UNIVERSITÉ DE TUNIS

INSTITUT SUPÉRIEUR DE GESTION

MASTER RECHERCHE

SPÉCIALITÉ

SCIENCES ET TECHNIQUES DE L'INFORMATIQUE DÉCISIONNELLE (STID)

OPTION

INFORMATIQUE ET GESTION DE LA CONNAISSANCE (IGC)

EXTENDED DYNAMIC PROGRAMMING FOR OPTIMIZING STRATEGIES IN POSSIBILISTIC DECISION TREES

ZEINEB ELKHALFI

NAHLA BEN AMOR	PROFESSEUR, ISG TUNIS	DIRECTEUR DU MÉMOIRE
WIDED GUEZGUEZ	MAITRE ASSISTANTE, ESSEC TUNIS	CO-DIRECTEUR

Laboratoire/Unité recherche: LARODEC

2013-2014

Abstract

This work studies sequential decision problems in the possibilistic framework. We are particularly interested by possibilistic decision trees that allows describing sequential decision problems and defining optimal strategies, w.r.t. a possibilistic decision criterion, through an evaluation algorithm. The major part of this work focuses on strategy optimization in decision trees for the possibilistic likely dominance criteria, namely $L\Pi$ and LN. It shows that even if the preference order induced by these two criteria is not fully transitive but only quasi-transitive, we can extend the Dynamic programming algorithm to generate the optimal strategies in polytime. Our goal is to give a deep study of the strategy optimization algorithms, especially the new Extended dynamic programming. We provides also an experimental analysis in order to prove its effectiveness and its interest versus standard Dynamic programming.

Keywords

Possibility theory, Possibilistic decision trees, Possibilistic Likely Dominance, Dynamic programming, Extended Dynamic Programming.

Résumé

Ce travail étudie les problèmes de décision séquentielle dans le cadre possibiliste. Nous nous intéressons particulièrement aux arbres de décision possibilistes qui permettent de décrire les problèmes de décision séquentielle et de définir une stratégie optimale, par rapport à un critère de décision possibiliste, grâce à un algorithme d'évaluation. La majeure partie de ce travail se concentre sur l'optimisation de stratégie dans les arbres de décision pour les critères de dominance vraisemblable possibilistes, à savoir $L\Pi$ et LN. Il montre que, même si l'ordre de préférence induit par ces deux critères est non transitif, sa quasi-transitivité peut être exploitée afin d'étendre l'algorithme de programmation dynamique et générer une stratégie optimale en temps polynomial. Notre objectif est de fournir une étude approfondie des algorithmes d'optimisation de stratégie, en particulier le nouvel algorithme de programmation dynamique étendue. Nous proposons également une analyse expérimentale afin de prouver son efficacité et son intérêt par rapport à la programmation dynamique standard.

Mots-clés

Théorie des possibilités, Arbres de décision possibilistes, Dominance vraisemblable possibiliste, Programmation dynamique, Programmation dynamique étendue.

Acknowledgment

I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

Foremost, I would like to express my sincere gratitude to my advisor Prof. Nahla Ben Amor. I cant say thank you enough for your continuous support of my graduate studies, your patience, motivation, enthusiasm, immense knowledge and guidance that helped me in all the time of research and writing of this dissertation. I feel motivated and encouraged every time I attend your meeting and I could not have imagined having a better advisor and mentor for my masters degree.

I would like to express my sincere gratitude to my co-advisor Dr. Wided Guezguez for her productive comments, assistance, encouragements and for all the interested discussion and comments. The guidance and support received from you was vital for the success of the project. One simply could not wish for a better or friendlier supervisor.

I would like to express my special appreciation and thanks to Hélène Fargier, for her interest to this work and her brilliant comments and suggestions, thanks to you.

Finally, I would like to thank my beloved family: my parents and my sisters for all of the sacrifices that you have made on my behalf and for supporting me spiritually throughout my life. I would also like to thank my best friend Beya who supported me, and incented me to strive towards my goal when there was no one to answer my queries.

Contents

General introduction **1**

1 Decision making under possibilistic uncertainty **3**

1.1 Introduction 3

1.2 Limitations of classical decision theory 4

1.3 Basics of possibilistic decision theory 5

1.4 Decisional aspect in possibility theory 6

1.5 Possibilistic Decision criteria 8

1.5.1 Qualitative possibilistic utilities 8

1.5.2 Possibilistic Choquet integrals 10

1.5.3 Possibilistic likely dominance 10

1.6 Conclusion 12

2 Optimizing strategies in possibilistic decision trees **13**

2.1 Introduction 13

2.2 Decision trees 14

2.3 Possibilistic decision trees 15

2.4 Evaluation of possibilistic decision trees 17

2.4.1 Strategy optimization problem in possibilistic decision trees 18
2.4.2 Dynamic programming 19
2.4.3 Branch and Bound 20
2.5 Optimizing possibilistic decision criteria 21
2.5.1 Optimizing qualitative possibilistic utilities 22
2.5.2 Optimizing possibilistic Choquet integrals 23
2.5.3 Optimizing possibilistic likely dominance 24
2.6 Conclusion 27

3 Extended Dynamic Programming 28
3.1 Introduction 28
3.2 Extended dynamic programming algorithm 28
3.2.1 Comparing lotteries by $L\Pi$ 29
3.2.2 Comparing lotteries by LN 30
3.3 Illustrative example 34
3.4 Recapitulation 37
3.5 Conclusion 38

4 Implementation and Experimentation 39
4.1 Introduction 39
4.2 Implementation 40
4.2.1 Framework 40
4.2.2 Main programs 41
4.2.3 Implementation tricks 43
4.3 Experimentation 45
4.3.1 Evaluation measures 45

4.3.2 Experimental Setup 49

4.3.3 Experimental results 50

4.4 Experimentation to the TV game show: ”Who Wants To Be A Millionaire?” . . . 53

4.4.1 Game’s rules 53

4.4.2 The general model 54

4.4.3 Possibilistic light ”Who Wants To Be A Millionaire?” 56

4.5 Conclusion 64

Conclusion and perspectives **66**

Bibliography **68**

Appendix 1 **71**

List of Figures

1.1 A possibilistic compound lottery (a) and its reduction (b) 7

2.1 Example of a probabilistic decision tree. 15

2.2 Example of possibilistic decision tree. 17

2.3 Possibilistic decision tree for Example 2.3. 22

2.4 Decision tree of Counter-example 2.1. 24

2.5 Decision tree of Counter-example 2.2. 25

2.6 Decision tree of Counter-example 2.3. 26

3.1 Decision tree of counter-example 2.2. 34

4.1 Main menu interface. 41

4.2 Evaluating possibilistic decision trees interface: Dynamic programming. 42

4.3 Evaluating/comparing lotteries interface. 44

4.4 Decision tree for Example 4.1. 46

4.5 Decision tree for Example 4.2. 47

4.6 Example of a complete binary decision tree for Seq=2. 49

4.7 Execution CPU time for DynProg and ExtDynProg (in ms). 50

4.8 Maximum size of the set S. 51

4.9 Average closeness values. 52

4.10 Refined decision tree for the TV game. 55

4.11 Process of the TV game. 56

4.12 Graphical component of the TV game decision tree. 57

List of Tables

3.1 Summary of strategy optimization in possibilistic decision trees. 38

4.1 Percentage of problems where $\delta^*_{ExtDynProg} \sim_O \delta^*_{DynProg}$. 51

4.2 Minimum and maximum action similarity values. 52

4.3 Numerical component of the game decision tree in the case of qualitative setting. 59

4.4 Numerical component of the game decision tree in the case of quantitative setting. 60

4.5 Optimal strategies of the game in the case of qualitative setting. 61

4.6 Optimal strategies of the game in the case of quantitative setting. 62

4.7 Optimal strategies of the game for adventurous decision makers. 63

4.8 Optimal strategies of the game for cautious decision makers. 64

List of Algorithms

1 DynProg(N:Node, δ: Strategy) 19

2 BB(δ: a (partial) strategy, O_μ : its corresponding value) with $\mu = N$ or $\mu = \Pi$ . . 21

3 ExtDynProg(N : node) 31

4 Prune$_O$(S: a set of pairs $(L : strategy, \delta_L : lottery)$) 31

General introduction

The growing interest in the Artificial Intelligence community towards algorithmic issues on decision making, and specifically decision making under uncertainty, motivates us to study the problem of possibilistic sequential decision making [26].

In sequential decision problem, decision maker does not make a simple decision but he follows a *strategy* i.e. a sequence of decisions that are executed successively. In many of these problems, decision maker must act under uncertainty such as in robot control, relief organization, medical diagnosis, modeling games etc.

In classical decision-making under uncertainty framework, one aims to maximize Expected utility, such approaches require probabilities for all events. So this framework is only appropriate when all numerical data are available, which is not always the case. Several non-classical theories of uncertainty have been suggested in order to deal with uncertain information including fuzzy sets theory [32], evidence theory [30] and possibility theory [33].

Possibility theory offers a natural and simple model to handle uncertain information. The development of possibilistic decision theory has led to the proposition of a series of *possibilistic decision criteria*, namely: optimistic and pessimistic possibilistic qualitative criteria [13], possibilistic likely dominance [9], binary possibilistic utility [17] and possibilistic Choquet integrals [27]. Moreover, several representation formalisms can be used for sequential decision problems, such as influence diagrams [19], Markov decision processes [7], valuation based systems [31] and decision trees [26] etc. We are interested to *decision trees*. It is important to note that, in this formalism, the set of potential strategies is combinatorial and it may grow exponentially. The computation of an optimal strategy for a given representation and a given decision criterion is an algorithmic issue.

Taking advantage of the fact that the optimistic and pessimistic possibilistic qualitative criteria are transitive and obey a weak form of the monotonicity property, [3] have indeed shown that

Dynamic programming algorithm [6] may be used to get an optimal strategy in polytime; this is also the case for the binary possibilistic utility [3]. For possibilistic Choquet Integrals, on the contrary, it has been shown that the problem of finding an optimal strategy is NP-hard [4].

The present reminder points out possibilistic likely dominance criteria [9] where computing an optimal strategy according to these criteria raises a challenging algorithmic problem due to its quasi-transitivity. We illustrate the weakness of the Dynamic programming algorithm when optimizing likely dominance and this is precisely the issue that we tackle in this master thesis. We propose to use an extension of the standard Dynamic programming algorithm, namely *Extended dynamic programming*, and show that it allows to find an optimal strategy in polynomial time. Such a strategy satisfies the optimality principle: any sub strategy of an optimal strategy is itself optimal: this is closely related to a condition of monotonicity.

The reminder is organized as follows: Chapter 1 presents the limitations of classical theory of uncertainty and then it describes the possibility theory as one of the best theories to handle imperfect information. Hence, decision making under possibility theory is developed. The second part shows a survey on possibilistic decision criteria that compares alternatives in order to choose the best decision. Chapter 2 defines standards and possibilistic decision trees. Then, it discusses existing algorithms for optimizing possibilistic decision criteria in such trees. Chapter 3 details Extended dynamic programming and its characteristics. Chapter 4 presents the implementation of a toolbox, that allows to evaluate possibilistic decision trees, and experimental results of optimizing strategy algorithms. It also provides an application of the optimizing strategy algorithms to the TV game namely ”Who Wants To Be A Millionaire?”, in the possibilistic framework. Appendix 1 presents main proofs of propositions presented in Chapter 3.

Chapter 1

Decision making under possibilistic uncertainty

1.1 Introduction

Decision-making problem can be regarded as a situation in which the decision maker has to select an action among several alternatives, according to its preferences.

In real world decision problems several types of uncertainty should be considered. Indeed, most of available decision models refer to probability theory for the representation of uncertainty. However, this framework has some limits since it cannot represent all kind of uncertainty. Therefore, several non-classical theories of uncertainty have been proposed in order to deal with imperfect information namely fuzzy sets theory [32], evidence theory [30] and possibility theory [33] etc. In this work, we will focus on decision making in the context of possibility theory.

This chapter provides a theoretical background of possibilistic decision theory: Section 1.2 presents uncertainty in decision problems and it gives some shortcomings of classical decision theory. Section 1.3 introduces the basics of possibility theory. Next, Section 1.4 tells about some decisional aspects of possibility theory, it especially details the possibilistic counterpart of probabilistic lottery. Finally, Section 1.5 reviews the most commonly used possibilistic decision criteria.

1.2 Limitations of classical decision theory

The problem of decision-making under uncertainty is to choose an act (or decision) among many different available alternatives, considering decision maker's knowledge about states of nature. Each act has uncertain consequences and each consequence has a utility. This process aims to maximize decision maker satisfaction.

In fact, there exist several classical decision theories that allow to model and solve decision-making under uncertainty problems. The use of the appropriate one depends on the context of the decision problem and the nature of uncertainty. Probabilistic decision theory is the most famous quantitative decision models. The concept of probability is described as an 'aleatory uncertainty', so, probability theory provides numerical measurements of the likelihood of an outcome of some random process e.g. game chance. Indeed, we use probability function that quantifies and represents uncertainty about states of nature by numerical values.

The most classical decision criterion is *Expected Utility* [24] that compares any two decisions, either one is better than the other, or the two are equivalent. This proposal was introduced by Bernoulli and developed by von Neumann and Morgenstern [24] in the context of objective probability also called by frequentest probability. In 1954, Savage has developed subjective Expected Utility [29] that models the personal degree of belief that the events occur. However, this classical decision model has some limits such as:

- It cannot represent all the decision maker behavior such as the case of total ignorance which is represented by equiprobability, so it formalizes randomness rather than ignorance.
- It cannot represent all kinds of uncertainty like qualitative uncertainty: in some situations, decision makers are unable to give exact numerical values to quantify decision problems but they can only provide an order relation between different values.
- Allais [1] and Ellsberg [14] have presented paradoxes i.e. experiences where EU cannot be used: In 1953, Allais paradox has shown that the independence axiom of the VNM's system is violated [1]. In 1961, Ellsberg has shown the contradiction of the Sure Thing Principle of subjective expected utility.

In order to overcome these limitations, alternative non-classical theories of uncertainty are used in decision making e.g. fuzzy sets theory [32], evidence theory [30] and possibility theory [33]. In this work, we are interested in possibilistic decision theory that aims to avoid weakness of EU decision models by using possibility theory for the representation of uncertainty.

1.3 Basics of possibilistic decision theory

Our choice of the possibilistic framework is motivated by the fact that *possibility theory* offers a natural and flexible model to represent and handle uncertain information. In fact, it is an appropriate framework for experts to express their opinions about uncertainty:

- **Numerically**, using possibility degrees.
- **Qualitatively** using total pre-order on the universe of discourse. This order can be represented by numerical values which have no sense but which express only the order.

Possibility theory, issued from Fuzzy Sets theory, was introduced by Zadeh [33] and further developed by Dubois and Prade [12]. The possibility distribution is a representation of a state of knowledge of an agent regarding the state of the world. It is denoted by π and it is a mapping from the universe of discourse Ω to a bounded linearly ordered scale L exemplified by the unit interval $[0, 1]$, we denote the function by: $\pi : \Omega \to [0, 1]$.

For vector $\omega \in \Omega$, $\pi(\omega) = 1$ means that realization ω is totally possible and $\pi(\omega) = 0$ means that ω is an impossible state. It is generally assumed that there exist at least one state ω which is totally possible: π is said then to be *normalized*.

In the possibilistic framework, extreme forms of partial knowledge can be captured, namely:

- *C*omplete knowledge i.e. $\exists\, \omega_0$ s.t. $\pi(\omega_0) = 1$ and $\forall\, \omega \neq \omega_0, \pi(\omega) = 0$.
- *T*otal ignorance i.e. $\forall \omega \in \Omega, \pi(\omega) = 1$ (all values in Ω are possible).

In possiblistic theory, there are two essential measures:

- **Possibility measure:** $\Pi(A) = \sup_{\omega \in A} \pi(\omega)$. $\Pi(A)$ is the possibility of an event A i.e. the degree of possibility of having one of the elements of A (for adventurous decision maker).
- **Necessity measure:** $N(A) = 1 - \Pi(\bar{A}) = 1 - \sup_{\omega \notin A} \pi(\omega)$. N(A) expresses the need for an event A i.e. the certainty of having one of the elements of A (for cautious decision).

Example 1.1. *Let us consider the possibility distribution π which shows the opinion of a doctor concerning the diagnosis of a patient. The universe of discourse related to this problem is the set of three diseases and a healthy case: $\Omega = \{d1, d2, d3, h\}$.*

$$\pi(d1) = 0.5, \quad \pi(d2) = 1, \quad \pi(d3) = 0.7, \quad \pi(h) = 0.$$

Note that Π *is normalized since* $max(0.5, 1, 0.7, 0) = 1$. *For instance, the degree of possibility that the patient has the disease* $d1$ *is* 0.5.

For the event A*: The patient suffers from d1 or d3, we have:*

- $\pi(A) = max(0.5, 0.7) = 0.7$
- $N(A) = 1 - max(1, 0) = 0.0$

In this theory, the possibilistic conditioning may be considered as a special case of informations fusion. It consists in revising the initial knowledge, represented by a possibility distribution π, and to change it into another possibility distribution $\pi' = \pi(.|\psi)$ with $\psi \neq \emptyset$ and $\Pi(\psi) > 0$.

The two interpretations of the possibilistic scale induce two definitions of the operation of conditioning:

- min-based conditioning, which is relative to the ordinal setting:

$$\pi(\omega|\psi) = \begin{cases} 1 & if\ \pi(\omega) = \Pi(\psi)\ \ and\ \ \omega \in \psi \\ \pi(\omega) & if\ \pi(\omega) < \Pi(\psi)\ \ and\ \ \omega \in \psi \\ 0 & otherwise. \end{cases} \tag{1.1}$$

- product-based conditioning, which is relative to the numerical setting:

$$\pi(\omega|\psi) = \begin{cases} \frac{\pi(\omega)}{\Pi(\psi)} & if\ \omega \in \psi \\ 0 & otherwise. \end{cases} \tag{1.2}$$

1.4 Decisional aspect in possibility theory

A decision can be seen as a possibility distribution over a unit set of outcomes (utilities) [13]. In a single stage decision making problem, utility function maps each outcome to a utility value in a totally ordered set $U = \{u_1, \ldots, u_p\}$. This function models the attractiveness of each outcome for the decision maker. An act can then be represented by a possibility distribution on U, also called a *simple possibilistic lottery*, and denoted by $\langle \lambda_1/u_1, \ldots, \lambda_p/u_p \rangle$ i.e. $\lambda_i = \Pi(u_i)$ is the possibility that the decision leads to an outcome of utility u_i; this possibility degree is also denoted by $L[u_i]$.

A *possibilistic compound lottery* $\langle \lambda_1/L_1, \ldots, \lambda_r/L_r \rangle$ is a possibility distribution over the set of r simple lotteries. The possibility $\Pi_{i,j}$ of getting an utility degree $u_j \in U$ from one of

its sub-lotteries L_i depends on the possibility λ_i of getting L_i and on the conditional possibility $\lambda_j^i = \Pi(u_j \mid L_i)$ of getting u_j from L_i i.e. $\Pi_{i,j} = min(\lambda_j, \lambda_j^i)$ in the case of ordinal setting and $\Pi_{i,j} = \lambda_j * \lambda_j^i$ in the case of numerical one.

Thus, [11, 13] have proposed to reduce a compound lottery $\langle \lambda_1/L_1, ..., \lambda_r/L_r \rangle$ into an equivalent simple lottery. Formally we have:

$$Reduction(\langle \lambda_1/L_1, \ldots, \lambda_r/L_r \rangle) = \langle \max_{j=1..r}(\otimes(\lambda_1^j, \lambda_j))/u_1, \ldots, \max_{j=1..r}(\otimes(\lambda_p^j, \lambda_j))/u_p \rangle. \quad (1.3)$$

where $\otimes = min$ for the ordinal setting and $\otimes = *$ for the numerical one.

Example 1.2. *Let* $L_1 = \langle 0.8/u_1, 1/u_2 \rangle$ *and* $L_2 = \langle 0.6/u_1, 0.9/u_2 \rangle$ *be two simple lotteries, the possibilistic compound lottery* $\langle 1/L_1, 0.6/L_2 \rangle$ *and its reduction is represented by 1.1 with* $\otimes = min$ *and* $\otimes = *$.

Figure 1.1: A possibilistic compound lottery (a) and its reduction (b)

We can see that, the reduction of a simple lottery is the simple lottery itself. Furthermore, the reduction of a compound lottery is a polynomial operation, because, min, $*$ and max are also polynomial operations. Formally, for any comparison criterion O, i.e. any preference order $\succeq_O$ defined on simple lotteries, can be extended to compound lotteries as follows:

$$L \succeq_O L' \text{ iff } Reduction(L) \succeq_O Reduction(L'). \quad (1.4)$$

1.5 Possibilistic Decision criteria

The interpretations of uncertainty and utility scales lead to several possibilistic decision criteria which can be gathered into three classes:

- **Utility and possibility scales are commensurate and purely ordinal:** we can find in this class possibilistic qualitative criteria such as possiblistic qualitative optimistic (denoted by U_{opt}), pessimistic (denoted by U_{pes}) [11, 13] and possibilistic binary utilities (denoted by PU) [17].
- **Utility and possibility scales are commensurate:** in this class we have possibility-based Choquet integrals (denoted by Ch_{Π}) and Necessity-based Choquet integrals (denoted by Ch_N) [27].
- **Utility and possibility scales are not commensurate:** we have possibility-based likely dominance (denoted by $L\Pi$) and Necessity-based likely dominance (denoted by LN) [9, 15].

1.5.1 Qualitative possibilistic utilities

Under the assumption that the utility scale and the possibility scale are commensurate and purely ordinal, Dubois and Prade [11, 13] have proposed qualitative pessimistic (denoted by U_{pes}) and optimistic (denoted by U_{opt}) utility degrees for evaluating any simple lottery $L = \langle \lambda_1/u_1, ..., \lambda_p/u_p \rangle$ (possibly issued from the reduction of a compound lottery):

$$U_{pes}(L) = \min_{i=1..p} \max(u_i, 1 - \lambda_i). \tag{1.5}$$

$$U_{opt}(L) = \max_{i=1..p} \min(u_i, \lambda_i). \tag{1.6}$$

Dubois and Prade [13] have proved that $\forall L, L' \in \mathcal{L}$ we have :

$$L \succeq_{U_{pes}} L' \iff U_{pes}(L) \geq U_{pes}(L'). \tag{1.7}$$

$$and \;\; L \succeq_{U_{opt}} L' \iff U_{opt}(L) \geq U_{opt}(L'). \tag{1.8}$$

U_{pes} generalizes the Wald criterion and estimates to what extend it is certain (i.e. necessary according to measure N) that L reaches a good utility. Its optimistic counterpart, U_{opt}, estimates to what extend it is possible that L reaches a good utility. Because decision makers are rather cautious than adventurous, the former is generally preferred to the latter. The preference order induced by U_{opt} and U_{pes} is obviously complete transitive [16, 17].

Example 1.3. *Let us consider the two lotteries* $L = \langle 1/0.2, 0.7/0.5, 0.4/0.6\rangle$ *and* $L' = \langle 1/0.8, 0.3/0.7, 0.5/0.9\rangle$. *Using Equation 1.5 we have:*

- $U_{pes}(L) = min(max(0.2, 0), max(0.5, 0.3), max(0.6, 0.6)) = 0.2.$
- $U_{pes}(L') = min(max(0.8, 0), max(0.7, 0.7), max(0.9, 0.5)) = 0.7.$

$\Rightarrow U_{pes}(L') > U_{pes}(L)$ *so* $L' \succeq_{U_{pes}} L$.

Using Equation 1.6 we have:

- $U_{opt}(L) = max(min(0.2, 1), min(0.5, 0.7), min(0.4, 0.6)) = 0.5.$
- $U_{opt}(L') = max(min(1, 0.8), min(0.3, 0.7), min(0.5, 0.9)) = 0.8.$

$\Rightarrow U_{opt}(L') > U_{opt}(L)$ *so* $L' \succeq_{U_{opt}} L$.

Claiming that the lotteries that realize in the best prize or in the worst prize play an important role in decision making, Giang and Shenoy [17] have proposed a bipolar model in which the utility of an outcome is a pair $u = \langle \overline{u}, \underline{u}\rangle$ where $\max(\overline{u}, \underline{u}) = 1$ i.e. the utility is binary in this sense that $\overline{u}$ is interpreted as the possibility of getting the ideal, good reward (denoted $\top$) and $\underline{u}$ is interpreted as the possibility of getting the anti-ideal, bad reward (denoted $\perp$).

$$\begin{aligned} PU(\langle \lambda_1/u_1, \ldots, \lambda_n/u_n\rangle) &= Reduction(\lambda_1/\langle \overline{u_1}/\top, \underline{u_1}/\perp\rangle, \ldots, \lambda_n/\langle \overline{u_n}/\top, \underline{u_n}/\perp\rangle) \\ &= \langle \max_{j=1..n}(\min(\lambda_j, \overline{u_j}))/\top, \max_{j=1..n}(\min(\lambda_j, \underline{u_j}))/\perp\rangle. \end{aligned} \tag{1.9}$$

We thus get, for any lottery L a binary utility $PU(L) = \langle \overline{u}, \underline{u}\rangle$ in U. Lotteries can then be compared:

$$L \succeq_{PU} L' \iff Reduction(L) \succeq_{PU} Reduction(L'). \tag{1.10}$$

The preference order induced by PU is transitive [18] and it collapses with the one induced by U_{opt} (resp. U_{pes}) whenever for any lottery, the possibility $\underline{u}$ (resp. $\overline{u}$) of getting the worst (resp. the best) utility is equal to 1. One shall thus say that PU captures U_{opt} and U_{pes} as particular cases.

1.5.2 Possibilistic Choquet integrals

In presence of heterogeneous information i.e. when the knowledge about the state of the world is possibilistic while the utility degrees are numerical and compensatory, Choquet integrals appear as a right way to extend expected utility to non-Bayesian models [8]. This model is a numerical, compensatory, way of aggregating uncertain utilities. Indeed, this approach allows the use of any monotonic set function μ, also called a capacity or fuzzy measure. Such measure capture probability measures, necessity, possibility measures and belief functions etc. as particular cases [27]. In the case of possibilistic decision making we have: for cautious (resp. adventurous) decision makers, the capacity μ is the necessity measure N (resp. the possibility measure Π):

$$Ch_{\Pi}(L) = u_1 + \sum_{i=2..p} (u_i - u_{i-1}) * \Pi(L \geq u_i). \tag{1.11}$$

$$Ch_{N}(L) = u_1 + \sum_{i=2..p} (u_i - u_{i-1}) * N(L \geq u_i). \tag{1.12}$$

Example 1.4. *Let us consider* $L_1 = \langle 1/10, 0.2/20, 0.7/30 \rangle$ *and* $L_2 = \langle 1/10, 0.1/30 \rangle$. *Using Equation 1.11 we have:*

- $Ch_{\Pi}(L_1) = 10 + [(20 - 10) * 0.2] + [(30 - 20) * 0.7] = 19.$
- $Ch_{\Pi}(L_2) = 10 + [(30 - 10) * 0.1] = 12.$

$\Rightarrow Ch_{\Pi}(L_1) > Ch_{\Pi}(L_2)$ *So* $L_1 \succeq_{Ch_{\Pi}} L_2$.

Let $L_1 = \langle 0.3/10, 0.5/20, 1/30 \rangle$ *and* $L_2 = \langle 1/10, 0.5/20, 0.2/30 \rangle$. *Using Equation 1.12 we have:*

- $Ch_{N}(L_1) = 10 + [(20 - 10) * (1 - 0.3)] + [(30 - 20) * (1 - 0.5)] = 22.$
- $Ch_{N}(L_2) = 10 + [(20 - 10) * (1 - 1)] = 10.$

$\Rightarrow Ch_{N}(L_1) > Ch_{N}(L_2)$ *So* $L_1 \succeq_{Ch_{N}} L_2$.

1.5.3 Possibilistic likely dominance

When the scales evaluating the utility and the possibility of the outcomes are not commensurate, [9, 15] propose to prefer, among two possibilistic decisions, the one that is more likely to overtake the other. Such a rule does not assign a utility degree to the decisions, but draws a pairwise

comparison. This rule states that for possibilistic likely dominance $(L\Pi, LN)$ given two lotteries L_1 and L_2, L_1 is as least as good as L_2 as soon as the likelihood (here, the necessity or the possibility) of the event *" The utility of L_1 is as least as good as the utility of L_2"* is greater or equal to the likelihood of the event *"The utility of L_2 is as least as good as the utility of L_1"*. Formally:

$$L_1 \succeq_{LN} L_2 \;\; iff \;\; N(L_1 \geq L_2) \geq N(L_2 \geq L_1). \tag{1.13}$$

$$L_1 \succeq_{L\Pi} L_2 \;\; iff \;\; \Pi(L_1 \geq L_2) \geq \Pi(L_2 \geq L_1). \tag{1.14}$$

where: $\Pi(L_1 \geq L_2) = \max\limits_{u_i^1, u_j^2 \; s.t \; u_i^1 \geq u_j^2} (\otimes(\lambda_i^1, \lambda_j^2))$,

and $N(L_1 \geq L_2) = 1 - \max\limits_{u_i^1, u_j^2 \; s.t \; u_i^1 < u_j^2} (\otimes(\lambda_i^1, \lambda_j^2))$.

where $\otimes$=min for the ordinal setting and $\otimes = *$ for the numerical one.

Example 1.5. *Taking the two lotteries $L_1 = \langle 1/0.3, 0.2/0.5 \rangle$ and $L_2 = \langle 1/0.4, 0.6/0.3 \rangle$.*

Using Equation 1.13 we have:

- $N(L_1 \geq L_2) = 1 - max(min(1, 1)) = 0.$
- $N(L_2 \geq L_1) = 1 - max(min(1, 0.2), min(0.6, 0.2)) = 0.4.$

$\Rightarrow N(L_2 \geq L_1) \geq N(L_1 \geq L_2)$, *so* $L_2 \succeq_{LN} L_1$.

Using Equation 1.14:

- $\Pi(L_1 \geq L_2) = max(min(1, 0.6), min(0.2, 1), min(0.2, 0.6)) = 0.6.$
- $\Pi(L_2 \geq L_1) = max(min(1, 1), min(0.6, 1), min(0.6, 0.2)) = 1.$

$\Rightarrow \Pi(L_2 \geq L_1) \geq \Pi(L_1 \geq L_2)$, *so* $L_2 \succeq_{L\Pi} L_1$.

The preference order induced on the lotteries is not transitive, but only quasi-transitive [9]. Obviously $L_1 \succeq_{LN} L_2 \; and \; L_2 \succeq_{LN} L3$ implies that $L_1 \succ_{LN} L_3$ $(resp.\; L_1 \succeq_{L\Pi} L_2$ and $L_2 \succeq_{L\Pi} L_3$ implies that $L_1 \succ_{L\Pi} L_3)$ but it may happen that $L_1 \sim_{LN} L_2, L_2 \sim_{LN} L_3$ $(resp.\; L_1 \sim_{L\Pi} L_2, L_2 \sim_{L\Pi} L_3)$ and $L_1 \succ_{LN} L_3$ $(resp.\; L_1 \succ_{L\Pi} L_3)$.

Note that contrary to U_{pes} and U_{opt}, which are purely ordinal, possibilistic likely dominance can be defined in the ordinal setting or the numerical setting of possibility theory.

1.6 Conclusion

As we have seen in this chapter, possibilistic decision theory allows us to overcome some weakness of classical decision theory. It offers a flexible and simple framework to represent all the decision maker behavior and all kinds of uncertainty. We especially detailed main decision criteria based on possibility theory namely optimistic and pessimistic qualitative utilities, binary possibilistic utility, possibilistic Choquet integrals and possibilistic likely dominance.

In next chapter, we will present the foundation of sequential decision problems in possibilistic decision trees where decision maker should choose a sequence of decisions. We aim to optimize strategies in the case of different possibilistic decision criterion presented in this Chapter.

Chapter 2

Optimizing strategies in possibilistic decision trees

2.1 Introduction

In the first chapter, we have been interested in one-stage decision problem that presents as a situation in which the decision maker has to take a single decision. Generally, the decision maker should choose multiple decisions over time i.e. the state of nature in step $t+1$ depends necessary on the consequence of the action executed at t. Indeed, in sequential decision making, a strategy is a function that assigns a decision to each state. Many graphical models have been proposed to represent these latter problems, such as influence diagrams [19], Markov decision processes [7] and decision trees [26] etc. The choice of a model depends on the decision problem and available data. Markov decision processes can represent sequential decision problems say repeatedly, and influence diagrams are rather implicit and compact. In this work, we are interested in the formalism of decision trees since they allow an explicit representation of a sequential decision problem.

Given a decision tree modeling a sequential decision problem, the question is how to find an optimal strategy w.r.t. a possibilistic decision criterion shown in the previous chapter. This chapter reviews main results presented in [4].

This chapter is organized as follows: Section 2.2 formally defines standards decision trees. Section 2.3 gives an overview on possibilistic decision trees. Then, Section 2.4 reviews existing algorithms to find optimal strategies with a reasonable complexity. Lastly, Section 2.5 presents the optimization problem w.r.t. each class of decision criteria.

2.2 Decision trees

A classical way for representing sequential decision problems is decision trees proposed by Raiffa in 1968 [26]. This framework is a graphical model that encodes explicitly the structure of the problem by representing each possible scenario by a path from the root to the leaves of the tree.

A decision tree is composed of a graphical component and a numerical one as detailed below.

- **The graphical component**

  A decision tree $\mathcal{T}$ is composed of a set of nodes $\mathcal{N}$ and a set of edges $\mathcal{E}$. The set $\mathcal{N}$ contains three kinds of nodes:

  - $D = \{D_0, ..., D_m\}$ is the set of decision nodes (represented by rectangular boxes). The labeling of the node is supposed to be in accordance with the temporal order, i.e. if D_i is a descendant of D_j, then $i > j$. The root of the tree is necessarily a decision node.
  - $C = \{C_1, ..., C_n\}$ is the set of chance nodes (represented by circles).
  - $LN = \{LN_1, ..., LN_k\}$ is the set of leaves, also called utility leaves: $\forall LN_i \in \mathcal{LN}$, $u(LN_i)$ is the utility of being eventually in node LN_i.

  For any $X_i \in N$, $Succ(X_i) \subseteq N$ denotes the set of its children. In addition, for any $D_i \in D$, $Succ(D_i) \subseteq C$ i.e. after a decision node we retrieve a chance node. For any $C_i \in C$, $Succ(C_i) \subseteq LN \cup D$ i.e. $Succ(C_i)$ is indeed the set of outcomes of the action C_i either a leaf node is observed, or a decision node is reached (and then a new action should be executed).

  In fact, the sequence or the path in which we visit the nodes from the root to the leaves corresponds to the complete sequence of observations and decisions and that is what is called scenario.

- **The numerical component**

  The numerical component of decision trees assigns utilities to leaves nodes and valuates the edges outgoing from chance nodes i.e. a link from a chance node (to decision node or leave) is labeled with the feasibility of the potential state.

  Since leaf nodes in $\mathcal{LN}$ are labeled with utility degrees, then each chance node $C_i \in C$ can be seen as a simple lottery (for the rightmost chance nodes) or as a compound lottery (for the inner chance nodes). The lottery associated to C_i is denoted by L_{C_i}.

For instance, in classical probabilistic decision trees the uncertainty concerning $C_i \in C$ is represented by a conditional probability distribution $P(C_i|Pa(C_i))$ where $Pa(C_i)$ is the set of parents of C_i. In addition, to each rightmost chance node $C_i \in C$ we can associate a probabilistic lottery relative to its outcomes.

Example 2.1. *Figure 2.1 shows an example of a probabilistic decision tree with six chance nodes* $C = \{C_1, C_2, C_3, C_4, C_5, C_6\}$*, three decision nodes* $D = \{D_0, D_1, D_2\}$ *and* $LN = U = \{0.1, 0.2, 0.3, 0.4, 0.6, 0.8\}$.

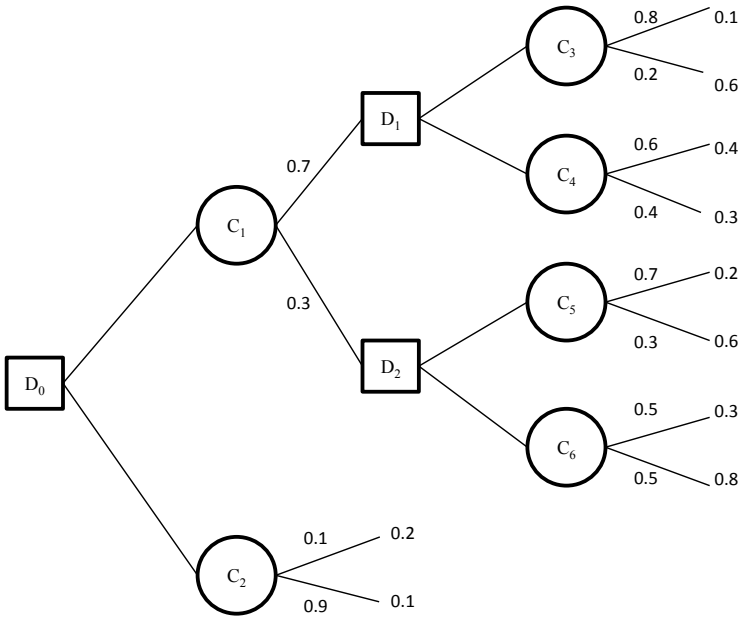

Figure 2.1: Example of a probabilistic decision tree.

2.3 Possibilistic decision trees

Possibilistic decision trees are an adaptation of standard decision trees [26] in the context of possibility theory [12]. They have the same graphical component as standards ones with a possibilistic labeling.

More precisely, the uncertainty pertaining to the possible outcomes of each $C_i \in \mathcal{C}$, is represented by a *conditional possibility distribution* π_i on $Succ(C_i)$, such that $\forall N \in Succ(C_i)$, $\pi_i(N) = \Pi(N|path(C_i))$ where $path(C_i)$ denotes all the value assignments to chance and decision nodes on the path from the root to C_i. In addition, a utility is assigned to each leaf nodes

which can be numerical (e.g. currency gain) or ordinal (e.g. satisfaction) according to the decision criterion.

Solving a decision tree amounts at building a *strategy* that selects an action (i.e. a chance node) for each reachable decision node. Formally, we define a strategy as a function δ from $\mathcal{D}$ to $\mathcal{C} \cup \{\perp\}$. $\delta(D_i)$ is the action to be executed when a decision node D_i is observed. $\delta(D_i) = \perp$ means that no action has been selected for D_i (because either D_i cannot be reached or the strategy is partially defined). Admissible strategies should be:

- *sound*: $\forall D_i \in \mathcal{D}, \delta(D_i) \in Succ(D_i) \cup \{\perp\}$.
- *complete*: (i)$\delta(D_0) \neq \perp$ and (ii)$\forall D_i$ s.t. $\delta(D_i) \neq \perp, \forall N \in Succ(\delta(D_i))$, either $\delta(N) \neq \perp$ or $N \in \mathcal{LN}$.

Let Δ be the set of sound and complete strategies that can be built from the decision tree. Any strategy in Δ can be viewed as a connected subtree of the decision tree whose arcs are of the form $(D_i, \delta(D_i))$ i.e. there being exactly one decision arc left at each decision node. Each strategy is a compound lottery and can be reduced into an equivalent simple one by adapting lottery reduction (Equation 1.3) to strategies.

Formally, the composition of lotteries will be applied from the leaves of the strategy to its root in a recursive manner. This ensures a polytime computation of the reduced lottery $Reduction(\delta)$ since operators $\max$ and $*$ (resp $\max$ and $\min$) are polytime. To define the notion of optimality, we should be able to compare different strategies, to this end we first have to select a decision criterion (O), then a strategy $\delta \in \Delta$, is said to be optimal w.r.t. $\succeq_O$ iff:

$$\forall \delta' \in \Delta, Reduction(\delta) \succeq_O Reduction(\delta'). \tag{2.1}$$

Example 2.2. *Figure 2.3 shows an example of a possibilistic decision tree.*

This decision tree encodes 5 strategies i.e. Δ=$\{\delta_1, \delta_2, \delta_3, \delta_4, \delta_5\}$ *where:*

- $\delta_1 = ((D_0, C_1), (D_1, C_3), (D_2, C_5))$;
- $\delta_2 = ((D_0, C_1), (D_1, C_3), (D_2, C_6))$;
- $\delta_3 = ((D_0, C_1), (D_1, C_4), (D_2, C_5))$;
- $\delta_4 = ((D_0, C_1), (D_1, C_4), (D_2, C_6))$;
- $\delta_5 = ((D_0, C_2))$.

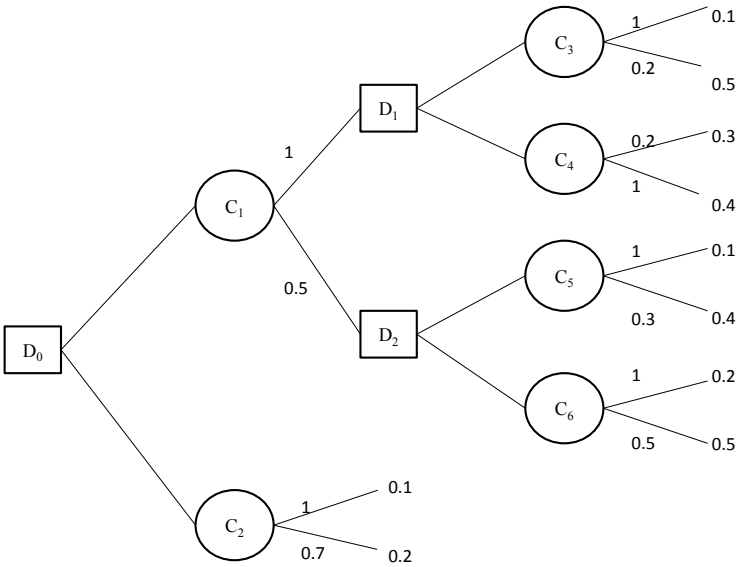

Figure 2.2: Example of possibilistic decision tree.

The strategy δ_1 *corresponds to the lottery* $\langle 1/L_{C_3}, 0.5/L_{C_5}\rangle$ *where* $L_{C_3} = \langle 1/0.1, 0.2/0.5\rangle$ *and* $L_{C_5} = \langle 1/0.1, 0.3/0.4\rangle$.

- *For* $\otimes = \min$ *we have:* $Reduction(\delta_1) = \langle 1/0.1, 0.3/0.4, 0.2/0.5\rangle$.
- *For* $\otimes = *$ *we have:* $Reduction(\delta_1) = \langle 1/0.1, 0.12/0.4, 0.2/0.5\rangle$.

2.4 Evaluation of possibilistic decision trees

Given a possibilistic decision tree, we are interested in the selection of an optimal strategy δ^* within Δ w.r.t. a decision criterion O. Formally, $\forall \delta_i \in \Delta$ we have $\delta^* \succeq_O \delta_i$ (i.e. δ^* is preferred to any strategy $\delta_i \in \Delta$ w.r.t. a decision criterion O). The set of potential strategies is combinatorial (i.e., its size increases exponentially with the size of the tree). Then, finding optimal strategy in possibilistic decision tree via an exhaustive enumeration of Δ is a highly computational task. In [4] N. Ben Amor and al. show that in a decision tree with m decision nodes and a branching factor equal to 2, the number of potential strategies is in $O(2^{\sqrt{m}})$. Then, the determination of an optimal strategy for a given representation and a given decision criterion is then an algorithmic issue in itself.

This section presents existing algorithms to find optimal strategies w.r.t. a decision criterion O.

2.4.1 Strategy optimization problem in possibilistic decision trees

For any possibilistic decision criteria presented in Section 1.5, the corresponding decision problem can be defined as follows:

Definition 2.1. *($DT - OPT - O$): Strategy optimization w.r.t. a possibilistic decision criterion O in a possibilistic decision tree.*

- *INSTANCE: A possibilistic decision tree $\mathcal{T}$, a constant lottery θ. A possibilistic decision criterion O.*
- *QUESTION: Does there exist a strategy $\delta \in \Delta$ such that $Reduction(\delta) \succeq_O \theta$*

For instance $DT - OPT - U_{opt}$ (resp. $DT - OPT - U_{pes}$, $DT - OPT - PU$, $DT - OPT - L\Pi$ $DT - OPT - LN$, $DT - OPT - Ch_N$ and $DT - OPT - Ch_\Pi$) corresponds to the optimization of the optimistic qualitative utility (resp. U_{pes}, PU, $L\Pi$, LN, Ch_N and Ch_Π). Each of these decision problem will be studied in what following.

It is necessary to note that the complexity of ($DT - OPT - O$) and the corresponding evaluation may be polynomial or NP-hard depending on some properties of the decision criterion O i.e. the monotonicity (or weak monotonicity) and the transitivity properties.

Definition 2.2. *A preference order is said to be weakly monotonic iff whatever L, L', L'', whatever (α, β) such that $max(\alpha, \beta) = 1$:*

$$L \succeq_O L' \;\Rightarrow\; \langle \alpha/L, \beta/L'' \rangle \succeq_O \langle \alpha/L', \beta/L'' \rangle \tag{2.2}$$

This property states that the combination of L (resp. L') with L'', does not change the initial order induced by O between L and L' - this allows DynProg to decide in favor of L or L' before considering the compound decision.

Recall that:

Definition 2.3. *A preference order is said transitive iff whatever L, L', L'':*

$$L \succ_O L' \;\; and \;\; L' \succ_O L'' \;\Rightarrow\; L \succ_O L''; \tag{2.3}$$

$$L \sim_O L' \;\; and \;\; L' \sim_O L'' \;\Rightarrow\; L \sim_O L''. \tag{2.4}$$

2.4.2 Dynamic programming

When the decision criterion O verifies the crucial properties of *weak monotonicity* and *transitivity*, an optimal strategy can be computed in polytime with respect to the size of the tree using a recursive method of *Dynamic programming* (DynProg) called backward search method or backward induction method (see Algorithm 1) [4, 3]. DynProg builds the best strategy backwards, optimizing the decisions from the leaves of the tree to its root. Note that this algorithm does not generate all the best strategies but returns only one among them.

Algorithm 1: DynProg(N:Node, δ: Strategy)

```
Data: O is the criterion to optimize.
Result: A lottery L
begin
    // Initialisation
    for i ∈ {1, ..., n} do L[u_i] ← 0;
    // Leaves
    if N ∈ LN then L[u(N)] ← 1;
    // Chance nodes
    if N ∈ C then
        // Reduce the compound lottery
        foreach Y ∈ Succ(N) do
            L_Y ← DynProg(Y, δ);
            for i ∈ {1, ..., n} do
(Line 1)        L[u_i] ← max(L[u_i], (⊗(π_N(Y), L_Y[u_i])));
    // Decision nodes
    if N ∈ D then
        // Choose the best decision
        Y* ← Succ(N).first;
        foreach Y ∈ Succ(N) do
            L_Y ← DynProg(Y, δ);
(Line 2)    if L_Y ≻_O L_Y* then Y* ← Y;
        δ(N) ← Y*;
        L ← L_Y*;
    return L;
```

The principle of backwards reasoning procedure is depicted in a recursive manner by Algorithm 1 for a preference order $\succeq_O$ over possibilistic lotteries. In this algorithm $L[u_i]$ is the possibility degree to have the utility u_i in the lottery L, $Succ(N).first$ is the first node in the set

of successors of N i.e. $Succ(N)$. $\otimes(Line(1))$ is the min operator for the ordinal setting and the product operator for the numerical setting.

The principle of DynProg is as follows: when each chance node is reached, an optimal sub-strategy is built for each of its children, these sub-strategies are combined w.r.t. their possibility degrees, and the resulting compound strategy is reduced: we get an equivalent simple lottery, representing the current optimal sub-strategy. When a decision node X is reached, a decision Y^* leading to a sub-strategy optimal w.r.t. $\succeq_O$ is selected among all the possible decisions $Y \in Succ(X)$, by comparing the simple lotteries equivalent to each sub-strategy.

Algorithm 1, Line (2) performs the comparison of two simple lotteries according to the criterion O to optimize. This procedure crosses each edge in the tree only once. When the comparison of simple lotteries by $\succeq_O$ (Algorithm 1, Line (2)) and the reduction operation (Algorithm 1, Line (1)) can be performed in polytime, its complexity is polynomial w.r.t. the size of the tree.

2.4.3 Branch and Bound

If the decision criterion is not monotonic, Branch and bound algorithm [4, 2] may compute an optimal strategy. It takes as argument a partial strategy δ and an upper bound of decision criterion's value O_μ of the best extension of the partial strategy (μ is either the necessity or the possibility measure, depending on the criterion that is to be optimized). It returns the decision criterion value of the best strategy found so far, μ^{opt}. As initial value for δ we retain the empty strategy $(\delta(Di) = \perp, \forall Di)$. For μ^{opt}, we can choose the strategy provided by the DynProg algorithm: indeed, even not necessarily providing an optimal strategy, this algorithm generally provides a good one.

At each step, the current partial strategy, δ, is developed by the choice of an action for some unassigned decision node. When several decision nodes are candidate, the one with the minimal rank (i.e. the former one according to the temporal order) is developed first.

The recursive procedure backtracks when either the current strategy is complete (then μ^{opt} and O_μ^{opt} may be updated) or proves to be worse than the current μ^{opt} in any case. In order to get an upper bound of the Choquet value of the best completion of δ, we call a function $Lottery_\mu$ ($\mu = N$ or Π) that computes a lottery that is better than all the complete strategies that extend δ when applied from a node given in argument and use the decision criterion value of this lottery as an upper bound of the Choquet value of the best strategy compatible with δ. Whenever $O_\mu(Lottery_\mu(\delta)) \leq O_\mu^{opt}$ the algorithm will backtrack, yielding the choice of another action for the last decision node considered. Moreover when δ is complete, $Lottery_\mu(\delta)$ returns $L(D_0, \delta)$; the upper bound is equal to the Choquet value when computed for a complete strategy. $Lottery_\mu(\delta)$ actually builds a lottery that *overcomes* with respect of μ all the possible extensions

of δ.

Algorithm 2: BB(δ: a (partial) strategy, O_μ : its corresponding value) with $\mu = N$ or $\mu = \Pi$

Result: O_μ^{opt} i.e. the value of δ^{opt}, the best strategy found so far

```
begin
    if δ = ∅ then D_pend ← {D_0} ;
    else
        D_pend ← {D_i ∈ 𝒟 s.t. δ(D_i) = ⊥ and ∃D_j, δ(D_j) ≠ ⊥ and D_i ∈ Succ(δ(D_j))};
    // Is δ a complete strategy ?
    if 𝒟_pend = ∅ then
        if O_μ^δ > O_μ^opt then
            δ^opt ← δ;
            O_μ^opt ← O_μ^δ;
    else
        D_next ← arg min_{D_i ∈ D_pend} i ;
        foreach C_i ∈ Succ(D_next) do
            δ(D_next) ← C_i;
            Eval ← O_μ(Lottery_μ(D_0, δ));
            if Eval > O_μ^opt then O_μ^opt ← BB(δ, O_μ^δ) ;
    return O_μ^opt;
```

2.5 Optimizing possibilistic decision criteria

Since the EU model, relative to a probabilistic representation of uncertainty (see Section 1.2), satisfies the monotonicity property. Thus for standard probabilistic decision trees [26], where the goal is to maximize expected utility, an optimal strategy can be computed in polytime with respect to the size of the tree using DynProg in the probabilistic framework [6].

What about possibilistic decision trees where O can be any possibilistic decision criterion (see Section 1.5) i.e. U_{opt}, U_{pes}, PU, $L\Pi$, LN, Ch_N and Ch_Π ?

In this section, we present a theoretical study on the problem of finding an optimal strategy depending on the optimization criteria properties.

2.5.1 Optimizing qualitative possibilistic utilities

Qualitative possibilistic Utilities (U_{opt}, U_{pes} and PU) respect the key properties of transitivity and weak monotonicity for both $\otimes = \min$ and the quantitative case $\otimes = *$ [13, 17].

Just like in the case of (EU), DynProg (Algorithm 1) may be used and an optimal strategy, with respect to U_{opt}, U_{pes}, or PU, can be built in polytime for both the qualitative case ($\otimes = \min$) and the quantitative case ($\otimes = *$) [3].

Example 2.3. *Let us consider the decision tree in Figure 2.3. Principle steps for the evaluation of this decision tree using DynProg (Algorithm 1) in the case of U_{opt} criterion are as follows:*

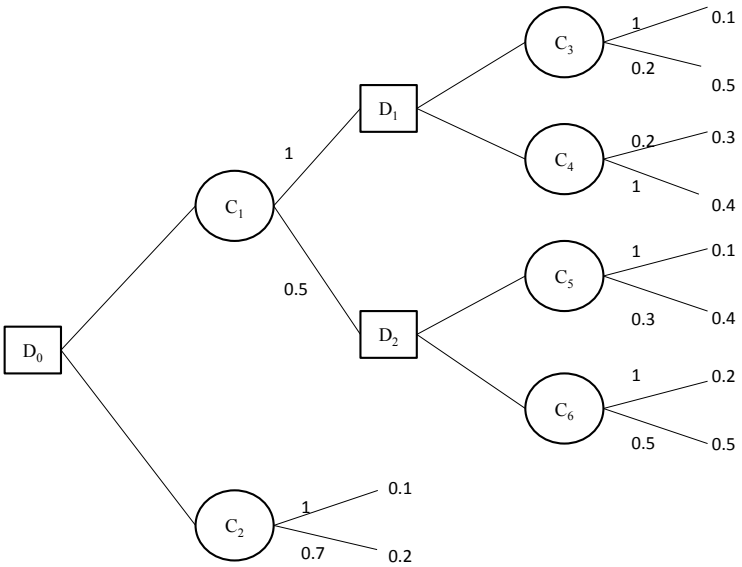

Figure 2.3: Possibilistic decision tree for Example 2.3.

- *Initially, we have $\delta = \emptyset$ and $N = D_0$ with $Succ(D_0) = \{C_1, C_2\}$.*
- *For $Y = C_1$, $L_{C_1} = DynProg(C_1, \delta)$. Since $Succ(C_1) = \{D_1, D_2\}$, we have $Y = D_1$ and $Y = D_2$.*
- *For $Y = D_1$, we have $L_{D_1} = DynProg(D_1, \delta)$ and $Succ(D_1) = \{C_3, C_4\}$:*
  - *If $Y = C_3$ then $L_{C_3} = \langle 1/0.1, 0.2/0.5, 0/0.3, 0/0.4, 0/0.2 \rangle$ and $U_{opt}(L_{C_3}) = max(min(1, 0.1), min(0.2, 0.5), min(0, 0.3), min(0, 0.4)) = 0.2$.*
  - *If $Y = C_4$ then $L_{C_4} = \langle 0/0.1, 0/0.5, 0.2/0.3, 1/0.4, 0/0.2 \rangle$ and $U_{opt}(L_{C_4}) = 0.4$.*

$\Rightarrow$ *Since* $U_{opt}(L_{C_4}) > U_{opt}(L_{C_3})$, *So* $Y^* = C_4,\ \delta(D_1) = \{C_4\}$
and $L_{D_1} = \langle 0/0.1, 0/0.5, 0.2/0.3, 1/0.4, 0/0.2\rangle$.

- *For* $Y = D_2$, *we have* $L_{D_2} = DynProg(D_2, \delta)$ *and* $Succ(D_2) = \{C_5, C_6\}$:

  - *If* $Y = C_5$ *then* $L_{C_5} = \langle 1/0.1, 0/0.5, 0/0.3, 0.3/0.4, 0/0.2\rangle$ *and* $U_{opt}(L_{C_5}) = 0.4$.
  - *If* $Y = C_6$ *then* $L_{C_6} = \langle 0/0.1, 0.5/0.5, 0/0.3, 0/0.4, 0.1/0.2\rangle$ *and* $U_{opt}(L_{C_6}) = 0.5$.

  $\Rightarrow$ *Since* $U_{opt}(L_{C_6}) > U_{opt}(L_{C_5})$, *So* $Y^* = C_6,\ \delta(D_2) = \{C_6\}$
  and $L_{D_2} = \langle 0/0.1, 0.5/0.5, 0/0.3, 0/0.4, 0.1/0.2\rangle$.

- $L_{C_1} = \langle 1/L_{D_1}, 0.5/L_{D_2}\rangle = \langle 1/0.1, 0.5/0.5, 0.2/0.3, 1/0.4, 0.1/0.2\rangle$
  and $U_{opt}(L_{C_1}) = 0.5$.

- *For* $Y = C_2,\ L_{C_2} = DynProg(C_2, \delta)$, *we have* $L_{C_2} = \langle 1/0.1, 0/0.5, 0/0.3, /0.4, 0.7/0.2\rangle$
  and $U_{opt}(L_{C_2}) = 0.2$.

  Since $U_{opt}(L_{C_1}) > U_{opt}(L_{C_2})$, *so* $Y^* = C_1,\ \ \delta(D_0) = C_1$
  and $\delta^* = \{(D_0, C_1), (D_1, C_4), (D_2, C_6)\}$ *with* $U^*_{opt} = 0.5$.

$\Rightarrow$ *The optimal strategy returned is:* $\delta^* = ((D_0, C_1), (D_1, C_4), (D_2, C_6))$.

2.5.2 Optimizing possibilistic Choquet integrals

Contrary to qualitative utilities, the situation is much lesser comfortable for Choquet integrals (either Ch_N or Ch_Π). The point is that the possibilistic Choquet Integrals (as many other Choquet integrals) does not satisfy the monotonicity principle neither in the ordinal setting nor in the numerical one.

In a recent work [4], N. Ben Amor et al. have shown that optimizing possibilistic Choquet integrals is intractable in the general case. Even though, it is impossible to solve the problem in polytime, to make the question polynomial, [4] found a solution: it is the case for binary decision trees, i.e. when the leaves involve two levels of utility. To optimize possibilistic Choquet integrals, we can proceed by implicit enumeration via a Branch and bound algorithm (BB) -see Algorithm 2 [4, 2].

As a consequence, DynProg cannot guarantee an optimal policy. This is illustrated by the following counterexample.

Counter-Example 2.1. *Consider the possibilistic decision tree in figure 2.4 with* $O = Ch_N$ *and* $\otimes = \min$.

- *For* D_1*, we Compute* $Ch_N(L_{C_1}) = 0.653$ *and* $Ch_N C_2 = 0.650$ *so* $L_{C_1} \succ_{Ch_N} L_{C_2}$*: DynProg selects* L_{C_1}*;*

- *For* D_0*, DynProg selects the strategy* $\delta_1^* = ((D_0, C_0), (D_1, C_1), (D_2, C_3))$*. Note that* $\delta_2^* = ((D_0, C_0), (D_1, C_2), (D_2, C_3))$ *is the alternative strategy, thus, using Equation 1.3 we have:* $Reduction(L_{\delta_1^*}) = \langle 0.2/0, 0.5/0.51, 1/1 \rangle$ *and* $Reduction(L_{\delta_2^*}) = \langle 0.1/0, 0.55/0.5, 1/1 \rangle$.

  Indeed, we can check that $Ch_N(Reduction(L_{\delta_1^*})) = 0.653 < Ch_N(Reduction(L_{\delta_2^*})) = 0.675$*. This contradicts the monotonicity property and then DynProg return a sub-strategy. Lastly, the optimal strategy for this example is* $\delta_2^* = ((D_0, C_0), (D_1, C_2), (D_2, C_3))$ *and its Choquet value is 0.675.*

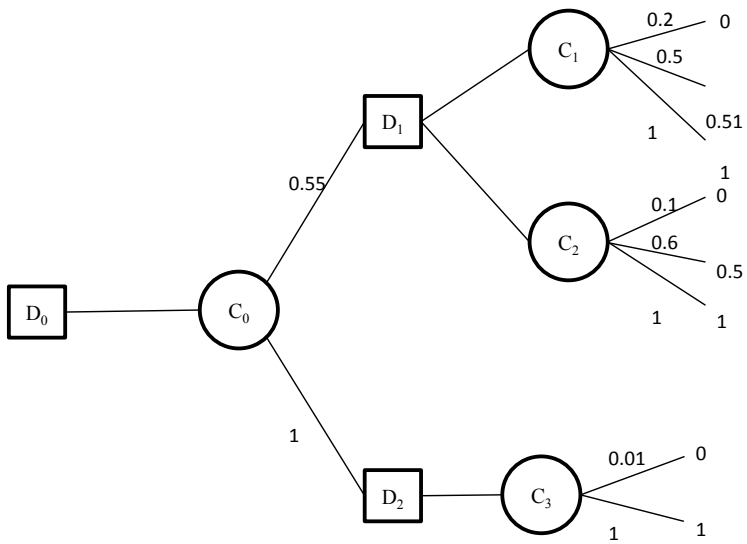

Figure 2.4: Decision tree of Counter-example 2.1.

2.5.3 Optimizing possibilistic likely dominance

Because this criterion is quasi-transitive (see Section 1.5.3), classical DynProg could not be applied. In the counterexamples below, the DynProg does not provide an optimal strategy when optimizing likely dominance criteria i.e. $L\Pi$ and LN.

Counter-Example 2.2. *[$O = L\Pi$]*
Consider now the possibilistic decision tree in figure 2.5 with $O = L\Pi$ and $\otimes = \min$. Using Equation 1.14, we can check that $L_{C_3} \sim_{L\Pi} L_{C_4}$, $L_{C_5} \sim_{L\Pi} L_{C_6}$, $L_{C_4} \sim_{L\Pi} L_{C_6}$ and $L_{C_3} \succ_{L\Pi} L_{C_2}$.

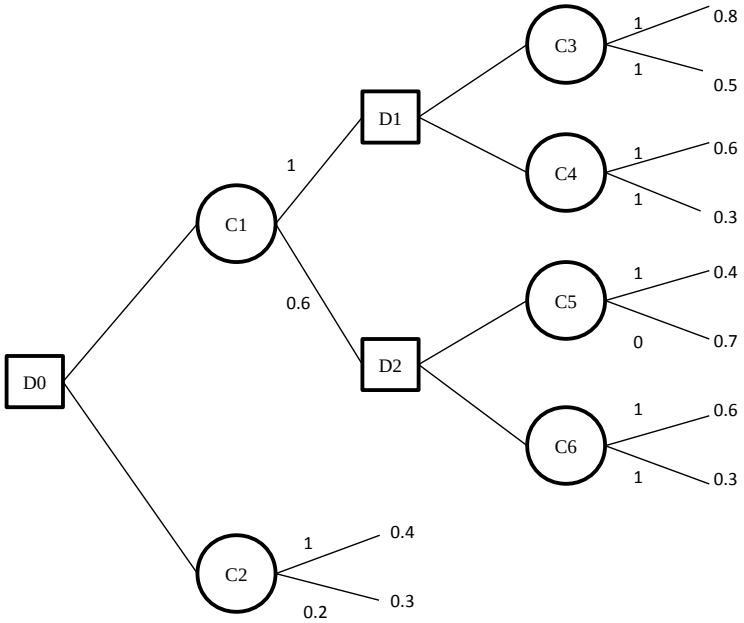

Figure 2.5: Decision tree of Counter-example 2.2.

If we apply DynProg we have:

- *For D_1, $L_{C_3} \sim_{L\Pi} L_{C_4}$: suppose that DynProg selects randomly L_{C_4};*
- *For D_2, $L_{C_5} \sim_{L\Pi} L_{C_6}$: suppose that DynProg selects randomly L_{C_6};*
- *For C_1 we have $L_{C_1} = \langle 1/L_{C_4}, 0.6/L_{C_6} \rangle$;*
- *For D_0, DynProg compares L_{C_1} and L_{C_2}. We can check that $L_{C_1} \sim_{L\Pi} L_{C_2}$, DynProg randomly selects one of them. Suppose that it retains L_{C_2}, thus, according to DynProg, the best strategy is $\delta^*_{DynProg} = ((D_0, C_2))$ while it is easy to check, with an exhaustive enumeration of Δ w.r.t. $L\Pi$, that the real optimal strategy is $\delta^* = ((D_0, C_1), (D_1, C_3), (D_2, C_6))$.*

Moreover, the lottery associated to the strategy constructed by DynProg is dominated by the lottery of δ^* w.r.t. $L\Pi$. Therefore, the strategy returned by DynProg may be suboptimal, dominated and completely unusable.

Counter-Example 2.3. *[$O = LN$]*
Let us Consider now the possibilistic decision tree in figure 2.5 with $O = LN$ and $\otimes = \min$. Using Equation 1.13, we can check that $L_{C_3} \sim_{LN} L_{C_4}$, $L_{C_3} \sim_{LN} L_{C_2}$, $L_{C_5} \succ_{LN} L_{C_6}$ and $L_{C_4} \succ_{LN} L_{C_2}$. If we apply DynProg we have:

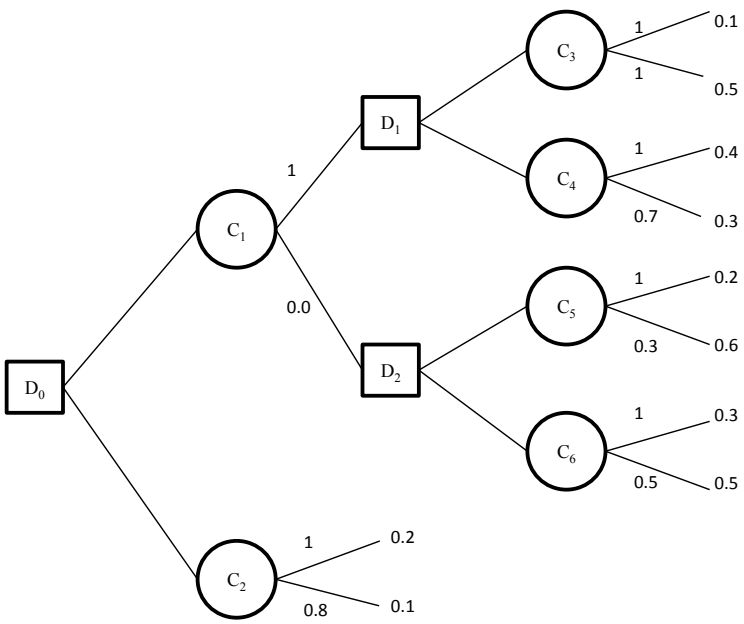

Figure 2.6: Decision tree of Counter-example 2.3.

- *For D_1, $L_{C_3} \sim_{LN} L_{C_4}$: suppose that DynProg selects randomly L_{C_3};*
- *For D_2, $L_{C_5} \succ_{LN} L_{C_6}$: DynProg selects L_{C_5};*
- *For C_1 we have $L_{C_1} = \langle 1/L_{C_3}, 0.0/L_{C_5} \rangle = \langle 1/L_{C_3} \rangle$;*
- *For D_0, DynProg compares L_{C_1} and L_{C_2}. We can check that $L_{C_1} \sim_{LN} L_{C_2}$ (we have $L_{C_3} \sim_{LN} L_{C_2}$), DynProg randomly selects one of them. Suppose that it retains L_{C_2}, thus the best strategy, according to DynProg, is $\delta^*_{DynProg} = ((D_0, C_2))$ while it is easy to check, with an exhaustive enumeration of Δ w.r.t. LN, that the real optimal strategy is $\delta^* = ((D_0, C_1), (D_1, C_4), (D_2, C_5))$.*

The problem is that DynProg takes back one lottery at a time which makes it inappropriate in situations of equivalent lotteries. More precisely, this algorithm may leave an equivalent lottery that dominates other lotteries.

2.6 Conclusion

In this chapter we have recalled the foundations of sequential decision problems i.e. optimizing strategies in possibilistic decision trees. After defining standard decision trees, we have proposed a brief overview on possibilistic decision trees which offer a natural and explicit model to handle possibilistic sequential decision problems. Then, we have studied the optimization problem where we aim to find an optimal strategy i.e. maximizing decision maker utilities w.r.t. a decision criterion. The simplest solving method is exhaustive enumeration of all possible strategies to find the best one. However, given the large number of strategies in the decision tree, this method is intractable. As a solution, we have to use a polytime algorithm according to the decision criterion.

For possibilistic decision criteria that satisfy the monotonicity and transitivity properties, we can apply Dynamic programming to find an optimal strategy in polytime. This is particularly true for possibilistic qualitative utilities. For possibilistic Choquet integrals, the optimizing problem is NP-hard but it can be solved by a Branch and bound algorithm. Concerning possibilistic likely dominance criteria which are quasi-transitive but not fully transitive, we have shown that Dynamic programming does not always succeed to build an optimal strategy. Dynamic programming can nevertheless be modified so as to keep a judicious *subset* of all the best (but possibly indifferent) strategies for the current subtree. The aim of the following chapter is to present an adaptation of this algorithm, namely Extended Dynamic Programming, proposed in [4].

Chapter 3

Extended Dynamic Programming

3.1 Introduction

As mentioned in the previous chapter, the preference order induced by possibilistic likely dominance ($L\Pi$ and LN) is not fully transitive. Then, even if the monotonicity principle is satisfied, the application of the standard dynamic programming can lead to suboptimal strategies. As a solution, we propose to use the algorithm of *Extended dynamic programming* [4], which obeys the same principle of Dynamic programming.

In this chapter, we describe the Extended dynamic programming algorithm and its strengths regarding Dynamic programming. Thus, Section 3.2 presents the basic ideas behind Extended dynamic programming also the algorithm and its complexity. Section 3.3 shows a detailed example. Subsection 3.4 provides a recapitulation of all decision criteria optimization. Appendix 1 presents main proofs of propositions presented in this chapter.

3.2 Extended dynamic programming algorithm

For possibilistic likely dominance and in the case of equivalent lotteries, DynProg algorithm can return a sub-optimal strategy since it makes a random choice. This failure of DynProg is due to the quasi-transitivity of the decision criteria.

In order to propose an evaluation algorithm for decision trees in the case of possibilistic likely dominance, we use an *Extended dynamic programming* (ExtDynProg) keeping the same principle of classical DynProg algorithm: the optimal solution is built by optimal sub-solutions i.e.

from the leaves of the tree to its root (see Algorithm 3). It stores the optimal choices themselves in such a manner that allows keeping a judicious subset of all the best (but possibly indifferent) strategies for each decision node [4]. The recursive procedure thus returns a *set* S of simple lotteries and their associated strategies: dominated strategies are removed by a call of a pruning function (see Algorithm 4), thanks to the monotonicity principle. At each chance node, a pointwise composition of the lotteries returned by the recursive calls is performed: this leads to a set of reduced and non-dominated lotteries.

The difficulty with this type of algorithms it that the size of the set of lotteries handled may grow exponentially. We show in the following that it is possible to keep it within polynomial bounds for possibility-based likely dominance ($L\Pi$) and necessity-based likely dominance (LN).

3.2.1 Comparing lotteries by $L\Pi$

When $O = L\Pi$, the comparison of two lotteries only depends on the relative position of their cores. Formally, for any lottery L, let $L_{=\alpha} = \{u_i, L[u_i] = \alpha\}$; the core of L corresponds to $L_{=1}$. It is easy to show that $L \succeq_{L\Pi} L'$ iff $\max(L_{=1}) \geq \min(L'_{=1})$. This allows us to replace any simple lottery L built during the computation by a simple lottery where any utility degree different from $\max(L_{=1})$ and $\min(L_{=1})$ receives a possibility degree of 0. This lottery is called by the *core lottery* of L and denoted by L^{core}; namely:

$$L^{core} = \langle 1/\min(L_{=1}), 1/\max(L_{=1})\rangle. \tag{3.1}$$

In [4] it has been proved that:

Proposition 3.1. [1] *$\forall L, L' \in \mathcal{L}, L \succeq_{L\Pi} L' \Leftrightarrow L^{core} \succeq_{L\Pi} L'$ and $L' \succeq_{L\Pi} L \Leftrightarrow L' \succeq_{L\Pi} L^{core}$.*

Example 3.1. *Let $L = \langle 0.6/0.1, 1/0.5, 1/0.6\rangle$ and $L' = \langle 0.3/0.1, 0.8/0.2, 0.8/0.3, 1/0.4\rangle$ be two simple lotteries. We have:*

- *$L_{=1} = \{0.5, 0.6\}$ and $L'_{=1} = \{0.4\}$;*
- *$L^{core} = \langle 1/0.5, 1/0.6\rangle$ and $L'^{core} = \langle 1/0.4\rangle$.*

It holds, in both numerical and ordinal settings, that:

- *$\Pi(L \geq L') = 1$ and $\Pi(L' \geq L) = 0.8 \;\Rightarrow\; L \succeq_{L\Pi} L'$;*
- *$\max(L^{core}) = 0.6 \geq \min(L'^{core}) = 0.4 \;\Rightarrow\; L \succeq_{L\Pi} L'$;*

[1] Proofs are in appendix

- $\Pi(L^{core} \geq L'^{core}) = 1$ *and* $\Pi(L'^{core} \geq L^{core}) = 0 \;\; \Rightarrow \;\; L^{core} \succeq_{L\Pi} L'^{core}$*;*
- $L \succeq_{L\Pi} L'$ *and it is easy to check that* $L^{core}) = 0.6 \succeq_{L\Pi} L'$.

A first consequence of proposition 3.1 is that replacing any simple lottery by its core lottery during the computation does not change the set of optimal strategies of the problem. A second consequence is that if $L^{core} = L'^{core}$, then for any L'', $L \succeq_{L\Pi} L'' \Leftrightarrow L' \succeq_{L\Pi} L''$ and $L'' \succeq_{L\Pi} L \Leftrightarrow L'' \succeq_{L\Pi} L'$ which implies that lotteries sharing the same core form an equivalence class. We shall then remove from S the pairs (L^{core}, δ) such that there exist in the set another pair (L'^{core}, δ'): since leading to the same core lottery, δ and δ' are equivalent w.r.t. $L\Pi$.

Thus the number of simple core lotteries is bounded by p^2, p being the number of different utility values labeling the leaves. As a consequence, the set S of lotteries handled by ExtDynProg (Algorithm 3) contains at most p^2 elements. The different loops on S are then also polynomially bounded: $O(p^4)$ is a trivial upper bound for the pruning function (assuming that the comparison of simple lotteries can be done in linear time) and the pointwise composition of the sets is bounded by $O(q.p^4)$, q being the number of children of the chance node.

The ExtDynProg procedure is thus polynomial in the size of the decision tree when $O = L\Pi$.

3.2.2 Comparing lotteries by LN

When $O = LN$, the principle is similar to $O = L\Pi$ but a little more tricky. The idea is to replace any simple lottery L by a lottery which keeps track of the bounds of the core (as it is the case with $L\Pi$ criterion) but also of the bound of the next cut of L, since only these two levels play a role in the $\succeq_{LN}$ comparison. We call this lottery the *2nd core lottery* of L and denote it L^{2nd}.

Formally, let $L_{=\alpha} = \{u_i, L[u_i] = \alpha\}$ and $\alpha^{2nd}(L) = \max\limits_{u_i, L[u_i]<1} L[u_i]$, then:

$$L^{2nd} = \begin{cases} \langle 1/min(L_{=1}), 1/max(L_{=1})\rangle \;\; if \;\; max(L_{=1}) \neq min(L_{=1}) \\ \langle 1/max(L_{=1}), \alpha^{2nd}(L)/min(L_{=\alpha^{2nd}}), \alpha^{2nd}(L)/max(L_{=\alpha^{2nd}})\rangle \;\; otherwise. \end{cases} \tag{3.2}$$

The second point corresponds to the case where L (and thus L^{2nd}) is unimodal with mode $\max(L_{=1}) = \min(L_{=1})$. In other terms, a 2nd core lottery is:

- either of the form $\langle 1/m, 1/M\rangle$ where $m = \min(L_{=1}) < M = \max(L_{=1})$ or;
- of the form $\langle \alpha/m, 1/u, \alpha/M\rangle$ where $u = \max(L_{=1})$, $\alpha = \alpha^{2nd}(L)$ and $m = \min(L_{=\alpha}) < M = \max(L_{=\alpha})$.

Algorithm 3: ExtDynProg(N : node)

```
Result: A set S of pairs (L, δ_L)
Data: O is the criterion to optimize (O = LΠ or O = LN).
Constant: The empty strategy δ_0
begin
    if N ∈ LN // Leaves
    then
        ∀u_i, L[u_i] ← 0;
        L[u(N)] ← 1;
        S ← {(L, δ_0)};
    if N ∈ C // Chance nodes
    then
        // Reduce the compound lotteries
        foreach Y_j ∈ Succ(N) do S_j ← ExtDynProg(Y_j) ;
        S ← S_1;
        for j ∈ {2, ..., |Succ(N)|} do
            Res ← ∅;
            foreach (L, δ) ∈ S, (L', δ') ∈ S_j do
                // Create a strategy δ'' and a lottery L''
                δ'' ← δ ∪ δ';
                ∀u_i, L''[u_i] ← 0;
                for i ∈ {1, ..., n} do
                    L''[u_i] ← max(L[u_i], (⊗(π_N(Y_j), L'[u_i])));
                    if O = LΠ then L'' ← L''^core ;
                    if O = LN then L'' ← L''^2nd ;
                    Res ← Res ∪ {(L'', δ'')};
            S ← Prune_O(Res);
    if N ∈ D // Decision nodes
    then
        // Build the set of possible decisions
        S ← ∅;
        S ← S ∪ ExtDynProg(Y)
        foreach (L, δ) ∈ S do δ(N) ← L;
        S ← Prune_O(S);
    return S;
```

Algorithm 4: Prune$_O$(S: a set of pairs $(L : strategy, \delta_L : lottery)$)

```
Data: O is the criterion to optimize (O = LΠ or O = LN)
Result: A pruned set Res of pairs (L', δ'_L)
begin
    Res ← S;
    foreach (L, δ) ∈ S do
        foreach (L', δ') ∈ S, δ' ≠ δ do
            if (L' ≻_O L) or (L is identical to L') then
                Remove (L, δ) from Res;
        if O = LN then
            if (L = ⟨1/u, α/m, α/M⟩) and (∃L' = ⟨1/u, α'/m', α'/M'⟩ ∈ S), with α' < α then
                Remove (L, δ) from Res;
    return Res;
```

Example 3.2. *Let* $L = \langle 0.6/0.1, 1/0.5, 1/0.6\rangle$ *and* $L' = \langle 0.3/0.1, 0.8/0.2, 0.8/0.3, 1/0.4\rangle$ *be two simple lotteries. Then:*

- $L^{core} = L^{2nd} = \langle 1/0.5, 1/0.6\rangle$*;*
- $L'^{core} = \langle 1/0.4\rangle$ *and* $L'^{2nd} = \langle 0.8/0.2, 0.8/0.3, 1/0.4\rangle$.

When two lotteries L and L' are to be compared w.r.t. LN, the decision is made at possibility degree 1 (i.e. the core level) as soon as one of them (say L) is not unimodal (and thus L^{2nd} is of the form $\langle 1/m, 1/M\rangle$). Indeed:

(i) If L'^{2nd} is of the form $\langle 1/m', 1/M'\rangle$ then $L \succ_{LN} L'$ when $m \geq M'$, $L' \succ_{LN} L$ when $m' \geq M$ and $L \sim_{LN} L'$ otherwise.
(ii) If L'^{2nd} is of the form $\langle \alpha'/m', 1/u', \alpha'/M'\rangle$ then $L \succ_{LN} L'$ when $m \geq u'$, $L' \succ_{LN} L$ when $u' \geq M$ and $L \sim_{LN} L'$ otherwise.

Let us now study the case where L and L' are unimodal, then L^{2nd} (resp. L'^{2nd}) can be written as $\langle \alpha/m, 1/u, \alpha/M\rangle$ (resp. $\langle \alpha'/m', 1/u', \alpha'/M'\rangle$).
(i) If $u \neq u'$, then only the modes are used to make the decision: either $u > u'$, and then $L \succ_{LN} L'$; or $u' > u$ and then $L' \succ_{LN} L$.
(ii) If $u = u'$, decision is made at level $\max(\alpha, \alpha')$; when $\alpha > \alpha'$, then either $u > M$ (and then $L' \succ_{LN} L$); or $u < m$ (and then $L \succ_{LN} L'$); or $M > u > m$ (and then $L \sim_{LN} L'$). Similar results hold for $\alpha' > \alpha$. $u' = u$ and $\alpha' = \alpha$, it is easy to show that the two lotteries are indifferent w.r.t. LN. When comparing L and L', only the information contained in L^{2nd} and L'^{2nd} matters. As stated by the following proposition [4]:

Proposition 3.2. $\forall L,\ L' \in \mathcal{L}, L \succeq_{LN} L' \ \Leftrightarrow\ L^{2nd} \succeq_{LN} L'$ *and* $L' \succeq_{LN} L \ \Leftrightarrow\ L' \succeq_{LN} L^{2nd}$.

This means that we can replace any simple lottery by its 2nd core lottery during the computation without changing the set of optimal strategies of the problem.

A second consequence of that proposition is that if $L^{2nd} = L'^{2nd}$ then for any L'', $L \succeq_{LN} L'' \Leftrightarrow L' \succeq_{LN} L''$ and $L'' \succeq_{LN} L \Leftrightarrow L'' \succeq_{LN} L'$. Lotteries sharing the same 2nd core form an equivalence class.

We shall then remove from S the pairs (L^{2nd}, δ) such that there exist in the set another pair (L'^{2nd}, δ') where $L^{2nd} = L'^{2nd}$: since leading to the same 2nd core lottery, δ and δ' are equivalent w.r.t. LN. The set S will thus contain pairs (L_i, δ_i) where each L_i is indifferent to others, while no pair of lotteries in S share the same 2nd core lottery. More formally [4]:

Proposition 3.3. *For any two indifferent second core lotteries of the form* $L = \langle 1/u, \alpha/m, \alpha/M \rangle$ *and* $L' = \langle 1/u', \alpha/m', \alpha/M' \rangle$ *such that* $\alpha < \alpha'$: $L \sim_{LN} L'$*iff* $u = u'$*and* $m < u < M$.

This means that in a set S of equivalent 2nd core lotteries, all the unimodal 2nd core lotteries share the same mode and for all of them, but maybe the one with the lowest α^{2nd}, $m < u < M$. This latter lottery is very significant, since one can forget about all the unimodal 2nd core lotteries in the set S for which α^{2nd} is not the lowest one. As shown by the following proposition [4]:

Proposition 3.4. *For any two indifferent second core lotteries of the form* $L = \langle 1/u, \alpha/m, \alpha/M \rangle$ *and* $L' = \langle 1/u, \alpha'/m', \alpha'/M' \rangle$*, if* $\alpha \leq \alpha'$*, then for any 2nd core lottery,* L'', L_b*, it holds that:*

(i) $\langle \gamma/L', \beta/L'' \rangle \succ_{LN} L_b \Rightarrow \langle \gamma/L, \beta/L'' \rangle \succ_{LN} L_b$

(ii) $L_b \succ_{LN} \langle \gamma/L', \beta/L'' \rangle \Rightarrow L_b \succ_{LN} \langle \gamma/L, \beta/L'' \rangle$

Actually, replacing L' by L in a strategy may break ties (for instance, it may happen that $\langle \gamma/L', \beta/L'' \rangle \sim_{LN} L_b$ and $\langle \gamma/L, \beta/L'' \rangle \succ_{LN} L_b$), but this can never change the strict preference. An important consequence of Proposition (3.3 (ii)) is that, if the strategy corresponding to a lottery L' belongs to the set of optimal lotteries, so does the strategy associated to L.

Thus, before returning the set S of lotteries computed by the recursive procedure, the pruning function Prune$_O$ (see Algorithm 4) can prune not only the dominated lotteries but also all the unimodal core lotteries except the one with the lowest α^{2nd}. As a consequence, the sets of lotteries handled by ExtDynProg when optimizing LN contains at most $2.p^2$ elements (p^2 multimodal core lotteries and p^2 unimodal 2nd core lotteries sharing the same α^{2nd}), all being indifferent to each other.

The ExtDynProg procedure is thus polynomial in the size of the decision tree when $O = LN$.

Corollary 3.1. *DT-OPT-*$L\Pi$ *and DT-OPT-*LN *belong to* P *for both* $\otimes = \min$ *and* $\otimes = *$.

The main difference between DynProg and ExtDynProg is that the latter returns a *set* S of simple lotteries rather than a single lottery, together with their associated strategies. Dominated lotteries/strategies are removed by a call to a pruning function. This is harmless, thanks to the monotonicity principle. At each chance node, a pointwise composition of the lotteries returned by the recursive calls is performed: this leads to a set of reduced lotteries that is pruned so as to keep, again, judicious non dominated decisions.

In the Example below, we aim to show that ExtDynProg succeeds to build an optimal strategy that DynProg could not find it in the Counter-Example 2.1.

3.3 Illustrative example

Let us consider the decision tree in Figure 3.1 with $O = L\Pi$ and $\otimes = min$.

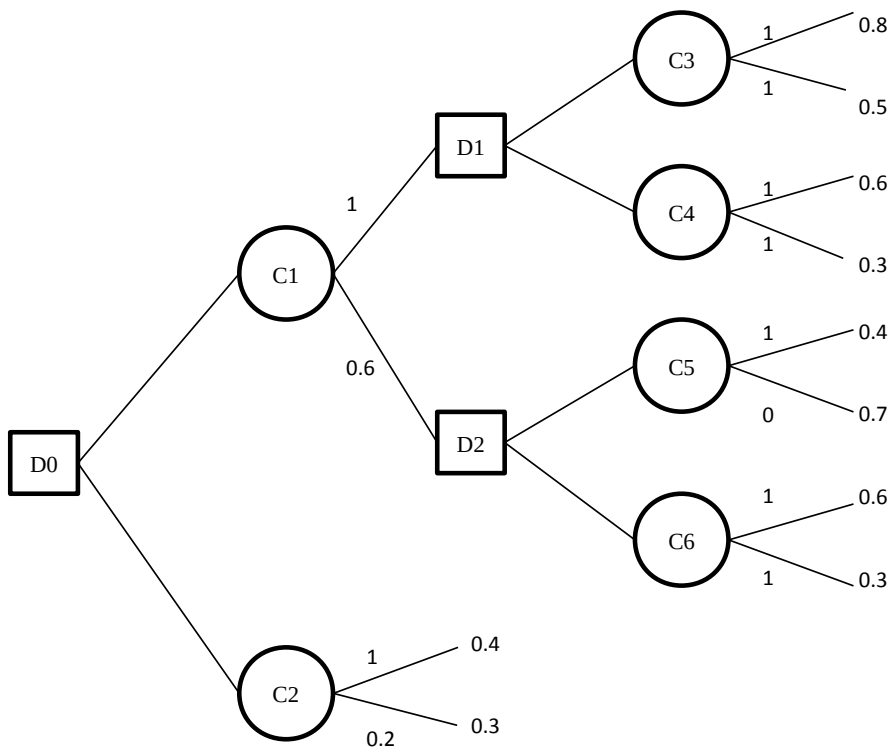

Figure 3.1: Decision tree of counter-example 2.2.

Principal steps for the evaluation of this decision tree using the ExtDynProg function are detailed in what follows:

- Initially, For D_0 we have $\delta = \emptyset$; and $S = \emptyset$ with $succ(D_0) = \{C_1; C_2\}$.
  $S = S \cup ExtDynProg(C_1)$ and $S = S \cup ExtDynProg(C_2)$ since $succ(C_1) = \{D_1; D_2\}$.
  We have $Y = D_1$ and $Y = D_2$.
- For $Y = D_1$ we have $S = \emptyset$ and $succ(D_1) = \{C_3; C_4\}$
  $S = S \cup ExtDynProg(C_3)$ and $S = S \cup ExtDynProg(C_4)$:

1. For $Y = C_3$, $succ(C_3) = \{LN_1; LN_2\}$ we have $Y = LN_1$ and $Y = LN_2$:
   $ExtDynProg(LN_1) : S1 = \{\langle 1/0.8\rangle, LN_1\}$;
   $ExtDynProg(LN_2) : S2 = \{\langle 1/0.5\rangle, LN_2\}$.
   For $Y = C_3$: we have $Res = \emptyset$; and $\delta'' = \delta \cup \delta' = (D_1, C_3)$;
   $L'' = \langle 1/0.8, 1/0.5\rangle$;
   $L'' \leftarrow L''^{core} = \langle 1/0.8, 1/0.5\rangle$;
   $Res = \{\langle 1/0.1\rangle, (D_1, C_3)\}$;
   $S = Prune_{L\Pi}(Res) = \{\langle 1/0.8, 1/0.5\rangle, (D_1, C_3)\}$.

2. For $Y = C_4$, $succ(C_4) = \{LN_3; LN_4\}$ we have $Y = LN_3$ and $Y = LN_4$:
   $ExtDynProg(LN_3) : S1 = \{\langle 1/0.6\rangle, L3\}$;
   $ExtDynProg(LN_4) : S2 = \{\langle 1/0.3\rangle, L4\}$.
   For $Y = C_4$: we have $Res = \emptyset$;
   $\delta'' = \delta \cup \delta' = (D_1, C_4)$;
   $L'' = \langle 1/0.6, 1/0.3\rangle$;
   $L'' \leftarrow L''^{core} = \langle 1/0.6, 1/0.3\rangle$;
   $Res = \{\langle 1/0.6, 1/0.3\rangle, (D_1, C_4)\}$;
   $S = Prune_{L\Pi}(Res) = \{\langle 1/0.6, 1/0.3\rangle, (D_1, C_4)\}$.

For $Y = D_1$ we have $S = \emptyset$ and $S = S \cup ExtDynProg(C_3)$
so, $S = \{\langle 1/0.8, 1/0.5\rangle, (D_1, C_3)\}$;
$S = S \cup ExtDynProg(C_4)$ so, $S = \{\langle 1/0.8, 1/0.5\rangle, (D_1, C_3); \langle 1/0.6, 1/0.3\rangle, (D_1, C_4)\}$;
$S = Prune_{L\Pi}(S) = \{\langle 1/0.8, 1/0.5\rangle, (D_1, C_3); \langle 1/0.6, 1/0.3\rangle, (D_1, C_4)\}$,
because $L_{C_4} \sim_{L\Pi} L_{C_3}$.

- For $Y = D_2$ we have $S = \emptyset$, $succ(D_2) = \{C_5; C_6\}$
  For $Y = C_5$, $S = S \cup ExtDynProg(C_5)$ since $succ(C_5) = \{LN_5; LN_6\}$
  we have $Y = LN_5$ and $Y = LN_6$:
  $ExtDynProg(LN_5) : S1 = \{\langle 1/0.4\rangle, LN_5\}$;
  $ExtDynProg(LN_6) : S2 = \{\langle 0.0/0.7\rangle, LN_6\}$.

  1. For $Y = C_5$: we have $Res = \emptyset$;
     $\delta'' = \delta \cup \delta' = (D_2, C_5)$;
     $L'' = \langle 1/0.4\rangle$;
     $L'' \leftarrow L''^{core} = \langle 1/0.4\rangle$;
     $Res = \{\langle 1/0.1\rangle, (D_2, C_5)\}$;
     $S = Prune_{L\Pi}(Res) = \{\langle 1/0.4\rangle, (D_2, C_5)\}$.

  2. For $Y = C_6$, since $succ(C_6) = \{LN_7; LN_8\}$ we have $Y = LN_7$ and $Y = LN_8$:
     $ExtDynProg(LN_7) : S1 = \{\langle 1/0.6\rangle, LN_7\}$;

$ExtDynProg(LN_8) : S2 = \{\langle 1/0.3\rangle, LN_8\}$.
For $Y = C_6$: we have $Res = \emptyset$;
$\delta'' = \delta \cup \delta' = (D_2, C_6)$;
$L'' = \langle 1/0.6, 1/0.3\rangle$;
$L'' \leftarrow L''^{core} = \langle 1/0.2\rangle$;
$Res = \{\langle 1/0.2\rangle, (D_2, C_6)\}$;
$S = Prune_{L\Pi}(Res) = \{\langle 1/0.6, 1/0.3\rangle, (D_2, C_6)\}$.

For $Y = D_2$ we have $S = \emptyset$ and $S = S \cup ExtDynProg(C_5)$ so, $S = \{\langle 1/0.4\rangle, (D_2, C_5)\}$;
$S = S \cup ExtDynProg(C_6)$ so, $S = \{\langle 1/0.4\rangle, (D_1, C_5); \langle 1/0.6, 1/0.3\rangle, (D_2, C_6)\}$;
$S = Prune_{L\Pi}(S) = \{\langle 1/0.4\rangle, (D_1, C_5); \langle 1/0.6, 1/0.3\rangle, (D_2, C_6)\}$ because $L_{C_5} \sim_{L\Pi} L_{C_6}$.

- For $Y = C_1$ we have
  $ExtDynProg(D_1) : S1 = \{\langle 1/0.8, 1/0.5\rangle, (D_1, C_3); \langle 1/0.6, 1/0.3\rangle, (D_1, C_4)\}$;
  and $ExtDynProg(D_2) : S2 = \{\langle 1/0.4\rangle, (D_2, C_5); \langle 1/0.6, 1/0.3\rangle, (D_2, C_6)\}$.

  1. $\delta''_1 = \delta_1 \cup \delta'_1 = ((D_0, C_1), (D_1, C_3), (D_2, C_5))$;
     $L''_1 = \langle 1/0.8, 1/0.5, 0.6/0.4\rangle$;
     $L''_1 \leftarrow L''^{core}_1 = \langle 1/0.8, 1/0.5\rangle$;
     $\Rightarrow \quad Res = \{\langle 1/0.8, 1/0.5\rangle; ((D_0, C_1), (D_1, C_3), (D_2, C_6))\}$.

  2. $\delta''_2 = \delta_1 \cup \delta'_2 = ((D_0, C_1), (D_1, C_3), (D_2, C_6))$.;
     $L''_2 = \langle 1/0.8, 1/0.5, 0.6/0.6, 0.6/0.3\rangle$;
     $L''_2 \leftarrow L''^{core}_2 = \langle 1/0.8, 1/0.5\rangle$;
     $\Rightarrow \quad Res = \{\langle 1/0.8, 1/0.5\rangle; ((D_0, C_1), (D_1, C_3), (D_2, C_6))\}$.

  3. $\delta''_3 = \delta_2 \cup \delta'_1 = ((D_0, C_1), (D_1, C_4), (D_2, C_5))$;
     $L''_3 = \langle 1/0.6, 1/0.3, 0.6/0.4\rangle$;
     $L''_3 \leftarrow L''^{core}_3 = \langle 1/0.6, 1/0.3\rangle$;
     $\Rightarrow \quad Res = \{\langle 1/0.6, 1/0.3\rangle; ((D_0, C_1), (D_1, C_4), (D_2, C_5))\}$.

  4. $\delta''_4 = \delta_2 \cup \delta'_2 = ((D_0, C_1), (D_1, C_4), (D_2, C_6))$;
     $L''_4 = \langle 1/0.6, 1/0.3\rangle$;
     $L''_4 \leftarrow L''^{core}_4 = \langle 1/0.6, 1/0.3\rangle$;
     $\Rightarrow \quad Res = \{\langle 1/0.6, 1/0.3\rangle; ((D_0, C_1), (D_1, C_4), (D_2, C_6))\}$.

  $S = Prune_{L\Pi}(Res) =$
  $\{\langle 1/0.8, 1/0.5\rangle; (D_0, C_1), (D_1, C_3), (D_2, C_5), \langle 1/0.6, 1/0.3\rangle; ((D_0, C_1), (D_1, C_4), (D_2, C_5))\}$
  (For $Y = C_1$ we have the size of S=2).

- For $Y = C_2$, since $succ(C_2) = \{LN_9; LN_{10}\}$ we have $Y = LN_9$ and $Y = LN_{10}$:
  $ExtDynProg(LN_9) : S1 = \{\langle 1/0.4\rangle, LN_9\}$;
  $ExtDynProg(LN_{10}) : S2 = \{\langle 0.2/0.3\rangle, L_{10}\}$.
  we have $Res = \delta'' = \delta \cup \delta' = C_2$;
  $L'' = \langle 1/0.4, 0.2/0.3\rangle$;
  $L'' \leftarrow L''^{core} = \langle 1/0.4\rangle$;
  $Res = \{\langle 1/0.1\rangle, (D_0, C_2)\}$;
  $S = Prune_{L\Pi}(Res) = \{\langle 1/0.4\rangle, (D_0, C_2)\}$ (the size of S=1).

- For $Y = D_1$ we have $S = \emptyset$ and $S = S \cup ExtDynProg(C_1)$
  Note that: $\delta_1 = ((D_0, C_1), (D_1, C_3), (D_2, C_5))$;
  and $\delta_2 = ((D_0, C_1), (D_1, C_4), (D_2, C_5))$.
  so, $S = \{\langle 1/0.8, 1/0.5\rangle; \delta_1, \langle 1/0.6, 1/0.3\rangle; \delta_2\}$
  $S = S \cup ExtDynProg(C_2)$
  so, $S = \{\langle 1/0.8, 1/0.5\rangle; \delta_1, \langle 1/0.6, 1/0.3\rangle; \delta_2; \langle 1/0.4\rangle, (D_0, C_2)\}$ (Here the size of S=3).
  $S = Prune_{L\Pi}(S) =$
  $\{\langle 1/0.8, 1/0.5\rangle; \delta_1, \langle 1/0.6, 1/0.3\rangle; \delta_2\}$
  because $max(L^{core}_{\delta_1}) > min(L^{core}_{C_2})$, so $L_{\delta_1} \succ_{L\Pi} L_{C_2}$.

ExtDynProg returns two equivalent optimal strategies that dominate the best strategy founded by DynProg in Counter-Example 2.1:

- $((D_0, C_1), (D_1, C_3), (D_2, C5))$;
- $((D_0, C_1), (D_1, C_4), (D_2, C5))$.

3.4 Recapitulation

Optimizing strategies algorithms depends on the monotonicity and transitivity properties of the corresponding possibilistic decision criterion. Table 3.1 gives the main revised results in previous chapters.

- Taking advantage of the fact that the optimistic and pessimistic possibilistic qualitative criteria (U_{opt}, U_{pes}) are transitive and obey a weak form of the monotonicity property, [3] have indeed shown that Dynamic programming [6] may be used to get an optimal strategy in polytime; this is also the case for the binary possibilistic utility (PU) [3].

Criterion (O)		Monotonicity	Transitivity	Complexity	Algorithm
Poss. Qualitative utilities	U_{opt}	Yes	Yes	P	DynProg
	U_{pes}	Yes	Yes	P	DynProg
	PU	Yes	Yes	P	DynProg
Poss. Choquet integrals	Ch_N	No	No	NP	BB
	Ch_Π	No	No	NP	BB
Poss. likely dominance	LN	Yes	No (quasi-transitive)	P	ExtDynProg
	$L\Pi$	Yes	No (quasi-transitive)	P	ExtDynProg

Table 3.1: Summary of strategy optimization in possibilistic decision trees.

- On the contrary, evaluating possibilistic Choquet integrals (Ch_Π and Ch_N) is NP-hard [3], thus, they are incompatible with Dynamic programming. Branch and bound algorithm, which uses DynProg, is able to build an optimal strategy in polytime.
- For possibilsitic likely dominance criteria ($L\Pi$ and LN), the preference order induced on the lotteries is quasi-transitive, this means that, even if the monotonicity principle is satisfied, the application of the standard DynProg can lead to suboptimal strategies. Ext-DynProg is able to built an optimal strategy in polytime.

3.5 Conclusion

In a decision tree where chance nodes present equivalent states Dynamic programming algorithm may perform an incoherent choice i.e. builds a sub-optimal strategy. So it is possible to run the backward induction and keep a set of optimal sub-strategies according to the decision criterion (in our case it may be $L\Pi$ or LN). This is the intuition behind Extended dynamic programming. This chapter gives a deep study of Extended dynamic programming algorithm in order to optimize strategies for possibilistic likely dominance criteria. In fact, this algorithm seems able to build, always, an optimal strategy in polytime.

To prove the effectiveness of this algorithm, we propose in next chapter an experimental study following a complete implementation of the algorithms discussed above.

Chapter 4

Implementation and Experimentation

4.1 Introduction

In order to study the feasibility of Extended dynamic programming algorithm for finding the optimal strategy w.r.t. possibilistic likely dominance in decision trees, we present in this chapter an experimental study.

We propose a toolbox implemented with Java language which allows the construction and the evaluation of possibilistic decision trees. Once the different algorithms are implemented, for checking the strength of Extended dynamic programming versus Dynamic programming and judging its performances, we have performed several tests and simulations. Main results presented in this Section are accepted for publication in [5].

This chapter is composed of three main sections: Section 4.2 discusses the technical implementation of different methods in our toolbox. Section 4.3 is devoted to the evaluation of the performance of implemented methods; we propose several evaluation measures and we give a comparative analysis between Extended dynamic programming and Dynamic programming. Then, we discuss experimental results obtained. Finally, Section 4.4 presents the application of possibilistic decision criteria on a simulation of the TV game ”Who wants to be a millionaire?” following [25, 20]. The primary aim of this section is to show how a real sequential decision problem can be easily modeled with a decision tree and solved using an optimizing strategy algorithm.

4.2 Implementation

In this section we present the development tool adopted and the framework, we show the main interfaces, then we detail programs.

4.2.1 Framework

We have developed our application with the environment ”NetBeans 6.5” with JDK 6.0. NetBeans is an integrated open source, available for Windows, Mac, Linux, and Solaris development environment. It supports a wide variety of programming languages such as Java, PHP, JavaScript and Ajax, Ruby and Ruby on Rails, C / C + +, XML, HTML. We chose this environment to take advantage of its tool palettes to construct interfaces (AWT and Swing components) intuitively. Also, it provides resources to develop the structural elements of the applications configuration, management of menus, windows, file management etc.

Indeed, we propose the implementation of three optimizing strategy algorithms detailed in previous chapters: Branch and bound, Dynamic programming and Extended dynamic programming.

These programs have as inputs:

- An evaluation algorithm: Dynamic programming, Branch and bound or Extended dynamic programming.
- A possibilistic decision tree: we have three structures:
    1. Binary decision trees: each node has exactly two children. This structure provides a simple and generalized application, since it is simple to comprehend and to interpret.
    2. We have implemented different tree structure compared to the first case. For example we add the case that we can have leaves just after decision nodes, or we have decision node after decision node and it may reflect real cases (surely have the utilities after the user decision).
    3. Decision trees with branching factor of 1, 2 and 3. This structure allows us to go further in our experimentation and get closer to real cases.

- A possibilistic decision criteria:

  For Dynamic programming we use possibilistic qualitative utilities and possibilistic likely dominance. In the other hand, for Branch and bound we optimize possibilistic Choquet integrals. Lastly, for Extended dynamic programming we optimize possibilistic likely dominance criteria.

- Type of interpretation for reduction of lotteries: Ordinal setting or numerical setting.

The outputs of our programs are: the optimal strategy i.e. we display the optimal choice (chance node) for each decision node, the elapsed time and the utility degree of the optimal strategy w.r.t. the decision criterion.

4.2.2 Main programs

We will present the most important interfaces each with a summary of its functionality. Figure 4.1 is the first interface that appears when you start the application. It allows the evaluation of possibilistic trees or the evaluation and the comparison of simple and compound lotteries.

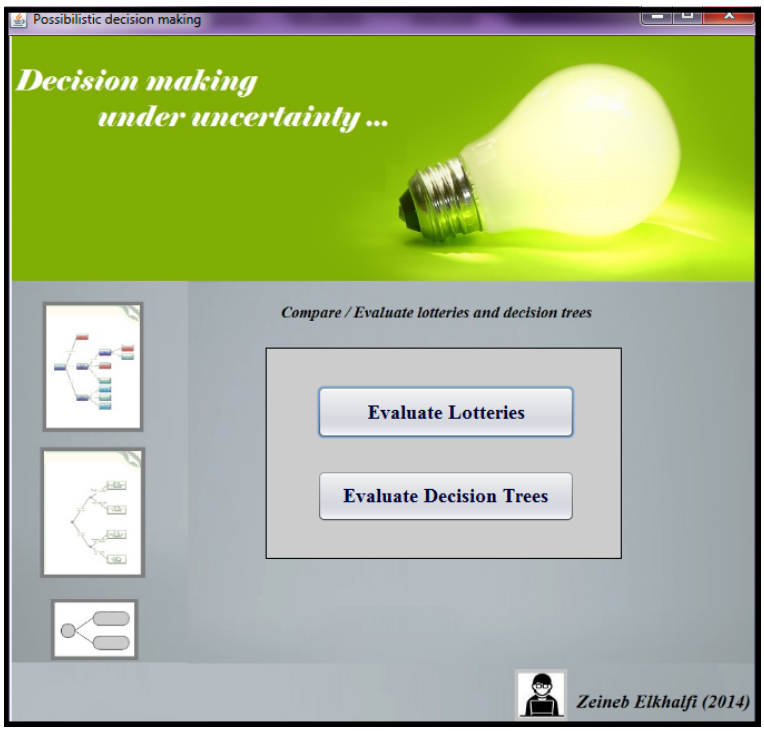

Figure 4.1: Main menu interface.

1. **Evaluation of possibilistic decision trees**

   The corresponding interface has three tabs. Figure 4.2 present the first tab corresponding to Dynamic programming algorithm. Each tab allows us to explore an algorithm, Dynamic programming, Branch and bound or Extended dynamic programming. It also offers with a set of buttons, several services such as the introduction of data by various ways, the configuration of options for decision criteria, the visualization of optimal strategies and export results etc.

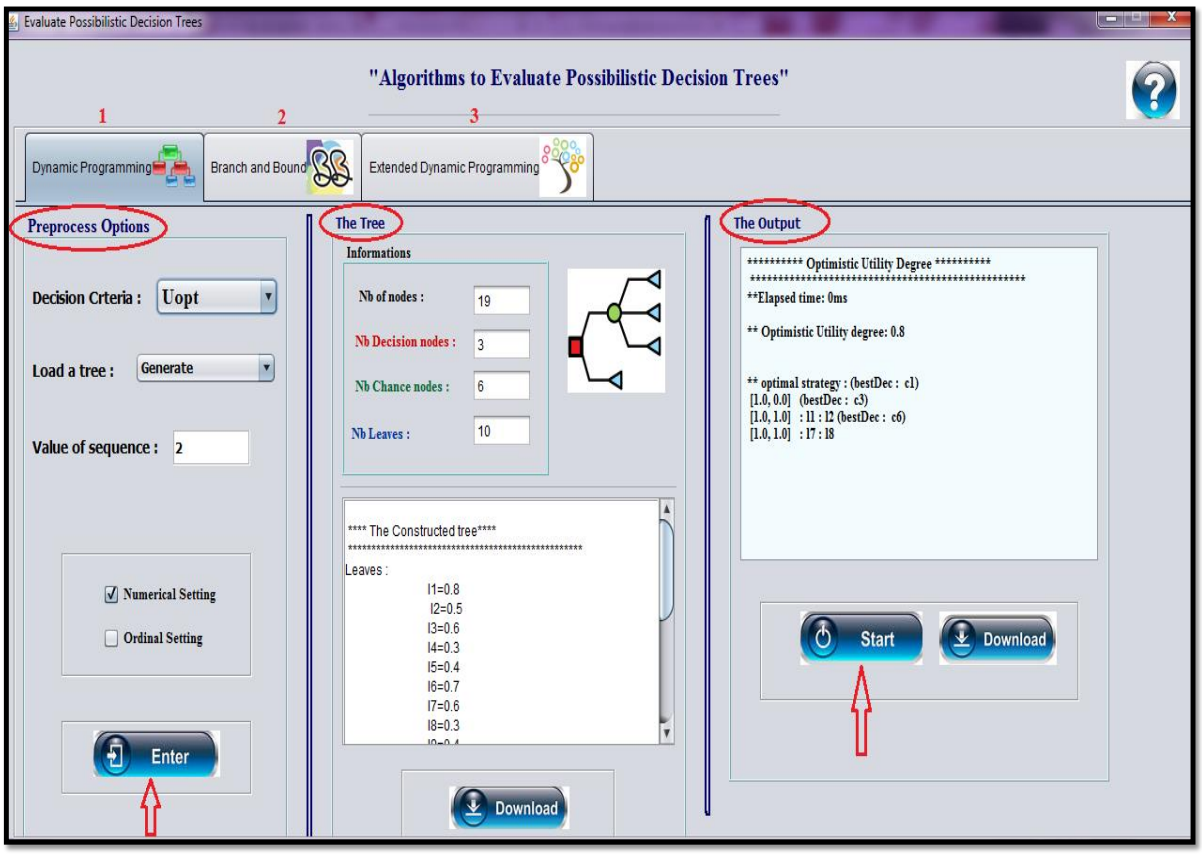

Figure 4.2: Evaluating possibilistic decision trees interface: Dynamic programming.

For each tab we use the same process:

(a) **Preprocess options:** The user can choose decision criterion: qualitative possibilistic utilities ”Uopt” (default) or ”Upes”, or possibilistic likely dominance $L\Pi$ or LN. Another option, configuring the reduction of composed lotteries; he has to choose the appropriate option: the ordinal or numerical parameter settings.

(b) **Load the decision tree:** There are three ways of loading trees:

- The item "download" permits to download a tree from a text file, this file must include utilities leaves, then the possibilities of all lotteries. By selecting this item, a "fileShooser" opens to retrieve the text file containing the data.
- The item "Random Generation" can automatically generate a tree with utilities and possibilities are random values respecting the standards of possibilistic decision trees. Moreover, the user can choose the size of the tree by setting the value of the sequence, i.e. the number of decision nodes by path (root $\Rightarrow$ leaves), this is done by introducing an integer in the text area "value of sequence".
- Finally the item "generate" loads a tree introducing in advance in the program code.

(c) **Strategy optimization and output display:** After having introduced the tree and configured the options, the "Enter" button allows the system to check whether the data chosen are compatible with the standards of possibilistic decision trees. If it is the case, it loads the tree in memory and displays data and statistics in this interface: the number of decision nodes, the number of chance nodes and the number of leaves, also, we show the structure of the tree with the nodes and lotteries. Otherwise an error message is displayed.

Such information may be stored in a text file for leaving a trace of the treated trees. Thus, the "download" button opens a "fileShooser" which allows creating or selecting a file in which you copy all the data on the tree. The "Start" button is used to start the optimization process. The result is displayed in the text area "the output". It displays the value of utility degree, the corresponding strategy as well as the optimal lottery.

2. **Evaluating/comparing lotteries interface**

This interface (Figure 4.3) allows us to evaluate and compare lotteries regarding a decision criterion. It presents other services as computing L^{core} and L^{2nd} and reducing compound lotteries.

4.2.3 Implementation tricks

Here we give additional guidance about the implementation. Indeed, we have used some optimization techniques in order to achieve a balance between fast execution speed and small code size.

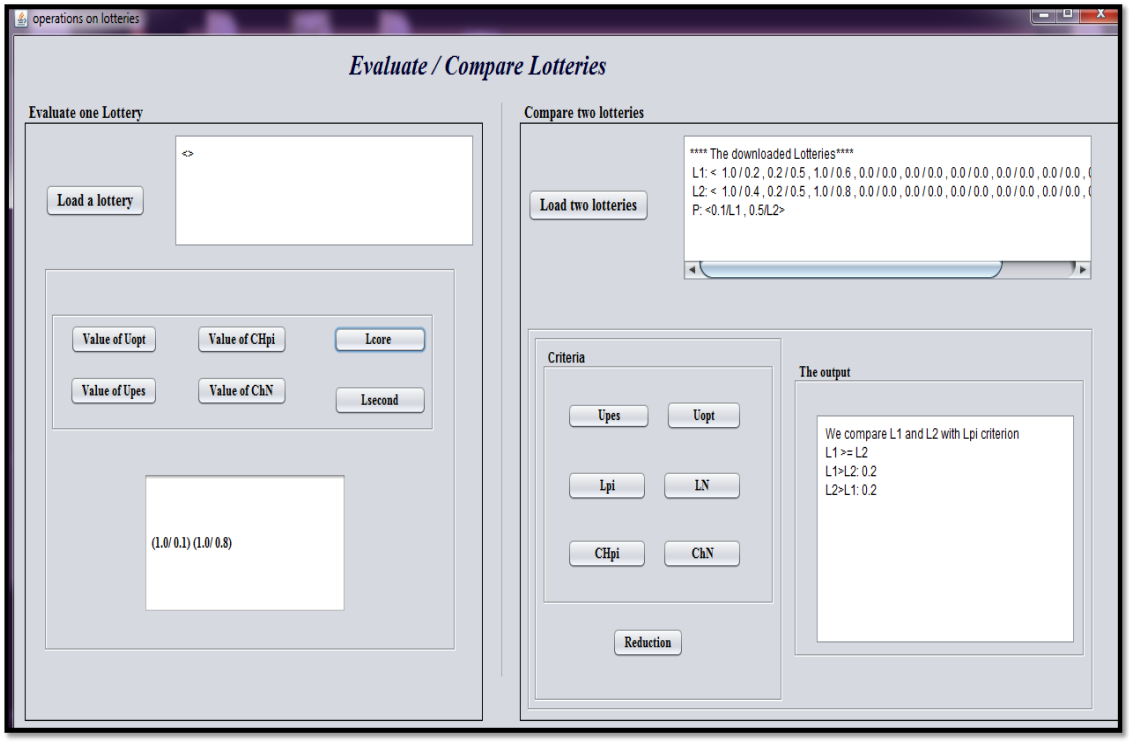

Figure 4.3: Evaluating/comparing lotteries interface.

- **Optimization of the data structure :** we have used an abstract class, namely ”PiTree”, to define and generate a possibilistic tree. In fact, it defines a tree as a node that points to another node. It creates nodes and affects the possibilities and returns the root (first decision node) as a pointer which allows to access directly to all child nodes. This structure is well suited for sequential decision problem i.e. recursive process and it allows to reduce time and space when generating big trees.

  The major problem with decision trees structure is that we have to store all utilities and possibility degrees and this may consumes a lot of memory. In order to better integrate the numerical component with the graphical component of the tree, we define one table that hold all utilities and for each chance node we store the possibility of each outcome.

- **Reducing the size of the set** S that handles the list of optimal sub-strategies of ExtDyn-Prog: the idea is to eliminate lotteries that we use to compute the compound lottery. So, at each step, we reduce the compound lottery and we eliminate the simple lotteries. This maintains a limited number of elements to handle.

4.3 Experimentation

This subsection is composed of three parts: in the first part, we propose some evaluation measures in order to compare the strategies returned by optimizing strategies algorithms (DynProg and ExtDynProg) w.r.t. likely dominance. In the second part, we will detail the experimental process and especially the construction of possibilistic decision trees. Finally, the third part presents different results obtained for each experiment done and analyzes these results to validate the proposed algorithm.

4.3.1 Evaluation measures

The difficulty with likely dominance criteria is that the strategy founded has not a utility degree i.e. this type of criteria draws a pairwise comparison. Therefore, to compare two optimal strategies (returned by the two algorithms) we propose three alternatives.

1. **Equivalence**

   Two strategies are equivalents iff:

   $$\mu(L_{\delta_{DynProg}} \geq L_{\delta_{ExtDynProg}}) = \mu(L_{\delta_{ExtDynProg}} \geq L_{\delta_{DynProg}}). \tag{4.1}$$

   where μ is the necessity measure N (resp. the possibility measure Π) for LN (resp. for $L\Pi$), and $L_{\delta_{DynProg}}$ (resp. $L_{\delta_{ExtDynProg}}$) is the reduced lottery representing the optimal strategy returned by DynProg (resp. ExtDynProg).

   Example 4.1. *Let us consider the possibilistic decision tree in Figure 4.4 with $O = L\Pi$ and $\otimes = min$.*

   - *The optimal strategy returned by ExtDynProg is :*
     $\delta_{ExtDynProg} = \{(D_0, C_1), (D_1, C_3), (D_2, C_6)\}$.
     The compound lottery representing this strategy is : $L_{\delta_{ExtDynProg}} = \langle 1/L_{C_3}, 0.6/L_{C_6}\rangle$.
     Then $Reduction(L_{\delta_{ExtDynProg}}) = \langle 1/0.8, 1/0.5, 0.6/0.6, 0.6/0.3\rangle$.

   - *The strategy returned by Dynamic programming is:*
     $\delta_{DynProg} = \{(D_0, C_1), (D_1, C_4), (D_2, C_5)\}\rangle$.
     The compound lottery representing this strategy is: $L_{\delta_{DynProg}} = \langle 1/L_{C_4}, 0.6/L_{C_5}\rangle$.
     Then $Reduction(L_{\delta_{DynProg}}) = \langle 1/0.6, 1/0.3, 0.6/0.4\rangle$.

   - $\Pi(L_{\delta_{DynProg}} \geq L_{\delta_{ExtDynProg}}) = max(min(1, 1), min(1, 0.6), min(0.6, 0.6)) = 1$.

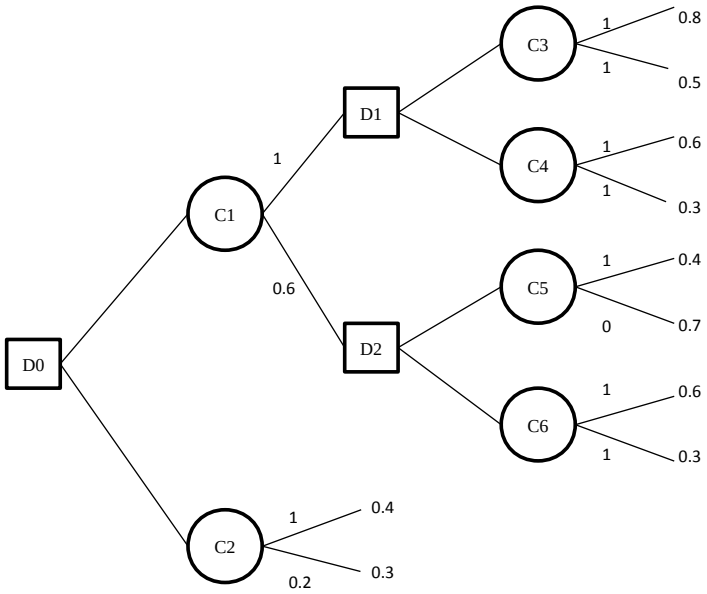

Figure 4.4: Decision tree for Example 4.1.

- $\Pi(L_{\delta_{ExtDynProg}} \geq L_{\delta_{DynProg}}) = 1.$

We conclude that the strategy of DynProg is equivalent to that of ExtDynProg w.r.t. $L\Pi$.

Proposition 4.1. *If the strategy returned by DynProg is equivalent to that of ExtDynProg w.r.t.* $L\Pi$ *so:*

$$\Pi(L_{\delta_{DynProg}} \geq L_{\delta_{ExtDynProg}}) = \Pi(L_{\delta_{ExtDynProg}} \geq L_{\delta_{DynProg}}) = 1$$

Proof. (Proposition 4.1)
Because the possibility distributions are normalized $max(\Pi(A), \Pi(\overline{A}) = 1)$,
thus $max(\Pi(L_{\delta_{ExtDynProg}} \geq L_{\delta_{DynProg}}), \Pi(L_{\delta_{DynProg}} \geq L_{\delta_{ExtDynProg}})) = 1$, With (equivalence) $\Pi(L_{\delta_{ExtDynProg}} \geq L_{\delta_{DynProg}}) = \Pi(L_{\delta_{DynProg}} \geq L_{\delta_{ExtDynProg}})$.

So, if the strategies returned by DynProg and ExtDynProg are equivalent, we have:
$\Pi(L_{\delta_{DynProg}} \geq L_{\delta_{ExtDynProg}}) = \Pi(L_{\delta_{ExtDynProg}} \geq L_{\delta_{DynProg}}) = 1.$ □

2. **Closeness**

   To compare two optimal strategies, we propose to compute the closeness value representing the similarity between the optimal strategies of both algorithms expressed by:

$$Closeness(\delta_{DynProg}, \delta_{ExtDynProg}) = \frac{\mu(L_{\delta_{DynProg}} \geq L_{\delta_{ExtDynProg}})}{\mu(L_{\delta_{ExtDynProg}} \geq L_{\delta_{DynProg}})} \tag{4.2}$$

where μ is the necessity measure N (resp. the possibility measure Π) for LN (resp. for $L\Pi$).

Example 4.2. *Let us consider the possibilistic decision tree in Figure 4.5 with $O = L\Pi$ and $\otimes = min$.*

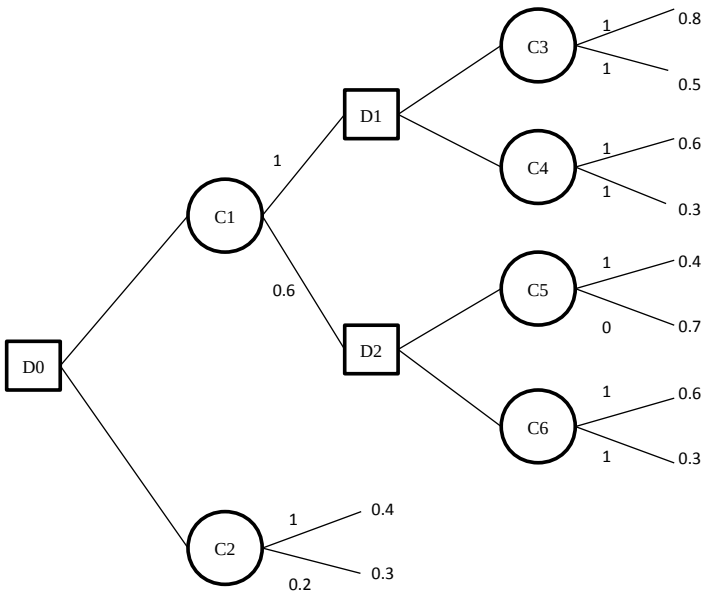

Figure 4.5: Decision tree for Example 4.2.

- *The optimal strategy returned by ExtDynProg is :*
  $\delta_{ExtDynProg} = \{(D_0, C_1), (D_1, C_3), (D_2, C_6)\}$.
  The compound lottery representing this strategy is: $L_{\delta_{ExtDynProg}} = \langle 1/L_{C_3}, 0.6/L_{C_6} \rangle$.
  Then $Reduction(L_{\delta_{ExtDynProg}}) = \langle 1/0.8, 1/0.5, 0.6/0.6, 0.6/0.3 \rangle$.

- *The strategy returned by Dynamic programming is :* $\delta_{DP} = \{(D_0, C_2)\}$.
  The compound lottery representing this strategy is: $L_{\delta_{DynProg}} = L_{C_3}$.
  Then $Reduction(L_{\delta_{DynProg}}) = \langle 1/0.4, 0.2/0.3 \rangle$.

- *We have* $\Pi(L_{\delta_{DynProg}} \geq L_{\delta_{ExtDynProg}}) = max(min(1, 0.6), min(0, 2, 0.6)) = 0.6$;
  and $\Pi(L_{\delta_{ExtDynProg}} \geq L_{\delta_{DynProg}}) = max(min(1, 1), min(1, 0.2), (0.6, 0.2)) = 1$.

- *Then* $\frac{\Pi(L_{\delta_{DynProg}} \geq L_{\delta_{ExtDynProg}})}{\Pi(L_{\delta_{ExtDynProg}} \geq L_{\delta_{DynProg}})} = \frac{0.6}{1} = 0.6.$

We conclude that the strategy of Dynamic programming is close to that of ExtDynProg with a match rate equal to 60% *(the gap indicator is equal to* 40%*).*

Proposition 4.2. *Let* $L_{\delta_{DynProg}}$ *(resp.* $L_{\delta_{ExtDynProg}}$*) be the lottery representing the strategy returned by DynProg (resp. ExtDynProg). Then:*

$\Pi(L_{\delta_{ExtDynProg}} \geq L_{\delta_{DynProg}}) = 1$ *in any case.*

Proof. (Proposition 4.2)
Consider that $L^*_{\delta_{ExtDynProg}}$ (resp. $L^*_{\delta_{DynProg}}$) is the lottery corresponding to an optimal strategy returned by ExtDynProg (resp. DynProg) w.r.t. $L\Pi$.

$\forall\ L$, $L^*_{\delta_{ExtDynProg}} \succeq_{L\Pi} L$ hence $\forall\ L$, $\Pi(L^*_{\delta_{DynProg}} \geq L) \geq \Pi(L \geq L^*_{\delta_{DynProg}})$.
Because the possibility distributions are normalized:
$max(\Pi(L^*_{\delta_{DynProg}} \geq L), \Pi(L \geq L^*_{\delta_{DynProg}})) = 1.$
We get $\Pi(L^*_{\delta_{ExtDynProg}} \geq L^*_{\delta_{DynProg}}) = 1.$ □

3. **Similarity**

We propose to compare two strategies according to their similarity degree that is the ratio between the number of similar actions in strategies and the total number of actions (denoted by N_A).

$$Sim(\delta_{DynProg}, \delta_{ExtDynProg}) = \frac{N_{A(\delta_{DynProg})=A(\delta_{ExtDynProg})}}{N_A}. \tag{4.3}$$

where:

- $N_{A(\delta_{DynProg})=A(\delta_{ExtDynProg})}$ is the number of similar actions between $\delta_{DynProg}$ and $\delta_{ExtDynProg}$.
- $A(\delta_{DynProg})$ is an action of a strategy found by DynProg.
- $A(\delta_{ExtDynProg})$ is an action of a strategy founded by ExtDynProg.

Example 4.3. *For decision tree in Figure 4.5 we consider the two strategies as follow:*

- $\delta_{DynProg} = ((D_0, C_1), (D_1, C_3), (D_2, C_5))$;
- $\delta_{ExtDynProg} = ((D_0, C_1), (D_1, C_4), (D_2, C_5))$.

We have $Sim(\delta_{DynProg}, \delta_{ExtDynProg}) = 2/3 = 0.67$ *i.e. the two strategies are similar for* 67%.

4.3.2 Experimental Setup

Experiments have been performed on an Intel-Pentium $1.60\ GHz$ CPU computer with $4GB$ of RAM. We aim, in particular, to compare the performances of the ExtDynProg approach with those of the sole DynProg procedure.

The tests were performed on 1000 complete binary decision trees in which every level, except leaves, is completely filled with two children (see Figure 4.6). These trees were generated randomly by varying the length of the path from the root to the farthest node. The number of decisions depends on this value (denoted by seq) and that we have varied from 2 to 5, with an alternation of decision and chance nodes (see 4.6): at each decision level i (i.e. odd levels from 1 to seq*2), the tree contains 2^{i-1} decision nodes followed by 2^i chance nodes. Thus, the number of chance nodes in the tree is equal to $n = m*2$ and the number of leaves is equal to $k = m+n+1$. This means that with $seq = 2$ (resp. 3, 4, 5), the number of decision nodes is equal to $m = 5$ (resp. 21, 85, 341) i.e. the number of nodes in the tree is equal to $|\mathcal{N}| = m + n + k = 31$ (resp. 127, 511, 2047) nodes. The utility values have been randomly fired in the set of $U = \{0, \ldots, 1\}$ following equiprobable distribution. Conditional possibilities relative to chance nodes are also chosen randomly in $[0, 1]$ and normalized.

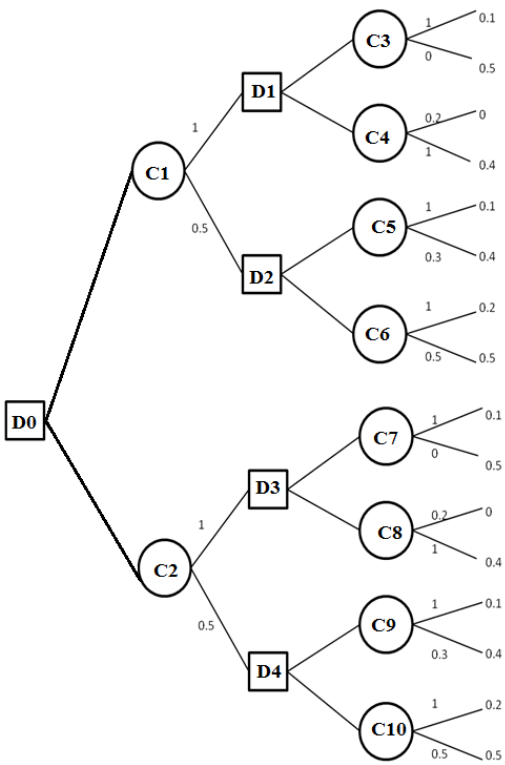

Figure 4.6: Example of a complete binary decision tree for Seq=2.

4.3.3 Experimental results

In this part we propose a comparative analysis of optimizing strategies algorithms. More precisely we compare the performance of DynProg with that of ExtDynProg.

Figure 4.7 presents different average execution CPU time for DynProg and for ExtDynProg. Unsurprisingly, the resolution time for both algorithms setting increases according to the size of the tree, it is better for DynProg than for ExtDynProg and this is due to the additional steps in the extended. The CPU time increases linearly for ExtDynProg (also for DynProg) and it remains affordable even with big trees (the maximal CPU time is about 3 seconds for 2047 nodes).

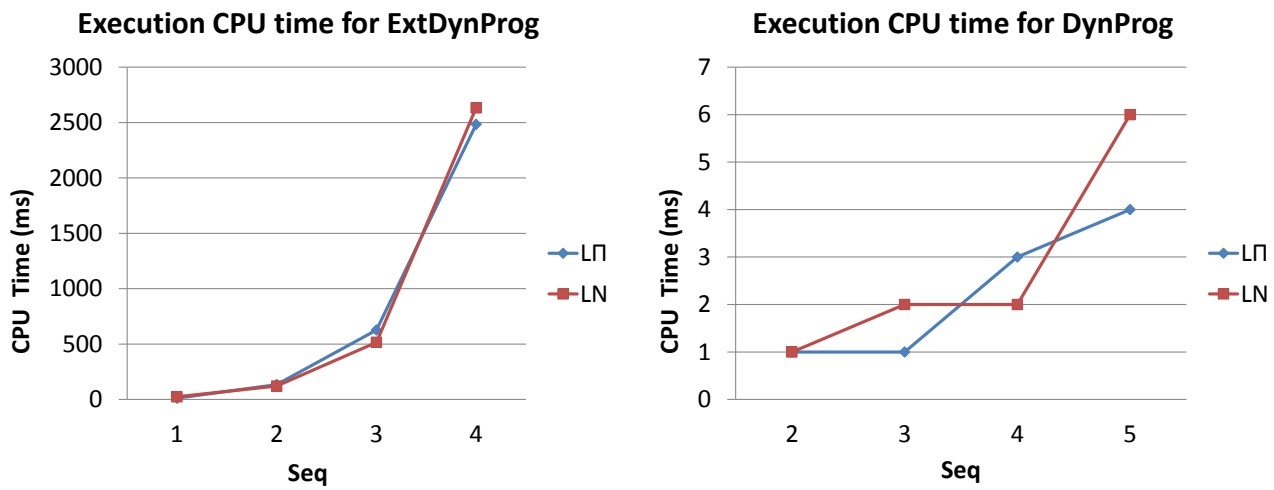

Figure 4.7: Execution CPU time for DynProg and ExtDynProg (in ms).

We are interested to measure the size of the set S, when executing ExtDynProg, in worst cases. Figure 4.8 shows that the maximum size of the set S remains very low during the execution of ExtDynProg: 8 elements in the worst case, for $seq = 5$ i.e. for trees with about 1000 leaves. We conclude that the real size of S is far below of p^2 founded with theoretical analysis (see Chapter 3 Section 3.2), hence, there is no risk to explode main memory.

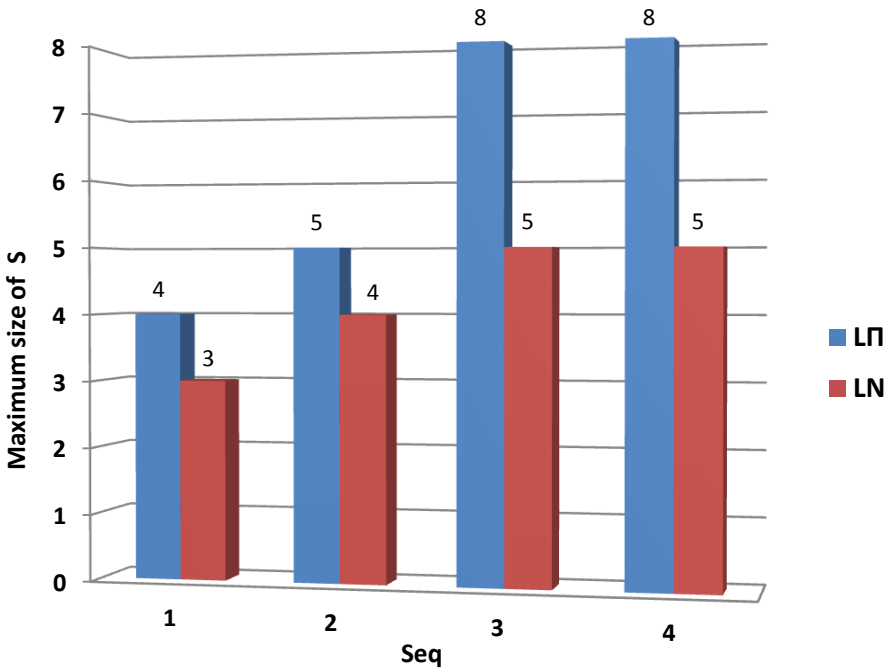

Figure 4.8: Maximum size of the set S.

We propose to evaluate the quality of solutions provided by DynProg by comparing them to those of ExtDynProg. Table 4.1 presents the number of cases where the solution of DynProg is equivalent to that of ExtDynProg i.e. ($\delta^*_{ExtDynProg} \sim_O \delta^*_{DynProg}$). Clearly, the results of DynProg are almost similar to those of ExtDynProg since the percentage of equivalent solutions is greater than 95% in any case.

Table 4.1: Percentage of problems where $\delta^*_{ExtDynProg} \sim_O \delta^*_{DynProg}$.

	Seq			
Criteria	2	3	4	5
$L\Pi$	98.7%	97%	96.6%	95.2%
LN	99%	97%	95.4%	96%

In the following, we are interested in measuring the quality of solutions for problems in which the strategies produced by the two algorithms are not equivalent.

Figure 4.9 shows that, when DynProg does not predict an optimal strategy, the quality of its solution is about $60 - 80\%$, i.e. acceptable but not more, and decreases when the size of the tree increases.

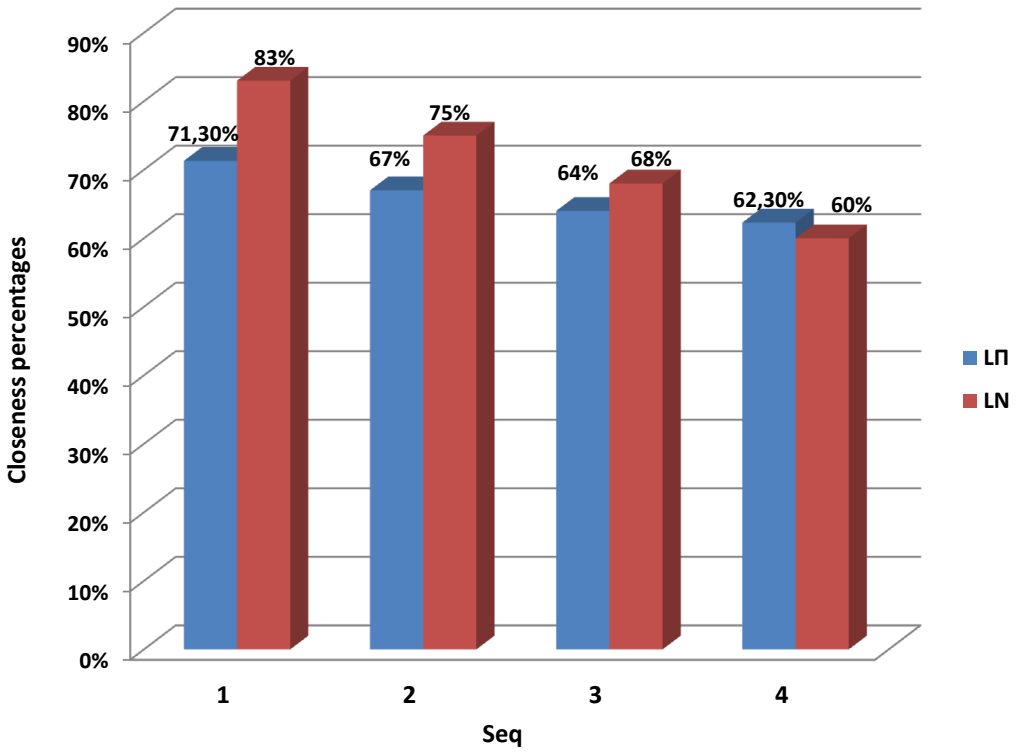

Figure 4.9: Average closeness values.

We compute the similarity between the optimal (non equivalent) strategies of DynProg and that of ExtDynProg (for $L\Pi$ and LN). The following table 4.2 reports the actual results that we found if we apply this alternative.

Table 4.2: Minimum and maximum action similarity values.

	Seq					
Criteria	2		3		4	
	min	max	min	max	min	max
$L\Pi$	0 %	0%	42.8%	57%	13.3%	53.3%
LN	34%	67%	42%	57%	10%	73.3%

We conclude that, the solution of DynProg is almost similar to that of ExtDynProg with a percentage around 50% when $seq = 3$ or $seq = 4$ in any case. For $seq = 2$ and $O = L\Pi$, we note that DynProg founds a different strategy (from the root of the decision tree).

In fact, ExtDynProg seems generally effective for randomly generated decision trees. To be convinced that this algorithm can handle real situations, we propose in next section its application to a concrete problem i.e. a TV game example.

4.4 Experimentation to the TV game show: "Who Wants To Be A Millionaire?"

In the previous section, we have used complete binary trees and we have generated them randomly. However, in many applications, decision trees are much less balanced and we do not have randomness and instability as a nuisance. Therefore, we propose to experiment ExtDynProg and DynProg (also BB) on a concrete problem where the decision tree has a natural structure and the possibility values reflect a real situation.

The problem that we propose to handle is a famous television game, well known under the name of *"Who Wants To Be A Millionaire?"*, following the work of [25, 20] in the probabilistic framework. It was debuted in the United Kingdom in September of 1998 and was very successful there. There were 45 versions presented in 71 countries [25] but the game's rules are similar in all countries. In this work, we will regard only the Spanish version reviewed in [20]. Indeed, we present the rules and basics ideas of the game, then, we show the general model i.e. the standard decision tree corresponding to the problem. Finally we detail our proposed model and the results.

4.4.1 Game's rules

One candidate is chosen out of a pool of ten and he has the chance of winning the top prize of $300000\ Euros$. To achieve this, he must answer 15 multiple-choice-questions, of increasing difficulty, correctly in a row.

In each step, he is shown the question and four possible answers before deciding whether to play or not. Once he has decided to stay in the game the next answer has to be correct to continue playing. Each question has a certain monetary value, given in Euros (see Table 4.3 column 1 and 2). If the candidate decides to stop and quit instead of answering: he then quits the game with the monetary value of the previous question. There are two stages (guarantee points) where the money is banked and cannot be lost even if the candidate offers an incorrect answer to one of the next questions: 1800 and 9000 Euros.

At any question, the contestant may use one or more of his three lifelines: phone a friend (discuss the question with a friend or relative on the phone for 30 seconds), ask the audience (the

audience votes and the percentage of votes each answer has received is shown) and 50:50 (the computer eliminates two possible answers leaving one wrong answer and the correct one).

4.4.2 The general model

In this work we adapt the general model proposed in [20]: if the contestant knows the answer of question n, he can directly give the correct answer, else he has to make a decision. A small part of the decision tree is represented in Figure 4.10 (the dotted lines represent omitted parts of the tree).

Chance nodes Q_1^i (resp. Q_2^i) represent question 1 (resp. 2), with two possible events (know the answer or not). They are represented by conditional probability distribution. Decision nodes D_1^i represent the decision to use an available lifeline, answer or quit facing question 1 (in fact, this latter opportunity becomes realistic only from question 2). Finally, the answer is represented by a chance node (A_1^i) where the probabilities of the events (correct or wrong answer) depend on the used lifelines.

In [20] Jeantet and Spanjaard have used the data provided by [25] to evaluate the different probabilities at the chance nodes. The whole decision tree has more than 75 million nodes and the depth was 30.

In what following, we propose a binary light version of the game, in the context of possibility decision theory.

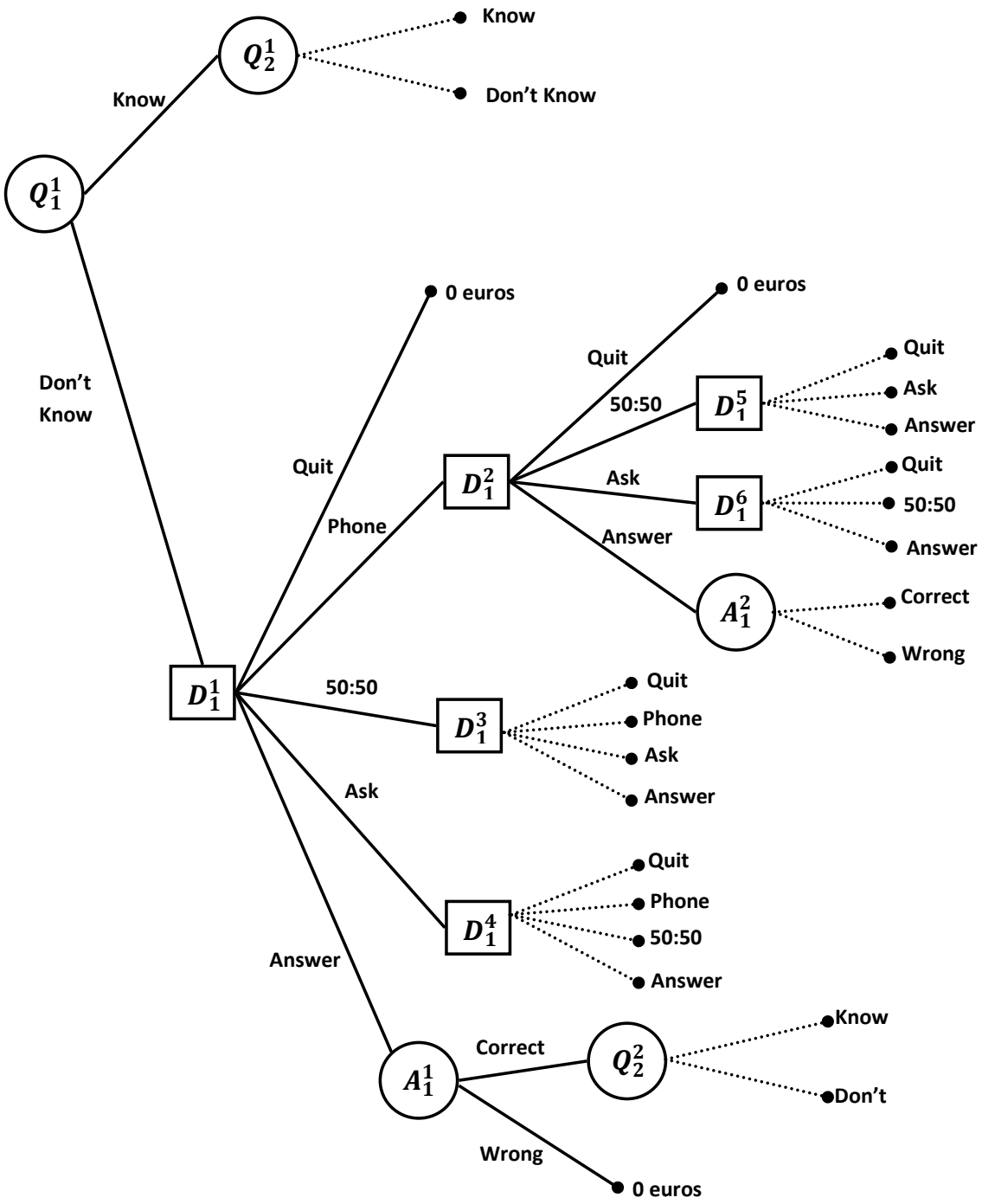

Figure 4.10: Refined decision tree for the TV game.

4.4.3 Possibilistic light "Who Wants To Be A Millionaire?"

We propose to transfer the game into a possibilistic sequential decision problem. The candidate is entitled to answer fifteen general knowledge questions, $n = 1, ..., 15$ (always four possible answers for each question). Figure 4.11 summarizes the game process that we propose to experiment, then, we model it in a binary possibilistic decision tree (using [22]).

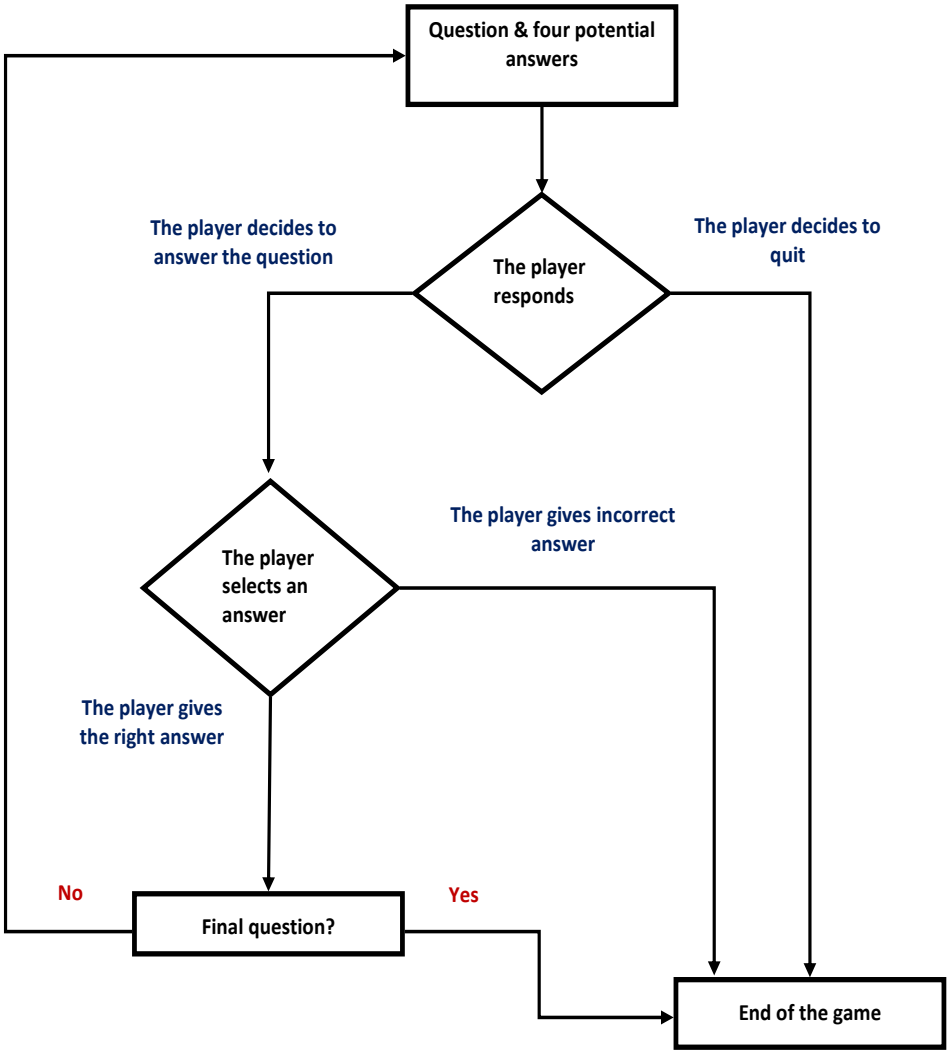

Figure 4.11: Process of the TV game.

Following the decision tree as described in figure 4.12: at each 'question' (decision node denoted by Q_n with n denoting the n'th question) the candidate has to decide whether to quit

voluntarily i.e. receives the money earned with the previous question, or guess the answer and thus continue the game [1]. There are two possible outcomes of the chance node 'selects an answer' (denoted by R_n):

- Takes the right answer, then the candidate receives an amount of money expressed by X_n plus the chance to continue the game with the next question. This amount of money increases from the first question with 150 Euro up to 300000 Euro. This is shown by the first two columns in Table 4.3.
- Answering incorrectly leads only to the lower value of the category, C_n. This value is zero for the first category ($n = 1, ..., 5$), 1800 Euro for the second ($n = 6, ..., 10$), and 9000 Euro for the third category (see column 3 in table 4.3).

Example 4.4. *If the candidate quits voluntarily before answering question number 7, the payoff is 2100 Euro (see column 4 in table 4.3). If he decides to answer question 7, he has a chance to answer this question correctly and the game starts with this next decision problem. If the candidate fails, he receives only 1800 Euro.*

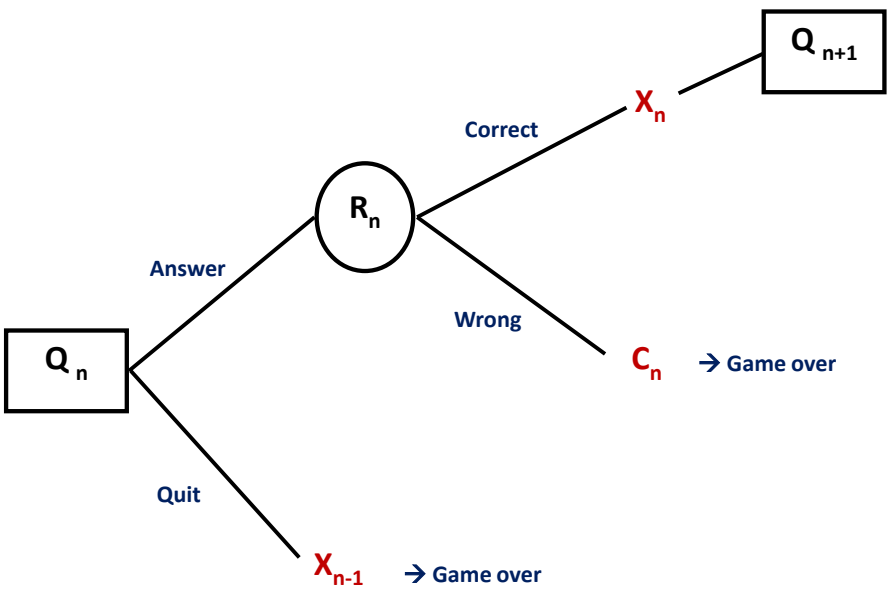

Figure 4.12: Graphical component of the TV game decision tree.

[1]Note that Figure 4.12 shows only a piece of the corresponding decision tree, it continues in the same manner for the next fourteen questions, until that each node is a leaf.

The situation can be described as a decision under possibilistic uncertainty: as long as a player knows the correct answer exactly, the situation could be described as a complete knowledge. If we assume that the candidate reaches one point in the game, where he is unsure of the right answer and indifferent which of the four answers is correct, we have a total ignorance.

The candidate can choose between a certain value X_{n-1} (column 4 in table 4.3) and two outcomes i.e. the lottery L_{R_n}. The payoff of this lottery can be expressed by X_n with the possibility $\pi(R_n = Correct)$ and the lower value C_n in the case of failure with possibility $\pi(R_n = Wrong)$. Note that, $max(\pi(R_n = Correct), \pi(R_n = Wrong)) = 1$ i.e. the possibility distributions are normalized. For simplicity, we assume that, for one question, the possibility $\pi(R_n = Correct) = 1$ if we are sure that the candidate is able to select the correct answer. Also the possibility $\pi(R_n = Correct) = \pi(R_n = Wrong) = 1$ if the candidate is indifferent between the four answers given to him i.e. total ignorance.

There exist no real evaluations of possibilities at each chance node. Therefore, we propose two solutions to quantify possibility distributions on $Succ(R_n)$: qualitative representation of uncertainty i.e. we define a total pre-order on the universe of discourse by referring to an analysis of the game, quantitative representation using probability-Possibility transformation of general model's probability distributions.

1. **Qualitative representation of uncertainty**

   The difficulty of questions increases at each step:

   $$difficulty_{Q_1} \leq difficulty_{Q_2} \leq ... \leq difficulty_{Q_{15}}.$$

   The possibility of the event *"the candidate give the correct answer to a question"* i.e. $\pi(R_n = Correct)$ decreases and the possibility of the event *"the candidate give the wrong answer to a question"* i.e. $\pi(R_n = Wrong)$ increases. For each $\pi(R_n = i)$ (i is the the outcome "Correct" or "Wrong"), we associate an element (possibility value) from the following totally ordered scale denoted by:

   $$\mathcal{L} = \{a_1 = 0.1, a_2 = 0.2, a_3 = 0.4, a_4 = 0.6, a_5 = 0.8, a_6 = 1\}$$

   such that:

   – $\pi(R_1 = Correct) \leq \pi(R_2 = Correct) \leq ... \leq \pi(R_{15} = Correct)$ and;

   – $\pi(R_1 = Wrong) \geq \pi(R_2 = Wrong) \geq ... \geq \pi(R_{15} = Wrong)$.

   Note that, the possibilistic scale $\mathcal{L}$ is numerical in order to compute the optimal strategy, then, only the order relation between values is significant, and not their real values (see Table 4.3 column 5 and 6).

Table 4.3: Numerical component of the game decision tree in the case of qualitative setting.

Q_n	X_n	C_n	X_{n-1}	$\pi(R_n = Correct)$	$\pi(R_n = Wrong)$
1	150	0	0	1	0.1
2	300	0	150	1	0.2
3	450	0	300	1	0.2
4	900	0	450	1	0.4
5	1800	0	900	1	0.6
6	2100	1800	1800	1	0.8
7	2700	1800	2100	1	1
8	3600	1800	2700	1	1
9	4500	1800	3600	1	1
10	9000	1800	4500	0.8	1
11	18000	9000	9000	0.6	1
12	36000	9000	18000	0.4	1
13	72000	9000	36000	0.2	1
14	144000	9000	72000	0.2	1
15	300000	9000	144000	0.1	1

2. **Quantitative representation of uncertainty**

   Probabilities used in the general model are derived from statistics (relative frequencies) produced on a large sample of the show [25]. Several researchers tackle different bridges between probability and possibility theories. Therefore, we aim to transform probabilistic distributions (of the general model) to possibilistic ones denoted by $p \rightarrow \pi$ using *probability-possibility transformation rules*.

   Several transformation rules are proposed in literature, the most common ones namely: Klir transformation (KT) [21], Optimal transformation (OT) [10] and Variable transformation (VT) [23] etc. In this work, we use the Klir transformation (KT) since it preserves appropriate scale and the amount of information contained in each distribution. It has considered the principle of uncertainty preservation under two scales, namely "The ratio scale" and "The log-interval scale", we will use the ratio scale named also the normalized transformations:

   Assume that the elements of Ω are ordered in such a way that: $\forall\ i = \{1..n\}$, $p_i > 0,\ pi > p_{i+1}$ and $p_{n+1} > 0$. We have:

$$\pi_i = \frac{p_i}{p_1}. \tag{4.4}$$

Table 4.4 shows the numerical compound of the game decision in the case of numerical interpretation: it presents probability distributions (column 5 and 6), where $p_{Q_n=c}$ (resp. $p_{Q_n=w}$) is the probability that the candidate selects the correct answer (resp. wrong answer), and transformed possibilistic distributions (column 7 and 8).

Example 4.5. *For* $\Omega_{15} = \{p_{Q_{15}} = c, p_{Q_{15}=w}\}$*, we have* $p_{Q_{15}=c} = 0.36$ *and* $p_{Q_{15}=w} = 0.64$*. By transforming* $p_{Q_{15}=i}$ *to* $\pi(R_{15} = i)$ *using 4.4, we obtain:*

- $\pi(R_{15} = Correct) = 0.36/0.64 = 0.56$.
- $\pi(R_{15} = Wrong) = 0.64/0.64 = 1$.

Table 4.4: Numerical component of the game decision tree in the case of quantitative setting.

Q_n	X_n	C_n	X_{n-1}	$p_{Q_n=c}$	$p_{Q_n=w}$	$\pi(R_n = Correct)$	$\pi(R_n = Wrong)$
1	150	0	0	0.97	0.03	1	0.03
2	300	0	150	0.95	0.05	1	0.05
3	450	0	300	0.92	0.08	1	0.08
4	900	0	450	0.86	0.14	1	0.16
5	1800	0	900	0.8	0.2	1	0.25
6	2100	1800	1800	0.79	0.21	1	0.26
7	2700	1800	2100	0.76	0.24	1	0.31
8	3600	1800	2700	0.63	0.37	1	0.58
9	4500	1800	3600	0.51	0.49	1	0.96
10	9000	1800	4500	0.43	0.57	0.75	1
11	18000	9000	9000	0.39	0.61	0.63	1
12	36000	9000	18000	0.38	0.62	0.61	1
13	72000	9000	36000	0.4	0.6	0.66	1
14	144000	9000	72000	0.37	0.63	0.58	1
15	300000	9000	144000	0.36	0.64	0.56	1

3. **Optimization of likely dominance**

   After the construction of the possibilistic decision tree that describes the sequential decision problem of "Who wants to be a millionaire?" game, we aim to apply the ExtDynProg and DynProg in order to obtain optimal strategies for this game.

   We first define a strategy in this model as the succession of optimal decisions that a candidate has to make in order to maximize his reward i.e. the question number where he must

quit the game. Table 4.5 summarizes optimal strategies obtained when optimizing likely dominance, in the case of qualitative setting ($\otimes = min$), and the corresponding expected reward.

Table 4.5: Optimal strategies of the game in the case of qualitative setting.

Q_n	$\delta^*_{L\Pi}(DynProg)$	$\delta^*_{L\Pi}(ExtDynProg)$	$\delta^*_{LN}(DynProg)$	$\delta^*_{LN}(ExtDynProg)$
1	Answer	Answer	Answer	Answer
2	Answer	Answer	Answer	Answer
3	Answer	Answer	Answer	Answer
4	Answer	Answer	Answer	Answer
5	Answer	Answer	Answer	Answer
6	Answer	Answer	Answer	Answer
7	Answer	Answer	Answer	Answer
8	Quit	Answer	Answer	Answer
9	-	Answer	Quit	Answer
10	-	Quit	-	Quit
11	-	-	-	-
12	-	-	-	-
13	-	-	-	-
14	-	-	-	-
15	-	-	-	-
Maximal expected reward	2700 Euro	4500 Euro	3600 Euro	4500 Euro

Note that:

- $\delta^*_{L\Pi}(DynProg)$ (resp. $\delta^*_{L\Pi}(ExtDynProg)$) is the "optimal" strategy returned by DynProg (resp. ExtDynProg) w.r.t. $L\Pi$.
- $\delta^*_{LN}(DynProg)$ (resp. $\delta^*_{LN}(ExtDynProg)$) is the "optimal" strategy returned by DynProg (resp. ExtDynProg) w.r.t. LN.

For instance, $\delta^*_{L\Pi}(ExtDynProg)$ shows that stopping at Q_{10} and answering all the previous questions is optimal to maximize the expected reward.

Table 4.6 summarizes optimal strategies obtained in the case of quantitative setting ($\otimes = *$), and the corresponding expected reward.

Table 4.6: Optimal strategies of the game in the case of quantitative setting.

Q_n	$\delta^*_{L\Pi}(DynProg)$	$\delta^*_{L\Pi}(ExtDynProg)$	$\delta^*_{LN}(DynProg)$	$\delta^*_{LN}(ExtDynProg)$
1	Answer	Answer	Answer	Answer
2	Answer	Answer	Answer	Answer
3	Answer	Answer	Answer	Answer
4	Answer	Answer	Answer	Answer
5	Answer	Answer	Answer	Answer
6	Answer	Answer	Answer	Answer
7	Quit	Answer	Answer	Answer
8	-	Answer	Quit	Answer
9	-	Answer	-	Answer
10	-	Quit	-	Quit
11	-	-	-	-
12	-	-	-	-
13	-	-	-	-
14	-	-	-	-
15	-	-	-	-
Maximal expected reward	2100 Euro	4500 Euro	2700 Euro	4500 Euro

Both tables (4.5 and 4.6) show that optimal strategies returned by ExtDynProg are better (maximize expected reward) than that returned by DynProg, w.r.t. $L\Pi$ and LN, in any case. Actually, in this decision tree we have equivalent states e.g. in the case of qualitative setting, for question Q_8 and Q_{11} w.r.t. $L\Pi$, and this is the main cause of the fail of DynProg.

Our result seems realistic since answering correctly depends on certain possibility value of question difficulty and in our model we estimate that the candidate is enable to give the right answer starting Q_7 (resp. Q_{10}) in the case of qualitative setting (resp. quantitative setting).

4. Optimization of possibilistic decision criteria

We propose to compare adventurous decision makers behavior and cautious decision makers behavior i.e. analyze results of all criteria presented in Section 1.5. We report optimal strategies computed with DynProg w.r.t. U_{opt} and U_{pes}, BB w.r.t. Ch_{Π} and Ch_N and ExtDynProg w.r.t. $L\Pi$ and LN.

Table 4.7 presents optimal strategies for adventurous decision makers i.e. U_{opt} using DynProg, Ch_{Π} using BB and $L\Pi$ using ExtDynProg. Our results confirm that solutions provided by DynProg, when optimizing U_{opt}, and BB, when optimizing Ch_{Π}, maximize the expected reward. However, these strategies are risky compared to those provided by ExtDynProg w.r.t. $L\Pi$.

Table 4.7: Optimal strategies of the game for adventurous decision makers.

Q_n	$\delta^*_{U_{opt}}(DynProg)$	$\delta^*_{Ch_{\Pi}}(BB)$	$\delta^*_{L\Pi}(ExtDynProg)$
1	Answer	Answer	Answer
2	Answer	Answer	Answer
3	Answer	Answer	Answer
4	Answer	Answer	Answer
5	Answer	Answer	Answer
6	Answer	Answer	Answer
7	Answer	Answer	Answer
8	Answer	Answer	Answer
9	Answer	Answer	Answer
10	Answer	Answer	Quit
11	Answer	Answer	-
12	Answer	Answer	-
13	Answer	Quit	-
14	Quit	-	-
15	-	-	-
Maximal expected reward	72000 Euro	36000 Euro	4500 Euro

Table 4.8 presents optimal strategies for cautious decision makers i.e. U_{pes} using DynProg, Ch_N using BB and LN using ExtDynProg. The optimal strategy returned by BB algorithm

when optimizing Ch_N maximize the expected reward i.e. 18000 Euro.

Table 4.8: Optimal strategies of the game for cautious decision makers.

Q_n	$\delta^*_{U_{pes}}(DynProg)$	$\delta^*_{Ch_N}(BB)$	$\delta^*_{LN}(ExtDynProg)$
1	Answer	Answer	Answer
2	Answer	Answer	Answer
3	Answer	Answer	Answer
4	Answer	Answer	Answer
5	Answer	Answer	Answer
6	Answer	Answer	Answer
7	Answer	Answer	Answer
8	Answer	Answer	Answer
9	Answer	Answer	Answer
10	Answer	Answer	Quit
11	Quit	Answer	-
12	-	Quit	-
13	-	-	-
14	-	-	-
15	-	-	-
Maximal expected reward	9000 Euro	18000 Euro	4500 Euro

4.5 Conclusion

In this chapter, main programs, the major evaluation measures, and the experimental process are detailed. Our experiments, provided in this chapter, show that Extended dynamic programming is not so time consuming (below 3 sec for problems involving about 2000 nodes). On the other hand, classical Dynamic programming is much faster and provides good results in almost any case.

We have experimented the proposed algorithm on a real decision problem: the TV game ”Who wants to be a millionaire?”. The contestant must answer a sequence of multiple-choice

questions (four possible answers) of increasing difficulty, numbered from 1 to 15. We have proposed a description of the game as a sequential decision problem representing with a possibilistic decision tree and solved by Extended dynamic programming, Dynamic programming and Branch and bound algorithm. Indeed, we proved the robustness of those algorithms especially Extended dynamic programming that find conservative optimal strategies. However, these strategies significantly limit the potential gain regarding possibilistic qualitative utilities and possibilistic Choquet integrals.

Conclusion and perspectives

When the information about uncertainty cannot be quantified in a simple, probabilistic way, the topic of possibilistic decision theory is often a natural one to consider [9, 11, 13]. In This reminder we have been interested to sequential decision problem in possibilistic trees where decision maker can choose between several punctual actions spread over time with incapacity to predict, with certainty, the outcome event. This problem exists in multiple domains such as automatic control, the planning of actions of robots, in-time stock management, modeling games etc.

To represent these problems several graphical models have been proposed, we have been interested in the formalism of decision trees since they allow an explicit representation of a problem of sequential decision. On the other hand, several possibilistic decision criteria were established to support possibilistic decision theory, e.g. optimistic and pessimistic possibilistic Qualitative Utilities [13], possibilistic Choquet Integrals [27], binary possibilistic utility [17] and possibilistic likely dominance [9, 15].

Evaluating a possibilistic decision tree w.r.t. a possibilistic decision criteria i.e. finding optimal strategy in a possibilistic decision tree is an algorithmic problem that depends on the properties of the decision criteria. The preference order induced by Qualitative Utilities criteria is monotonic and completely transitive. These properties allow a polytime solving of the problem by DynProg. For possibilistic Choquet Integrals, on the contrary, it has been shown in [4] that the problem of finding an optimal strategy is NP-hard. It can be solved by a Branch and bound algorithm [4, 2].

In this work we have focused on the particular case of possibilistic likely dominance criteria, where the preference order induced on the lotteries is monotonic but only quasi-transitive, which implies that classical Dynamic programming may fail to find an optimal strategy. Nevertheless, these properties (monotonicity and quasi-transitivity) allow us to propose a polytime adaptation of Dynamic programming, namely, Extended dynamic programming.

In order to evaluate our algorithm, we have proposed an experimental study based on the comparison between Dynamic programming and Extended dynamic programming. In fact we have proposed three new evaluation measures and this is was necessary because likely dominance criteria make a pairwise comparison. Our experiments prove the effectiveness of this algorithm especially for necessity-based likely dominance. However, we observed that Dynamic programming is faster and provides good results practically as Extended dynamic programming. Nevertheless, the quality of the solution provided by Dynamic Programming is not so good when this algorithm fails to get an optimum, hence the interest of the Extended algorithm.

Our results are limited by the difficulty of randomly generated instances i.e. generated trees are balanced. Indeed, strategies lead to lotteries that can be quite similar which is not an encouraging framework. To be convinced that Extended dynamic programming can handle real-life situations we applied it to a concrete problem: the popular game show ”Who wants to be a millionaire?”. The results of experiments are quite encouraging: regarding Dynamic programming, Extended dynamic programming provides best strategies, w.rt. likely dominance criteria, that maximize expected reward. However, regarding qualitative utilities and Choquet integrals criteria, optimal strategies w.r.t. likely dominance criteria have the minimal expected reward.

As future work, we plan to address the following tasks:

- To adapt Extended dynamic programming to more sophisticated decision models, such as possibilistic influence diagrams [16] and possibilistic Markov decision processes [28].
- To look for other decision criteria (probabilistic or possibilistic), analyze and optimize them using the appropriate optimizing strategy algorithm.
- The example of the TV game can be a motivating case to treat more critical problems such as medical diagnostic test and therapy planning.

Bibliography

[1] M. Allais. Le Comportement de l'Homme Rationnel devant le Risque: Critique des Postulats et Axiomes de l'Ecole Americaine. *Econometrica*, 21(4):503–546, 1953.

[2] N. Ben Amor, H. Fargier, and W. Guezguez. Necessity-based choquet integrals for sequential decision making under uncertainty. *IPMU'2010*, pages 222–236, 2010.

[3] N. Ben Amor, H. Fargier, and W. Guezguez. On the complexity of decision making in possibilistic decision trees. In *UAI'11*, pages 203–210, 2011.

[4] N. Ben Amor, H. Fargier, and W. Guezguez. Possibilistic sequential decision making. *International Journal of Approximate Reasoning*, 55:1269 – 1300, 2014.

[5] N. Ben Amor, Z. El Khalfi, H. Fargier, and W. Guezguez. Optimiser les critères de dominance possibiliste par programmation dynamique : principes et expérimentations. *Accepted for publication in 23 éme rencontres francophones sur la Logique Floue et ses Applications*, 2014.

[6] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[7] R. Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, tm.6, 1957.

[8] G. Choquet. Theory of capacities. *Annales de l'institut Fourier*, 5:131–295, 1954.

[9] D. Dubois, H. Fargier, and P. Perny. Qualitative decision theory with preference relations and comparative uncertainty: An axiomatic approach. *Artif. Intell.*, 148(1-2):219–260, 2003.

[10] D. Dubois, L. Foulloy, G. Mauris, and H. Prade. Probability-possibility transformations, triangular fuzzy sets, and probabilistic inequalities. *Reliable Computing*, 10(4):273–297, 2004.

[11] D. Dubois, L. Godo, H. Prade, and A. Zapico. Making decision in a qualitative setting: from decision under uncertainty to case-based decision. In *6th International Conference on Principles of Knowledge Representation and Reasoning*, KR'1998, pages 594–605, 1998.

[12] D. Dubois and H. Prade. *Possibility Theory: An Approach to Computerized Processing of Uncertainty*. Plenum Press, New York, 1988.

[13] D. Dubois and H. Prade. Possibility theory as a basis for qualitative decision theory. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1924–1930, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[14] D. Ellsberg. Risk, ambiguity, and the savage axioms. *The Quarterly Journal of Economics*, 75(4):643–669, 1961.

[15] H. Fargier and P. Perny. Qualitative models for decision under uncertainty without the commensurability assumption. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, pages 188–195, 1999.

[16] L. Garcia and R. Sabbadin. Possibilistic influence diagrams. In *ECAI'06*, pages 372–376, 2006.

[17] P. 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*, UAI'01, pages 162–170, 2001.

[18] P. Giang and P.P. Shenoy. Two axiomatic approaches to decision making using possibility theory. *European Journal of Operational Research*, 162(2):450–467, 2005.

[19] R.A. Howard and J.E. Matheson. Influence diagrams. *In the principles and applications of decision analysis*, 2, 1984.

[20] G. Jeantet and O. Spanjaard. Rank-dependent probability weighting in sequential decision problems under uncertainty. In Jussi Rintanen, Bernhard Nebel, J. Christopher Beck, and Eric A. Hansen, editors, *ICAPS*, pages 148–155. AAAI, 2008.

[21] G.J Klir and B. Parviz. Probability-possibility transformations: a comparison. *International Journal of General Systems*, 21(3):291–310, 1992.

[22] E.E. Lehmann and S.Warning. The impact of gender on individual decisions: Evidence from the ”millionaire show”. Technical Report 325, Konstanz, 2003.

[23] M.S. Mouchaweh, S.M Bouguelid, P. Billaudel, and B.Riera. Variable probability-possibility transformation. In *25th European Human Decision-Making and Manual Control*, pages 417–428, Valenciennes, France, 2006.

[24] J. Neumann and O. Morgenstern. Theory of games and economic behavior. In *Princeton University Press*, 1948.

[25] 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.

[26] H. Raiffa. *Decision Analysis – Introductory Lectures on Choices under Uncertainty*. Addison-Wesley, Reading, MA, 1968.

[27] Y. Rebille. Decision making over necessity measures through the choquet integral criterion. *Fuzzy Sets and Systems*, pages 3025–3039, 2006.

[28] R. Sabbadin, H. Fargier, and J. Lang. Towards qualitative approaches to multi-stage decision making. *International Journal of Approximate Reasoning*, 19:441 – 471, 1998.

[29] L.J. Savage. *The Foundations of Statistics*. J. Wiley, New York, 1954. second revised edition, 1972.

[30] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[31] P.P. Shenoy. Fuzzy logic for the management of uncertainty. chapter Valuation-based Systems: A Framework for Managing Uncertainty in Expert Systems, pages 83–104. John Wiley & Sons, Inc., New York, NY, USA, 1992.

[32] L.A. Zadeh. Fuzzy sets. *Information Control*, 8:338–353, 1965.

[33] L.A. Zadeh. Fuzzy sets as a basis for theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

Appendix 1

Proof. (Proposition 3.1)
We give the proof for $\forall L, L' \in \mathcal{L}, L \succeq_{L\Pi} L' \iff L^{core} \succeq_{L\Pi} L'$ (The proof of $\forall L, L' \in \mathcal{L}, L' \succeq_{L\Pi} L \iff L' \succeq_{L\Pi} L^{core}$ is similar).

$L \succeq_{L\Pi} L' \iff \Pi(L \geq L') \geq \Pi(L' \geq L)$. Because the possibility distributions are normalized, $max(\Pi(L \geq L') \geq \Pi(L' \geq L) = 1)$: $L \succeq_{L\Pi} L'$ iff $\Pi(L \geq L') = 1$.
$\Pi(L \geq L') = 1$ if there exists $u \in L_{=1}$ and $u' \in L'_{=1}$ such that $u \geq u'$. This condition is equivalent to $\max(L_{=1}) \geq u'$.

Then $L \succeq_{L\Pi} L'$ implies that $L^{core} \succeq_{L\Pi} L'$ (the core of L^{core} precisely contains $\max(L_{=1})$). Reciprocally, if $L^{core} \succeq_{L\Pi} L'$, then $\exists u'$ s.t. $\max(L_{=1}) \geq u'$, and thus $L \succeq_{L\Pi} L'$. □

Proof. (Proposition 3.2)
We give the proof for $\forall L, L' \in \mathcal{L}, L \succeq_{LN} L' \iff L^{2nd} \succeq_{LN} L'$
(The proof of $L' \succeq_{LN} L \iff L' \succeq_{LN} L^{2nd}$ is similar).

$L \succeq_{LN} L'$ iff $1 - \Pi(L' \succ L) \geq 1 - \Pi(L \succ L')$, which is equivalent to $\Pi(L \succ L') \geq \Pi(L' \succ L)$. This inequality is satisfied in three cases only:

- $\max(L_{=1}) > \min(L'_{=1})$ (then $\Pi(L \succ L') = 1$).
- $\max(L_{=1}) = \min(L'_{=1}) = \max(L'_{=1}) = \min(L_{=1})$ (let u denote this value), $\alpha^{2nd} \geq \alpha'^{2nd}$ and $\max(L_{=\alpha^{2nd}}) > u$ (then $\Pi(L \succ L') = \alpha^{2nd}$ and $\Pi(L' \succ L) \leq \alpha^{2nd}$).
- $\max(L_{=1}) = \min(L'_{=1}) = \max(L'_{=1}) = \min(L_{=1})$ (let u denote this value), $\alpha'^{2nd} \geq \alpha^{2nd}$ and $u > \min(L'_{=\alpha'^{2nd}})$.

Since L^{2nd} precisely contains $\max(L_{=1})$ and $\min(L_{=1})$ (at level 1) and $\max(L_{=\alpha^{2nd}})$ (at level α^{2nd}), $L \succeq_{LN} L' \implies L^{2nd} \succeq L'$.

Reciprocally, if $L^{2nd} \succeq L'$, then it is easy to check that one of the three previous cases is fulfilled and thus $L \succeq_{LN} L'$. □

Proof. (Proposition 3.3)
Let $L = \langle 1/u, \alpha/m, \alpha/M \rangle$ and $L' = \langle 1/u', \alpha'/m', \alpha'/M' \rangle$, with $\alpha \neq \alpha'$ (Similar results hold for $\alpha' > \alpha$).

- If $u \neq u'$, then only the modes are used to make the decision: either $u > u'$, and then $L \succ_{LN} L'$ i.e. $\Pi(L > L') = 1$ and $\Pi(L' < L) < 1$; or $u' > u$ and then $L' \succ_{LN} L$ i.e. $\Pi(L > L') < 1$ and $\Pi(L' < L) = 1$.

- If $u = u'$, decision is made at level $\max(\alpha, \alpha')$; when $\alpha > \alpha'$, then either $u > M$ (and then $L' \succ_{LN} L$); or $u < m$ (and then $L \succ_{LN} L'$); or $M > u > m$ (and then $L \sim_{LN} L'$).

So, $u = u'$ is a necessary condition for the indifference of the two lotteries.

Moreover, $M = u$ (resp. $m = u$) is not possible since the possibility of u is equal to 1 for both L and L' and the possibility of M (resp. m) is strictly lower than 1. Thus two cases are possible:

- Case 1: $M = m$, then the two possible cases are:

  - If $u > M = m$, then $\Pi(L' > L) = \alpha$ (at level α, L' can take value u and L value m) and $\Pi(L > L') < \alpha$ (one have go below α to find a value for L strictly greater than u); $\Pi(L > L')$ cannot be equal to $\Pi(L' > L')$, i.e. L and L' cannot be indifferent.

  - If $u < M = m$, then $\Pi(L > L') = \alpha$ (at level α, L' can take value u and L value m) and $\Pi(L' > L) < \alpha$ (one have go below α to find a value for L strictly lower than u); i.e. L and L' cannot be indifferent.

- Case 2: $M \neq m$, then by construction $M > m$ i.e. three cases are possible:

  - If $u > M$, then $\Pi(L' > L) = \alpha$ (at level α, L' can take value u and L value M) and $\Pi(L > L') < \alpha$ (one have go below α to find a value for L strictly greater than u). So $L \sim_{LN} L'$ is not satisfied.

- If $u < m$, then $\Pi(L > L') = \alpha$ (at level α, L' can take value u and L value m) and $\Pi(L' > L) < \alpha$ (one have go below α to find a value strictly lower than u). So $L \sim_{LN} L'$ implies $m \leq u$. $L \sim_{LN} L'$ is not satisfied.

- If $M > u > m$, then $\Pi(L > L') = \alpha = \Pi(L' > L)$.

Hence, when $M > m$, $L \sim_{LN} L'$ implies $M > u > m$.
Reciprocally, it is easy to check that $M > u > m$ and $\alpha > \alpha'$ imply $\Pi(L > L') = \alpha$ and $\Pi(L' > L) = \alpha$. □

Proof. (Proposition 3.4, item (i))
We have $L_1 = \langle \gamma/L, \beta/L'' \rangle$ and $L_2 = \langle \gamma/L', \beta/L'' \rangle$ where $L = \langle 1/u, \alpha/m, \alpha/M \rangle$ and $L' = \langle 1/u, \alpha'/m', \alpha'/M' \rangle$ are two indifferent 2nd core lotteries (i.e. $L \sim_{LN} L'$), $\alpha < \alpha'$ and $\max(\gamma, \beta) = 1$.

Thanks to Proposition 3.2 we can assume without loss of generality that L_1, L_2 and L_b are 2nd core lotteries (if it is not the case, simply replace them by their 2nd core lotteries). Since the cores of L and L' are equal, the cores of L_1 and L_2 are also equal. So L_1 is unimodal iff L_2 is unimodal (and both share the same mode).

- If L_b is not unimodal, then the decision is made at possibility level 1, where L_1 and L_2, are identical. Hence the property.

- If L_1 and L_2 are not unimodal, then the decision is made at possibility level 1, where L_1 and L_2, are identical. Hence the property.

- If the three lotteries are unimodal but the mode of L_1 and L_2 is not equal to the one of L_b, then the decision is made at possibility level 1, where L_1 and L_2 are identical. Hence the property.

- If L_b, L_1 and L_2 are unimodal and share the same mode, u.
  L_1 can be written as $\langle 1/u, a_1/m_1, a_1/M_1 \rangle$, with $m_1 < M_1$ and $a_1 < 1$,
  L_2 can be written as $\langle 1/u, a_2/m_2, a_2/M_2 \rangle$, with $m_2 < M_2$ and $a_2 < 1$,
  L_b can be written as $\langle 1/u, b/m_b, b/M_b \rangle$, with $m_b < M_b$ and $b < 1$.

  Notice that $u \notin \{M_1, m_1, M_2, m_2, M_b, m_b\}$ (the possibility degree of u is equal to 1 while the possibility degrees of M_1, m_1, M_2, m_2, M_b and m_b are equal to a_1, a_2 or b which are all strictly less than 1).

  Since $\alpha < \alpha'$, $a_1 < a_2$. Moreover, LN satisfies the principle of weak monotonicity, thus $L \sim_{LN} L'$ implies $L_1 \sim_{LN} L_2$. Thanks to Proposition 3.3, $L_1 \sim_{LN} L_2$ and $a_1 < a_2$ imply

$m_2 < u < M_2$.

By hypothesis, $L_2 \succ_{LN} L_b$. We have $m_2 < u < M_2$, suppose that $a_2 \geq b$, this implies $\Pi(L_2 > L_b) = a_2$ and $\Pi(L_b > L_2) = a_2$. Thus $a_2 \geq b$ contradicts the hypothesis. Hence $b > a_2$.

Since $b > a_2$, $\Pi(L_2 > L_b) > \Pi(L_b > L_2)$ iff $\Pi(L_2 > L_b) = b$ and $\Pi(L_b > L_2) < b$. Which means that at level b, $u > M_b$ (if $u < M_b$, $\Pi(L_b > L_2) = b$ and $u = M_b$ is not possible).

$a_1 < b$ (Since $a_1 < a_2$ and $a_2 < b$): the comparison of L_1 and L_b is made at level b. From $u > M_b$ we conclude $\Pi(L_1 > L_b) = b$ and $\Pi(L_b > L_1) < b$, i.e. $L_1 \succ_{LN} L_b$. Hence when L_b, L_1 and L_2 are unimodal and share the same mode, $L_2 \succ_{LN} L_b$ implies $L_1 \succ_{LN} L_b$.

□

Proof. (Proposition 3.4, item (ii))
The proof is similar to the previous one. Thanks to Proposition 3.2 we can assume without loss of generality that L_1, L_2 and L_b are 2nd core lotteries. Consider the case where they are unimodal and share the same mode, u (if not, refer to the previous proof). Then:
L_1 can be written as $\langle 1/u, a_1/m_1, a_1/M_1 \rangle$, with $m_1 < M_1$ and $a_1 < 1$,
L_2 can be written as $\langle 1/u, a_2/m_2, a_2/M_2 \rangle$, with $m_2 < M_2$ and $a_2 < 1$,
L_b can be written as $\langle 1/u, b/m_b, b/M_b \rangle$, with $m_b < M_b$ and $b < 1$.

Following the previous proof, we have $u \notin \{M_1, m_1, M_2, m_2, M_b, m_b\}$ and also $L_1 \sim_{LN} L_2$ and $a_1 < a_2$ imply $m_2 < u < M_2$.

By hypothesis, $L_b \succ_{LN} L_2$. But $a_2 \geq b$ and $m_2 < u < M_2$ imply that $\Pi(L_2 > L_b) = a_2$ and $\Pi(L_b > L_2) = a_2$. Thus $a_2 \geq b$ contradicts the hypothesis. Hence $b > a_2$.

Since $b > a_2$, $\Pi(L_2 < L_b) > \Pi(L_b < L_2)$ iff $\Pi(L_2 < L_b) = b$ and $\Pi(L_b < L_2) < b$. Which means that at level b, $u < m_b$ (if $u > m_b$, $\Pi(L_b < L_2) = b$ and $u = m_b$ is impossible).

$a_1 < b$ (Since $a_1 < a_2$ and $a_2 < b$): the comparison of L_1 and L_b is made at level b. From $u < m_b$ we conclude that $\Pi(L_1 < L_b) = b$ and $\Pi(L_b < L_1) < b$, i.e. $L_1 \succ_{LN} L_1$. Hence, when L_1, L_2 and L_b are unimodal and share the same mode, $L_b \succ_{LN} L_2$ implies $L_b \succ_{LN} L_1$. □