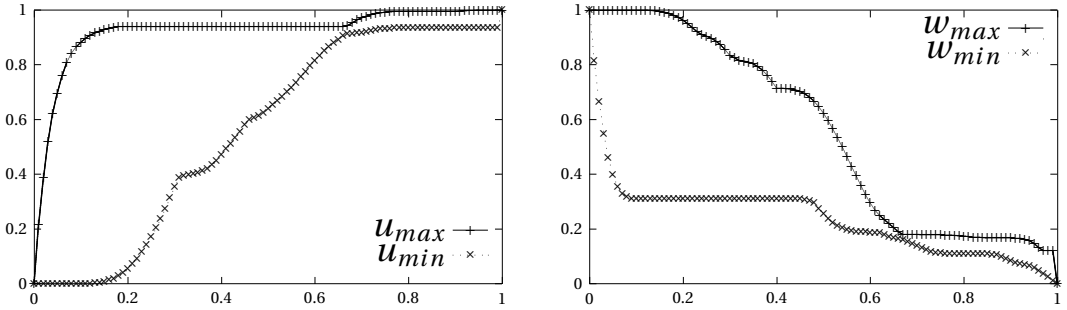
Quality of the returned lottery. We recall that the bound $\mathrm{UB}^{\mathscr{S}}(l^*, l_*)$ is a pessimistic estimate of the actual loss $\mathrm{L}^{\mathscr{S}}(l_*)$ that would be entailed by stopping the elicitation process and choosing lottery l_*:

$$\mathrm{L}^{\mathscr{S}}(l_*) = \max_{l' \in \mathscr{L}} \{u_h(l')\, w_h(l_*) - u_h(l_*)\, w_h(l')\}.$$

Figure 5.14 evaluates how far $\mathrm{UB}^{\mathscr{S}}(l^*, l_*)$ is from $\mathrm{L}^{\mathscr{S}}(l_*)$ for $\mathscr{S} \in \{\mathrm{LA}, \mathrm{OA}\}$ (the two best strategies with respect to the number of queries). We observe that during the entire elicitation process, $\mathrm{L}^{\mathscr{S}}(l_*)$ is much lower than $\mathrm{UB}^{\mathscr{S}}(l^*, l_*)$ for both $\mathscr{S} = \mathrm{LA}$ and $\mathscr{S} = \mathrm{OA}$. In fact, $\mathrm{L}^{\mathscr{S}}(l_*)$ reaches 0 after only 10 queries for both elicitation strategies. This shows that, even if one decides an early interruption of the elicitation process (with respect to the estimate $\mathrm{UB}^{\mathscr{S}}(l^*, l_*)$), the chosen lottery l_* is likely to be optimal with respect to u_h and w_h.

Interest of the incremental approach. In Figure 5.15a and 5.15b, we plot the model that has been learned with our strategy on one run of 50 queries. The lines u_{min} and u_{max} (resp. w_{min} and w_{max}) give the minimal and maximal values that can be taken by the spline function u (resp. w) approximating u_h (resp. w_h) in each point according to the answers of the decision maker. Interestingly enough, it appears that functions u and w are far from being fully elicited, while the estimated loss is already tiny and lottery l_* is almost surely optimal. This illustrates the main interest of the incremental approach that interweaves elicitation and optimization.

Computation time. We monitored the computation time required by our approach as it is crucial that the decision maker does not wait a prohibitively long time between two queries. Even for a large set $\mathscr{L}$ consisting of 1400 lotteries, the overall computation time



(a) Minimal and maximal values that can be taken by the spline representing u_h at the end of one run.

(b) Minimal and maximal values that can be taken by the spline representing w_h at the end of one run

Figure 5.15: Model that has been learned on one run.

for 30 queries is 12 seconds and, for instance, determining the 20th query to be asked requires less than 0.5 seconds (average times on 50 runs).

Other experiments. We tested how our approach is affected by varying the size of $\mathscr{L}$ and the maximum number of branches in the lotteries. Whichever strategy $\mathscr{S}$ is used, we did not observe a significant increase in the number of queries required to decrease efficiently $\mathrm{UB}^{\mathscr{S}}(l^*, l_*)$.

5.4.6 Conclusion of the Section: the Incremental Elicitation of WEU

In this section, we designed an elicitation protocol dedicated to the WEU model. Then, we showed how it could be used in an incremental elicitation procedure to solve the problem of identifying, within a given set of lotteries, a near-optimal solution. The efficiency of the method relies on a redefinition of functions u and w as monotone spline functions, which considerably reduces the elicitation burden while keeping a high descriptive power. Moreover, we have shown that WEU has the advantage that functions u and w can easily be elicited concurrently in an incremental elicitation setting.

5.5 Conclusion

In this chapter, we have addressed several decision problems under risk. The traditional formulation of these decision problems require to know the exact numerical values of some parameters (e.g., the reward function of an MDP for the expected total discounted reward criterion). In this chapter, we studied three settings in which this information was not available, i.e., the preferences were incompletely specified or only of ordinal nature.

In a first section, we only assumed that we had a preference order over the possible episodes of a finite horizon Markov decision process. In this ordinal setting, we designed a solution method to find a near-optimal policy with respect to a quantile criterion. This solution method requires to solve repeatedly the Markov decision process with different expected utility functions and returns a near-optimal wealth Markovian policy.

In a second section, we focused on an interactive optimization method, called Interactive Value Iteration (IVI), to solve an infinite horizon Markov decision process with an imprecisely known reward function. IVI determines a near-optimal policy with respect

to the total discounted expected reward criterion by querying a tutor when the uncertainty over the reward function does not make it possible for the method to continue the resolution process. We designed an improved version of the original IVI method by implementing several ideas (e.g., reasoning about the order in which we ask the queries) that can greatly reduce the number of queries issued by the procedure.

Lastly, in a third section, we designed an elicitation protocol to elicit the WEU model. Then we showed how to integrate this protocol in an incremental procedure in order to identify a near-optimal solution with only few queries. The efficiency of our method relies on a maxmin optimization criterion to guide the querying step and on a compact representation of the parameters of the WEU model using spline functions.

5.6 References

P. Abbeel and A.Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proc. Twenty-first Inter. Conf. on Machine Learning*, International Conference on Machine Learning '04, New York, NY, USA, 2004. ACM. 159

R. Akrour, M. Schoenauer, and M. Sebag. APRIL: active preference-learning based reinforcement learning. *CoRR*, abs/1208.0984, 2012. 149

P. Alizadeh, Y. Chevaleyre, and J.-D. Zucker. Approximate regret based elicitation in Markov decision process. In *Computing & Communication Technologies-Research, Innovation, and Vision for the Future (RIVF), 2015 IEEE RIVF International Conference on*, pages 47–52. IEEE, 2015. 160

P. Alizadeh, Y. Chevaleyre, and F. Lévy. Advantage based value iteration for markov decision processes with unknown rewards. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 3837–3844. IEEE, 2016. 160

J.A. Bagnell, A.Y. Ng, and J.G. Schneider. Solving uncertain markov decision processes. Technical report, Carnegie Mellon University, 2001. 159

L. Barrett and S. Narayanan. Learning all optimal policies with multiple criteria. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 41–47, New York, NY, USA, 2008. ACM. 160

N. Bäuerle and J. Ott. Markov decision processes with average value-at-risk criteria. *Mathematical Methods of Operations Research*, 74(3):361–379, 2011. 148

J.C. Beatty and B.A. Barsky. *An introduction to splines for use in computer graphics and geometric modeling*. Morgan Kaufmann, 1987. 178

N. Benabbou and P. Perny. On possibly optimal tradeoffs in multicriteria spanning tree problems. In *Algorithmic Decision Theory - 4th International Conference, ADT 2015, Lexington, KY, USA, September 27-30, 2015, Proceedings*, pages 322–337, 2015. 173

N. Benabbou, P. Perny, and P. Viappiani. Incremental elicitation of choquet capacities for multicriteria decision making. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 87–92, 2014. 173

D.F. Benoit and D. Van den Poel. Benefits of quantile regression for the analysis of customer lifetime value in a contractual setting: An application in financial services. *Expert Systems with Applications*, 36:10475–10484, 2009. 148, 157

H. Bleichrodt and J.L. Pinto. A parameter-free elicitation of the probability weighting function in medical decision analysis. *Management science*, 46(11):1485–1496, 2000. 173

H. Bleichrodt, A. Cillo, and E. Diecidue. A quantitative measurement of regret theory. *Management Science*, 56(1):161–175, 2010. 173

J. Boger, J. Hoey, P. Poupart, C. Boutilier, G. Fernie, and A. Mihailidis. A planning system based on Markov decision processes to guide people with dementia through activities of daily living. *IEEE Transactions on Information Technology in Biomedicine*, 2006. 168, 171

V. Borkar and R. Jain. Risk-constrained Markov decision processes. *IEEE Trans. on Automatic Control*, 59(9):2574–2579, 2014. 148

M. Boussard, M. Bouzid, A.-I. Mouaddib, R. Sabbadin, and P. Weng. *Markov Decision Processes in Artificial Intelligence*, chapter Non-Standard Criteria, pages 319–359. Wiley, 2010. 148

C. Boutilier, R. Das, J.O. Kephart, G. Tesauro, and W.E. Walsh. Cooperative Negotiation in Autonomic Systems Using Incremental Utility Elicitation. In *In Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 89–97, 2003. 168, 171

D. Braziunas. *Decision-theoretic elicitation of generalized additive utilities*. PhD thesis, University of Toronto (Canada), 2012. 173

R. Busa-fekete, B. Szörényi, P. Weng, W. Cheng, and E. Hullermeier. Preference-based evolutionary direct policy search, 2014. 149

A. Carpentier and M. Valko. Extreme bandits. In *Neural Information Processing Systems*, 2014. 149

G Casella and E I George. Explaining the Gibbs sampler. *The American Statistician*, 46:167–174, 1992. 165

U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. In *AAAI/IAAI*, pages 363–369, 2000. 173

S.H. Chew. A generalization of the quasilinear mean with applications to the measurement of income inequality and decision theory resolving the Allais paradox. *Econometrica: Journal of the Econometric Society*, pages 1065–1092, 1983. 174

Y. Chow and M. Ghavamzadeh. Algorithms for CVaR optimization in MDPs. In *Neural Information Processing Systems*, 2014. 148

G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. *ACM SIGOPS Operating Systems Review*, 41(6):205–220, 2007. 148

E. Delage and S. Mannor. Percentile optimization in uncertain markov decision processes with application to efficient exploration. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 225–232, New York, NY, USA, 2007. ACM. 148

M.M. Fard and J. Pineau. Non-deterministic policies in markovian decision processes. *J. Artif. Intell. Res.(JAIR)*, 40:1–24, 2011. 160

J.A. Filar, L.C.M. Kallenberg, and H.-M. Lee. Variance-penalized Markov decision processes. *Mathematics of Operations Research*, 14:147–161, 1989. 148

J.A. Filar, D. Krass, and K.W. Ross. Percentile performance criteria for limiting average Markov decision processes. *IEEE Trans. on Automatic Control*, 40(1):2–10, 1995. 148

J.E. Filar. Percentiles and markov decision processes. *Operations Research Letters*, 2:13–15, 1983. 148

C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pages 49–58, 2016. 159

P.C. Fishburn. Transitive measurable utility. *Journal of Economic Theory*, 31(2):293–317, 1983. 175

V. Freire da Silva and A. Reali Costa. A geometric approach to find nondominated policies to imprecise reward mdps. *Machine Learning and Knowledge Discovery in Databases*, pages 439–454, 2011. 160

H. Gilbert, O. Spanjaard, P. Viappiani, and P. Weng. Reducing the Number of Queries in Interactive Value Iteration. In *Proceedings of the International Conference on Algorithmic Decision Theory 2015*, 2015. 146

H. Gilbert, N. Benabbou, P. Perny, O. Spanjaard, and P. Viappiani. Incremental Decision Making Under Risk with the Weighted Expected Utility Model. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2017. 146

H. Gilbert, P. Weng, and Y. Xu. Optimizing Quantiles in Preference-based Markov Decision Processes. In *Proceedings of AAAI 2017*, 2017. 146

R. Givan, S. Leach, and T. Dean. Bounded-parameter Markov decision process. *Artif. Intell.*, 122(1-2):71–109, 2000. 159

P. Hou, W. Yeoh, and P. Varakantham. Revisiting risk-sensitive MDPs: New algorithms and results. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 136–144, 2014. 148

J.-Y. Jaffray. Implementing resolute choice under uncertainty. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 1998. 152

G. Jeantet, P. Perny, and O. Spanjaard. Sequential Decision Making with Rank Dependent Utility: a Minimax Regret Approach. In *Proc. of AAAI 2012*, pages 1931–1937, 2012. 152

M. Jin, A. Damianou, Abbeel P., and C. Spanos. Inverse reinforcement learning via deep gaussian process. 2017. 159

P. Jorion. *Value-at-Risk: The New Benchmark for Managing Financial Risk*. McGraw-Hill, 2006. 148

P.J. Korhonen and J. Laakso. A visual interactive method for solving the multiple criteria problem. *European Journal of Operational Research*, 24(2):277–287, 1986. 173

S. Levine, Z. Popovic, and V. Koltun. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems*, pages 19–27, 2011. 159

Y. Liu and S. Koenig. Functional value iteration for decision-theoretic planning with general utility functions. In *Proceedings of AAAI 2006*, volume 21, page 1186, 2006. 148, 155

D.J. Lizotte, M. Bowling, and S.A. Murphy. Linear fitted-Q iteration with multiple reward functions. *Journal of Machine Learning Research*, 13:3253–3295, 2012. 160

S. Mannor and J. Tsitsiklis. Mean-variance optimization in Markov decision processes. In *International Conference on Machine Learning*, 2011. 148

E.F. McClennen. *Rationality and dynamic choice: Foundational explorations*. Cambridge university press, 1990. 151

T.W. Archibald K. McKinnon and L.C. Thomas. On the generation of Markov decision processes. In *Journal of the Operational Research Society*, pages 354–361, 1995. 156

T.H. Nguyen, A. Yadav, B. An, M. Tambe, and C. Boutilier. Regret-based optimization and preference elicitation for Stackelberg security games with uncertainty. In *Proceedings of AAAI 2014*, pages 756–762, 2014. 174

P. Perny, P. Viappiani, and A. Boukhatem. Incremental preference elicitation for decision making under risk with the rank-dependent utility model. In *Proceedings of UAI 2016*, July 2016. 174, 178

B. Piot, M. Geist, and O. Pietquin. Boosted and reward-regularized classification for apprenticeship learning. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, pages 1249–1256, 2014. 159

L.A. Prashanth and M. Ghavamzadeh. Actor-critic algorithms for risk-sensitive MDPs. In *Neural Information Processing Systems*, pages 252–260, 2013. 148

D. Ramachandran and E. Amir. Bayesian Inverse Reinforcement Learning. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 51:2586–2591, 2007. 159

J.O. Ramsay. Monotone regression splines in action. *Statistical science*, pages 425–441, 1988. 178

K. Regan and C. Boutilier. Regret-based reward elicitation for markov decision processes. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 444–451, Arlington, Virginia, United States, 2009. AUAI Press. 160, 173

K. Regan and C. Boutilier. Robust policy computation in reward-uncertain mdps using nondominated policies. In Maria Fox and David Poole, editors, *AAAI*. AAAI Press, 2010. 160

K. Regan and C. Boutilier. Eliciting additive reward functions for markov decision processes. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2159–2164. AAAI Press, 2011. 160

K. Regan and C. Boutilier. Robust online optimization of reward-uncertain mdps. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2165–2171. AAAI Press, 2011. 160

S. Rosenthal and M.M. Veloso. Monte Carlo preference elicitation for learning additive reward functions. In *RO-MAN*, pages 886–891. IEEE, 2012. 166

M. Rostek. Quantile maximization in decision theory. *Review of Economic Studies*, 77(1):339–371, 2010. 150

U.A. Syed. *Reinforcement learning without rewards*. Princeton University, 2010. 159

B. Szörenyi, R. Busa-Fekete, P. Weng, and E. Hüllermeier. Qualitative multi-armed bandits: A quantile-based approach. In *International Conference on Machine Learning*, pages 1660–1668, 2015. 149

A.L. Thomaz, G. Hoffman, and C. Breazeal. Real-Time Interactive Reinforcement Learning for Robots. In *AAAI Workshop Human Comprehensible Machine Learning*, pages 9–13, 2005. 159

K. Van Moffaert, M.M. Drugan, and A. Nowé. Learning sets of pareto optimal policies. In *Thirteenth International Conference on Autonomous Agents and Multiagent Systems-Adaptive Learning Agents Workshop (ALA)*, 2014. 160

P. Viappiani and C. Boutilier. Optimal Bayesian Recommendation Sets and Myopically Optimal Choice Query Sets. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.*, pages 2352–2360, 2010. 173

P. Wakker and D. Deneffe. Eliciting von neumann-morgenstern utilities when probabilities are distorted or unknown. *Management science*, 42(8):1131–1150, 1996. 173

T. Wang and C. Boutilier. Incremental Utility Elicitation with the Minimax Regret Decision Criterion. In *Proceedings of the International Joint Conference on Artificial Intelligence'03*, pages 309–316, 2003. 173

P. Weng and B. Zanuttini. Interactive value iteration for markov decision processes with unknown rewards. In Francesca Rossi, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*. IJCAI/AAAI, 2013. 147, 159, 160, 161, 162, 163

P. Weng, R. Busa-Fekete, and E. Hüllermeier. Interactive Q-Learning with Ordinal Rewards and Unreliable Tutor. In *ECML/PKDD Workshop Reinforcement Learning with Generalized Feedback*, September 2013. 160

P. Weng. Markov decision processes with ordinal rewards: Reference point-based preferences. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling, ICAPS 2011, Freiburg, Germany June 11-16, 2011*, 2011. 148, 161

P. Weng. Ordinal decision models for markov decision processes. In *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, pages 828–833, 2012. 148, 151, 161

D.J. White. Mean, variance, and probabilistic criteria in finite Markov decision processes: A review. *Journal of Optimization Theory and Applications*, 56(1):1–29, 1988. 148

M.A. Wiering and E.D. De Jong. Computing optimal stationary policies for multi-objective markov decision processes. In *Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007. IEEE International Symposium on*, pages 158–165. IEEE, 2007. 160

A. Wilson, A. Fern, and P. Tadepalli. A bayesian approach for policy learning from trajectory preference queries. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1133–1141. Curran Associates, Inc., 2012. 149

C. Wirth and G. Neumann. Model-free preference-based reinforcement learning. In *EWRL*, 2015. 149

R. Wolski and J. Brevik. QPRED: Using quantile predictions to improve power usage for private clouds. Technical report, UCSB, 2014. 148

C. Wu and Y. Lin. Minimizing risk models in Markov decision processes with policies depending on target values. *Journal of mathematical analysis and applications*, 231:41–67, 1999. 148

H. Xu and S. Mannor. Parametric regret in uncertain Markov decision processes. In *CDC*, pages 3606–3613. IEEE, 2009. 159, 160

X. Yin and B. Sinopoli. Adaptive robust optimization for coordinated capacity and load control in data centers. In *International Conference on Decision and Control*, 2014. 156

J.Y. Yu and E. Nikolova. Sample complexity of risk-averse bandit-arm selection. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2013. 149

S.X. Yu, Y. Lin, and P. Yan. Optimization models for the first arrival target distribution function in discrete time. *Journal of mathematical analysis and applications*, 225:193–223, 1998. 148

S. Zionts and J. Wallenius. An interactive programming method for solving the multiple criteria problem. *Management science*, 22(6):652–663, 1976. 173

Chapter 6

A Double Oracle Approach for Robust Combinatorial Optimization Problems

" I have a robust sense of humour which helps me deal with problems. "

P. Mayle

Section Contents

Summary of the chapter

In this chapter, we present a game-theoretic view of robust combinatorial optimization problems with interval data. This game-theoretic view enables to design an efficient procedure to compute a lower bound on the value of an optimal minmax regret solution. We show how this lower bound compares to other lower bounds proposed in the literature

and show how this lower bound can be efficiently integrated in a branch and bound procedure to find an optimal minmax regret solution. Numerical tests are provided that point out the efficiency of the approach.

This chapter is based on the following publication [Gilbert and Spanjaard, 2017].

6.1 Introduction

In this chapter, we bridge the gap between the setting of combinatorial optimization (which is succinctly presented in Chapter 2) and the one of robust optimization (which is succinctly presented in Chapter 1). Indeed, we study in this chapter robust combinatorial optimization problems.

The definition of an instance of a combinatorial optimization problem requires to specify parameters (e.g., costs of edges in a network problem) which can be uncertain or imprecise (see Section 1.4). For instance, instead of specifying scalar values, there are situations where only an interval of possible values is known for each parameter. It is then assumed that the different parameters (e.g., the different costs of edges) may take any value of the given interval independently of the other parameters. The assignment of a scalar value in each interval is called a *scenario*.

One may then optimize the worst case performance (i.e., performance of the considered solution in the worst possible scenario) but this approach often leads to an overly conservative solution. A less conservative approach, which is known as minmax regret optimization, minimizes the maximum difference in the objective value of a solution over all scenarios, compared to the best possible objective value attainable in this scenario. Unfortunately, the minmax regret versions of combinatorial optimization problems with interval data are often NP-hard.

For tackling minmax regret optimization problems, it is therefore useful to investigate efficient algorithms with performance guarantee. Kasperski and Zielinski [2006] proved that the algorithm that returns a midpoint solution (i.e., an optimal solution in the scenario where one takes the middle of each interval) has an approximation ratio of 2. Recently, Chassein and Goerigk [2015] presented a new approach for determining a lower bound on the objective value of a minmax regret solution. It allows them to compute a tighter *instance-dependent* approximation ratio of a midpoint solution, that is of course at most 2. Moreover they showed how their bound can be used in a branch and bound framework for the minmax regret version of the shortest path problem with interval data, and obtained an improvement on the computation times compared to state-of-the-art approaches, especially when the uncertainty is high. In this chapter, following Chassein and Goerigk, we also present a very general approach to compute a lower bound on the objective value of a minmax regret solution, which further improves the instance-dependent approximation ratio of a midpoint solution.

The presented approach relies on a game-theoretic view of robust optimization. Such a game-theoretic view has already been adopted in the literature by Mastin *et al.* [2015]. They view robust optimization as a two-player zero-sum game where one player (optimizing player) selects a feasible solution, and the other one (adversary) selects a possible scenario for the values of the parameters. More precisely, they consider a randomized model where the optimizing player selects a probability distribution over solutions (called a mixed solution hereafter) and the adversary selects values with knowledge of the player's distribution but not its realization. They show that the determination of a mixed Nash equilibrium of this game (and thus of a minmax regret mixed solution for the optimizing player) can be performed in polynomial time provided the standard version of

the problem (i.e., where the precise values of the parameters are known) is itself polynomial. They use a linear programming formulation that may have an exponential number of constraints, but for which a polynomial time separation oracle amounts to solving an instance of the standard problem.

We also compute a mixed Nash equilibrium of the game but we use a double oracle approach [McMahan *et al.*, 2003] that is shown to be more efficient in computation times. Furthermore, we show how the bound resulting from this computation can be used in a branch and bound algorithm for the minmax regret version of a (deterministic) robust optimization problem. We illustrate the interest of this approach on the robust shortest path problem with interval data. Comparing our results with the ones obtained by Chassein and Goerigk [2015] and Montemanni *et al.* [2004] we find considerable improvements in computation times on some classes of instances.

The remainder of the chapter is structured as follows. Section 6.2 recalls the definition of a minmax regret problem and introduces the notations of the chapter. In Section 6.3, a game-theoretic view of minmax regret optimization, similar to the one proposed by Mastin *et al.* [2015], is developed. This view helps us derive a precise lower bound on the objective value of a minmax regret solution. The method to compute this lower bound uses a double oracle algorithm described in Section 6.3.4. In Section 6.4 we discuss how our work can be used efficiently in a branch and bound algorithm. Finally, in Section 6.5, the approach is tested on the minmax regret version of the shortest path problem with interval data.

6.2 Minmax Regret Problems

A combinatorial optimization problem is stated as follows (in minimization):

$$\min_{\mathbf{x}\in\mathscr{X}} \sum_{i=1}^{n} c_i x_i \tag{6.1}$$

where $\mathscr{X} \subset \{0,1\}^n =: \mathbb{B}^n$ denotes the set of feasible solutions, and c_i represents the cost of element i. We refer to (6.1) as the standard optimization problem.

In robust optimization, instead of considering only a single cost vector $\mathbf{c}$, we assume there is an uncertainty set $\mathscr{U} \subset \mathbb{R}^n$ of different possible cost vectors. Each possible cost vector $\mathbf{c}$ is also called a scenario. Two approaches can be distinguished according to the way the set of scenarios is defined: the *interval model* where each c_i takes value in an interval $[\underline{c}_i, \overline{c}_i]$ and where the set of scenarios is defined implicitly as the Cartesian product $\mathscr{U} = \times_{i=1}^{n} [\underline{c}_i, \overline{c}_i]$; the *discrete scenario model* where the possible scenarios consist in a finite set of cost vectors. Intuitively, a *robust* solution is a solution that remains suitable whatever scenario finally occurs. Several criteria have been proposed to formalize this: the *minmax* criterion consists of evaluating a solution on the basis of its worst value over all scenarios, and the *minmax regret* criterion consists of evaluating a solution on the basis of its maximal deviation from the optimal value over all scenarios.

We consider here the approach by minmax regret optimization in the interval model. For a feasible solution $\mathbf{x}$, the regret $Reg(\mathbf{x},\mathbf{c})$ induced by a scenario $\mathbf{c}$ is defined as the difference between the realized objective value $val(\mathbf{x},\mathbf{c})$ and the best possible objective value $\min_{\mathbf{y}\in\mathscr{X}} val(\mathbf{y},\mathbf{c})$ in this scenario, where $val(\mathbf{x},\mathbf{c}) = \sum_{i=1}^{n} c_i x_i$. Similarly, for a feasible solution $\mathbf{x}$, the regret $Reg(\mathbf{x},\mathbf{y})$ induced by another feasible solution $\mathbf{y}$ is defined by maximizing over possible scenarios $\mathbf{c} \in \mathscr{U}$ the difference $Reg(\mathbf{x},\mathbf{y},\mathbf{c})$ between the realized objective values $val(\mathbf{x},\mathbf{c})$ and $val(\mathbf{y},\mathbf{c})$. The (maximal) regret $Reg(\mathbf{x})$ of a feasible solution

$\mathbf{x}$ is then defined as the maximal regret induced by a scenario or equivalently the maximal regret induced by another solution. More formally, we have:

$$Reg(\mathbf{x},\mathbf{y},\mathbf{c}) = val(\mathbf{x},\mathbf{c}) - val(\mathbf{y},\mathbf{c}) \tag{6.2}$$

$$Reg(\mathbf{x},\mathbf{c}) = \max_{\mathbf{y}\in\mathscr{X}} Reg(\mathbf{x},\mathbf{y},\mathbf{c}) = val(\mathbf{x},\mathbf{c}) - val(\mathbf{x}^{\mathbf{c}},\mathbf{c}) \tag{6.3}$$

$$Reg(\mathbf{x},\mathbf{y}) = \max_{\mathbf{c}\in\mathscr{U}} Reg(\mathbf{x},\mathbf{y},\mathbf{c}) \tag{6.4}$$

$$Reg(\mathbf{x}) = \max_{\mathbf{c}\in\mathscr{U}} Reg(\mathbf{x},\mathbf{c}) = \max_{\mathbf{y}\in\mathscr{X}} Reg(\mathbf{x},\mathbf{y}) \tag{6.5}$$

where $\mathbf{x}^{\mathbf{c}} = \arg\min_{\mathbf{x}\in\mathscr{X}} \sum_{i=1}^{n} c_i x_i$ is an optimal solution of the standard optimization problem for cost vector $\mathbf{c}$. In the following, for brevity, we may denote by $val^*(\mathbf{c}) = val(\mathbf{x}^{\mathbf{c}},\mathbf{c})$ the objective value of an optimal solution of the standard optimization problem for cost vector $\mathbf{c}$.

In a minmax regret optimization problem, the goal is to find a feasible solution with minimal regret. The minmax regret optimization problem can thus be formulated as:

$$\min_{\mathbf{x}\in\mathscr{X}} Reg(\mathbf{x}) \tag{6.6}$$

The optimal value for this problem is denoted by OPT in the remainder of the chapter. As stated in the introduction of this chapter, most minmax regret versions of standard optimization problems are NP-hard [Aissi *et al.*, 2009]. It is therefore worth investigating approximation algorithms. To the best of our knowledge, the most general polynomial time approach in this concern is the midpoint algorithm proposed by Kasperski and Zielinski [2006] which returns a solution minimizing $\sum_{i=1}^{n} \hat{c}_i x_i$, where $\hat{c}_i = (\underline{c}_i + \overline{c}_i)/2$. We denote by $\mathbf{x}^{\hat{\mathbf{c}}}$ such a solution.

The regret of a midpoint solution $\mathbf{x}^{\hat{\mathbf{c}}}$ is always not more than 2OPT [Kasperski and Zielinski, 2006]. While there are problem instances where this approximation guarantee is tight, the regret is actually smaller for a lot of instances. In order to account for that, one needs to design a tighter *instance-dependent* ratio. In this line of research, Chassein and Goerigk [2015] show how to improve the 2-approximation guarantee for every specific instance with only small computational effort. To this end they propose a new lower bound $\mathrm{LB}_{\mathrm{CG}}$ on the minmax regret value. This lower bound is then used to compute an instance-dependent approximation guarantee $Reg(\mathbf{x}^{\hat{\mathbf{c}}})/\mathrm{LB}_{\mathrm{CG}}$ for a midpoint solution $\mathbf{x}^{\hat{\mathbf{c}}}$. This guarantee equals 2 in the worst case.

In this chapter, we investigate another lower bound based on game-theoretic arguments. This lower bound corresponds to the value of a mixed Nash equilibrium in a game that will be specified in the sequel. We prove that this lower bound is tighter than the one presented by Chassein and Goerigk [2015]. Therefore, this lower bound further improves the instance-dependent approximation guarantee of a midpoint solution.

Note that Mastin *et al.* [2015] investigated a similar game-theoretic view of robust optimization. They showed how to compute a mixed Nash equilibrium by using a mathematical programming formulation involving an exponential number of constraints. They propose to solve this program via a cutting-plane method (this type of method is described in subsection 3.1.4 on page 87) relying on a polynomial time separation oracle provided the standard version of the tackled problem is itself polynomial. In this case, the complexity of computing a mixed Nash equilibrium is polynomial by the polynomial time equivalence of *OPTIMIZATION* and *SEPARATION* (see Theorem 10 on page 89) but no numerical tests have been carried out in their paper. We consider here the same game but we propose a *double oracle* approach to compute a mixed Nash equilibrium. While

the proposed double oracle approach is not a polynomial time algorithm, it will reveal more efficient in practice than a cutting plane approach.

In the next section, a game-theoretic view of robust combinatorial optimization is presented.

6.3 Game-Theoretic View

The minmax regret optimization problem defined by (6.6) induces a zero-sum two-player game where the sets of pure strategies are the set of feasible solutions $\mathscr{X}$ for player 1 (also called **x**-player) and the set of scenarios $\mathscr{U}$ for player 2 (also called **c**-player). The resulting payoff is then given by $Reg(\mathbf{x},\mathbf{c})$, the regret of using solution $\mathbf{x}$ in scenario $\mathbf{c}$. Obviously, the **x**-player (resp. **c**-player) aims to minimize (resp. maximize) $Reg(\mathbf{x},\mathbf{c})$. The **x**-player (resp. **c**-player) can also play a mixed strategy by playing according to a probability distribution $P_{\mathscr{X}}$ (resp. $P_{\mathscr{U}}$) over set $\mathscr{X}$ (resp. $\mathscr{U}$). We denote by $P_{\mathscr{X}}(\mathbf{x})$ (resp. $P_{\mathscr{U}}(\mathbf{c})$) the probability to play strategy $\mathbf{x}$ (resp. $\mathbf{c}$) in mixed strategy $P_{\mathscr{X}}$ (resp. $P_{\mathscr{U}}$). In the following, we restrict ourselves to mixed strategies over a *finite* set of pure strategies. We will show at the end of this section that this assumption is not restrictive. We denote by $\Delta_{\mathscr{X}}$ (resp. $\Delta_{\mathscr{U}}$) the set of all possible mixed strategies for the **x**-player (resp. **c**-player).

The regret functions $Reg(.), Reg(.,.), Reg(.,.,.)$ are then extended to mixed strategies by linearity in probability. For instance, if the **c**-player plays $P_{\mathscr{U}}$, the regret of the **x**-player for playing $P_{\mathscr{X}}$ is the expectancy $\sum_{\mathbf{x}\in\mathscr{X}}\sum_{\mathbf{c}\in\mathscr{U}} P_{\mathscr{X}}(\mathbf{x})P_{\mathscr{U}}(\mathbf{c})Reg(\mathbf{x},\mathbf{c})$.

6.3.1 Best Response Functions and Lower Bound

Given the strategy of one player, a best reponse is an optimal pure strategy of the other player. We will call a best **x**-response (resp. **c**-response) a best response of the **x**-player (resp. **c**-player). We can now state some results on the best responses of each player.

Observation 2. *A best* $\mathbf{x}$*-response to a scenario* $\mathbf{c}$ *is given by* $\mathbf{x}^{\mathbf{c}}$.

The next observation is a well known characterization of a worst scenario for a given feasible solution and has been used in many related papers [Aissi *et al.*, 2009; Karasan *et al.*, 2002].

Observation 3. *A best* $\mathbf{c}$*-response to a feasible solution* $\mathbf{x}$ *is given by* $\mathbf{c}^{\mathbf{x}}$ *defined by* $c_i^{\mathbf{x}} = \overline{c}_i$ *if* $x_i = 1$ *and* $\underline{c}_i$ *otherwise.*

The scenario $\mathbf{c}^{\mathbf{x}}$ belongs to the subset of *extreme scenarios*: a scenario $\mathbf{c}$ is said to be an extreme scenario if $\forall i \in \{1,\dots,n\}$, $c_i = \underline{c}_i$ or $c_i = \overline{c}_i$. Given an extreme scenario $\mathbf{c}$, we will denote by $\neg\mathbf{c}$ the opposite extreme scenario defined by $\neg c_i = \underline{c}_i$ if $c_i = \overline{c}_i$ and $\overline{c}_i$ otherwise. While $\mathbf{c}^{\mathbf{x}}$ is the most penalizing scenario for solution $\mathbf{x}$, $\neg\mathbf{c}^{\mathbf{x}}$ is the most favorable scenario for solution $\mathbf{x}$.

The following observation will reveal useful in the proof of Proposition 15 below.

Observation 4. *Given two feasible solutions* $\mathbf{x}$ *and* $\mathbf{y}$ *in* $\mathscr{X}$, $Reg(\mathbf{x},\mathbf{y})$ *is achieved by both* $\mathbf{c}^{\mathbf{x}}$ *and* $\neg\mathbf{c}^{\mathbf{y}}$*:* $Reg(\mathbf{x},\mathbf{y}) = Reg(\mathbf{x},\mathbf{y},\mathbf{c}^{\mathbf{x}}) = Reg(\mathbf{x},\mathbf{y},\neg\mathbf{c}^{\mathbf{y}})$.

Observation 5 gives us a convenient way to determine a best response for the **x**-player against a mixed strategy of the **c**-player.

Observation 5. *A best* $\mathbf{x}$*-response to a mixed strategy* $P_{\mathscr{U}}$ *is given by* $\mathbf{x}^{\tilde{\mathbf{c}}}$ *where* $\tilde{\mathbf{c}}$ *is defined by* $\tilde{\mathbf{c}} = \sum_{\mathbf{c}\in\mathscr{U}} P_{\mathscr{U}}(\mathbf{c})\mathbf{c}$.

Proof. It follows from the following sequence of equalities:

$$\begin{aligned}\min_{\mathbf{x}\in\mathscr{X}} Reg(\mathbf{x}, \mathrm{P}_{\mathscr{U}}) &= \min_{\mathbf{x}\in\mathscr{X}} \left(\sum_{\mathbf{c}\in\mathscr{U}} \mathrm{P}_{\mathscr{U}}(\mathbf{c})(val(\mathbf{x},\mathbf{c}) - val^*(\mathbf{c})) \right) \\ &= \min_{\mathbf{x}\in\mathscr{X}} \left(\sum_{\mathbf{c}\in\mathscr{U}} \mathrm{P}_{\mathscr{U}}(\mathbf{c}) val(\mathbf{x},\mathbf{c}) \right) - \sum_{\mathbf{c}\in\mathscr{U}} \mathrm{P}_{\mathscr{U}}(\mathbf{c}) val^*(\mathbf{c}) \\ &= \min_{\mathbf{x}\in\mathscr{X}} \left(val(\mathbf{x}, \sum_{\mathbf{c}\in\mathscr{U}} \mathrm{P}_{\mathscr{U}}(\mathbf{c})\mathbf{c}) \right) - \sum_{\mathbf{c}\in\mathscr{U}} \mathrm{P}_{\mathscr{U}}(\mathbf{c}) val^*(\mathbf{c})\end{aligned}$$

In the last line, the second term does not depend on $\mathbf{x}$. By definition, the solution that minimizes the first term is $\mathbf{x}^{\tilde{\mathbf{c}}}$. □

Conversely, determining a best response for the $\mathbf{c}$-player against a mixed strategy of the $\mathbf{x}$-player is slightly more involved.

To give the intuition of the way a best $\mathbf{c}$-response (i.e., a worst scenario) to a mixed strategy can be characterized, it is useful to come back to the case of a best $\mathbf{c}$-response to a pure strategy. Given a feasible solution $\mathbf{x}$, the worst scenario $\mathbf{c}^{\mathbf{x}}$ in Observation 3 is the one that *penalizes* most $\mathbf{x}$. Actually, one can consider a dual viewpoint where one *favors* most the solution $\mathbf{y}$ that will induce the max regret for $\mathbf{x}$. This corresponds to scenario $\neg\mathbf{c}^{\mathbf{y}}$, and it is another worst scenario for $\mathbf{x}$. Let us illustrate this point on a simple example.

Example 51. *Consider the very simple minmax regret optimization problem defined by* $n = 5$, $c_1 \in [3,4]$, $c_2 \in [1,5]$, $c_3 \in [1,2]$, $c_4 \in [3,3]$, $c_5 \in [0,6]$ *and* $\mathscr{X} = \{\mathbf{x} \in \mathbb{B}^5 : \sum_{i=1}^5 x_i = 2\}$. *Let* $\mathbf{x}$ *be a feasible solution defined by* $x_2 = x_3 = 1$ *and* $x_i = 0$ *for* $i \notin \{2,3\}$. *The worst scenario* $\mathbf{c}^{\mathbf{x}}$ *as defined in Observation 3 is obtained for* $c_1 = 3$, $c_2 = 5$, $c_3 = 2$, $c_4 = 3$ *and* $c_5 = 0$. *The best feasible solution* $\mathbf{x}^{\mathbf{c}^{\mathbf{x}}}$ *in scenario* $\mathbf{c}^{\mathbf{x}}$ *is defined by* $x_3 = x_5 = 1$ *and* $x_i = 0$ *for* $i \notin \{3,5\}$. *It is then easy to see that the scenario* $\mathbf{c}$ *defined by* $c_i = \underline{c}_i$ *if* $x_i^{\mathbf{c}^{\mathbf{x}}} = 1$ *and* $\overline{c}_i$ *otherwise yields the same regret for* $\mathbf{x}$, *and is therefore also a worst scenario for* $\mathbf{x}$. *In the small numerical example, it corresponds to* $c_1 = 4$, $c_2 = 5$, $c_3 = 1$, $c_4 = 3$ *and* $c_5 = 0$.

Now, this idea can be generalized to characterize a best $\mathbf{c}$-response to a mixed strategy $\mathrm{P}_{\mathscr{X}}$, by looking for a feasible solution $\mathbf{y}$ that maximizes $Reg(\mathrm{P}_{\mathscr{X}}, \mathbf{y})$:

1. identify a specific scenario $\tilde{\mathbf{c}}^{\mathrm{P}_{\mathscr{X}}}$ (to be precisely defined below, and that amounts to $\mathbf{c}^{\mathbf{x}}$ if $\mathrm{P}_{\mathscr{X}}$ chooses $\mathbf{x}$ with probability 1) that takes into account the expected value of x_i $(i = 1,\ldots,n)$ given $\mathrm{P}_{\mathscr{X}}$;

2. compute the best feasible solution $\mathbf{x}^{\tilde{\mathbf{c}}^{\mathrm{P}_{\mathscr{X}}}}$ regarding this scenario; this is the above-mentioned solution $\mathbf{y}$ that will induce the max regret for $\mathrm{P}_{\mathscr{X}}$;

3. the worst scenario $\mathbf{c}^{\mathrm{P}_{\mathscr{X}}}$ for $\mathrm{P}_{\mathscr{X}}$ is defined by $c_i^{\mathrm{P}_{\mathscr{X}}} = \underline{c}_i$ if $x_i^{\tilde{\mathbf{c}}^{\mathrm{P}_{\mathscr{X}}}} = 1$ and $\overline{c}_i$ otherwise; this is indeed a scenario that maximizes the regret of choosing $\mathrm{P}_{\mathscr{X}}$ instead of $\mathbf{x}^{\tilde{\mathbf{c}}^{\mathrm{P}_{\mathscr{X}}}}$.

This principle can be formalized as indicated in Proposition 15 below. Note that this result was first stated by Mastin *et al.* [2015]. For the sake of completeness, we provide a proof of the result.

Proposition 15 (Mastin *et al.* [2015])**.** *Given a probability distribution* $\mathrm{P}_{\mathscr{X}}$, *let* $\mathbf{x}^{\tilde{\mathbf{c}}^{\mathrm{P}_{\mathscr{X}}}}$ *be an optimal solution in scenario* $\tilde{\mathbf{c}}^{\mathrm{P}_{\mathscr{X}}}$ *defined by* $\tilde{c}_i^{\mathrm{P}_{\mathscr{X}}} = \underline{c}_i + (\overline{c}_i - \underline{c}_i)\sum_{\mathbf{x}\in\mathscr{X}} x_i \mathrm{P}_{\mathscr{X}}(\mathbf{x})$. *A best* $\mathbf{c}$*-response to* $\mathrm{P}_{\mathscr{X}}$ *is given by extreme scenario* $\mathbf{c}^{\mathrm{P}_{\mathscr{X}}}$ *where* $\mathbf{c}^{\mathrm{P}_{\mathscr{X}}}$ *is defined by* $c_i^{\mathrm{P}_{\mathscr{X}}} = \underline{c}_i$ *if* $x_i^{\tilde{\mathbf{c}}^{\mathrm{P}_{\mathscr{X}}}} = 1$ *and* $\overline{c}_i$ *otherwise.*

Proof. Instead of looking for an extreme scenario $\mathbf{c}$ that maximizes $Reg(P_{\mathscr{X}}, \mathbf{c})$, we look for a feasible solution $\mathbf{y}$ that maximizes $Reg(P_{\mathscr{X}}, \mathbf{y})$ or equivalently $Reg(P_{\mathscr{X}}, \mathbf{y}, \neg\mathbf{c}^{\mathbf{y}})$ (due to Observation 4).

The following analysis shows how to compute such a solution.

$$\begin{aligned} Reg(P_{\mathscr{X}}, \mathbf{y}, \neg\mathbf{c}^{\mathbf{y}}) &= \sum_{\mathbf{x}\in\mathscr{X}} P_{\mathscr{X}}(\mathbf{x})(\sum_{i=1}^{n}(x_i - y_i)\neg c_i^{\mathbf{y}}) \\ &= \sum_{i=1}^{n}((\sum_{\mathbf{x}\in\mathscr{X}} P_{\mathscr{X}}(\mathbf{x})x_i) - y_i)\neg c_i^{\mathbf{y}} \\ &= \sum_{i:y_i=1}((\sum_{\mathbf{x}\in\mathscr{X}} P_{\mathscr{X}}(\mathbf{x})x_i) - 1)\underline{c}_i \\ &+ \sum_{i:y_i=0}(\sum_{\mathbf{x}\in\mathscr{X}} P_{\mathscr{X}}(\mathbf{x})x_i)\overline{c}_i \end{aligned}$$

Therefore if we denote by D_i the contribution of having $y_i = 1$ instead of 0 (i.e., how having $y_i = 1$ impacts $Reg(P_{\mathscr{X}}, \mathbf{y}, \neg\mathbf{c}^{\mathbf{y}})$) then:

$$\begin{aligned} D_i &= ((\sum_{\mathbf{x}\in\mathscr{X}} P_{\mathscr{X}}(\mathbf{x})x_i) - 1)\underline{c}_i - (\sum_{\mathbf{x}\in\mathscr{X}} P_{\mathscr{X}}(\mathbf{x})x_i)\overline{c}_i \\ &= -(\underline{c}_i + (\overline{c}_i - \underline{c}_i)(\sum_{\mathbf{x}\in\mathscr{X}} P_{\mathscr{X}}(\mathbf{x})x_i)) \end{aligned}$$

Hence:

$$\begin{aligned} Reg(P_{\mathscr{X}}, \mathbf{y}, \neg\mathbf{c}^{\mathbf{y}}) &= \sum_{i:y_i=1}((\sum_{\mathbf{x}\in\mathscr{X}} P_{\mathscr{X}}(\mathbf{x})x_i) - 1)\underline{c}_i \\ &+ \sum_{i:y_i=0}(\sum_{\mathbf{x}\in\mathscr{X}} P_{\mathscr{X}}(\mathbf{x})x_i)\overline{c}_i \\ &= \sum_{i:y_i=1} D_i + \sum_{i=1}^{n}(\sum_{\mathbf{x}\in\mathscr{X}} P_{\mathscr{X}}(\mathbf{x})x_i)\overline{c}_i \end{aligned}$$

As the second term of the sum does not depend on $\mathbf{y}$ our optimization problem amounts to find $\mathbf{y}$ that maximizes $\sum_{i:y_i=1} D_i$ or equivalently that minimizes $\sum_{i:y_i=1} -D_i$ which is exactly a standard optimization problem with costs $-D_i$. □

We will now use the following proposition from Chassein and Goerigk to show how the notion of best response can help us in designing a lower bound on OPT.

Proposition 16 (Chassein and Goerigk [2015])**.** *Let $P_{\mathscr{U}}$ be a mixed strategy of the $\mathbf{c}$-player, we have:*

$$OPT \geq \min_{\mathbf{x}\in\mathscr{X}} Reg(\mathbf{x}, P_{\mathscr{U}}) \tag{6.7}$$

In other words, the expected regret of the $\mathbf{x}$-player when playing a best response to a mixed $\mathbf{c}$-strategy is always a lower bound on OPT. The lower bound we study in this chapter, LB^*, is the best possible lower bound of the type described by proposition 16. More formally :

$$LB^* = \max_{P_{\mathscr{U}}\in\Delta_{\mathscr{U}}} \min_{\mathbf{x}\in\mathscr{X}} Reg(\mathbf{x}, P_{\mathscr{U}}) \tag{6.8}$$

We will soon see how to compute LB^* by using a double oracle approach. Before that, let us see how LB^* relates to other bounds proposed in the literature.

6.3.2 Relation with Chassein and Goerigk's Bound

Let us denote by LB_{CG} the lower bound designed by Chassein and Goerigk. In order to compare LB_{CG} with LB^*, we now provide a game-theoretic interpretation of LB_{CG}.

Definition 43. *A mixed* $\mathbf{c}$*-strategy* $\text{P}_{\mathscr{U}}$ *is said to be* centered *if:*

$$\forall i \in \{1,\dots,n\}, \sum_{\mathbf{c}\in\mathscr{U}} \text{P}_{\mathscr{U}}(\mathbf{c})c_i = \hat{c}_i = \frac{\underline{c}_i + \overline{c}_i}{2} \tag{6.9}$$

where we recall that $\hat{\mathbf{c}}$ is the midpoint scenario defined by $\hat{c}_i = \frac{\underline{c}_i+\overline{c}_i}{2}$, $\forall i \in \{1,\dots,n\}$. Put another way, the previous definition means that, in a centered strategy, the expected cost of each element i is the arithmetic mean of $\underline{c}_i$ and $\overline{c}_i$.

The lower bound given by Chassein and Goerigk [2015] is the best possible lower bound of the type described by proposition 16 when the $\mathbf{c}$-player only considers the restricted set $\hat{\Delta}_{\mathscr{U}}$ of centered mixed strategies with two equally likely extreme scenarios $\mathbf{c}$ and $\neg\mathbf{c}$ as a support (which guarantees that the mixed strategy is centered because $(c_i + \neg c_i)/2 = (\underline{c}_i + \overline{c}_i)/2$). We can now write more formally the lower bound designed by Chassein and Goerigk as:

$$\text{LB}_{\text{CG}} = \max_{\text{P}_{\mathscr{U}}\in\hat{\Delta}_{\mathscr{U}}} \min_{\mathbf{x}\in\mathscr{X}} \text{R}eg(\mathbf{x}, \text{P}_{\mathscr{U}}) \tag{6.10}$$

As $\hat{\Delta}_{\mathscr{U}} \subset \Delta_{\mathscr{U}}$, LB^* will always be at least as accurate as LB_{CG}, i.e., $\text{LB}^* \geq \text{LB}_{\text{CG}}$. Another direct consequence of this observation is that $\text{LB}^* \geq \text{R}eg(\mathbf{x}^{\hat{\mathbf{c}}})/2$ because Chassein and Goerigk proved that:

$$\frac{\text{OPT}}{\text{LB}_{\text{CG}}} \leq \frac{\text{R}eg(\mathbf{x}^{\hat{\mathbf{c}}})}{\text{LB}_{\text{CG}}} \leq 2$$

We now provide a very simple example where $\text{LB}^* > \text{LB}_{\text{CG}}$.

Example 52. *Consider the minmax regret optimization problem defined by* $n = 2$, $c_1 \in [5,10]$, $c_2 \in [7,12]$, *and* $\mathscr{X} = \{\mathbf{x} \in \mathbb{B}^2 : x_1 + x_2 = 1\}$, *where* $c_1x_1 + c_2x_2$ *is the value (to minimize) of a feasible solution* $\mathbf{x}$ *in scenario* $\mathbf{c}$. *There are two feasible solutions* $\mathbf{x}^1 = (1,0)$ *and* $\mathbf{x}^2 = (0,1)$, *and four extreme scenarios* $\mathbf{c}^1 = (5,12)$, $\mathbf{c}^2 = (10,7)$, $\mathbf{c}^3 = (5,7)$ *and* $\mathbf{c}^4 = (10,12)$, *where* $\mathbf{x} = (a,b)$ *(resp.* $\mathbf{c} = (a,b)$*) means that* $x_1 = a, x_2 = b$ *(resp.* $c_1 = a, c_2 = b$*).*

In this problem, set $\hat{\Delta}_{\mathscr{U}}$ *consists of two mixed strategies, namely:*

- *strategy* A*: play* $\mathbf{c}^1 = (5,12)$ *with probability* $1/2$*, and play* $\mathbf{c}^2 = (10,7)$ *with probability* $1/2$*;*
- *strategy* B*: play* $\mathbf{c}^3 = (5,7)$ *with probability* $1/2$*, and play* $\mathbf{c}^4 = (10,12)$ *with probability* $1/2$.

Regardless of which strategy A *or* B *is played by the* $\mathbf{c}$*-player, the best response for the* $\mathbf{x}$*-player is to play* $\mathbf{x}^1 = (1,0)$. *This yields an expected regret of* $(1/2)\times 0 + (1/2)\times 3 = 3/2$ *(resp.* 0*) for the* $\mathbf{x}$*-player if strategy* A *(resp.* B*) is played by the* $\mathbf{c}$*-player. Therefore, for this problem instance,* $\text{LB}_{\text{CG}} = \max\{3/2, 0\} = 3/2$.

Now, for bound LB^**, consider the (non-centered) mixed strategy consisting in playing scenario* $\mathbf{c}^1 = (5,12)$ *with probability 0.3 and scenario* $\mathbf{c}^2 = (10,7)$ *with probability 0.7. This mixed strategy yields an expected regret of* 2.1 *regardless of the fact that the* $\mathbf{x}$*-player plays strategy* $\mathbf{x}^1 = (1,0)$ *(expected regret of* $0.3\times 0 + 0.7\times 3$*) or strategy* $\mathbf{x}^2 = (0,1)$ *(expected regret of* $0.3\times 7 + 0.7\times 0$*). Thus* $\text{LB}^* \geq 2.1 > \text{LB}_{\text{CG}}$.

6.3.3 Relation with Nash Equilibrium

We show here that determining LB^* amounts to identifying a mixed Nash equilibrium of the game induced on feasible solutions and scenarios. First, it is well-known in game theory that $\min_{\mathbf{x}\in\mathcal{X}} Reg(\mathbf{x}, P_{\mathcal{U}}) = \min_{P_{\mathcal{X}}\in\Delta_{\mathcal{X}}} Reg(P_{\mathcal{X}}, P_{\mathcal{U}})$, i.e., there always exists a best response that is a pure strategy (see Proposition 7 on page 82). Therefore our lower bound can be rewritten as:

$$LB^* = \max_{P_{\mathcal{U}}\in\Delta_{\mathcal{U}}} \min_{P_{\mathcal{X}}\in\Delta_{\mathcal{X}}} Reg(P_{\mathcal{X}}, P_{\mathcal{U}})$$

In a zero-sum two-player game, a mixed Nash equilibrium is a couple $(P_{\mathcal{X}}, P_{\mathcal{U}})$ such that:

$$\max_{P_{\mathcal{U}}\in\Delta_{\mathcal{U}}} \min_{P_{\mathcal{X}}\in\Delta_{\mathcal{X}}} Reg(P_{\mathcal{X}}, P_{\mathcal{U}}) = \min_{P_{\mathcal{X}}\in\Delta_{\mathcal{X}}} \max_{P_{\mathcal{U}}\in\Delta_{\mathcal{U}}} Reg(P_{\mathcal{X}}, P_{\mathcal{U}})$$

Thus, the lower bound LB^* corresponds to the value of a mixed Nash Equilibrium (NE). The next theorem states that a mixed Nash equilibrium always exists in the game induced by feasible solutions and scenarios.

Theorem 22. *There exists a (possibly mixed) Nash equilibrium in the zero-sum two-person game induced by the feasible solutions and the possible scenarios.*

Proof. Consider the game where the pure strategies of the **c**-player are restricted to the extreme scenarios. The number of feasible solutions and extreme scenarios is finite. Therefore, the game is finite. In a finite zero-sum two-player game, the von Neumann's minimax theorem insures that there exists a (possibly mixed) Nash equilibrium. It is then easy to realize that this NE is also an NE of the original game. Indeed, as a best **c**-response can always be found as an extreme scenario, the **c**-player will not find a better response in the original game than the one she is playing in the restricted game. □

Determining such a mixed Nash equilibrium can be done by linear programming [Chvatal, 1983] (see subsection 3.1.1 on page 83). Indeed, consider a zero-sum two-person game where player 1 (resp. player 2) has k (resp. l) pure strategies and denote by M_{ij} the payoff of player 1 when strategies i and j are played respectively by player 1 and player 2. Then, a mixed Nash equilibrium of this game can be determined by solving the following Linear Program (LP), where p_i denotes the probability that player 1 plays pure strategy i:

$$\begin{aligned}
&\min_{v,p_1,\dots,p_k} v \\
&v \geq \sum_{i=1}^{k} p_i M_{ij} \qquad \forall j \in \{1,\dots,l\} \qquad (6.11)\\
&\sum_{i=1}^{k} p_i = 1 \\
&p_i \geq 0 \qquad \forall i \in \{1,\dots,k\}
\end{aligned}$$

Constraints (6.11) insure that v is the maximum payoff that player 2 can achieve with her pure strategies (and therefore with *any* strategy, because there always exists a best response that is a pure strategy) if player 1 plays the mixed strategy induced by $p_1,\dots,p_k$. An optimal mixed strategy for player 1 is then given by the optimal values of variables $p_1,\dots,p_k$ (an optimal strategy for player 2 is given by the optimal dual variables) and v gives the value of the game (which is LB^* in our case).

Nevertheless, for the game considered here between the **x**-player and the **c**-player, the complete linear program, that involves as many variables as there are pure strategies for

both players, would be huge, and it is therefore not worth considering its generation in extension. To tackle this issue, Mastin *et al.* [2015] relies on a cutting-plane method.

To give a better insight of their approach it is useful to revise the definition of the set of pure strategies of the **c**-player. Note that, when defining the game between the **x**-player and the **c**-player, the only pure strategies that matter are those that are best responses to some mixed strategy of the adversary. Thus, an important consequence of Observation 4 is that the set of pure strategies of the **c**-player can be restricted to the set $\{\neg\mathbf{c}^{\mathbf{y}} : \mathbf{y} \in \mathscr{X}\}$, i.e., the **c**-player chooses a solution **y** and set $c_i = \underline{c}_i$ if $y_i = 1$, and $c_i = \overline{c}_i$ otherwise[1]. Put another way, the "relevant" pure strategies of the **c**-player can be directly characterized as feasible solutions **y** (each feasible solution **y** corresponds to scenario $\neg\mathbf{c}^{\mathbf{y}}$).

Given a feasible solution $\mathbf{y} \in \mathscr{X}$, the corresponding scenario $\neg\mathbf{c}^{\mathbf{y}}$ is compactly defined by $c_i = \underline{c}_i y_i + \overline{c}_i(1 - y_i)$ for $i \in \{1, \ldots, n\}$. Hence,

$$\begin{aligned} Reg(\mathrm{P}_{\mathscr{X}}, \mathbf{y}) &= \sum_{i=1}^{n} (\underline{c}_i y_i + \overline{c}_i(1 - y_i))\, t_i - \sum_{i=1}^{n} \underline{c}_i y_i \\ &= \sum_{i=1}^{n} \overline{c}_i t_i - \sum_{i=1}^{n} y_i(\underline{c}_i + t_i(\overline{c}_i - \underline{c}_i)) \end{aligned}$$

where $t_i = \sum_{\mathbf{x} \in \mathscr{X}} x_i \mathrm{P}_{\mathscr{X}}(\mathbf{x})$ denotes the probability that element i belongs to the solution that is drawn according to $\mathrm{P}_{\mathscr{X}}(\mathbf{x})$. Note that equation $t_i = \sum_{\mathbf{x} \in \mathscr{X}} x_i \mathrm{P}_{\mathscr{X}}(\mathbf{x})$ defines a mapping from $\Delta_{\mathscr{X}}$ to the convex hull $\mathrm{CH}(\mathscr{X})$ of $\mathscr{X}$, defined by:

$$\mathrm{CH}(\mathscr{X}) = \left\{ \sum_{\mathbf{x} \in \mathscr{X}} p_{\mathbf{x}} \mathbf{x} : p_{\mathbf{x}} \geq 0 \text{ and } \sum_{\mathbf{x} \in \mathscr{X}} p_{\mathbf{x}} = 1 \right\}.$$

Conversely, from any vector **t** in the convex hull of $\mathscr{X}$, it is possible to compute in polynomial time a mixed strategy $\mathrm{P}_{\mathscr{X}}$ such that $\sum_{\mathbf{x} \in \mathscr{X}} x_i \mathrm{P}_{\mathscr{X}}(\mathbf{x}) = t_i$ if the standard problem is of polynomial complexity [Mastin *et al.*, 2015].

This observation led Mastin *et al.* [2015] to consider the following LP for Robust Optimization (RO), where constraints (6.12) are the analog of constraints (6.11) above:

$$\mathscr{P}_{\mathrm{RO}} \begin{cases} \min\limits_{v, t_1, \ldots, t_n} v \\ v \geq \sum\limits_{i=1}^{n} \overline{c}_i t_i - \sum\limits_{i=1}^{n} y_i(\underline{c}_i + t_i(\overline{c}_i - \underline{c}_i)) \qquad \forall \mathbf{y} \in \mathscr{X} \\ \mathbf{t} \in \mathrm{CH}(\mathscr{X}) \end{cases} \tag{6.12}$$

This formulation takes advantage of the fact that if one can optimize over $\mathscr{X}$ in polynomial time, then one can separate over $\mathrm{CH}(\mathscr{X})$ in polynomial time by the polynomial time equivalence of *OPTIMIZATION* and *SEPARATION* (see Theorem 10 on page 89). Furthermore, even if the number of constraints in (6.12) may be exponential as there is one constraint per feasible solution $\mathbf{y} \in \mathscr{X}$, it follows from Proposition 15 that these constraints can be handled efficiently by using a separation oracle computing an optimal solution in scenario defined by $c_i = \underline{c}_i + t_i(\overline{c}_i - \underline{c}_i)$. This separation oracle is also polynomial time if one can optimize over $\mathscr{X}$ in polynomial time. As noted by Mastin *et al.* [2015], it implies that the complexity of determining a mixed NE is polynomial if the standard version of the tackled problem is polynomially solvable.

[1] Actually, it can even be restricted to the set $\{\neg\mathbf{c}^{\mathbf{y}} : \mathbf{y} \in \mathscr{X}$ and **y** optimal for $\neg\mathbf{c}^{\mathbf{y}}\}$.

An equivalent formulation to $\mathscr{P}_{\text{RO}}$, that turned out to be more efficient in practice in our numerical tests, reads as follows, where π is the value of an optimal solution in the scenario defined by $c_i = \underline{c}_i + t_i(\overline{c}_i - \underline{c}_i)$:

$$\mathscr{P}'_{\text{RO}} \begin{cases} \min\limits_{\pi, t_1, \ldots, t_n} \sum\limits_{i=1}^{n} \overline{c}_i t_i - \pi \\ \sum\limits_{i=1}^{n} y_i(\underline{c}_i + t_i(\overline{c}_i - \underline{c}_i)) \geq \pi \qquad \forall \mathbf{y} \in \mathscr{X} \\ \mathbf{t} \in \text{CH}(\mathscr{X}) \end{cases} \tag{6.13}$$

In the next subsection, we adopt a double oracle approach to solve the game induced by the feasible solutions and the possible scenarios. While the method of Mastin *et al.* [2015] dynamically generates the strategies of the **c**-player (strategies $\neg\mathbf{c}^{\mathbf{y}}$ for $\mathbf{y} \in \mathscr{X}$) and uses an implicit representation for the strategies of the **x**-player (a mixed strategy $\text{P}_{\mathscr{X}}$ is induced from $\mathbf{t} \in \text{CH}(\mathscr{X})$), our algorithm dynamically generates both the strategies of the **c**-player and the ones of the **x**-player. As we will see, our alternative method is much more efficient on some classes of instances.

6.3.4 A Double Oracle Approach

The game can be solved by specifying a double oracle algorithm [McMahan *et al.*, 2003] adapted to our problem (this method is detailed in subsection 3.1.3 on page 85). To instantiate this method, a best response oracle must be designed for both players. The best response oracle of the **x**-player (resp. **c**-player) is denoted by $\mathscr{O}_{\mathbf{x}}$ (resp. $\mathscr{O}_{\mathbf{c}}$) and its restricted set of strategies is denoted by $\text{S}_{\mathbf{x}}$ (resp. $\text{S}_{\mathbf{c}}$). Given a mixed strategy $\text{P}_{\mathscr{X}}$ (resp. $\text{P}_{\mathscr{U}}$), $\mathscr{O}_{\mathbf{c}}(\text{P}_{\mathscr{X}})$ (resp. $\mathscr{O}_{\mathbf{x}}(\text{P}_{\mathscr{U}})$) returns a pure strategy **c** (resp. **x**) that maximizes $Reg(\text{P}_{\mathscr{X}}, \mathbf{c})$ (resp. minimizes $Reg(\mathbf{x}, \text{P}_{\mathscr{U}})$). Those two procedures directly follow from Proposition 15 and Observation 5. Each procedure only requires to solve one standard optimization problem. The method is presented in Algorithm 18. One can also notice that at each step, due to Proposition 16, the regret obtained by $\mathbf{x} = \mathscr{O}_{\mathbf{x}}(\text{P}_{\mathscr{U}})$ against the mixed strategy $\text{P}_{\mathscr{U}}$ is always a lower bound on OPT. Therefore we can hope that a good lower bound on OPT can be obtained before convergence and after only a few iterations of the algorithm.

Algorithm 18: Double Oracle Algorithm for Robust Optimization

Data: Feasible solutions $\mathscr{X}$ and possible scenarios $\mathscr{U}$, singletons $\text{S}_{\mathbf{x}} = \{\mathbf{x}\}$ including an arbitrary feasible solution and $\text{S}_{\mathbf{c}} = \{\mathbf{c}\}$ including an arbitrary extreme scenario

Result: a (possibly mixed) NE

converge ← False
while *converge is False* **do**
    Find a Nash equilibrium $(\text{P}_{\mathscr{X}}, \text{P}_{\mathscr{U}})$ of the game $\text{M}(\text{S}_{\mathbf{x}}, \text{S}_{\mathbf{c}})$[2]
    Find $\mathbf{x} = \mathscr{O}_{\mathbf{x}}(\text{P}_{\mathscr{U}})$ and $\mathbf{c} = \mathscr{O}_{\mathbf{c}}(\text{P}_{\mathscr{X}})$
    if $\mathbf{x} \in \text{S}_{\mathbf{x}}$ *and* $\mathbf{c} \in \text{S}_{\mathbf{c}}$ **then** converge ← True
    else add **x** to $\text{S}_{\mathbf{x}}$ and/or **c** to $\text{S}_{\mathbf{c}}$
return $(\text{P}_{\mathscr{X}}, \text{P}_{\mathscr{U}})$

[2] M denotes here the payoff matrix of the two-player zero-sum game involving the **x**-player and the **c**-player.

Note that it is guaranteed that feasible solutions that are not *weak* will never be generated by the double oracle algorithm. A feasible solution is said to be weak if it is optimal for some scenario. Subsequently, an element $i \in \{1,\dots,n\}$ is said to be a weak element if it is part of a weak solution. Because each oracle solves a standard optimization problem for a specific scenario, a feasible solution that is not weak will never be generated. A preprocessing step could therefore be done by removing non-weak elements. Note however that the difficulty of this preprocessing step is highly dependent on the problem under study. For instance, while deciding whether an edge is weak or not can be done in polynomial time for the robust spanning tree problem [Yaman *et al.*, 1999], it has been shown to be NP-complete for the robust shortest path problem [Karasan *et al.*, 2002].

According to the problem instance and the uncertainty set over costs $\mathcal{U}$, the set of generated strategies for both players may either 1) remain relatively small or on the contrary 2) become very large. Therefore, the running time of the double oracle algorithm will be highly dependent on the fact that the problem instance belongs to case 1) or 2). Our intuition is that, for real world problems involving a small number of feasible solutions that deserve to be considered, the problem instance will yield case 1). Secondly, for instances yielding the second situation, we expect the double oracle algorithm to provide a good lower bound on OPT after only a few iterations of the algorithm.

6.4 Adapting the Lower Bound for a Branch and Bound Procedure

We now show how our lower bound can be adapted to be used in a branch and bound procedure for determining a minmax regret (pure) solution. It is well-known that two key ingredients for a successful branch and bound procedure are: i) the accuracy of the lower bound, ii) the efficiency of its computation [Conforti *et al.*, 2014]. In order to speed up the computation of the lower bound at each node, one can take advantage of the information obtained at the father node. This is the topic of this section.

In the branch and bound used in the sequel of the chapter, a node, defining a restricted set $\mathcal{X}' \subseteq \mathcal{X}$ of feasible solutions, is characterized by a couple $(\text{IN}(\mathcal{X}'), \text{OUT}(\mathcal{X}'))$, where $\text{IN}(\mathcal{X}') \subseteq \{i \in \{1,\dots,n\} : x_i = 1, \forall \mathbf{x} \in \mathcal{X}'\}$ is the set of all elements that are enforced to be part of every feasible solution in $\mathcal{X}'$, and $\text{OUT}(\mathcal{X}') \subseteq \{i \in \{1,\dots,n\} : x_i = 0, \forall \mathbf{x} \in \mathcal{X}'\}$ is the set of all elements that are forbidden in any feasible solution of $\mathcal{X}'$. The branching scheme consists in partitioning $\mathcal{X}'$ into two sets by making mandatory or forbidden an element $k \in \{1,\dots,n\} \setminus (\text{IN}(\mathcal{X}') \cup \text{OUT}(\mathcal{X}'))$:

- the first child is the set $\mathcal{X}'' \subseteq \mathcal{X}'$ characterized by $\text{IN}(\mathcal{X}'') = \text{IN}(\mathcal{X}') \cup \{k\}$ and $\text{OUT}(\mathcal{X}'') = \text{OUT}(\mathcal{X}')$,
- the second child is the set $\mathcal{X}''' \subseteq \mathcal{X}'$ characterized by $\text{IN}(\mathcal{X}''') = \text{IN}(\mathcal{X}')$ and $\text{OUT}(\mathcal{X}''') = \text{OUT}(\mathcal{X}') \cup \{k\}$.

At a node $\mathcal{X}'$ of the branch and bound tree, the computation of the lower bound amounts to determine:

$$\max_{\text{P}_{\mathcal{U}} \in \Delta_{\mathcal{U}}} \min_{\mathbf{x} \in \mathcal{X}'} Reg(\mathbf{x}, \text{P}_{\mathcal{U}}) \tag{6.14}$$

The double oracle approach will be unchanged except that given a probability distribution $\text{P}_{\mathcal{U}}$ over extreme scenarios, the best response procedure of the $\mathbf{x}$-player will now return the best possible response in $\mathcal{X}'$. The double oracle algorithm will hence generate a restricted game $\text{M}_{\mathcal{X}'} = \text{M}(\text{S}_{\mathbf{x}}, \text{S}_{\mathbf{c}})$ where all strategies in $\text{S}_{\mathbf{x}}$ are in $\mathcal{X}'$.

Two things can be noted to speed up the computation of the lower bound at nodes $\mathscr{X}''$ and $\mathscr{X}'''$. To initialize the set of feasible solutions in $M_{\mathscr{X}''}$ and $M_{\mathscr{X}'''}$, we partition the set $S_{\mathbf{x}}$ of generated solutions in $M_{\mathscr{X}'}$ into two sets $S_{\mathbf{x}}^k = \{\mathbf{x} \in S_{\mathbf{x}} : x_k = 1\}$ and $S_{\mathbf{x}}^{\overline{k}} = \{\mathbf{x} \in S_{\mathbf{x}} : x_k = 0\}$. The former is used as initial set in $M_{\mathscr{X}''}$ and the latter in $M_{\mathscr{X}'''}$. The handling of set $S_{\mathbf{c}}$ differs. The set of generated scenarios is indeed not dependent on the explored search node as the branching scheme does not add constraints on the scenarios. Therefore at each new node the current set $S_{\mathbf{c}}$ of all scenarios generated so far in the course of the branch and bound is the new initial set of scenarios in Algorithm 18.

6.5 Application to the Robust Shortest Path Problem

In this section, we illustrate how this work can be specified for the robust shortest path problem [Karasan *et al.*, 2002; Montemanni and Gambardella, 2004, 2005; Montemanni *et al.*, 2004], regarding both the computation of LB* and the determination of a minmax regret path. For comparison, we describe some other approaches in the literature that have been used for these problems (computation of LB* and determination of a minmax regret path).

6.5.1 Problem Description

In the robust shortest path problem, a directed graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ is given where $\mathscr{V}$ is a set of vertices, numbered from 1 to $|\mathscr{V}|$, and $\mathscr{E}$ is a set of edges. A starting vertex $s \in \mathscr{V}$ and a destination vertex $t \in \mathscr{V}$ are given and an interval $[\underline{c}_{ij}, \overline{c}_{ij}] \subset \mathbb{R}^+$ is associated to each edge $(i, j) \in \mathscr{E}$, where $\underline{c}_{ij}$ (resp. $\overline{c}_{ij}$) is a lower (resp. upper) bound on the cost induced by edge (i, j). The set $\mathscr{U}$ of possible scenarios is then defined as the Cartesian product of these intervals: $\mathscr{U} = \times_{(i,j)\in\mathscr{E}}[\underline{c}_{ij}, \overline{c}_{ij}]$. Without loss of generality, we assume that $s = 1$ and $t = |\mathscr{V}|$. The set of all paths from s to t is denoted by $\mathscr{X}$. An example of a graph with interval data is shown in Figure 6.1. In the robust shortest path problem we consider here, one wishes to determine a path $\mathbf{x} \in \mathscr{X}$ that minimizes $\max_{\mathbf{c}\in\mathscr{U}} val(\mathbf{x}, \mathbf{c}) - val^*(\mathbf{c})$, where $val(\mathbf{x}, \mathbf{c})$ is the value of path $\mathbf{x}$ in scenario $\mathbf{c}$ and $val^*(\mathbf{c})$ is the value of a shortest path in scenario $\mathbf{c}$.

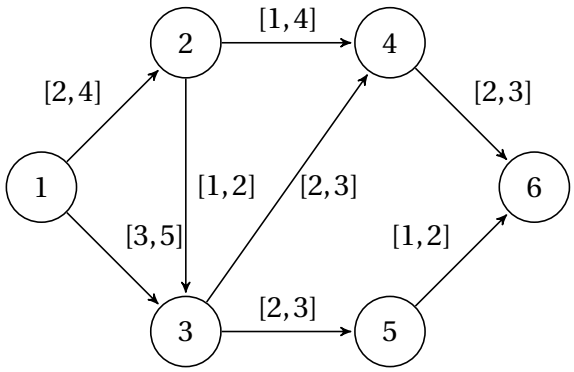


Figure 6.1: An example of a graph with interval data.

6.5.2 Various Approaches for Computing LB*

In the experiments, we will compare the running times of our double oracle method for computing LB* with the following alternative approaches.

LP formulation by Karasan et al.

The Robust Shortest Path problem (RSP) in the interval model was first studied by Karasan *et al.* [2002] who provided the following mixed integer linear program $\mathscr{P}_{\text{KPY}}$ (from now on, we adopt the convention to use the initials of the authors as subscripts):

$$\mathscr{P}_{\text{KPY}}\begin{cases} \min \sum_{(i,j)\in\mathscr{E}} \overline{c}_{ij}x_{ij} - \pi_{|\mathscr{V}|} \\ \text{subject to} \\ \quad \pi_j \le \pi_i + \underline{c}_{ij} + (\overline{c}_{ij} - \underline{c}_{ij})x_{ij},\ \forall (i,j)\in\mathscr{E} \\ \quad b_j = -\sum_{(i,j)\in\mathscr{E}} x_{ij} + \sum_{(j,k)\in\mathscr{E}} x_{jk} \\ \quad \pi_1 = 0 \\ \quad \pi_j \ge 0,\ j\in\{1,2,\ldots,|\mathscr{V}|\} \\ \quad x_{ij}\in\{0,1\} \end{cases}$$

where $b_j = 1$ if $j = 1$, $b_j = -1$ if $j = |\mathscr{V}|$, and $b_j = 0$ otherwise. Variable x_{ij} indicates if edge (i,j) belongs to the minmax regret solution and variable π_i denotes the distance between s and i under the worst possible scenario regarding the solution path characterized by variables x_{ij}. We denote by $\widehat{\mathscr{P}}_{\text{KPY}}$ the relaxed version of this mixed integer linear program where $x_{ij} \ge 0$. One can notice that this relaxed version solves exactly the same problem as our double oracle algorithm:

Proposition 17. *Let* $\widehat{\text{LB}}$ *denote the optimal value of* $\widehat{\mathscr{P}}_{\text{KPY}}$*, then* $\widehat{\text{LB}} = \text{LB}^*$.

Proof. Note that $\widehat{\mathscr{P}}_{\text{KPY}}$ turns out to be a network flow problem, where x_{ij} is the flow on edge (i,j).

We first prove that $\widehat{\text{LB}} \le \text{LB}^*$. Consider the mixed strategy of the $\mathbf{x}$-player (that plays over paths) which realizes regret LB^*. Let us denote by $p_1,\ldots,p_q$ the corresponding probability distribution over paths $\text{P}_1,\ldots,\text{P}_q$ in $\mathscr{X}$. One can define a flow $\mathbf{x}$ on $\mathscr{G}$ by setting $x_{ij} = \sum_{\text{P}_k:(i,j)\in\text{P}_k} p_k$. This flow is feasible for $\widehat{\mathscr{P}}_{\text{KPY}}$. The value of the objective function for this flow corresponds then exactly to LB^*, because:

$$\begin{aligned} \sum_{(i,j)\in\mathscr{E}} \overline{c}_{ij}x_{ij} - \pi_{|\mathscr{V}|} &= \max_{\mathbf{y}\in\mathscr{X}} \sum_{(i,j)\in\mathscr{E}} \overline{c}_{ij}x_{ij} - \sum_{(i,j)\in\mathscr{E}} (\underline{c}_{ij} + (\overline{c}_{ij} - \underline{c}_{ij})x_{ij})y_{ij} \\ &= \max_{\mathbf{y}\in\mathscr{X}} \sum_{(i,j)\in\mathscr{E}} x_{ij}(\overline{c}_{ij}(1-y_{ij}) + \underline{c}_{ij}y_{ij}) - \sum_{(i,j)\in\mathscr{E}} \underline{c}_{ij}y_{ij} \\ &= \max_{\mathbf{y}\in\mathscr{X}} \sum_{(i,j)\in\mathscr{E}} (x_{ij} - y_{ij})\neg c^{\mathbf{y}}_{ij} = \max_{\mathbf{y}\in\mathscr{X}} \sum_{k=1}^{q} p_k \text{Reg}(\text{P}_k, \neg\mathbf{c}^{\mathbf{y}}) \end{aligned}$$

where $\text{Reg}(\text{P}_k, \neg\mathbf{c}^{\mathbf{y}})$ is the regret induced by path P_k in scenario $\neg\mathbf{c}^{\mathbf{y}}$.

We now prove $\text{LB}^* \le \widehat{\text{LB}}$. The *flow decomposition theorem* [Ahuja *et al.*, 1993] states that any feasible flow $\mathbf{x}$ on graph $\mathscr{G}$ can be decomposed into at most $|\mathscr{E}|$ paths in $\mathscr{X}$ and cycles. In our case, it is easy to see that there always exists an optimal flow $\mathbf{x}^*$ for $\widehat{\mathscr{P}}_{\text{KPY}}$ that does not include cycles. Such an optimal flow can therefore be decomposed into at most $|\mathscr{E}|$ paths in $\mathscr{X}$, denoted by $\text{P}_1,\ldots,\text{P}_q$ in the following. Let us consider the mixed strategy that plays each path P_k ($k\in\{1,\ldots,q\}$) with a probability p_k equal to the flow on the path. The regret of this mixed strategy is exactly $\widehat{\text{LB}}$, which can be shown by reversing the above sequence of equalities. □

Note that this connection between the game-theoretic view and the relaxed versions of Mixed Integer Linear Programming (MILP) formulations of minmax regret problems apply to other robust combinatorial optimization problems. For instance, this connection is also valid for the MILP developed by Yaman *et al.* [1999] to solve the minmax regret spanning tree problem. Actually, Mastin *et al.* [2015] proved that, for any robust combinatorial optimization problem, the value of a minmax regret mixed strategy corresponds to the optimal value of linear program $\mathscr{P}'_{\text{RO}}$ given at the end of Section 6.3 (that may involve an exponential number of constraints).

LP formulation by Mastin et al. and a variant

We present below the linear program $\mathscr{P}_{\text{MJC}}$ obtained by specifying $\mathscr{P}'_{\text{RO}}$ to the case of RSP[3]. Note that this program returns for each edge (i,j) a probability x_{ij}, from which a probability distribution over paths in $\mathscr{X}$ can be inferred in polynomial time.

$$\mathscr{P}_{\text{MJC}} \begin{cases} \min\limits_{\pi, x_{ij}:(i,j)\in\mathscr{E}} \sum\limits_{(i,j)\in\mathscr{E}} \overline{c}_{ij} x_{ij} - \pi \\ \text{subject to} \\ \quad \pi \leq \sum\limits_{(i,j)\in\mathscr{E}} y_{ij}(\underline{c}_{ij} + x_{ij}(\overline{c}_{ij} - \underline{c}_{ij})) \qquad \forall \mathbf{y} \in \mathscr{X} \qquad (6.15) \\ \quad b_j = -\sum\limits_{(i,j)\in\mathscr{E}} x_{ij} + \sum\limits_{(j,k)\in\mathscr{E}} x_{jk} \qquad (6.16) \\ \quad x_{ij} \geq 0 \qquad \forall (i,j) \in \mathscr{E} \end{cases}$$

Constraints (6.15) are the specification of constraints (6.13) in the case of RSP, and therefore insure, at optimum, that $\sum_{(i,j)\in\mathscr{E}} \overline{c}_{ij} x_{ij} - \pi$ is the value of a min max regret mixed strategy. Furthermore, flow constraints (6.16) make sure that variables x_{ij} are consistent, i.e., they induce a probability distribution over paths in $\mathscr{X}$. It is thus possible to compute LB^* by solving this linear program with a cutting-plane algorithm.

In the experiments we also tested the linear program $\mathscr{D}_{\text{MJC}}$ where one adopts the dual viewpoint (viewpoint of the **c**-player that aims to maximize the regret)[4]:

$$\mathscr{D}_{\text{MJC}} \begin{cases} \max\limits_{\mu, y_{ij}:(i,j)\in\mathscr{E}} \mu - \sum\limits_{(i,j)\in\mathscr{E}} \underline{c}_{ij} y_{ij} \\ \text{subject to} \\ \quad b_j = -\sum\limits_{(i,j)\in\mathscr{E}} y_{ij} + \sum\limits_{(j,k)\in\mathscr{E}} y_{jk} \\ \quad \mu \leq \sum\limits_{(i,j)\in\mathscr{E}} x_{ij}(\underline{c}_{ij} y_{ij} + (1 - y_{ij})\overline{c}_{ij}) \qquad \forall \mathbf{x} \in \mathscr{X} \qquad (6.17) \\ \quad y_{ij} \geq 0 \qquad \forall (i,j) \in \mathscr{E} \end{cases}$$

Flow variables y_{ij} induce a mixed strategy $\text{P}_{\mathscr{X}}$ over paths: the flow can indeed be decomposed into paths by the flow decomposition theorem (each cycle could be removed), and each path $\mathbf{z} \in \mathscr{X}$ among them has a probability $\text{P}_{\mathscr{X}}(\mathbf{z})$ to be picked equal to the flow on $\mathbf{z}$. Constraints (6.17) ensure that μ is the minimum expected cost for the **x**-player given the mixed strategy of the **c**-player defined by variables y_{ij} because:

$$\text{Reg}(\mathbf{x}, \text{P}_{\mathscr{X}}) = \sum_{(i,j)\in\mathscr{E}} \Big(x_{ij}(\underline{c}_{ij} y_{ij} + (1 - y_{ij})\overline{c}_{ij}) \Big) - \sum_{(i,j)\in\mathscr{E}} \underline{c}_{ij} y_{ij}$$

[3] Note that the obtained linear program is close to the Benders reformulation of $\mathscr{P}_{\text{KPY}}$ proposed by Montemanni and Gambardella [2005].

[4] Even if it is a dual viewpoint, note that $\mathscr{D}_{\text{MJC}}$ is not the dual program of $\mathscr{P}_{\text{MJC}}$.

for all $\mathbf{x} \in \mathscr{X}$ and $\mathbf{y} \in \mathrm{CH}(\mathscr{X})$. Let us now explain this equation. After committing to $\mathrm{P}_{\mathscr{X}}$, the $\mathbf{c}$-player will draw a path including edge (i,j) with probability y_{ij}. If the drawn path contains (i,j) then cost c_{ij} is set to $\underline{c}_{ij}$. Thus, the expected cost of edge (i,j) for the $\mathbf{c}$-player is $\underline{c}_{ij} y_{ij}$. If the drawn path does not contain (i,j) then cost c_{ij} is set to $\overline{c}_{ij}$. Consequently, if the $\mathbf{x}$-player chooses path $\mathbf{x}$, the expected cost of edge (i,j) is $x_{ij}(\underline{c}_{ij} y_{ij} + (1 - y_{ij})\overline{c}_{ij})$. The equation then follows from linearity of expectation with respect to edge costs.

This analysis shows that $\mathscr{D}_{\mathrm{MJC}}$ computes $\max_{\mathrm{P}_{\mathscr{X}} \in \Delta_{\mathscr{X}}} \min_{\mathbf{x} \in \mathscr{X}} Reg(\mathbf{x}, \mathrm{P}_{\mathscr{X}})$ where $\mathrm{P}_{\mathscr{X}}$ induces the mixed strategy $\mathrm{P}_{\mathscr{U}} \in \Delta_{\mathscr{U}}$ defined by $\mathrm{P}_{\mathscr{U}}(\neg \mathbf{c}^{\mathbf{y}}) = \mathrm{P}_{\mathscr{X}}(\mathbf{y})$. We recall that $\mathscr{P}_{\mathrm{MJC}}$ computes $\min_{\mathrm{P}_{\mathscr{X}} \in \Delta_{\mathscr{X}}} \max_{\mathbf{y} \in \mathscr{X}} Reg(\mathrm{P}_{\mathscr{X}}, \mathbf{y})$ where $\mathbf{y}$ induces the pure strategy $\neg \mathbf{c}^{\mathbf{y}} \in \mathscr{U}$. By the minimax theorem, the optimal values of $\mathscr{P}_{\mathrm{MJC}}$ and $\mathscr{D}_{\mathrm{MJC}}$ therefore coincide.

Note that there are as many constraints in $\mathscr{P}_{\mathrm{MJC}}$ and $\mathscr{D}_{\mathrm{MJC}}$ as there are paths in $\mathscr{X}$. We now detail how to overcome this difficulty with a cutting plane method (cutting plane methods are further detailed in subsection 3.1.4 on page 87). Given a subset $\mathscr{X}' \subseteq \mathscr{X}$ of paths, let us denote by $\mathscr{P}_{\mathrm{MJC}}(\mathscr{X}')$ (resp. $\mathscr{D}_{\mathrm{MJC}}(\mathscr{X}')$) the linear program where one considers only constraints induced by paths in $\mathscr{X}'$. Starting from a small subset $\mathscr{X}'$ of paths, one solves $\mathscr{P}_{\mathrm{MJC}}(\mathscr{X}')$ (resp. $\mathscr{D}_{\mathrm{MJC}}(\mathscr{X}')$). There are two possibilities: either the found optimal solution is feasible for $\mathscr{P}_{\mathrm{MJC}}$ (resp. $\mathscr{D}_{\mathrm{MJC}}$), in which case it is optimal for $\mathscr{P}_{\mathrm{MJC}}$ (resp. $\mathscr{D}_{\mathrm{MJC}}$), or it violates at least one constraint in $\mathscr{P}_{\mathrm{MJC}}$ (resp. $\mathscr{D}_{\mathrm{MJC}}$). To identify the current situation, we solve the shortest path problem with cost function $c_{ij} = \underline{c}_{ij} + x_{ij}(\overline{c}_{ij} - \underline{c}_{ij})$ (resp. $c_{ij} = \underline{c}_{ij} y_{ij} + (1 - y_{ij})\overline{c}_{ij}$). This constitutes our separation oracle. If the cost of that path is strictly smaller than π (resp. μ), then it means that at least one constraint is violated, and any constraint associated to a shortest path is a most violated one. In this case, we add to $\mathscr{X}'$ the obtained shortest path and we reoptimize. If the regret induced by a shortest path is greater than or equal to π (resp. μ), then it means that no constraint is violated in $\mathscr{P}_{\mathrm{MJC}}$ (resp. $\mathscr{D}_{\mathrm{MJC}}$), and therefore we have solved $\mathscr{P}_{\mathrm{MJC}}$ (resp. $\mathscr{D}_{\mathrm{MJC}}$).

Our approach

In order to specify Algorithm 18 for the robust shortest path problem, one needs to specify the two best response oracles $\mathscr{O}_{\mathbf{x}}(\mathrm{P}_{\mathscr{U}})$ (best response of the $\mathbf{x}$-player to a mixed strategy $\mathrm{P}_{\mathscr{U}}$ of the $\mathbf{c}$-player) and $\mathscr{O}_{\mathbf{c}}(\mathrm{P}_{\mathscr{X}})$ (best response of the $\mathbf{c}$-player to a mixed strategy $\mathrm{P}_{\mathscr{X}}$ of the $\mathbf{x}$-player). These two oracles can be implemented by using Dijkstra's algorithm.

- For $\mathscr{O}_{\mathbf{x}}(\mathrm{P}_{\mathscr{U}})$ the cost of (i,j) is defined as $\sum_{\mathbf{c} \in \mathscr{U}} \mathrm{P}_{\mathscr{U}}(\mathbf{c}) c_{ij}$. Any optimal path found by running Dijkstra's algorithm is then a best response (by Observation 5).

- For $\mathscr{O}_{\mathbf{c}}(\mathrm{P}_{\mathscr{X}})$ the cost of (i,j) is defined as $\underline{c}_{ij} + \sum_{\mathbf{x} \in \mathscr{X}} \mathrm{P}_{\mathscr{X}}(\mathbf{x}) x_{ij}(\overline{c}_{ij} - \underline{c}_{ij})$. For any optimal path $\mathbf{z}$ found by running Dijkstra's algorithm, $\neg \mathbf{c}^{\mathbf{z}}$ is then a best response (by Proposition 15).

As the robust shortest path problem is defined as a single-source single-target problem, we implemented a bidirectional variant of Dijkstra's algorithm [Pohl, 1969] that make it possible to find an optimal path by only exploring a small part of the graph. Consequently, the double oracle algorithm has the advantage over the linear programming formulations that it does not need an explicit representation of graph $\mathscr{G}$.

For this reason, and as it will be shown by the numerical tests, depending on the structure of graph $\mathscr{G}$, running the double oracle algorithm is much faster than solving $\widehat{\mathscr{P}}_{\mathrm{KPY}}$ by linear programming. Regarding the comparison with the solution of $\mathscr{P}_{\mathrm{MJC}}$ and $\mathscr{D}_{\mathrm{MJC}}$, both the double oracle algorithm and the cutting plane algorithms make calls to the oracles defined above. These three methods are betting on a low number of "relevant" paths in the

problem so that it could converge in few iterations. However, while the double oracle uses both oracles $\mathscr{O}_{\mathbf{x}}(\mathrm{P}_{\mathscr{U}})$ and $\mathscr{O}_{\mathbf{c}}(\mathrm{P}_{\mathscr{X}})$, the cutting plane algorithm for solving $\mathscr{P}_{\mathrm{MJC}}$ (resp. $\mathscr{D}_{\mathrm{MJC}}$) only uses $\mathscr{O}_{\mathbf{c}}(\mathrm{P}_{\mathscr{X}})$ (resp. $\mathscr{O}_{\mathbf{x}}(\mathrm{P}_{\mathscr{U}})$) and an implicit representation of *all* strategies of the **x**-player (resp. **c**-player) through flow constraints on variables x_{ij} (resp. y_{ij}), one variable per edge (i,j) in $\mathscr{G}$. The sequence of LPs that must be solved can therefore be costly for large graphs, especially the first LP in the sequence (the other LPs are computed much faster thanks to reoptimization).

6.5.3 Branch and Bound Algorithm for the Minmax Regret Shortest Path Problem

In the experiments, we will compare the performances of the branch and bound procedure for determining a minmax regret path developed in this chapter with other branch and bounds proposed in the literature.

Branch and bound by Montemanni et al.

Montemanni *et al.* [2004] proposed a branch and bound algorithm for the minmax regret shortest path problem. The branching strategy generates a search tree where each node corresponds to a subset $\mathscr{X}'$ of feasible solutions characterized by a subset $\mathrm{IN}(\mathscr{X}')$ of mandatory edges and a subset $\mathrm{OUT}(\mathscr{X}')$ of forbidden edges. Note that the branching is performed in a way insuring that: $\mathrm{IN}(\mathscr{X}')$ always contains a chain of connected edges which form a subpath starting from s. The branching strategy is illustrated in Figure 6.2 on page 207. For a detailed presentation of the branching strategy, the reader is referred to the article by Montemanni *et al.* [2004]. The authors use the following lower bound $\mathrm{LB}_{\mathrm{MGD}}(\mathscr{X}')$ on the max regret of a subset $\mathscr{X}'$ of feasible solutions defined by $\mathrm{IN}(\mathscr{X}')$ and $\mathrm{OUT}(\mathscr{X}')$:

$$\mathrm{LB}_{\mathrm{MGD}}(\mathscr{X}') = \mathrm{SP}(\mathbf{c}^{\mathscr{E}}, \mathrm{IN}(\mathscr{X}'), \mathrm{OUT}(\mathscr{X}')) - \mathrm{SP}(\mathbf{c}^{\mathscr{E}\setminus\mathrm{OUT}(\mathscr{X}')}, \emptyset, \emptyset) \tag{6.18}$$

where $\mathrm{SP}(\mathbf{c}, \mathrm{A}, \mathrm{B})$ is the value of the shortest $s-t$ path including all edges from A and excluding all edges from B in scenario $\mathbf{c}$, and $\mathbf{c}^{\mathrm{A}}$ is the scenario where all the edges in A are set to their upper bound and all others are set to their lower bound. The computation of $\mathrm{LB}_{\mathrm{MGD}}(\mathscr{X}')$ requires two runs of Dijkstra's algorithm: one run for computing $\mathrm{SP}(\mathbf{c}^{\mathscr{E}}, \mathrm{IN}(\mathscr{X}'), \mathrm{OUT}(\mathscr{X}'))$ and one run for computing $\mathrm{SP}(\mathbf{c}^{\mathscr{E}\setminus\mathrm{OUT}(\mathscr{X}')}, \emptyset, \emptyset)$.

Branch and bound by Chassein and Goerigk

Chassein and Goerigk [2015] used the same branching strategy as Montemanni et al., but changed the lower bounding procedure as detailed in section 6.3.2. When focusing on a subset $\mathscr{X}'$ of feasible solutions, we denote by $\mathrm{LB}_{\mathrm{CG}}(\mathscr{X}')$ the lower bound adapted from $\mathrm{LB}_{\mathrm{CG}}$. In order to define $\mathrm{LB}_{\mathrm{CG}}(\mathscr{X}')$, the authors consider the set $\hat{\Delta}_{\mathscr{U}}(\mathscr{X})$ of mixed strategies with two equally likely extreme scenarios $\mathbf{c}^1$ and $\mathbf{c}^2$ such that $c^1_{ij} = c^2_{ij} = \overline{c}_{ij}$ (resp. $\underline{c}_{ij}$) if $(i,j) \in \mathrm{IN}(\mathscr{X}')$ (resp. $\mathrm{OUT}(\mathscr{X}')$) and $c^1_{ij} + c^2_{ij} = \underline{c}_{ij} + \overline{c}_{ij}$ if $(i,j) \notin \mathrm{IN}(\mathscr{X}') \cup \mathrm{OUT}(\mathscr{X}')$. The lower bound $\mathrm{LB}_{\mathrm{CG}}(\mathscr{X}')$ then reads as follows:

$$\max\left\{ \sum_{(i,j)\in\mathrm{IN}(\mathscr{X}')} \overline{c}_{ij} - \hat{c}_{ij} + \mathrm{SP}(\hat{\mathbf{c}}, \mathrm{IN}(\mathscr{X}'), \mathrm{OUT}(\mathscr{X}')) - \mathbb{E}_{\mathrm{P}_{\mathscr{U}}}(val^*(\mathrm{C})) : \mathrm{P}_{\mathscr{U}} \in \hat{\Delta}_{\mathscr{U}}(\mathscr{X}) \right\}$$

where $\hat{c}_{ij} = (\underline{c}_{ij} + \overline{c}_{ij})/2$, C is a random variable on $\mathscr{U}$, and $\mathbb{E}_{\mathrm{P}}(\mathrm{X})$ denotes the expectancy of random variable X given the probability distribution P on the domain of X. The maximization operation is performed by resorting to Suurballe's algorithm [Suurballe and Tarjan,

1984]. Overall, the computation of $\mathrm{LB}_{\mathrm{CG}}(\mathscr{X}')$ requires three runs of Dijkstra's algorithm: one run for computing $\mathrm{SP}(\hat{\mathbf{c}}, \mathrm{IN}(\mathscr{X}'), \mathrm{OUT}(\mathscr{X}'))$ and two runs involved in Suurballe's algorithm.

Our approach

Our branch and bound also uses the branching strategy proposed by Montemanni et al., but relies on the double oracle algorithm for the lower bounding procedure. In the double oracle algorithm, the best responses are computed by resorting to Dijkstra's algorithm both for the **x**-player and the **c**-player:

- **x**-player: the best response procedure uses Dijkstra's algorithm to find the best $s_{in} - t$ path according to the scenario as defined in Observation 5, where s_{in} is the last node on the subpath defined by $\mathrm{IN}(\mathscr{X}')$ and where all edges in $\mathrm{OUT}(\mathscr{X}')$ have costs set to infinity. The subpath defined by $\mathrm{IN}(\mathscr{X}')$ and this path are then concatenated to obtain the best **x**-response.

- **c**-player: the best response procedure uses Dijkstra's algorithm to find the best $s - t$ path according to the scenario defined in Proposition 15. The best **c**-response is then obtained by setting the costs of edges along this path to their lower bound, and the costs of other edges to their upper bound.

As all the best responses already generated are transmitted from a search tree node to its children (see the end of Section 6.4), the computation of the lower bound speeds up with the depth of the considered node in the branch and bound procedure.

6.5.4 Numerical Tests

Our approach is evaluated using two different experiments. In a first experiment, we compare the accuracy and computation time of different lower bounds on the minmax regret. In a second experiment, we test how a branch and bound procedure using the lower bound investigated in this chapter compares with the previous results by Chassein and Goerigk [2015] and Montemanni *et al.* [2004].

Configuration and implementation details. All times are wall-clock times on a 2.4GHz Intel Core i5 machine with 8GB main memory. Our implementation is in C++, with external calls to GUROBI version 5.6.3 when (possibly mixed) linear programs need to be solved. Lastly, all single-source single-target Dijkstra algorithms are implemented in a bidirectional fashion and the lower bound designed by Chassein and Goerigk is computed using Suurballe's algorithm [Suurballe and Tarjan, 1984]. All results are averaged over 100 runs.

Description of the instances. Chassein and Goerigk [2015] considered two different types of randomly generated graph classes. We consider the same graph classes and respect the same experimental protocol.

The generation of the costs is parametrized by two parameters, namely r and d. For an edge (i, j), a random number m is first sampled from the interval $[1, r]$. The lower bound cost $\underline{c}_{ij}$ is sampled uniformly from $[(1-d)m, (1+d)m]$ and the upper bound cost $\overline{c}_{ij}$ is then chosen uniformly from interval $[\underline{c}_{ij}, (1+d)m]$. Thus, parameter d enables to control the cost variability (the cost variability increases with d).

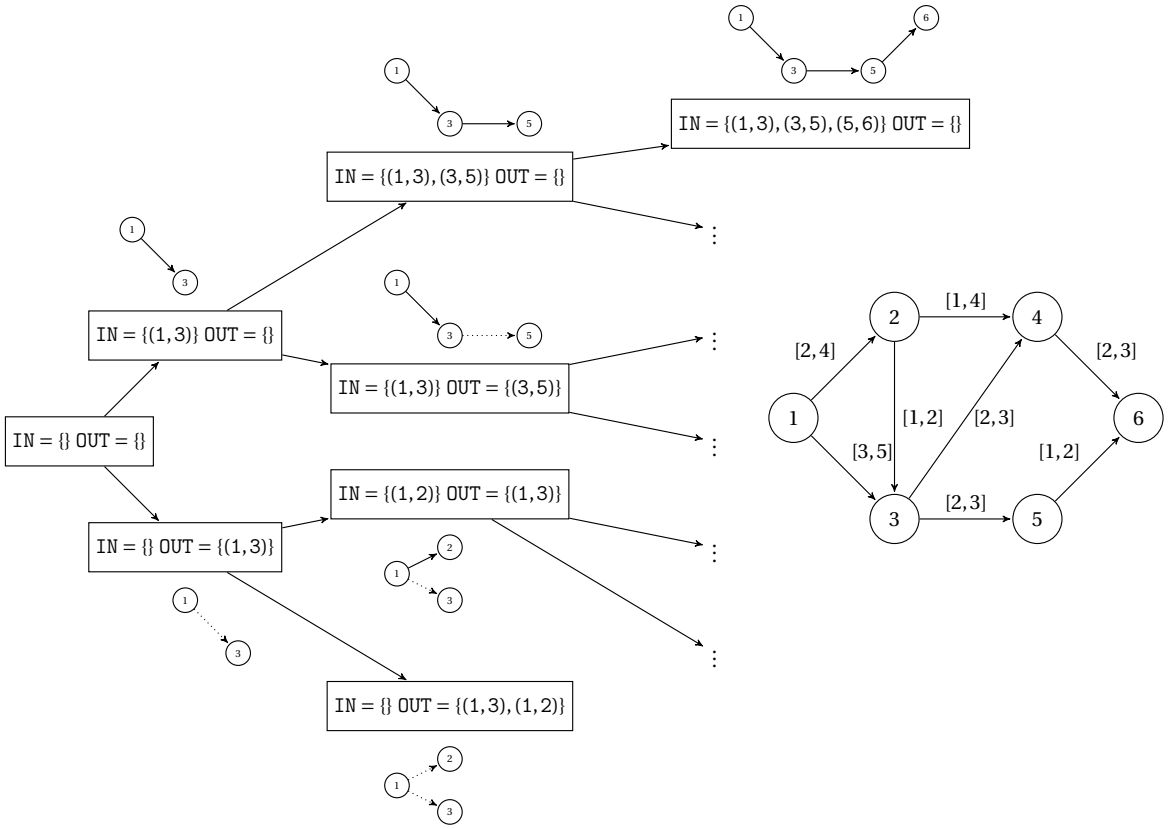


Figure 6.2: Beginning of the entire search tree in the branch and bound procedure for the graph represented in Figure 6.1 (reproduced on the right side of the figure). Each search node is defined by a set IN of mandatory edges (solid edges) and a set OUT of forbidden edges (dotted edges).

The graph family R-n-r-d-δ consists of randomly generated graphs such that each graph has n nodes and an approximate edge density of δ (i.e., the probability that an edge exists is δ). The starting vertex s is the first generated node of the graph and the destination vertex t is the last generated node of the graph.

The second graph family K-n-r-d-w consists of layered graphs. Every layer is completely connected to the next layer. The starting node s is connected to the first layer and the last layer is connected to the destination node t. The overall graph consists of n nodes where every layer contains w nodes.

First Experiment. In this experiment, we compare the computation time and accuracy of several lower bounds on the minmax regret value:

- We will use LB^*_n to denote the lower bound obtained when the double oracle algorithm is stopped after n iterations regardless of the fact that convergence has been attained or not. The returned lower bound is then $Reg(\mathscr{O}_{\mathbf{x}}(\text{P}_{\mathscr{U}}), \text{P}_{\mathscr{U}})$, where $\mathscr{O}_{\mathbf{x}}(\text{P}_{\mathscr{U}})$ is the last best $\mathbf{x}$-response generated.
- The lower bound studied in this chapter is denoted by LB^*. We may also use the notations LB^*_{DO}, $\text{LB}^*_{\widehat{\mathscr{P}}_{\text{KPY}}}$, $\text{LB}^*_{\mathscr{P}_{\text{MJC}}}$ or $\text{LB}^*_{\mathscr{D}_{\text{MJC}}}$ according to the method used to compute LB^*:
  - Notation LB^*_{DO} means that LB^* is computed with the double oracle algorithm ran until convergence.
  - Notation $\text{LB}^*_{\widehat{\mathscr{P}}_{\text{KPY}}}$ means that LB^* is computed using linear program $\widehat{\mathscr{P}}_{\text{KPY}}$.

  - Notation $LB^*_{\mathscr{P}_{MJC}}$ means that LB^* is computed using the cutting plane method to solve $\mathscr{P}_{MJC}$.
  - Notation $LB^*_{\mathscr{D}_{MJC}}$ means that LB^* is computed using the cutting plane method to solve $\mathscr{D}_{MJC}$.

- The lower bound designed by Chassein and Goerigk is denoted by LB_{CG}.

- We recall that, as shown by Kasperski and Zielinski [2006], the regret of the midpoint solution is not more than $2 \cdot OPT$. Thus, the regret of the midpoint solution divided by 2 is a lower bound on OPT, which we denote by LB_{KZ}.

To compare the accuracy of these lower bounds, the following ratios are computed. The closer they are to 1, the more accurate are the lower bounds.

- We denote by Gap-medSol the approximation guarantee obtained for the midpoint solution. It corresponds to the regret of the midpoint solution divided by the considered lower bound.

- When considering the double oracle algorithm, we will denote by minSol the element of minimum regret in the set composed of the midpoint solution and of the admissible solutions of the restricted game (i.e., generated by the double oracle algorithm). Gap-minSol represents the obtained approximation guarantee for the minSol solution. It corresponds to the regret of the minSol solution divided by the considered lower bound.

- Lastly, we denote by Gap-Opt the gap between the considered lower bound and the optimal value OPT. It corresponds to OPT divided by the considered lower bound. The value OPT is computed using $\mathscr{P}_{KPY}$.

The results are presented in Tables 1 to 4 for R-graphs and 5 to 8 for K-graphs[5]. Notation "inf" stands for an infinite gap (obtained if the lower bound is 0 with $OPT \neq 0$). The following observations can be made.

For R-graphs, we see that the computation of LB^*_{DO} is much faster than the computation of $LB^*_{\widehat{\mathscr{P}}_{KPY}}$ while it is the opposite in K-graphs. Indeed, intuitively, in R-graphs few paths will be interesting (paths with few edges) and the double oracle algorithm will converge after few iterations while in K-graphs, the number of interesting paths may explode with the size of the graph (all paths have the same number of edges).

For the same reason, the computation of $LB^*_{\mathscr{P}_{MJC}}$ and $LB^*_{\mathscr{D}_{MJC}}$ are more effective than the one of $LB^*_{\widehat{\mathscr{P}}_{KPY}}$ for R-graphs and less effective for K-graphs. The computation times of $LB^*_{\mathscr{P}_{MJC}}$ and $LB^*_{\mathscr{D}_{MJC}}$ are similar on R-graphs but the computation of $LB^*_{\mathscr{P}_{MJC}}$ is much more efficient on K-graphs than the one of $LB^*_{\mathscr{D}_{MJC}}$. While the computation of $LB^*_{\mathscr{P}_{MJC}}$ and $LB^*_{\mathscr{D}_{MJC}}$ are more effective than the computation of LB^*_{DO} on K-graphs, they are clearly outperformed on R-graphs. For those graphs the double oracle algorithm is able to focus on a small portion of the graph while the restricted linear programs in the cutting plane method must take into account the whole graph.

For both types of graphs, LB^* is of course more computationally demanding than LB_{KZ} and LB_{CG} but it is much more accurate and this is often true after only a few iterations of the double oracle algorithm (see LB^*_{15} for instance).

[5]We highlight the fact that Gap-medSol for LB_{KZ} is equal to 1 (and not 2) for instances where the regret of the midpoint solution is 0. Consequently the average Gap-medSol, somewhat counter-intuitively, can be strictly less than 2 in some tables.

Finally, note that Gap-minSol can be slightly better than Gap-medSol, but globally, it does not improve much the approximation guarantee. The reason is that the regret of the midpoint solution is often close to OPT, as confirmed by the numerical data.

We have performed the same experiments on two valued graphs representing real cities (the edge values represent distances):

- a graph representing New York city with 264346 nodes and 733846 edges,
- a graph representing the San Fransisco Bay with 321270 nodes and 800172 edges.

Those graphs are available on the website of the 9^{th} DIMACS implementation challenge on the computation of shortest paths[6]. For these two graphs, for each instance, the starting node and the destination node are randomly drawn according to a uniform distribution. For each cost c_{ij}, the lower bound $\underline{c}_{ij}$ (resp. upper bound $\overline{c}_{ij}$) is randomly drawn according to a uniform distribution in interval $[c_{ij} - c_{ij}/10, c_{ij}]$ (resp. $[c_{ij}, c_{ij} + c_{ij}/10]$). The results are presented in Tables 9 and 10. For those two graphs, the computation of LB^*_{DO} is much faster than: the computation of $\text{LB}^*_{\widehat{\mathscr{P}}_{\text{KPY}}}$ (twice faster for New York and 9 times faster for San Francisco Bay), the computation of $\text{LB}^*_{\mathscr{D}_{\text{MJC}}}$ (5 times faster for New York and 2.5 times faster for San Francisco Bay), and the computation of $\text{LB}^*_{\mathscr{P}_{\text{MJC}}}$ (5 times faster for New York and 3.2 times faster for San Francisco Bay).

Regarding the accuracy of the considered lower bounds, one can observe that the lower bound obtained after 20 iterations of the double oracle algorithm is close to LB^* and provides a much better approximation guarantee than LB_{KZ} or LB_{CG}.

Second Experiment. In a second experiment, we compare the performances of four exact solution approaches: three branch and bound algorithms and the approach consisting in solving $\mathscr{P}_{\text{KPY}}$ (denoted MILP in the following). The three branch and bound algorithms use a best-first exploration strategy of the search tree generated by the branching procedure designed by Montemanni *et al.* [2004]. They only differ by the lower bound used. Algorithm BB_{MGD} uses LB_{MGD}, algorithm BB_{CG} uses LB_{CG} and algorithm BB^* uses LB^*_{DO}.

For small instances of each graph class, we compare the time needed for each algorithm to solve the problem to optimality and the number of nodes expanded in the branch and bound tree. The solution times are measured in milliseconds. The results, presented in Figures 6.3 and 6.4, are averaged over 100 randomly generated graphs. For the computation of LB^*_{DO} the number of paths in the restricted game is upper bounded by 50. Put another way, if 50 paths are generated in the restricted game, then the lower bounding procedure is stopped and the current lower bound is returned (note that the number of iterations is not upper bounded by 50, as the number of generated scenarios is not). Note that the curves are represented in a logarithmic scale on the y-axis for the graphs giving the evolution of the solution time (left side of Figures 6.3 and 6.4).

For R-graphs we see that all three branch and bound algorithms outperform the algorithm solving the MILP, except for BB_{MGD} that performs very badly if the cost variability is too large. This can be seen in Figure 6.2.d) where the number of expanded nodes explodes for high cost variabilities. Except for very low cost variabilities, BB^* performs best on average thanks to the very low number of expanded nodes.

For K-graphs we see that all three branch and bound algorithms are outperformed by the algorithm solving the MILP, except when the layer size is large. Indeed, as already

[6] http://www.dis.uniroma1.it/challenge9/download.shtml

	time for lb (ms)				Gap-medSol				Gap-minSol				Gap-Opt			
	mean	std	min	max	mean	std	min	max	mean	std	min	max	mean	std	min	max
MILP	2.95	3.08	1	23	-	-	-	-	-	-	-	-	-	-	-	-
LB^*_5	2.33	1.75	0	10	1.15	0.22	1.00	1.88	1.14	0.22	1.00	1.88	1.14	0.22	1.00	1.88
LB^*_{10}	2.57	1.92	0	9	1.15	0.22	1.00	1.88	1.14	0.22	1.00	1.88	1.14	0.22	1.00	1.88
LB^*_{15}	2.60	1.89	0	7	1.15	0.22	1.00	1.88	1.14	0.22	1.00	1.88	1.14	0.22	1.00	1.88
LB^*_{20}	2.43	1.83	0	9	1.15	0.22	1.00	1.88	1.14	0.22	1.00	1.88	1.14	0.22	1.00	1.88
LB^*_{DO}	2.51	1.95	0	9	1.15	0.22	1.00	1.88	1.14	0.22	1.00	1.88	1.14	0.22	1.00	1.88
$LB^*_{\widehat{\mathscr{D}}_{KPY}}$	0.61	0.49	0	1	1.15	0.22	1.00	1.88	-	-	-	-	1.14	0.22	1.00	1.88
$LB^*_{\mathscr{D}_{MJC}}$	0.65	0.59	0	3	1.15	0.22	1.00	1.88	-	-	-	-	1.14	0.22	1.00	1.88
$LB^*_{\mathscr{D}_{MJC}}$	0.56	0.52	0	2	1.15	0.22	1.00	1.88	-	-	-	-	1.14	0.22	1.00	1.88
LB_{CG}	0.08	0.27	0	1	1.45	0.49	1.00	2.00	-	-	-	-	1.45	0.49	1.00	2.00
LB_{KZ}	0.06	0.24	0	1	1.47	0.50	1.00	2.00	-	-	-	-	1.46	0.50	1.00	2.00

Table 6.1: R10-1000-0.5-1

	time for lb (ms)				Gap-medSol				Gap-minSol				Gap-Opt			
	mean	std	min	max	mean	std	min	max	mean	std	min	max	mean	std	min	max
MILP	96.36	21.28	63	155	-	-	-	-	-	-	-	-	-	-	-	-
LB^*_5	6.28	3.95	1	17	1.25	0.51	1.00	4.61	1.24	0.51	1.00	4.61	1.24	0.50	1.00	4.61
LB^*_{10}	6.50	4.55	1	24	1.18	0.24	1.00	1.99	1.18	0.24	1.00	1.99	1.18	0.24	1.00	1.99
LB^*_{15}	6.60	4.64	1	28	1.18	0.24	1.00	1.99	1.18	0.24	1.00	1.99	1.18	0.24	1.00	1.99
LB^*_{20}	6.37	4.34	1	26	1.18	0.24	1.00	1.99	1.18	0.24	1.00	1.99	1.18	0.24	1.00	1.99
LB^*_{DO}	6.14	4.10	1	25	1.18	0.24	1.00	1.99	1.18	0.24	1.00	1.99	1.18	0.24	1.00	1.99
$LB^*_{\widehat{\mathscr{D}}_{KPY}}$	17.64	2.66	13	25	1.18	0.24	1.00	1.99	-	-	-	-	1.18	0.24	1.00	1.99
$LB^*_{\mathscr{D}_{MJC}}$	10.38	4.34	4	24	1.18	0.24	1.00	1.99	-	-	-	-	1.18	0.24	1.00	1.99
$LB^*_{\mathscr{D}_{MJC}}$	8.51	3.08	4	25	1.18	0.24	1.00	1.99	-	-	-	-	1.18	0.24	1.00	1.99
LB_{CG}	1.51	0.62	0	3	1.64	0.47	1.00	2.00	-	-	-	-	1.63	0.46	1.00	2.00
LB_{KZ}	0.89	0.49	0	2	1.67	0.47	1.00	2.00	-	-	-	-	1.67	0.47	1.00	2.00

Table 6.2: R100-1000-0.5-0.5

	time for lb (ms)				Gap-medSol				Gap-minSol				Gap-Opt			
	mean	std	min	max	mean	std	min	max	mean	std	min	max	mean	std	min	max
MILP	8729.55	1113.63	5967	11226	-	-	-	-	-	-	-	-	-	-	-	-
LB^*_5	47.29	28.43	3	96	1.30	0.41	1.00	4.39	1.28	0.41	1.00	4.39	1.28	0.40	1.00	4.39
LB^*_{10}	53.67	36.84	4	127	1.23	0.24	1.00	1.96	1.21	0.23	1.00	1.96	1.21	0.23	1.00	1.96
LB^*_{15}	53.25	36.74	3	127	1.23	0.24	1.00	1.96	1.21	0.23	1.00	1.96	1.21	0.23	1.00	1.96
LB^*_{20}	52.72	36.01	3	128	1.23	0.24	1.00	1.96	1.21	0.23	1.00	1.96	1.21	0.23	1.00	1.96
LB^*_{DO}	54.98	37.47	4	127	1.23	0.21	1.00	1.96	1.21	0.23	1.00	1.96	1.21	0.23	1.00	1.96
$LB^*_{\widehat{\mathscr{D}}_{KPY}}$	1734.40	218.08	847	2166	1.23	0.21	1.00	1.96	-	-	-	-	1.21	0.23	1.00	1.96
$LB^*_{\mathscr{D}_{MJC}}$	501.34	72.44	378	650	1.23	0.21	1.00	1.96	-	-	-	-	1.21	0.23	1.00	1.96
$LB^*_{\mathscr{D}_{MJC}}$	542.17	196.21	283	1410	1.23	0.21	1.00	1.96	-	-	-	-	1.21	0.23	1.00	1.96
LB_{CG}	22.32	2.97	16	31	1.73	0.44	1.00	2	-	-	-	-	1.71	0.43	1.00	2.00
LB_{KZ}	8.10	2.23	2	16	1.74	0.44	1.00	2.00	-	-	-	-	1.72	0.44	1.00	2.00

Table 6.3: R500-1000-0.5-0.5

	time for lb (ms)				Gap-medSol				Gap-minSol				Gap-Opt			
	mean	std	min	max	mean	std	min	max	mean	std	min	max	mean	std	min	max
MILP	59920.20	7593.73	43871	83568	-	-	-	-	-	-	-	-	-	-	-	-
LB^*_5	131.93	73.71	8	285	1.31	0.37	1.00	3.47	1.29	0.33	1.00	3.19	1.29	0.33	1.00	3.19
LB^*_{10}	145.32	97.80	7	454	1.28	0.28	1.00	1.97	1.26	0.27	1.00	1.97	1.26	0.26	1.00	1.97
LB^*_{15}	145.44	97.22	8	460	1.28	0.28	1.00	1.97	1.26	0.27	1.00	1.97	1.26	0.26	1.00	1.97
LB^*_{20}	146.04	98.59	8	455	1.28	0.28	1.00	1.97	1.26	0.27	1.00	1.97	1.26	0.26	1.00	1.97
LB^*_{DO}	146.47	98.45	8	458	1.28	0.28	1.00	1.97	1.26	0.27	1.00	1.97	1.26	0.26	1.00	1.97
$LB^*_{\widehat{\mathscr{D}}_{KPY}}$	6883.86	860.32	3877	9275	1.28	0.28	1.00	1.97	1.26	0.27	1.00	1.97	1.26	0.26	1.00	1.97
$LB^*_{\mathscr{D}_{MJC}}$	2328.92	300.24	1757	3195	1.28	0.28	1.00	1.97	1.26	0.27	1.00	1.97	1.26	0.26	1.00	1.97
$LB^*_{\mathscr{D}_{MJC}}$	3128.35	1231.49	1458	7638	1.28	0.28	1.00	1.97	1.26	0.27	1.00	1.97	1.26	0.26	1.00	1.97
LB_{CG}	87.46	10.67	53	117	1.80	0.38	1.00	2.00	-	-	-	-	1.77	0.38	1.00	2.00
LB_{KZ}	24.06	6.95	5	38	1.83	0.38	1.00	2.00	-	-	-	-	1.80	0.38	1.00	2.00

Table 6.4: R1000-1000-0.5-0.5

	time for lb (ms)				Gap-medSol				Gap-minSol				Gap-Opt			
	mean	std	min	max	mean	std	min	max	mean	std	min	max	mean	std	min	max
MILP	60.13	24.59	19	139	-	-	-	-	-	-	-	-	-	-	-	-
LB^*_5	7.01	1.12	6	11	inf	inf	1.52	inf	inf	inf	1.52	inf	inf	nan	1.52	inf
LB^*_{10}	14.21	1.31	13	21	1.79	0.24	1.37	3.18	1.78	0.23	1.37	3.18	1.73	0.22	1.37	3.09
LB^*_{15}	24.28	2.05	23	32	1.56	0.11	1.29	1.91	1.55	0.11	1.29	1.91	1.51	0.10	1.29	1.87
LB^*_{20}	35.93	1.90	34	46	1.48	0.10	1.25	1.73	1.47	0.10	1.25	1.73	1.44	0.08	1.25	1.71
LB^*_{DO}	110.06	38.38	38	218	1.42	0.08	1.19	1.68	1.40	0.08	1.19	1.68	1.37	0.07	1.19	1.64
$LB^*_{\widehat{\mathscr{P}}_{KPY}}$	4.64	0.67	4	8	1.42	0.08	1.19	1.68	-	-	-	-	1.37	0.07	1.19	1.64
$LB^*_{\mathscr{P}_{MJC}}$	32.20	12.60	11	70	1.42	0.08	1.19	1.68	-	-	-	-	1.37	0.07	1.19	1.64
$LB^*_{\mathscr{D}_{MJC}}$	33.32	11.23	13	73	1.42	0.08	1.19	1.68	-	-	-	-	1.37	0.07	1.19	1.64
LB_{CG}	0.50	0.50	0	1	1.91	0.09	1.66	2.00	-	-	-	-	1.85	0.10	1.63	2.00
LB_{KZ}	0.20	0.40	0	1	2.00	0.00	2.00	2.00	-	-	-	-	1.94	0.07	1.72	2.00

Table 6.5: K102-1000-1.-2

	time for lb (ms)				Gap-medSol				Gap-minSol				Gap-Opt			
	mean	std	min	max	mean	std	min	max	mean	std	min	max	mean	std	min	max
MILP	592.29	250.74	335	2305	-	-	-	-	-	-	-	-	-	-	-	-
LB^*_5	21.51	2.87	19	34	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf
LB^*_{10}	48.23	3.86	44	70	1.50	0.34	1.05	2.55	1.48	0.33	1.03	2.55	1.43	0.32	1.03	2.47
LB^*_{15}	79.97	3.75	74	109	1.28	0.16	1.04	1.88	1.25	0.15	1.02	1.77	1.22	0.15	1.02	1.74
LB^*_{20}	117.36	5.05	107	147	1.20	0.13	1.03	1.81	1.16	0.12	1.01	1.81	1.14	0.11	1.01	1.80
LB^*_{DO}	298.43	125.86	107	847	1.11	0.05	1.02	1.23	1.05	0.03	1.01	1.15	1.05	0.02	1.01	1.15
$LB^*_{\widehat{\mathscr{P}}_{KPY}}$	117.69	10.20	91	145	1.11	0.05	1.02	1.23	-	-	-	-	1.05	0.02	1.01	1.15
$LB^*_{\mathscr{P}_{MJC}}$	137.13	42.56	58	367	1.11	0.05	1.02	1.23	-	-	-	-	1.05	0.02	1.01	1.15
$LB^*_{\mathscr{D}_{MJC}}$	260.66	72.08	109	507	1.11	0.05	1.02	1.23	-	-	-	-	1.05	0.02	1.01	1.15
LB_{CG}	3.62	0.52	3	5	1.64	0.10	1.42	1.91	-	-	-	-	1.56	0.08	1.40	1.77
LB_{KZ}	2.30	0.48	2	4	2.00	0.00	2.00	2.00	-	-	-	-	1.90	0.05	1.68	2.00

Table 6.6: K402-1000-1.-10

	time for lb (ms)				Gap-medSol				Gap-minSol				Gap-Opt			
	mean	std	min	max	mean	std	min	max	mean	std	min	max	mean	std	min	max
MILP	4344.55	2043.85	2136	12849	-	-	-	-	-	-	-	-	-	-	-	-
LB^*_5	55.68	2.68	52	71	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf
LB^*_{10}	124.02	4.82	116	148	1.64	0.50	1.15	5.51	1.63	0.50	1.15	5.51	1.56	0.48	1.15	5.31
LB^*_{15}	201.77	6.63	189	228	1.39	0.20	1.11	2.36	1.38	0.20	1.09	2.36	1.32	0.18	1.09	2.19
LB^*_{20}	291.10	9.70	276	319	1.30	0.15	1.11	2.29	1.29	0.15	1.11	2.29	1.23	0.14	1.08	2.19
LB^*_{DO}	4700.47	1741.47	1924	11682	1.10	0.03	1.04	1.21	1.05	0.02	1.02	1.12	1.04	0.01	1.02	1.09
$LB^*_{\widehat{\mathscr{P}}_{KPY}}$	1032.94	82.58	833	1321	1.10	0.03	1.04	1.21	-	-	-	-	1.04	0.01	1.02	1.09
$LB^*_{\mathscr{P}_{MJC}}$	1096.37	241.22	710	1770	1.10	0.03	1.04	1.21	-	-	-	-	1.04	0.01	1.02	1.09
$LB^*_{\mathscr{D}_{MJC}}$	3985.08	680.85	2283	5834	1.10	0.03	1.04	1.21	-	-	-	-	1.04	0.01	1.02	1.09
LB_{CG}	9.71	0.81	8	15	1.63	0.06	1.50	1.81	-	-	-	-	1.55	0.05	1.43	1.68
LB_{KZ}	5.97	0.71	5	10	2.00	0.00	2.00	2.00	-	-	-	-	1.90	0.05	1.75	1.99

Table 6.7: K1002-1000-1.-10

	time for lb (ms)				Gap-medSol				Gap-minSol				Gap-Opt			
	mean	std	min	max	mean	std	min	max	mean	std	min	max	mean	std	min	max
MILP	22826.90	9761.98	7450	57675	-	-	-	-	-	-	-	-	-	-	-	-
LB^*_5	118.77	4.69	110	140	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf
LB^*_{10}	256.66	9.72	242	294	1.72	0.62	1.20	5.06	1.71	0.62	1.20	5.06	1.63	0.59	1.16	4.81
LB^*_{15}	413.50	11.51	393	449	1.42	0.20	1.15	2.33	1.42	0.20	1.15	2.33	1.34	0.18	1.12	2.23
LB^*_{20}	588.99	14.33	558	629	1.34	0.15	1.15	2.01	1.34	0.15	1.15	2.01	1.27	0.14	1.10	1.89
LB^*_{DO}	52684.10	23175.80	21865	180398	1.10	0.02	1.06	1.16	1.05	0.01	1.02	1.09	1.04	0.01	1.02	1.06
$LB^*_{\widehat{\mathscr{P}}_{KPY}}$	2569.08	359.57	1966	3913	1.10	0.02	1.06	1.16	-	-	-	-	1.04	0.01	1.02	1.06
$LB^*_{\mathscr{P}_{MJC}}$	6159.52	1311.35	3953	9760	1.10	0.02	1.06	1.16	-	-	-	-	1.04	0.01	1.02	1.06
$LB^*_{\mathscr{D}_{MJC}}$	34758.00	5787.99	21827	53184	1.10	0.02	1.06	1.16	-	-	-	-	1.04	0.01	1.02	1.06
LB_{CG}	20.39	0.95	19	26	1.62	0.04	1.54	1.73	-	-	-	-	1.54	0.04	1.45	1.67
LB_{KZ}	12.23	0.65	11	15	2.00	0.00	2.00	2.00	-	-	-	-	1.90	0.03	1.81	1.97

Table 6.8: K2002-1000-1.-10

	time for lb (ms)				Gap-medSol				Gap-minSol			
	mean	std	min	max	mean	std	min	max	mean	std	min	max
LB^*_5	1529.98	1077.65	47	3643	3.03	8.33	1.12	82.80	3.03	8.33	1.09	82.80
LB^*_{10}	3393.42	2359.67	103	8095	1.40	0.20	1.06	2.27	1.39	0.20	1.06	2.25
LB^*_{15}	5752.06	3963.95	149	13300	1.30	0.14	1.05	1.77	1.29	0.15	1.05	1.77
LB^*_{20}	8083.67	5669.42	141	19378	1.27	0.12	1.05	1.66	1.26	0.12	1.05	1.61
LB^*_{DO}	48716.95	64774.72	139	333969	1.23	0.12	1.05	1.62	1.22	0.12	1.04	1.58
$LB^*_{\widehat{\mathscr{P}}_{KPY}}$	107127.28	37887.70	12683	230967	1.23	0.12	1.05	1.62	-	-	-	-
$LB^*_{\mathscr{P}_{MJC}}$	262040.03	278177.71	6284	1398590	1.23	0.12	1.05	1.62	-	-	-	-
$LB^*_{\mathscr{D}_{MJC}}$	253150.12	249415.21	17698	1315000	1.23	0.12	1.05	1.62	-	-	-	-
LB_{CG}	658.59	119.41	494	987	1.96	0.05	1.73	2.00	-	-	-	-
LB_{KZ}	248.71	176.25	7	598	2.00	0.00	2.00	2.00	-	-	-	-

Table 6.9: New York City

	time for lb (ms)				Gap-medSol				Gap-minSol			
	mean	std	min	max	mean	std	min	max	mean	std	min	max
LB^*_5	2243.13	1482.35	26	5773	1.70	0.43	1.00	3.37	1.69	0.43	1.00	3.34
LB^*_{10}	4717.81	3290.32	26	12826	1.51	0.25	1.00	2.07	1.49	0.25	1.00	2.07
LB^*_{15}	6693.88	5134.00	24	19480	1.47	0.23	1.00	1.89	1.46	0.23	1.00	1.89
LB^*_{20}	8278.93	7044.32	26	27046	1.47	0.23	1.00	1.89	1.45	0.23	1.00	1.89
LB^*_{DO}	11580.90	14054.12	29	75825	1.46	0.23	1.00	1.89	1.45	0.23	1.00	1.89
$LB^*_{\widehat{\mathscr{P}}_{KPY}}$	90649.60	34228.70	12531	250604	1.46	0.23	1.00	1.89	-	-	-	-
$LB^*_{\mathscr{P}_{MJC}}$	37484.20	27275.40	6586	182457	1.46	0.23	1.00	1.89	-	-	-	-
$LB^*_{\mathscr{D}_{MJC}}$	29387.00	14808.80	11616	95791	1.46	0.23	1.00	1.89	-	-	-	-
LB_{CG}	817.32	143.97	608	1397	1.94	0.15	1.00	2.00	-	-	-	-
LB_{KZ}	357.48	228.77	6	896	1.98	0.14	1.00	2.00	-	-	-	-

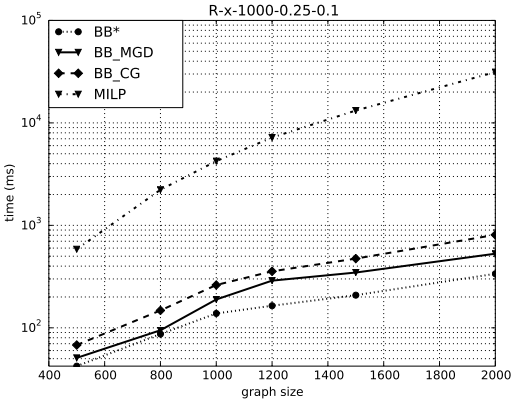
Table 6.10: San Francisco Bay

noted by Chassein and Goerigk, the number of edges in the graph increases with the layer size, which impacts badly on the computation times for the MILP. Regarding the way the branch and bounds compare themselves, usually BB_{MGD} performs least except when the layer size is large, and BB_{CG} performs slightly better than BB^*. As seen in the previous experiment, LB^*_{DO} is indeed not favored by K-graphs due to the possible large number of interesting paths between s and t.

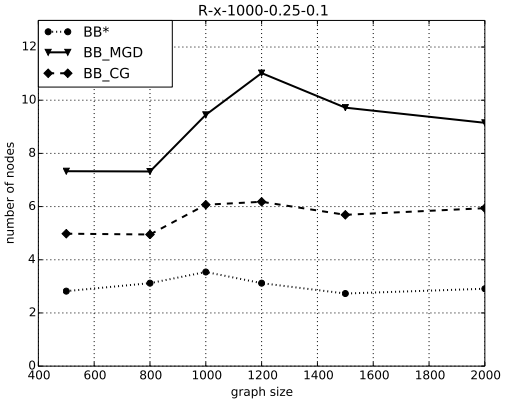
We conclude the experiments by comparing the three branch and bound algorithms on "large" R-graph instances. The results are presented in Table 6.11. Algorithm BB^* explores very few nodes and outperforms the two other branch and bound algorithms on the three types of graphs considered. This experiment confirms that BB_{MGD} is very sensitive to cost variability explaining its poor behavior on graphs with high cost variability. Finally, BB_{CG} is more penalized by the large size of the graph. This is easily explained by the fact that Suurballe's algorithm (called many times in BB_{CG}) requires to run a single-source *all-target* Dijkstra's algorithm while only single-source *single-target* Dijkstra's algorithms are required in BB^* and BB_{MGD}.

	R-10000-1000-0.5-0.001			R-10000-1000-1.-0.001			R-10000-1000-0.5-0.1		
	BB^*	BB_{CG}	BB_{MGD}	BB^*	BB_{CG}	BB_{MGD}	BB^*	BB_{CG}	BB_{MGD}
time (ms)	118.29	1006.06	1017.89	259.47	3336.59	39430.90	17536.20	187045.00	51615.70
number of nodes	4.42	16.16	77.16	7.17	48.13	3037.98	4.31	12.94	58.93

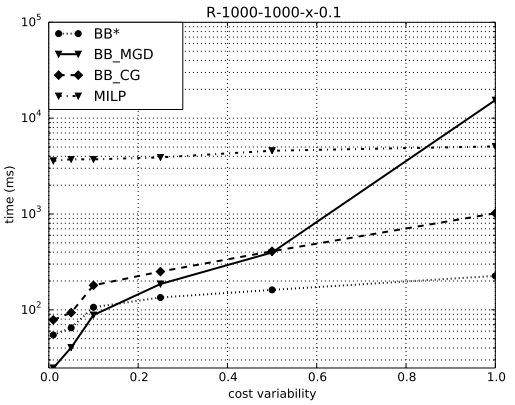
Table 6.11: Large R-graphs



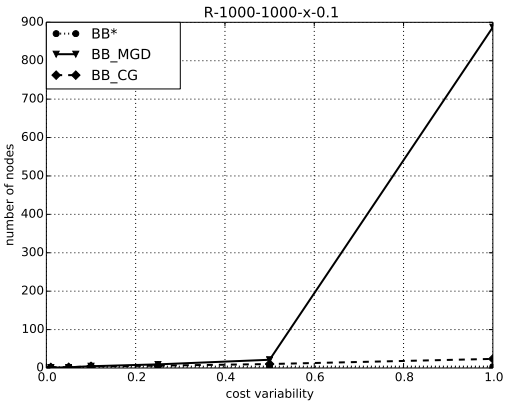
a) R-Graphs with different numbers of nodes



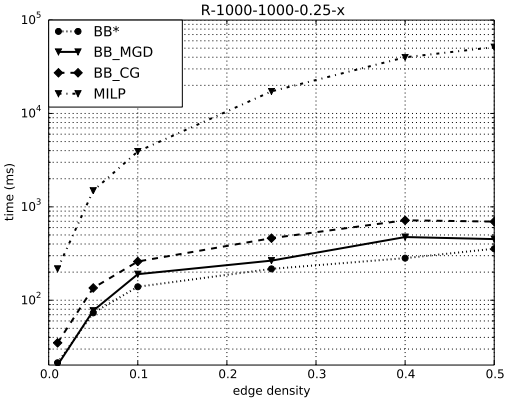
b) R-Graphs with different numbers of nodes



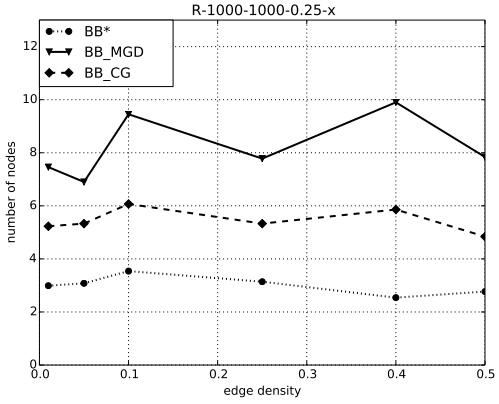
c) R-Graphs with different cost variabilities



d) R-Graphs with different cost variabilities

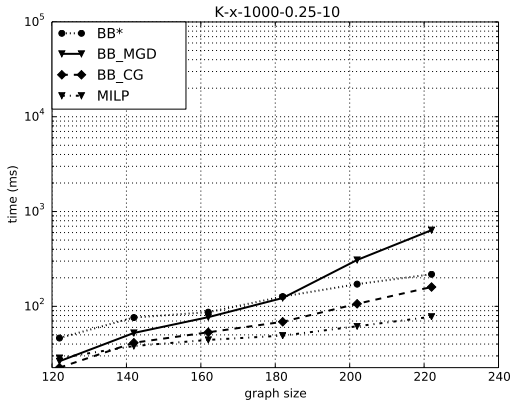


e) R-Graphs with different edge densities

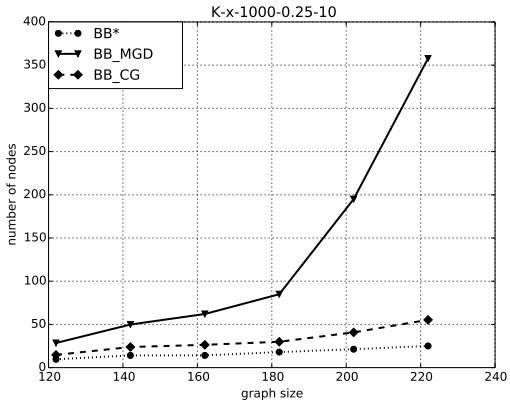


f) R-Graphs with different edge densities

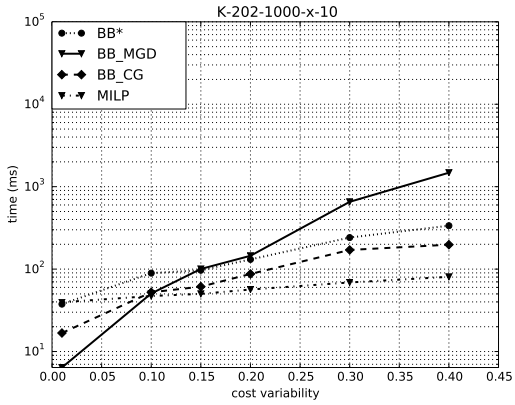
Figure 6.3: Average computation time and number of nodes of the branch and bound tree for R-graphs



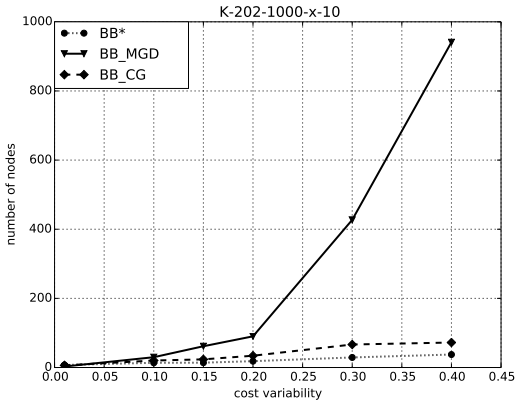
a) K-Graphs with different numbers of nodes



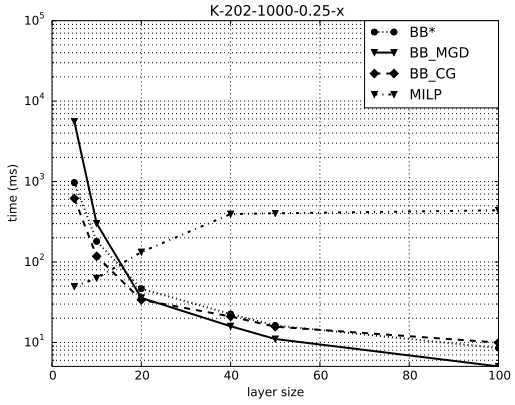
b) K-Graphs with different numbers of nodes



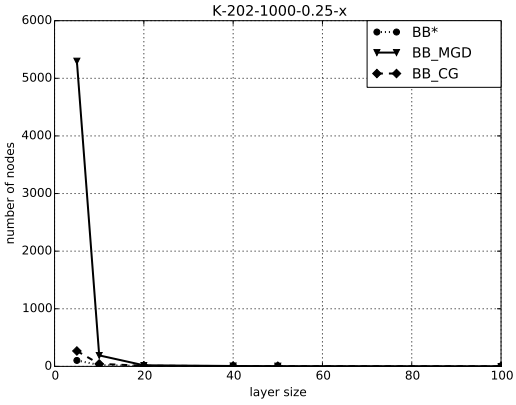
c) K-Graphs with different cost variabilities



d) K-Graphs with different cost variabilities



e) K-Graphs with different layer sizes



f) K-Graphs with different layer sizes

Figure 6.4: Average computation time and number of nodes of the branch and bound tree for K-graphs

6.6 Conclusion

In this chapter, we addressed robust combinatorial optimization problems with interval data. In this domain, we have presented a game-theoretic view of the optimization of the minmax regret criterion similar to the one proposed by Mastin *et al.* [2015]. In this game theoretic view, the optimization (in a randomized setting) of the minmax regret criterion is formulated as the search for a Nash equilibrium in a two-player zero-sum game where one player selects a feasible solution, and the other player selects a possible scenario.

Based on this game-theoretic view of robust optimization, we derived a general anytime double oracle algorithm to compute an accurate lower bound on the value of an optimal minmax regret solution. We have seen that this (instance dependent) lower bound makes it possible to find an approximation ratio for the midpoint solution (the optimal solution in the scenario where each cost is set to the middle of the corresponding interval) which is worth two in the worst case.

Furthermore, we discussed how this lower bound can be efficiently used in a branch and bound algorithm to compute a minmax regret solution, and provided experimental results on the robust shortest path problem. Compared to other approaches proposed in the literature, this approach leads to a significant improvement of the computation times on many instances.

6.7 References

R. K Ahuja, T. L Magnanti, and J. B Orlin. *Network flows: theory, algorithms, and applications*. Prentice hall, 1993. 202

H. Aissi, C. Bazgan, and D. Vanderpooten. Min-max and min-max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427–438, 2009. 192, 193

A. B. Chassein and M. Goerigk. A new bound for the midpoint solution in minmax regret optimization with an application to the robust shortest path problem. *European Journal of Operational Research*, 244(3):739–747, 2015. 190, 191, 192, 195, 196, 205, 206

V. Chvatal. *Linear programming*. Macmillan, 1983. 197

M. Conforti, G. Cornuéjols, and G. Zambelli. *Integer programming*, volume 271. Springer, 2014. 200

H. Gilbert and O. Spanjaard. A game theoretic bound for minmax regret optimization problems with interval data. In *European Journal of Operational Reasearch*, 2017. 190

O. E. Karasan, M. C. Pinar, and H. Yaman. Robust Shortest Path Problem with Interval Data. *Computers and Operations Research*, 2002. 193, 200, 201, 202

A. Kasperski and P. Zielinski. An approximation algorithm for interval data minmax regret combinatorial optimization problems. *Information Processing Letters*, 97(5):177–180, 2006. 190, 192, 208

A. Mastin, P. Jaillet, and S. Chin. Randomized minmax regret for combinatorial optimization under uncertainty. In *Int. Symp. on Algorithms and Computation*, pages 491–501. Springer, 2015. 190, 191, 192, 194, 198, 199, 203, 215

H.B. McMahan, G.J. Gordon, and A. Blum. Planning in the presence of cost functions controlled by an adversary. In *International Conference on Machine Learning*, pages 536–543, 2003. 191, 199

R. Montemanni and L.M. Gambardella. An exact algorithm for the robust shortest path problem with interval data. *Computers and Operations Research*, 31(10):1667–1680, September 2004. 201

R. Montemanni and L.M. Gambardella. The robust shortest path problem with interval data via benders decomposition. *4OR*, 3(4):315–328, 2005. 201, 203

R. Montemanni, L.M. Gambardella, and A.V. Donati. A branch and bound algorithm for the robust shortest path problem with interval data. *Operations Research Letters*, 32(3):225–232, 2004. 191, 201, 205, 206, 209

I. Pohl. *Bi-directional and heuristic search in path problems*. PhD thesis, Dept. of Computer Science, Stanford University., 1969. 204

J.W. Suurballe and R.E. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14(2):325–336, 1984. 205, 206

H. Yaman, O. E. Karasan, and M.C. Pinar. Minimum spanning tree problem with interval data. *Operations Research Letters*, 29:2001, 1999. 200, 203

Chapter 7

Fair Multi-Agent Optimization with Ordered Weighted Averages and Mixture Operators

" These men ask for just the same thing, fairness, and fairness only. This, so far as in my power, they, and all others, shall have. "

A. Lincoln

Section Contents

Summary of the chapter

In this chapter, we study multi-agent optimization problems with two classes of aggregation operators, namely Ordered Weighted Averages (OWAs) and Mixture Operators (MOs). These operators are, under known conditions on their parameters (see Chapter 1), compatible with the Pigou-Dalton transfer principle (fairness) and their optimization yields Pareto optimal solutions (efficiency). Thus, their optimization makes it possible to find fair and efficient solutions.

In the first part of the chapter, we focus on OWA operators with decreasing weights and we argue that considering randomized strategies in multi-agent optimization problems

makes it possible to obtain solutions with enhanced fairness properties. Then, we develop a game-theoretic view of the determination of an optimal randomized strategy with OWAs and derive complexity results and solution methods from it.

In the second part of the chapter, we come back to deterministic solutions and consider the optimization of MOs in allocation problems and in proportional representation problems. By identifying both problems as fractional programming problems, we are able to determine complexity results and solution methods.

This chapter is based on the following publication [Gilbert, 2017].

7.1 Introduction

Multi-agent optimization deals with problems where multiple agents are involved in the choice of a feasible solution. Multi-agent optimization procedures are required in many problems studied in artificial intelligence, such as proportional representation (typically winner determination under the Monroe and Chamberlin-Courant multiwinner voting rules, which determine the set of representatives by optimizing the total satisfaction or dissatisfaction of the voters [Procaccia *et al.*, 2008; Skowron *et al.*, 2015]), group recommendation (for instance movies to put on in-flight entertainment system [Skowron *et al.*, 2016]), fair division of indivisible goods [Lang and Rothe, 2016], or paper assignment problems (that consist in assigning reviewers to conference paper submissions [Goldsmith and Sloan, 2007; Nguyen *et al.*, 2016]).

In this chapter, we are more especially interested in fair multi-agent optimization, i.e., in procedures favoring solutions that fairly share satisfaction among agents. There are of course several ways of formalizing what is meant by "fair". We mean here by fair optimization that, when considering the vector of agent's satisfactions, it should be both Pareto optimal and well-balanced in the sense that there is no other solution that can be obtained from it by Pigou-Dalton transfers (this notion of fairness in multi-agent decision problems is presented in Section 1.3 on page 21). To obtain such solutions, we study multi-agent optimization problems with two classes of aggregation operators, namely Ordered Weighted Averages (OWAs) and Mixture Operators (MOs), which are, under known conditions on their parameters, compatible with the Pigou-Dalton transfer principle (OWAs and MOs are thoroughly presented in subsection 1.2.1 on page 15 and Section 1.3 on page 21).

In the first part of the chapter, we focus on OWA operators with decreasing weights. We argue that, due to the possibly conflicting agents' preferences, all feasible solutions can be unfair to some extent. In this concern, we show that randomness can help to "even things out" and introduce new solutions with enhanced fairness properties. We then develop a game-theoretic view of the determination of an optimal randomized strategy for OWAs with decreasing weights. Using the oracle methods that can be used with such game-theoretic formulations (see Section 3.1 on page 80), we derive complexity results and efficient solution methods.

In the second part of the chapter, we come back to deterministic solutions and consider the optimization of MOs in allocation problems and in proportional representation problems. The optimization of MOs in both problems will reveal NP-hard. However, we will be able to design solution methods by identifying these problems as fractional programming ones. Also, interestingly, we will see that some sub-problems will reveal of polynomial complexity, as the multi-agent assignment problem or the proportional representation problem under Chamberlin-Courant's voting scheme with single-peaked preferences.

The content of this chapter is summarized in Table 7.1.

	Section 1	Section 2	
Aggregation Operator	OWA	MO	
Setting	Randomized	Deterministic	
Multi-agent Decision Problem	General multi-agent problems	Allocation problems	Proportional representation problems
Complexity Results	P under the condition of Theorem 23	NP-hard in the general case, P for the assignment problem	NP-hard in the general case, P under some structures of preferences
Types of Algorithmic methods	Oracle methods from section 3.1	Oracle methods from section 3.2	

Table 7.1: Contributions of Chapter 7.

7.2 Fair Randomized Optimization with the OWA Operator

In this section, we study fair randomized optimization of multi-agent decision problems with OWA operators. Why do we consider randomized solutions? Our opinion is that due to the possibly conflicting agents' preferences, all feasible solutions can be unfair to some extent. In this case, randomization can introduce new solutions with enhanced fairness properties. The idea is that, instead of looking for a fair deterministic solution (also called *pure* solution) which may not exist, we will look for a lottery over feasible solutions (also called a *mixed* solution, adopting the game-theoretic terminology). A famous example by Machina [1989] is enlightening to illustrate this idea.

Example 53 (Machina's mom)**.** *A mother with two children has one indivisible treat. She can give it to either one of her children but not to both. In that case it is reasonable to imagine that the mother would prefer tossing a coin to decide which of her child should have the treat instead of choosing herself (i.e., choosing between the two feasible pure assignments, where a child receives the treat while the other one receives nothing). Indeed, in this case, both children have equal chances of having the treat. If the problem is a repeated one, this strategy also seems optimal as both children will receive the treat with equal frequencies in the long run.*

Randomization has mainly been studied for the assignment problem where, given n objects and n agents with individual preferences on the objects, one wishes to assign exactly one object to each agent in a fair and efficient manner. An extension of "mom's approach" to this problem with n agents is called the random serial dictatorship procedure: order the agents uniformly at random, and let them successively choose an object in that order [Abdulkadiroğlu and Sönmez, 1998]. This procedure induces a *bistochastic* matrix $P = (p_{ij})$ (i.e., a square matrix each of whose rows and columns sums to 1) where p_{ij} is the probability that agent i obtains object j. By the Birkhoff-von Neumann theorem [Von Neumann, 1953], matrix P may be represented by a (not necessarily unique) probability distribution over pure assignments, that is, there exists a mixed assignment whose

inferred allocation probabilities coincide with p_{ij}. Note that other more sophisticated random assignment rules can be found in the literature [Bogomolnaia and Moulin, 2001; Hylland and Zeckhauser, 1979; Katta and Sethuraman, 2006]. All these approaches are however hard to transpose to other multiagent optimization problems as they are dedicated to the assignment problem.

The notion of *popular mixed assignment* can nevertheless be transposed to any multiagent optimization problem. A pure assignment is deemed popular if there exists no other assignment that a majority of agents prefers to the former [Gärdenfors, 1975]. This definition has been extended to mixed assignments by Kavitha *et al.* [2011]: a mixed assignment p is popular if there is no other assignment q such that the expected number of agents who prefer the outcome of q to that of p is greater than $n/2$. By the minimax theorem, a popular mixed assignment always exists, and Kavitha *et al.* proposed a linear programming approach to compute it in polynomial time. The axiomatic properties of popular mixed assignments have been studied by Aziz *et al.* [2013]. Regarding fairness, they have shown that there always exists a popular mixed assignment that satisfies *equal treatment of equals*, i.e., agents with identical preferences receive identical random allocations. Equal treatment of equals is considered as a "minimal test for fairness" [Moulin, 1991].

Yet, the choice of a popular mixed solution does not preclude the possibility that a minority of agents are unsatisfied with all pure solutions considered in that mixed solution (this phenomenon is sometimes referred to as *tyranny of the majority*). In this section, following Lesca and Perny [2010] and Endriss [2013] who also studied fair optimization but in a deterministic setting, we adopt the viewpoint of the measurement of inequality [Moulin, 1991; Sen, 1973] to compare vectors of agents' expected utilities induced from mixed solutions. Our aim is to provide a procedure to determine a mixed solution optimizing a criterion compatible with the Pigou-Dalton principle. This holds in particular for an OWA with decreasing weights. In the following, we provide two generic procedures for computing a mixed solution optimizing an OWA with decreasing weights. They are both based on a game-theoretic view of fair multi-agent optimization problems. One procedure is based on a cutting plane method while the other one is a double oracle algorithm (these types of methods are presented in Section 3.1 on page 80).

7.2.1 Description of the Problem

We adopt the same setting as in Section 1.3. Let $\mathscr{P}$ be a multi-agent optimization problem, and $\mathcal{N} = \{1,\dots,n\}$ be the finite set of agents involved in $\mathscr{P}$. We denote by $\mathcal{X} \subseteq \{0,1\}^p$ the set of feasible solutions of $\mathscr{P}$ and we assume that $\mathcal{X}=\{\mathbf{x} \in \{0,1\}^p : A\mathbf{x} = \mathbf{b}\}$ where A is a *constraint matrix* involving a polynomial number (in p) of rows and $\mathbf{b}$ is integral. A feasible solution of $\mathscr{P}$ is thus represented by a binary vector $\mathbf{x} \in \mathcal{X}$. Each agent i is endowed with a utility function $v_i : \mathcal{X} \to \mathbb{R}^+$ (to maximize, without loss of generality) defined by $v_i(\mathbf{x}) = \sum_{j=1}^{p} v_{ij}x_j$ where v_{ij} is the utility of having $x_j = 1$ for agent i. Every solution $\mathbf{x}$ induces therefore a utility vector $\mathbf{v}(\mathbf{x}) = (v_1(\mathbf{x}),\dots,v_n(\mathbf{x}))$.

Example 54 (Machina's mom cont')**.** *Let us come back to Example 53. We have* $\mathcal{X} = \{\mathbf{x} \in \{0,1\}^2 : x_1 + x_2 = 1\}$*, where* $x_i = 1$ *if child* i *receives the treat. There are therefore two pure solutions* $\mathbf{x^1} = (1,0)$ *and* $\mathbf{x^2} = (0,1)$*. We assume that* $v_i(\mathbf{x}) = x_i$ *for* $i \in \{1,2\}$ *(the utility of* i *is 1 if she receives the treat, 0 otherwise). The Lorenz curves associated with* $\mathbf{v}(\mathbf{x^1})$ *and* $\mathbf{v}(\mathbf{x^2})$ *are identical. They correspond to the bold curve in Figure 7.1 (The notion of Lorenz curve is defined in Definition 22 on page 22). The surface of the area in light gray is an indicator of the level of fairness (the smaller the fairer).*

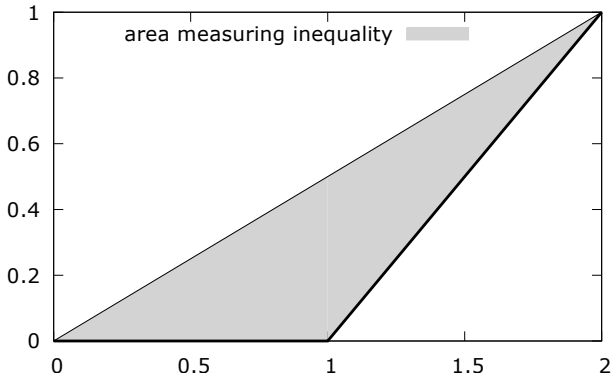


Figure 7.1: Lorenz curve and inequality measurement.

The OWA operator in the deterministic and randomized setting. The OWA operator has been presented in subsection 1.2.1 in a deterministic setting. We recall the definition of this operator. Given $\mathbf{w} = (w_1, \ldots, w_n)$, a vector of positive weights summing up to one. The $\texttt{OWA}_{\mathbf{w}}(\cdot)$ operator induced by $\mathbf{w}$ is defined by:

$$\forall \mathbf{x} \in \mathbb{R}^n, \texttt{OWA}_{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{n} w_i x_{\boldsymbol{\sigma}(i)}$$

where $\boldsymbol{\sigma}$ is a permutation of $\{1, \ldots, n\}$ such that $x_{\boldsymbol{\sigma}(1)} \leq x_{\boldsymbol{\sigma}(2)} \leq \ldots \leq x_{\boldsymbol{\sigma}(n)}$. Importantly, we also recall that this operator enables to find efficient (Pareto optimal) and fair solutions (with respect to the Pigou-Dalton principle) if the weights are strictly positive and strictly decreasing, which we assume in the sequel.

We now investigate the properties that hold for mixed solutions optimizing an OWA operator. A mixed solution is a lottery over solutions in $\mathscr{X}$ and is denoted by $\mathrm{P}_{\mathscr{X}}$. The set of all possible mixed solutions is denoted by $\Delta_{\mathscr{X}}$. Given a mixed solution $\mathrm{P}_{\mathscr{X}}$, we denote by $\mathrm{P}_{\mathscr{X}}(\mathbf{x})$ the probability assigned to pure solution $\mathbf{x}$. As usual, the definition of a utility vector is extended by linearity to mixed solutions. More formally, $v_i(\mathrm{P}_{\mathscr{X}}) = \sum_{\mathbf{x} \in \mathscr{X}} \mathrm{P}_{\mathscr{X}}(\mathbf{x}) v_i(\mathbf{x})$.

When considering mixed strategies, two different notions of Pareto-optimality are distinguished:

Definition 44. *[Katta and Sethuraman, 2006] Given a mixed strategy* $\mathrm{P}_{\mathscr{X}}$ *over pure solutions in* $\mathscr{X}$*:*

- $\mathrm{P}_{\mathscr{X}}$ *is* ex ante *Pareto-optimal if* $\mathrm{P}_{\mathscr{X}}$ *is Pareto-optimal in* $\Delta_{\mathscr{X}}$*.*
- $\mathrm{P}_{\mathscr{X}}$ *is* ex post *Pareto-optimal if it is a probability distribution over Pareto-optimal pure solutions in* $\mathscr{X}$*.*

An OWA optimal mixed solution is always ex ante Pareto-optimal (if weights are strictly positive), by compatibility of OWA with Pareto dominance [Fodor *et al.*, 1995]. Note that ex ante Pareto-optimality implies ex post Pareto-optimality.

We now illustrate, in Example 55, the enhanced abilities of OWA optimal mixed solutions with respect to fairness, compared to pure solutions.

Example 55 (Machina's mom cont')**.** *Coming back again to Example 53, consider pure solutions* $\mathbf{x^1}, \mathbf{x^2}$ *and the mixed solution* $\mathrm{P}_{\mathscr{X}}$ *defined by* $\mathrm{P}_{\mathscr{X}}(\mathbf{x^i}) = 0.5$ *for* $i \in \{1, 2\}$*. This corresponds to the mixed solution discussed earlier in Example 53 in which both children receive the treat with equal probabilities. The weights of the OWA operator are set to* $w_1 = 0.75$ *and* $w_2 = 0.25$*. With these (strictly decreasing) weights, we have:*

$$\begin{aligned}
\mathtt{OWA}_{\mathbf{w}}(\mathbf{v}(\mathbf{x^1})) &= \mathtt{OWA}_{\mathbf{w}}((1,0)) = 0.75\times 0 + 0.25\times 1 = 0.25\\
\mathtt{OWA}_{\mathbf{w}}(\mathbf{v}(\mathbf{x^2})) &= \mathtt{OWA}_{\mathbf{w}}((0,1)) = 0.75\times 0 + 0.25\times 1 = 0.25\\
\mathtt{OWA}_{\mathbf{w}}(\mathbf{v}(\mathrm{P}_{\mathscr{X}})) &= \mathtt{OWA}_{\mathbf{w}}((0.5,0.5)) = 0.75\times 0.5 + 0.25\times 0.5 = 0.5
\end{aligned}$$

because $\mathbf{v}(\mathrm{P}_{\mathscr{X}}) = 0.5\mathbf{v}(\mathbf{x^1}) + 0.5\mathbf{v}(\mathbf{x^2}) = (0.5, 0.5)$. *The mixed solution* $\mathrm{P}_{\mathscr{X}}$ *is therefore preferred to both pure solutions. This is not surprising as the Lorenz curve associated with* $\mathbf{v}(\mathrm{P}_{\mathscr{X}})$ *coincides with the diagonal in Figure 7.1.*

As witnessed by this example, for mixed solutions in allocation problems, the Pigou-Dalton principle guarantees the desirable property of equal treatment of equals (more precisely, there always exists *an* optimal mixed solution where agents with equal preferences receive the same random allocation), which is of course not the case in pure strategies. We now turn in the next subsection to the question of computing an OWA optimal mixed solution.

7.2.2 A Game-Theoretic View

To model the problem of determining an OWA optimal mixed solution as the determination of a mixed Nash equilibrium in a zero-sum two-player game, the following observation, holding for decreasing weights w_i, reveals useful:

$$\forall \mathrm{P}_{\mathscr{X}} \in \Delta_{\mathscr{X}}, \quad \mathtt{OWA}_{\mathbf{w}}(\mathbf{v}(\mathrm{P}_{\mathscr{X}})) = \min_{\boldsymbol{\sigma}\in\Sigma} \mathtt{WA}_{\mathbf{w}_{\boldsymbol{\sigma}}}(\mathbf{v}(\mathrm{P}_{\mathscr{X}}))$$

where Σ is the set of all permutations of $\{1,\ldots,n\}$ and $\mathbf{w}_{\boldsymbol{\sigma}} = (w_{\boldsymbol{\sigma}(1)},\ldots,w_{\boldsymbol{\sigma}(n)})$. Thus, maximizing $\mathtt{OWA}_{\mathbf{w}}(\mathbf{v}(\mathrm{P}_{\mathscr{X}}))$ amounts to the following max min optimization problem:

$$\max_{\mathrm{P}_{\mathscr{X}}\in\Delta_{\mathscr{X}}} \min_{\boldsymbol{\sigma}\in\Sigma} \mathtt{WA}_{\mathbf{w}_{\boldsymbol{\sigma}}}(\mathbf{v}(\mathrm{P}_{\mathscr{X}})).$$

This is exactly the problem faced by player 1 in the zero-sum two-player game where the sets of pure strategies are the set of feasible solutions $\mathscr{X}$ for player 1 (also called $\mathbf{x}$-player) and the set of permutations Σ for player 2 (also called $\boldsymbol{\sigma}$-player), and where the payoffs are given by values $\mathtt{WA}_{\mathbf{w}_{\boldsymbol{\sigma}}}(\mathbf{v}(\mathbf{x}))$ for $\mathbf{x}\in\mathscr{X}$ and $\boldsymbol{\sigma}\in\Sigma$. Similarly to the $\mathbf{x}$-player, we denote by P_Σ a mixed strategy of the $\boldsymbol{\sigma}$-player, and by Δ_Σ the set of all her mixed strategies. Note that the WA operator is extended to mixed strategies by linearity:

$$\mathtt{WA}_{\mathbf{w}_{\mathrm{P}_\Sigma}}(\mathbf{v}(\mathrm{P}_{\mathscr{X}})) = \sum_{\mathbf{x}\in\mathscr{X}} \sum_{\boldsymbol{\sigma}\in\Sigma} \mathrm{P}_{\mathscr{X}}(\mathbf{x})\mathrm{P}_\Sigma(\boldsymbol{\sigma})\mathtt{WA}_{\mathbf{w}_{\boldsymbol{\sigma}}}(\mathbf{v}(\mathbf{x})).$$

Given any two-player zero-sum game, determining a mixed Nash equilibrium can be done by linear programming [Chvatal, 1983] (as explained in subsection 3.1.1 on page 83). Indeed, assume that player 1 (resp. player 2) has k (resp. m) pure strategies and that M_{ij} denotes the payoff of player 1 when strategies i and j are played by respectively player 1 and player 2. Then, a mixed Nash equilibrium of this game can be determined by solving the linear program $\mathscr{P}_{\mathrm{NE}}$ (NE for Nash Equilibrium) given below:

$$\underbrace{\begin{cases}
\max\limits_{v,p_1,\ldots,p_k} v & \\
\sum\limits_{i=1}^{k} p_i \mathrm{M}_{ij} \geq v \quad \forall j\in\{1,\ldots,m\} & (7.1)\\
\sum\limits_{i=1}^{k} p_i = 1 & \\
v\in\mathbb{R} \quad p_i \geq 0 \quad \forall i\in\{1,\ldots,k\} &
\end{cases}}_{\mathscr{P}_{\mathrm{NE}}}
\qquad
\underbrace{\begin{cases}
\max\limits_{v,p_{\mathbf{x}}:\mathbf{x}\in\mathscr{X}} v & \\
\sum\limits_{i=1}^{n} w_{\boldsymbol{\sigma}(i)} \big(\sum\limits_{\mathbf{x}\in\mathscr{X}} p_{\mathbf{x}} v_i(\mathbf{x})\big) \geq v \quad \forall \boldsymbol{\sigma}\in\Sigma & (7.2)\\
\sum\limits_{\mathbf{x}\in\mathscr{X}} p_{\mathbf{x}} = 1 & \\
v\in\mathbb{R} \quad p_{\mathbf{x}} \geq 0 \quad \forall \mathbf{x}\in\mathscr{X} &
\end{cases}}_{\mathscr{P}_{\mathrm{FO}}}$$

where p_i denotes the probability that player 1 selects pure strategy i and v denotes the value of the game. In the two-player zero-sum game defined by the OWA and the multi-agent optimization problem, $k = |\mathscr{X}|$, $m = |\Sigma|$ and for all $i, j \in \{1,\ldots,k\} \times \{1,\ldots,m\}$, $M_{ij} = \mathtt{WA}_{\mathbf{w}_{\boldsymbol{\sigma}^j}}(\mathbf{v}(\mathbf{x}^i))$, where $\boldsymbol{\sigma}^j$ (resp. $\mathbf{x}^i$) is the j^{th} element of Σ (resp. i^{th} element of $\mathscr{X}$). Hence, program $\mathscr{P}_{\text{NE}}$ can be rewritten as program $\mathscr{P}_{\text{FO}}$ (FO for Fair Optimization), given above, where constraints 7.2 are just the specification of constraints 7.1 in $\mathscr{P}_{\text{NE}}$ to our setting. However, program $\mathscr{P}_{\text{FO}}$ is too large to be solved directly as it involves an exponential number of variables and constraints ($|\mathscr{X}|+1$ variables and $|\Sigma|+1$ constraints).

This game-theoretic interpretation makes it possible to use the oracle methods presented in section 3.1 (on page 80).

Cutting Plane Method. The method proposed here is based on a reformulation of $\mathscr{P}_{\text{FO}}$ that involves a polynomial number of variables, by redefining the solution space. We recall that a solution $\mathbf{x} \in \mathscr{X}$ is encoded by a vector of p binary variables. A mixed solution $\mathrm{P}_{\mathscr{X}}$ thus induces a vector $\hat{\mathbf{x}} = (\hat{x}_1,\ldots,\hat{x}_p)$ where $\hat{x}_i = \sum_{\mathbf{x}\in\mathscr{X}} \mathrm{P}_{\mathscr{X}}(\mathbf{x}) x_i$. We denote by $\mathrm{CH}(\mathscr{X}) = \{\sum_{\mathbf{x}\in\mathscr{X}} \mathrm{P}_{\mathscr{X}}(\mathbf{x})\mathbf{x} : \mathrm{P}_{\mathscr{X}} \in \Delta_{\mathscr{X}}\}$ the convex hull of $\mathscr{X}$. Given $\hat{\mathbf{x}} \in \mathrm{CH}(\mathscr{X})$, we say that $\mathrm{P}_{\mathscr{X}}$ *implements* $\hat{\mathbf{x}}$ if $\sum_{\mathbf{x}\in\mathscr{X}} \mathrm{P}_{\mathscr{X}}(\mathbf{x})\mathbf{x} = \hat{\mathbf{x}}$. By Carathéodory's theorem, for any $\hat{\mathbf{x}} \in \mathrm{CH}(\mathscr{X})$ there exists a mixed solution that implements it with a combination of at most $p+1$ pure solutions (as $\mathscr{X} \subset \mathbb{R}^p$). This point is illustrated in Example 56.

Example 56. *Consider a one-to-one assignment problem with 3 agents. We denote by* (i,j,k) *the allocation where agents 1 (resp.* 2,3*) receives object i (resp.* j,k*). The set CH*($\mathscr{X}$) *includes here all bistochastic matrices of dimension 3. The bistochastic matrix*

$$\hat{\mathbf{x}} = \begin{pmatrix} 0.8 & 0.2 & 0 \\ 0 & 0.3 & 0.7 \\ 0.2 & 0.5 & 0.3 \end{pmatrix}$$

denotes the solution where agent i *receives object* j *with probability* $\hat{x}_{ij}$*. This solution can be implemented by the mixed solution* $\mathrm{P}_{\mathscr{X}}$ *where pure assignments* $(1,2,3)$*,* $(1,3,2)$ *and* $(2,3,1)$ *are returned with probabilities* 0.3*,* 0.5 *and* 0.2 *respectively. Indeed:*

$$0.3 \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + 0.5 \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} + 0.2 \times \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0.8 & 0.2 & 0 \\ 0 & 0.3 & 0.7 \\ 0.2 & 0.5 & 0.3 \end{pmatrix}$$

Hence, for any fair multi-agent optimization problem, problem $\mathscr{P}_{\text{FO}}$ can be reformulated as follows (CP for Cutting Plane):

$$\mathscr{P}_{\text{CP}} \begin{cases} \max\limits_{v,\hat{x}_1,\ldots,\hat{x}_p} v & \\ v \leq \sum\limits_{i=1}^{n} w_{\boldsymbol{\sigma}(i)} v_i(\hat{\mathbf{x}}) & \forall \boldsymbol{\sigma} \in \Sigma \quad (7.3) \\ \hat{\mathbf{x}} \in \mathrm{CH}(\mathscr{X}) & (7.4) \end{cases}$$

because $v_i(\hat{\mathbf{x}}) = \sum_{\mathbf{x}\in\mathscr{X}} \mathrm{P}_{\mathscr{X}}(\mathbf{x}) v_i(\mathbf{x})$ if $\mathrm{P}_{\mathscr{X}}$ implements $\hat{\mathbf{x}}$. Note that the linearization used here for the OWA objective function (valid only if the weights are decreasing) coincides with one proposed by Ogryczak and Śliwiński [2003]; by comparison, the above reasoning proves that determining an optimal mixed solution amounts to determining an optimal solution in $\mathrm{CH}(\mathscr{X})$. We will soon see how one can recover from an optimal solution in $\mathrm{CH}(\mathscr{X})$ a mixed strategy $\mathrm{P}_{\mathscr{X}}$ that implements it.

Even if the number of constraints in 7.3 (resp. in 7.4) is (resp. may be) exponential in n (resp. p) as $|\Sigma| = n!$ (resp. the number of facets of $\mathrm{CH}(\mathscr{X})$ may be exponential in p), these constraints can be handled efficiently by resorting to a *cutting plane algorithm* (this type of method is presented in subsection 3.1.4 on page 87). A cutting plane algorithm makes it possible to solve a linear program involving an exponential number of constraints to define the polyhedron P of feasible solutions, provided there exists a *separation oracle*. In program $\mathscr{P}_{\mathrm{CP}}$, polyhedron P is defined by constraints 7.3 and 7.4. Given $(v,\mathbf{x}) \in \mathbb{R} \times [0,1]^p$, a separation oracle should determine whether $(v,\mathbf{x})$ belongs to P or not, and finds a separating hyperplane in the latter case. The proposed separation oracle consists of a separation oracle for the polyhedron P_1 defined by constraints in 7.3 and a separation oracle for the polyhedron P_2 defined by constraints in 7.4:

- Given $(v,\mathbf{x})$, a separation oracle for P_1 consists in sorting values $v_1(\mathbf{x}) \dots v_n(\mathbf{x})$ to generate a permutation that minimizes $\sum_{i=1}^{n} w_{\boldsymbol{\sigma}(i)} v_i(\mathbf{x})$. Sorting these values can of course be performed in polynomial time. If $v \leq \sum_{i=1}^{n} w_{\boldsymbol{\sigma}(i)} v_i(\mathbf{x})$, then $\mathbf{x}$ belongs to P_1, otherwise $v = \sum_{i=1}^{n} w_{\boldsymbol{\sigma}(i)} v_i(\mathbf{x})$ defines a separating hyperplane.

- Regarding polyhedron P_2, if we can optimize over $\mathscr{X}$ in polynomial time, then we can separate over $\mathrm{CH}(\mathscr{X})$ in polynomial time by the polynomial time equivalence of optimization and separation [Grötschel *et al.*, 1981] (optimization → separation) (Theorem 10 on page 89).

Combining both oracles yields a separation oracle for polyhedron $\mathrm{P} = \mathrm{P}_1 \cap \mathrm{P}_2$, which is polynomial time if we can optimize over $\mathscr{X}$ in polynomial time. In this case, the complexity of solving $\mathscr{P}_{\mathrm{CP}}$ is polynomial by the polynomial time equivalence of optimization and separation (separation → optimization) (Theorem 10 on page 89).

Once an OWA optimal solution $\hat{\mathbf{x}} \in \mathrm{CH}(\mathscr{X})$ has been found, it remains to *actually* compute a mixed solution in $\Delta_{\mathscr{X}}$ that implements $\hat{\mathbf{x}}$. We illustrate the importance of this step in Example 57.

Example 57. *We come back to the one-to-one assignment problem with 3 agents of Example 56. In this problem, we recall that the point in CH($\mathscr{X}$) corresponding to the bistochastic matrix*

$$\hat{\mathbf{x}} = \begin{pmatrix} 0.8 & 0.2 & 0 \\ 0 & 0.3 & 0.7 \\ 0.2 & 0.5 & 0.3 \end{pmatrix}$$

can be implemented by the mixed solution $\mathrm{P}_{\mathscr{X}}$ *where pure assignments* $(1,2,3)$, $(1,3,2)$ *and* $(2,3,1)$ *are returned with probabilities* 0.3, 0.5 *and* 0.2 *respectively. Assume this is an OWA optimal solution. Using the mixed solution* $\mathrm{P}_{\mathscr{X}}$ *is easy; one just have to sample assignments* $(1,2,3)$, $(1,3,2)$ *and* $(2,3,1)$ *with probabilities* 0.3, 0.5 *and* 0.2. *However, it is unclear how to use the corresponding bistochastic matrix directly to sample assignments in an optimal way.*

For this purpose, following Mastin *et al.* [2015], let us consider the linear program $\mathscr{P}_{\mathrm{CH}(\mathscr{X})\to\Delta}$ below. Clearly, a feasible solution of $\mathscr{P}_{\mathrm{CH}(\mathscr{X})\to\Delta}$ induces a mixed solution $\mathrm{P}_{\mathscr{X}}$ that implements $\hat{\mathbf{x}}$, by setting $\mathrm{P}_{\mathscr{X}}(\mathbf{x}) = p_{\mathbf{x}}$. Nevertheless, this program has an exponential number of variables ($|\mathscr{X}|$ variables $p_{\mathbf{x}}$). To tackle this issue, let us consider the dual program $\mathscr{D}_{\mathrm{CH}(\mathscr{X})\to\Delta}$ below, where w is the dual variable of constraint 7.6 and variables u_i are the dual variables of constraints in 7.5.

$$\underbrace{\begin{cases} \min\limits_{p_{\mathbf{x}}:\mathbf{x}\in\mathscr{X}} 0 & \\ \sum\limits_{\mathbf{x}\in\mathscr{X}} p_{\mathbf{x}} x_i = \hat{x}_i \quad \forall i \in \{1,\ldots,p\} & (7.5) \\ \sum\limits_{\mathbf{x}\in\mathscr{X}} p_{\mathbf{x}} = 1 & (7.6) \\ p_{\mathbf{x}} \geq 0 \quad \forall \mathbf{x} \in \mathscr{X} & \end{cases}}_{\mathscr{P}_{\mathrm{CH}(\mathscr{X})\to\Delta}} \qquad \underbrace{\begin{cases} \max\limits_{w,u_1,\ldots,u_p} w - \sum\limits_{i=1}^{p} \hat{x}_i u_i & \\ w \leq \sum\limits_{i=1}^{p} u_i x_i \quad \forall \mathbf{x} \in \mathscr{X} & (7.7) \\ w \in \mathbb{R} \quad u_i \in \mathbb{R} \quad \forall i \in \{1,\ldots,p\} & (7.8) \end{cases}}_{\mathscr{D}_{\mathrm{CH}(\mathscr{X})\to\Delta}}$$

The probability of a pure solution $\mathbf{x}$ in the resulting mixed solution is given by the dual variable of the corresponding constraint in 7.7. The separation oracle for constraints in 7.7 requires to solve the single objective minimization variant of $\mathscr{P}$ where the utility of having $x_i = 1$ is given by variable u_i and where one aims to minimize the sum of utility values. If this problem is polynomial, the complexity of solving $\mathscr{D}_{\mathrm{CH}(\mathscr{X})\to\Delta}$ is polynomial by the polynomial time equivalence of optimization and separation (separation → optimization), and the mixed solution resulting from the optimization is guaranteed to have a polynomially bounded support in $\mathscr{X}$ (the support is the set of pure solutions with nonzero probability). By solving successively $\mathscr{P}_{\mathrm{CP}}$ and $\mathscr{D}_{\mathrm{CH}(\mathscr{X})\to\Delta}$, we can conclude:

Theorem 23. *Any randomized fair optimization problem for which the sum of utilities of the agents can be optimized in polynomial time in p is polynomially solvable in p and n.*

This result applies for instance to fair multi-agent one-to-one and many-to-many allocation problems, and to fair multi-agent matroid problems (where each agent assigns a cost to each element of the ground set). Note that, from a practical viewpoint, this approach is more especially relevant if the constraint matrix defining $\mathscr{X}$ is totally unimodular, in which case $\mathrm{CH}(\mathscr{X})$ is obtained by linear programming relaxation. In this case, the separation oracle of polyhedron P_2 is unnecessary.

In the next subsection, we present another algorithm to solve the game induced by the randomized OWA optimization problem. This algorithm has no polynomial time guarantee, but is competitive in practice.

Double Oracle Algorithm. The game can be solved by specifying a double oracle algorithm adapted to our problem (this type of approach is presented in subsection 3.1.3 on page 85). We define two oracles $\mathscr{O}_{\boldsymbol{\sigma}}(\cdot)$ and $\mathscr{O}_{\mathbf{x}}(\cdot)$ (whose implementations are described latter). Given a mixed strategy $\mathrm{P}_{\mathscr{X}}$ (resp. P_{Σ}), $\mathscr{O}_{\boldsymbol{\sigma}}(\mathrm{P}_{\mathscr{X}})$ (resp. $\mathscr{O}_{\mathbf{x}}(\mathrm{P}_{\Sigma})$) returns a pure strategy $\boldsymbol{\sigma}$ (resp. $\mathbf{x}$) that minimizes $\mathtt{WA}_{\mathbf{w}_{\boldsymbol{\sigma}}}(\mathbf{v}(\mathrm{P}_{\mathscr{X}}))$ (resp. maximizes $\mathtt{WA}_{\mathrm{P}_{\Sigma}}(\mathbf{v}(\mathbf{x}))$). The algorithm starts by considering only small subsets $\mathrm{S}_{\mathbf{x}}$ and $\mathrm{S}_{\boldsymbol{\sigma}}$ of pure strategies (singletons in Algorithm 19) for the $\mathbf{x}$-player and the $\boldsymbol{\sigma}$-player, and then grows those sets in every iteration by applying the best-response oracles to the optimal strategies (given by the current NE) in the game M restricted to pure strategies in $\mathrm{S}_{\mathbf{x}}$ and $\mathrm{S}_{\boldsymbol{\sigma}}$. At each iteration of the algorithm, an NE (in mixed strategy) of $\mathrm{M}(\mathrm{S}_{\mathbf{x}}, \mathrm{S}_{\boldsymbol{\sigma}})$ is computed via linear program $\mathscr{P}_{\mathrm{FO}}$ (where Σ is replaced by $\mathrm{S}_{\boldsymbol{\sigma}}$ in the definition of constraints 7.2 and $\mathscr{X}$ is replaced by $\mathrm{S}_{\mathbf{x}}$ in the last line). Convergence is achieved when the best-response oracles generate pure strategies that are already present in sets $\mathrm{S}_{\mathbf{x}}$ and $\mathrm{S}_{\boldsymbol{\sigma}}$.

We now specify the procedures $\mathscr{O}_{\mathbf{x}}(\cdot)$ and $\mathscr{O}_{\boldsymbol{\sigma}}(\cdot)$ used in our double oracle algorithm. Given a mixed strategy P_{Σ} of the $\boldsymbol{\sigma}$-player, $\mathscr{O}_{\mathbf{x}}(\mathrm{P}_{\Sigma})$ is a pure solution $\mathbf{x}$ which maximizes

[1] M denotes here the payoff matrix of the two-player zero-sum game involving the $\mathbf{x}$-player and the $\boldsymbol{\sigma}$-player, and $\mathrm{M}(\mathrm{S}_{\mathbf{x}}, \mathrm{S}_{\boldsymbol{\sigma}})$ the subgame restricted to $\mathrm{S}_{\mathbf{x}}$ and $\mathrm{S}_{\boldsymbol{\sigma}}$.

Algorithm 19: Double Oracle Algorithm for Fair Optimization

Data: Singletons $S_{\mathbf{x}} = \{\mathbf{x}\}$ and $S_{\boldsymbol{\sigma}} = \{\boldsymbol{\sigma}\}$ including an arbitrary solution and an arbitrary permutation
Result: a (possibly mixed) NE

converge ← False
while *converge is False* **do**
  Find a Nash equilibrium $(P_{\mathscr{X}}, P_{\Sigma})$ of the game $M(S_{\mathbf{x}}, S_{\boldsymbol{\sigma}})$[1]
  Find $\mathbf{x} = \mathscr{O}_{\mathbf{x}}(P_{\Sigma})$ and $\boldsymbol{\sigma} = \mathscr{O}_{\boldsymbol{\sigma}}(P_{\mathscr{X}})$
  if $\mathbf{x} \in S_{\mathbf{x}}$ *and* $\boldsymbol{\sigma} \in S_{\boldsymbol{\sigma}}$ **then**
    converge ← True
  else
    add $\mathbf{x}$ to $S_{\mathbf{x}}$ and/or $\boldsymbol{\sigma}$ to $S_{\boldsymbol{\sigma}}$
return $(P_{\mathscr{X}}, P_{\Sigma})$

$\sum_{\boldsymbol{\sigma}\in\Sigma} P_{\Sigma}(\boldsymbol{\sigma})\mathtt{WA}_{\mathbf{w}_{\boldsymbol{\sigma}}}(\mathbf{v}(\mathbf{x}))$ which can be rewritten as:

$$\sum_{\boldsymbol{\sigma}\in\Sigma} P_{\Sigma}(\boldsymbol{\sigma}) \sum_{i=1}^{n} w_{\boldsymbol{\sigma}(i)} v_i(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{\boldsymbol{\sigma}\in\Sigma} P_{\Sigma}(\boldsymbol{\sigma}) w_{\boldsymbol{\sigma}(i)} \right) v_i(\mathbf{x}).$$

Thus, computing $\mathscr{O}_{\mathbf{x}}(P_{\Sigma})$ amounts to solve problem $\mathscr{P}$ according to a WA criterion with weights $\hat{\boldsymbol{\sigma}}$ defined by $\hat{\boldsymbol{\sigma}}_i = \sum_{\boldsymbol{\sigma}\in\Sigma} P_{\Sigma}(\sigma) w_{\boldsymbol{\sigma}(i)}$. Note that, as optimal solutions according to WA criteria with strictly positive weights are Pareto optimal, Pareto-dominated solutions will not be generated by the algorithm.

We now turn to the best response procedure for the $\boldsymbol{\sigma}$-player. Given a mixed strategy $P_{\mathscr{X}}$ of the $\mathbf{x}$-player, $\mathscr{O}_{\boldsymbol{\sigma}}(P_{\mathscr{X}})$ is a permutation in Σ which minimizes $\mathtt{WA}_{\mathbf{w}_{\boldsymbol{\sigma}}}(\mathbf{v}(P_{\mathscr{X}}))$. Similarly to the separation oracle in the cutting plane method for solving $\mathscr{P}_{\mathrm{CP}}$, best response $\mathscr{O}_{\boldsymbol{\sigma}}(P_{\mathscr{X}})$ can be computed by sorting vector $\mathbf{v}(P_{\mathscr{X}})$. Interestingly, note that procedure $\mathscr{O}_{\boldsymbol{\sigma}}(\cdot)$ is independent of the multi-agent optimization problem $\mathscr{P}$ considered.

There is no polynomial guarantee on the number of iterations of this double oracle algorithm. It will nevertheless reveal efficient in practice, as will be shown by the numerical tests.

7.2.3 Numerical Tests

We tested our approaches on the one-to-one assignment problem with a number n of agents varying from 50 to 300. Given an assignment $\mathbf{x}$, the utility $v_i(\mathbf{x})$ is given by the expression $\sum_{j=1}^{n} v_{ij} x_{ij}$ where $x_{ij} = 1$ if agent i receives object j (0 otherwise) and v_{ij} is a scalar value giving the utility that agent i gets by receiving object j. For this problem, program $\mathscr{P}_{\mathrm{CP}}$ reads as follows (CPA for Cutting Plane for Assignment):

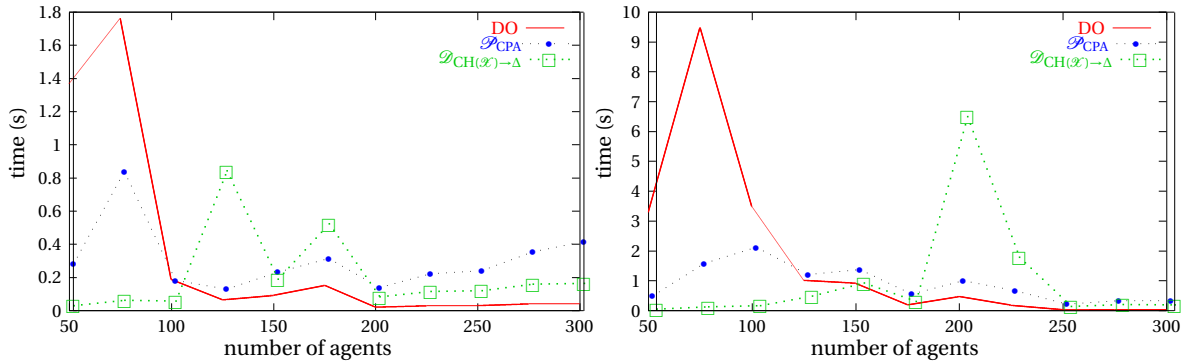
$$\mathscr{P}_{\mathrm{CPA}} \begin{cases} \max\limits_{v, x_{11}, \ldots, x_{nn}} v & \\ v \leq \sum\limits_{i=1}^{n} p_{\boldsymbol{\sigma}(i)} \sum\limits_{j=1}^{n} x_{ij} v_{ij} & \forall \boldsymbol{\sigma} \in \Sigma \\ \sum\limits_{j=1}^{n} x_{ij} = 1 & \forall i \in \{1, \ldots, n\} \\ \sum\limits_{i=1}^{n} x_{ij} = 1 & \forall j \in \{1, \ldots, n\} \\ x_{ij} \geq 0 & \forall i, j \in \{1, \ldots, n\} \end{cases}$$

Both the separation oracle for $\mathscr{D}_{\mathrm{CH}(\mathscr{X})\to\Delta}$ and the procedure $\mathscr{O}_{\mathbf{x}}(\cdot)$ in the double oracle algorithm require to solve a standard assignment problem, which can be performed in polynomial time using the Hungarian method [Kuhn, 1955].

Performances in computation times. We compare the solution times of program $\mathscr{P}_{\mathrm{CPA}}$, program $\mathscr{D}_{\mathrm{CH}(\mathscr{X})\to\Delta}$ and the Double Oracle algorithm (abbreviated by DO in the figures). For this purpose, we carried out numerical tests[2] on randomly generated instances of increasing sizes. The weights w_i used are the ones defining the Gini social-evaluation function [Moulin, 1991]:

$$\forall i \in \{1,\dots,n\}, \quad w_i = (2(n-i)+1)/n^2$$

and the utilities v_{ij} are uniformly drawn as positive integer values in $[1,\mathrm{V}]$. In Figure 7.2 (a), V is set to 20 while in Figure 7.2 (b) it is set to 30. For all values of n and V considered, the average computation times are less than 10 seconds, which shows the practicality of the methods. On both figures, we observe that the computation time first increases, reaches a peak and then decreases. This is due to a *phase transition* phenomenon: if $n \gg \mathrm{V}$ it is likely that there will exist a pure assignment $\mathbf{x}$ such that $v_i(\mathbf{x}) = \mathrm{V}$ for all agents i. In that case, the three methods $\mathscr{P}_{\mathrm{CPA}}$, $\mathscr{D}_{\mathrm{CH}(\mathscr{X})\to\Delta}$ and DO will converge in very few iterations. In both figures, we observe that the sequence of the peaks is identical: DO first, $\mathscr{P}_{\mathrm{CPA}}$ second, and $\mathscr{D}_{\mathrm{CH}(\mathscr{X})\to\Delta}$ third. Note that DO and $\mathscr{D}_{\mathrm{CH}(\mathscr{X})\to\Delta}$ seem to be more affected by an increase of V than $\mathscr{P}_{\mathrm{CPA}}$. This can be explained by the fact that both algorithms rely on an oracle method (the Hungarian method) which is more impacted by an increase of V than the oracle used in $\mathscr{P}_{\mathrm{CPA}}$ which is simply a sorting algorithm. Lastly, the graphs clearly show that the choice of the algorithm to use should be made according to the ratio n/V. If this ratio is low (less than 5 for instance), one should prefer the combination of $\mathscr{P}_{\mathrm{CPA}}$ and $\mathscr{D}_{\mathrm{CH}(\mathscr{X})\to\Delta}$, while if it is higher, DO will be more efficient.



(a) Costs are generated in $[1,20]$. (b) Costs are generated in $[1,30]$.

Figure 7.2: Computation time (in seconds) as n increases. Results averaged on 20 instances.

Let us summarize this section. We tackled the randomized version of fair multi-agent optimization problems with an OWA with decreasing weights as optimization criterion. Thanks to a game-theoretic view of this problem, two solution methods based on dynamic calls to oracles were proposed. The first method we studied is a cutting plane method and the second method we studied is a double oracle method. The numerical

[2] All methods were coded in C++ using Gurobi 5.6.3 as solver for the linear programs. Times are wall-clock times on a 2.4 GHz Intel Core i5 machine with 8GB of RAM.

tests carried out show the practicality of both methods.

In the next section, we study deterministic fair optimization with another aggregation operator that can satisfy the Pigou-Dalton transfer principle, namely the mixture operator.

7.3 Fair Optimization with the MO

In this section, we study the optimization of a Mixture Operator (MO) in two well known types of multi-agent decision problems, namely allocation problems and proportional representation problems. This class of aggregation operators (which is presented in subsection 1.2.1) makes it possible to determine fair (with respect to the Pigou-Dalton transfer principle) and efficient (i.e., Pareto optimal) solutions. We recall that MOs resemble OWA operators (studied in the previous section) but their weights depend on the values that are at stake in the evaluated vector and not on the ranks of the components (see Chapter 1 for a comparison of MOs with OWAs). Optimizing MOs in both allocation and proportional representation problems will reveal NP-hard in general. However, interestingly, we will see that the optimization of some sub-classes of problems will reveal of polynomial complexity, as one-to-one allocation problems (also called *assignment problems*) or proportional representation problems in Chamberlin-Courant's voting scheme with single-peaked preferences.

These results are presented in the two following subsections. While the first one is devoted to allocation problems, the second one is devoted to proportional representation problems.

7.3.1 Optimization of MOs in Allocation Problems

This subsection deals with allocation problems in which a set of m indivisible objects must be assigned to a set of n agents. In general allocation problems, objects can be shared by several agents and each agent can receive several objects. This setting generalizes the assignment problem, where $m = n$ and where each agent should receive exactly one object. The applications of those problems are numerous. While a typical example of general allocation problem is given by the paper assignment problem (that consist in assigning reviewers to conference paper submissions [Goldsmith and Sloan, 2007; Nguyen *et al.*, 2016]), examples of applications for the one-to-one allocation problem (also called *assignment problems*) include the assignment of jobs to workers [Caron *et al.*, 1999] or the assignment of time slots to students in a university [de Werra, 1985].

The hunt for an optimization criterion that could enable to efficiently compute a fair and efficient allocation while being able to account for a wide scope of preferences has led to the investigation of many aggregation operators and of their computational properties. Examples of operators which have been investigated for allocation problems include the min operator [Bouveret *et al.*, 2005], the leximin operator [Garg *et al.*, 2010], the OWA operator [Nguyen *et al.*, 2016] and the Choquet integral [Lesca and Perny, 2010].

In this subsection, we investigate the use of MOs to solve allocation problems.

Allocation Problems. A general allocation problem is defined via the following constraints. For each agent $i \in \{1,\dots,n\}$, the number of items that can be allocated to her is an integer restricted to an interval $[\alpha_i, \beta_i]$. Similarly, for each object $j \in \{1,\dots,m\}$, the number of agents that can share object j is an integer restricted to an interval $[\gamma_j, \delta_j]$. A solution of

the problem is an allocation of the m objects to the n agents that respects the constraints induced by values α_i, β_i, γ_j and δ_j for $i \in \{1,\dots,n\}$ and $j \in \{1,\dots,m\}$. The special case where $m = n$, $\alpha_i = \beta_i = 1$ for each agent i and $\gamma_j = \delta_j = 1$ for each object j corresponds to the *one-to-one allocation problem*, in which each agent receives exactly one object. We will denote by $\mathbf{a}$ a feasible allocation[3] and by $\mathscr{A}$ the set of all feasible allocations.

Each agent has preferences over objects. This information is encapsulated by nm utility values v_{ij} that represent the level of satisfaction of agent i if she gets object j in the allocation. Furthermore, for each agent i, her utility v_i is supposed to be additive. More formally, if agent i receives a set of objects $S \subseteq \{1,\dots,m\}$ in allocation $\mathbf{a}$, then $v_i(\mathbf{a}) = \sum_{j\in S} v_{ij}$. We assume that all utility values, v_i, belong to some open interval $D \subset \mathbb{R}^+$ and we denote by $\mathbf{v}(\mathbf{a}) = (v_1(\mathbf{a}),\dots,v_n(\mathbf{a})) \in D^n$ the vector giving the utilities of each agent for allocation $\mathbf{a}$. A possible allocation $\mathbf{a}_1$ will then be preferred to another allocation $\mathbf{a}_2$ if, globally, $\mathbf{a}_1$ satisfies more the agents than $\mathbf{a}_2$. The goal of the problem is to find a solution that maximizes the overall satisfaction of the agents. This can be formalized by the definition of a criterion to maximize.

Here we assume that this criterion is a mixture operator $M_w : D^n \to \mathbb{R}$. More formally, an allocation $\mathbf{a}_1$ is preferred to an allocation $\mathbf{a}_2$ if $M_w(\mathbf{v}(\mathbf{a}_1)) \geq M_w(\mathbf{v}(\mathbf{a}_2))$. The problem of finding an optimal allocation is then written as follows:

$$\max_{\mathbf{a}\in\mathscr{A}} \quad M_w(\mathbf{v}(\mathbf{a})) = \sum_{i=1}^{n} \frac{w(v_i(\mathbf{a}))}{\sum_{j=1}^{n} w(v_j(\mathbf{a}))} v_i(\mathbf{a}) \tag{7.9}$$

We recall that the optimization of the MO will lead to a fair and efficient allocation provided that function $x \mapsto w(x)(x-y)$ is strictly increasing and strictly concave for all y in D (see subsection 1.2.1 and section 1.3).

We now investigate the complexity of solving allocation problems with mixture operators. While optimizing an MO in general allocation problems will reveal NP-hard, we will see that it becomes polynomial in assignment problems.

Complexity Results and Solution Methods. General allocation problems with fair aggregation operators (i.e., aggregation operators compatible with the Pigou-Dalton transfer principle) are in general hard to solve and they reveal NP-hard most of the time. Unsurprisingly, allocation problems with mixture operators do not depart from this rule.

Proposition 18. *In general allocation problems, the problem $\mathscr{P}_\alpha$ consisting in deciding whether there exists an assignment $\mathbf{a}$ with value $M_w(\mathbf{v}(\mathbf{a})) \geq \alpha$ is an NP-complete decision problem for any fixed positive α.*

Proof. The proof is similar to the one given by Golden and Perny [2010] to prove the NP-completeness of allocation problems with a fair OWA operator. Problem $\mathscr{P}_\alpha$ is clearly in NP. To establish NP-completeness, we reduce the NP-complete *Partition Problem* to our problem. The Partition Problem is stated as follows:

Instance: finite set $O = \{o_1,\dots,o_m\}$ of objects and a size $s(o) \in \mathbb{N}$ for each $o \in O$.

Question: is it possible to partition O into two sets of objects of equal weights?

From an instance of Partition Problem, we construct in polynomial time an instance of $\mathscr{P}_\alpha$ with $n = 2$, $\alpha_1 = \alpha_2 = 0$, $\beta_1 = \beta_2 = m$ (there is no constraints on the number of

[3] **a** for **a**llocation.

objects an agent can receive), $\delta_j = \gamma_j = 1$ (the objects cannot be shared), and $v_{1j} = v_{2j} = s(o_j), \forall j \in \{1,\dots,m\}$. Moreover we set $\alpha = \sum_{o \in O} s(o)/2$ and $w(x) = 4\alpha - x$ which defines a monotone increasing fair mixture operator $M_w(\cdot)$. Hence, the answer to $\mathscr{P}_\alpha$ is YES if and only if the answer to the partition problem is YES. Indeed, if there is a solution to the partition problem, then there exists an assignment $\mathbf{a}$ with $\mathbf{v}(\mathbf{a}) = (\alpha, \alpha)$ and $M_w(\mathbf{v}(\mathbf{a})) = \alpha$. Moreover, if the answer to the partition problem is NO, then any partition of O into two subsets is unbalanced and corresponds to an assignment $\mathbf{a}$ with $\mathbf{v}(\mathbf{a}) = (\alpha - \epsilon, \alpha + \epsilon)$, $\epsilon \in (0, \alpha]$. As $M_w(\cdot)$ is fair, we have $M_w(\mathbf{v}(\mathbf{a})) < \alpha$. So there is no assignment $\mathbf{a}$ such that $M_w(\mathbf{v}(\mathbf{a})) = \alpha$; the answer to $\mathscr{P}_\alpha$ is NO. □

Nevertheless, as stated earlier, optimizing an MO in an assignment problem is a problem of polynomial complexity. To see this, let us denote by $u(\cdot)$ the function defined on $\mathtt{D}$ by $u(x) = w(x)x$. By abuse of notation, given a vector $\mathbf{x} \in \mathtt{D}^n$, we denote by $u(\mathbf{x})$ the sum $\sum_{i=1}^n u(x_i)$ and by $w(\mathbf{x})$ the sum $\sum_{i=1}^n w(x_i)$. By using these notations, the definition of a mixture operator $M_w(\cdot)$ can be rewritten as follows:

$$\forall \mathbf{x} \in \mathtt{D}^n, M_w(\mathbf{x}) = \frac{\sum_{i=1}^n u(x_i)}{\sum_{j=1}^n w(x_j)} = \frac{u(\mathbf{x})}{w(\mathbf{x})}$$

Therefore, the mixture operator can be seen as the ratio of two linear objective functions. More precisely, in the case of the assignment problem the objective function can be rewritten as[4]:

$$M_w(\mathbf{v}(\mathbf{a})) = \frac{\sum_{i=1}^n u(v_{ij}) x_{ij}^{\mathbf{a}}}{\sum_{j=1}^n w(v_{ij}) x_{ij}^{\mathbf{a}}}$$

where binary variable $x_{ij}^{\mathbf{a}}$ equals 1 if agent i receives object j in assignment $\mathbf{a}$ and 0 otherwise. Thus, maximizing $M_w(\mathbf{v}(\mathbf{a}))$ corresponds to the following mathematical program:

$$\begin{aligned}
\max_{x_{ij}:(i,j)\in[|1,n|]^2} & \frac{\sum_{i=1}^n u(v_{ij}) x_{ij}}{\sum_{j=1}^n w(v_{ij}) x_{ij}} \\
\sum_{j=1}^n x_{ij} &= 1 \qquad \forall i \in \{1,\dots,n\} \\
\sum_{i=1}^n x_{ij} &= 1 \qquad \forall j \in \{1,\dots,n\} \\
x_{ij} &\in \{0,1\} \qquad \forall i,j \in \{1,\dots,n\}
\end{aligned}$$

which is a fractional version of the standard assignment problem. Hence, an optimal solution of this problem can be determined by using the fractional programming methods presented in subsection 3.2 (on page 90), namely Megiddo's method and Dinkelbach's method[5], where the oracle amounts to solve a standard assignment problem (i.e., maximizing the sum of utilities of the agents) with utilities of the form $\tilde{v}_{ij}^{\lambda} = u(v_{ij}) - \lambda w(v_{ij})$. This oracle can be implemented by using the Hungarian algorithm [Kuhn, 1955]. Megiddo's method for the Hungarian algorithm relies on the redefined *sum*, *comparison* and *identification of a zero* operations described on page 90. Megiddo's method and

[4]Note that, in general allocation problems, the MO cannot be written in this fashion as functions u and w would be applied to a sum of utilities

[5]Note that there exists other fractional programming methods which are dedicated to the fractional assignment problem [Kabadi and Punnen, 2008; Shigeno *et al.*, 1995].

Dinkelbach's method are polynomial time methods provided they rely on a polynomial time oracle. As the Hungarian algorithm is polynomial time, the complexity of solving the resulting fractional programming problem is also polynomial.

We now turn to the numerical tests in which the efficiency of the two polynomial methods, based on Megiddo's method and Dinkelbach's method (see subsection 3.2), is tested for solving assignment problems with mixture operators.

Numerical Tests. We compare the execution times[6] of both methods (denoted by DIN for Dinkelbach and MEG for Megiddo) with a number n of agents varying from 50 to 400. In both methods, the oracle $\mathscr{O}$ implements the Hungarian algorithm [Kuhn, 1955]. To show the impact of conflicts in the preferences of the agents, for each instance with n agents, the utilities v_{ij} were generated in two different ways:

- In a first experiment, the utilities v_{ij} were randomly generated as positive integers in $[1, n]$. Those utility values are said to be decorrelated.
- In a second experiment, for each object j, a utility value $v(j)$ was randomly generated as a positive integers in $[1, \lfloor 0.8n \rfloor]$. The meaning of $v(j)$ is that all agents agree on the fact that object j has a utility value close to $v(j)$. The utilities v_{ij} were then randomly generated as positive integers in $[v(j), v(j) + \lfloor 0.2n \rfloor]$. Those utility values are said to be correlated.

The weight function used is $w(x) = 2n - x$. As argued in Section 1.3 on page 25, this weight function enables to have a fair monotone increasing mixture operator.

The average computation times (in seconds) as well as the number of calls to the oracle $\mathscr{O}$ are given in Table 7.2 for instances with decorrelated utility values and in Table 7.3 for instances with correlated utility values. We observe that both algorithms are practical and that DIN goes much faster than algorithm MEG with both types of utility values. This is easily explained by the fact that algorithm MEG requires more calls to the oracle $\mathscr{O}$ than algorithm DIN in both cases. Lastly, for both methods, instances with more conflicting preferences are harder to solve, as can be seen by the raise in computation times from Table 7.2 to Table 7.3.

n		50	100	150	200	250	300	350	400
DIN	time	0.006	0.030	0.074	0.154	0.261	0.394	0.590	0.879
	nbc	2.68	2.84	2.88	2.98	3.00	3.00	3.00	3.00
MEG	time	0.013	0.067	0.163	0.337	0.583	0.896	1.415	2.022
	nbc	5.02	5.68	5.90	5.90	6.14	6.22	6.58	6.18

Table 7.2: Evolution of the average computation times in seconds and of the average number of calls to algorithm $\mathscr{O}$ (nbc in the table) for algorithms DIN and MEG as n increases. Instances are generated with decorrelated utility values.

To sum up, we have seen that while solving general allocation problems with MOs is NP-hard, the assignment problem can be solved in polynomial time with fractional programming methods. In our computational tests, Megiddo's method and Dinkelbach's method revealed efficient, solving instances with up to 400 agents in less than 3 seconds

[6] All methods were implemented in C++. Times are wall-clock times on a 2,4 GHz Intel Core i5 machine with 8GB of RAM. All results are averaged over 50 runs.

n		50	100	150	200	250	300	350	400
DIN	time	0.007	0.054	0.170	0.495	0.962	1.706	2.943	4.497
	nbc	2.12	2.16	2.20	2.50	2.64	2.62	2.70	2.74
MEG	time	0.030	0.299	1.202	3.510	6.509	13.759	35.742	40.789
	nbc	7.34	10.28	12.76	14.26	15.02	17.46	25.58	20.28

Table 7.3: Evolution of the average computation times in seconds and of the average number of calls to algorithm $\mathcal{O}$ (nbc in the table) for algorithms DIN and MEG as n increases. Instances are generated with correlated utility values

(resp. 41 seconds) for instances with decorrelated utility values (resp. correlated utility values).

We now turn to another multi-agent optimization problem that requires fairness, namely the proportional representation problem.

7.3.2 Optimization of MOs in Proportional Representation Problems

Proportional Representation (PR) problems deal with multi-winner voting rules where one aims to elect a subset of candidates rather than a single one. In multi-winner election rules, a set of voters express preferences over a set of candidates. The objective is then to determine k winning candidates, each candidate representing a subset of voters, such that each voter is satisfied by the winning candidate that represents her. Multi-winner election rules are important for both political elections (i.e., electing committees of representatives) and multi-agent recommendation systems (e.g., choosing a set of dinning menus for a conference) [Elkind *et al.*, 2014; Skowron *et al.*, 2016].

Two main multi-winner voting schemes have been designed for PR problems, namely *Monroe*'s Voting Scheme [Monroe, 1995] (abbreviated by MVS) and *Chamberlin-Courant*'s Voting Scheme [Chamberlin and Courant, 1983] (abbreviated by CCVS). In these two frameworks, a feasible solution is characterized by a set of k winning candidates and an assignment from winning candidates to voters. Each voter is then represented by the elected candidate assigned to her. While CCVS does not constraint the possible assignments from winning candidates to voters, MVS imposes that the k sets consisting of the voters represented by the same candidate should be equally sized. The choice of a solution is then based on the ballots, where each voter ranks the candidates from best to worst. Indeed, the solution chosen should maximize the utilities of the voters, where the utility (i.e., satisfaction level) of a voter depends on the rank she gave to the candidate assigned to her. In the *utilitarian version* of MVS and CCVS, the goal is to find a solution maximizing the sum of voter's utilities. However maximizing the sum of utilities can yield an unfair solution as it compensates between the utilities of the different voters. Thus, several alternative optimization criteria have been investigated for multi-winner voting rules to address this problem. In CCVS, Betzler *et al.* [2013] proposed to maximize the utility of the least happy voter (the one with the minimum utility among the voters). This is known as the *egalitarian version* of CCVS. However, maximizing the minimum utility value of the voters can be considered extreme as it does not take into account the satisfaction of all but one voter. To address this issue, Elkind and Ismaili [2015] extended this approach to Ordered Weighted Averages (OWAs) of utilities, which enable to smoothly interpolate between the egalitarian version and the utilitarian version of CCVS.

In this subsection, we investigate the use of mixture operators to find an efficient (i.e., Pareto optimal) and fair (with respect to the Pigou-Dalton transfer principle) solution in

CCVS and MVS.

Regarding the complexity of PR problems, Procaccia *et al.* [2008] proved that winner determination under the utilitarian versions of MVS and CCVS are both NP-hard problems even if the utility values are based on approval ballots (i.e., where voters attribute binary values to candidates, a value of 1 meaning that the voter approves the candidate). Similarly, for the egalitarian version of CCVS, Betzler *et al.* [2013] proved that winner determination is NP-hard. More positive results where obtained by resorting to approximation algorithms or special structures of preferences. Approximation algorithms with performance guarantees for PR problems were given by Lu and Boutilier [2011] and Skowron *et al.* [2015]. Betzler *et al.* [2013] showed that, for single-peaked preferences, winner determination in CCVS is a polynomial time problem by designing a dynamic programming solution method. In a nutshell, a preference profile is single-peaked if there exists an axis on which the candidates and voters can be placed such that the preferences of each voter are defined by the distances between her and the different candidates, the closer the better [Black, 1948]. Their results where extended by Cornaz *et al.* [2012] for preferences "close" to be single-peaked (in a sense formalized in their paper). Another well-known structure of preferences in social choice is single-crossing preferences. In a nutshell, a preference profile is single-crossing if the voters can be ordered such that for each pair of candidates the voters can be separated such that the voters on each side of the separation prefer the same candidate. Skowron *et al.* [2013] showed that, for single-crossing preferences, winner determination in CCVS is also a polynomial time problem. These results where extended to some extent to OWA operators by Elkind and Ismaili [2015]. The authors investigated several classes of weights $\mathbf{w}$ defining families of OWA operators that enable to obtain a compromise between the utilitarian and the egalitarian versions of PR problems in CCVS. Interestingly, they showed that if the preferences of the voters present particular structures (e.g., single-crossing, single-peaked), then it becomes possible to design polynomial or pseudo-polynomial solution methods for some of these OWA families.

We now investigate the complexity of solving PR problems under CCVS and MVS with an MO as optimization criterion and show that we obtain the same complexity results as in the utilitarian case. More precisely, we prove that for single-peaked or single-crossing preferences, optimizing MOs in PR problems are polynomial problems.

Proportional Representation Problems. Let $\mathrm{V} = \{1,\ldots,n\}$ be a set of n voters and $\mathrm{C} = \{1,\ldots,m\}$ be a set of m candidates. A solution of the PR problem is a set of k winning candidates $\{c_1,\ldots,c_k\} \subseteq \mathrm{C}$, and k sets S_j ($j \in \{c_1,\ldots,c_k\}$), where S_j is the subset of voters represented by candidate j. We recall that, while in MVS the sets S_j should be of the same size, it is not the case in CCVS. Consequently, in CCVS, a voter is always represented by the candidate she likes most in the set of k winning candidates. We will denote by $\mathbf{e} = \{c_1, \mathrm{S}_{c_1}; \ldots; c_k, \mathrm{S}_{c_k}\}$ a feasible solution[7] and by $\mathscr{E}$ the set of all feasible solutions.

Each voter has preferences over candidates. These preferences are expressed by a *preference profile* P of size mn, where the k^{th} column of P is the preference order of voter k. From profile P, nm utility values v_{ij} are derived that represent the level of satisfaction of voter i if she is represented by candidate j. Given a feasible solution $\mathbf{e}$, the utility $v_i(\mathbf{e})$ of voter i is then given by v_{ij} if i belongs to S_j in $\mathbf{e}$. The PR problem aims to determine a solution $\mathbf{e} \in \mathscr{E}$ such that the utilities of the voters are maximized. We assume that all utility values v_i belong to some open interval $\mathrm{D} \subset \mathbb{R}^+$ and we denote by $\mathbf{v}(\mathbf{e}) = (v_1(\mathbf{e}),\ldots,v_n(\mathbf{e})) \in \mathrm{D}^n$ the vector giving the utilities of each voter for solution $\mathbf{e}$. A solution $\mathbf{e}_1$ will then be pre-

[7] **e** for **e**lected committee.

ferred to another solution $\mathbf{e}_2$ if, globally, $\mathbf{e}_1$ satisfies more the voters than $\mathbf{e}_2$. This is formalized by an aggregation criterion to maximize.

Here, we assume that this criterion is given as a mixture operator $\mathrm{M}_w : \mathtt{D}^n \to \mathbb{R}$. More formally, a feasible solution $\mathbf{e}_1$ is preferred to a solution $\mathbf{e}_2$ if $\mathrm{M}_w(\mathbf{v}(\mathbf{e}_1)) \geq \mathrm{M}_w(\mathbf{v}(\mathbf{e}_2))$. The problem of finding an optimal solution is then written as follows:

$$\max_{\mathbf{e}\in\mathscr{E}} \quad \mathrm{M}_w(\mathbf{v}(\mathbf{e})) = \sum_{i=1}^{n} \frac{w(v_i(\mathbf{e}))}{\sum_{j=1}^{n} w(v_j(\mathbf{e}))} v_i(\mathbf{e}) \tag{7.10}$$

where $w : \mathtt{D} \to (0,\infty]$ is a positive weighting function. Again, we recall that the optimization of the MO will lead to a fair and efficient solution provided that function $x \mapsto w(x)(x-y)$ is strictly increasing and strictly concave for all $y \in \mathtt{D}$ (see subsection 1.2.1 and section 1.3). Note that if function w is constant then the MO boils down to the arithmetic mean used in the utilitarian version of PR problems. Furthermore, the special case of the Lehmer mean is defined by $w(x) = x^{p-1}$ where p is a parameter. If p tends towards $-\infty$, then the Lehmer mean tends towards the min operator used in the egalitarian version of PR problems. More generally, we can find compromises between the utilitarian and the egalitarian variants of PR problems by varying the decreasingness of w.

We illustrate the notations in Example 58.

Example 58. *Four friends, denoted by f_1, f_2, f_3 and f_4, living in Linné street are organizing an event and need to cook exactly two dishes. They must decide in which houses they will cook (they all want to cook). Hence, in this problem, there are four voters, four candidates H_1, H_2, H_3 and H_4 where H_i denotes the house of f_i and two winning candidates (i.e., $n = m = 4$ and $k = 2$). The position of each house on Linné street is given in Figure 7.3.*

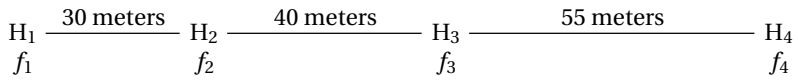


Figure 7.3: The positions of each house on Linné street.

Each f_i likes to cook and hates walking. Thus, they would like the house assigned to them to be as close as possible to their house. The preference profile in this example is shown in Table 7.4 (for instance the first column states that for f_1, $\mathrm{H}_1 \succ \mathrm{H}_2 \succ \mathrm{H}_3 \succ \mathrm{H}_4$).

f_1	f_2	f_3	f_4
H_1	H_2	H_3	H_4
H_2	H_1	H_2	H_3
H_3	H_3	H_4	H_2
H_4	H_4	H_1	H_1

Table 7.4: Preference profile of the four voters.

From this preference profile $n \times m$ utility values v_{ij} can be computed by using a satisfaction function v as $v(c) = 5 - \mathtt{rk}(c)$ where $\mathtt{rk}$ is the rank of candidate c. The v_{ij} values according to this satisfaction function are given in Table 7.5.

In this example, in the CCVS, there is no restriction on the number of people that can be assigned to a house. However, the four friends might not accept that one of them is left alone. Therefore, they might decide to use the MVS in which each chosen house is assigned

	f_1	f_2	f_3	f_4
H_1	4	3	1	1
H_2	3	4	3	2
H_3	2	2	4	3
H_4	1	1	2	4

Table 7.5: Utility values for the proportional representation problem of Example 58.

two people. Our goal in this section is to find a best solution according to an MO. For instance, if $w(x) = 8 - x$ *and* $\mathbf{e} = \{H_2, \{f_1, f_2\}, H_3, \{f_3, f_4\}\}$*, then the value of solution* $\mathbf{e}$ *is:*

$$\begin{aligned} M_w(\mathbf{v}(\mathbf{e})) &= M_w(3,4,4,3) \\ &= \frac{3w(3)}{2w(3)+2w(4)} + \frac{4w(4)}{2w(3)+2w(4)} + \frac{4w(4)}{2w(3)+2w(4)} + \frac{3w(3)}{2w(3)+2w(4)} \\ &= 2\frac{3\times 5}{10+8} + 2\frac{4\times 4}{10+8} = \frac{31}{9} \end{aligned}$$

and we aim to find the solution with maximal value.

We now investigate the complexity of winner determination in multi-winner voting rules with fair mixture operators.

Complexity and Solution Methods. Let u denote the function defined on $\mathtt{D}$ by $u(x) = w(x)x$. By abuse of notation, given a vector $\mathbf{x} \in \mathtt{D}^n$, we denote by $u(\mathbf{x})$ the sum $\sum_{i=1}^n u(x_i)$ and by $w(\mathbf{x})$ the sum $\sum_{i=1}^n w(x_i)$. As we have already seen, by using these notations, the definition of an MO $M_w(\cdot)$ can be rewritten as follows:

$$\forall \mathbf{x} \in \mathtt{D}^n, M_w(\mathbf{x}) = \frac{\sum_{i=1}^n u(x_i)}{\sum_{j=1}^n w(x_j)} = \frac{u(\mathbf{x})}{w(\mathbf{x})} \tag{7.11}$$

Fractional programming (presented in subsection 3.2) is dedicated to this type of objective functions. We will now adapt and present three fractional programming methods for PR problems. The first one relies on a linearization trick that we will use to design a mixed integer linear program to optimize an MO in a PR problem. The second and third ones are the two parametric methods presented in subsection 3.2 that we will use to design polynomial time algorithms to solve PR problems with an MO criterion when a special structure of preferences enables to solve the utilitarian version of the PR problem in polynomial time.

A Mixed Integer Linear Program. Note that if function w is constant, then one recovers the utilitarian version of PR problems which is NP-hard. Thus, it follows that PR problems under CCVS or MVS with an MO as decision criterion are also NP-hard. Yet, the NP-hardness of winner determination under CCVS and MVS has not prevented researchers from designing solution procedures for these problems. Indeed, Potthoff and Brams [1998] investigated the use of an integer linear program to solve PR problems. We denote by $\mathscr{IP}$ the integer program they proposed. Program $\mathscr{IP}$ is given below. It includes nm binary variables x_{ij}, where x_{ij} takes value 1 if voter i is represented by candidate j, and m binary variables z_j, where z_j takes value 1 if candidate j is a winning candidate. Constraint (7.12) ensures that the election has k winning candidates. The n constraints in (7.13) makes sure that each voter is only represented by one candidate. Lastly, constraints (7.14) and (7.15) specify a lower bound L and an upper bound U on the

number of voters that can be represented by the same candidate. The pair (L, U) in MVS (resp. CCVS) is equal to $(\lfloor n/k \rfloor, \lceil n/k \rceil)$ (resp. $(0, n)$).

$$\mathscr{IP} \begin{cases} \max \sum_{i,j \in \mathrm{V} \cdot \mathrm{C}} v_{ij} x_{ij} & \\ \sum_{j \in \mathrm{C}} z_j = k & (7.12) \\ \sum_{j \in \mathrm{C}} x_{ij} = 1, \quad \forall i \in \mathrm{V} & (7.13) \\ \sum_{i \in \mathrm{V}} x_{ij} \geq \mathrm{L} z_j, \quad \forall j \in \mathrm{C} & (7.14) \\ \sum_{i \in \mathrm{V}} x_{ij} \leq \mathrm{U} z_j, \quad \forall j \in \mathrm{C} & (7.15) \\ z_j \in \{0,1\}, \quad \forall j \in \mathrm{C} & \\ x_{ij} \in \{0,1\}, \quad \forall i,j \in \mathrm{V} \times \mathrm{C} & \end{cases}$$

This program can be adapted to obtain a Mixed Integer Linear Program (MILP) $\mathscr{MIP}^{\mathrm{MO}}$ (given below) to solve a PR problem according to an MO. We now describe how this program has been obtained. With the criterion M_w, the objective function becomes:

$$\frac{\sum_{i,j \in \mathrm{V} \cdot \mathrm{C}} u(v_{ij}) x_{ij}}{\sum_{i,j \in \mathrm{V} \cdot \mathrm{C}} w(v_{ij}) x_{ij}} \tag{7.16}$$

To linearize this objective function, we use a general method proposed by Williams [1974]. It consists in introducing a continuous variable λ into the problem to represent the expression given in Equation 7.16. The objective is then to maximize this variable. By definition of λ, the following condition should hold:

$$\sum_{i,j \in \mathrm{V} \cdot \mathrm{C}} w(v_{ij}) \lambda x_{ij} - \sum_{i,j \in \mathrm{V} \cdot \mathrm{C}} u(v_{ij}) x_{ij} = 0$$

However, note that this equation is not linear in the variables of the problem because of the quadratic terms λx_{ij}. Therefore, to enforce this equation in program $\mathscr{MIP}^{\mathrm{MO}}$, one introduces nm continuous variables y_{ij} taking values in $\mathbb{R}^+$ to replace expressions λx_{ij}. To make sure that $y_{ij} = \lambda x_{ij}$, one imposes the following constraints:

$$y_{ij} \leq x_{ij} \lambda^u, \qquad \forall i,j \in \mathrm{V} \cdot \mathrm{C} \tag{7.17}$$

$$\sum_{j \in \mathrm{C}} y_{ij} = \lambda, \qquad \forall i \in \mathrm{V} \tag{7.18}$$

where λ^u denotes an upper bound on λ. Such an upper bound can easily be obtained by computing $(\max_{i,j \in \mathrm{V} \times \mathrm{C}} v_{ij})$[8]. While Equation 7.17 ensures that $y_{ij} = 0$ if $x_{ij} = 0$, Equation 7.18 ensures that $y_{ij} = \lambda$ if $x_{ij} = 1$. Indeed, in Equation 7.18, only one of the y_{ij} is not zero due to constraints 7.13 in program $\mathscr{IP}$ and Equation 7.18 imposes that this variable equals λ. The final program $\mathscr{MIP}^{\mathrm{MO}}$ involves $nm+1$ additional continuous variables

[8] $(\max_{i,j \in \mathrm{V} \times \mathrm{C}} v_{ij})$ is an upper bound on λ because an MO has an averaging behavior.

and $nm+n+1$ additional constraints.

$$\mathscr{MIP}^{\mathrm{MO}}\begin{cases} \max \lambda \\ \sum_{j\in \mathrm{C}} z_j = k \\ \sum_{j\in \mathrm{C}} x_{ij} = 1, \quad \forall i \in \mathrm{V} \\ \sum_{i\in \mathrm{V}} x_{ij} \geq \mathrm{L} z_j, \quad \forall j \in \mathrm{C} \\ \sum_{i\in \mathrm{V}} x_{ij} \leq \mathrm{U} z_j, \quad \forall j \in \mathrm{C} \\ \sum_{i,j\in \mathrm{V}\cdot\mathrm{C}} (w(v_{ij})y_{ij} - u(v_{ij})x_{ij}) = 0 \\ \sum_{j\in \mathrm{C}} y_{ij} = \lambda, \quad \forall i \in \mathrm{V} \\ y_{ij} \leq x_{ij}\lambda^u, \quad \forall i,j \in \mathrm{V}\times\mathrm{C} \\ z_j \in \{0,1\}, \quad \forall j \in \mathrm{C} \\ x_{ij} \in \{0,1\}, \quad \forall i,j \in \mathrm{V}\times\mathrm{C} \\ y_{ij} \in \mathbb{R}^+, \quad \forall i,j \in \mathrm{V}\times\mathrm{C} \\ \lambda \in \mathbb{R} \end{cases}$$

We illustrate the notations of this MILP in Example 59.

Example 59. *We come back to the proportional representation problem of Example 58. We decide to use the MVS to solve this representation problem. Therefore, in the program above* L *and* U *are set to 2. Furthermore, we decide to use as optimization criterion the MO with weighting function* $w(x) = 8 - x$*. Then, an optimal solution is* $\mathbf{e} = \{\mathrm{H}_2, \{f_1, f_2\}, \mathrm{H}_3, \{f_3, f_4\}\}$*, which corresponds to* $\lambda = \frac{31}{9}$ *(the optimal value of the problem),* $z_2 = z_3 = 1$ *(the elected candidates are* H_2 *and* H_3*), and* $y_{12} = y_{22} = y_{33} = y_{43} = \lambda = \frac{31}{9}$*,* $x_{12} = x_{22} = x_{33} = x_{43} = 1$ *(voters 1 and 2 are represented by candidate 2 and voters 3 and 4 are represented by candidate 3). All other variables are equal to 0.*

This approach is of course not polynomial but has the advantage of its generality.

We now show that if the preferences of the agents are single-peaked or single-crossing, then optimizing an MO is polynomial. In these cases, we can design more efficient approaches as will be shown by the numerical tests.

A Parametric Approach. We now restrict our attention to preference profiles that are either single-peaked or single-crossing.

Definition 45. *A preference profile is single-peaked with respect to an ordering of candidates* $c_1 < c_2 < \ldots < c_m$ *(also called axis) if for each voter* v *there exists a candidate* c^* *such that: i)* v *ranks* c^* *first, ii) if* $c^* < c < c'$*, then* v *prefers* c *to* c' *and iii) if* $c < c' < c^*$*, then* v *prefers* c' *to* c*.*

Definition 46. *A preference profile is single-crossing with respect to an ordering of voters* $(v_1, \ldots, v_n)$ *if for each pair of candidates* c_1*,* c_2 *there exists an* $i \in \{0, \ldots, n\}$ *such that voters* $v_1, \ldots, v_i$ *prefer* c_1 *to* c_2*, and voters* $v_{i+1}, \ldots, v_n$ *prefer* c_2 *to* c_1*.*

An example of these types of preferences was given in Example 58. Indeed, it is easy to see that the preference profile in this example is single-peaked; the preferences of each

voter only depend on the distances between the candidates (i.e., the houses) and their ideal candidate (i.e., their home) on an axis (i.e., Linné street), the closer the better. This preference profile is also single-crossing with respect to the ordering of voters on the axis.

We now assume that the preference profile of the voters is consistent with one of this two structures. In this case, it is known that the utilitarian version of the PR problem can be solved with a polynomial time algorithm that we denote by $\mathscr{O}$ [Betzler *et al.*, 2013; Skowron *et al.*, 2013]. Using algorithm $\mathscr{O}$, we can use the two methods (DIN and MEG) presented in subsection 3.2 to obtain two polynomial time methods to solve the PR problem with respect to an MO. Megiddo's method for these problems relies on the redefined *sum* and *comparison* operations described on page 90.

These two parametric methods are of polynomial complexity if an oracle can solve utilitarian versions of the PR problem with utilities of the form $\tilde{v}_{ij}^{\lambda} = u(v_{ij}) - \lambda w(v_{ij})$ where $\lambda \geq 0$ (as the v_{ij} values are positive). For these problems to be of polynomial complexity, these new utility values $\tilde{v}_{ij}^{\lambda}$ need of course to be consistent with the preferences of the agents. Stated differently, if voter i prefers candidate j_1 to candidate j_2 then $\tilde{v}_{ij_1}^{\lambda}$ should be greater than $\tilde{v}_{ij_2}^{\lambda}$. Therefore, function $v_{ij} \mapsto u(v_{ij}) - \lambda w(v_{ij}) = w(v_{ij})(v_{ij} - \lambda)$ should be increasing. Note that this assumption is not restrictive as it is exactly the condition under which an MO is increasing. This assumption holds if w is a decreasing function and u is an increasing function (sufficient condition).

Let us now turn to the numerical tests in which the efficiency of the different solution methods are compared.

Numerical Tests. We now compare the execution times of the proposed solution methods in two different experiments[9]. In both experiments, values v_{ij} are set to $m - \mathtt{rk}_i(j)$ (where $\mathtt{rk}_i(j)$ denotes the rank of candidate j in the preference order of voter i) and the weighting function w of the MO is defined by $w(x) = 2m - x$, so that the transfer principle holds. Lastly, for an election with n voters, values m and k were set respectively to $n/5$ and $n/20$.

In a first experiment, we compare the execution times of the two instantiations of $\mathscr{MIP}^{\mathrm{MO}}$ in the CCVS and MVS frameworks. We denote these programs by $\mathscr{M}^{\mathrm{MO}}_{\mathrm{CCVS}}$ and $\mathscr{M}^{\mathrm{MO}}_{\mathrm{MVS}}$. For each instance, the preferences of the voters were generated uniformly at random. The average computation times (in seconds) over 50 instances are given in Table 7.6 for instances with a number of voters ranging from 50 to 100.

Obviously, we observe that solving these two programs are computationally demanding and that the computation time increases quickly with the the number of voters. For instance, solving programs $\mathscr{M}^{\mathrm{MO}}_{\mathrm{CCVS}}$ and $\mathscr{M}^{\mathrm{MO}}_{\mathrm{MVS}}$ takes less than 2 seconds for elections with 50 voters but more than 25 seconds for elections with 100 voters. However, these MILPs have the advantage of being very general.

In a second experiment, we restrict ourselves to the CCVS framework with randomly generated single-peaked preferences , and we compare the execution times of the two methods presented in subsection 3.2, denoted by DIN (for Dinkelbach) and MEG (for Megiddo). In methods DIN and MEG, the algorithm $\mathscr{O}$ used (see Section 3.2) is the dynamic programming algorithm proposed by Betzler *et al.* [2013]. The average computation times (in seconds) over 50 instances, as well as the average number of calls to algorithm $\mathscr{O}$, are given in Table 7.7 for instances with a number of voters ranging from 100 to

[9] All methods were implemented in C++ using Gurobi version 5.6.3 to solve the LPs. Times are wall-clock times on a 2.4 GHz Intel Core i5 machine with 8GB of RAM.

n	50	60	70	80	90	100
$\mathscr{M}^{\text{MO}}_{\text{CCVS}}$	0.79	1.19	2.22	3.35	13.45	28.52
$\mathscr{M}^{\text{MO}}_{\text{MVS}}$	1.01	1.60	3.30	5.34	14.78	25.66

Table 7.6: Evolution of the average computation time in seconds to solve programs $\mathscr{M}^{\text{MO}}_{\text{CCVS}}$ and $\mathscr{M}^{\text{MO}}_{\text{MVS}}$ as n increases.

1000.

We observe that both methods DIN and MEG are very fast and require very few calls to algorithm $\mathscr{O}$. On these instances, method DIN seems to perform best and requires less calls to algorithm $\mathscr{O}$.

n		100	200	300	400	500	600	700	800	900	1000
DIN	time	<0.001	0.005	0.017	0.037	0.074	0.132	0.276	0.416	0.618	1.064
	nbc	2.62	2.86	2.94	3	2.98	2.98	3	3	3	3
MEG	time	0.001	0.017	0.064	0.171	0.370	0.676	1.46	2.36	3.87	6.98
	nbc	3.42	7.18	8.9	11.32	11.94	12.4	13.34	14.32	16.2	15.04

Table 7.7: Evolution of the average computation time in seconds and the average number of calls to algorithm $\mathscr{O}$ (nbc in the table) for methods DIN and MEG as n increases.

Let us summarizes this subsection. We have studied PR problems with an MO as decision criterion and we saw that we could recover the same complexity results as in the utilitarian case. More precisely, while solving general PR problems with MOs are NP-hard, they can be solved with two polynomial time fractional solution methods if the preferences of the voters abide to a particular structure enabling to solve the utilitarian version of the PR problem in polynomial time. Moreover, we also designed a MILP to solve general PR problems according to an MO criterion. In our computational tests, the two fractional methods based on Megiddo's method and Dinkelbach's method revealed efficient, solving instances with up to 1000 voters in less than 2 seconds and 7 seconds respectively.

7.4 Conclusion

In this chapter, we have tackled multi-agent optimization problems with two aggregation operators as decision criteria, namely Ordered Weighted Averages (OWAs) with decreasing weights and Mixture Operators (MOs). These two types of operators are, under known conditions on their parameters, compatible with the Pigou-Dalton principle. Thus, their optimization makes it possible to find fair solutions in a multi-agent setting.

In a first section, we investigated the randomized version of fair multi-agent optimization problems with an OWA with decreasing weights as decision criterion. Thanks to a game-theoretic view of this problem, two solution methods based on dynamic calls to oracles were developed. The first method is a cutting plane method and the second method is based on a double oracle approach. The numerical tests carried out have shown the practicality of both methods.

In a second section, we studied the resolution of allocation problems and proportional representation problems with an MO as decision criterion. While the optimization of MOs in both problems revealed NP-hard in general, we have seen that optimizing an

MO can be performed in polynomial time in assignment problems and in proportional representation problems where the preferences of the agents abide to some particular structures (single-peaked, single-crossing). We have presented solution methods based on fractional programming for both types of problems. The numerical tests carried out have shown the practicality of the different methods designed in this section.

7.5 References

A. Abdulkadiroğlu and T. Sönmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3):689–701, 1998. 219

H. Aziz, F. Brandt, and P. Stursberg. On popular random assignments. In *Inter. Symp. on Algorithmic Game Theory*, pages 183–194. Springer, 2013. 220

N. Betzler, A. Slinko, and J. Uhlmann. On the computation of fully proportional representation. *Journal of Artificial Intelligence Research*, 47:475–519, 2013. 232, 233, 238

D. Black. On the rationale of group decision-making. *Journal of political economy*, 56(1):23–34, 1948. 233

A. Bogomolnaia and H. Moulin. A new solution to the random assignment problem. *J. of Economic theory*, 100(2):295–328, 2001. 220

S. Bouveret, M. Lemaître, H. Fargier, and J. Lang. Allocation of indivisible goods: a general model and some complexity results. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1309–1310. ACM, 2005. 228

G. Caron, P. Hansen, and B. Jaumard. The assignment problem with seniority and job priority constraints. *Operations Research*, 47(3):449–453, 1999. 228

J.R. Chamberlin and P.N. Courant. Representative deliberations and representative decisions: Proportional representation and the borda rule. *American Political Science Review*, 77(03):718–733, 1983. 232

V. Chvatal. *Linear programming*. Macmillan, 1983. 222

D. Cornaz, L. Galand, and O. Spanjaard. Bounded single-peaked width and proportional representation. In *Proceedings of the European Conference on Artificial Intelligence 2012*, pages 270–275. IOS Press, 2012. 233

D. de Werra. An introduction to timetabling. *European journal of operational research*, 19(2):151–162, 1985. 228

E. Elkind and A. Ismaili. OWA-based extensions of the chamberlin–courant rule. In *Proceedings of the International Conference on Algorithmic Decision Theory 2015, (ADT)*, pages 486–502. Springer, 2015. 232, 233

E. Elkind, P. Faliszewski, P. Skowron, and A. Slinko. Properties of multiwinner voting rules. In *Proceedings AAMAS 2014*, 2014. 232

U. Endriss. Reduction of economic inequality in combinatorial domains. In *Proceedings of AAMAS 2013*, pages 175–182, 2013. 220

J. Fodor, J.L. Marichal, and M. Roubens. Characterization of the ordered weighted averaging operators. *IEEE Trans. on Fuzzy Systems*, 3(2):236–240, 1995. 221

P. Gärdenfors. Match making: assignments based on bilateral preferences. *Systems Research and Behavioral Science*, 20(3):166–173, 1975. 220

N. Garg, T. Kavitha, A. Kumar, K. Mehlhorn, and J. Mestre. Assigning papers to referees. *Algorithmica*, 58(1):119–136, 2010. 228

H. Gilbert. Fair Proportional Representation Problems with Mixture Operators. In *Proceedings of the International Conference on Algorithmic Decision Theory 2017*, 2017. 218

B. Golden and P. Perny. Infinite order lorenz dominance for fair multiagent optimization. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 383–390. International Foundation for Autonomous Agents and Multiagent Systems, 2010. 229

J. Goldsmith and R. H. Sloan. The ai conference paper assignment problem. In *2007 AAAI Workshop*, 2007. 218, 228

M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981. 224

A. Hylland and R. Zeckhauser. The efficient allocation of individuals to positions. *Journal of Political economy*, 87(2):293–314, 1979. 220

S.N. Kabadi and A.P. Punnen. A strongly polynomial simplex method for the linear fractional assignment problem. *Operations Research Letters*, 36(4):402–407, 2008. 230

A-K Katta and J. Sethuraman. A solution to the random assignment problem on the full preference domain. *Journal of Economic Theory*, 131(1):231–250, 2006. 220, 221

T. Kavitha, J. Mestre, and M. Nasre. Popular mixed matchings. *Theoretical Computer Science*, 412(24):2679–2690, 2011. 220

H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. 227, 230, 231

J. Lang and J. Rothe. Fair division of indivisible goods. In *Economics and Computation*, pages 493–550. Springer, 2016. 218

J. Lesca and P. Perny. LP solvable models for multiagent fair allocation problems. In *Proceedings of the European Conference on Artificial Intelligence*, pages 393–398, 2010. 220, 228

T. Lu and C. Boutilier. Budgeted social choice: From consensus to personalized decision making. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 11, pages 280–286, 2011. 233

M.J. Machina. Dynamic consistency and non-expected utility models of choice under uncertainty. *Journal of Economic Literature*, 27(4):1622–1668, 1989. 219

A. Mastin, P. Jaillet, and S. Chin. Randomized minmax regret for combinatorial optimization under uncertainty. In *Int. Symp. on Algorithms and Computation*, pages 491–501. Springer, 2015. 224

B.L. Monroe. Fully proportional representation. *American Political Science Review*, 89(04):925–940, 1995. 232

H. Moulin. *Axioms of cooperative decision making*. Number 15. Cambridge University Press, 1991. 220, 227

J. Nguyen, G. Sánchez-Hernández, N. Agell, X. Rovira, and C. Angulo. An OWA-Based Multi-Criteria System for Assigning Papers to Reviewers. In *Proceedings of the 19th Int. Conf. of the Catalan Association for AI*, pages 253–262, 2016. 218, 228

W. Ogryczak and T. Śliwiński. On solving linear programs with the ordered weighted averaging objective. *European Journal of Operational Research*, 148(1):80–91, 2003. 223

R.F. Potthoff and S.J. Brams. Proportional representation: Broadening the options. *Journal of Theoretical Politics*, 10(2):147–178, 1998. 235

A.D. Procaccia, J.S. Rosenschein, and A. Zohar. On the complexity of achieving proportional representation. *Social Choice and Welfare*, 30(3):353–362, 2008. 218, 233

A. Sen. *On economic inequality*. Oxford University Press, 1973. 220

M. Shigeno, Y. Saruwatari, and T. Matsui. An algorithm for fractional assignment problems. *Discrete Applied Mathematics*, 56(2-3):333–343, 1995. 230

P. Skowron, L. Yu, P. Faliszewski, and E. Elkind. The complexity of fully proportional representation for single-crossing electorates. In *International Symposium on Algorithmic Game Theory*, pages 1–12. Springer, 2013. 233, 238

P. Skowron, P. Faliszewski, and A. Slinko. Achieving fully proportional representation: Approximability results. *Artificial Intelligence*, 222:67–103, 2015. 218, 233

P. Skowron, P. Faliszewski, and J. Lang. Finding a collective set of items: From proportional multirepresentation to group recommendation. *Artificial Intelligence*, 241:191–216, 2016. 218, 232

J. Von Neumann. A certain zero-sum two-person game equivalent to the optimal assignment problem. *Contributions to the Theory of Games*, 2:5–12, 1953. 219

H.P. Williams. Experiments in the formulation of integer programming problems. In *Approaches to Integer Programming*, pages 180–197. Springer, 1974. 236

Conclusion and Future Works

" Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning. "

W. Churchill

In this thesis, we aimed to solve complex decision problems coming from different domains (e.g., sequential decision under risk, robust combinatorial optimization, fair combinatorial optimization), according to sophisticated decision criteria (e.g., skew-symmetric bilinear utilities, minmax regret). Our approach to address these problems was to reformulate them in a convenient way to be able to use powerful oracle methods. This chapter recalls the different contributions of the thesis and gives some research directions for each domain that has been considered.

Contributions in Sequential Decision Making under Risk with SSB Utilities

In Chapter 4, we investigated the resolution of sequential decision problems under risk with the Skew-Symmetric Bilinear (SSB) utility model and the Weighted Expected Utility (WEU) model (a special case of SSB), in decision trees and in finite horizon Markov Decision Processes (MDPs). We worked under the paradigm of resolute choice with root dictatorship. Furthermore, to overcome the intransitivity of the SSB utility model, we used the minmax rule in the tournament induced over the plans.

The first question we wished to answer concerned the properties of the optimal plans. For instance, we wanted to know if it is always possible to find a deterministic plan that is at least as good as all other plans (even randomized ones) with respect to the minmax rule. The answer is negative for general SSB utility functions and positive for WEU functions. More precisely, while in decision trees (resp. finite horizon MDPs) an SSB optimal strategy (resp. policy) is in general randomized (resp. can be found as a randomized Markovian policy in an augmented MDP), a WEU optimal strategy (resp. policy) can always be found as a deterministic strategy (resp. a deterministic Markovian policy in an augmented MDP).

Secondly, we were interested in the complexity of determining an optimal SSB or WEU plan. For the SSB utility model, we proved that while determining an optimal mixed strategy in decision trees is a problem of polynomial complexity, determining a deterministic strategy which is SSB optimal within the set of deterministic strategies is an NP-hard problem. Interestingly, for the WEU model, both problems become polynomial. In Markov decision problems with finite horizon, we prove that finding an optimal randomized policy for SSB is a pseudo-polynomial problem. Similarly, we prove that finding an optimal

deterministic policy for WEU is a pseudo-polynomial problem. These policies are found as Markovian policies in an augmented version of the MDP.

Thirdly, in both representations (decision trees and MDPs), we designed operational oracle methods to determine an SSB (resp. a WEU) optimal plan. These oracle methods are obtained by identifying the search for an optimal plan as the resolution of a specific zero-sum two-player game (resp. the resolution of a fractional programming problem).

Future works

Other representations of sequential decision problems. A first future work would be obviously to see how our results can be adapted to other representations of sequential decision problems, such as influence diagrams [Howard and Matheson, 1981] or factored MDPs [Boutilier *et al.*, 2000].

Other decision criteria to define a winner of the tournament. Instead of the minmax rule, we could investigate other solution concepts from tournament theory to define an optimal solution of the tournament on plans induced by the SSB model. These other solution concepts could formalize (differently) an optimal plan (e.g., Copeland value [Copeland, 1951]) or an optimal subset of plans (e.g., Schwartz set [Schwartz, 1972]).

Investigating the resolute choice with selves paradigm. In Chapter 4, we have only investigated the paradigm of resolute choice with root dictatorship. We recall that with non-independent preferences, plans that seem optimal for the decision maker at the beginning of the decision problem may not seem optimal anymore in later decision epochs. In the paradigm of resolute choice with root dictatorship we compute an optimal solution with respect to the beginning of the sequential decision problem and we claim that the decision maker should "resolutely" stick to this plan. The prescriptive limitation of this approach is that the decision maker might encounter in the future a situation in which what seems optimal to her is too different from the prescription from the past. In that case, she will want to change the plan and the resolute assumption may fail. A better approach would be to use the resolute choice with multiple selves paradigm in which we search for a plan which is close to optimum in all possible futures. Stated differently, we look for a compromise between the decision maker and the projections of the decision maker (other selves) in future decision epochs. A future work is to extend our results to this other paradigm.

Contributions in Decision Making under Risk with Ordinal or Imprecise Parameters

In Chapter 5, we have addressed decision problems under risk with ordinal or imprecise parameters. In many works in the literature, it is assumed that the values of the parameters characterizing the preferences of the decision maker are known exactly. In this chapter, we studied three settings in which the exact values of these parameters are not available. We explored two possible approaches to tackle this difficulty: 1) working with decision criteria adapted to this lack of information and 2) specifying the values of these parameters by using an elicitation procedure.

In the first section of Chapter 5, we only assumed that we had a preference order over the possible episodes of a finite horizon Markov decision process and we designed a so-

lution method to find a near-optimal policy with regard to a quantile criterion. This solution method solves repeatedly the Markov decision process with different expected utility functions and returns a near-optimal Markovian policy in an augmented version of the MDP.

In the second section of Chapter 5, we addressed the improvement of an interactive optimization method designed to solve an infinite horizon Markov decision process with an imprecisely known reward function. This procedure, named Interactive Value Iteration (IVI), determines a near-optimal policy with respect to the total discounted expected reward criterion by querying a tutor when the knowledge about the reward function does not make it possible to continue the resolution process. We improved the original IVI method by implementing several ideas, in particular controlling the order in which we ask the queries or delaying asking a query in the hope that new dominance relations appear. These modifications greatly reduced the number of queries asked to the tutor.

Finally, in the third section of Chapter 5, we addressed a more simple optimization problem (we assumed the solutions could be explicitly listed) but a more sophisticated decision model, namely the Weighted Expected Utility (WEU) model. While a decision model specifies the parameters that represent the preferences of the decision maker (two real functions in the case of WEU), it does not say how to set correctly the values of these parameters. An elicitation procedure can be designed in order to determine adequate values for the parameters. We designed an elicitation protocol to elicit the WEU model. Then we showed how to integrate this protocol in an incremental procedure in order to identify a near-optimal WEU solution with only few queries. The efficiency of our method relies on a maxmin optimization criterion to guide the querying step (i.e., to determine which query should be asked next) and on a compact representation of functions u and w in WEU by using spline functions.

Future works

Improving a Bayesian version of IVI. In the second section of Chapter 5, we have improved the Interactive Value Iteration (IVI) procedure to reduce the number of queries required to find a near-optimal policy. One limitation of this procedure is that it assumes that the decision maker never makes mistakes. Indeed, each answer of the decision maker induces a constraint on the set of possible reward functions which is never questioned later in the procedure. To address this issue, Weng *et al.* [2013] have adapted the IVI procedure to a Bayesian setting in which the possible errors of the decision maker do not preclude from finding a near-optimal policy. One possible future work could be to see if we could similarly improve this Bayesian procedure to reduce the number of queries required to identify a near-optimal policy.

Improving the incremental elicitation procedure for the WEU model. One future work to improve the incremental elicitation procedure for the WEU model is to use a better criterion than the maxmin one to guide the querying step. Indeed, while this criterion is easy to compute it can be seen as overly conservative. Alternatively, the minmax regret criterion could be used. The minmax regret criterion would yield a tighter upper bound on the maximal loss induced by of a lottery and would lead to better querying strategies. However, this criterion entails algorithmic difficulties. A second possible improvement concerns the robustness of our elicitation procedure. Indeed, in this procedure, we assumed that the decision maker never makes mistakes. Again, this hypothesis is unlikely to be true. One possible way to tackle this issue would be to adapt our procedure to a Bayesian

setting, modeling the probabilities of obtaining each possible response of a query.

Eliciting other SSB utility functions. A natural extension of our work on the elicitation of the parameters of WEU would be to design incremental elicitation procedures for the SSB model or for other sub-classes of the SSB model as the SSB separable model developed by Nakamura [1989] and presented on page 37. The main difficulty of the elicitation of the SSB model is that it requires to elicit a bivariate function, which is more involved than eliciting unary functions (as functions u and w in WEU). To address this difficulty, we could investigate appropriate bivariate spline functions [Chui, 1988], i.e., piecewise polynomial parametric functions that can accurately represent bivariate functions. The incremental elicitation of the SSB model also entails new difficulties as it can represent intransitive preferences. Indeed, as the preference relation over solutions may not be transitive, it would be more difficult to identify dominated solutions that can be excluded from the set of possibly optimal ones during the elicitation process.

Contributions in Robust Combinatorial Optimization

In Chapter 6, we have presented a game-theoretic view of the optimization of the minmax regret criterion in combinatorial optimization problems with interval data. This game-theoretic view made it possible to design an efficient anytime procedure to compute a lower bound on the value of an optimal minmax regret solution. This procedure is based on a double oracle approach. We have seen that this (instance dependent) lower bound is tighter than the other lower bounds proposed in the literature. In particular, this lower bound enables to compute an approximation ratio for the midpoint solution (the optimal solution in the scenario where each cost is set to the middle of the corresponding interval) which is worth 2 in the worst case. Moreover, we explained how this lower bound can be efficiently integrated in a branch and bound procedure to find an optimal minmax regret solution. This approach was then instantiated on the robust shortest path problem with interval data, for which our numerical tests showed the efficiency of the approach.

Future works

Designing a polynomial variant of the double oracle algorithm. A first future work would be to improve the lower bounding procedure we proposed. This procedure is based on a double oracle algorithm that incrementally builds a restricted game in normal form such that solving this restricted game (i.e., determining a mixed Nash equilibrium of the game) is equivalent to solving the entire game. This approach does not take advantage of the structure inherent to the combinatorial optimization problem considered. Similarly to the technique of Chapter 4 (where we have modified the double oracle to take advantage of the structure of the decision tree), we could adapt the double oracle used here to take advantage of the graph structure underlying the combinatorial optimization problem. For instance, in the robust shortest path problem, one could build a subgraph such that an optimal randomized minmax regret solution in the subgraph is also an optimal randomized minmax regret solution in the entire graph. This would lead to a polynomial time variant of the double oracle algorithm to obtain a lower bound on the optimal value of robust shortest path problems, and would probably greatly improve the performance of the method in layered graphs. Indeed, in such graphs, the number of paths between the leftmost and rightmost vertices is combinatorial in the size of the graph and these paths are all exactly of the same length. Therefore, it is unlikely that we can find few paths

that dominate all others with regard to the minmax regret criterion. Hence, the number of iterations of the "standard" double oracle algorithm can be very large. However, the number of iterations of a double oracle algorithm which would take advantage of the structure of the graph would be bounded by the number of edges of the graph, as each iteration of the algorithm would at least add a new edge to the subgraph being constructed.

Robust optimization of fractional programming problems. A second future work would be to address the robust optimization of fractional programming problems under different types of uncertainty over the cost parameters. Obtaining results in this domain could have important implications for robust optimization with the WEU model or with mixture operators. For instance, results in robust fractional programming could be used in the incremental elicitation procedure for the WEU model to guide the querying step (as discussed earlier). The robust optimization of fractional programming problems is a rather new domain [Gorissen, 2015; Sun *et al.*, 2017] and, interestingly, the interval model is already attracting attention [Hladík, 2010].

Contributions in Fair Multi-Agent Optimization

In Chapter 7, we tackled multi-agent optimization problems with two different aggregation operators, namely Ordered Weighted Averages (OWAs) and Mixture Operators (MOs). These two types of operators are, under known conditions on their parameters, compatible with the Pigou-Dalton principle (ensuring fairness) and Pareto-optimality (ensuring efficiency). Hence, their optimization makes it possible to find fair and efficient solutions in a multi-agent decision problem.

In the first part of Chapter 7, we investigated the optimization of an OWA with decreasing weights in a randomized version of fair multi-agent optimization problems. Considering the randomized setting instead of the deterministic one makes more solutions available, that may have better fairness properties. We designed a game-theoretic model of this problem which enabled us to obtain two solution methods based on dynamic calls to oracles. The first method is a cutting plane method and the second method is a double oracle method.

In the second part of Chapter 7, we studied the optimization of an MO in two types of multi-agent problems, namely allocation problems and proportional representation problems. While the optimization of an MO revealed to be NP-hard in both problems, we have seen that optimizing an MO can be performed in polynomial time in assignment problems (a subclass of allocation problems where there are as many agents as objects to be shared and each agent should receive exactly one object) and in proportional representation problems where the preferences of the agents abide to some particular structures (e.g., single-peaked, single-crossing). We have presented solution methods based on fractional programming for both types of problems.

Future works

Investigating the optimization of MOs in other multi-agent problems. A direct extension of this work would be obviously to investigate the optimization of MOs in other multi-agent problems or to find more general results that could apply to all or some classes of multi-

agent problems.

Approximation algorithms for the optimization of MOs. A more specific future direction concerns approximation algorithms. Indeed, several papers in fractional programming investigate how to design a polynomial-time approximation algorithm with performance guarantee for a fractional problem when a polynomial-time approximation algorithm with performance guarantee is known for the standard version of the problem [Correa *et al.*, 2006, 2010; Hashizume *et al.*, 1987]. Hence, following the same line, approximation algorithms for allocation problems (with a sub-modular utility function [Feige and Vondrak, 2006]) or proportional representation problems [Lu and Boutilier, 2011; Skowron *et al.*, 2015] may be adapted for the optimization of an MO.

References

C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial intelligence*, 121(1):49–107, 2000. 244

C.K. Chui. *Multivariate splines*. SIAM, 1988. 246

A.H. Copeland. A reasonable social welfare function. In *Mimeographed notes from a Seminar on Applications of Mathematics to the Social Sciences, University of Michigan*, 1951. 244

J.R. Correa, C.G. Fernandes, and Y. Wakabayashi. Approximating rational objectives is as easy as approximating linear ones. In *Scandinavian Workshop on Algorithm Theory*, pages 351–362. Springer, 2006. 248

J.R. Correa, C.G. Fernandes, and Y. Wakabayashi. Approximating a class of combinatorial problems with rational objective function. *Mathematical programming*, 124(1-2):255–269, 2010. 248

U. Feige and J. Vondrak. Approximation algorithms for allocation problems: Improving the factor of 1-1/e. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 667–676. IEEE, 2006. 248

B.L. Gorissen. Robust fractional programming. *Journal of Optimization Theory and Applications*, 166(2):508–528, 2015. 247

S. Hashizume, M. Fukushima, N. Katoh, and T. Ibaraki. Approximation algorithms for combinatorial fractional programming problems. *Mathematical Programming*, 37(3):255–267, 1987. 248

M. Hladík. Generalized linear fractional programming under interval uncertainty. *European Journal of Operational Research*, 205(1):42–46, 2010. 247

R.A. Howard and J.E. Matheson. *Influence diagrams*. Stanford Research Institute, 1981. 244

T. Lu and C. Boutilier. Budgeted social choice: From consensus to personalized decision making. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 11, pages 280–286, 2011. 248

Y. Nakamura. Risk attitudes for nonlinear measurable utility. *Annals of Operations Research*, 19(1):311–333, 1989. 246

T. Schwartz. Rationality and the myth of the maximum. *Noûs*, 6(2):97–117, 1972. 244

P. Skowron, P. Faliszewski, and A. Slinko. Achieving fully proportional representation: Approximability results. *Artificial Intelligence*, 222:67–103, 2015. 248

X-K Sun, X-J Long, H-Y Fu, and X-B Li. Some characterizations of robust optimal solutions for uncertain fractional optimization and applications. *Journal of Industrial and Management Optimization*, 13(2):803–824, 2017. 247

P. Weng, R. Busa-Fekete, and E. Hüllermeier. Interactive Q-Learning with Ordinal Rewards and Unreliable Tutor. In *ECML/PKDD Workshop Reinforcement Learning with Generalized Feedback*, September 2013. 245

Résumé. Cette thèse s'inscrit dans le cadre de la *théorie de la décision algorithmique*, qui est une discipline au croisement de la théorie de la décision, la recherche opérationnelle et l'intelligence artificielle. Dans cette thèse, nous étudions l'utilisation de plusieurs *modèles décisionnels* pour résoudre des problèmes de *décision séquentielle dans l'incertain*, d'*optimisation robuste*, et d'*optimisation multi-agents équitable*. Pour résoudre efficacement ces problèmes, nous utilisons des méthodes de type maître-esclaves, dites *à base d'oracles* dans la thèse. Ces méthodes permettent de résoudre des problèmes de grande taille en procédant de manière incrémentale. Une attention particulière est portée au modèle de *l'espérance d'utilité antisymétrique et bilinéaire*, au modèle de *l'espérance d'utilité pondérée* et à leurs pendants en décision multicritère. L'intérêt de ces modèles est multiple. En effet, ils étendent les modèles standards (e.g., modèle de l'espérance d'utilité) et permettent de représenter un spectre étendu de préférences tout en conservant leurs bonnes propriétés théoriques et algorithmiques. La thèse apporte des réponses sur des aspects théoriques (e.g., résultats de complexité algorithmique) et sur des aspects opérationnels (e.g., conception de méthodes de résolution efficaces) aux problèmes soulevés par l'emploi de ces critères dans les contextes susmentionnés.

Mots-clefs : Théorie de la décision algorithmique · Décision séquentielle dans l'incertain · Optimisation robuste · Optimisation équitable · Méthodes à base d'oracles · Élicitation de préférences

Abstract. This thesis falls within the area of *algorithmic decision theory*, which is at the crossroads between decision theory, operational research and artificial intelligence. In this thesis, we study several *decision models* to solve problems in different domains: *sequential decision problems under risk*, *robust optimization problems*, and *fair multi-agent optimization problems*. To solve these problems efficiently, we use master-slave algorithms which solve the problem through an incremental process. These procedures, referred to as *oracle methods* in the thesis, make it possible to solve problems of large size. A particular attention is given to the *skew-symmetric bilinear utility model*, the *weighted expected utility model* and their counterparts in multicriteria decision making. These models are interesting at several respects. They extend the standard models (e.g., the expected utility model) and allow to represent a broader class of preferences while retaining their good theoretical and algorithmic properties. The thesis focuses both on theoretic (e.g., complexity results) and operational (e.g., design of practically efficient solution methods) aspects of the problems raised by the use of these criteria in the domains aforementioned.

Keywords: Algorithmic decision theory · Sequential decision making under uncertainty · Robust optimization · Fair optimization · Oracle methods · Preference elicitation