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

René Aïd[1], Vincent Grellier[1], Arnaud Renaud[2], Olivier Teytaud[2]

[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, la prise en compte du risque par une mesure de type Value-At-Risk rend le problème non-séparable ; 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 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.

Une solution classique pour optimiser le choix d'un acteur lorsque le critère est séparable 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. Parmi les indicateurs statistiques pouvant être utilisés pour caractériser là une variable aléatoire C représentant un coût, on peut citer : l'**espérance** $E(C)$ (i.e. la

moyenne), la **variance** (elle est égale à l'espérance de $(C - E(C))^2$), la **demi-variance** - 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 mauvaise aussi bien une stratégie potentiellement économe qu'une stratégie potentiellement très coûteuse), la **probabilité de chute** (shortfall probability, ou Risk At Value) qui 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**), égale, pour un seuil de risque r donné, à C' minimal tel que $P(C > C') \leq r$. Naturellement, 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 altérer la stratégie de manière à optimiser un compromis espérance/risque du type $(1 - \alpha)E + \alpha VaR$, non-séparable.

Cet article est structuré comme suit : la section 2 présente la problématique, 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 méthode

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 l'on souhaite rapprocher d'un ensemble de "trajectoires" donné. Concrètement, on dispose d'un processus inconnu $p(t)$ (à temps discret ici, entre 1 et T), et d'un système $x_{t+1} = f(x_t, u(t, p(t), x_t), p(t))$, et l'on souhaite déterminer une stratégie (on dit aussi un contrôleur) $u(.,.,.)$ tel que l'espérance de $c_1(x_1, u_1) + c_2(x_2, u_2) + \cdots + c_T(x_T, u_T)$ soit minimale (u_i désigne $u(t, p(t), x_t)$) ; on dit alors que $u()$ est un contrôleur optimal en espérance. Dans l'esprit des méthodes "minimax" ou plus précisément du principe 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 $c_t(x_t, u_t) + c_{t+1}(x_{t+1}, u_{t+1}) + \cdots + c_T(x_T, u_T)$ pour une stratégie optimale $u()$ (on montre que $V()$ ne dépend pas de la stratégie optimale $u()$), et ensuite à effectuer à chaque pas de temps $u(t, p(t), x_t) = argmin_u c_t(x_t, u) + V(t + 1, p(t + 1), x_{t+1})$. La fonction $V(.,.,.)$ 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 $V(.,.,.)$ effectivement égal à une espérance, d'une part, mais aussi 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. Concrètement : la demande $d(t)$ est le produit d'un processus de Markov $d_1(t)$ et d'un bruit indépendant $d_2(t)$: $d(t) = d_1(t)d_2(t)$, la production $p_c(t)$ des cen-

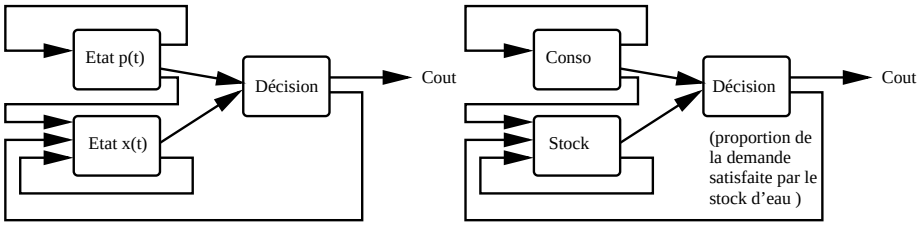

FIG. 1 – Cadre général de l'optimisation dynamique et cadre particulier ici développé. Les contraintes sont de produire autant d'électricité qu'il y a de demande. L'électricité peut-être produite à partir du stock d'eau (hydroélectricité) ou à partir de groupes dits thermiques (qui peuvent en fait modéliser aussi bien de réels groupes thermiques que l'accès à un marché). Produire de l'électricité dans les groupes thermiques à un instant donné conduit à un coût total fonction *convexe* de la quantité à produire. D'autre part, 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 ; intuitivement il s'agit d'utiliser de manière régulière (en raison de la convexité du coût de la production thermique) la production thermique, par un usage raisonné du stock d'eau, pour réduire le coût total (il faut noter toutefois que les pas de temps ne sont pas identiques, il ne s'agit là que d'une illustration intuitive). 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 (sous réserve qu'il ne soit pas à vide !) 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 en cas de stock nul on va devoir alors utiliser des groupes thermiques coûteux).

trales thermiques est comprises entre $p_{min}(t)$ et $p_{max}(t)$, la production $p_h(t)$ des centrales hydrauliques est comprise entre 0 et p_h^{max}, la contrainte couplante de demande est $p_c(t) + p_h(t) + d_-(t) = d(t)$, où $p_c(t)$ est la production des centrales thermiques, $p_h(t)$ est la production hydraulique, $d_-(t)$ est la défaillance, la production hydraulique doit faire en sorte de vérifier les contraintes suivantes : $stock(t) \geq 0$ et $stock(t) \leq stock_{max}(t)$. 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 (notée comme une énergie), $dev(t)$ est du déversement. La fonction de coût c est la somme sur les instants t des $c(t, p_c(t))$ où $c(t, .)$ 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 x_t contient $d_1(t)$, $stock(t)$. La stratégie optimale $u()$ détermine $p_c(t)$ et $p_h(t)$. Les programmations dynamiques utilisées pour comparaison travaillent sur ce même état x_t.

2.2 Méthodes d'apprentissage par renforcement

2.2.1 Présentation générale du cadre acteur-critique.

Les méthodes de type acteur-critique 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 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 une application qui à une situation associe un coût à venir (moyen, ou moyen altéré par une notion de risque), et l'acteur est l'algorithme qui prend la décision (qui choisit la décision minimisant la somme du coût instantané et du coût à venir). L'idée générale des méthodes acteur-critique (voir (Bertsekas et al, 1996)) est la suivante (illustrée sur la figure 2, gauche) :

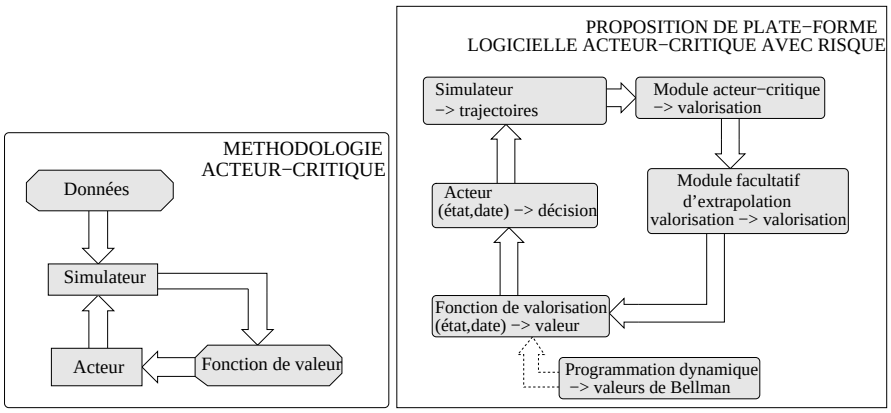

FIG. 2 – A gauche : méthode générale acteur-critique. A droite : architecture choisie.

- On dispose d'un acteur initial.
- Etape critique : on évalue, pour chaque état, par simulation avec l'acteur courant comme stratégie, l'espérance du coût de gestion à partir de cet état. Cette évaluation est appelée "fonction de valeur".
- On modifie l'acteur de manière à ce qu'il optimise la fonction de valeur (si le module acteur est en fait purement une optimisation de la fonction de valeur, on parle de méthode purement critique ; si l'acteur est lui-même une application calibrée à partir d'exemples, 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).
– On retourne à l'étape "critique".

2.2.2 Introduction du risque en apprentissage par renforcement.

Les méthodes acteur-critique (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.

2.2.3 Comparaison avec la programmation dynamique classique.

On peut comparer ces méthodes acteur-critique 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 algorithmes de gradient stochastique coûteux 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 l'acteur, pas de la descente de gradient, 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 de type extrapolation élaborant une fonction V 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)). Il est possible d'utiliser en outre, pour stabiliser l'algorithme, un acteur minimisant non pas la fonction de valeur à un pas de temps à partir de l'état courant, mais à plusieurs pas, et ce à la fois pendant la phase de calcul de la fonction de valeur et pendant l'utilisation réelle. Cette technique a été utilisée expérimentalement avec succès sur un problème vaste (Peret et al, 2002).
• **Le renforcement peut commodément travailler sur de simples chroniques et non un modèle stochastique**, alors que les techniques basées sur l'optimisation stochastique de chroniques arborescentes par programmation dynamique supposent un modèle explicite. Il permet en outre d'ajouter de nouvelles chroniques à un modèle sans partir de 0 ; cette propriété peut en particulier être utile lorsque l'on suppose l'existence de pannes pour l'allocation de canaux (voir (Bertsekas et al, 1996; Singh et al, 1996)) ou

le routage ((Boyan et al, 1993)).
• **Le renforcement permet naturellement de prendre en compte directement 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**.
• **Les méthodes par renforcement s'accommodent plus facilement de données non-discrétisées** (voire non-discrétisées en temps, voir par exemple (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.

3 Algorithmes retenus et mise en œuvre

3.1 Choix algorithmiques et méthodologiques.

Les méthodes par renforcement incluent le Q-learning et l'apprentissage $TD(\lambda)$ sur la fonction de valeur. Le Q-learning consiste en l'évaluation, non pas exactement de la fonction de valeur V, mais de $Q(p, x, u)$ l'espérance du coût de gestion *sachant que la décision u est prise* et que l'on est dans l'état x et que le processus est à la valeur p (Q dépend donc de u). La fonction Q que l'on cherche à approcher inclut donc en quelque sorte le comportement du système à étudier ; en terminologie de recherche opérationnelle, on dirait que la commande fait partie de l'espace d'état. Le Q-learning est donc principalement reconnu pour sa capacité à traiter des problèmes dépourvus de modélisation explicite. Ici, on dispose d'un simulateur ; aussi, une forme d'apprentissage par renforcement permettant d'en tirer parti sera préférée : $TD(\lambda)$**, sans Q-learning**. $TD(\lambda)$ 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 $[0, 1]$. **Nous utiliserons** $TD(1)$**, qui permet l'introduction commode de notions de risque**. L'idée sera donc d'approximer $V(., t, .)$ à l'aide des coûts constatés $c_t(x_t) + c_{t+1}(x_{t+1}) + \cdots + c_T(x_T)$. Nous procéderons par **mises 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" les paramètres peuvent converger sans qu'il y ait convergence de l'acteur). **L'introduction du risque sera faite par l'intermédiaire de la Value-At-Risk**. Le module d'extrapolation 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), mais en outre on se ramène, comme on le détaillera plus loin, à un problème d'optimisation quadratique avec contraintes linéaires. Nous comparerons les méthodes

avec et sans module arborescent (voir section (4.1.2)).

La structure choisie est décrite en figure 2 (droite). Le couplage renforcement/optimisation stochastique permettra de tirer parti de la stabilité de la programmation dynamique et de la capacité du renforcement à transformer une gestion optimale en espérance (fournie approximativement au moins par l'optimisation stochastique) en une stratégie comportant une notion de risque (après renforcement lors d'une phase de simulation). Le module d'extrapolation est ici une extrapolation par fonctions convexes, sans critères de régularisation ; de tels critères de régularisation sont nécessaire lorsque le nombre de variables d'états (à valoriser) devient important, notamment au niveau des variables continues ; comme dans le cadre d'une programmation dynamique, on conçoit bien que si l'état comporte 12 variables continues, il est nécessaire de fortement régulariser la fonction de Bellman, pour que des contraintes fortes de structures permettent à quelques valeurs seulement de produire une courbe de Bellman crédible en dimension 12. 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 du renforcement. Le module acteur-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 de l'acteur (le nombre de simulations entre chaque mise à jour ; suivant une terminologie usuelle, le "degré d'optimisme" de la méthode), le pas de descente de gradient 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 les valeurs de Bellman en cours de simulation ou une fois une (ou plusieurs) simulation entière réalisée ?

3.2 Le composant critique.

Le composant critique doit être capable de fournir une représentation par une fonction affine par morceaux de la fonction de valeur $V(p,t,x)$ choisie, pour un acteur donnée. Pour nous, x est un niveau de stock. Pour cela :

- On effectue des simulations.
- On obtient des "exemples" (p,t,x,c), où t est une date, p un état, x un stock, c un coût à venir obtenu sur une simulation. La méthode utilisée est dite "every-visit",

i.e. chaque passage d'une simulation par un triplet (p, t, x) donne lieu à la mise en mémoire d'un coût à venir c. La base obtenue est donc de longueur égale au produit du nombre de simulations par le nombre de pas de temps.
- On approxime l'espérance par des lignes brisées (une par pas de temps).
- On approxime la Value-At-Risk par des lignes brisées (une par pas de temps).
- 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. Soit V_k^E et V_k^V, fonctions estimations à l'étape k de l'espérance et de la Value-At-Risk du coût à venir. $V_{n+1}^E(p, t, ., c)$ et $V_{n+1}^V(p, t, ., c)$ sont les fonctions lignes-brisées (points de césure fixes) qui minimisent le problème somme des termes $T1, T2, T3, T4$:

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

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

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

où les x_i sont une discrétisation du stock et les valeurs $Q(.)$ sont des approximations de la Value-At-Risk obtenues sur les données (p, t, x, c),

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

pour les x_i une discrétisation du niveau de stock. Les contraintes sont les suivantes :
$V_k^E(p, t, .)$ est décroissante convexe,
$V_k^V(p, t, .)$ est décroissante convexe,
$V_k^V \geq V_k^E$, en tout point.

V_k est alors obtenue par combinaison linéaire de V_k^E et V_k^V. Les α_i évoluent au fil du renforcement. Le choix des paramètres α_1,α_2,α_3,α_4 a été fixé comme suit : α_1, comme α_3, est égal à 1, et α_2, comme α_4, est égal au nombre de simulations déjà passées par le pas de temps considéré ; ainsi on a une sorte de moyenne sur tous les passages réalisée (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 le 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 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 résultats théoriques en terme de régularisation, de pas de gradient, sont encore insuffisamment maitrisés ; on peut peut-être aussi craindre qu'il s'agisse d'un défaut intrinsèque de la méthode.

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.

La littérature évoque un grand nombre de méthodes heuristiques qui permettent de lutter contre les mauvais comportements parfois observés en renforcement cités ci-dessus :

• Pendant le renforcement, on tire au sort, uniformément entre 0 et le stock maximal, le niveau initial du stock. Ainsi, les valeurs de Bellman seront connues sur une plage plus importante et sont supposées plus robustes qu'avec l'unique valeur initiale.

• Pendant le renforcement, on peut bruiter 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.

• 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**.

• Lorsque le taux d'erreur ne diminue pas, voire augmente, on fait décroitre le pas de la descente de gradient.

• Afin d'éviter des conséquences nuisibles du resserrement des trajectoires, on peut décider de ne modifier les valeurs critiques qu'au voisinage du début du renforcement. Cela sera appelé par la suite **limitation de l'étendue du renforcement**.

• On peut aussi partir d'une nappe acceptable de valeurs de Bellman (par exemple obtenue par programmation dynamique) pour améliorer grandement la vitesse de convergence.

• La troncature des simulations à un nombre de pas de temps fini (le coût des pas de temps ultérieurs étant remplacés 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 en partie 4.1.2.

La nature statistique de la méthode la rend en fait intrinsèquement parallèle et coûteuse en temps de calcul.

3.4 L'algorithme.

L'algorithme résultant est le suivant :

• Effectuer une programmation dynamique pour déterminer une première fonction de

valeur (valeurs de Bellman), correspondant à une optimisation en espérance.
• Pour $debut$ variant de T à 1 avec un pas de Δ :
• • Simuler N trajectoires sur les pas de temps de $debut$ à T : les niveaux de stock initiaux sont non-uniformes, car les petits niveaux de stock sont jugés plus importants.
• • Evaluer des Value-At-Risk au seuil de confiance de 95 % en différents points en regroupant, pour chaque pas de temps et chaque état, les niveaux de stocks visités par groupes
• • • Initialiser le terme T3 à zéro.
• • • Pour chaque pas de temps entre $debut$ et $debut + etendueDuRenforcement$:
• • • • Regrouper les niveaux de stocks visités par paquets de taille prédéterminée.
• • • • Placer pour chaque paquet un point dans le terme T3, avec s_i le milieu du paquet, et pour Value-At-Risk le quantile correspondant du paquet.
• • Mettre à jour la fonction de valeur (voir section (3.2)) :
• • • Construire T1, T2, T4.
• • • Optimiser.
• Simuler N trajectoires sur l'ensemble de la période.

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 (section 4.3), une étude de complexité (section 4.4).

4.1 Résultats numériques

Le bruit $d_2(t)$ n'est pas utilisé dans les programmations dynamiques, ce qui favorise le renforcement (des programmations dynamiques adéquates pourraient prendre en compte $d_2(t)$) ; soulignons toutefois que cette prise en compte est plus facile en renforcement, et indépendante de connaissances structurées sur $d_2(t)$, de simples chroniques expérimentales étant suffisantes (pas besoin de modèle).

4.1.1 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. L'espérance est nettement améliorée (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

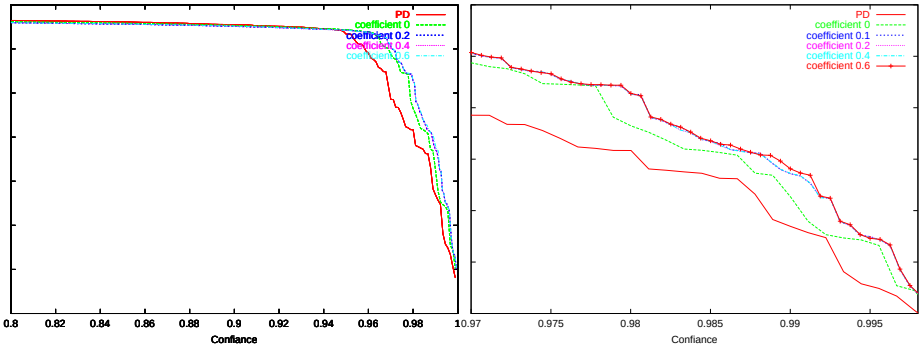

FIG. 3 – Courbes de Value-At-Risk (exprimées en bénéfice) avec différents algorithmes (programmation dynamique et renforcement avec différents paramètres de prise en compte du risque). A gauche, un zoom sur les risques $< 20\%$; à droite, un zoom très rapproché. Les courbes correspondant aux valeurs 0.1, 0.2 et 0.4 sont presque confondues ; 0.6 est légèrement meilleur pour les petits niveaux de risque.

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 décroit lentement. Le cas dépendant 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.

4.1.2 Résultats obtenus avec un module arborescent.

Cette section s'intéresse à une modification de l'algorithme comme suit :

- On détermine une constante L fixe.
- Chaque simulation, réalisée à partir d'un certain pas de temps p, 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 $p_a = min(p+L, p_f)$ avec p_f 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 de Bellman en p_f et $c(t)$ le coût au pas de temps t, le coût global de la simulation est $c(p) + c(p+1) + \cdots + c(p_a) + V$.

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 rapprocher des couplages entre arbres minimax et apprentissage d'une fonction de valeurs (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 $V(.)$. Inconvénient : biais introduit dans le cadre de l'optimisation avec prise en compte du risque. La figure 4 (en haut) montre des résultats obtenus avec une optimi-

sation 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 est très efficace en termes de temps de calcul ; il 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, 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, **le renforcement apparait 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 adéquatement. Les généralisations possibles par le 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.

En espérance, le choix $K = 1$ fait que la méthode est simplement, T fois, une approximation de fonction de fonction de valeur pour un pas de temps donné. Si la fonction de valeur est juste entre les pas de temps t et T, à ϵ près, alors la stratégie est optimale, à $(T - t)\epsilon$ près, entre t et T. Elle devient donc optimale à $(T - t)\epsilon + \epsilon'$ près, avec ϵ' l'erreur d'approximation de la fonction de valeur, qui décroit vers 0 avec le nombre d'exemples avec probabilité 1 car la famille de fonctions est uniformément lischitzienne. Par récurrence, pour tout ϵ'', à la limite d'un grand nombre d'exemples, la stratégie obtenue est optimale à ϵ'' près (sous réserve que la famille de fonctions choisie permette d'approximer la fonction de valeurs, ce qui demande dans notre cas de faire tendre vers l'infini le nombre de points de discrétisation x_i).

Avec prise en compte du risque, il faut pour avoir convergence que l'estimation de la fonction de valeurs converge ; cela est le cas seulement sans module arborescent. Sans module arborescent, les simulations sont complètes et donc permettent bien d'avoir une estimation de la quantité estimée. Sous réserve que les hypothèses structurantes que

nous avons faites sur le compromis espérance/value-at-risk soient justes (on pourrait se passer de ces hypothèses, uniquement destinées à améliorer la vitesse de convergence de l'algorithme, en augmentant le nombre de scénarios, le caractère Glivenko-Cantelli de la famille de fonctions étant garanti, dans notre cas, par le fait que la dérivée de la fonction de valeur est nécessairement bornée), la value-at-risk (qui est un quantile d'une variable aléatoire bornée) comme l'espérance convergent. Leur somme pondérée converge donc.

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 $L < \infty$.

4.4 La complexité.

Le renforcement peut être très coûteux en temps de calcul. La complexité est, avec une méthode progressive augmentant le nombre de pas de temps de K en K, et avec S scénarios par état initial possible, de l'ordre de $(S/K) \times T^2/2 \times n_e$ avec T le nombre de pas de temps et n_e le nombre d'états. 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 $ST/(2Kn_en_s)$. En ajoutant un module arborescent, le rapport devient $SL/(Kn_en_s)$, avec L la longueur des simulations ; avec $L = K$, qui n'introduit pas de biais en optimisation d'espérance, on arrive à $S/(n_en_s)$, où l'on retrouve 1 si l'on choisit un échantillonnage de scénarios adéquat pour tomber sur le cas particulier du 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, sont désormais de l'ordre de quelques minutes en optimisation en espérance et 3 heures (5000 scénarios) en optimisation avec prise en compte du risque, avec un code réalisé en modeleur et sans travail fin d'optimisation (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'un module d'extrapolation, combiné à une fonction de coût adéquate 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 cela.

5 Conclusions

Les résultats obtenus mettent en évidence la possibilité de traiter de l'optimisation d'un critère comportant explicitement une notion de risque grâce au renforcement. Le renforcement, puisque capable de traiter des critères non-séparables, peut en particulier traiter des critères $(1-\alpha)E + \alpha VaR$.

Les trois options majeures du renforcement dont l'efficacité est très clairement ressortie de l'étude sont **la méthode progressive, la limitation d'étendue, le composant arborescent**. Ces trois méthodes sont fortement efficaces tant en termes de temps de calcul qu'en termes de stabilité de l'évaluation.

Le renforcement a été plus efficace que la programmation dynamique, et plus coûteux en temps, en optimisation de l'espérance. Les résultats obtenus avec un coefficient non nul de gestion du risque montrent bien l'effet "potentiomètre" souhaité ; on arrive à réduire le risque et la distance entre risque à 5 % et espérance. Par ailleurs, **plus l'environnement est incertain, plus le renforcement est stable**, car l'espace d'états est mieux visité. 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 ; notons aussi que l'on peut traiter en renforcement de simples chroniques au lieu d'un modèle de Markov. Signalons aussi que divers prolongements (non-discrétisation, grand nombre de variables et donc non-agrégation des lacs dans le cas de la production hydro-électrique, éventuellement non-discrétisation en temps) seraient possibles en se basant sur l'utilisation d'un module d'extrapolation.

Références

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, 1955.

D.P. Bertsekas, J.N. Tsitsiklis. Neuro-dynamic programming, Athena Scientific, 1996.

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, 1993.

S. Corallupi, S. Marcus, Risk-sensitive and minimax control of discrete-time, finite state Markov Decision Processes. Automatica, 35, 301-309, 1999.

R. Coulom, Apprentissage par renforcement utilisant des réseaux de neurones, avec des applications au contrôle moteur, Thèse de doctorat, Institut National Polytechnique de Grenoble, 2002.

Dash Associates. Xpress-Mosel, Reference Manual. Version 1.0, September 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, 2001.

K. Doya. Reinforcement learning, Conférence invitée CAP 2002.

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, 2001.

M. Heger, Consideration of risk in reinforcement learning. Proceedings of ECML pp105-111, Morgan Kaufman, 1994.

R. Munos, A study of reinforcement learning in the continuous case by the means of viscosity solutions. To appear in Machine Learning Journal, 1-37.

R. Neuneir, 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, 1996.

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, 1998.

R. Neuneier, O. Mihatsch, Risk-sensitive reinforcement learning. MIT Press, NIPS 1999.

L. Peret, F. Garcia, Une approche arborescente pour améliorer une politique issue d'un apprentissage par renforcement, actes de CAP 2002.

S.P. Singh, D.P. Bertsekas, Reinforcement learning for Dynamic Channel Allocation in Cellular Telephone Systems, NIPS 9, MIT Press, 1996.

R.S. Sutton, A.G. Barto, Reinforcement learning, 1998.

G. Tesauro. Neurogammon wins Computer Olympiad. Neural Computation 1, 321-323, 1989.

V.-N. Vapnik, The Nature of Statistical Learning, Springer, 1996.

A.-W. van der Vaart, J.-A. Wellner, Weak convergence and Empirical Processes, Springer, 1996.

C. Watkins, Learning from Delayed Reward, Ph.D thesis, Kings College, University of Cambridge, 1989.

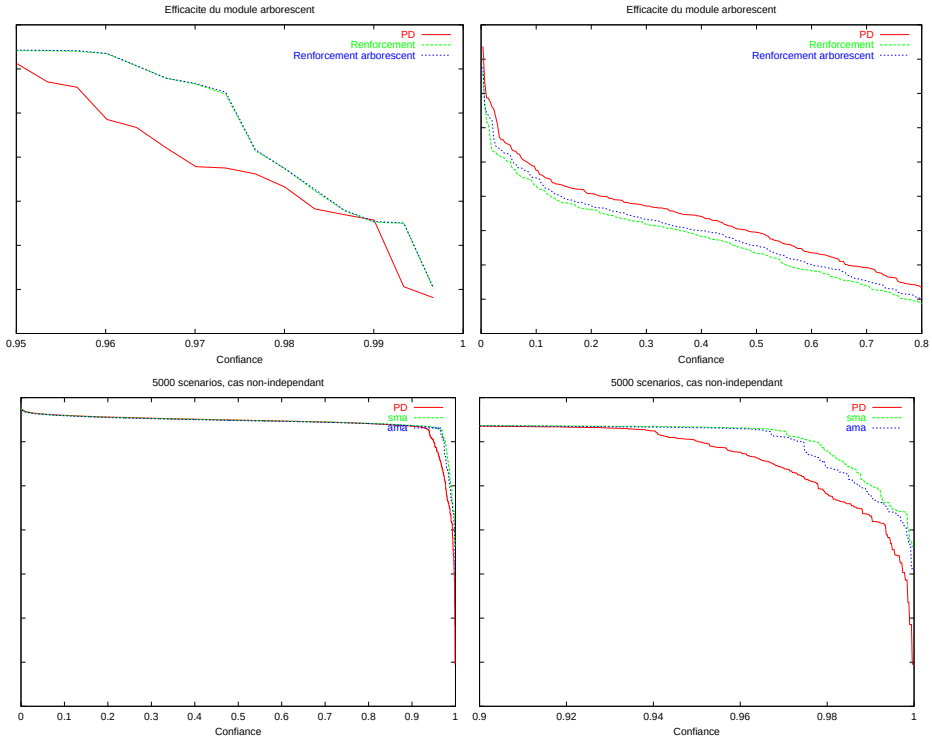

FIG. 4 – En haut : courbes de Value-At-Risk (exprimées en bénéfice), dans le cas indépendant, avec module arborescent ou non, comparées avec la courbe obtenue en programmation dynamique. Le module arborescent a donné les meilleurs résultats en espérance. Ici $L = 5$. En bas : courbes de Value-At-Risk (exprimées en bénéfice), dans le cas non-indépendant, avec module arborescent ou non, comparées avec le courbe obtenue en programmation dynamique. Afin de réduire le biais introduis 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.