Optimisation du critère d'Hurwicz pour les arbres de décision hasard en situation d'incertain total

Gildas Jeantet

MajecSTIC 2009
Avignon, France, du 16 au 18 novembre 2009

Optimisation du critère d'Hurwicz pour les arbres de décision hasard en situation d'incertain total

Gildas Jeantet[1]

1 : Laboratoire d'Informatique de Paris 6 (LIP6-CNRS), Université Pierre et Marie Curie (UPMC), 104 avenue du Président Kennedy, 75016 Paris, France.

Contact : gildas.jeantet@lip6.fr

Résumé
Cet article est consacré aux problèmes de décision séquentielle dans l'incertain lorsque le décideur a une ignorance complète sur les probabilités des événements. Nous étudions ici le problème de la détermination d'une stratégie optimale au sens d'Hurwicz dans les arbres de décision hasard. Après avoir montré que l'approche par programmation dynamique classique est inopérante pour optimiser le critère d'Hurwicz dans un arbre de décision hasard, nous proposons un algorithme polynomial pour résoudre ce problème. Enfin nous fournissons des tests numériques effectués sur des instances générées aléatoirement pour illustrer les performances de notre algorithme.

Abstract
This article is devoted to sequential decision problems under complete ignorance. We study here the problem of determining an optimal strategy according to the Hurwicz criterion in decision trees. After showing that the rolling back method is sub-optimal for the Hurwicz criterion in a decision tree, we propose a polynomial algorithm to solve this problem. Finally we provide numerical tests performed on randomly generated instances to illustrate the performance of our algorithm.

Mots-clés : théorie de la décision algorithmique, décision séquentielle dans l'incertain, arbre de décision hasard

Keywords: algorithmic decision theory, decision under uncertainty, decision tree

1. Introduction

De nombreuses problématiques en IA peuvent se formaliser comme un problème de *décision séquentielle dans l'incertain* : navigation de robot, diagnostics médicaux ou encore dans l'élaboration de joueurs artificiels. La décision séquentielle dans l'incertain concerne les problèmes de décision dans lesquels les conséquences des actions ne sont pas connues avec certitude. Lorsque cette incertitude est probabilisée, on parle de *décision séquentielle dans le risque*. Dans un cadre risqué, de nombreux modèles décisionnels ont été proposés, dont le plus connu est l'espérance d'utilité [8] lorsqu'il s'agit de décrire des comportements décisionnels dit *conséquentialiste* (comportements dans lesquels seules les perspectives futures sont prises en compte pour prendre une décision) et le modèle de l'espérance d'utilité dépendant du rang [5] pour traduire des comportements décisionnels plus complexes. Cependant, il n'est pas toujours possible de connaître les probabilités des événements et ces modèles ne sont alors plus applicables dans un cadre d'incertitude non probabiliste. Nous nous intéressons ici au cas où le décideur n'a aucune information sur la plausibilité d'un événement. On parle alors d'*incertain total* ou d'*ignorance complète*. Dans ce cadre, le décideur doit prendre plusieurs décisions séquentiellement (établir une *stratégie*) menant à plusieurs conséquences potentielles. Chaque conséquence est évaluée numériquement par une *fonction d'utilité* (propre à chaque décideur) traduisant l'importance accordée par le décideur à une conséquence. Dans un tel cadre on s'intéresse alors à déterminer la stratégie offrant les récompenses les plus attractives pour le décideur. Le critère d'Hurwicz permet précisément d'évaluer

une stratégie dans ce cadre en prenant en compte le degré de pessimisme du décideur. Dans une première partie (section 2) nous faisons un bref état de l'art sur les différents critères de décision proposés dans la littérature pour le cadre de l'incertain total, puis nous rappelons (section 3) le formalisme des arbres de décision hasard (modèle graphique permettant de réprésenter un problème de décision séquentielle dans l'incertain). Dans un second temps (section 4) nous mettons en avant le côté combinatoire induit par les arbres de décision hasard et montrons que le critère d'Hurwicz est incohérent dynamiquement. Ces deux points nous conduisent alors à developper un algorithme de résolution fondée sur la programmation dynamique biobjectif (section 5). Enfin nous illustrons les performances de cet algorithme à l'aide d'expérimentations numériques réalisées sur des instances générées aléatoirement.

2. Critères de décision dans l'incertain total

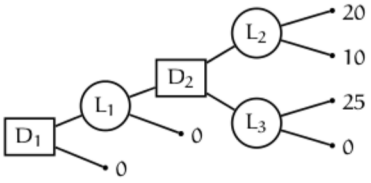
Nous présentons ici les critères de décision les plus couramment cités dans la littérature pour le cas de l'incertain total. Le décideur est amené à choisir entre plusieurs décisions dont les conséquences sont incertaines. Considérons la situation de choix sous incertitude représentée par le tableau A dans la figure 1. Les décisions $D = \{d_1, d_2, d_3\}$ sont notées horizontalement, et les états du monde possibles $\Theta = \{\theta_1, \theta_2, \theta_3\}$ sont notés verticalement. Le nombre à l'intersection de la ligne d_i et de la colonne θ_j est l'utilité $u(d_i, \theta_j)$ de la conséquence de la décision d_i si θ_j se réalise. Les critères les plus simples sont *Maximax* et *Maximin*, qui consistent à prendre la décision offrant la meilleure conséquence pour maximax (i.e. $\max_{d \in D} \max_{\theta \in \Theta} u(d, \theta)$) ou à prendre la décision qui maximise la pire conséquence pour maximin (i.e. $\max_{d \in D} \min_{\theta \in \Theta} u(d, \theta)$). Ainsi, pour le critère maximax, dans le problème A de la figure 1, la décision d_1 est préférée à d_3 qui est elle-même préférée à d_2. Pour le critère maximin, la décision d_2 est préférée à d_3 qui est elle-même préférée à d_1. Ces critères sont bien évidemment peu discriminants et peuvent *a priori* mener à des préférences discutables. Dans l'optique d'améliorer le critère maximin, Savage [7] propose le critère du *Minimax-Regret*. L'idée motrice est de calculer une matrice de "regrets" à partir de la matrice des conséquences. Le regret se calcule, pour chaque décision d_i, en faisant la différence entre la meilleure utilité qui puisse advenir et l'utilité de la conséquence de l'événement θ_j considéré. On choisit ensuite la décision dont le "regret" maximum est minimal. En d'autres termes on pose $r_{ij} = \max_{\theta \in \Theta} u(d_i, \theta) - u(d_i, \theta_j)$ puis on prend une décision dont la valeur est $\min_i \max_j r_{ij}$. Par exemple, pour le problème A de la figure 1, la matrice de regrets obtenue est représentée sur le tableau B de la figure 1, ce qui conduit aux préférences suivantes : d_2 est préférée à d_1 qui est elle-même préférée à d_3. Le grand reproche généralement émit à l'encontre de ce critère est de ne pas être indépendant vis-à-vis d'une tierce alternative. En d'autres termes, les préférences induites par ce critère sont instables en fonction des décisions possibles. Par exemple si on retire la première décision on obtient la matrice de regret représentée par le tableau C (figure 1), ce qui nous conduit aux préférences suivantes : d_3 préféré à d_2. Ce renversement de préférence est un comportement non souhaitable pour un critère de décision. Un autre critère souvent employé dans la littérature, est le critère de Laplace (aussi appelé le critère de la raison insuffisante). Ce critère stipule qu'en cas d'incertitude totale, il faut supposer *a priori* l'équiprobabilité des événements de la nature, et choisir la décision qui offre la plus grande espérance d'utilité. Les préférences induites par ce critère pour le problème A de la figure 1 sont les suivantes : d_1 et d_2 sont indifférentes, mais sont préférées à d_3. L'application de ce critère bute cependant sur la question de l'identification des états du monde futurs. En d'autres termes la présence d'un événement peut mener à un renversement de préférence. Enfin un dernier critère fréquemment employé dans la littérature est le critère d'Hurwicz. Il généralise les critères maximax et maximin. Ce critère propose d'évaluer une décision en prenant en compte simultanément la plus petite et la plus grande conséquence. Plus formellement, l'évaluation d'une décision d se fait par la formule suivante : $\alpha \min_{\theta \in \Theta} u(d, \theta) + (1 - \alpha) \max_{\theta \in \theta} u(d, \theta)$ avec $\alpha \in [0; 1]$. Il offre ainsi la possiblité de s'adapter au décideur selon son degré de pessimisme. En effet, plus α est proche de 1, plus le décideur prendra en compte la plus mauvaise conséquence potentielle pour évaluer la stratégie, ce qui signifie qu'il va tendre à maximiser la plus petite conséquence potentielle, et inversément, plus α sera proche de 0 plus le décideur prendra en compte la plus grande conséquence pour évaluer une stratégie. Nous verrons par la suite que le critère d'Hurwicz induit une difficulté algorithmique lorsqu'on l'applique à des problèmes de décision séquentielle.

A	θ_1	θ_2	θ_3
d_1	11	0	5
d_2	5	9	2
d_3	1	4	10

B	θ_1	θ_2	θ_3
d_1	0	9	5
d_2	6	0	8
d_3	10	5	0

C	θ_1	θ_2	θ_3
d_2	0	0	8
d_3	4	5	0

FIG. 1 – Exemple de situation sous incertitude totale et matrices de regret.



D_1	D_2	C_s	$\alpha = 0$	0.2	0.5	0.8	1
Haut	Haut	{0;10;20}	20	16	10	4	0
Haut	Bas	{0;0;25}	25	20	12.5	5	0
Bas	–	{0}	0	0	0	0	0

FIG. 2 – Exemple d'arbre de décision et évaluation de ses stratégies par le critère d'Hurwicz.

3. Arbres de décision hasard

Un *arbre de décision hasard* [6] est un formalisme simple et explicite pour représenter une situation de décision séquentielle dans l'incertain. Il s'agit d'une arborescence dont l'ensemble des noeuds $\mathcal{N}$ peut se diviser en trois sous-ensembles de noeuds disjoints : l'ensemble $\mathcal{N}_D$ des *nœuds de décision* (représentés par des carrés), l'ensemble $\mathcal{N}_C$ des *nœuds de hasard* (représentés par des cercles) et l'ensemble $\mathcal{C}$ des conséquences (feuilles de l'arborescence). Les arcs issus d'un noeud de décision représentent les différentes alternatives envisageables par le décideur, tandis que les arcs issus des noeuds de hasard représentent les différents événements extérieurs pouvant se produire. Un exemple d'arbre de décision est donné dans la figure 2. Le décideur est amené à prendre une première décision en D_1, lui laissant le choix entre une conséquence nulle (décision Bas) ou d'aller sur le noeud de hasard L_1 (décision Haut). En L_1, un événement indépendant de la volonté du décideur conditionne la suite des événements : il peut soit se retrouver en D_2 soit se retrouver à une conséquence nulle. S'il est en D_2 le décideur a le choix entre deux décisions et ainsi de suite. On appelle stratégie, un ensemble d'arcs de décision, pour lequel toutes les situations ont été envisagées. Par exemple dans l'arbre de décision hasard de la figure 2, il y a exactement trois stratégies : $s_1 = \{D_1 = \text{Haut}, D_2 = \text{Haut}\}$, $s_2 = \{D_1 = \text{Haut}, D_2 = \text{Bas}\}$ et $s_3 = \{D_1 = \text{Bas}\}$. Chaque stratégie s conduit à un ensemble de conséquences potentielles $C_s \subset \mathcal{C}$. Par exemple, la stratégie $s_1 = \{D_1 = \text{Haut}, D_2 = \text{Haut}\}$ conduit à l'ensemble de conséquences potentielles $C_{s_1} = \{0, 10, 20\}$. Nous supposons ici que l'évaluation d'une stratégie se fait uniquement à partir des conséquences potentielles auxquelles elle conduit. Dans un tel formalisme, le critère d'Hurwicz se définit alors comme suit : soit s une stratégie menant à l'ensemble de conséquences potentielles $C_s = \{u_1, u_2, ..., u_n\}$ de sorte que $u_1 \leq u_2 \leq ... \leq u_n$, l'évaluation de la stratégie s par le critère d'Hurwicz est $\alpha u_1 + (1-\alpha)u_n$ où α est un réel compris entre 0 et 1. Le tableau de la figure 2 donne la valeur de chaque stratégie de l'arbre de décision hasard de la figure 2 en fonction de quelques valeurs de α. Les colonnes D_1 et D_2 indiquent les décisions correspondantes en chacun des noeuds de décision pour la stratégie, la colonne C_s donne l'ensemble des conséquences potentielles offertes par la stratégie. Un décideur ayant des préférences qui suivent le critère d'Hurwicz sélectionne alors la stratégie dont l'évaluation par le critère d'Hurwicz est maximale.

4. Détermination d'une stratégie optimale au sens du critère d'Hurwicz

Dans cette section nous montrons que l'espace des stratégies est combinatoire, puis nous montrons comment l'approche classique par programmation dynamique, généralement utilisée pour le calcul de la stratégie optimale dans les problèmes de décision séquentielle, peut aboutir ici à

une stratégie sous-optimale.

Espace des solutions. Considérons un arbre décision hasard binaire complet de profondeur $2p$ tel que les nœuds de profondeur paire soient des nœuds de décision (ou des conséquences) et les nœuds de profondeur impaire soient des nœuds de hasard. Nous nous intéressons ici à comptabiliser le nombre de stratégies possibles (autrement dit de solutions réalisables) en fonction de la taille de l'instance. On définit comme taille de l'instance le nombre de nœuds de décision. Ce nombre est en effet du même ordre de grandeur que le nombre de nœuds de l'arbre. Remarquons qu'il y a 1 nœud de décision de profondeur 0, 4 nœuds de décision de profondeur 2, 16 de profondeur 4... Le nombre total de nœuds de décision est donc égal à la somme des termes d'une suite géométrique de raison 4 : $n = \sum_{i=0}^{p-1} 4^i = \frac{4^p-1}{4-1}$. Exprimons maintenant le nombre de stratégies en fonction de la profondeur. Pour cela, on procède par induction arrière dans l'arbre, en remontant le nombre de stratégies jusqu'à la racine. On commence par étiqueter à 2 les nœuds de décision qui ne possèdent aucun nœud de décision dans leur descendance. On applique ensuite les relations de récurrence suivantes : le nombre de stratégies à partir d'un nœud de hasard donné est égal au produit du nombre de stratégies à partir de ses successeurs, et le nombre de stratégies à partir d'un nœud de décision donné est égal à la somme du nombre de stratégies à partir de ses successeurs. Ainsi, le nombre total de stratégies à partir d'un nœud de décision N_D peut se calculer à l'aide de la suite récurrente (v_k) suivante : $v_0 = 2$, $v_k = 2v_{k-1}^2$, où k indique le nombre de nœuds de décision (N_D exclu) sur un chemin quelconque de N_D vers un nœud terminal. Le terme général de cette suite est $2^{(2^{k+1}-1)}$. On peut vérifier facilement qu'on a $k = p - 1$ à la racine. Par conséquent, le nombre total de stratégies est $v_{p-1} = 2^{(2^p-1)} \in \Theta(2^{\sqrt{n}})$ (puisque $n = (4^p - 1)/3$). Ainsi, le nombre de stratégies potentielles étant exponentiel en la taille de l'instance, il est nécessaire de développer un algorithme d'optimisation combinatoire pour déterminer la stratégie optimale.

Incohérence dynamique. L'espace des stratégies étant combinatoire comme nous l'avons vu dans la partie précédente, il n'est alors pas envisageable d'énumérer toutes les stratégies d'un arbre de décision hasard pour ensuite les évaluer et sélectionner la meilleure. Nous illustrons ici, à l'aide d'un exemple dû à Jaffray [2], que l'algorithme classique dans les arbres de décision hasard pour déterminer une stratégie optimale au sens d'un critère donné n'est pas valide (pour $\alpha \in]0;1[$). Posons $\alpha = 0.5$ et plaçons nous dans l'arbre de décision hasard de la figure 2. En D_2 le décideur préfère la décision Haut à la décision Bas (pour Haut l'ensemble des conséquences potentielles est $\{10, 20\}$ ce qui donne une évaluation de 15 contre 12.5). Le sous-arbre issu de la décision Bas en D_2 est donc supprimé. En D_1 le décideur préfère la décision Haut qui offre une évaluation de 10 contre 0. La stratégie renvoyée par programmation dynamique est donc $\{D_1 = \text{Haut}, D_2 = \text{Haut}\}$. Cette stratégie est sous-optimale puisque comme nous pouvons le voir dans le tableau de la figure 2, la stratégie optimisant le critère d'Hurwicz pour $\alpha = 0.5$ est $\{D_1 = \text{Haut}, D_2 = \text{Bas}\}$. L'approche classique par programmation dynamique est donc caduque puisqu'une *sous-stratégie* (une sous-stratégie est une stratégie dans le sous arbre induit par un noeud de l'arbre) d'une stratégie optimale n'est pas nécessairement optimale. Cela résulte d'une propriété intrinsèque au critère d'Hurwicz qui permet de décrire des comportements *non-conséquentialistes*, c'est à dire qu'un décideur ayant des préférences qui suivent le critère d'Hurwicz ne prends pas seulement en compte les scénarios futurs mais également les scénarios contrefactuels et passés. Il est bien connu que pour faire de la programmation dynamique il est nécessaire que le critère de décision soit conséquentialiste. Il est donc préférable d'adopter une approche de choix résolu [3] qui consiste à fixer une stratégie depuis la racine et de ne plus en dévier par la suite.

5. Algorithme de résolution fondée sur la programmation dynamique biobjectif

Dans cette partie nous nous intéressons à déterminer une stratégie optimale pour un décideur ayant des préférences qui suivent le critère d'Hurwicz. Pour toute stratégie s (ou sous-stratégie) menant à l'ensemble de conséquences $C_s = \{u_1, u_2, ..., u_n\} \subset \mathcal{C}$ telles que $u_1 \leq u_2 \leq ... \leq u_n$, on définit le vecteur $\langle \underline{u}; \overline{u} \rangle \in \mathcal{C}^2$ de telle sorte que $\underline{u} = u_1$ et $\overline{u} = u_n$. L'évaluation d'une stratégie par le critère d'Hurwicz peut alors se faire en évaluant le vecteur qui caractérise cette stratégie.

Le critère d'Hurwicz est bien entendu compatible avec la dominance de Pareto[1] au sens large (notée $\succsim$ dans la suite) sur l'ensemble des vecteurs de $\mathcal{C}^2$. En effet il est bien évident que pour tout $(V_1, V_2) \in \mathcal{C}^2 \times \mathcal{C}^2$:

$$V_1 \succsim V_2 \Leftrightarrow \begin{cases} \underline{u_1} \geq \underline{u_2} \\ \overline{u_1} \geq \overline{u_2} \end{cases} \Rightarrow \forall \alpha \in [0;1], \alpha\underline{u_1} + (1-\alpha)\overline{u_1} \geq \alpha\underline{u_2} + (1-\alpha)\overline{u_2}$$

où $V_1 = \langle \underline{u_1}; \overline{u_1} \rangle$ et $V_2 = \langle \underline{u_2}; \overline{u_2} \rangle$. L'idée ici est alors de construire l'ensemble Ω des vecteurs (qui caractérisent les stratégies) non dominés au sens de Pareto (si deux vecteurs sont égaux on n'en garde qu'un seul dans Ω) puis d'évaluer, à l'aide du critère d'Hurwicz, chaque vecteur de Ω et de retenir celui d'évaluation maximale. Notons que dans la pratique, on conserve la stratégie correspondante à chacun des vecteurs non-dominés de Ω, mais par soucis de simplicité, nous ne travaillerons que sur les vecteurs induits. Nous allons maintenant voir comment construire l'ensemble Ω de manière efficace. Nous proposons ici de construire l'ensemble Ω par programmation dynamique. En effet, $\succsim$ vérifie le principe de monotonie [4] (condition suffisante pour faire de la programmation dynamique) :

Proposition 1 *Pour tout vecteur* $V_1 = \langle \underline{u_1}; \overline{u_1} \rangle$, $V_2 = \langle \underline{u_2}; \overline{u_2} \rangle$ *et* $V_3 = \langle \underline{u_3}; \overline{u_3} \rangle$ *on a :*

$$V_1 \succsim V_2 \Rightarrow V_1 \oplus V_3 \succsim V_2 \oplus V_3$$

où $\oplus$ *est l'opérateur définit par* $V_1 \oplus V_2 = \langle \min\{\underline{u_1}; \underline{u_2}\}; \max\{\overline{u_1}; \overline{u_2}\} \rangle$. *En effet, on a :*

$$V_1 \succsim V_2 \Leftrightarrow \begin{cases} \underline{u_1} \geq \underline{u_2} \\ \overline{u_1} \geq \overline{u_2} \end{cases} \Rightarrow \begin{cases} \min\{\underline{u_1}; \underline{u_3}\} \geq \min\{\underline{u_2}; \underline{u_3}\} \\ \max\{\overline{u_1}; \overline{u_3}\} \geq \max\{\overline{u_2}; \overline{u_3}\} \end{cases} \Leftrightarrow V_1 \oplus V_3 \succsim V_2 \oplus V_3$$

Comme $\succsim$ respecte le principe de monotonie, il est possible de construire Ω par programmation dynamique. En chaque nœud $N \in \mathcal{N}$ on calcule un ensemble Ω_N de vecteur non dominé au sens de Pareto sur l'ensemble des sous-stratégies induit par N (i.e. l'ensemble des stratégies dans le sous-arbre induit par le nœud N). L'ensemble Ω que l'on cherche à construire est donc l'ensemble associé à la racine de l'arbre de décision hasard. Chacun de ces ensembles est construits par induction arrière dans l'arbre de décision hasard. En d'autres termes, on construit l'ensemble Ω_N à partir des ensembles construits en chaque fils de N. On note $\succsim_{LEX}$ la relation d'ordre lexicographique entre vecteur de $\mathcal{C}^2$ telle que $\langle \underline{u_1}; \overline{u_1} \rangle \succsim_{LEX} \langle \underline{u_2}; \overline{u_2} \rangle$ si et seulement si $((\underline{u_1} > \underline{u_2})$ ou $((\underline{u_1} = \underline{u_2})$ et $(\overline{u_1} \geq \overline{u_2})))$. Remarquons qu'une famille de n vecteurs $\langle a_i; b_i \rangle$ non dominés classés par ordre croissant lexicographique vérifie $a_i < a_{i+1}$ et $b_i > b_{i+1}$ (*propriété lexicographique*). Tous les ensembles Ω_N que nous manipulerons par la suite seront supposé ordonnés lexicographiquement de manière croissante et vérifieront donc cette propriété. On notera $\min_{LEX} \Omega_N$ le plus petit élément au sens lexicographique de l'ensemble Ω_N.

On initialise l'induction arrière en construisant les ensembles Ω_N en chaque conséquence : pour tout noeud de conséquence N muni de la conséquence $c \in \mathcal{C}$, on fixe $\Omega_N = \{\langle c; c \rangle\}$. Les ensembles construits sont bien des ensembles de vecteurs non dominés et classés lexicographiquement puisqu'ils ne possèdent qu'un seul élément. Dans la suite nous allons voir comment on construit, lors de la programmation dynamique, les ensembles Ω_N lorsque N est un noeud de décision ou un noeud de hasard.

Construction de Ω_N si N est un nœud de décision. Si $N \in \mathcal{N}_D$, alors Ω_N est l'ensemble des vecteurs non dominés de l'ensemble $\Gamma_N^D = \bigcup_{N' \in fils(N)} \Omega_{N'}$ où $fils(N)$ est l'ensemble des fils de N. Afin de construire efficacement Ω_N (l'approche naïve étant en $O(n^2)$) il est nécessaire de rappeller une propriété clef utile qui permettra de construire cet ensemble en $O(n)$.

Proposition 2 *Soit* N_1 *et* N_2 *deux ensembles de vecteurs dans* $\mathcal{C}^2$, *on pose* $\langle \underline{u_1}; \overline{u_1} \rangle = \min_{LEX} \Omega_{N_1}$ *et* $\langle \underline{u_2}; \overline{u_2} \rangle = \min_{LEX} \Omega_{N_2}$ *et on suppose que* $\langle \underline{u_1}; \overline{u_1} \rangle \succsim_{LEX} \langle \underline{u_2}; \overline{u_2} \rangle$. *Sous de telles conditions on vérifie la propriété suivante : si* $\langle \underline{u_2}; \overline{u_2} \rangle$ *n'est pas dominé par* $\langle \underline{u_1}; \overline{u_1} \rangle$ *alors aucun vecteur de* N_1 *ne domine* $\langle \underline{u_2}; \overline{u_2} \rangle$

[1] Pour rappel, $V_1 = \langle \underline{u_1}; \overline{u_1} \rangle$ domine au sens large de Pareto le vecteur $V_2 = \langle \underline{u_2}; \overline{u_2} \rangle$ si et seulement si $\underline{u_1} \geq \underline{u_2}$ et $\overline{u_1} \geq \overline{u_2}$.

Preuve. On a $\underline{u_2} \geq \underline{u_1}$ ou $\overline{u_2} \geq \overline{u_1}$. De plus, grâce à la propriété lexicographique, on sait que pour tout $\langle \underline{u}; \overline{u} \rangle \in N_1 \backslash \{\langle \underline{u_1}; \overline{u_1} \rangle\}$ on a $\underline{u_1} < \underline{u}$ et $\overline{u_1} > \overline{u}$. On déduit que $\underline{u_2} > \underline{u}$ ou $\overline{u_2} > \overline{u}$. Le vecteur $\langle \underline{u_2}; \overline{u_2} \rangle$ n'est donc dominé par aucun vecteur de N_1.
On peut étendre facilement la démonstration au cas où il y a plus de deux ensembles. ■

Cette proposition nous permet d'introduire le lemme suivant :

Lemme 1 *Pour tout noeud de décision* N*, on peut construire en temps linéaire l'ensemble* Ω_N *à partir des ensembles* $\Omega_{N'}$, $N' \in fils(N)$.

Preuve. Pour chaque ensemble $\Omega_{N'}$ on récupère le plus petit élément lexicographique. Plus formellement on construit l'ensemble $\Omega_{min} = \bigcup_{N' \in fils(N)} \min_{LEX} \Omega_{N'}$. Parmi ces éléments récupérés, on note $V_{min} \in \min_{LEX} \Omega_{min}$ et $M \in fils(N)$ tel que $V_{min} \in \Omega_M$. On compare V_{min} avec tous les vecteurs de l'ensemble Ω_{min} privé de V_{min}. Si V_{min} est non dominé au sens de Pareto alors on peut l'ajouter à l'ensemble Ω_N, en effet il est le plus petit élément non dominé de l'ensemble Γ_N^D (voir *Proposition 2*). Il suffit ensuite de répéter l'opération jusqu'à que tous les ensembles $\Omega_{N'}$ soient vides en prenant soin de supprimer V_{min} de l'ensemble Ω_M avant de réitérer les opérations. Cette méthode permet de construire lexicographiquement Ω_N en $O(n)$ si n est la taille de l'arbre de décision hasard. ■

Nous illustrons la construction de l'ensemble Ω_N lorsque $N \in \mathcal{N}_D$ au travers de l'exemple suivant.

Exemple 1 *Soit* $\Omega_{N_1} = \{\langle 1;7 \rangle; \langle 2;4 \rangle\}$, $\Omega_{N_2} = \{\langle 2;6 \rangle; \langle 3;5 \rangle\}$ *et* $\Omega_{N_3} = \{\langle 1;6 \rangle; \langle 4;4 \rangle\}$ *avec* $N_1, N_2, N_3 \in fils(N)$. *Nous décrivons ci-dessous la suite des comparaisons effectuées pour construire l'ensemble* Ω_N *lorsque* N *est un noeud de décision.*
- $V_{min} = \langle 1;6 \rangle$ *est dominé dans l'ensemble* $\Omega_{min} = \{\langle 1;7 \rangle; \langle 2;6 \rangle; \langle 1;6 \rangle\}$, *on le supprime de* Ω_{N_3}.
- $V_{min} = \langle 1;7 \rangle$ *n'est pas dominé dans l'ensemble* $\Omega_{min} = \{\langle 1;7 \rangle; \langle 2;6 \rangle; \langle 4;4 \rangle\}$, *on le supprime de* Ω_{N_1} *et on l'ajoute à* Ω_N.
- $V_{min} = \langle 2;4 \rangle$ *est dominé dans l'ensemble* $\Omega_{min} = \{\langle 2;4 \rangle; \langle 2;6 \rangle; \langle 4;4 \rangle\}$, *on le supprime de* Ω_{N_1}.
- $V_{min} = \langle 2;6 \rangle$ *n'est pas dominé dans l'ensemble* $\Omega_{min} = \{\langle 2;6 \rangle; \langle 4;4 \rangle\}$, *on le supprime de* Ω_{N_2} *et on l'ajoute à* Ω_N.
- $V_{min} = \langle 3;5 \rangle$ *n'est pas dominé dans l'ensemble* $\Omega_{min} = \{\langle 3;5 \rangle; \langle 4;4 \rangle\}$, *on le supprime de* Ω_{N_2} *et on l'ajoute à* Ω_N.
- $V_{min} = \langle 4;4 \rangle$ *n'est pas dominé dans l'ensemble* $\Omega_{min} = \{\langle 4;4 \rangle\}$, *on le supprime de* Ω_{N_3} *et on l'ajoute à* Ω_N.
- Ω_{N_1}, Ω_{N_2} *et* Ω_{N_3} *sont vides on s'arrête.*

On obtient alors l'ensemble $\Omega_N = \{\langle 1;7 \rangle; \langle 2;6 \rangle; \langle 3;5 \rangle; \langle 4;4 \rangle\}$.

Construction de Ω_N si N est un nœud de hasard. Si $N \in \mathcal{N}_C$, alors Ω_N est l'ensemble des vecteurs non-dominés de l'ensemble $\Gamma_N^H = \{V_1 \oplus V_2 \oplus ... \oplus V_k : (V_1, V_2, ..., V_k) \in \Omega_{N_1} \times \Omega_{N_2} \times ... \times \Omega_{N_k}\}$ tels que $N_1, N_2, ..., N_k$ sont les fils de N. Afin de construire efficacement Ω_N (l'approche naïve étant en $O(n^4)$) il est nécessaire d'introduire une propriété clef qui permettra de construire cet ensemble en $O(n)$.

Proposition 3 *Soit* N_1 *et* N_2 *deux ensembles de vecteurs , on pose* $\langle \underline{u_1}; \overline{u_1} \rangle = \min_{LEX} \Omega_{N_1}$ *et* $\langle \underline{u_2}; \overline{u_2} \rangle = \min_{LEX} \Omega_{N_2}$ *et on suppose que* $\langle \underline{u_1}; \overline{u_1} \rangle \succsim_{LEX} \langle \underline{u_2}; \overline{u_2} \rangle$. *Sous de telles conditions on vérifie la propriété suivante :*

$$\langle \underline{u_2}; \overline{u_2} \rangle \oplus \langle \underline{u_1}; \overline{u_1} \rangle \succsim \langle \underline{u_2}; \overline{u_2} \rangle \oplus \langle \underline{u}; \overline{u} \rangle \text{ pour tout vecteur } \langle \underline{u}; \overline{u} \rangle \in N_1 \backslash \{\langle \underline{u_1}; \overline{u_1} \rangle\}$$

Preuve. On a $\underline{u_2} < \underline{u_1}$ ou ($\underline{u_2} = \underline{u_1}$ et $\overline{u_2} \leq \overline{u_1}$). De plus, grâce à la propriété lexicographique, on sait que pour tout $\langle \underline{u}; \overline{u} \rangle \in N_1 \backslash \{\langle \underline{u_1}; \overline{u_1} \rangle\}$, $\underline{u_1} < \underline{u}$ et $\overline{u_1} > \overline{u}$. On déduit que $\max\{\overline{u_1}; \overline{u_2}\} \geq \max\{\overline{u}; \overline{u_2}\}$ et $\min\{\underline{u_1}; \underline{u_2}\} = \min\{\underline{u}; \underline{u_2}\} = \underline{u_2}$ autrement dit $\langle \underline{u_2}; \overline{u_2} \rangle \oplus \langle \underline{u_1}; \overline{u_1} \rangle \succsim \langle \underline{u_2}; \overline{u_2} \rangle \oplus \langle \underline{u}; \overline{u} \rangle$.
On peut étendre facilement la démonstration au cas où il y a plus de deux ensembles. ■

Cette proposition nous permet d'introduire le lemme suivant :

Lemme 2 *Pour tout noeud de hasard* N*, on peut construire en temps linéaire l'ensemble* Ω_N *à partir des ensembles* $\Omega_{N'}$, $N' \in fils(N)$.

Preuve. Pour chaque ensemble $\Omega_{N'}$ on récupère le plus petit élément lexicographique. Plus formellement on construit l'ensemble $\Omega_{min} = \bigcup_{N' \in fils(N)} \min_{LEX} \Omega_{N'}$. Parmi ces éléments récupérés, on note $V_{min} \in \min_{LEX} \Omega_{min}$ et $M \in fils(N)$ tel que $V_{min} \in \Omega_M$. On construit le vecteur V en appliquant l'opérateur $\oplus$ sur l'ensemble des plus petits vecteurs de chaque ensemble (i.e. $V = V_1 \oplus V_2 \oplus ... \oplus V_k$ tels que $\bigcup_{i=1}^{i=k} V_i = \Omega_{min}$) et on ajoute V à Ω_N. On supprime V_{min} de l'ensemble Ω_M car on sait que toutes agrégation avec l'opérateur $\oplus$ de l'élément V_{min} fournira alors un vecteur dominé par V (voir *Proposition 3*). Enfin on repète ce processus jusqu'a qu'au moins un des ensembles Ω_N soit vide. Pour finir il suffit de supprimer tous les éléments dominés de Ω_N (lorsque les éléments sont triés dans l'ordre lexicographique, ce qui est le cas ici, cette étape se fait linéairement). Cette méthode permet de construire lexicographiquement Ω_N en $O(n)$ si n est la taille de l'arbre de décision hasard. ■

Nous illustrons la construction de l'ensemble Ω_N lorsque $N \in \mathcal{C}$ au travers de l'exemple suivant.

Exemple 2 *Soit* $\Omega_{N_1} = \{\langle 1;7\rangle; \langle 2;4\rangle\}$, $\Omega_{N_2} = \{\langle 2;6\rangle; \langle 3;5\rangle\}$ *et* $\Omega_{N_3} = \{\langle 1;6\rangle; \langle 4;4\rangle\}$ *avec* $N_1, N_2, N_3 \in fils(N)$. *Nous décrivons ci-dessous la suite des opérations effectuées pour construire l'ensemble* Ω_N *lorsque* N *est un noeud de hasard.*
– $V = \langle 1;7\rangle \oplus \langle 2;6\rangle \oplus \langle 1;6\rangle = \langle 1;7\rangle$ *; on l'ajoute à* Ω_N *et on supprime* $V_{min} = \langle 1;6\rangle$ *de* Ω_{N_3}.
– $V = \langle 1;7\rangle \oplus \langle 2;6\rangle \oplus \langle 4;4\rangle = \langle 1;7\rangle$ *; on l'ajoute à* Ω_N *et on supprime* $V_{min} = \langle 1;7\rangle$ *de* Ω_{N_1}.
– $V = \langle 2;4\rangle \oplus \langle 2;6\rangle \oplus \langle 4;4\rangle = \langle 2;6\rangle$ *; on l'ajoute à* Ω_N *et on supprime* $V_{min} = \langle 2;4\rangle$ *de* Ω_{N_1}.
– Ω_{N_1} *est vide on s'arrête.*
On obtient alors l'ensemble $\Omega_N = \{\langle 1;7\rangle; \langle 1;7\rangle; \langle 2;6\rangle\}$. *Après suppression des éléments dominés on déduit* $\Omega_N = \{\langle 1;7\rangle; \langle 2;6\rangle\}$.

Une fois la programmation dynamique achevée, il suffit alors d'évaluer tous les vecteurs de l'ensemble construit en la racine avec le critère d'Hurwicz. En guise d'illustration nous appliquons l'algorithme sur l'arbre de la figure 2. Les ensembles construits sont successivement : $\Omega_{L_2} = \{\langle 10;20\rangle\}$, $\Omega_{L_3} = \{\langle 0;25\rangle\}$, $\Omega_{D_2} = \{\langle 10;20\rangle; \langle 0;25\rangle\}$, $\Omega_{L_1} = \{\langle 0;25\rangle\}$ et $\Omega_{D_1} = \{\langle 0;25\rangle\}$. La stratégie optimale au sens d'Hurwicz pour tout α est donc la stratégie caractérisée par le vecteur $\langle 0;25\rangle$, ce qui correspond à la stratégie $\{D_1 = Haut; D_2 = Bas\}$.
Pour achever cette étude algorithmique nous sommes maintenant en mesure d'énoncer le résultat de complexité suivant :

Théorème 1 *Le calcul d'une stratégie optimisant le critère d'Hurwicz peut se réaliser via un algorithme en* $O(n^2)$.

Preuve. Pour prouver ce théorème, remarquons que $|\Omega| \leq |\mathcal{C}|$. En effet, pour chaque valeur de la première composante vectorielle il ne peut y avoir qu'un seul non dominé au plus. Ainsi le nombre de conséquences d'un arbre de décision hasard étant en $O(n)$, le nombre maximum de vecteurs non dominés est alors en $O(n)$ également. Un ensemble de vecteurs de non dominés est construit en chaque noeud de l'arbre avec des méthodes en $O(n)$ (d'après le *lemme 1* et le *lemme 2*). Ce qui fournit bien une compléxité de $O(n^2)$ pour l'algorithme proposé puisque le nombre de noeuds d'une arbre de décision hasard est en $O(n)$. ■

Dans la section suivante nous illustrons les performances de l'algorithme proposé au travers d'expérimentation numériques.

6. Expérimentations numériques

Nous avons testé sur des instances générées aléatoirement l'algorithme proposé dans la section précédente. Les arbres générés sont des arbres binaires et complet (i.e. chaque niveau de l'arbre est complet) en alternant les noeuds de décision et de hasard sur chaque branche de l'arbre. Les utilités sont des réels qui ont été générés aléatoirement sur chaque conséquence. L'algorithme a été implémenté en C++ et les expériences ont été lancées sur un ordinateur équipé d'un processeur

Prof. (Noeuds)	5 (63)		7 (255)		9 (1023)		11 (4095)		13 (16383)		15 (65535)		17 (262143)		19 (1048575)		21 (4194303)		23 (16777215)	
	card.	tps.	card.	tps.	card.	tps.	card.	tps.	card.	tps.	card.	tps.	card.	tps.	card.	tps.	card.	tps.	card.	tps.
Moy	2	0	4	0	7	0	13	0	24	0	47	0.02	90	0.14	174	0.58	348	2.09	714	8.31
Max	5	0	6	0	14	0	22	0	39	0	68	1	115	1	216	1	570	3	1164	9

TAB. 1 – Résultats numériques.

Pentium IV à 1.3 GHz avec 3.5 Go de RAM. Le tableau 1 donne pour chaque profondeur d'arbre (500 instances ont été généré pour chaque profondeur) le cardinal maximal moyen et maximal des ensembles construit en chaque noeud de l'arbre. Nous donnons également le temps moyen en secondes ainsi que le temps maximum constaté d'éxécution de l'algorithme. Comme le traduit le tableau 1 nous sommes capable de traiter des instances de profondeur 23 (plus de 16 000 000 de noeuds). Si nous ne sommes pas allés plus loin c'est essentiellement dû au fait que la mémoire requise pour stocker les arbres de décision hasard de plus grande profondeur devient trop grande.

7. Conclusion

Après avoir exposé brièvement les problèmes algorithmiques induit par le calcul de la stratégie optimale au sens du critère d'Hurwicz nous avons proposé un algorithme de complexité polynomiale pour le résoudre. Les expérimentations numériques réalisées fournissent des temps d'exécution compétitifs. Cependant, la taille des arbres de décision hasard étant rapidement prohibitive, il est envisageable de se tourner vers d'autres modèles graphiques (e.g., les *diagrammes d'influence* [1]) afin de pouvoir traiter des instances plus grandes. Il serait également intéressant d'étudier le problème de la détermination d'une stratégie optimisant le critère d'Hurwicz dans le cadre des probabilités imprécises [2].

8. Remerciements

Je tiens à remercier Olivier Spanjaard pour avoir porté mon attention sur le sujet étudié ici et pour ses nombreux conseils ainsi que Patrice Perny pour ses remarques toujours pertinentes.

Bibliographie

1. R. Howard et J. Matheson. Influence diagrams. *Readings on the Principles and Applications of Decision Analysis*, pages 721–762, 1984.
2. J.-Y. Jaffray et M. Jeleva. Information processing under imprecise risk with the hurwicz criterion. In *5th international symposium on imprecise probability : theories and applications*, pages 233–242, 2007.
3. E.F. McClennen. *Rationality and Dynamic choice : Foundational Explorations*. Cambridge University Press, 1990.
4. T.L. Morin. Monotonicity and the principle of optimality. In *Journal of Mathematical Analysis and Applications*, volume 86, pages 665–674, 1982.
5. J. Quiggin. *Generalized Expected Utility Theory : The Rank-Dependent Model*. Kluwer, 1993.
6. H. Raiffa. *Decision Analysis : Introductory Lectures on Choices under Uncertainty*. Addison-Wesley, 1968.
7. L. Savage. The theory of statistical decision. In *Journal of the American Statistical Association*, volume 46, pages 55–67, 1951.
8. J. von Neuman et O. Morgenstern. *Theory of games and economic behaviour*. Princeton University Press, 1947.