

Approche algorithmique de la recherche d'une stratégie RDU-optimale dans un arbre de décision

Gildas Jeantet, Olivier Spanjaard

▶ **To cite this version:**

Gildas Jeantet, Olivier Spanjaard. Approche algorithmique de la recherche d'une stratégie RDU-optimale dans un arbre de décision. 9ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2008), Feb 2008, Clermont-Ferrand, France. pp.79-94. hal-01303913

HAL Id: hal-01303913
https://hal.science/hal-01303913

Submitted on 30 Jun 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Approche algorithmique de la recherche d'une stratégie RDU-optimale dans un arbre de décision

G. Jeantet et O. Spanjaard

LIP6, 4 place Jussieu, 75252 Paris cedex 05
{gildas.jeantet,olivier.spanjaard}@lip6.fr

Résumé Le problème de la recherche d'une stratégie EU-optimale (i.e., optimale au sens de l'utilité espérée) dans un arbre décision hasard se résout en temps linéaire en fonction du nombre d'arcs par programmation dynamique [11]. Nous nous intéressons ici à une variante plus difficile de ce problème, où l'on recherche une stratégie RDU-optimale (i.e., optimale au sens de l'utilité dépendant du rang). L'utilité dépendant du rang [10] présente une plus grande richesse descriptive que l'utilité espérée car elle permet un traitement non linéaire des probabilités. Le problème algorithmique qui s'ensuit dans les arbres décision hasard est cependant plus difficile car la programmation dynamique ne s'applique plus. Nous établissons ici que le problème est NP-difficile. Nous proposons un algorithme de séparation et évaluation pour le résoudre, et présentons des résultats numériques montrant l'efficacité de notre approche.

Mots-Clefs. Théorie de la décision ; Algorithmique ; Utilité dépendant du rang ; Arbres décision hasard ; Complexité ; Séparation et évaluation.

1 Introduction

Il est des situations de choix où les conséquences des actions potentielles ne peuvent être déterminées avec certitude. Lorsque cette incertitude est probabilisée (autrement dit lorsque la probabilité d'occurrence de chacune des conséquences est connue), on parle de *décision dans le risque*. Une action potentielle peut alors être vue comme une distribution de probabilité sur l'ensemble des conséquences. L'objet de la théorie de la décision dans le risque est entre autres d'étudier et de modéliser le comportement d'un décideur en situation de choix entre de telles actions potentielles (pour une introduction au domaine, voir par exemple l'ouvrage de Gayant [4]). Les premiers travaux dans cette optique remontent au modèle de l'espérance d'utilité (EU) proposé par von Neumann et Morgenstern [12]. Dans ce modèle, les individus assignent une valeur numérique, nommée utilité, à chaque conséquence. L'évaluation d'une distribution de probabilité se fait ensuite via un calcul d'espérance d'utilité. Une distribution de probabilité est alors préférée à une autre si son espérance d'utilité est plus grande.

En pratique, l'ensemble des actions potentielles est souvent défini en compréhension. C'est le cas en particulier dans les problèmes de décision séquentielle, où l'on doit prendre une séquence de décisions conditionnellement à des événements. On parle alors de *stratégie*. Un moyen commode de représenter de façon

compacte un problème de décision séquentielle dans le risque est d'utiliser un *arbre décision hasard*. Il s'agit d'une arborescence comportant trois types de nœuds : les *nœuds de décision* (représentés par des carrés), les *nœuds de hasard* (représentés par des cercles), et les *nœuds terminaux* (les feuilles de l'arborescence). Les branches issues d'un nœud de décision correspondent à différentes décisions possibles, tandis que celles issues d'un nœud de hasard correspondent aux différents événements possibles, dont on connait les probabilités. Enfin, les valeurs figurant au niveau des feuilles de l'arborescence correspondent aux utilités des différentes conséquences. Remarquons que l'usage veut qu'on omette les orientations des arcs lorsqu'on représente les arbres décision hasard. Nous illustrons maintenant l'usage de cet outil sur un exemple de choix de contrat d'assurance pour un bien immobilier [3].

Exemple 1 *Considérons un contrat d'assurance auquel on peut décider de souscrire pour une année* 1 *et/ou une année* 2. *Bien évidemment, on ne sait pas si l'on sera amené ou non à le faire valoir à la suite d'un dommage (cambriolage, inondation...). On suppose ici que la prise en charge d'un dommage non couvert coûte 2. Par ailleurs, la probabilité de subir un dommage durant l'année* 2 *est conditionnée aux événements de l'année* 1 *: elle est de* $\frac{3}{5}$ *l'année 1, mais elle passe à* $\frac{1}{4}$ *l'année 2 si on a déjà subi un dommage lors de l'année 1 (sinon elle reste inchangée). La première année, on peut souscrire (décision 1) ou non (décision 0) un contrat d'assurance couvrant les dommages dont le coût est de 1. La deuxième année, on peut ou non renouveller ce contrat pour le même coût. L'utilité du décideur est calculée en fonction du coût total* $x = 2s + a$, *où* s *est le nombre de dommages non couverts et* a *est le nombre de souscriptions au contrat d'assurance. On pose ici que la fonction d'utilité* u *est* $u(x) = 4 - x$. *L'arbre décision hasard correspondant à ce problème est représenté sur la figure 1.*

Il est important de remarquer que dans un arbre décision hasard binaire complet (i.e., comportant deux décisions (resp. événements) possibles à chaque nœud de décision (resp. hasard) et dont chaque niveau est complètement rempli), le nombre de stratégies possibles est exponentiel dans le nombre de nœuds de décision (rappelons qu'une stratégie est caractérisée par la donnée des choix effectués aux différents nœuds de décision). Plus précisément, si on note n le nombre de nœuds de décision, on peut montrer que le nombre de stratégies possibles est en $\Theta(2^{\sqrt{n}})$ (section 3). Il est cependant bien connu qu'il existe un algorithme linéaire (i.e., en $O(n)$) permettant de déterminer, par programmation dynamique, une stratégie optimale au sens du modèle EU. En effet, une telle stratégie vérifie le principe d'optimalité : toute sous-stratégie d'une stratégie optimale est optimale. L'idée de l'algorithme consiste donc à procéder par induction arrière à partir des nœuds terminaux, afin de déterminer en chaque nœud l'espérance d'utilité d'une sous-stratégie optimale :

- en un nœud de hasard, l'espérance d'utilité optimale est égale à l'espérance des utilités optimales de ses successeurs ;
- en un nœud de décision, l'espérance d'utilité optimale est égale à la plus grande des espérances d'utilité optimales de ses successeurs.

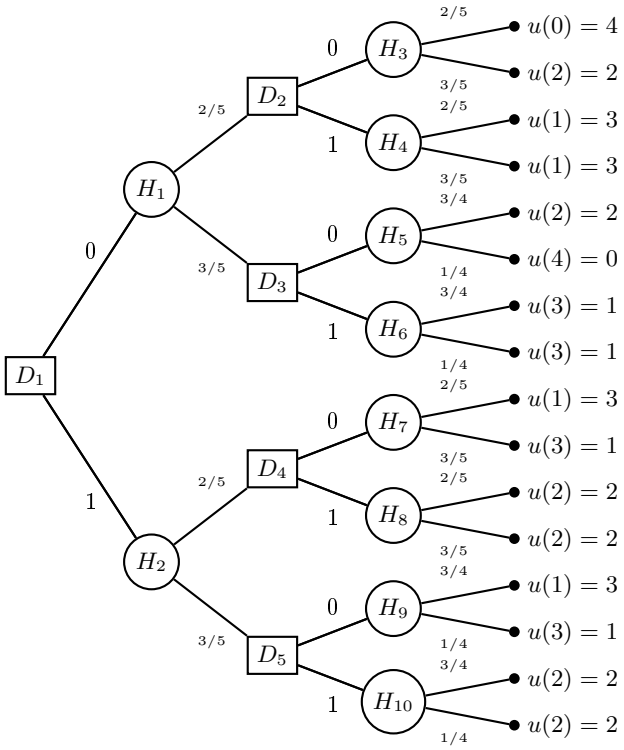


Fig. 1. Exemple d'arbre décision hasard.

Exemple 2 *Dans l'exemple précédent du contrat d'assurance, l'algorithme remonte la valeur* $max(\frac{14}{5}, 3) = 3$ *en* D_2, $\frac{3}{2}$ *en* D_3, 2 *en* D_4 *(en prenant la décision 1) et* $\frac{10}{4}$ *en* D_5 *(en prenant la décision 0). Par conséquent, l'espérance en* H_1 *vaut* $3 \times \frac{2}{5} + \frac{3}{2} \times \frac{3}{5} = \frac{21}{10}$ *et en* H_2 *elle vaut* $\frac{23}{10}$*. La stratégie optimale au sens de EU consiste donc à souscrire au contrat la première année et à ne souscrire au contrat la seconde année que si l'on n'a pas subi de dommage lors de la première.*

La simplicité d'utilisation du modèle EU, ainsi que son attrait sur le plan normatif, lui ont permis de régner sans partage ces soixante dernières années. Pourtant, les mises en défaut répétées du modèle sur le plan descriptif ont fini par éroder sa position. En particulier, de nombreuses expériences mettent en évidence que les individus sous-évaluent les fortes probabilités et surévaluent les faibles probabilités [1,7]. De ce fait, bien souvent, le modèle EU n'est pas à même de rendre compte du comportement décisionnel observé. Face à ce constat, de nouveaux modèles ont été développés : certains se fondent sur une représentation alternative de l'incertitude comme l'offre par exemple la théorie des possibilités [2], d'autres prennent en compte explicitement la perception déformée des probabilités par le décideur. Dans cette dernière démarche, Quiggin [10] a proposé le modèle *Rank Dependent Utility* (RDU), qui permet de rendre compte d'un plus large éventail de comportements décisionnels. Cependant, la non-linéarité du critère RDU (i.e., $\mathrm{RDU}(\lambda X + Y) \neq \lambda \mathrm{RDU}(X) + \mathrm{RDU}(Y)$) invalide toute

approche par induction arrière pour déterminer la stratégie optimale. Cela pose un réel problème algorithmique au vu du nombre combinatoire de stratégies possibles, qui rend impraticable leur énumération complète.

Le propos de ce papier est précisément d'étudier le problème consistant à déterminer la stratégie optimale dans un arbre décision hasard au sens de RDU et de proposer une méthode par énumération implicite pour le résoudre. Le papier est organisé comme suit. Dans un premier temps nous formalisons la problématique et rappellons les bases du modèle RDU. Nous mettons ensuite en évidence l'impossibilité de procéder par programmation dynamique pour résoudre ce problème et nous prouvons que ce dernier est NP-difficile (section 3.3). Nous exposons alors un algorithme de résolution par énumération implicite, fondé sur une borne calculable en temps quadratique, et nous terminons enfin en présentant les résultats d'expérimentations numériques (section 4).

2 Formalisation du problème

2.1 Notations et Définitions

Dans un arbre décision hasard $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ ayant pour racine un nœud de décision N_r, nous notons $\mathcal{N}_D \subset \mathcal{N}$ (resp. $\mathcal{N}_H \subset \mathcal{N}$) l'ensemble des nœuds de décision (resp. hasard). De plus, nous notons $\mathcal{C} \subset \mathcal{N}$ l'ensemble des nœuds terminaux. Le graphe est valué comme suit : à tout arc $E = (H, N) \in \mathcal{E}$ tel que $H \in \mathcal{N}_H$, on associe la probabilité $p(E)$ de l'événement correspondant ; à tout nœud terminal $C \in \mathcal{C}$, on associe son utilité notée $u(C)$. Par ailleurs, nous appelons $past(N)$ le *passé* de $N \in \mathcal{N}$, i.e. l'ensemble des arcs le long du chemin allant de N_r à N dans $\mathcal{T}$. Enfin, nous notons $S(N)$ l'ensemble des successeurs de N dans $\mathcal{T}$, et $\mathcal{T}(N)$ le sous-arbre de $\mathcal{T}$ de racine N.

Soit $\mathcal{T}$ un arbre décision hasard et $\mathcal{N}^\Delta \subseteq \mathcal{N}$ un ensemble de nœuds contenant :

- la racine N_r de $\mathcal{T}$,
- un et un seul successeur pour chaque nœud de décision $N \in \mathcal{N}_D^\Delta = \mathcal{N}_D \cap \mathcal{N}^\Delta$,
- tous les successeurs pour chaque nœud de hasard $N \in \mathcal{N}_H^\Delta = \mathcal{N}_H \cap \mathcal{N}^\Delta$.

L'ensemble d'arcs $\Delta = \{(N, N') : N \in \mathcal{N}_D^\Delta, N' \in \mathcal{N}^\Delta\} \subseteq \mathcal{E}$ définit une *stratégie* de $\mathcal{T}$ dès lors que le sous-graphe induit par $\mathcal{N}^\Delta$ est un arbre. Etant donné un nœud de décision N, la restriction d'une stratégie de $\mathcal{T}$ au sous-arbre $\mathcal{T}(N)$, qui n'est autre qu'une stratégie de $\mathcal{T}(N)$, est appellée *sous-stratégie*. Nous notons $\mathcal{D}$ l'ensemble des stratégies.

Soit $S = \{u_1, \ldots, u_k\}$ un ensemble fini d'utilités. On appelle *loterie* une distribution de probabilité P sur S. On note $L = (p_1, u_1; \ldots; p_k, u_k)$ la loterie qui aboutit à une utilité u_i avec une probabilité $p_i = P(\{u_i\})$. Afin d'alléger certaines notations, on peut considérer la loterie L comme une fonction de $S \mapsto [0, 1]$ telle

que $L(u_i) = p_i$. Dans un arbre décision hasard, à toute stratégie il est possible d'associer une loterie. En effet, on peut déterminer la probabilité p_C d'obtenir une conséquence $C \in \mathcal{C}$ en calculant :

$$p_C = \prod_{(H,N) \in past(C)} p((H,N)) \text{ où } H \in \mathcal{N}_H$$

La valeur d'une stratégie selon EU (resp. RDU) est égale à la valeur de la loterie correspondante selon EU (resp. RDU).

Exemple 3 *Dans l'exemple du contrat d'assurance, la stratégie EU-optimale correspond à la loterie* $(\frac{3}{20}, 1; \frac{8}{20}, 2; \frac{9}{20}, 3)$ *dont l'espérance est bien* $\frac{46}{20} = \frac{23}{10}$.

2.2 Rappels sur RDU

Le modèle RDU repose sur deux paramètres : une fonction d'utilité qui est déjà présente dans le modèle EU, et une fonction φ de déformation des probabilités. Il s'agit d'une fonction strictement croissante sur $[0,1]$ telle que $\varphi(0) = 0$ et $\varphi(1) = 1$. Cette déformation des probabilités porte, non sur des probabilités simples, mais sur des cumuls de probabilités. Pour rappel, étant donnée une loterie $L = (p_1, u_1; \ldots; p_k, u_k)$, on appelle *fonction décumulative* de L la fonction $G_L : S \mapsto [0,1]$ qui associe à chaque utilité u_i la probabilité d'avoir au moins cette utilité. Plus formellement, $G_L(x) = \sum_{i:u_i \geq x} p_i$. La valeur selon RDU d'une loterie L est alors définie de la manière suivante :

$$RDU(L) = u_{(1)} + \sum_{i=2}^{k} [u_{(i)} - u_{(i-1)}] \varphi(G_L(u_{(i)}))$$

où (.) correspond à une permutation de $\{1, \ldots, k\}$ telle que $u_{(1)} \leq \ldots \leq u_{(k)}$. Ce critère peut être interprété comme suit : on est sûr d'obtenir au moins une utilité de $u_{(1)}$, puis on est susceptible d'obtenir un supplément d'utilité de $u_{(2)} - u_{(1)}$ avec une masse de probabilité $\varphi(G_L(u_{(2)}))$, puis un supplément d'utilité de $u_{(3)} - u_{(2)}$ avec une masse de probabilité $\varphi(G_L(u_{(3)}))$, et ainsi de suite...

Exemple 4 *Considérons la stratégie EU-optimale de l'exemple 2. La loterie correspondante est* $L = (\frac{3}{20}, 1; \frac{8}{20}, 2; \frac{9}{20}, 3)$. *Sa valeur RDU se calcule comme suit :* $RDU(L) = 1 + \varphi(\frac{17}{20}) \times (2-1) + \varphi(\frac{9}{20}) \times (3-2)$. *Supposons que* $\varphi(p) = 0.25$ *pour* $0 < p \leq 0.5$, *et* $\varphi(p) = 0.75$ *pour* $0.5 < p < 1$. *On obtient alors* $RDU(L) = 1 + 0.75 \times 1 + 0.25 \times 1 = 2$.

L'intérêt de déformer des cumuls de probabilités, et non directement les probabilités elles-mêmes (comme c'est par exemple le cas dans le modèle de Handa [5]), est d'obtenir un critère de choix compatible avec la dominance stochastique. On dit qu'une loterie $L = (p_1, u_1; \ldots; p_k, u_k)$ domine stochastiquement une loterie $L' = (p'_1, u'_1; \ldots; p'_k, u'_k)$ si $\forall x \in \mathbb{R}, G_L(x) \geq G_{L'}(x)$, autrement dit, pour tout $x \in \mathbb{R}$, la probabilité d'obtenir une utilité d'au moins x avec la loterie L est au moins aussi grande qu'avec la loterie L'. La compatibilité avec la dominance stochastique signifie que $RDU(L) \geq RDU(L')$ dès lors que L domine stochastiquement L' [10]. Cette propriété est bien entendu souhaitable pour décrire un comportement rationnel, et elle est bien vérifiée par le modèle RDU (contrairement au modèle de Handa).

3 Etude du problème

3.1 Espace des solutions

Considérons un arbre décision hasard binaire complet $\mathcal{T}$ de profondeur $2p$ tel que les nœuds de profondeur paire soient des nœuds de décision (ou des nœuds terminaux) 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 $\mathcal{T}$. Remarquons qu'il y a 1 nœud de décision pour la profondeur 0, 4 nœuds de décision pour la profondeur 2, 16 pour la profondeur 4... Le nombre total de nœuds de décision dans $\mathcal{T}$ est donc égal à la somme des termes d'une suite géométrique de raison 4 : $n = |\mathcal{N}_D| = \sum_{i=0}^{p-1} 4^i = \frac{4^p-1}{4-1}$. Exprimons maintenant le nombre de stratégies en fonction de la profondeur. Pour cela, on procède par induction arrière sur $\mathcal{T}$, 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 (u_k) suivante : $u_0 = 2$, $u_k = 2u_{k-1}^2$, 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 dans $\mathcal{T}$ est $|\mathcal{D}| = u_{p-1} = 2^{(2^p-1)} \in \Theta(2^{\sqrt{n}})$ (puisque $n = (4^p-1)/3$). Ainsi, le nombre de stratégies potentielles étant exponentiel de la taille de l'instance, il est nécessaire de développer un algorithme d'optimisation combinatoire pour déterminer la stratégie optimale. Nous montrons ci-dessous que la tâche est d'autant plus délicate que la programmation dynamique ne s'applique plus lorsqu'on optimise selon RDU.

3.2 Monotonie et indépendance

Il est bien connu que la programmation dynamique repose sur le respect d'une condition de *monotonie* [9] sur la fonction de valuation. Dans notre contexte, cette condition peut se formuler comme suit sur la fonction de valuation V des loteries :

$$\forall \alpha \in [0,1], \quad V(L) \geq V(L') \Longrightarrow V(\alpha L + (1-\alpha)L'') \geq V(\alpha L' + (1-\alpha)L'')$$

où L, L', L'' sont des loteries quelconques et $\alpha L + (1-\alpha)L''$ est la loterie définie par $(\alpha L + (1-\alpha)L'')(x) = \alpha L(x) + (1-\alpha)L''(x)$. Cette condition *algorithmique* peut être interprétée, dans le cadre de la théorie de la décision, comme une forme affaiblie de l'axiome d'*indépendance* utilisé par von Neumann et Morgenstern [12]

dans la caractérisation du critère EU. Cet axiome stipule que la combinaison de deux loteries L et L' avec une troisième L'' n'inverse pas l'ordre des préférences (induit par V) : si L est strictement préférée à L', alors $\alpha L + (1-\alpha)L''$ est strictement préférée à $\alpha L' + (1-\alpha)L''$. Pour $V \equiv EU$ la propriété de monotonie est vérifiée. Par contre, pour $V \equiv RDU$, la propriété n'est plus valide, comme le montre l'exemple suivant.

Exemple 5 *Soient trois loteries* $L = (0.5, 1; 0.5, 10)$, $L' = (1, 5)$ *et* $L'' = L$. *Supposons que les préférences du décideur suivent le modèle RDU avec la fonction* φ *suivante :* $\varphi(0) = 0$, $\varphi(p) = 0.45$ *si* $0 < p \leq 0.7$, $\varphi(p) = 1$ *si* $p > 0.7$. *Les valeurs selon RDU de* L *et* L' *sont :*

$$RDU(L) = 1 + (10-1)\varphi(0.5) = 5.05$$
$$RDU(L') = 5$$

Ainsi, on a $RDU(L) \geq RDU(L')$. *D'après la propriété de monotonie pour* $\alpha = 0.6$, *on devrait donc avoir* $RDU(0.6L+0.4L'') \geq RDU(0.6L'+0.4L'')$. *Pourtant, on a :*

$$RDU(0.6L + 0.4L'') = 1 + (10-1)\varphi(0.5) = 5.05$$
$$RDU(0.6L' + 0.4L'') = 1 + (5-1)\varphi(0.6+0.2) + (10-5)\varphi(0.2) = 7.25$$

et donc $RDU(0.6L+0.4L'') < RDU(0.6L'+0.4L'')$. *Par conséquent, la propriété de monotonie n'est pas vérifiée.*

De par la violation du principe de monotonie, la mise en œuvre d'une procédure de programmation dynamique pour RDU dans un arbre décision hasard peut conduire à une stratégie sous-optimale. Une telle procédure peut même conduire à une stratégie stochastiquement dominée. En effet, considérons l'arbre de décision de la figure 2, construit à l'aide de l'exemple 5. Dans cet arbre décision hasard, les valeurs RDU des différentes stratégies possibles à la racine sont :

$RDU(\{(D_1, H_2)\}) = 1 + (5-1)\varphi(0.6+0.2) + (8-5)\varphi(0.2) = 6.35$
$RDU(\{(D_1, H_1), (D_2, H_3), (D_3, H_4)\}) = 1 + (10-1)\varphi(0.5) = 5.05$
$RDU(\{(D_1, H_1), (D_2, \delta_1), (D_3, \delta_2)\}) = 5$
$RDU(\{(D_1, H_1), (D_2, \delta_1), (D_3, H_4)\}) = 7.25$
$RDU(\{(D_1, H_1), (D_2, H_3), (D_3, \delta_2)\}) = 5.05$

Ainsi, la stratégie optimale à la racine est $\{(D_1, H_1), (D_2, \delta_1), (D_3, H_4)\}$. Pourtant, en procédant par programmation dynamique, on obtient en D_2 : $RDU(\{(D_2, H_3)\}) = 1 + (10-1)\varphi(0.5) = 5.05$ et $RDU(\{(D_2, \delta_1)\}) = 5$. C'est donc la sous-stratégie $\{(D_2, H_3)\}$ qui est retenue en D_2, et de même la sous-stratégie $\{(D_3, H_4)\}$ qui est retenue en D_3. Par suite, en D_1, c'est la stratégie $\{(D_1, H_2)\}$ (6.35 contre 5.05 pour $\{(D_1, H_2)\}$), dominée stochastiquement par $\{(D_1, H_1), (D_2, \delta_1), (D_3, H_4)\}$, qui est retournée.

Un décideur utilisant le critère RDU doit donc faire du *choix résolu* [8], c'est-à-dire qu'il doit choisir une stratégie à la racine de l'arbre et s'y tenir (faute de quoi il pourrait se retrouver comme ci-dessus à suivre une stratégie stochastiquement dominée). Nous nous intéressons ici à déterminer une stratégie RDU-optimale vue de la racine (puis à ne pas en dévier). Remarquons qu'un tel procédé nous assure de ne pas rencontrer de sous-stratégie stochastiquement dominée, contrairement à la méthode consistant à faire de l'induction arrière

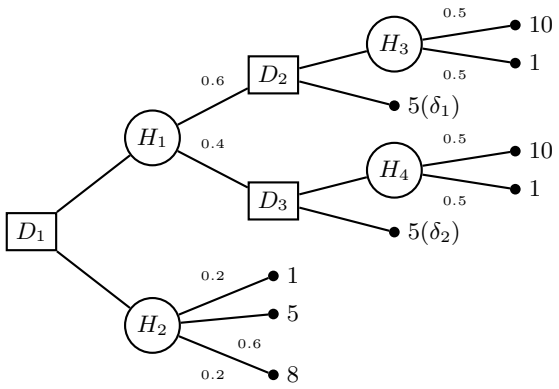


Fig. 2. RDU ne vérifie pas la propriété de monotonie.

selon RDU. D'autres approches de choix résolu ont été envisagées pour déterminer une stratégie raisonnable à l'aide du critère RDU. On peut mentionner en particulier les travaux de Jaffray et Nielsen [6], dont la démarche diffère de celle du présent papier. En effet, ils considèrent chaque nœud de décision de l'arbre décision hasard comme étant un *ego* du décideur, et visent à déterminer une stratégie réalisant un compromis entre ces différents egos, en s'assurant que toutes les sous-stratégies sont proches de l'optimum pour RDU et stochastiquement non-dominées.

3.3 Complexité du problème

Nous prouvons maintenant que le problème consistant à déterminer une stratégie RDU-optimale est NP-difficile, si on pose que la taille d'une instance du problème correspond au nombre de nœuds de décision impliqués.

Proposition 1 *La recherche d'une stratégie RDU-optimale (problème* RDU-OPT*) dans un arbre décision hasard est un problème NP-difficile.*

Démonstration. On s'appuie sur une réduction polynomiale du problème 3-SAT vers le problème RDU-OPT. Le problème 3-SAT se formule comme suit :

INSTANCE : un ensemble X de variables booléennes, une collection C de clauses sur X telle que $|c| = 3$ pour toute clause $c \in C$.
QUESTION : Existe-t-il une instanciation des variables booléennes de X qui satisfait simultanément toutes les clauses de C ?

Soient $X = \{x_1, \ldots, x_n\}$ et $C = \{c_1, \ldots, c_m\}$. La construction polynomiale d'un arbre décision hasard à partir d'une instance du problème 3-SAT se réalise comme suit. On définit un nœud de décision pour chaque variable de X. Etant donnée x_i une variable de X, le nœud de décision associé dans l'arbre décision hasard, noté également x_i, a deux fils : le premier (nœud de hasard noté V_i) correspond à l'instanciation vrai de x_i, et le second (nœud de hasard noté F_i) correspond à l'instanciation faux de x_i. Soient $\{c_{i_1}, \ldots, c_{i_j}\} \subseteq C$ le sous-ensemble

de clauses dans lesquelles figurent le littéral positif x_i, et $\{c_{i'_1}, \ldots, c_{i'_k}\} \subseteq C$ le sous-ensemble de clauses dans lesquelles figurent le littéral négatif $\bar{x}_i$. Pour chaque clause c_{i_h} $(1 \leq h \leq j)$ on crée comme fils de V_i un nœud terminal noté c_{i_h}, correspondant à la clause c_{i_h}. On crée en outre un fils supplémentaire de V_i noté c_0, correspondant à une conséquence c_0 fictive. De même, on crée un fils de F_i pour chaque clause $c_{i'_h}$ $(1 \leq h \leq k)$, ainsi qu'un fils supplémentaire correspondant à la conséquence c_0 fictive. Le nœud V_i comporte donc $j+1$ fils, tandis que le nœud F_i comporte $k+1$ fils. Afin de constituer un unique arbre décision hasard, on ajoute un nœud de hasard H père de tous les nœuds de décision x_i $(1 \leq i \leq n)$. Enfin, on rajoute un nœud de décision à la racine, ayant H comme unique fils. L'arbre décision hasard ainsi construit comporte $n+1$ nœuds de décision, $2n+1$ nœuds de hasard et au plus $2n(m+1)$ nœuds terminaux. Sa taille est donc en $O(nm)$, ce qui garantit bien la polynomialité de la transformation. A titre d'illustration, sur la partie gauche de la figure 3, nous donnons l'arbre décision hasard obtenu pour l'instance suivante de 3-SAT : $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee x_3 \vee x_4) \wedge (\overline{x_2} \vee \overline{x_3} \vee \overline{x_4})$.

Remarquons qu'on peut établir une bijection entre l'ensemble des stratégies dans l'arbre décision hasard et l'ensemble des instanciations dans le problème 3-SAT de départ. Il suffit pour ce faire de poser $x_i = 1$ dans le problème 3-SAT si et seulement si l'arc (x_i, V_i) figure dans la stratégie, et $x_i = 0$ si et seulement si c'est l'arc (x_i, F_i) qui figure dans la stratégie. Une instanciation satisfaisante (i.e., qui satisfait simultanément toutes les clauses) dans 3-SAT correspond à une stratégie où toute clause c_i $(1 \leq i \leq m)$ figure comme conséquence possible (elle figure donc de une à trois fois). Pour compléter la réduction, il s'agit donc maintenant de définir d'une part les probabilités assignées aux arcs issus des nœuds H, V_i et F_i, et d'autre part les utilités des conséquences et la fonction φ. La réduction va consister à les définir de façon à ce que seules les stratégies correspondant à des instanciations satisfaisantes maximisent RDU. Plus précisément, nous visons à ce que :

(i) la valeur RDU d'une stratégie ne dépende que de l'*ensemble* (et non du multi-ensemble) de ses conséquences possibles (autrement dit l'ensemble des clauses satisfaites par l'instanciation correspondante),

(ii) la valeur RDU d'une stratégie correspondant à une instanciation satisfaisante vaille exactement m,

(iii) si une stratégie est susceptible de conduire à un ensemble de conséquences possibles qui est strictement inclus dans l'ensemble des conséquences d'une autre stratégie, la valeur RDU de cette dernière soit strictement supérieure.

Pour ce faire, après avoir affecté la probabilité $\frac{1}{n}$ aux arcs issus de H, on définit les autres probabilités et les utilités de la façon suivante $(i \neq 0)$:

$$\begin{cases} p_i = (\frac{1}{10})^i \\ u(c_i) = \sum_{j=1}^{i} 10^{j-1} \end{cases}$$

où p_i désigne la probabilité de tout arc conduisant à la conséquence c_i. Pour les arcs de type (V_j, c_0) (resp. (F_j, c_0)), on pose $u(c_0) = 0$ et on affecte la probabilité

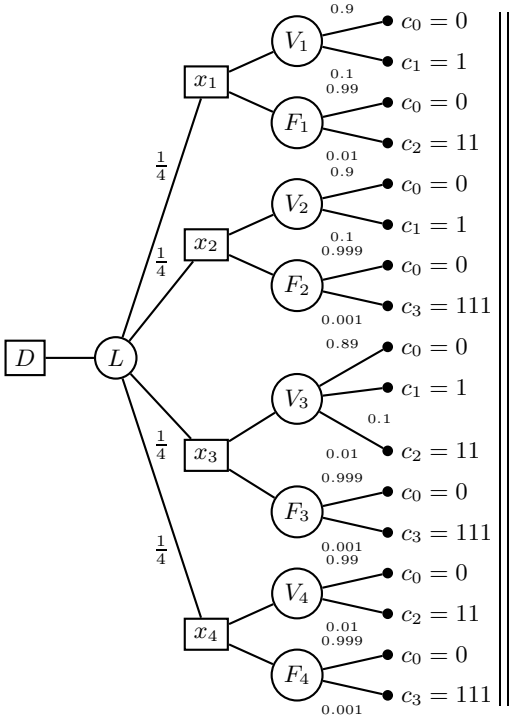


FIG. 3. Exemple de réduction

qui complémente à 1 l'ensemble des probabilités affectées aux arcs issus de V_j (resp. F_j). Notons que cette dernière probabilité est bien positive car la somme des p_i est strictement inférieure à 1. Enfin, la fonction φ est définie comme suit[1] :

$$\varphi(p) = \begin{cases} 0 \text{ si } p \in [0; \frac{p_m}{n}[\\ p_i \text{ si } p \in [\frac{p_{i+1}}{n}; \frac{p_i}{n}[\text{ pour } i < m \\ 1 \text{ si } p \in [\frac{p_1}{n}; 1[\end{cases}$$

A titre d'illustration, sur la partie droite de la figure 3, nous indiquons la fonction φ obtenue pour l'instance de 3-SAT indiquée plus haut.

Dans la suite, on considère une stratégie quelconque Δ, induisant une loterie notée L, et on note $I \subseteq \{0, \ldots, m\}$ l'ensemble des indices des conséquences possible de Δ. Remarquons que la conséquence c_0 est toujours présente dans une stratégie Δ. On appelle $\alpha_i \in \{1, 2, 3\}$ le nombre d'occurrences de la conséquence c_i dans Δ. Par abus de notation, nous confondons ci-dessous c_i et $u(c_i)$.

Preuve de (i). La valeur RDU d'une stratégie Δ quelconque vaut $RDU(L) = c_0 \times \varphi(1) + \sum_{i \in I}(c_i - c_{prec_I(i)})\varphi\left(\sum_{\substack{j \in I \\ j \geq i}} \alpha_j \frac{p_j}{n}\right)$, où $prec_I(i) = \max\{j \in I : j < i\}$. Montrons que $\forall i \in I, \varphi\left(\sum_{\substack{j \in I \\ j \geq i}} \alpha_j \frac{p_j}{n}\right) = \varphi\left(\sum_{\substack{j \in I \\ j \geq i}} \frac{p_j}{n}\right)$.

[1] Remarquons qu'en toute rigueur cette fonction φ est croissante seulement au sens large, mais le lecteur pourra se convaincre facilement qu'on peut l'adapter légèrement pour qu'elle devienne strictement croissante.

Par croissance de φ, on a $\varphi\left(\sum_{\substack{j\in I\\ j\geq i}} \frac{p_j}{n}\right) \leq \varphi\left(\sum_{\substack{j\in I\\ j\geq i}} \alpha_j \frac{p_j}{n}\right) \leq \varphi\left(\sum_{\substack{j\in I\\ j\geq i}} 3\frac{p_j}{n}\right)$.
On a donc $\varphi\left(\sum_{\substack{j\in I\\ j\geq i}} \frac{1}{n}\left(\frac{1}{10}\right)^j\right) \leq \varphi\left(\sum_{\substack{j\in I\\ j\geq i}} \alpha_j \frac{p_j}{n}\right) \leq \varphi\left(\sum_{\substack{j\in I\\ j\geq i}} \frac{3}{n}\left(\frac{1}{10}\right)^j\right)$.
Comme $\varphi(\sum_{\substack{j\in I\\ j\geq i}} \frac{1}{n}(\frac{1}{10})^j) = \varphi(\sum_{\substack{j\in I\\ j\geq i}} \frac{3}{n}(\frac{1}{10})^j) = p_{i-1}$, on a par encadrement $\varphi(\sum_{\substack{j\in I\\ j\geq i}} \alpha_j \frac{p_j}{n}) = \varphi(\sum_{\substack{j\in I\\ j\geq i}} \frac{p_j}{n})$. Or $c_0 \times \varphi(1) = 0$. On conclut donc que $RDU(L) = \sum_{i\in I}(c_i - c_{prec_I(i)})\varphi(\sum_{\substack{j\in I\\ j\geq i}} \frac{p_j}{n})$.

Preuve de (ii). Considérons une stratégie Δ^* correspondant à une instanciation satisfaisante, et la loterie induite L^* où toutes les conséquences c_i de C sont possibles. D'après (i), on a $RDU(L^*) = \sum_{i=1}^{m}(c_i - c_{i-1})\varphi(\sum_{j=i}^{m} \frac{p_j}{n})$. On remarque que pour tout $i \leq m$, $(c_i - c_{i-1})\varphi(\sum_{j=i}^{m} \frac{p_j}{n}) = 10^{i-1} \times p_{i-1} = 10^{i-1} \times (\frac{1}{10})^{i-1} = 1$. Par conséquent, $RDU(L^*) = m$.

Preuve de (iii). Soient Δ (resp. Δ') une stratégie quelconque de loterie induite L (resp. L') et $I \subseteq \{0, \ldots, m\}$ (resp. $J = I \cup \{k\}$) l'ensemble des indices de ses conséquences possibles. On suppose ici que $k < \max I$, le cas $k = \max I$ étant évident. Par définition, $\{i \in I : i \neq k\} = \{i \in J : i \neq k\}$. On peut donc écrire la valeur RDU de Δ comme une sommes de trois termes :
$RDU(L) = \sum_{\substack{i\in J\\ i\leq k-1}} (c_i - c_{prec_J(i)})\varphi\left(\sum_{\substack{j\in I\\ j\geq i}} \frac{p_j}{n}\right) + (c_k - c_{prec_J(k)})\varphi\left(\sum_{\substack{j\in I\\ j\geq k}} \frac{p_j}{n}\right) + \sum_{\substack{i\in J\\ i\geq k+1}} (c_i - c_{prec_J(i)})\varphi\left(\sum_{\substack{j\in J\\ j\geq i}} \frac{p_j}{n}\right)$
De la même manière, la valeur RDU de la stratégie Δ' s'écrit également comme une somme de trois termes :
$RDU(L') = \sum_{\substack{i\in J\\ i\leq k-1}} (c_i - c_{prec_J(i)})\varphi\left(\sum_{\substack{j\in J\\ j\geq i}} \frac{p_j}{n}\right) + (c_k - c_{prec_J(k)})\varphi\left(\sum_{\substack{j\in J\\ j\geq k}} \frac{p_j}{n}\right) + \sum_{\substack{i\in J\\ i\geq k+1}} (c_i - c_{prec_J(i)})\varphi\left(\sum_{\substack{j\in J\\ j\geq i}} \frac{p_j}{n}\right)$
Par croissance de φ, on a $I \subseteq J \Rightarrow \forall i \leq k-1, \varphi(\sum_{\substack{j\in I\\ j\geq i}} \frac{p_j}{n}) \leq \varphi(\sum_{\substack{j\in J\\ j\geq i}} \frac{p_j}{n})$. Ainsi le premier terme de $RDU(L)$ est inférieur ou égal au premier terme de $RDU(L')$. On vérifie facilement que $\varphi(\sum_{\substack{j\in I\\ j\geq k}} \frac{p_j}{n}) = p_{succ_I(k)-1}$ et $\varphi(\sum_{\substack{j\in J\\ j\geq k}} \frac{p_j}{n}) = p_{prec_J(k)} = p_{k-1}$, où $succ_I(i) = \min\{j \in I : j > i\}$. Or $p_{succ_I(k)-1} < p_{k-1}$ car $succ_I(k) - 1 > k - 1$. Donc le second terme de $RDU(L)$ est strictement inférieur au second terme de $RDU(L')$. Enfin, le troisième terme de $RDU(L)$ est bien évidemment égal au troisième terme de $RDU(L')$. Par conséquent $RDU(L) < RDU(L')$.

On conclut de (i),(ii) et (iii) que toute stratégie correspondant à une instanciation non-satisfaisante présente une valeur RDU strictement inférieure à m, et que toute stratégie correspondant à une instanciation satisfaisante présente une valeur RDU exactement égale à m. Trouver une instanciation satisfaisante dans 3-SAT revient donc à trouver une stratégie valant m dans RDU-OPT. ■

Dans la section suivante, nous décrivons un algorithme pour déterminer la stratégie optimale depuis la racine au sens de RDU. Nous procédons par énumération implicite puisque ni une énumération exhaustive des stratégies ni une induction arrière ne sont envisageables.

4 Résolution du problème

4.1 Algorithme d'énumération implicite

Nous présentons ici une méthode par séparation et évaluation pour déterminer la stratégie optimale au sens de RDU dans un arbre décision hasard. Le principe de séparation consiste à partitionner l'ensemble des stratégies possibles en fonction du choix d'une arête (N, N') donnée en un nœud de décision N. Plus formellement, les nœuds de l'arbre de recherche sont caractérisés par une *stratégie partielle*, qui définit un sous-ensemble de stratégies. Soit $\mathcal{T}$ un arbre décision hasard et $\mathcal{N}^\Gamma$ un ensemble de nœuds contenant :

- la racine N_r de $\mathcal{T}$,
- un et un seul successeur pour chaque nœud de décision $N \in \mathcal{N}_D^\Gamma = \mathcal{N}_D \cap \mathcal{N}^\Gamma$.

L'ensemble des arcs orientés $\Gamma = \{(N, N') : N \in \mathcal{N}_D^\Gamma, N' \in \mathcal{N}^\Gamma\} \subseteq \mathcal{E}$ définit une *stratégie partielle* de $\mathcal{T}$ dès lors que le sous-graphe induit par $\mathcal{N}^\Gamma$ est un arbre. Une stratégie Δ est dite *compatible* avec une stratégie partielle Γ si $\Gamma \subseteq \Delta$. Le sous-ensemble de stratégies caractérisé par une stratégie partielle correspond à l'ensemble des stratégies compatibles. Toute stratégie partielle n'est cependant pas susceptible d'être envisagée dans l'arbre de recherche. En effet, les stratégies partielles rencontrées dans l'arbre de recherche respecte un ordre de priorité sur les nœuds de décision sélectionnés dans $\mathcal{N}^\Gamma$ (afin d'éviter les doublons) : si deux nœuds de décision sont susceptibles de prolonger une même stratégie partielle, celui de plus petit rang sera prioritaire sur l'autre pour entrer dans $\mathcal{N}^\Gamma$. Le rang d'un nœud est donné par une fonction $rg : \mathcal{N}_D \mapsto \{1, 2, \ldots, |\mathcal{N}_D|\}$ telle que :

$$\begin{cases} rg(N_r) = 1 \\ |past(N)| > |past(N')| \Rightarrow rg(N) > rg(N') \\ |past(N)| = |past(N')| \text{ et } EU(\mathcal{T}(N)) > EU(\mathcal{T}(N')) \Rightarrow rg(N) < rg(N') \end{cases}$$

où $EU(\mathcal{T}(N))$ correspond à la valeur optimale de EU dans $\mathcal{T}(N)$.

Exemple 6 *Pour l'arbre décision hasard de la figure 1, il existe une unique fonction rg possible définie par :* $rg(D_1) = 1, rg(D_2) = 2, rg(D_3) = 4, rg(D_4) = 3, rg(D_5) = 5$.

L'algorithme 1 décrit la procédure d'énumération implicite que nous proposons. Il prend en argument une stratégie partielle Γ et un réel RDU_{opt} qui correspond à la valeur RDU de la meilleure stratégie trouvée jusqu'alors dans l'exploration. Cette dernière est effectuée en profondeur d'abord. L'ensemble $\mathcal{N}_1$ désigne les nœuds de décision candidats pour prolonger la stratégie partielle Γ. Parmi ceux-ci, le nœud dont la valeur de la fonction rg est minimale est noté N_{min}. L'ensemble $\mathcal{E}_{min}$ de ses arêtes incidentes définit les différents prolongements de Γ envisagés (autrement dit, les fils du nœud associé à Γ dans l'arbre de recherche). Pour toute stratégie partielle Γ (autrement dit, en chaque nœud de l'arbre de recherche), on dispose d'une fonction d'évaluation ev représentant une borne supérieure de la valeur RDU de toute stratégie compatible avec Γ.

Algorithme 1 : **BB**(Γ, RDU_{opt})

$\mathcal{N}_1 \leftarrow \{N_1 \in \mathcal{N}_D : \forall (N,H) \in \mathcal{N}_D \times \mathcal{N}_H, ((N,H) \in past(N_1) \Rightarrow (N,H) \in \Gamma)\}$;
$N_{min} \leftarrow \arg\min_{N \in \mathcal{N}_1} rg(N)$;
$\mathcal{E}_{min} \leftarrow \{(N_{min}, H) \in \mathcal{E} : H \in S(N_{min})\}$;
pour chaque $(N,H) \in \mathcal{E}_{min}$ **faire**
  si $ev(\Gamma \cup \{(N,H)\}) > RDU_{opt}$ **alors**
    $RDU_{temp} \leftarrow$ **BB**$(\Gamma \cup \{(N,H)\}, RDU_{opt})$;
    si $RDU_{temp} > RDU_{opt}$ **alors**
      $RDU_{opt} \leftarrow RDU_{temp}$;
    fin
  fin
fin
retourner RDU_{opt}

Bien que cela ne soit pas précisé dans l'algorithme, remarquons qu'en pratique nous utilisons l'heuristique consistant à développer en priorité le fils dont la valeur de la fonction d'évaluation est la plus élevée. Nous détaillons maintenant les principales caractéristiques de notre algorithme.

Initialisation. Une méthode par séparation et évaluation est notoirement plus efficace quand une bonne solution est connue avant de démarrer la recherche. Dans notre méthode, la borne supérieure (RDU_{opt}) est initialisée avec la valeur RDU de la stratégie obtenue par programmation dynamique selon le critère EU. En effet, on peut penser que la stratégie ainsi obtenue sera de bonne qualité, et permettra donc d'éviter une exploration trop approfondie de sous-espaces ne comportant pas de bonnes solutions.

Fonction d'évaluation. L'évaluation d'un ensemble de stratégies induit par une stratégie partielle Γ se fait à l'aide d'une fonction ev. Le principe de cette évaluation est de déterminer une loterie qui domine stochastiquement toutes les loteries associées aux stratégies compatibles avec Γ, et d'évaluer cette loterie selon le critère RDU. On s'assure ainsi que cette évaluation est bien un majorant puisque le critère RDU respecte la dominance stochastique, c'est-à-dire que si une loterie L domine stochastiquement une loterie L', alors $RDU(L) \geq RDU(L')$. Pour déterminer une telle loterie, on procède par programmation dynamique sur l'arbre décision hasard. L'initialisation de la procédure se fait au niveau des nœuds terminaux : à tout nœud terminal $C \in \mathcal{C}$ est affecté la loterie $(1, u(C))$. Ensuite, en chaque nœud $N \in \mathcal{N}$, on remonte une loterie qui domine stochastiquement toutes les loteries du sous-arbre $\mathcal{T}(N)$. Plus précisément, en un nœud de hasard H, on calcule la loterie L^H induite par les loteries de ses fils de la manière suivante :

$$\forall u,\ L^H(u) = \textstyle\sum_{N \in S(H)} p((H,N)) \times L^N(u)$$

où L^N correspond à la loterie remontée au nœud N. Par ailleurs, en chaque nœud de décision D, on applique la relation de récurrence suivante exprimée sur les fonctions décumulatives[2] (pour simplifier l'écriture) :

[2] Notons qu'on peut manipuler indifféremment une loterie ou sa fonction décumulative, car seule cette dernière importe pour le calcul de RDU.

$$\begin{cases} \forall u,\ G_{L^D}(u) = G_{L^N}(u) & \text{si } \exists N \in S(D) : (D,N) \in \Gamma \\ \forall u,\ G_{L^D}(u) = \max_{N \in S(D)} G_{L^N}(u) & \text{sinon} \end{cases}$$

Enfin, la valeur retournée par ev est $RDU(L^{N_r})$.

Exemple 7 *Reprenons l'arbre décision hasard de la figure 1 et faisons l'hypothèse que* $\Gamma = \{(D_1, H_1), (D_3, H_6)\}$. *Les loteries remontées en chaque nœud seront alors :* $L^{H_3} = (\frac{3}{5}, 2; \frac{2}{5}, 4)$, $L^{H_4} = (1, 3)$, $L^{H_6} = (1, 1)$, $L^{D_2} = (\frac{3}{5}, 3; \frac{2}{5}, 4)$ *(car* $G_{L^{D_2}} = (\max(1,1),\ 2; \max(\frac{2}{5}, 1), 3; \max(\frac{2}{5}, 0), 4)$*)*, $L^{D_3} = L^{H_6} = (1, 1)$, $L^{H_1} = (\frac{3}{5} \times 1, 1, \frac{2}{5} \times \frac{3}{5}, 3; \frac{2}{5} \times \frac{2}{5}, 4) = (\frac{3}{5}, 1, \frac{6}{25}, 3; \frac{4}{25}, 4)$, $L^{D_1} = L^{H_1} = (\frac{3}{5}, 1, \frac{6}{25}, 3; \frac{4}{25}, 4)$. *La valeur retournée par la fonction d'évaluation pour* $\Gamma = \{(D_1, H_1), (D_3, H_6)\}$ *sera donc* $ev(\Gamma) = RDU((\frac{3}{5}, 1; \frac{6}{25}, 3; \frac{4}{25}, 4))$.

4.2 Expérimentations numériques

L'algorithme a été implémenté en C++ et les test ont été menés sur un ordinateur équipé d'un biprocesseur Intel à 2.13 Ghz avec 3.5 Go de mémoire vive. Les arbres décision hasard sur lesquels nous avons testé notre algorithme sont des arbres binaires complets de profondeur paire. Les utilités et les probabilités ont été générées de manière aléatoire. Les utilités varient de 1 à 500. La profondeur des arbres varie quant à elle de 4 à 14 (donc de 5 à 5461 nœuds de décision), avec une alternance de nœuds de décision et de nœuds de hasard. Pour chaque niveau de profondeur, 100 arbres ont été générés. La courbe de gauche (resp. droite) de la figure 4 représente le nombre moyen de nœuds développés dans l'arbre d'exploration (resp. le temps moyen d'exécution en sec. de l'algorithme) selon la profondeur. L'axe des ordonnées est exprimé sur une échelle logarithmique (en base 4) car le nombre de nœuds de décision est multiplié par 4 en ordre de grandeur pour chaque incrément de la profondeur. On remarque que, sur les instances tirées aléatoirement, la croissance du nombre de nœuds développés (resp. du temps d'exécution) apparaît comme linéaire du nombre de nœuds de décision pour les tailles traitées ici. Les plus grandes instances se rapprochent des tailles d'arbre décision hasard limites stockables en machine (une augmentation de 30% seulement de la profondeur peut être envisagée). Le saut de complexité se situe au-delà des tailles traitées ici.

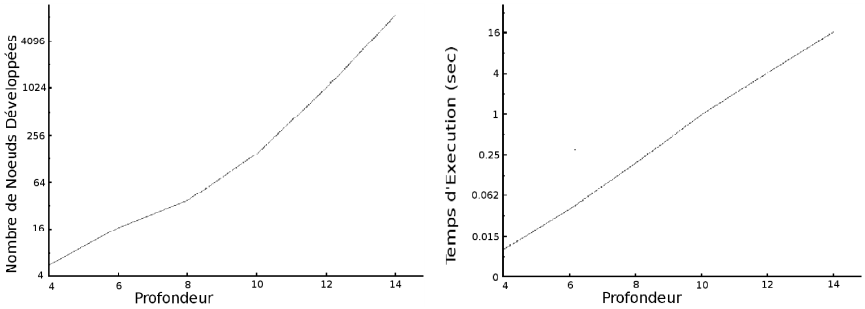


Fig. 4. Comportement de l'algorithme en fonction de la profondeur.

5 Conclusion

Dans ce papier, nous avons mené une étude algorithmique du problème de la recherche d'une stratégie RDU-optimale dans un arbre décision hasard. Nous avons en particulier montré que ce problème est NP-difficile. Nous avons ensuite proposé un algorithme d'énumération implicite pour déterminer une stratégie RDU-optimale. Les tests numériques conduits montrent que cet algorithme permet de résoudre avec des temps compétitifs des instances dont la taille approche la limite mémoire imposée par la machine. Un sujet d'étude intéressant pour des travaux futurs serait justement de concevoir des algorithmes de résolution pour des problèmes de décision dans le risque modélisés à l'aide d'un diagramme d'influence. Un diagramme d'influence est un graphe orienté sans circuit représentant de façon compacte un arbre décision hasard en exploitant les symétries présentes. Par exemple, sur l'arbre de la figure 2, les nœuds D_2 et D_3 peuvent être "factorisés" en un seul car les sous-arbres associés sont identiques. Néanmoins, une difficulté supplémentaire pour la résolution est qu'une stratégie RDU-optimale peut conduire à ne pas faire le même choix en deux nœuds de décision distincts associés à des sous-arbres pourtant identiques (alors qu'il existe toujours une stratégie EU-optimale où l'on prend la même décision). C'est le cas par exemple en D_2 et D_3 pour l'arbre de la figure 2.

Remerciements

Nous remercions Patrice Perny qui a porté notre attention sur le sujet étudié ici, Christophe Gonzales avec qui nous avons eu de multiples échanges qui ont contribué à ce travail, ainsi que les relecteurs anonymes pour leurs suggestions pertinentes.

Références

1. Allais, M. (1979) : The foundation of a positive theory of choice involving risk ans a criticism of the postulate and axioms of the american school. In *Expected utility hypotheses and the Allais paradox*, pages 27 145. Dordrecht, Holland. Original work published in 1952.
2. Dubois, D., Prade, H. and Sabbadin, R. (2001) : Decision-theoretic foundations of qualitative possibility theory. *European Journal of Operational Research*, 128(3), 459-478.
3. Escoffier, B. and Spanjaard, O. (2005) : *Programmation dynamique*. Dans l'ouvrage collectif Optimisation combinatoire, Volume 1. Edité par V. Th. Paschos. Hermes.
4. Gayant, J.-P. (2001) : *Risque et décision*, Vuibert.
5. Handa, J. (1977) : Risk, probabilities and a new theory of cardinal utility. *Journal of Political Economics*, 85, 97-122.
6. Jaffray, J-Y. and Nielsen, T. D. (2006) : An operational approach to rational decision making based on rank dependent utility. *European Journal of Operational Research*, 169(1), 226-246.

7. Kahneman, D. and Tversky, A. (1979) : Prospect Theory : An Analysis of Decision under Risk. *Econometrica*, 47, 263-291.

8. McClennen, E.F. (1990) : *Rationality and Dynamic choice : Foundational Explorations*, Cambridge University Press, Cambridge.

9. Morin, T.L. (1982) : Monotonicity and the principle of optimality. *Journal of Mathematical Analysis and Applications*, 86, 665-674.

10. Quiggin, J. (1982) : A theory of anticipated utility. *Journal of Economic Behavior and Organisation*, 3(4), 323-343.

11. Raiffa, H. (1968) : *Decision Analysis : Introductory Lectures on Choices under Uncertainty*, Addison-Wesley.

12. von Neuman, J. and Morgenstern, O. (1947) : *Theory of games and economic behaviour*. Princeton University Press, 2nd edition.