Exploration prudente : une approche par méthode de Monte-Carlo arborescente contrainte

Nicolas Galichet, Michèle Sebag

Exploration prudente : une approche par méthode de Monte-Carlo arborescente contrainte

N. Galichet M. Sebag

TAO, CNRS – INRIA – LRI
Université Paris Sud, F-91405 Orsay
Nicolas.Galichet@lri.fr, Michele.Sebag@lri.fr

Résumé

En robotique autonome, nous souhaitons permettre à l'agent d'explorer son environnement afin d'y effectuer les tâches désirées. Cette exploration autonome pose le problème de la sécurité de l'agent évoluant dans un environnement potentiellement dangereux. Cet article présente, dans un contexte d'apprentissage par renforcement, une implémentation d'exploration prudente par méthode de Monte-Carlo contrainte nommée Educated MCTS. Cette approche maintient à jour parallèlement à l'exploration un modèle permettant de se restreindre aux états proches de ceux connus et supposés sûrs. Les résultats expérimentaux montrent que Educated MCTS permet une amélioration significative du compromis exploration-sécurité.

Mots Clef

robotique autonome, sécurité, apprentissage par renforcement, problème de décision markovien, PDM, machine à vastes marges monoclasse, SVM, Monte-Carlo, MCTS, Upper Confidence Bound, UCB, Upper Confidence Bound applied to Tree, UCT, exploration prudente

Abstract

In an autonomous robotic framework, an agent has to explore an unknown and possibly harmful environnement. This autonomous exploration raises the problem of the robot's safety. This article introduces an algorithm implementing a cautious exploration in a reinforcement learning context. The algorithm, entitled Educated MCTS, is designed as a constrained Monte-Carlo Tree Search method. The constraint is expressed through a model characterizing the known states and forces the exploration to consider only points close to these states. Experimental results show that Educated MCTS allows a significative improvement of the exploration-safety tradeoff

Keywords

autonomous robotic, safety, reinforcement learning, markov decision process, MDP, One-Class SVM, MCTS, Monte-Carlo, Upper Confidence Bound, UCB, Upper Confidence Bound applied to Tree, UCT, cautious exploration

1 Introduction

1.1 Enjeux

La robotique autonome est la discipline scientifique s'intéressant à la conception et à l'étude d'agents capables d'agir sans aide humaine extérieure continue. Une de ses principales hypothèses de travail est que l'environnement dans lequel l'agent sera amené à évoluer ne peut être intégralement caractérisé à l'étape de conception. Ainsi, nous cherchons dans ce cadre à permettre au robot d'acquérir par lui-même des informations relatives à son environnement par le biais d'une exploration autonome.

Cet objectif soulève de manière immédiate le problème de la sécurité de l'agent, puisque, comme l'environnement ne peut être supposé connu *a priori*, nous ne pouvons garantir qu'il est sans danger pour le robot.

D'une manière complémentaire, l'ajout d'un critère de sécurité en exploration peut amener une nouvelle méthodologie de conception de contrôleurs robotiques. En effet, un procédé répandu, notamment en robotique évolutionnaire [16], est de faire évoluer les contrôleurs en simulation et d'appliquer le résultat de ces simulations sur robots réels. Cette méthode présente l'avantage majeur d'être peu couteuse en temps. Cependant, l'expérience montre que les résultats obtenus en simulation sont difficilement reproductibles sur robots réels (*Reality gap*) [13]. L'ajout d'un critère de sécurité faciliterait ici la mise au point de contrôleurs directement *in situ*, en fournissant des garanties quant à la sécurité de l'agent.

L'objet de cet article est de présenter une approche d'exploration prudente dans le cadre de l'apprentissage par renforcement [1]. L'algorithme résultant, nommé Educated MCTS, base son exploration sur une méthode arborescente de type Monte-Carlo. Nous y intégrons une contrainte de sécurité par l'introduction d'un modèle que nous nommerons **modèle du** DEJA-VU et caractérisant l'espace connu. Ce modèle mis à jour parallèlement à l'exploration contraint celle-ci à ne considérer que des actions amenant à des états proches de ceux déjà visités et supposés sûrs.

1.2 Apprentissage par renforcement

L'apprentissage par renforcement [1] est un formalisme considérant un agent placé dans un environnement et dont le but est de maximiser, tout en interagissant, un signal de récompense sur un horizon de temps fixé. Formellement, on cherche à résoudre un problème de décision markovien (PDM) $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$:

- $\mathcal{S}$ espace d'états, $\mathcal{A}$ espace d'actions
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow [0, 1]$ probabilités de transitions et $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow \mathbb{R}$ récompenses.
- $\gamma \in [0, 1]$ facteur d'actualisation.

Résoudre un problème d'apprentissage par renforcement consiste alors à déterminer une politique π optimale, i.e. à déterminer selon l'état dans lequel l'agent se trouve la ou les action(s) à choisir pour maximiser la récompense cumulée sur T pas de temps. Formellement, on veut déterminer $\pi : \mathcal{S} \longrightarrow \mathcal{A}$ politique déterministe maximisant $V^\pi(s) = \mathbb{E}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t, s_{t+1}) | s_0 = s, a_t = \pi(s_t)]$.

1.3 Travaux antérieurs

La notion de prudence dans ce cadre n'a été considérée, à notre connaissance, que par les travaux de Fonteneau et al. [2, 3]. Ces travaux se placent dans le cas de l'apprentissage par renforcement en mode *batch*, ce qui signifie que les seules informations pour établir une politique sont des observations de la dynamique du système (c-à-d un ensemble de quadruplets de la forme (état, action, récompense obtenue, état résultant)).

Sous certaines hypothèses (transitions et récompenses déterministes et lipschitziennes [1]), il y est dérivé un algorithme de généralisation (*Cautious Generalization Reinforcement Learning*, CGRL) qui, étant donné un état initial et un échantilllon de trajectoires du système, retourne une politique maximisant une borne inférieure sur la récompense cumulée dans l'environnement le plus défavorable que l'on puisse considérer au vu des observations. Par cette conception, CGRL a montré de manière expérimentale des propriétés de prudence, en évitant l'exploration d'actions ou d'états inconnus.

CGRL constitue donc une première approche d'exploration prudente. Cependant, elle souffre de certains inconvénients [2, 3] :

- Un échantillon de trajectoires du système est nécessaire au préalable.
- L'algorithme repose sur l'hypothèse forte que fonctions de transitions et de récompenses sont lipschitziennes, et que des bornes supérieures sur les constantes de Lipschitz sont connues. Cette hypothèse ne peut pas être supposée vérifiée dans des applications intéressantes. De plus, la qualité des garanties de sécurité repose sur les valeurs des constantes de Lipschitz et sur la densité de l'échantillon disponible.
- La contrainte de sécurité n'est pas explicite. Elle est implicitement prise en compte dans le cadre lipschitzien : le danger de s'écarter des états visités est borné de manière linéaire par la distance aux paires d'état-actions connues.

Nous proposons donc une nouvelle approche basée sur une méthode de Monte-Carlo arborescente (*Monte-Carlo Tree Search*, MCTS) à laquelle nous ajouterons un critère explicite et numérique de sécurité par le moyen d'un modèle du DEJA-VU.

2 Educated MCTS

Après avoir introduit les notations et les hypothèses de travail (basées sur [2, 3]), cette section décrit les deux composantes de l'algorithme Educated MCTS, les méthodes de Monte-Carlo arborescentes et la machine à vastes marges monoclasse (*One-class Support Vector Machines*). L'algorithme complet est finalement présenté et discuté.

2.1 Hypothèses de travail

La suite de cet article considère des PDM $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ possédant les caractéristiques suivantes :

- Les espaces d'états $\mathcal{S}$ et d'actions $\mathcal{A}$ sont finis.
- Les transitions sont déterministes : pour chaque couple d'état-action $(s, a) \in \mathcal{S} \times \mathcal{A}$, il existe un unique état résultant s' tel que $P(s, a, s') = 1$.
- On associe à chaque état $s \in \mathcal{S}$ une récompense $r(s) \in \mathbb{R}$. À un couple d'état-action (s, a) est associée la récompense de l'unique état résultant.
- L'horizon de temps T est fini et $\gamma = 1$.

On désigne par **modèle du PDM** la donnée des fonctions de transitions et de récompenses. Nous supposerons ce modèle inconnu *a priori*. L'approche présentée requiert uniquement le fait que la récompense cumulée associée à une trajectoire complète soit donnée à la fin du parcours de celle-ci.

Nous ferons de plus l'hypothèse que la région dangereuse correspond à la région où la récompense instantanée est mauvaise (fortement négative dans la suite), et que la fonction de récompense est continue. Cette dernière hypothèse doit nous permettre de détecter la situation dans laquelle l'agent s'approche de la zone dangereuse. Nous supposerons donc l'espace d'états muni d'une métrique permettant de définir une notion de continuité pour r. Il est cependant intéressant de noter que, dans la mesure où Educated MCTS se fonde sur un échantillonage de trajectoires complètes, l'hypothèse plus faible que seul l'espace des trajectoires est métrique est suffisante.

L'approche proposée repose sur ces deux hypothèses : la démarche consiste à explorer l'espace de recherche en détectant graduellement les contours de la région dangereuse, pour ne pas s'y aventurer.

2.2 Exploration par méthode de Monte-Carlo arborescente

L'algorithme proposé se fonde sur l'exploration de l'espace de recherche par méthode de Monte-Carlo arborescente

1. une fonction $f : (X, \|.\|_X) \longrightarrow (Y, \|.\|_Y)$ est dite lipschitzienne s'il existe une constante C telle que $\|f(x) - f(x')\|_X \leq C\|x - x'\|_Y \forall x, y$.

(*Monte-Carlo Tree Search*, MCTS), que nous allons rappeler brièvement en renvoyant le lecteur intéressé à [4, 7] pour une présentation détaillée.

Rappelons tout d'abord le principe du bandit manchot [5]. Considérons k actions possibles, le choix de la i-ème action conduisant à une récompense de 1 avec la probabilité p_i (et à une récompense 0 sinon), avec p_i inconnu pour tout i. L'objectif du bandit manchot est de choisir à chaque pas de temps une action, telle que la récompense cumulée soit maximale. Cet objectif demande de résoudre le compromis *Exploration vs Exploitation* (EvE) : pour maximiser la récompense cumulée, le plus sûr est de choisir l'action qui s'est révélée empiriquement la meilleure (Exploitation) ; mais il faut être sûr qu'on a bien identifié la meilleure (Exploration). Notons μ_i la récompense cumulée reçue lors des n_i fois où l'action i a été choisie. Un algorithme optimal en terme de récompense cumulée, appelé *Upper Confidence Bound* [2] et proposé par Auer et al [6], consiste à choisir à chaque pas de temps l'action i^* donnée par

$$i^* = argmax\{\frac{\mu_i}{n_i} + C\sqrt{\frac{\log \sum_{j=1}^{k} n_j}{n_i}}, i = 1 \dots k\} \quad (1)$$

où C est un paramètre dépendant du problème et contrôlant le compromis EvE.

L'algorithme *Upper Confidence Tree*, qui est l'une des méthodes MCTS les mieux étudiées, étend le principe du bandit manchot et l'algorithme UCB au cas d'une séquence arborescente de choix de la manière suivante [4, 7]. La version modifiée de UCT proposée dans [7] permet de construire l'arbre de recherche graduellement. Dans cet algorithme, l'arbre, initialement réduit à son nœud racine, est parcouru un nombre N de fois ; lors de chaque parcours,

1. L'état (ou nœud) initial s_1 est la racine de l'arbre ;
2. Considérons le nœud courant s_i de l'arbre, les actions possibles en ce nœud et les récompenses cumulées associées ; le choix de l'action a_i est donné par UCB ;
3. L'état suivant s_{i+1} est déterminé à partir de s_i, a_i ;
4. Si s_{i+1} est une feuille de l'arbre :
   - une action a_{i+1} est choisie aléatoirement, conduisant au nœud s_{i+2} et la branche (a_{i+1}, s_{i+2}) est accrochée au nœud s_{i+1} (qui n'est donc plus une feuille).
   - une procédure aléatoire (généralement par tirage uniforme) est suivie pour sélectionner les actions $a_{i+2}, \dots, a_T$ jusqu'à l'horizon temporel T fixé ;
   - la récompense globale R associée au chemin $(s_1, a_1, \dots s_T, a_T)$ est calculée ;
   - la récompense associée (respectivement le compteur associé) à tous les nœuds visités de l'arbre (états et actions) est incrémentée de R (resp. de 1).
5. Sinon, i est incrémenté ; goto 2.

Après N parcours (ou simulations), l'arbre comprend au plus N nœuds [3]. Sa croissance est asymétrique, biaisée vers l'approfondissement des nœuds conduisant à une meilleure récompense. L'arbre ainsi construit définit une politique partielle : le choix de l'action en tout nœud de l'arbre est donné par l'action la plus visitée en ce nœud. Le coût d'UCT, linéaire en fonction de N, est ainsi contrôlé par l'utilisateur (algorithme *any-time*).

2.3 Critère d'admissibilité dynamique

Le deuxième composant de Educated MCTS est la caractérisation dynamique de la zone qui peut être visitée sans danger, définissant les états et les actions admissibles. Cette caractérisation se base sur les machines à vastes marges (*Support Vector Machines*, SVM). Le lecteur est supposé ici familier avec ces algorithmes et est renvoyé aux articles introductifs [8, 9] pour de plus amples détails.

L'argument de continuité de la fonction récompense (section 2.1) assure qu'au voisinage de bons états, il n'y a pas de danger. Il nous reste ainsi à caractériser la région connue et bonne.

L'approche proposée se fonde sur la machine à vastes marges monoclasse, étendant le principe des SVMs en classification [8, 9] au cas de l'estimation de densité de probabilité [11]. Formellement, étant donné un ensemble de points $\mathcal{E} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ que l'on suppose échantillonné selon une distribution $\mathcal{D}$, la machine à vastes marges monoclasse détermine le support de $\mathcal{D}$ et plus précisément l'ensemble de volume minimal contenant une proportion spécifiée des points de $\mathcal{E}$.

L'algorithme fonctionne comme une SVM classique (astuce du noyau puis séparation par un hyperplan de marge maximale) en considérant comme unique membre de la seconde classe l'origine. La solution cherchée est obtenue en résolvant le problème d'optimisation quadratique :

$$\min_{w,\xi,\rho} \frac{1}{2}\|w\|^2 - \rho + \frac{1}{\nu p}\sum_{i=1}^{p} \xi_i$$

sous contraintes

$$w^T \phi(x_i) \geq \rho - \xi_i$$
$$\xi_i \geq 0$$

avec w un vecteur normal à l'hyperplan séparateur, ρ seuil à dépasser pour être dans le support estimé, ϕ un plongement dans un espace de grande dimension et ν un paramètre contrôlant la fraction de points extérieurs au support estimé [11].

La résolution de ce programme aboutit à la détermination d'une fonction h définie par

$$h(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x)$$

2. Les auteurs proposent plusieurs variantes de UCB. Nous ne considérerons que la version présentée dans la suite de cet article

3. Notons qu'un même état peut figurer à plusieurs endroits de l'arbre ; il s'agit à proprement parler d'un DAG.

avec α_i, $i = 1, \ldots, n$ suite de coefficients affectés aux points de $\mathcal{E}$ et $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ le noyau.
Intuitivement, $h(x)$ avec $x \in \mathcal{S}$ représente la « confiance » de l'agent dans l'état x : plus $h(x)$ est élevé, plus l'état x est proche des états couramment considérés comme sûrs.

2.4 Exploration prudente

L'algorithme proposé diffère de UCT (section 2.2) à deux endroits : le choix de l'action courante, filtré par le modèle DEJA-VU d'une part ; et la construction et la mise à jour du DEJA-VU d'autre part.
Le choix d'une action a_i à effectuer dans l'état s_i (soit dans l'arbre soit dans la phase aléatoire) se limite aux actions admissibles. Une action a est admissible dans un état s si et seulement si l'unique état résultant s' ($P(s, a, s') = 1$) est admissible au sens de DEJA-VU c'est-à-dire

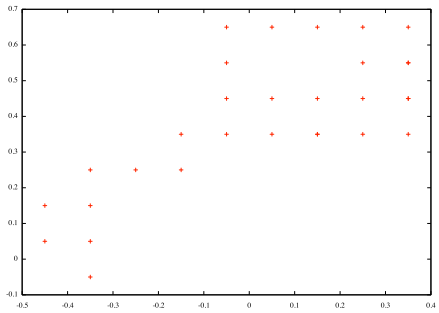
$$h(s') \geq \tau$$

(où τ est un seuil dynamiquement défini lors de la mise à jour de DEJA-VU, voir ci-dessous).
Si aucune action n'est admissible en un nœud de l'arbre, l'algorithme marque le nœud comme cul-de-sac, puis backtracke et revient récursivement à l'état s_{i-1}. Si le backtrack remonte au nœud initial de l'arbre, l'algorithme s'arrête.

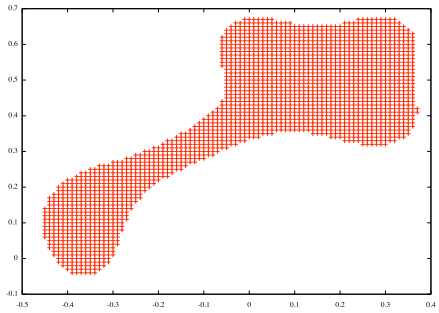
L'initialisation du modèle DEJA-VU correspond à l'amorçage du processus. On supposera que l'on dispose initialement d'une trajectoire sûre, par exemple donnée par l'expert. Les points visités lors de cette trajectoire servent à initialiser $\mathcal{E}$ et l'arbre de recherche. On pose $\tau = \min_i h(x_i) - \sigma(h(x_1), \ldots, h(x_n))$ (σ désignant la variance).
On peut associer à chacun de ces états le score

$$\frac{\mu_i}{n_i} - \frac{C}{\sqrt{n_i}} \qquad (2)$$

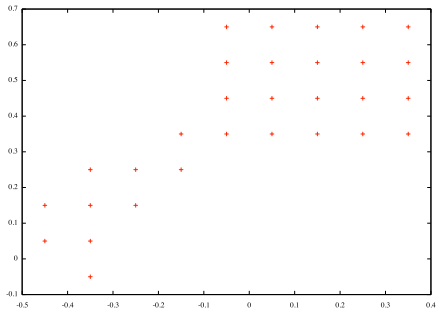
On pose m la valeur minimale de ce score pour les points de la trajectoire initiale. m servira à filtrer les points utilisés pour la mise à jour de DEJA-VU.
La figure 1 montre un exemple de trajectoire initiale, le support correspondant estimé par la SVM et les points sûrs correspondants. Les paramètres utilisés par toutes les machines à vastes marges monoclasses de cet article sont constants et donnés en annexe.
La mise à jour de DEJA-VU après chaque [4] simulation prend en entrée l'ensemble des états s_i visités depuis le lancement de l'algorithme [5]. Ces états sont ordonnés par valeur décroissante du score 2.



(a) Trajectoire initiale



(b) Support estimé



(c) Points admissibles

FIGURE 1 – Exemple d'initialisation du DEJA-VU

4. En pratique, la mise à jour est effectuée toutes les p simulations pour une raison de coût de calcul.

5. Le surcoût de mémoire entraîné par ce stockage reste raisonnable, chaque état distinct étant représenté une seule fois. Le nombre d'états stockés est donc généralement largement inférieur au nombre de nœuds de l'arbre et au pire $max(card(\mathcal{S}), T \times N)$ avec N nombre de simulations). La taille de l'arbre de recherche reste ainsi le principal goulot d'étranglement en terme d'encombrement mémoire.

Seuls les $r\%$ premiers états (où r est un paramètre spécifié par l'utilisateur) font partie de l'ensemble $\mathcal{E}$ utilisé pour apprendre DEJA-VU ; on impose de plus que chacun de ces états ait une valeur de critère 2 supérieure à m. On s'assure cependant dans tous les cas que $\mathcal{E}$ contient au moins autant de points que d'états distincts dans la trajectoire initiale.
Enfin le seuil τ est donné par le minimum entre :

1. la moyenne des $h(x_i)$ où $\mathbf{x}_i$ décrit l'ensemble des états non retenus (si $r = 1$ et que tous les états sont dans $\mathcal{E}$, τ est fixé avec la dernière valeur des $h(\mathbf{x}_i)$)
2. $h(\text{origine})$.

Le marquage des nœuds de l'arbre est réinitialisé à chaque mise à jour.

2.5 Discussion

Educated MCTS fonde l'exploration sur UCT, auquel est ajoutée une contrainte de sécurité permettant de ne considérer que des états sûrs.
UCT permet tout d'abord de bénéficier d'une approche bien fondée du compromis EvE, réglable par le choix de la constante C (Eq 1). Une connaissance du modèle (transitions et récompenses) ou d'échantillons préalables d'états-actions ne sont de plus pas nécessaires, l'algorithme opérant un échantillonnage autonome de trajectoires.
La contrainte de sécurité se met ensuite en place via une SVM à une classe. Il est essentiel de souligner ici que les points d'entraînement doivent être choisis avec précaution, afin de ne pas graduellement intégrer des états dangereux.
On propose pour cela d'évaluer la sécurité d'un état par le score 2, inspiré de celui d'UCB (Eq 1).
Le score 1 correspond à une borne supérieure, dérivée de l'inégalité de Hoeffding, sur la récompense associée à un bras [5] : on choisit le bras pouvant apporter la plus grande récompense (« optimisme face à l'inconnu »).
Le score 2 correspond lui à une borne inférieure et à une démarche pessimiste : un état est d'autant plus sûr que la pire récompense envisageable associée est élevée. Les réels r et m permettent enfin d'entraîner le modèle DEJA-VU qu'avec une proportion fixée des meilleurs états selon ce score.

3 Expériences

Cette section concerne la validation expérimentale de Educated MCTS. Nous présentons dans un premier temps le problème étudié, le protocole utilisé avant de présenter les résultats.

3.1 Problème du *puddle world*

Nous reprenons l'expérience du *puddle world* proposée par Sutton [10] et utilisée pour tester CGRL [2]. Les résultats présentés et ceux de CGRL ne sont pas directement comparables, CGRL faisant les hypothèses supplémentaires suivantes :

1. Les fonctions de transitions et de récompenses sont lipschitziennes, et nous disposons de bornes supérieures sur les constantes de Lipschitz.

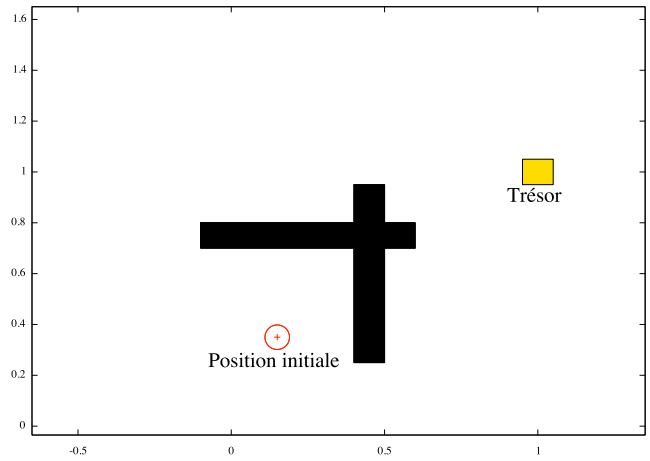


FIGURE 2 – Problème du puddle world

2. Un échantillon de trajectoires observées du système est disponible.

Le problème considère un agent situé d'une part d'un puits et cherchant à se rapprocher d'un trésor situé d'autre part (figure 2). Les spécifications du PDM associé sont données en annexe.
Les récompenses associées aux points du puits sont très mauvaises (en moyenne -1200) et bonnes sur le trésor (23). Le problème est ainsi conçu de manière à encourager la prise de risques en récompensant au mieux les trajectoires rasant le puits et atteignant ainsi rapidement le trésor ; un pas dans le puits dégradant cependant de manière irréversible le gain cumulé.

3.2 Objectif des expérimentations

Nous désirons observer les effets de la contrainte de sécurité sur le compromis efficacité de l'exploration-prise de risques. Pour cela, nous allons comparer les performances obtenues par UCT sans contrainte et par Educated MCTS avec différents paramétrages.
La performance d'un algorithme de Monte-Carlo est généralement évaluée par la récompense moyenne obtenue après un certain budget de simulations. Ce critère reste pertinent ici, mais présente l'inconvénient de considérer la performance globalement sans en discriminer les différents facteurs explicatifs.
Nous avons donc choisi de lire les résultats de nos expérimentations comme ceux d'un problème d'optimisation multi-objectifs, avec :

1. **Exploration** : le premier objectif est de trouver le trésor. Numériquement, cet objectif se traduit par la distance minimale au trésor dans la trajectoire retournée par UCT et Educated MCTS.
2. **Danger** : l'agent cherche à éviter les points du puits. Numériquement, ceci se traduit par le nombre de pas qu'une exécution de UCT ou Educated MCTS passe dans le puits au cours des N simulations.

Dans ce contexte multi-objectifs, nous comparerons deux solutions (exécutions de Educated MCTS ou UCT) selon les critères de Pareto. Nous considérons principalement la notion de **domination** de Pareto : une solution a **domine** une solution b au sens de Pareto, si a présente des valeurs plus petites sur les deux objectifs que b. On désigne par **front de Pareto** l'ensemble des solutions non-dominées.

3.3 Protocole expérimental

Les paramètres de Educated MCTS font intervenir le paramètre C d'exploration de UCT, ainsi que les paramètres de la SVM. Ces derniers ont été fixés par des expériences préliminaires et sont indiqués en annexe. Il fait également intervenir des paramètres p et r définis en section 2.4. Enfin, Educated MCTS est initialisé par une trajectoire initiale aléatoire définissant le seuil m et garantie sûre par un expert. Dans un souci de reproductibilité, la trajectoire initiale fournie par l'expert est tirée aléatoirement parmi les trajectoires assez sûres (c-à-d sans points du puits).

Nous proposons le protocole expérimental suivant :

- 25 exécutions indépendantes de Educated MCTS et UCT sans contrainte. UCT servira d'algorithme de référence.
- 1 millions de simulations pour UCT et Educated MCTS.
- $p = 10$, r varie de 0.1 à 1.
- C prend les valeurs 190, 220, 270 et 800.

Ces valeurs de C ont été choisies selon le résultat moyen qu'elles impliquaient sur UCT avec 1 million de simulations et sur 100 essais. Pour $C = 190$ et $C = 270$, nous avons obtenu les meilleurs performances en terme de récompenses moyennes et d'atteinte du trésor. Pour $C = 220$, les résultats ont montré la variance la plus forte et 800 est une valeur très grande, pour laquelle UCT obtient des résultats mauvais selon les deux objectifs (sur-exploration).

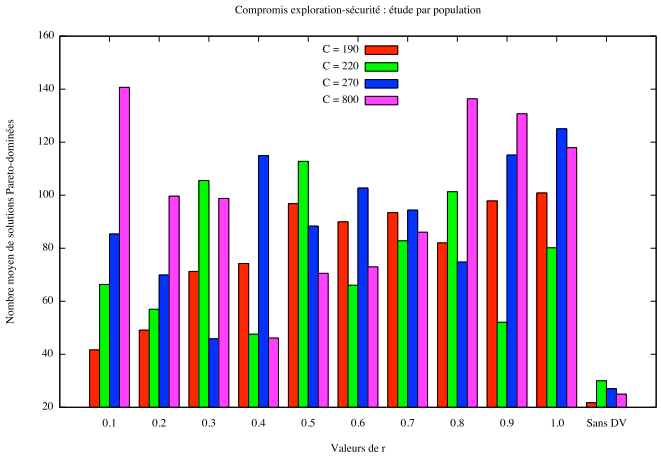
Les résultats de ces exécutions sont représentés sous deux formes. Nous disposons de 11 populations de 25 individus relatives à 10 réglages distincts de Educated MCTS et à UCT. Nous comparons chacune de ces populations à toutes les autres en comptant le nombre moyen de solutions qu'elles dominent au sens de Pareto.

Nous proposons ensuite de représenter les résultats de quelques populations pour permettre une interprétation de l'influence des paramètres et comprendre comment le front de Pareto est amélioré.

3.4 Résultats

La figure 3 présente les résultats globaux sous l'angle du compromis exploration / sécurité. Pour chaque paire (C, r), le nombre moyen de solutions qu'elle domine est indiqué (le plus haut est le mieux). La première remarque est ainsi que l'algorithme de référence UCT a de mauvaises performances. La visualisation multi-critères (Fig. 4) indique en effet que les solutions d'UCT se répartissent en deux groupes : un premier groupe de solutions, sur la droite, correspond aux trajectoires qui restent proches de la position initiale ; celles ci sont loin du trésor et dominées par toutes

FIGURE 3 – Étude du compromis exploration-sécurité. 25 exécutions sont effectuées pour chaque paire de paramètres (C, r) ; la solution (distance au trésor, nombre de pas dans le puits) de chaque exécution est comparée aux solutions de toutes les autres exécutions, et son score est donné par le nombre de solutions qu'elle domine au sens de Pareto. Pour chaque paire (C, r) est indiqué le score moyen des solutions obtenues (le plus haut, le mieux). Le dernier histogramme sur la droite correspond à l'algorithme de référence UCT.



les autres solutions. Un petit nombre de solutions arrive au trésor, après avoir passé un temps modéré dans le puits.

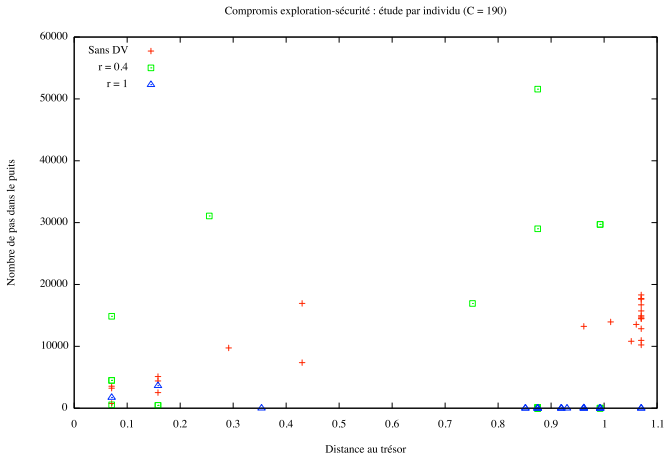
Les solutions obtenues par Educated MCTS au contraire sont significativement meilleures au sens du compromis exploration / sécurité. Les solutions obtenues par exemple par le paramétrage le plus conservateur ($r = 1$) ne prennent aucun risque et ne visitent jamais le puits ; ces solutions ne sont ainsi jamais dominées, mais en dominent beaucoup d'autres, ce qui explique la hauteur de la barre associée.

Il importe d'appréhender ce résultat avec ses limites : les solutions obtenues ne s'approchent pas plus du trésor que celles de l'algorithme de référence ; le gain réside dans le fait qu'elles découvrent le trésor *sans visiter le puits*.

L'augmentation du paramètre d'exploration C (paramètre du bandit manchot, section 2.2) améliore globalement les résultats au sens de Pareto, toutes choses égales par ailleurs. L'interprétation proposée pour ce fait est la suivante. Lorsque la valeur de C augmente (et à la limite lorsqu'elle tend vers l'infini), Educated MCTS tend vers une exploration aléatoire de l'arène sous contrainte de DEJA-VU. La qualité et la représentativité de DEJA-VU augmentent donc, ce qui garantit ensuite le fait que la contrainte issue de DEJA-VU est effective et que les trajectoires correspondantes ne visitent pas le puits. Ce dernier fait résulte en un score élevé ; cependant ce score est plus dû au critère de sécurité qu'à celui de l'exploration.

L'augmentation du paramètre r (contrôlant le support de DEJA-VU) tend également à améliorer les résultats. Quand

FIGURE 4 – Étude du compromis exploration-sécurité. Solutions obtenues pour UCT (légende croix) et Educated MCTS, avec $C = 190$ et $r = 1$ (légende triangle) ou $r = 0.4$ (légende carré). Chaque solution est représentée par sa distance au trésor en abscisse, et le nombre de pas passé dans le puits en ordonnée.



r tend vers 1, la totalité des points visités (au-dessus du seuil m issu de la trajectoire initiale) est intégrée dans l'apprentissage de DEJA-VU, ce qui garantit également une bonne représentativité de DEJA-VU et une exploration conservative.

Cette interprétation est confirmée par la figure 4. Dans un souci de lisibilité, seules les solutions obtenues pour $C = 190$ et $r = 1$ ou $r = 0.4$ sont représentées, ainsi que les solutions de l'algorithme de référence. Chaque solution est représentée dans le plan (distance min. au trésor, nombre de pas de temps de visite du puits). Les solutions obtenues pour $r = 0.4$, si elles se retrouvent en moyenne plus près du trésor que les solutions de référence, visitent intensivement le puits en encourent donc un grand danger. L'ajustement du seuil τ (section 2.4) est vraisemblablement trop bas pour empêcher une exploration risquée. En dépit du fait que la contrainte ne permet pas d'éviter le danger, elle contribue néanmoins à éviter le comportement sous-optimal qui consiste à rester près de la position initiale.

4 Conclusion et perspectives

Le but du travail présenté est de remédier aux dangers de l'apprentissage par renforcement (AR), dans le cas où l'exploration de l'espace de recherche entraîne des risques physiques. Dans le cas général, l'AR fait intervenir deux objectifs : l'identification des zones prometteuses, à visiter, et l'identification des zones non prometteuses ou dangereuses, à éviter. Ces deux objectifs ne se distinguent pas en simulation. Lorsque le robot explore matériellement un environnement par contre, le fait d'éviter les zones dangereuses devient un objectif prioritaire dans la mesure où il conditionne la survie du robot.

L'approche proposée dans cet article repose sur deux hypothèses. La première est que les zones dangereuses sont effectivement non prometteuses (la récompense associée à une trajectoire visitant une zone dangereuse est dissuasive). La seconde est que la fonction de récompense est continue : les zones dangereuses sont « suffisamment éloignées » des bonnes trajectoires. Cette hypothèse diffère de l'approche de CGRL [2] à deux niveaux : l'évaluation d'une trajectoire se fait uniquement à la fin de celle-ci, nous pouvons nous passer de l'hypothèse de récompenses instantanées et plus encore de celle de leur Lipschitz-continuité [6].

Sous ces hypothèses, la contribution faite réalise l'intégration dynamique d'une approche Monte-Carlo arborescente (MCTS), et de la caractérisation adaptative de la zone « sûre » sous forme d'un modèle DEJA-VU. Formellement, l'exploration MCTS est réalisée sous contrainte : seules les actions/états admissibles au sens du modèle sont effectuées. Parallélement, chaque exploration conduit à étendre la zone visitée, et à mettre ainsi à jour le modèle. La difficulté consiste à généraliser prudemment le modèle ; faute de quoi, l'hypothèse de continuité nous conduirait à inclure graduellement la zone dangereuse dans la couverture du modèle. Cette difficulté a été résolue dans l'algorithme Educated MCTS présenté de manière paramétrique (paramètre r).

Les résultats préliminaires obtenus démontrent l'intérêt de l'approche. L'option la plus conservative ($r = 1$) évite radicalement la zone dangereuse, tout en découvrant quelquefois le trésor. Par comparaison, l'algorithme de référence s'approche également du trésor mais n'évite la zone dangereuse que très rarement. L'évaluation des performances en termes de domination de Pareto confirme l'amélioration significative apportée par Educated MCTS en moyenne, quant au compromis exploration / sécurité.

Educated MCTS repose ici sur deux conditions nécessaires : les hypothèses relatives à la fonction de récompense (continuité et valeurs mauvaises sur le danger) et la présence d'informations fournies par un expert. D'autres hypothèses ont été formulées sur le problème applicable (déterminisme des transitions et finitude des espaces d'actions et d'états). Ces dernières constituent des hypothèses simplificatrices définissant un premier cadre de travail, et ne remettent pas fondamentalement en cause la conception d'Educated MCTS. L'adaptation au cas continu (états et actions) et non déterministe est ainsi la première perspective d'extension. Celle-ci se fondera sur les adaptions d'UCT au cas continu [15] et non déterministe [17].

À ces extensions du premier cadre de travail s'ajoutent de nouvelles perspectives. La première concerne l'adaptation en-ligne du modèle DEJA-VU, pour contrôler la couverture du modèle. L'idée consiste à réguler les risques pris en fonction des progrès de l'exploration, et à n'autoriser une augmentation de la couverture du modèle que si les perfor-

6. L'existence d'une fonction de récompense instantanée continue rendrait en effet le problème trivial : il suffit de rebrousser chemin lorsque cette récompense devient mauvaise.

mances plafonnent.
La seconde utilise de manière plus active le modèle DEJA-VU pour guider l'exploration. Présentement, DEJA-VU est utilisé *a posteriori* pour filtrer les actions considérées comme dangereuses ; la question est de savoir si DEJA-VU pourrait également être utilisé pour guider l'exploration et sélectionner les actions opportunes dans l'esprit de l'heuristique RAVE [14].
Une troisième perspective enfin consiste à régler l'exploration et le déclenchement des actions de manière plus fine en fonction de DEJA-VU. Dans le contexte courant, une action donnée est en effet accomplie ou bloquée. Il est envisageable de l'exécuter avec plus ou moins d'intensité, en fonction des informations fournies par le modèle. Ainsi dans le cas du robot marcheur, la stratégie consisterait non à interdire de visiter les zones dangereuses, mais à marcher plus lentement en fonction du danger estimé.

Annexe

Spécifications du *puddle world*. Les expérimentations considèrent le PDM suivant :
- Une arêne 21 x 21 points.
  $\mathcal{S} = \{(-0.65, -0.35) + (i * 0.1, j * 0.1) \text{ avec } i, j = 0, \ldots, 20\}$
- Actions : Un pas de longueur 0.1 dans la direction Nord, Sud, Ouest ou Est. On ne peut pas sortir de l'arêne ou revenir directement sur ses pas.
- Position initiale $(x_0, y_0) = (0.15, 0.35)$
- $T = 30$.

Récompense pour $x = (x_1, x_2)$,

$$r(x) = a\mathcal{N}_{\mu_1,\Sigma_1}(x) + b\mathcal{N}_{\mu_2,\Sigma_2}(x) + c\mathcal{N}_{\mu_3,\Sigma_3}(x)$$

avec

$$\mathcal{N}_{\mu,\Sigma}(x) = \frac{1}{2\pi\sqrt{|\Sigma|}} e^{\frac{-(x-\mu)\Sigma^{-1}(x-\mu)^t}{2}}$$

et

$$\mu_1 = (1,1)\ \Sigma_1 = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.2 \end{pmatrix}$$
$$\mu_2 = (0.25, 0.75)\ \Sigma_2 = \begin{pmatrix} 0.02 & 0 \\ 0 & 0.001 \end{pmatrix}$$
$$\mu_3 = (0.45, 0.6)\ \Sigma_3 = \begin{pmatrix} 0.001 & 0 \\ 0 & 0.02 \end{pmatrix}$$
a = 30, b = -40, c = -100

La SVM monoclasse a été implémentée grâce à libsvm [12]. Nous nous sommes servis d'un noyau gaussien $K(x, y) = e^{-\gamma\|x-y\|^2}$ avec $\gamma = 100$ et $\nu = 0.01$.

Références

[1] Sutton R. S. et Barto A. G., *Reinforcement Learning : An Introduction,* MIT Press, 1998.

[2] R. Fonteneau, S.A. Murphy, L. Wehenkel et D. Ernst, A cautious approach to generalization in reinforcement learning, *Proceedings of The International Conference on Agents and Artificial Intelligence (ICAART 2010)* ,pp. 64-73 , 2010.

[3] R. Fonteneau, S.A. Murphy, L. Wehenkel et D. Ernst, Towards Min Max Generalization in Reinforcement Learning. *In Agents and Artificial Intelligence : International Conference, ICAART 2010*, pp. 61-77, J. Filipe, A. Fred, and B.Sharp. Springer, Heidelberg, 2011.

[4] Levente Kocsis et Csaba Szepesvári, Bandit based monte-carlo planning, *In ECML-06. Number 4212 in LCNS*, pp. 282-293, 2006.

[5] T.L. Lai et H. Robbins, Asymptotically efficient adaptive allocation rules, *Advances in Applied Mathematics*,6, pp. 4-22, 1985.

[6] Peter Auer, Nicolò Cesa-Bianchi et Paul Fischer, Finite-time analysis of the multiarmed bandit problem, *Machine Learning*,47, pp. 235-256, 2002.

[7] Sylvain Gelly, Yizao Wang, Rémi Munos et Olivier Teytaud, Modification of UCT with patterns in Monte-Carlo Go, *Research Report*, INRIA RR-6062, 2006.

[8] Bernhard E. Boser, Isabelle M. Guyon et Vladimir N. Vapnik, A training algorithm for optimal margin classifiers, *In Proceeding of the 5th annual ACM workshop on computational learning theory*, pp. 144-152, 1992.

[9] Corinna Cortes et Vladimir Vapnik, Support-vector networks, *In Machine Learning*, pp 273-297, 1995.

[10] Richard S. Sutton, Generalization in reinforcement learning : Successful examples using space coarse coding, *In Advances in Neural Information Precessing Systems*,8 , pp 1038-1044, 2006.

[11] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola et Robert C. Williamson, Estimating the support of a high dimensional distribution, *Neural Computation*, 13 , pp 1443-1471, 2001.

[12] Chih-Chung Chang et Chih-Jen Lin, LIBSVM : A library for support vector machines, *ACM Transactions on Intelligent Systems and Technology*,2,1–27,2011.

[13] M.J. Mataric et D. Cliff Challenges in evolving controllers for physical robots, *Robotics and Autonomous Systems*,19,67–83,1996.

[14] Sylvain Gelly et David Silver, Combining online and offline knowledge in UCT *Proceedings of the 24th international conference on Machine learning, ICML 07*,273–280,2007.

[15] Adrien Couëtoux, Jean-Baptiste Hoock, Nataliya Sokolovska, Olivier Teytaud et Nicolas Bonnard, Continuous Upper Confidence Trees, *LION 2011*, 433–445, 2011.

[16] Stefano Nolfi et Dario Floreano, *Evolutionary Robotics : The Biology,Intelligence,and Technology,* MIT Press, 2000.

[17] István Szita, Guillaume Chaslot et Pieter Spronck, Monte-Carlo Tree Search in Settlers of Catan, *Advances in Computer Games. 12th International Conference, ACG2009*, 21–32, 2010.