

Application de l'apprentissage par renforcement à la gestion du risque

René Aïd¹, Vincent Grellier¹, Arnaud Renaud², Olivier Teytaud²

¹ EDF R&D 1, Avenue du Général de Gaulle, 92 140 Clamart, www.edf.fr

² ARTELYS, 215 Avenue Jean-Jacques Rousseau, 92 136 Issy-Les-Moulineaux, www.artelys.com

Résumé La programmation dynamique stochastique est un principe de décomposition classique pour l'optimisation dynamique. Elle permet l'optimisation de tout critère séparable. En particulier, l'espérance est un critère séparable. Par contre, l'ajout d'une prise en compte du risque par une mesure de type Value-At-Risk rend le problème non-séparable, donc le traitement par programmation dynamique stochastique standard est impossible. Cet article présente une application de techniques d'apprentissage par renforcement compatibles avec un critère non-séparable. La mise en œuvre pratique est faite dans le cadre de la production électrique par le parc de production thermo-hydraulique d'EdF. Les courbes de Value-At-Risk obtenues montrent le succès de l'approche : augmenter le paramètre α du critère $(1 - \alpha)E + \alpha VaR$ conduit à des risques plus faibles.

Mots clef Renforcement – Risque – Gestion de stock – Optimisation dynamique stochastique

1 Introduction

La gestion d'un complexe thermo-hydraulique est un problème dynamique et stochastique. La présence de stocks hydrauliques lui confère sa structure dynamique (l'état est le niveau de stock) et les incertitudes sur les données (demande, apports, prix de marché, disponibilité des moyens de production) sa structure stochastique. La gestion optimisée d'un tel système nécessite l'élaboration d'une stratégie d'utilisation du stock permettant de minimiser l'espérance de coût (ou de maximiser l'espérance de profit) sur un ensemble de scénarios. Il s'agit, alors que le futur est incertain, de savoir dans quelle proportion on doit utiliser le stock pour satisfaire la demande, et dans quelle mesure il est préférable de conserver ce stock pour plus tard.

Une solution classique pour optimiser le choix d'une stratégie lorsque le critère est séparable (ie, lorsque la stratégie peut se construire pas de temps par pas de temps, à rebours) est de faire appel à la programmation dynamique ; en particulier, l'espérance est séparable. Malheureusement, dans de nombreux cas, optimiser l'espérance est périlleux ; malgré un très bon résultat en moyenne, la probabilité d'une catastrophe est trop forte. Le cas des marchés financiers l'illustre bien, d'où le succès de la gestion du risque dans le cadre d'applications à composante financière notamment. Il faut donc :

- choisir un critère autre que l'espérance comme critère d'optimisation, le critère choisi devant privilégier des solutions robustes ;
- réussir à optimiser ce critère, alors même qu'il n'est pas compatible avec la technique classique de programmation dynamique.

Des caractéristiques statistiques classiques d'une variable aléatoire C représentant un coût sont les suivantes :

- l'**espérance** $E(C)$ (i.e. la moyenne),
- la **variance** $var(C) = E(C - E(C))^2$,
- la **demi-variance** : il s'agit de l'espérance de $[(C - E(C)) \times 1_{C > E(C)}]^2$, où 1_P désigne la fonction valant 1 lorsque P a lieu et 0 sinon ; la demi-variance répond à la critique faite à la variance de considérer comme équivalentes une stratégie potentiellement très rentable et une stratégie potentiellement très coûteuse,
- la **probabilité de chute** (shortfall probability, ou Risk At Value) : cela désigne, pour un seuil de coût C' , la probabilité pour que $C \geq C'$ (il s'agit donc aussi de $E(1_{C \geq C'})$),
- et la **VaR (Value At Risk)** $VaR(C, r)$, égale, pour un seuil de risque r donné, à C' minimal tel que $P(C > C') \leq r$; c'est-à-dire que $VaR(C, r)$ est minimal parmi les solutions de $P(C > VaR(C, r)) \leq r$.

Les représentations de type "Value-At-Risk" ou "Risk-At-Value" sont les plus complètes au sens où toute la distribution des coûts est représentée. Cette étude est centrée sur la notion de VaR comme mesure de risque. Par l'utilisation de l'apprentissage par renforcement, nous souhaitons ainsi déterminer une stratégie optimisant un compromis espérance/risque du type $(1 - \alpha)E + \alpha VaR$, non-séparable.

La section 2 présente la problématique et l'état de l'art, la section 3 présente l'algorithme retenu et sa mise en œuvre, la section 4 présente un bilan, la section 5 conclut.

2 Problématique et méthodes

Nous présentons dans cette partie la problématique de l'optimisation dynamique stochastique et le problème spécifique ici étudié (2.1), puis les méthodes d'apprentissage par renforcement (2.2).

2.1 Optimisation dynamique stochastique et gestion de stocks hydrauliques

Cadre général de l'optimisation dynamique stochastique : (voir figure 1) dans un cadre très général, un problème d'optimisation dynamique stochastique est constitué d'un système qui évolue dans le temps, qui dispose d'un état interne, et que dont on souhaite optimiser les trajectoires, pour un critère donné. Concrètement, on dispose :

- d'un processus stochastique $p(t)$ (à temps discret ici, entre 0 et T) : $p(0), \dots, p(T)$.

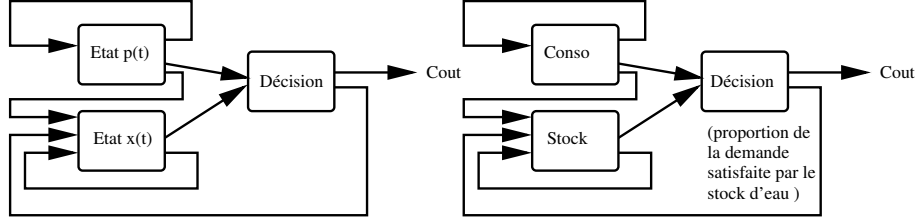


Fig. 1. Cadre général de l'optimisation dynamique (à gauche) et cadre particulier ici développé (à droite). La consommation est produite d'un processus de Markov et d'un bruit blanc (mais un cadre plus général serait possible). Les contraintes sont de produire autant d'électricité qu'il y a de demande ; produire de l'électricité dans les groupes thermiques coûte cher, alors qu'utiliser le stock d'eau pour produire de l'électricité ne coûte rien immédiatement, mais il se peut qu'à long terme on paie cher le fait de ne pas avoir conservé assez d'eau ; en effet, sans eau, il devient nécessaire d'utiliser des groupes de production thermiques d'autant plus coûteux que la production nécessaire est forte. Une stratégie prudente consiste à n'utiliser le stock d'eau que parcimonieusement, afin d'éviter des coûts importants, et une stratégie risquée consiste à utiliser le stock intégralement pour toujours éviter d'utiliser la production thermique (quitte à risquer des frais beaucoup plus forts les années où la consommation reste forte longtemps, car le stock finit par être à vide).

– d'une équation d'évolution du système : $x_{t+1} = f(x_t, u(p(t), t, x_t), p(t))$, et l'on souhaite déterminer une stratégie (on dit aussi un contrôleur) $u(\cdot, \cdot, \cdot)$ tel que l'espérance de $c_1(x_1, u_1) + c_2(x_2, u_2) + \dots + c_T(x_T, u_T)$ soit minimale (u_t désigne $u(p(t), t, x_t)$).

On dit alors que $u()$ est un contrôleur optimal en espérance.

Principe de décomposition de Bellman : Dans l'esprit du principe de décomposition de Bellman ([Bellman, 1955], dit aussi principe de Hamilton-Jacobi-Bellman), une façon de faire consiste à déterminer $V(p(t), t, x_t)$ égal à l'espérance de $c_t(x_t, u_t) + c_{t+1}(x_{t+1}, u_{t+1}) + \dots + c_T(x_T, u_T)$ conditionnellement à $p(t)$ et avec $x(t)$ donné, pour une stratégie optimale $u()$; c'est-à-dire qu'on cherche V telle que pour u optimal,

$$V(\tilde{p}, t, \tilde{x}) = E(c_t(x_t, u_t) + c_{t+1}(x_{t+1}, u_{t+1}) + \dots + c_T(x_T, u_T) | p(t) = \tilde{p})$$

avec $x_t = \tilde{x}$ et $x_{t+1} = f(x_t, u(p(t), t, x_t), p(t))$.

Cette formule définit V **fonction de valeur**, dépendant de la stratégie u . V est la fonction de valeur associée à u , et l'on va essentiellement s'intéresser à V associé à une stratégie optimale u . On peut montrer qu'en fait $V()$ est le même pour toute stratégie optimale $u()$.

Si l'on a réussi à estimer V , fonction de valeur associée à une stratégie optimale, alors on connaît une stratégie optimale. En effet, il suffit alors de déterminer, à chaque pas de temps, $u(p(t), t, x_t) \in \operatorname{argmin}_u c_t(x_t, u) + V(p(t+1), t+1, x_{t+1})$. La fonction $V(\cdot, \cdot, \cdot)$ est appelée fonction de Bellman, ou fonction de valorisation, ou, dans le cadre de l'apprentissage

par renforcement, fonction critique ou fonction de valeur. Ici, on se préoccupera de calculer $V(\cdot, \cdot, \cdot)$ effectivement égal à une espérance, d'une part, mais on se préoccupera surtout de V égal à un compromis $(1 - \alpha)$ Espérance + α Value-At-Risk, permettant d'optimiser non pas en espérance, mais en espérance et risque.

Dans le problème ici traité : on considère un stock hydraulique, à utiliser conjointement à un ensemble de centrales thermiques pour satisfaire une demande stochastique en électricité.

Précisément :

- tous les jours, on doit s'efforcer de satisfaire une demande en électricité. Cette demande est partagée en 3 quantités¹, la troisième étant presque toujours nulle :
 - production d'électricité par centrales thermiques ;
 - production d'électricité par centrales hydroélectriques ;
 - défaillance (c'est-à-dire non-satisfaction de la demande).
- la production par centrales thermiques a un coût croissant convexe (ie coût marginal croissant) en fonction de la production, car plus l'on doit produire fortement, plus l'on est obligé d'utiliser de centrales, et donc plus l'on en utilise des moins économiques². La production est aussi bornée par les capacités de production.
- la production hydroélectrique a nombre d'avantages, car il s'agit de rentabiliser l'eau fournie naturellement (fonte des neiges, etc) sous forme d'électricité. Malheureusement, la capacité est limitée, et il faut être prudent ; si l'on utilise très abondamment les stocks d'eau, pour peu que la demande soit un peu plus forte que prévue ou les arrivées d'eau un peu plus faibles que prévues, on risque de ne plus avoir du tout d'eau au moment où le besoin est le plus crucial, conduisant au mieux à des surcoûts, au pire à des défaillances.

Le problème est donc :

- d'avoir un modèle aussi précis que possible des aléas (apports en eau, consommation), et notamment de leur dépendance à la météorologie ; les prévisions météorologiques de long terme avec leurs intervalles de confiance sont donc particulièrement importantes ;
- de savoir, à une date donnée, en fonction du modèle précédemment défini et des niveaux de stocks courants, quelle part de la demande doit être satisfaite par la production thermique et quelle part doit être satisfaite par hydroélectricité.

Formellement :

¹ D'autres productions d'électricité existent mais sont préliminairement "défaillées" de la demande sans perte de généralité, car non-paramétrables.

² On peut aussi noter que le coût, à production égale, n'est pas le même selon le jour de l'année ; indisponibilités programmées ou fortuites pour maintenance par exemple, ou autres contraintes techniques.

- l'aléa ³ est la demande $d(t)$ en électricité; c'est le produit d'un processus de Markov $d_1(t)$ et d'un bruit indépendant $d_2(t)$: $d(t) = d_1(t)d_2(t)$; dans toute la suite, le **niveau de demande** désigne $d_1(t)$;
- la production $p_c(t)$ des centrales thermiques est à choisir par la stratégie; elle est comprise entre $p_{min}(t)$ et $p_{max}(t)$ (contraintes d'exploitation des centrales),
- la production $p_h(t)$ des centrales hydrauliques est à choisir par la stratégie; est comprise entre 0 et p_h^{max} ,
- l'équation d'évolution du stock est la suivante : $stock(t+1) = stock(t) + a(t) - p_h(t) - dev(t)$. où $a(t)$ est un apport en eau (noté comme une énergie; il s'agit d'arrivée naturelle des eaux dans les stocks par fonte des neiges ou autres facteurs), $dev(t)$ est du déversement (ie, $dev(t) > 0$ signifie que de l'eau est déversée du barrage sans produire d'énergie, à seule fin d'éviter de dépasser le niveau maximal du stock).
- la stratégie doit satisfaire les contraintes suivantes :
 - la contrainte couplante de demande $p_c(t) + p_h(t) = d(t)$,
 - $stock(t) \geq 0$ et $stock(t) \leq stock_{max}(t)$.
- la fonction de coût c est la somme sur les instants t des $c(t, p_c(t))$ où $c(t, \cdot)$ est une fonction de coût dépendant de t , croissante et convexe, plus une pénalisation de stock final $c(stock(T))$ (décroissante). Les coûts $c_t(x_t, u_t)$ sont ainsi $c(t, p_c(t))$ pour $t < T$ et $c(stock(T))$ pour $t = T$. L'**état** désigne le couple $(d_1(t), stock(t))$ (ie, l'état interne $stock(t)$ et l'état du processus stochastique $d_1(t)$).

La stratégie $u(\cdot)$ détermine $p_c(t)$ et $p_h(t)$. Pour faire le lien entre le problème spécifique ci-dessus défini et le cadre général défini plus haut, on a :

- un état interne $x_t = stock(t)$,
- un aléa $p(t) = (d_1(t), a(t))$ (ici $a(\cdot)$ est déterministe),
- une loi d'évolution $f(x_t, u_t, p(t)) = stock(t) + a(t) - p_h(t) - dev(t)$ (où $dev(t)$ est fixé par la satisfaction de la contrainte $stock(t) \leq stock_{max}(t)$).
- on cherche une stratégie $u_t = u(p(t), t, stock(t))$

Les programmations dynamiques utilisées pour référence travaillent sur l'état interne x_t et l'aléa $d_1(t)$ ($a(t)$ étant déterministe).

2.2 Méthodes d'apprentissage par renforcement

Présentation générale. Les méthodes d'apprentissage par renforcement sont venues de la communauté intelligence artificielle. Souvent tournées vers la robotique, parvenues à un fort prestige depuis TD-gammon (meilleur logiciel actuel de backgammon, jeu très difficile, mal traité par les techniques classiques, [Tesauro, 1989]), elles deviennent une référence classique dans les domaines usuellement traités par programmation dynamique. Son origine entraîne un vocabulaire quasi-anthropomorphe : un acteur désigne par la suite une stratégie de gestion (i.e., une méthode décidant d'une action en fonction de l'état et du pas de temps - on parle aussi de stratégie de gestion), et un critique est un

³ Un aléa est un processus stochastique; il s'agit ici de la demande en électricité.

outil permettant d'évaluer à quel point on est dans une "bonne" situation (typiquement, en vocabulaire plus algorithmique, un critique est une application qui évalue une valeur de Bellman éventuellement nuancée par une notion de risque). Très concrètement, le critique est donc un module qui à une situation associe un coût à venir (moyen, ou moyen altéré par une notion de risque), et l'acteur est le module qui prend la décision. L'idée générale des méthodes d'apprentissage par renforcement (voir [Bertsekas et al, 1996, Sutton et al, 1998]) est la suivante :

- On dispose d'un module-acteur et/ou d'un module-critique initial.
- Réalisation de simulations à partir du module acteur courant.
- Si l'on dispose d'un module-critique, mise à jour du module critique (de manière plus ou moins incorporée aux simulations selon les algorithmes) : on évalue, pour tout état, en fonction de résultats de simulations avec l'acteur courant comme stratégie, l'espérance (ou une autre statistique que l'espérance) du coût de gestion à partir de cet état. La fonction obtenue, qui à un état associe une valeur, est appelée "fonction de valeur" ; le module critique est le module qui la calcule.
- Si l'on dispose d'un module-acteur, mise à jour du module acteur (de manière plus ou moins incorporée aux simulations selon les algorithmes) : on modifie l'acteur de manière à ce qu'il optimise la fonction de valeur.
- On retourne à l'étape "simulations".

Si le module acteur est en fait purement dépendant de la fonction de valeur (on prend ses décisions en optimisant le coût futur moyen tel qu'estimé par la fonction de valeur), on parle de méthode purement critique ; si l'acteur est lui-même une application calibrée à partir des simulations et de la fonction critique, on parle de méthode acteur-critique ; si l'on procède directement par modification de la fonction acteur sans utiliser de module critique, on parle de méthode purement acteur.

Le lecteur est renvoyé à [Bertsekas et al, 1996] ou [Sutton et al, 1998] pour plus d'informations.

Introduction du risque en apprentissage par renforcement. Les méthodes d'apprentissage par renforcement (Q-learning, évoqué plus bas mais non testé ici car vraisemblablement peu adéquat, $TD(\lambda)$), actuellement en fort essor dans le cadre de l'optimisation d'une espérance, commencent à comporter des notions de risque, et s'appliquent désormais dans le domaine financier [Neuneier, 1996], [Neuneier, 1998], [Dempster et al, 2001]. L'introduction du risque peut se faire de plusieurs façons différentes : définition d'états "catastrophes" à éviter (voir par exemple [Geibel, 2001]) et utilisation directe d'un critère autre que l'espérance. Par exemple, citons [Coraluppi et al, 1999] avec l'optimisation au pire cas (pas forcément réaliste dans un cadre pratique, notamment dans notre cas). [Heger, 1994] considère en outre le critère moyenne-variance et des critères négligeant un faible pourcentage de mauvais cas. [Neuneier et al, 1999] définissent des algorithmes plus ou moins averses au risque selon un paramètre κ . Nous choisissons dans la présente étude l'optimisation d'un compromis espérance/value-at-risk.

Comparaison avec la programmation dynamique classique. On peut comparer ces méthodes d'apprentissage par renforcement à la programmation dynamique classique comme suit :

- **Les techniques classiques d'optimisation stochastique comme la programmation dynamique ont l'avantage d'une robustesse certaine** ; elles disposent d'algorithmes dont le temps de calcul est déterminé à l'avance, et les résultats sont bien stables, alors que les techniques d'apprentissage par renforcement sont basées sur des simulations de type Monte-Carlo coûteuses en puissance de calcul, demandant souvent un grand nombre d'essais - erreurs (voir [Bertsekas et al, 1996]) dû au grand nombre d'hyperparamètres (fréquence de mise à jour de la fonction de valeur, pas de la "descente de gradient"⁴, mais aussi compromis entre l'espérance et le risque si l'on utilise une contrainte de type Risk-At-Value, et paramètres de régularisation si l'on utilise des outils d'extrapolation élaborant une fonction de valeur à partir d'exemples), et ne conduisant à de bons résultats que de manière instable. Toutefois, les méthodes de programmation de l'apprentissage par renforcement commencent à être bien maîtrisées ([Sutton et al, 1998]) et des résultats de convergence ont été prouvés (voir [Bertsekas et al, 1996]).

- **L'apprentissage par renforcement peut commodément travailler sur de simples chroniques historiques⁵ et en particulier ne nécessite pas la définition structurée d'un processus stochastique**, alors que les techniques basées sur l'optimisation stochastique de chroniques arborescentes⁶ par programmation dynamique supposent un modèle. Il permet en outre d'ajouter de nouvelles chroniques à un modèle sans partir de zéro ; cette propriété peut en particulier être utile lorsque l'on requiert l'adaptativité face à des pannes pour l'allocation de canaux (voir [Bertsekas et al, 1996, Singh et al, 1996]) ou le routage ([Boyan et al, 1993]).

- **L'apprentissage par renforcement permet naturellement de prendre en compte des caractéristiques de la distribution de coût plus élaborées que des critères séparables** ; en particulier l'espérance pondérée par le risque et non seulement l'espérance. Ce point est le sujet principal de cet article ; on utilise des simulations pour disposer d'une représentation de la distribution des coûts futurs autre que le coût futur moyen. Une méthode propageant un chiffre à rebours comme la programmation dynamique ne peut fonctionner car le critère n'est pas séparable et nous proposons une méthode itérative permettant de faire converger la fonction de valeur en horizon fini.

- **Les méthodes par renforcement s'accommodent plus facilement de données non-discrétisées** (voire non-discrétisées en temps, voir par exemple

⁴ Le "pas" de la descente de gradient désigne ici le paramètre quantifiant les poids relatifs de la fonction de valeur précédente et des résultats de simulation.

⁵ Une chronique historique note ici une suite déterministe de valeurs constatées dans la réalité.

⁶ Une chronique arborescente est un modèle de Markov tel que pour chaque état x la probabilité de transition de y vers x n'est non-nulle que pour un seul x (il n'y a pas de "recombinaison").

[Doya, 2002]), **ou de grands nombres de variables d'état**, lorsqu'elles sont couplées à des méthodes d'extrapolation ([Bertsekas et al, 1996], chapitres 3 et 6, [Sutton et al, 1998]). ([Coulom, 2002]) montre un essai réussi avec 4 variables continues de décision et 12 variables d'état. On peut noter que le couplage de la programmation dynamique et de méthodes poussées d'extrapolation pourrait s'envisager, quoiqu'un peu moins commodément qu'en apprentissage par renforcement.

3 Algorithmes retenus et mise en œuvre

On présente dans cette section des choix algorithmiques et méthodologiques (3.1), le fonctionnement du module critique (3.2), la mise en œuvre pratique (3.3), le pseudo-code de l'algorithme (3.4), et enfin des compléments algorithmiques testés (3.5).

3.1 Choix algorithmiques et méthodologiques.

$TD(\lambda)$ a été expérimenté avec succès et stabilité sur une vaste gamme de problèmes; en outre, s'il exige un simulateur, il est parfaitement compatible avec des contraintes complexes à moindre coût de programmation et de temps de calcul. $TD(\lambda)$ est basé sur la mise-à-jour d'une fonction de valeur, analogue des valeurs de Bellman, par simulations. Il peut être basé sur des simulations de différentes formes selon la valeur de λ , paramètre compris dans $[0, 1]$ (voir [Sutton, 1988]). **Nous utiliserons $TD(1)$, qui permet l'introduction commode de notions de risque.** $TD(1)$, contrairement à d'autres $TD(\lambda)$, fournit la distribution réelle des coûts futurs correspondant à l'état considéré et non des coûts futurs perturbés par la fonction de valeur; l'estimation de Value-At-Risk est ainsi directe. L'idée sera donc d'approximer $V(\cdot, t, \cdot)$ à l'aide des coûts constatés $c_t(x_t) + c_{t+1}(x_{t+1}) + \dots + c_T(x_T)$.

Nous procéderons par **misés à jour dites "non-optimistes"**, i.e. basées sur des mises à jour lentes (précisément, on effectue un nombre conséquent de simulations avant de procéder à une mise à jour de la fonction de valeur), méthode réputée plus robuste (voir les phénomènes d'oscillations cités dans [Bertsekas et al, 1996] : dans le cadre "optimiste" la fonction de valeur peut converger sans qu'il y ait convergence de la stratégie).

L'introduction du risque sera faite par l'intermédiaire de la Value-At-Risk. Le module d'extrapolation, permettant d'extrapoler des value-at-risk pour des niveaux de stocks continus à partir d'exemples en nombre fini, comportera l'introduction de contraintes sur les fonctions de valeurs, supposées convexes notamment; non seulement ces contraintes sont à peu près réelles (dans le cas d'une espérance pure tout du moins, on sait que la valorisation est convexe - ce n'est pas, en toute rigueur, nécessairement le cas pour une Value-At-Risk), mais en outre on se ramène ainsi, comme on le détaillera plus loin, à un problème d'optimisation quadratique convexe avec contraintes linéaires, donc à un problème pour lequel des méthodes efficaces et robustes existent.

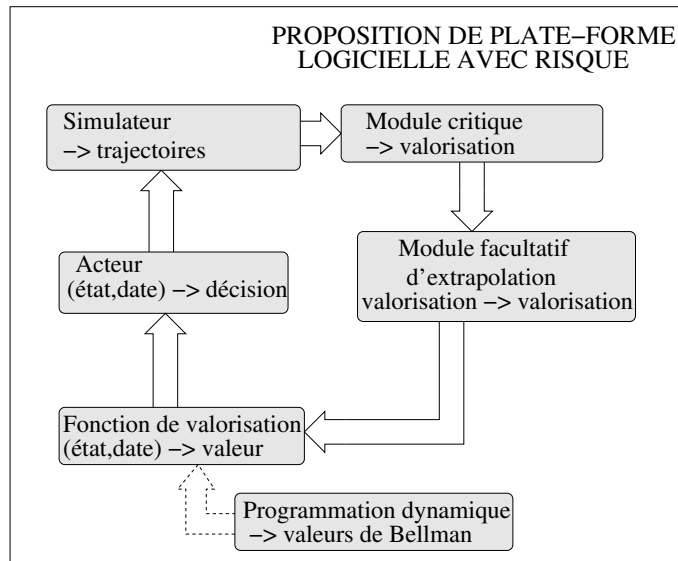


Fig. 2. Architecture choisie.

La structure choisie est décrite en figure 2.

Une programmation dynamique stochastique fournit une première fonction de valeur, dans un cas sans Value-At-Risk et avec un aléa un peu simplifié, afin de fournir une première fonction de valeur (ie, un point initial pour les itérations de l'apprentissage par renforcement) pas trop mauvaise (c'est la première ligne de la section 2.2). Le couplage apprentissage par renforcement/programmation dynamique stochastique permettra de tirer parti de la stabilité de la programmation dynamique et de la capacité de l'apprentissage par renforcement à transformer une gestion optimale en espérance (fournie approximativement au moins par la programmation dynamique) en une stratégie comportant une notion de risque (après apprentissage par renforcement lors d'une phase de simulation). Le module d'extrapolation est ici une extrapolation par fonctions convexes, sans critères de régularisation (on ne pénalise pas la dérivée seconde ou des critères de ce type). L'utilisation de régularisations (en limitant la dérivée seconde de la fonction de valeur par exemple) est nécessaire lorsque le nombre de variables d'états devient important, notamment au niveau des variables continues ; ce n'est pas le cas ici.

La programmation dynamique est seulement destinée à fournir une valorisation préliminaire ne prenant pas en compte la notion de risque ; théoriquement facultative, elle fournira néanmoins une base solide de valorisation, susceptible de compenser les difficultés de paramétrage de l'apprentissage par renforcement. Le module critique, disposant de trajectoires complètes, permet de prendre en compte des notions plus complexes que l'espérance : Risk-At-Value

(RaV) ou Value-At-Risk (VaR) notamment. On peut considérer par exemple, avec C la variable aléatoire associée au coût de gestion, l'optimisation de $C1 = (1 - \alpha)E(C) + \alpha P(C > K)$, avec E l'opérateur espérance et P la probabilité, i.e. prise en compte de RaV, ou l'optimisation de $C2 = (1 - \alpha)E(C) + \alpha K$ avec K tel que $P(C > K) = 1 - \eta$, i.e. prise en compte de VaR. Dans le cadre du Risk-At-Value et en toute rigueur, il faudrait, pour optimiser par exemple le risque d'un coût de gestion supérieur à K , utiliser des fonctions basées non seulement sur l'état et la date, mais aussi sur le coût de gestion passé. Dans le cadre de la Value-At-Risk, cela n'est pas nécessaire : le critère $C2$ ne change en fonction du coût passé qu'à une constante près, et donc le coût passé n'a pas à influencer l'acteur.

Les paramètres à fixer sont les suivants : la fréquence de mise à jour (le nombre de simulations entre chaque mise à jour de $V()$ et π ; suivant une terminologie usuelle, le "degré d'optimisme" de la méthode), le pas de descente et son évolution dans le temps, le seuil de confiance en œuvre dans la notion de risque (typiquement, 5%), le paramètre de compromis entre espérance et VaR, les paramètres du module de régularisation, et enfin des choix de synchronisme ou asynchronisme ont enfin à être faits : modifie-t-on la fonction $V()$ (donc $\pi()$) en cours de simulation ou une fois une (ou plusieurs) simulation(s) entière(s) réalisée(s) ?

3.2 Le composant critique.

Le composant critique doit être capable de fournir une représentation par une ligne brisée⁷ de la fonction de valeur $V(p, t, x)$ choisie, pour un acteur donné. Pour nous, p est la valeur du processus stochastique (la demande), t la date, x est un niveau de stock. Pour cela, pour chaque étape $n = 1, 2, 3, \dots$:

- On effectue des simulations (la procédure de simulation et notamment l'échantillonnage seront détaillés plus tard).
- On obtient des "exemples" (p, t, x, c) , où t est une date, p un état du processus stochastique, x un stock, c un coût à venir obtenu sur une simulation.
- On approxime l'espérance du coût futur par des lignes brisées (une par pas de temps et par état de l'aléa, la ligne brisée relie donc des couples (niveau de stock, coût futur moyen)).
- On approxime la Value-At-Risk par des lignes brisées (une par pas de temps et par état de l'aléa, la ligne brisée relie donc des couples (niveau de stock, quantile⁸ du coût)).
- On détermine alors les valeurs, en combinant la Value-At-Risk et l'espérance.

Les deux approximations (Value-At-Risk et espérance) sont obtenues de manière itérative : à chaque itération, on estime par simulations Value-At-Risk

⁷ Une ligne brisée convexe permet d'avoir une optimisation, pour le choix de la décision, qui soit une optimisation quadratique convexe à contraintes linéaires

⁸ Le quantile h d'une variable aléatoire est la quantité que cette variable aléatoire excède avec probabilité $1 - h$.

et espérance pour la politique de gestion obtenue, puis on adapte la politique de gestion en fonction de la fonction de valeur obtenue (qui combine Value-At-Risk et espérance). Soit V_n^E et V_n^V , fonctions estimations à l'étape n respectivement de l'espérance et de la Value-At-Risk du coût à venir. V_n^E et V_n^V sont des lignes brisées. A partir de V_n^E et V_n^V , on construit V_{n+1}^E et V_{n+1}^V .

$V_{n+1}^E(p, t, \cdot, c)$ et $V_{n+1}^V(p, t, \cdot, c)$ sont les fonctions lignes-brisées (à points de césure fixés cs_i) qui minimisent (sous contraintes définies plus bas) la somme des termes $T1, T2, T3, T4$:

$$T1 = \alpha_1 \sum_{(p,t,x,c)} (c - V_{n+1}^E(p, t, x))^2$$

(terme de rappel vers les quadruplets (p, t, x, c) rencontrés en simulation, en espérance)

$$T2 = \alpha_2 \sum_{(p,t,cs_i)} (V_n^E(p, t, cs_i) - V_{n+1}^E(p, t, cs_i))^2$$

(terme de rappel vers l'estimation d'espérance précédente)

$$T3 = \alpha_3 \sum_{(p,t,abs_j,c)} (V_{n+1}^V(p, t, abs_j) - VaR_j)^2$$

où les valeurs abs_j sont des approximations de la Value-At-Risk obtenues sur les données (p, t, x, c) (détails d'évaluation plus bas), (terme de rappel vers les Value-At-Risk estimées)

$$T4 = \alpha_4 \sum_{(p,t,cs_i)} (V_n^V(p, t, cs_i) - V_{n+1}^V(p, t, cs_i))^2$$

(terme de rappel vers la fonction de Value-At-Risk précédente)

On cherche donc une fonction qui soit "pas trop différente" de la fonction précédente et "pas trop loin" des résultats de simulation. Les contraintes sont les suivantes :

$V_{n+1}^E(p, t, \cdot)$ est décroissante convexe,

$V_{n+1}^V(p, t, \cdot)$ est décroissante convexe,

$V_{n+1}^V \geq V_{n+1}^E$, en tout point (ce qui est, en toute rigueur, non garanti, mais très probable), contrainte couplante entre valeur-à-risque et espérance ; le problème est peu difficile et le couplage n'est donc pas problématique.

V_{n+1} est alors obtenue par combinaison linéaire de V_{n+1}^E et V_{n+1}^V .

Il s'agit donc d'un programme quadratique convexe ; le logiciel Xpress-MP, avec l'interface Mosel, est utilisé.

Le choix des paramètres $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ a été fixé comme suit : α_1 , comme α_3 , est égal à 1 divisé par le nombre de termes dans $T1$ et $T3$ respectivement, et α_2 , comme α_4 , est égal au nombre de jeux de N simulations déjà passées par le pas de temps considéré divisé par le nombre de termes dans $T2$ et $T4$ respectivement ; ainsi on a une sorte de moyenne sur tous les passages réalisés (aux satisfactions de contrainte près) : la nouvelle courbe est la moyenne de k fois

l'ancienne courbe (si elle a été renforcée k fois), une fois la nouvelle. Lorsque l'ancienne courbe est élaborée par une autre technique que l'apprentissage par renforcement, typiquement la programmation dynamique, un nombre est fixé arbitrairement comme analogue du nombre de renforcements réalisés à chaque pas de temps. Ce choix est une analogie avec la technique de sous-gradient ; toutefois, il faut bien voir que l'acteur évoluant à chaque mise-à-jour du critique, l'algorithme est une sorte d'algorithme du point fixe (on pourrait notamment être tenté de pondérer plus fortement les expériences récentes, puisque correspondant à un acteur plus "proche" de l'acteur courant). Par souci de simplicité, de réduction du nombre de paramètres, nous en resterons à l'approche ci-dessus.

3.3 Mise en œuvre pratique.

Les expérimentations en apprentissage par renforcement font appel à un grand nombre d'hyperparamètres peu évidents à fixer. On peut avoir le sentiment d'un domaine encore jeune, où les robustifications⁹ possibles ne sont pas toutes trouvées.

Parmi les problèmes souvent évoqués (voir [Bertsekas et al, 1996]), figurent les oscillations des paramètres et de la fonction objectif, et des divergences inattendues après une bonne première phase de décroissance de la fonction de coût. Ajoutons un problème lié à l'introduction de la notion de risque : la complexité en échantillon, c'est-à-dire le nombre de simulations nécessaire à un bon comportement de l'algorithme, est très important. Nous présentons ci-dessous l'algorithme mis en œuvre, puis les améliorations possibles pour lutter contre ces difficultés (que nous avons effectivement rencontrées) (3.5).

3.4 L'algorithme.

En très résumé et en excluant les améliorations diverses, l'algorithme est le suivant :

1. détermination d'une première fonction de valeur par programmation dynamique ;
2. simulations avec pour stratégie l'optimisation basée sur la fonction de valeur (ie pendant ces simulations la décision est celle qui optimise le coût prédit par la fonction de valeur) ;
3. détermination d'une nouvelle fonction de valeur, qui colle approximativement avec les résultats de simulation.
4. on retourne en 2 à moins de n'avoir plus de temps devant soi.

L'algorithme résultant est précisément le suivant et dépend des paramètres $1 \leq \textit{etendueDuRenforcement}$, $1 \leq \Delta \leq \textit{etendueDuRenforcement}$, et $0 \leq r \leq 1$ où r est le niveau de risque considéré :

⁹ Le terme possiblement peu élégant de robustification est utilisé pour "augmentation de robustesse".

Algorithme 1.1 Apprentissage par renforcement pour la gestion du risque

Effectuer une programmation dynamique pour déterminer une première fonction de valeur (valeurs de Bellman), correspondant à une optimisation en espérance.

Pour tout $debut$ de T à 1 avec un pas de $-\Delta$ **Faire**

Simuler N trajectoires sur les pas de temps de $debut$ à T : les niveaux de stock initiaux pour ces trajectoires de $debut$ à T sont répartis de manière non-uniforme car les petits niveaux de stock sont plus critiques.

Soit $L = (p_i, stock_i, pdt_i, c_i)$ la liste des quadruplets (niveau de demande, stock initial, pas de temps, coût simulé entre le pas de temps pdt et le pas de temps T) obtenus en simulation ; L comporte $N \times etendueDuRenforcement$ couples car chaque simulation fournit un coût futur pour chaque valeur de pdt entre $debut$ et $debut + etendueDuRenforcement - 1$ (compris).

Pour tout k entre 0 et $etendueDuRenforcement - 1$ **Faire**

Pour tout p état du processus stochastique à l'instant $debut + k$ **Faire**

Construire L' liste des éléments de L tels que $p_i = p$ et $pdt_i = debut + k$.

Soit $0 = a_0 < a_1 < a_2 < \dots < a_K = stock_{max}$ tels que le nombre de triplets $(stock_i, debut + k, c_i)$ dans L' tels que $stock_i \in [a_j, a_{j+1}]$ soit le même pour tout j , à 1 près.

Soit VaR_j le quantile $1-r$ des c_i pour $stock_i$ dans $[a_j, a_{j+1}]$ parmi les éléments de L' ; soit abs_j la moyenne des $stock_i$ correspondants. On dispose désormais d'une famille de (abs_j, VaR_j) pour la valeur de k et la valeur de p considérée. Placer chaque (abs_j, VaR_j) dans le terme formel T3.

Construire les termes formels T1, T2, T4.

Optimiser $T1+T2+T3+T4$ comme expliqué en 3.2 pour obtenir une nouvelle fonction de valeur.

Fin Pour

Fin Pour

Fin Pour

Simuler un grand nombre de trajectoires sur l'ensemble de la période et calculer les indicateurs de risque obtenus (espérance, courbe de Value-At-Risk).

3.5 Alternatives et compléments

La nature statistique de la méthode la rend en fait intrinsèquement parallèle certes, mais relativement instable et coûteuse en temps de calcul sans un certain nombre de précautions (à l'instar d'une méthode de Monte-Carlo). La littérature évoque un certain nombre de méthodes heuristiques, que nous avons testées, qui permettent de lutter contre les mauvais comportements parfois observés en renforcement cités précédemment :

- Pendant les simulations à partir de l'instant 0, on tire au sort, uniformément entre 0 et le stock maximal, le niveau initial du stock, alors même que le niveau initial à l'instant 0 est en fait connu et que l'on pourrait donc se passer d'un échantillonnage large. Ainsi, les valeurs de Bellman seront connues sur une plage plus importante et sont supposées plus robustes qu'avec l'unique valeur initiale. Cette adaptation a été testée avec un succès réel.
- Pendant l'apprentissage par renforcement, on peut bruyter la prise de décision. L'objectif est là aussi de robustifier l'approche en permettant aux trajectoires

de mieux recouvrir l'ensemble des valeurs possibles et de réduire le risque d'optimum local. Cette heuristique a été testée (avec des résultats acceptables) puis abandonnée, les autres améliorations ayant permis de rendre ce bruitage, peu satisfaisant intellectuellement, superflu.

- La période de temps simulée est tout d'abord très courte, sur la fin de la période d'étude, et augmente régulièrement jusqu'à atteindre la période intégrale. Ceci évite que des valeurs très mal estimées servent de base pour en calculer d'autres, ce qui peut conduire à des trajectoires très mal placées et donc à une très mauvaise estimation des valeurs de Bellman. Cette méthode sera appelée par la suite méthode **progressive**. Très concrètement, dans la section 3.4, cela signifie réduire le paramètre Δ . L'effet stabilisant est très net.

- Lorsque le coût en simulation ne diminue pas ou augmente, on fait décroître les coefficients α_1 et α_3 de la section 3.2; cela est analogue à la très classique réduction de pas de gradient lorsque la fonction de coût ne décroît pas.

- Afin d'éviter des conséquences nuisibles du resserrement des trajectoires¹⁰, on peut décider de ne modifier la fonction de valeur qu'au voisinage du début du renforcement. En effet, si l'on simule entre *debut* et T comme expliqué en section 3.4, les trajectoires au pas de temps $debut + k$ pour k grand peuvent être très proches les unes les autres et conduire à des difficultés numériques. Très concrètement, cette méthode se programme par des valeurs du paramètre *etendueDuRenforcement* (section 3.4) proches de Δ . Cela sera appelé par la suite **limitation de l'étendue du renforcement**. Ce changement a été testé avec des succès certains, en couplage avec la méthode progressive.

- On part d'une nappe acceptable de valeurs de Bellman (ici obtenue par programmation dynamique) pour améliorer grandement la vitesse de convergence. Ceci est très stabilisant et relativement peu coûteux en temps de calcul.

- La troncature des simulations à un nombre de pas de temps fini (le coût des pas de temps ultérieurs étant remplacé par son espérance estimée, évaluée à partir de la nappe de valeurs de Bellman). Cette méthode, proche de celle utilisée dans [Peret et al, 2002], est appelée **module arborescent**, et détaillée ci-dessous :

- On détermine une constante L fixe.
- Chaque simulation, réalisée à partir d'un certain pas de temps *debut*, n'est déroulée que sur L pas au maximum (moins si l'on atteint le pas de temps final auparavant) : la simulation s'arrête en $pdt_a = \min(debut + L, T)$ avec T l'index du pas de temps final.
- Le coût restant est remplacé par la valeur fournie par la fonction critique en espérance; avec V la valeur fournie comme espérance de coût futur en pdt_a , le coût estimé de la simulation à partir du pas de temps *debut* est alors $c(debut, p_c(debut)) + c(debut+1, p_c(debut+1)) + c(debut+2, p_c(debut+2)) + \dots + c(pdt_a - 1, p_c(pdt_a - 1)) + V$.

On pourra trouver une étude de cette méthode (dans le cadre d'une optimisation en espérance) dans [Peret et al, 2002]. Cette méthode est aussi à rap-

¹⁰ Le resserrement est nuisible car des trajectoires serrées conduisent à une remise en cause de la fonction de valeur par apprentissage sur des données où le bruit (lié à l'aspect stochastique des simulations) est prépondérant devant l'écart intrinsèque.

procher des couplages entre arbres minimax et apprentissage d'une fonction de valeur (par exemple TD-gammon). Les avantages sont les suivants : disparition du terme quadratique en le nombre de pas de temps dans la complexité en temps, réduction de la variance de l'estimation de $V(\cdot)$. Inconvénient : biais introduit dans le cadre de l'optimisation avec prise en compte du risque, car on remplace une distribution de coûts par le coût estimé par V .

4 Bilan

Cette section comporte des résultats pratiques sur le risque (section 4.1), une comparaison avec la programmation dynamique (section 4.2), une discussion sur la convergence de la méthode (section A), une étude de complexité (section 4.4).

4.1 Résultats numériques

Le bruit $d_2(t)$ n'est pas utilisé dans les programmations dynamiques, alors qu'il peut l'être commodément avec l'apprentissage par renforcement ; des programmations dynamiques adéquates pourraient prendre mieux en compte la présence de $d_2(t)$. Cette prise en compte est facile en apprentissage par renforcement, et indépendante de connaissances structurées sur $d_2(t)$, de simples chroniques expérimentales étant suffisantes (pas besoin de modèle de $d_2(t)$).

Résultats obtenus. Afin d'optimiser en performance au niveau des temps de calcul, l'échantillonnage de niveaux initiaux de stocks est effectué exclusivement entre le minimum et le maximum des niveaux de stocks constatés en simulation, à un bruit additif près (centré, uniforme, d'amplitude 20 % du niveau de stock).

Optimisation en espérance : le temps de calcul est d'environ un quart d'heure, avec un code peu optimisé réalisé en Xpress-mosel ([Dash, 2001]) sur un Pentium III à 1133 MHz, pour un jeu de données de 365 pas de temps d'une journée. L'espérance est nettement améliorée par rapport à une méthode par programmation dynamique (le gain moyen a été optimisé de 3.5 %), mais cela est certainement dû au fait que le bruit $d_2(t)$ n'est pas pris en compte dans la programmation dynamique. L'étude détaillée scénario par scénario montre une amélioration en particulier sur les scénarios difficiles.

Optimisation du risque : nous présentons seulement le cas de pas de temps indépendants (le modèle de Markov d_1 est dégénéré au sens où la matrice de transition a toutes ses lignes identiques). La figure 3 montre les résultats obtenus en renforcement d'une combinaison $(1 - \alpha)E + \alpha VaR$, avec $\alpha \in \{0, 0.1, 0.2, 0.4, 0.6\}$. On utilise 1600 scénarios, le coefficient attribué au risque varie en entraînant avec lui la courbe de Value-At-Risk ; et les pas de temps sont indépendants (les 5 états de demande sont équiprobables). On observe bien l'effet "potentiomètre" souhaité ; plus le coefficient associé au risque dans le critère à optimiser est grand, plus la courbe de risque décroît lentement. Le cas dépendant (probabilités de transition inégales) est moins net au sens où l'on n'a pas la même décroissance élégamment monotone du risque, mais la baisse de risque est néanmoins réelle.

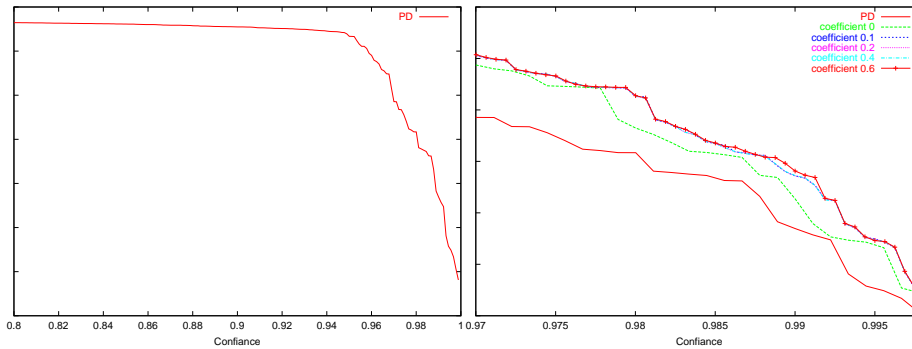


Fig. 3. Courbes de Value-At-Risk (exprimées en bénéfice, ie les "bonnes" situations sont en haut) avec différents algorithmes (programmation dynamique et renforcement avec différents paramètres de prise en compte du risque). Aucune unité n'est présentée, tant pour des raisons de confidentialité qu'en raison d'une place insuffisante pour spécifier les coûts précisément intégrés à l'étude. L'abscisse représente le quantile considéré, l'ordonnée représente le bénéfice. Lecture : au seuil de risque de 2 % (abscisse 0.98=1-risque), un coefficient 0.2 de pondération du risque conduit à un meilleur bénéfice. A gauche, un zoom sur les risques < 20% ; à droite, un zoom très rapproché. Les courbes correspondant aux valeurs 0.1, 0.2 et 0.4 sont presque confondues ; 0.6 est légèrement meilleur pour les cas les plus mauvais.

Résultats obtenus avec un module arborescent. La figure 4 (en haut) montre des résultats obtenus avec une optimisation en espérance pure, avec 300 scénarios, des pas de temps indépendants. La figure 4 (en bas) montre des résultats obtenus en optimisation du risque, avec 5000 scénarios, et des pas de temps dépendants. En conclusion : le module arborescent, très efficace en termes de temps de calcul, améliore en outre les performances en optimisation en espérance ; il dégrade légèrement les résultats en optimisation du risque par rapport au renforcement non arborescent.

4.2 L'apprentissage par renforcement et la programmation dynamique.

La programmation dynamique présente l'avantage fort d'être robuste, rapide en petite dimension, via le principe de décomposition de Bellman ([Bellman, 1955]). Le renforcement permet de gérer des cas plus complexes que la programmation dynamique. Après les diverses modifications apportées par rapport au cadre générique TD(1), i.e. méthode progressive, limitation d'étendue, et module arborescent, l'apprentissage par renforcement apparaît assez proche d'une programmation dynamique : si le pas de la méthode progressive est 1, si $L = 1$, on retrouve exactement la programmation dynamique pour peu que l'échantillonnage soit choisi de manière adéquate. Les généralisations possibles

par l'apprentissage par renforcement sont les suivantes : il permet d'inclure des contraintes de risque, il permet de traiter directement des scénarios sans construction d'un modèle de Markov ou d'un arbre de représentation, et il est totalement adaptatif : on peut ajouter de nouvelles chroniques sans faire tourner l'algorithme sur l'ensemble des chroniques. Au cours d'une utilisation en situation réelle, il permet d'optimiser le calcul des valorisations aux alentours du point courant (ce que l'on pourrait imaginer aussi en programmation dynamique). Ce point a été particulièrement développé dans [Peret et al, 2002] (avec un module arborescent) dans le cadre de la maintenance d'une constellation satellitaire. On peut noter que le Q-learning permet des généralisations supplémentaires, qui nous paraissent peu utiles dans le cadre de la gestion de stocks.

4.3 Convergence de la méthode.

La preuve de convergence de la méthode est présentée en annexe. Les grandes lignes en sont les suivantes :

- par la "méthode progressive", il suffit de montrer que l'estimation pour un pas de temps fonctionne si la fonction de valeur pour les pas de temps ultérieurs sont déjà bien estimés. Le coeur de l'argument est là : on évite via la méthode progressive (et en horizon fini) le souci associé à la démonstration de convergence pour une méthode de point fixe. Hors méthode progressive, les preuves de convergence en apprentissage par renforcement sont considérablement plus délicate.
- On se ramène donc à un problème d'apprentissage, où des quantiles et moyennes conditionnelles sont à estimer à partir d'exemples.

4.4 La complexité.

Le renforcement peut être très coûteux en temps de calcul. La complexité est, avec une méthode progressive Δ , et avec S scénarios par niveau initial de demande, de l'ordre de $(S/\Delta) \times T^2/2 \times n_e$ avec T le nombre de pas de temps et n_e le nombre d'états du processus stochastique pour un pas de temps donné. La programmation dynamique a une complexité de l'ordre $T \times n_s \times n_e^2$, avec n_s le nombre de pas de discrétisation du stock. Le rapport de complexité est donc de l'ordre de $ST/(2\Delta n_e n_s)$. En ajoutant un module arborescent, le rapport devient $SL/(\Delta n_e n_s)$, avec L la longueur des simulations; avec $L = \Delta$, qui n'introduit pas de biais en optimisation d'espérance, on arrive à $S/(n_e n_s)$, où l'on retrouve 1 si l'on choisit un échantillonnage de scénarios adéquat pour tomber sur le cas particulier de l'apprentissage par renforcement qu'est la programmation dynamique. Les performances sur le jeu à 52 pas de temps (un pas de temps est une semaine), avec 5 états à chaque pas de temps, sont désormais de l'ordre de quelques minutes en optimisation en espérance et 3 heures¹¹ (5000 scénarios) en optimisation avec prise en compte du risque, avec un code réalisé à l'aide d'un modeleur (langage haut niveau d'optimisation, sans travail fin d'optimisation ;

¹¹ au lieu de 10 sans module arborescent.

modeleur Xpress-mosel [Dash, 2001], temps de calcul sur un processus Pentium III - 1133 MHz). Avec la méthode progressive, la limitation d'étendue et la méthode arborescente, **les temps de calcul sont donc devenus acceptables** à 52 pas de temps.

La raison du surcoût en temps de calcul lors du passage de l'optimisation en espérance à l'optimisation avec risque est claire ; nous regroupons les points par paquets de 40 ou 60 pour évaluer une Value-At-Risk, ce qui implique qu'il faut soixante points pour en avoir un seul à approximer. L'utilisation d'une fonction de coût adéquate (autre que L^2) pour faire converger la courbe de Value-At-Risk, pourrait être envisagée ; par exemple, si la fonction de coût pénalise d'un coût 19 le fait d'avoir un point au dessus de la courbe V_k^v et d'un coût 1 le fait d'avoir un point au dessous de la courbe V_k^v , la courbe V_k^v trouve son équilibre au quantile à 95 % (l'équilibre est en effet atteint lorsque 19 points sont au-dessous de la courbe pour 1 point en dessus).

Les spécificités de la grande dimension pourraient être développées par l'étude du module d'extrapolation. Le lecteur est renvoyé vers [van der Vaart et al, 1996] ou [Vapnik, 1995] pour des statistiques applicables à ce problème.

5 Conclusions

Les résultats obtenus mettent en évidence la possibilité de traiter l'optimisation d'un critère comportant explicitement une notion de risque grâce à l'apprentissage par renforcement. Le renforcement, puisque capable de traiter des critères non-séparables, peut en particulier traiter des critères $(1 - \alpha)E + \alpha VaR$. Les résultats obtenus avec un coefficient non nul de gestion du risque montrent bien l'effet "potentiomètre" souhaité ; on arrive à réduire le risque (ainsi que la classique distance entre risque à 5 % et espérance).

Les trois compléments à l'apprentissage par renforcement dont l'efficacité est très clairement ressortie de l'étude sont **la méthode progressive, la limitation d'étendue, le composant arborescent** (section 3.5). Ces trois méthodes sont fortement efficaces tant en termes de temps de calcul qu'en termes de stabilité de l'évaluation.

Par ailleurs, **plus l'environnement est "incertain", plus le renforcement est stable**, car l'espace d'états est "mieux" visité qu'avec des trajectoires complètement déterministes ; des pas de temps où, de manière certaine, le niveau de stock se retrouve dans un mince segment conduit à une fonction de valeur estimée à partir d'exemples alors que le bruit est plus ample que l'écart entre les espérances conditionnelles. Cela est à rapprocher du succès du renforcement par rapport aux résolutions par arbre minimax dans le cadre des jeux à forte composante stochastique (backgammon, jeu réputé à très forte composante stochastique, par rapport aux échecs). Comme expliqué dans [Tesauro, 1989], une forte composante stochastique permet une bonne répartition des trajectoires d'état, donc réduit les problèmes numériques de convergence. L'utilisation du

renforcement est donc particulièrement souhaitée dès que les modèles prennent en compte des incertitudes variées.

Signalons enfin que divers prolongements en gestion de stocks (absence de modélisation des chroniques et non-discrétisation des aléas ; grand nombre de variables et donc non-agrégation des lacs dans le cas de la production hydro-électrique via un module d'extrapolation sophistiqué ; éventuellement non-discrétisation en temps) seraient vraisemblablement possibles grâce à l'apprentissage par renforcement ; ces points restent à explorer en gestion de stocks.

A Convergence de la méthode.

Considérons une optimisation purement en espérance, avec $\Delta = \text{etendueDuRenforcement} = 1$. Supposons nos fonctions de coûts bornées, ce qui est le cas pour notre application. Supposons que la fonction de valeur est décroissante convexe et lipschitzienne en fonction du niveau de stock, ce qui est un résultat classique pour la fonction de valeur "coût moyen" en gestion de stocks. On peut au choix supposer un module arborescent ou non, pour toute valeur $L > 0$.

Montrons que la fonction de valeur, pour une grille suffisamment fine, et pour N suffisamment grand, est arbitrairement proche de la fonction de valeur de la stratégie optimale au sens L^2 .

L'algorithme développé utilise des fonctions affines par morceaux. Ces fonctions affines ont leurs points de césure pré-déterminés ; considérons une finesse de grille donnée.

a) Examinons le comportement de l'algorithme pour une étape donnée correspondant à une valeur du paramètre *debut* (voir 3.4 : il s'agit d'un nouveau pas de la méthode progressive).

Nécessairement, $k = 0$ puisque $\text{etendueDuRenforcement} = 1$.

L'ensemble des fonctions linéaires sur chaque segment, sans contrainte de continuité, est de VC-dimension finie car sur chaque segment la restriction de cette famille de fonctions est linéaire.

On a donc une famille de fonctions de VC-dimension finie comme fonction de valeur, car c'est un sous-ensemble de l'ensemble des fonctions linéaires sur chaque segment (on a simplement ajouté des contraintes de continuité).

Les exemples fournis au module critique, pour ces paramètres *debut* et k , sont donc indépendants et identiquement distribués. La VC-dimension étant finie, on a donc convergence vers la fonction affine par morceaux la plus proche de la fonction de valeur associée à la stratégie sur les pas de temps futurs, pour la grille que l'on s'est donnée, au sens L^2 , lorsque le nombre de simulations tend vers l'infini.

Cela nous fournit donc la convergence, pour chaque valeur différente de *debut*, vers la fonction de valeur *correspondant à la stratégie utilisée pour les pas de temps qui suivent* le pas de temps *debut*.

b) Faisons maintenant l'hypothèse de récurrence selon laquelle la fonction de valeur est, asymptotiquement en la discrétisation en cs_i et en a_j , ainsi

qu'asymptotiquement en N , optimale à ϵ près à partir du pas de temps $debut+1$, pour la norme L^2 . Pour $debut = T - 1$, l'hypothèse de récurrence est exacte.

Supposons maintenant l'hypothèse de récurrence vraie pour l'étape $debut+1$, et montrons là pour $debut$, pour une valeur de ϵ . Le problème d'optimisation sous contrainte de minimisation de $T1$ ($T3$ et $T4$ n'ont pas de sens ici puisque l'on est en espérance pure; $T2$ est nul si l'on n'a pas de fonction de valeur initiale, ou $T2$ a une pondération faible si le nombre de simulations est grand) conduit, asymptotiquement en le nombre de simulations, à la fonction de valeur parmi F la plus proche de la fonction de valeur de la stratégie optimale parmi nos fonctions affines par morceau à ϵ près pour L^2 . Si la grille est assez fine, elle peut approcher arbitrairement bien la fonction de valeur; en particulier, on peut s'imposer une grille assez fine pour obtenir une précision $\epsilon + \epsilon'$ pour tout ϵ' , à la limite de N grand.

On peut donc, par récurrence, garantir que les fonctions de valeur résultant de l'optimisation de $T1$ pour $k = 0$ sont optimales à ϵ près pour l' $\epsilon > 0$ de son choix, pour la norme L^2 . Avec $etendueDuRenforcement = \Delta = 1$, ces fonctions de valeur ne sont jamais remises en cause. La convergence est donc acquise; elle est vraie pour L^2 pour la mesure de Lebesgue, et donc aussi pour la norme L^∞ puisque le coût est Lipschitzien en le stock.

On pourrait en fait alléger les hypothèses; $\Delta = 1$ est nécessaire comme peuvent le montrer des contre-exemples, mais $etendueDuRenforcement$ est plus libre.

Le cas avec prise en compte du risque, en maintenant $\Delta = etendueDuRenforcement = 1$, mais en l'absence d'un module arborescent, demande de supposer que :

- la contrainte $V_{n+1}^V \geq V_{n+1}^E$ n'empêche pas la famille de fonctions choisie d'approcher la fonction de valeur optimale; il suffit donc de supposer le niveau r de risque suffisamment fin pour que la value-at-risk excède l'espérance de coût d'une quantité $ecart > 0$ uniformément, ce qui est raisonnable pour des jeux de données non pathologiques.
- la contrainte de convexité de la Value-At-Risk ne pose pas souci, ce qui n'est par contre pas nécessairement vrai. Cette contrainte, uniquement destinée à accélérer la convergence et stabiliser l'algorithme, pourrait tout à fait être supprimée de notre méthode et nous ne la considérerons pas par la suite.

Montrons donc la convergence de la méthode dans le cadre avec risque. On suppose la value-at-risk lipschitzienne en le stock, ce qui est vrai dans notre cas. Les convergences qui suivent sont annoncées pour la convergence presque sûre.

Considérons les couples (stock initial, coût futur) pour un niveau de demande initial donné pour le pas de temps $debut$. Considérons la famille des applications $q_{a,b,s}$ de R^2 dans R paramétrées par a, b, s qui à (x, y) associe 1 si $x \in [a, b]$ et $y > s$, -1 si $x \in [a, b]$ et $y \leq s$, 0 dans les autres cas. Cette famille est de VC-dimension finie donc la moyenne des $q_{a,b,s}(x, y)$ converge uniformément vers son espérance, uniformément en a, b, s , à mesure que le nombre de couples $(x, y) =$ (stock initial, coût futur) tend vers l'infini. En particulier, chaque écart entre la

fréquence de $(y > s \text{ et } x \in [a, b])$ et la fréquence de $(y \leq s \text{ et } x \in [a, b])$ est exacte à une précision donnée près tendant vers 0. Le risk-at-value moyen parmi les x dans $[a_j, a_{j+1}]$ est donc asymptotiquement bien estimé, uniformément, puisque le nombre de x dans $[a_j, a_{j+1}]$ est par définition proportionnel à N/K . La value-at-risk est l'inverse de la risk-at-value et ces deux fonctions réciproques ont leurs dérivées bornées uniformément dans notre cas ; donc la value-at-risk converge elle aussi uniformément. Les quantiles VaR_j sur les $[a_j, a_{j+1}]$ (par décroissance de la value-at-risk en fonction du niveau de stock) sont donc comprises entre la value-at-risk en a_j et la value-at-risk en a_{j+1} .

Or les intervalles $[a_j, a_{j+1}]$ sont asymptotiquement en K ($K =$ nombre de segments $[a_j, a_{j+1}]$ plus un) de largeur tendant uniformément vers 0, presque sûrement puisque la VC-dimension des intervalles est finie. La value-at-risk étant lipschitzienne en le stock, on finit par avoir à la fois $a_{j+1} - a_j$ tendant vers 0 uniformément (en j) mais aussi l'écart entre les value-at-risk associées à a_{j+1} et a_j (uniformément en j aussi).

Alors, quel que soit ϵ , si K est assez grand, si la discrétisation de l'approximation linéaire par morceaux est suffisamment fine, si N est grand devant K (ie on considère la limite en $\min(K, 1/\max(cs_{i+1} - cs_i)) \rightarrow \infty$ de la limite en $N \rightarrow \infty$), alors le terme $T3$ ne comporte que des couples (abs_j, VaR_j) à distance de la courbe de value-at-risk majorée par ϵ en abscisse et en ordonnée et à distance 2ϵ les uns des autres. Donc nécessairement la courbe de Value-At-Risk choisie par optimisation de T_3 converge vers la véritable courbe de value-at-risk à mesure que le nombre de simulations augmente.

Il reste à voir que l'optimum de $T1$ et l'optimum de $T3$ ne se perturbent pas l'un l'autre via les contraintes définies en 3.2. La seule contrainte couplante est $V_{n+1}^V \geq V_{n+1}^E$. Puisque nous avons convergence de V_{n+1}^E et V_{n+1}^V vers l'espérance et la value-at-risk respectivement, puisque l'on a supposé $ecart > 0$, alors cette contrainte n'est pas active asymptotiquement (l'optimum est le même avec et sans contrainte).

Nous avons donc bien convergence de la fonction de valeur dans ce cadre aussi.

Signalons qu'une modification possible du module arborescent consiste à remplacer la valeur d'espérance utilisée à la fin de la simulation comme évaluation du coût par une valeur extraite par un-plus-proche-voisin (ou un autre algorithme) des coûts constatés. On perdrait ainsi le biais introduit en risque par le fait que $L < \infty$.

Références

- [Bellman, 1955] R. Bellman. Dynamic programming and multi-stage decision processes of stochastic type, in H.A. Antosiewicz (ed), Proceedings of the second symposium in linear programming, vol. 2, NBS and USAF Washington D.C., pp 229-250.
- [Bertsekas et al, 1996] D.P. Bertsekas, J.N. Tsitsiklis. Neuro-dynamic programming, Athena Scientific.
- [Boyan et al, 1993] J.A. Boyan, M.L. Littman, Packet routing in dynamically changing networks : A reinforcement learning approach. In Jack D. Cowan, Gerald

- Tesauro, and Joshua Alspector, editors, NIPS 6, pp 671–678. Morgan Kaufmann, San Francisco CA.
- [Coraluppi et al, 1999] S. Corallupi, S. Marcus, Risk-sensitive and minimax control of discrete-time, finite state Markov Decision Processes. *Automatica*, 35, 301-309.
- [Coulom, 2002] R. Coulom, Apprentissage par renforcement utilisant des réseaux de neurones, avec des applications au contrôle moteur, Thèse de doctorat, Institut National Polytechnique de Grenoble.
- [Dash, 2001] Dash Associates. Xpress-Mosel, Reference Manual. Version 1.0, September 2001.
- [Dempster et al, 2001] M.A.H. Dempster, T.W. Payne, Y. Romahi, G. Thompson, Computational learning techniques for intraday FX trading using popular technical indicators, *IEEE transactions on Neural Networks*, Special Issue on Computational Finance, 12, pp 744-754.
- [Doya, 2002] K. Doya. Reinforcement learning, Actes de CAP'02.
- [Geibel, 2001] P. Geibel, Reinforcement Learning With Bounded Risk, In : C. E. Brodley, and A. P. Danyluk, editors, "Machine Learning - Proceedings of the Eighteenth International Conference (ICML01)", pages 162-169. Morgan Kaufmann Publishers, San Francisco, CA.
- [Heger, 1994] M. Heger, Consideration of risk in reinforcement learning. Proceedings of ECML pp105-111, Morgan Kaufman.
- [Munos, 2000] R. Munos, A study of reinforcement learning in the continuous case by the means of viscosity solutions. *Machine Learning Journal*, vol. 40, n°3, 265-299.
- [Neuneier, 1996] R. Neuneier, Optimal asset allocation using adaptive dynamic programming, in *Advances in Neural Information Processings Systems*, D.S. Touretzky, M.C. Mozer, M.E. Hasjselmo, eds, vol. 8, MIT Press.
- [Neuneier, 1998] R. Neuneier, Enhancing Q-learning for optimal asset allocation, in *Advances in Neural Information Processing Systems*, M.I. Jordan, M.J. Kearns, S.A. Solla, eds, vol.10, MIT Press,
- [Neuneier et al, 1999] R. Neuneier, O. Mihatsch, Risk-sensitive reinforcement learning. MIT Press, NIPS'99.
- [Peret et al, 2002] L. Peret, F. Garcia, Une approche arborescente pour améliorer une politique issue d'un apprentissage par renforcement, actes de CAP.
- [Singh et al, 1996] S.P. Singh, D.P. Bertsekas, Reinforcement learning for Dynamic Channel Allocation in Cellular Telephone Systems, NIPS 9, MIT Press.
- [Sutton et al, 1998] R. Sutton, A.G. Barto, Reinforcement learning.
- [Sutton, 1988] R. Sutton, Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1) :9-44.
- [Tesauro, 1989] G. Tesauro. Neurogammon wins Computer Olympiad. *Neural Computation* 1, 321-323.
- [Vapnik, 1995] V.-N. Vapnik, *The Nature of Statistical Learning*, Springer.
- [van der Vaart et al, 1996] A.-W. van der Vaart, J.-A. Wellner, *Weak convergence and Empirical Processes*, Springer.
- [Watkins, 1989] C. Watkins, Learning from Delayed Reward, Ph.D thesis, Kings College, University of Cambridge.

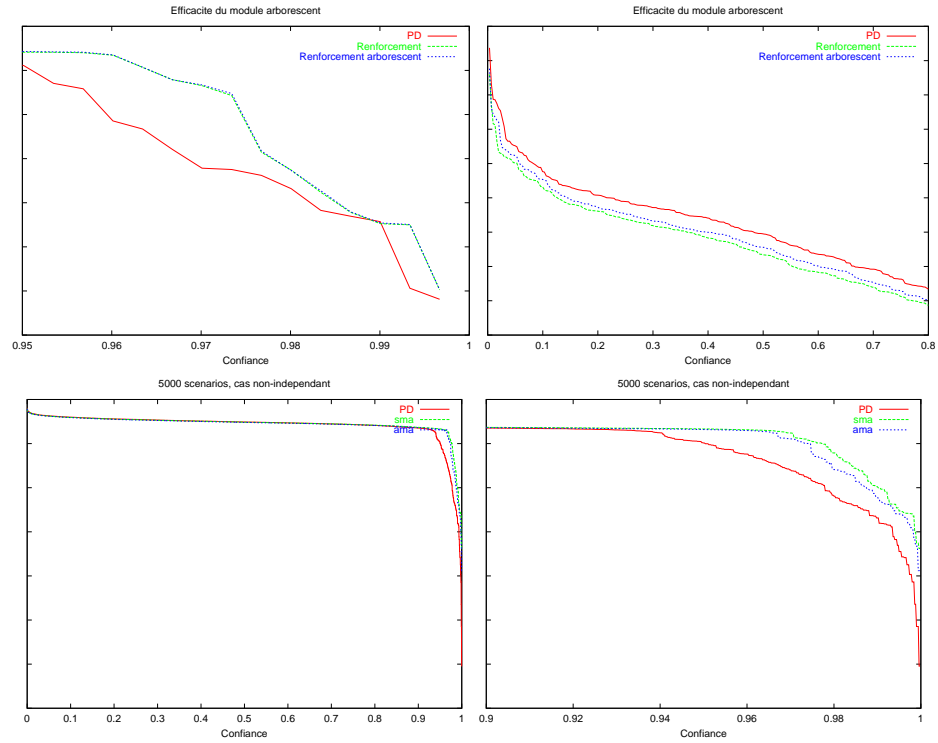


Fig. 4. En haut : courbes de Value-At-Risk (exprimées en bénéfice), dans le cas indépendant, avec module arborescent ou non, comparées avec la courbe obtenue en programmation dynamique ; en haut à gauche, "renforcement" et "renforcement arborescent" sont confondus. Le module arborescent a donné les meilleurs résultats en espérance (l'absence d'unités sur l'axe des ordonnées est trompeur ; l'écart semble graphiquement en faveur de la programmation dynamique, mais en fait les niveaux de risque 20 à 80 % correspondent à des écarts très faibles alors que les niveaux de risque 0 à 5 % correspondent à des écarts très forts, d'où une supériorité de la méthode par apprentissage par renforcement. Ici $L = 5$. En bas : courbes de Value-At-Risk (exprimées en bénéfice), dans le cas non-indépendant, avec module arborescent ou non (sma=sans module arborescent, ama=avec module arborescent), comparées avec le courbe obtenue en programmation dynamique. Afin de réduire le biais introduit par la troncature des simulations, on a choisi $L = 12$ pour le module arborescent. On constate néanmoins une légère dégradation des résultats par rapport au cadre sans module arborescent. Le temps de calcul est passé de 10h à 3h20.