Exploration et Exploitation dans des MDPs Cybernétiques

Filipo Studzinski Perotto

Université de Toulouse 1 - Capitole
Institut de Recherche en Informatique de Toulouse
Équipe LILaC
filipo.perotto@irit.fr

Résumé : Dans cet article on présente l'algorithme "average engaged climber" (AEC), une version modifiée de la méthode d'itération sur les valeurs pour l'estimation de la fonction d'utilité dans un *processus de décision markovien* (MDP) à horizon infini. Plus précisément, la méthode présentée vise à résoudre le dilemme entre exploration et exploitation dans une classe particulière de problèmes appelée *MDP cybernétique* (C-MDP). Dans ces problèmes, l'agent est doté d'une énergie interne dont la valeur est affectée par les récompenses reçues au long de son cycle de vie, et il doit la maintenir au-dessus d'un seuil de viabilité, afin d'assurer sa propre survie. Ainsi, l'agent doit chercher à optimiser son comportement, tout en tenant compte des coûts des actions exploratoires. L'originalité de la méthode proposée repose sur le fait de construire explicitement deux politiques différentes (une pour explorer, l'autre pour exploiter), et de déterminer les bons moments pour changer de stratégie en utilisant une notion d'engagement. L'algorithme estime l'utilité des paires état-action en approchant la récompense moyenne espérée de chacune. L'intérêt potentiel de l'idée est appuyé par des résultats expérimentaux préliminaires.

Mots-clés : Adaptation et Apprentissage Automatique, Apprentissage par Renforcement, Processus de Décision Markovien, Dilemme Exploration-Exploitation

1 Introduction

Trouver l'équilibre entre les comportements d'exploration et d'exploitation est un dilemme essentiel affronté par tout agent qui doit s'adapter à son environnement de façon autonome. Le défi de trouver un bon compromis entre l'exploration (apprendre de nouvelles choses) et l'exploitation (agir de manière optimale en fonction de ce qui est déjà connu) constitue un problème largement étudié dans les domaines de la prise de décision séquentielle et de l'apprentissage par renforcement (Puterman, 1994; Sutton & Barto, 1998; Sigaud & Buffet, 2010; Wiering & Otterlo, 2012; Feinberg & Shwartz, 2002). L'agent est censé favoriser les actions bien récompensées, mais il doit à la fois expérimenter des actions inconnues (ou méconnues) pour pouvoir découvrir de nouvelles façons d'atteindre de meilleures récompenses.

En *apprentissage par renforcement avec modèle* (*MBRL*), l'agent doit simultanement (a) apprendre un modèle du monde (c'est-à-dire, approcher progressivement les fonctions de récompense et de transition à partir de ses observations) et (b) profiter de ses connaissances pour optimiser son comportement (c'est-à-dire, définir une politique d'actions à partir des modèles afin de maximiser l'espérance de récompenses futures). Prises ensemble, ces deux tâches créent une dépendance cyclique : la façon dont l'agent se comporte conditionne ce qu'il peut apprendre, et ce qu'il sait conditionne la manière dont il se comporte. Telle est l'origine du *dilemme exploration-exploitation*, et cela pose deux défis supplémentaires à l'agent : (c) trouver un équilibre entre l'exploration et l'exploitation, et (d) réaliser l'exploration de manière efficace.

Dans cet article, nous nous intéressons à l'ensemble des problèmes représentés par des *processus de décision markoviens* (MDPs) ergodiques, et qui se déroulent sur un horizon temporel infini. Ces sont des MDPs non-épisodiques, qui ne présentent pas des buts ou des états terminaux, où le cycle de vie de l'agent n'est pas limité dans le temps. Plus précisément, nous nous intéressons aux *MDPs cybernétiques* (C-MDPs), dans lesquels l'agent essaie d'apprendre une politique optimale, mais en tenant compte des coûts associés à ses actions.

Dans un C-MDP, l'agent possède une variable ω qui représente son "énergie". L'énergie n'est pas une

observation au sens classique, mais elle doit être considerée au moment de la prise de décision. Le processus de décision cybernétique se constitue ainsi comme un problème de survie. L'énergie de l'agent se modifie comme suit, où $\{r \in \mathbb{R}\}$ est la récompense immédiate reçue par l'agent :

$$\omega' \leftarrow \omega + r \tag{1}$$

Le niveau d'énergie de l'agent ω est initialisé avec une valeur positive (l'énergie initiale $\{\omega_0 \in \mathbb{R} \mid \omega_0 > 0\}$), et ce niveau évolue comme la récompense cumulative. L'énergie est définie comme suit :

$$\omega_n = \omega_0 + \sum_{t=1}^{n} r_t \tag{2}$$

Le problème prend tout son sens quand on considère les MDPs dans lesquels les récompenses peuvent être positives et négatives, c'est-à-dire $\{\exists s \exists a \mid \Pr(R(s,a) > 0) > 0\} \wedge \{\exists s \exists a \mid \Pr(R(s,a) < 0) > 0\}$. L'agent est tenu de maintenir son niveau d'énergie, que nous pouvons interpréter comme une variable essentielle (Ashby, 1956), dans des limites de viabilité. Ainsi, dans un C-MDP, l'agent doit apprendre une politique optimale (ou quasi-optimale), tout en préservant $\omega > 0$. Autrement dit, l'agent doit à la fois apprendre et survivre.

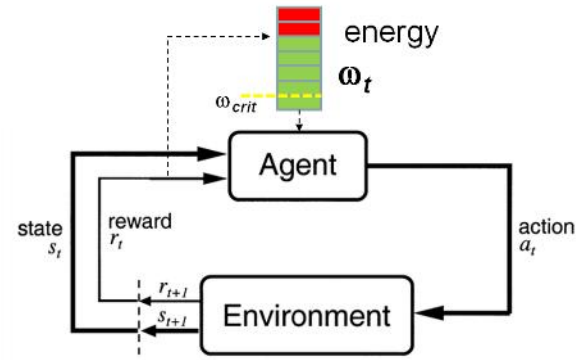

FIGURE 1 – C-MDP : la version cybernétique du problème classique d'apprentissage par renforcement.

Dans cet article, on aborde le problème de trouver le bon moment pour basculer entre une stratégie d'exploration et une stratégie d'exploitation. Même si le dilemme exploration-exploitation est étudié depuis plusieurs années, de notre point de vue, les approches standards mélangent le choix de la stratégie et le choix de l'action dans une même étape de décision. Dans cet article on veut analyser le potentiel avantage de distinguer ces deux choix plus clairement.

Cette recherche trouve sa motivation dans la notion intuitive que, en général, lorsque l'agent suit une politique, il enchaîne des actions spécifiques afin d'atteindre une certaine récompense intéressante à la fin de la séquence. Une fois engagé dans une séquence d'actions, il n'est pas très intéressant de changer de stratégie avant d'atteindre la récompense, sinon l'effort investi sera perdu. Le même *insight* s'applique à l'exploration : lorsque l'agent dépense son temps et son énergie à voyager à travers l'espace d'états en cherchant une certaine situation spécifique qui lui permettrait d'apprendre quelque chose d'important sur le monde, l'occasion ne devrait pas être manquée à cause d'un soudain changement de stratégie.

Dans (Perotto, 2015), la méthode *engaged climber* (EC) a été proposée, le concept d'*engagement* a été introduit, ainsi que l'idée du maintien explicite de deux politiques (l'une pour exploiter, l'autre pour explorer) issues de deux estimations d'utilité différentes : les *récompenses attendues* et les *progrès d'apprentissage espérés*. L'idée est que l'agent reste concentré sur la même stratégie afin d'atteindre le prochain "pic" (une récompense importante ou une grande découverte), comme un "grimpeur" qui s'engage avec un objectif, poursuivant sa démarche jusqu'à l'arrivée au sommet.

Cependant, la méthode EC calcule l'utilité des actions en fonction des récompenses cumulées actualisées (pondérées par un facteur de réduction γ). Ce critère d'optimalité n'est pas toujours adapté aux problèmes à horizon illimité. En plus, la version préliminaire de la méthode EC nécessite d'un réglage manuel de certains paramètres comme le temps moyen d'engagement avec une stratégie ou le seuil d'identification d'un pic de récompense, valeurs qui sont fortement dépendantes du contexte de chaque scenario.

En vue de cela, la contribution de cet article est une version améliorée de la méthode EC. Dans cette nouvelle version, appelée *average engaged climber* (AEC), les *temps d'engagement* ainsi que les états considérés comme des *pics* sont dynamiquement définis et associés à chaque paire état-action dans chacune des fonctions d'utilité (exploration / exploitation) afin d'estimer combien de temps l'agent doit suivre la même politique s'il veut concrétiser l'utilité espérée pour la stratégie choisie. Ces paramètres sont appris par l'agent en même temps qu'il estime les fonctions d'utilité.

Différemment de la version précédente, l'algorithme présenté dans cet article détermine l'utilité d'une action par la moyenne des récompenses prévues, au lieu d'utiliser la somme actualisée des récompenses. Pour certains problèmes d'apprentissage par renforcement non-épisodiques, cette dernière approche peut piéger l'agent en évaluant des politiques sous-optimales comme étant meilleures que les politiques optimales (Tadepalli, 2010).

Cet article est organisé comme suit : la section 2 revisite les concepts principaux de l'apprentissage par renforcement et des processus de décision markoviens, en se concentrant sur les méthodes standards pour équilibrer l'exploration et l'exploitation, et sur les méthodes standards pour estimer la valeur d'une politique par récompense moyenne ; la section 3 introduit les méthodes EC et AEC, et montre des résultats expérimentaux ; la section 4 présente les conclusions.

2 Processus de Décision Markoviens

Les *Processus de Décision Markoviens* (MDPs) sont au centre d'un *framework* largement utilisé pour représenter les problèmes de *contrôle automatisé*, de *prise de décision*, de *planification*, et d'*apprentissage par renforcement* (Puterman, 1994; Sutton & Barto, 1998; Sigaud & Buffet, 2010; Wiering & Otterlo, 2012; Feinberg & Shwartz, 2002; Uther, 2010). Un MDP est typiquement représenté par une machine à états finis discrète et stochastique : à chaque pas de temps la machine est dans un état s ; l'agent observe l'état et il choisi une action a à réaliser ; ensuite la machine change vers un nouvel état s' et donne à l'agent la récompense r correspondante. Il n'y a pas une phase d'entrainement séparée, et l'agent doit apprendre à coordonner ses actions par essais et erreurs.

D'une façon générale, un MDP peut être formellement défini comme un quadruplet $\{S, A, T, R\}$ où $S = \{s_1, s_2, ..., s_{|S|}\}$ est l'ensemble des états du système, $A = \{a_1, a_2, ..., a_{|A|}\}$ est l'ensemble des actions de l'agent, $T(s, a, s') = \Pr(s'|s, a)$ est la fonction de transition, et $R(s, a, s', r) = \Pr(r|s, a, s')$ est la fonction de récompense. La fonction de transition T définit la dynamique du système en déterminant l'état suivant s' à partir de l'état actuel s et de l'action exécutée a. La fonction de récompense R définit la récompense immédiate après la transition de s à s' avec a. Une politique déterministe π établit une correspondance entre les états et les actions sous la forme $\pi(s) = a$, ce qui permet de définir le comportement de l'agent en indiquant une action à effectuer pour chaque état du système.

Résoudre un MDP signifie trouver la politique d'actions qui maximise les récompenses reçues par l'agent au cours du temps. Comme la politique optimale peut être difficile à apprendre, dans la pratique on espère d'un bon algorithme qu'il soit capable d'apprendre une politique quasi-optimale avec une forte probabilité (Valiant, 1984).

Lorsque les fonctions de récompense et de transition sont données, le MDP peut être résolu par *programmation dynamique*. En MBRL, l'agent approche graduellement les paramètres de ces deux fonctions. Après chaque observation, les modèles sont mis à jour de façon incrémentielle. Ensuite, à partir de ces modèles, l'agent résout le MDP en estimant l'utilité de chaque paire état-action, et finalement en associant les états avec les meilleures actions, de manière à définir une politique (Sutton & Barto, 1998).

2.1 Conditions d'Optimalité

Quand l'agent a un horizon de temps infini ou illimité, son expérience ne peut pas être découpée en épisodes. En vue de cela, l'évaluation des politiques se fait, généralement, en utilisant la récompense cumulée actualisée, où un facteur de pondération $\{\gamma \in \mathbb{R} \mid 0 \leq \gamma < 1\}$ réduit le poids des récompenses futures par rapport aux récompenses immédiates. La récompense cumulative actualisée est définie comme suit :

$$v = \sum_{t=1}^{\infty} \gamma^t . r_t \tag{3}$$

Une fonction d'utilité V peut être itérativement approchée par *itération sur les valeurs* (*value-iteration*) à partir des modèles T et R, puis une politique optimale peut être extrait de V comme suit :

$$Q'(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} T(s,a,s')V(s') \tag{4}$$

$$V'(s) \leftarrow \max_a [Q'(s,a)] \tag{5}$$

$$\pi'(s) \leftarrow \arg\max_a [Q'(s,a)] \tag{6}$$

La valeur cumulée des récompenses actualisées est toujours finie, ce qui assure la convergence des méthodes itératives vers une solution optimale. Cependant, dans de nombreux domaines, il n'y a aucune interprétation naturelle pour le facteur de réduction γ. Encore plus problématique, dans des domaines récurrents (où les récompenses tardives ont la même importance que les récompenses précoces) un tel critère peut fausser l'utilité réelle de certaines séquences d'actions. Un critère d'optimalité alternatif est la récompense moyenne obtenue par pas de temps (Tadepalli & Ok, 1998), généralement appelé *gain*.

Sous quelques suppositions raisonnables, la récompense moyenne (ou le gain) d'une politique donnée π converge vers une valeur unique $\rho(\pi)$ indépendamment de l'état de départ (Puterman & Patrick, 2010), où :

$$\rho(\pi) = \lim_{n\to\infty} 1/n \sum_{t=1}^{n} r_t \tag{7}$$

De cette façon, pour une politique π donnée, le gain pour tout état est équivalent à la récompense moyenne, $g(s) = \rho(\pi)$. Une politique π est dite optimale sur le gain si elle maximise $\rho(\pi)$, c'est-à-dire, si à long terme elle offre la moyenne des récompenses la plus élevée.

L'utilisation de la récompense moyenne (gain) comme critère d'optimalité amène l'agent à un circuit optimal dans le MDP. Cependant, même si le gain $\rho(\pi)$ d'une politique π est indépendant de l'état initial du processus, la récompense totale reçue dans un instant donné ne l'est pas, soit $n.\rho(\pi) \neq \sum_{t=1}^{n} r_t$. Un deuxième critère d'optimalité est alors nécessaire afin de comparer les états qui ne font pas partie du circuit optimal. Cette différence, appelée *biais*, est notée $h(s)$. Les politiques biais-optimales sont, parmi toutes les politiques gain-optimales, celles qui optimisent également les récompenses transitoires initiales. La combinaison de biais et de gain permet à l'agent d'atteindre de manière optimale le circuit optimal (Tadepalli & Ok, 1998).

La méthode standard en apprentissage par renforcement utilisant la récompense moyenne est l'algorithme *H-Learning* (Tadepalli, 2010). Dans cet algorithme, les valeurs de gain et de biais sont itérativement mises à jour comme suit, où α est le taux d'apprentissage et r est la récompense immédiate reçue :

$$\rho' \leftarrow (1-\alpha)\rho + \alpha(r - h(s) + h(s')). \tag{8}$$

$$h'(s) \leftarrow R(s,a) + h(s') - \rho \tag{9}$$

Estimer la moyenne des récompenses ρ pendant l'apprentissage n'est pas évidant, car sa valeur est déformée par les actions exploratoires. Pour cette raison, cette estimation doit être révisée uniquement lorsque l'agent exécute des actions d'exploitation. En outre, parce que la récompense moyenne ρ et le biais h sont calculés simultanément, la convergence ne peut être garantie que si ρ est actualisé à une échelle de temps beaucoup plus lente que h.

2.2 Dilemme Exploration-Exploitation

La méthode la plus basique pour équilibrer l'exploration et l'exploitation est l'algorithme ε-*greedy*, qui définit des proportions fixes de temps pour explorer ou pour exploiter. Dans cet algorithme, à chaque pas de temps, l'agent peut soit exécuter une action aléatoire avec une probabilité ε, soit il peut suivre la politique optimale courante. Une autre méthode classique est l'algorithme *softmax*, où l'agent sélectionne parmi les actions possibles proportionnellement à leur utilité estimée. Ainsi, la fonction d'utilité estimée sert à construire une politique stochastique. Dans *softmax*, les actions les plus bien évaluées sont exécutées avec des probabilités plus élevées, mais les actions moins récompensées peuvent être aussi choisies de temps à autre. Un paramètre de *température* τ est utilisé pour définir comment la différence entre les utilités estimées

affecte le choix de l'action. Concernant ces deux algorithmes, ε ou τ sont, à la base, des paramètres réglés à la main. Plus la valeur de ε ou de τ est haute, plus l'agent aura tendance à présenter des comportements exploratoires, tandis que des valeurs faibles entraînent une sélection d'actions dirigée vers l'exploitation.

Le problème avec ces méthodes basiques est que le paramètre d'exploration n'est pas ajusté au cours du processus d'apprentissage. Pour rendre ce paramètre évolutif, et donc les algorithmes plus souples et adaptables, un méta-paramètre peut être inclus afin de réguler le paramètre d'exploration (Auer *et al.*, 2002). C'est le cas de ε-*first*, méthode qui effectue une exploration pleine pendant une période de temps prédéterminée, et passe à l'exploitation totale dans un deuxième temps. Alternativement, l'algorithme *decreasing*-ε propose la réduction progressive de la valeur du paramètre ε, de sorte que la probabilité d'exploration diminue à un taux de $1/n$, où n est le temps écoulé (en cycles). La méthode *softmax* peut être modifiée de la même manière, en faisant la valeur de τ diminuer au fil du temps. Toutefois, dans tous ces cas, la valeur du méta-paramètre est fixe.

Afin de réguler dynamiquement le taux de décroissance de ε, l'algorithme *VDBE* (Tokic, 2010; Tokic & Palm, 2011) utilise le paramètre $\varepsilon(s)$ comme une fonction qui mesure l'incertitude associée à l'estimation de l'utilité de chaque état. Cette incertitude est donnée par les fluctuations de la propre estimation. L'idée est que l'agent doit être plus exploratoire dans les régions de l'environnement où son modèle est encore instable.

Une autre approche standard pour résoudre le dilemme exploration-exploitation est l'*optimisme face à l'incertitude*. Cette approche consiste à ajouter un bonus sur l'utilité des paires état-action moins fréquentées (Meuleau & Bourgine, 1999). Cela peut être fait par une initialisation optimiste des estimations dans le modèle de récompenses, couplée à une sélection d'actions basée sur une politique unique d'exploitation (*greedy*). Ainsi les actions qui conduisent l'agent à des situations peu connues seront probablement préférées. Plus une paire état-action est expérimentée, plus la récompense estimée pour elle s'approchera de sa vraie valeur. Ces algorithmes présentent une phase initiale plus exploratoire, qui est progressivement remplacée par un comportement d'exploitation, suivant la convergence des modèles de récompense et de transition.

Cette approche, qui pousse l'agent vers l'inconnu, est utilisée dans l'algorithme *R-Max* (Brafman & Tennenholtz, 2002), une des méthodes standards pour résoudre le dilemme exploration-exploitation en MBRL. *R-Max* généralise E^3 (Kearns & Singh, 2002), qui constitue une autre méthode aussi bien établie. Directement ou indirectement, ces algorithmes prennent en compte le nombre de fois qu'une paire état-action a été visitée. Les paires moins visitées ont un bonus d'exploration supplémentaire, qui augmente l'utilité estimée de ses récompenses. Les modèles estimés sont *probablement approximativement corrects* (PAC) après un nombre suffisant d'expériences.

Une fois que la méthode pour équilibrer l'exploration et l'exploitation est choisie, un autre problème est de définir la meilleure façon d'explorer l'environnement. Les méthodes de la famille ε ou *softmax* supposent la réalisation d'une exploration non-dirigée (*undirected*) (Kaelbling *et al.*, 1996), simplement en rajoutant une part de hasard dans le processus de sélection d'actions. Performer une exploration non-dirigée signifie exécuter des actions aléatoires. Cependant, en procédant ainsi (exécutant des actions au hasard), l'agent peut finir par gâcher son effort d'exploration en expérimentant des paires état-action déjà bien modélisées, ou par s'auto-pénaliser en se dirigeant vers des récompenses dangereusement négatives connues.

Contrairement, les méthodes d'exploration dirigées (*directed exploration*) préfèrent essayer les paires état-action les moins visitées, en pondérant cette préférence avec leur utilité (les récompenses futures prévues). Ce type d'exploration est réalisé par des algorithmes comme *R-Max* (Brafman & Tennenholtz, 2002), *UCB* (Auer *et al.*, 2002), or E^3 (Kearns & Singh, 2002). Des estimations à priori optimistes sont associées aux paires état-action moins fréquentes afin d'induire l'agent à les essayer. Donc, si un état à découvrir n'est pas trop loin, et qu'il n'y a pas d'évidence sur de possibles récompenses excessivement négatives sur le chemin, l'agent sera amené à voyager à travers l'espace pour atteindre cet état méconnu, tout simplement en suivant sa politique d'exploitation. Les méthodes d'exploration dirigée ajoutent le bonus d'exploration dans l'estimation de la récompense, ce qui conduit l'agent à une phase initiale plutôt tournée vers l'exploration, et qui se transforme progressivement en exploitation, à mesure que le monde devient suffisamment connu.

3 Le Grimpeur Engagé

Dans (Perotto, 2015) une version préliminaire de la méthode *engaged climber* (EC) a été présentée. L'idée de créer explicitement deux politiques séparées (une pour l'exploration et l'autre pour l'exploitation) y était déjà présente, ainsi que l'idée de rester engagé avec la stratégie choisie pour un certain temps avant de réévaluer le choix. La métaphore pour illustrer l'*insight* est celle d'un "grimpeur engagé" qui, à un moment donné, voit deux pics intéressants à escalader. Une fois que le pic est choisi, le grimpeur restera engagé avec la mission de l'atteindre, en persévérant dans sa stratégie. Le choix n'est pas réévalué ou remis en question au cours de la montée, à moins qu'une situation inattendue et critique se produise. Cette réflexion est illustrée dans la figure 2.

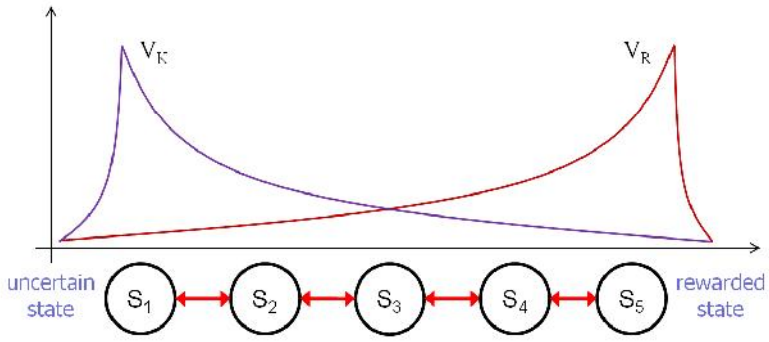

FIGURE 2 – Deux pics en vue : à gauche un état intéressant à explorer, et à droite un état intéressant à exploiter.

Cependant, la version de la méthode EC proposée en (Perotto, 2015) suppose un réglage manuel des certains paramètres pour être en mesure de fonctionner. En outre, cette version calcule l'utilité en utilisant les récompenses cumulées actualisées. Dans des MDP à horizon infini, comme les MDPs cybernétiques utilisés dans les expériences présentées à la fin de la section, une condition d'optimalité fondée sur la moyenne des récompenses correspond mieux au comportement attendu de l'agent.

Dans un MDP stationnaire et ergodique à horizon temporel non-limité, toute politique optimale contient au moins un ensemble fermé et cyclique de paires état-action. Cet ensemble forme un "circuit optimal", vers où conduisent tous les autres états, comme illustre la figure 3.

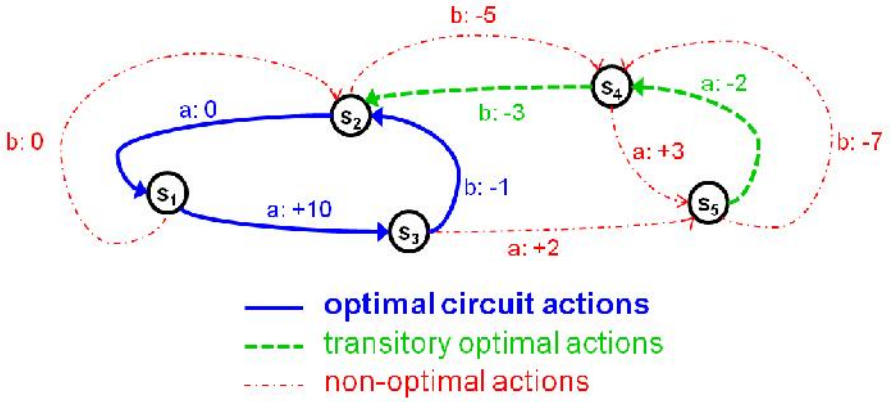

FIGURE 3 – Le circuit optimal est celui qui a le meilleur gain (récompense moyenne par cycle) dans le MDP. Les actions transitoires optimales sont celles qui ont le meilleur biais pour atteindre le circuit optimal.

Dans cet article, on présente l'algorithme *Average Engaged Climber* (AEC), une version modifiée de l'algorithme EC (Perotto, 2015), qui estime le gain et le biais de chaque état, similairement à ce qui est fait dans l'algorithme *H-learning* (Tadepalli, 2010). AEC fonctionne par itération de valeurs, calculant

les fonctions d'utilité à partir des modèles de transition et de récompense. Pendant le calcul, la politique d'exploitation issue de la fonction d'utilité calculée conduira l'agent, à partir de chaque état, soit vers un état intéressant (un " pic " de récompense) dans un avenir proche, soit vers un circuit. Le gain g représente la récompense moyenne de la politique dans une limite de temps à l'infini. Après la convergence, cette valeur doit correspondre à la récompense moyenne dans le circuit optimal. Le biais h représente la différence liée aux récompenses transitoires, et il est défini par la récompense moyenne du meilleur chemin vers le circuit optimal.

Pour être en mesure d'estimer ces moyennes, chaque état est associé à un état cible $t(s) = s^\star$ (l'état de plus grande utilité sur un horizon proche) et à un temps d'engagement $e(s) \in \mathbb{N}$ (le nombre estimé de pas de temps pendant lesquels la politique doit être suivie pour que l'agent puisse atteindre l'état $s^\star$ à partir de s).

Les modèles de transition et de récompense ($\widehat{T}$ et $\widehat{R}$) sont associés à une mesure de confiance $\{\sigma \in \mathbb{R} \mid 0 \leq \sigma \leq 1\}$. $\sigma_{\widehat{T}}(s,a)$ représente la confiance sur la probabilité estimée des transitions à partir de l'état s avec l'action a, donnée par $\widehat{T}$. De la même façon, $\sigma_{\widehat{R}}(s,a)$ représente la confiance sur la récompense prévue par $\widehat{R}$ pour l'exécution de l'action a dans l'état s.

L'algorithme maintient simultanément deux fonctions d'utilité. La fonction $V_R(s,a)$ représente l'*utilité de récompense*, calculée à partir du modèle de récompenses $\widehat{R}$. La fonction $V_K(s,a)$ représente l'*utilité d'apprentissage*, calculée à partir de la confidence dans les modèles, $\sigma_{\widehat{T}}$ et $\sigma_{\widehat{R}}$. Les utilités estimées par ces deux fonctions sont accompagnées d'une valeur qui correspond à l'engagement nécessaire $e_R(s,a)$ et $e_K(s,a)$ (en pas de temps), et de l'indication d'un état-cible $t_R(s,a)$ et $t_K(s,a)$, qui représente le "pic" à atteindre. Ainsi, V_K est l'utilité qui guide l'agent vers des possibles améliorations de ses connaissances, tandis que V_R est l'utilité classique, qui guide l'agent vers les bonnes récompenses.

Les fonctions $V(s,a)$ sont composées d'une paire de fonctions $V(s,a) = \langle g(s,a), h(s,a)\rangle$. Pendant le calcul par itération de valeurs, $V(s,a)$ est équivalente à $g(s,a)$, à moins que $g(s,a)$ soit encore indéfini. Dans ce cas, la valeur de $V(s,a)$ est équivalente à $h(s,a)$. Dans la comparaison entre deux valeurs, si le gain est équivalent, le biais est utilisé pour briser l'égalité. Les politiques $\pi_R(s)$ et $\pi_K(s)$ sont définies pour un état donné s comme l'action a qui maximise l'utilité respective.

Les fonctions d'utilité sont mises à jour par les méthodes $UpdateV_R(\widehat{T},\widehat{R})$ et $UpdateV_K(\widehat{T},\sigma)$ par *itération sur les valeurs* suivant les équations suivantes :

$$V_R(s) = \max_a \left[\sum \widehat{T}(s,a,s').\frac{(\widehat{R}(s,a) + (V_R(s').e(s'))}{e(s')+1}\right] \tag{10}$$

$$V_K(s) = \max_a \left[\sum \widehat{T}(s,a,s').\frac{(\sigma(s,a) + (V_K(s').e(s'))}{e(s')+1}\right] \tag{11}$$

La boucle principale de fonctionnement de la méthode AEC reste assez standard : l'agent interagit avec l'environnement, ensuite il se base sur l'expérience récente pour mettre à jour ses modèles, puis il calcule les utilités, et finalement il redéfinit les politiques. La différence par rapport aux algorithmes classiques est le maintien de deux fonctions d'utilité et de deux politiques (explorer/exploiter). Une autre différence est la séparation de la prise de décisions en deux étapes distinctes : le choix de la stratégie, et le choix de l'action.

L'algorithme 1 présente la boucle principale de AEC. À chaque pas de temps, après avoir observé l'état actuel du processus et son propre niveau d'énergie, l'agent choisit (ou maintient) une stratégie (explorer ou exploiter). Lorsque la stratégie est choisie, une action peut être sélectionnée à partir de la politique correspondante (π_K ou π_R). Une fois que l'action est exécutée, l'agent reçoit une récompense et observe l'état suivant du processus. Après cette phase d'interaction, la phase d'apprentissage intervient dans l'algorithme. D'abord les modèles de transition et de récompense ($\widehat{T}$ et $\widehat{R}$), sont mis à jour, ensuite les fonctions d'utilité (V_K et V_R) et de confiance (σ_K et σ_R), et enfin les politiques (π_K et π_R).

Algorithme 1 (Lifecycle())

$Initialize()$

loop

$s \leftarrow ObserveState()$
$\omega \leftarrow ObserveEnergy()$
$\pi \leftarrow ChooseStrategy()$

$a \leftarrow \pi(s)$
$ExecuteAction(a)$
$r \leftarrow GetReward()$
$s' \leftarrow ObserveState()$

$\widehat{T} \leftarrow UpdateT(\widehat{T}, s, a, s')$
$\widehat{R} \leftarrow UpdateR(\widehat{R}, s, a, s', r)$

$V_R \leftarrow UpdateV_R(\widehat{T}, \widehat{R})$
$V_K \leftarrow UpdateV_K(\widehat{T}, \sigma(\widehat{T}), \sigma(\widehat{R}))$

$\pi_R \leftarrow UpdatePolicy(V_R)$
$\pi_K \leftarrow UpdatePolicy(V_K)$

end loop

Construire un modèle du monde à partir de l'interaction constitue un défi à part entière, et il est en dehors de la portée de cet article. Dans la littérature scientifique en MBRL, plusieurs algorithmes sont proposés pour approcher les fonctions de transition et de récompense. Dans nos expériences, on utilise des méthodes simples et largement utilisées : l'estimation du *maximum de vraisemblance* (*maximum likelihood*) pour apprendre $\widehat{T}$, et la *moyenne mobile exponentielle* (*exponential moving averaging*), pour apprendre $\widehat{R}$.

Dans la méthode AEC, lorsque l'agent choisit une stratégie (l'exploration ou l'exploitation), il reste engagé avec elle pendant le nombre de cycles défini par la valeur de $e(s)$, qui indique le temps d'engagement associé à l'état courant au moment du choix. L'engagement est censé donner à l'agent suffisamment de temps pour atteindre le pic ciblé, qui correspond à l'état $t(s)$. Le mouvement est dit réussi si l'agent, après un temps $e(s)$, à partir de l'état s, en suivant sans hésitation la stratégie choisie, arrive à l'état $t(s)$. Lorsque la durée de l'engagement est terminée, l'agent redéfinit sa stratégie et prend un nouvel engagement.

La méthode $ChooseStrategy()$ (algorihm 2) identifie trois cas différents : (1) lorsqu'une situation critique est détectée, la stratégie est définie en fonction du type de situation ; (2) sinon, si l'agent est toujours engagé, la stratégie reste inchangé ; (3) finalement, si l'engagement précédant est terminé, une nouvelle stratégie est choisie de manière stochastique selon un paramètre ε.

Trois différents types de situation critique peuvent être identifiés : (a) il n'y a rien à exploiter à partir de l'état actuel, $V_R(s) < 0$, dans ce cas l'exploration doit être préférée ; (b) il n'y a plus rien à explorer à partir de l'état actuel, $V_K(s) = 0$, alors l'exploitation doit être préférée ; (c) l'énergie ω de l'agent est en dessous du niveau critique ω_{crit}, donc l'exploitation doit être préférée.

La variable ξ indique le nombre de cycles que l'agent doit rester encore engagé. La valeur de ξ est décrémentée à chaque pas de temps, et elle est mise à jour lorsque l'agent choisit une nouvelle stratégie de façon à correspondre à la valeur de $e(s)$ défini pour la stratégie choisie, où s est l'état actuel au moment de la prise de décision.

Algorithme 2 (ChooseStrategy())
if $V_R(s) < 0.0$ ***and*** $V_K(s) > 0.0$ ***then***
{nothing to exploit from here, change to exploration (case 1a)}
$\pi \leftarrow \pi_K$; $\xi \leftarrow e_K(s)$
else if $V_K(s) = 0.0$ ***then***
{nothing to explore from here, change to exploitation (case 1b)}
$\pi \leftarrow \pi_R$; $\xi \leftarrow e_R(s)$
else if $\omega < \omega_{crit}$ ***and*** $V_R(s) > 0$ ***then***
{energy level is critical, change to exploitation (case 1c)}
$\pi \leftarrow \pi_R$; $\xi \leftarrow e_R(s)$
else if $\xi = 0$ ***then***
{engagement is over, choose strategy using epsilon (case 2)}
if $rnd() > \varepsilon$ ***then***
$\pi \leftarrow \pi_R$; $\xi \leftarrow e_R(s)$
else
$\pi \leftarrow \pi_K$; $\xi \leftarrow e_K(s)$
end if

else
{otherwise preserve engaged strategy decrementing engagement time (case 3)}
$\xi \leftarrow \xi - 1$
end if

Comme nous l'avons déjà mentionné, parfois la stratégie du moment, avec laquelle l'agent est engagé, ne fait plus de sens (explorer quand une alerte d'énergie existe, explorer quand il n'y a plus rien à découvrir à partir de l'état actuel, ou exploiter lorsque la politique courante prévoit des récompenses négatives), dans ces cas, l'engagement doit être rompu.

L'algorithme AEC a été conçu pour traiter les C-MDPs où l'énergie ω de l'agent évolue suivant la récompense cumulée. En général, lorsque l'énergie devient inférieure à un certain niveau critique, la stratégie d'exploitation doit être activée afin de faire l'énergie revenir à un état non-critique, ce qui assure la survie de l'agent.

L'algorithme 3 montre comment l'utilité est calculée. Pour chaque paire état-action, cinq valeurs doivent être mises à jour à chaque itération : le gain $g(s,a)$, ce qui représente la récompense moyenne prévue à l'intérieur du circuit à être atteint, la taille $e_g(s,a)$ de tel circuit, le biais $h(s,a)$, qui représente la récompense moyenne dans le chemin transitoire qui amène à ce circuit, la taille $e_h(s,a)$ de ce chemin, et l'état cible $t(s,a)$, qui représente un pic d'utilité au but d'un chemin transitoire qui n'a pas encore trouvé un circuit, ou un point d'entrée dans un circuit.

Au début, toutes les paires état-action sont initialisées avec un gain indéfini, noté par un engagement nul $e_g(s,a) = 0$, mais avec un biais défini équivalent à la récompense attendue. À chaque nouvelle itération, la trajectoire transitoire est augmentée d'un état, et le biais (récompense moyenne dans la trajectoire) recalculé. Lorsque la paire état-action atteint un circuit, alors le gain est défini avec la récompense moyenne dans le circuit.

Ainsi, au cours des itérations, une paire état-action sera dans l'une de ces quatre situations : (a) simplement initialisée avec la récompense immédiate attendue comme biais, (b) une partie d'une trajectoire vers un pic, (c) une partie d'un circuit, ou (d) une partie d'une trajectoire vers un circuit. À la convergence, toutes les paires état-action doivent faire partie d'un circuit optimal ou d'un chemin optimal vers un circuit optimal.

Algorithme 3 (UpdateValues())
for all *(s,a)* ***do***
$s' \leftarrow ExpectedNextState(s, a, \widehat{T})$
$r \leftarrow ExpectedNextReward(s, a, s', \widehat{R})$
if $e_h(s') = 0$ ***and*** $e_g(s') = 0$ ***then***
{first visit (case a)}
$h(s,a) \leftarrow r$
$e_h(s,a) \leftarrow 1$
$g(s,a) \leftarrow 0.0$
$e_g(s,a) \leftarrow 0$
$t(s,a) \leftarrow s'$
else if $e_h(s') > 0$ ***and*** $e_g(s') = 0$ ***then***
{transient trajectory (case b)}
$h(s,a) \leftarrow ((h(s') * e_h(s')) + r)/(e_h(s') + 1)$
$e_h(s,a) \leftarrow e_h(s) + 1$
$g(s,a \leftarrow 0.0$
$e_g(s,a) \leftarrow 0$
$t(s,a) \leftarrow t(s')$
else if $e_h(s') = 0$ ***and*** $e_g(s') > 0$ ***then***
{immediate next state is in a circuit (case c)}
$h(s,a) \leftarrow r$
$e_h(s,a) \leftarrow 1$
$g(s,a) \leftarrow g(s')$
$e_g(s,a) \leftarrow e_g(s')$
$t(s,a) \leftarrow t(s')$
else if $e_h(s') > 0$ ***and*** $e_g(s') > 0$ ***then***

{trajectory toward a circuit (case d)}
$h(s,a) \leftarrow ((h(s') * c_h(s')) + r)/(c_h(s') + 1)$
$c_h(s,a) \leftarrow c_h(s) + 1$
$g(s,a) \leftarrow g(s')$
$e_g(s,a) \leftarrow e_g(s)$
$l(s,a) \leftarrow l(s')$
end if
end for
for all (s) ***do***
$V(s) \leftarrow \max_a \langle g(s,a), h(s,a) \rangle$
end for

3.1 Résultats Expérimentaux

L'expérience de la chaîne en boucle, version récurrente du problème proposé dans (Kaelbling *et al.*, 1996), configure un scénario de récompense retardée où l'agent doit découvrir la séquence unique d'actions qui mène au bout de la chaîne, comme illustre la figure 4. Une simple mauvaise action au milieu de la séquence suffit pour renvoyer l'agent de retour au point de départ. Une exploration non-dirigée demande un nombre de cycles qui s'élève à $O(2^{|S|})$ rien que pour atteindre la grande récompense une première fois, alors qu'une exploration dirigée nécessite seulement $O(|S|^2)$ cycles. Si'l n'y a pas de coût d'exploration, ni de limite de temps, une méthode comme *R-Max*, optimiste face à l'incertitude, suffit pour résoudre le problème : l'agent procédera à une exploration intensive de l'environnement avant de commencer progressivement à l'exploiter. Cependant, dans un C-MDP, un équilibre plus raffiné entre l'exploration et de l'exploitation se fait nécessaire.

Dans le scénario expérimental proposé, les fonctions de transition et de récompense sont déterministes. L'agent connaît le modèle de transition mais il doit apprendre un modèle de récompenses. Une chaîne comportant 10 états donne une récompense négative (-1,0) lorsque l'agent avance vers le dernier état, et une petite récompense positive (+0,5) quand il prend l'action qui lui renvoie au premier état. Quand l'agent effectue la séquence correcte d'actions, traversant la chaîne en parcourant tous ses états, il reçoit la grande récompense (+20,0). Ainsi, pour atteindre la grande récompense l'agent doit persévérer dans la marche, et accepter de recevoir une série de récompenses négatives avant de trouver son bonheur.

Dans cette expérience, le paramètre de confiance σ est mis à 1 quand une paire état-action est observée, autrement il a la valeur 0. L'énergie $\omega = 20 + \sum_{l=1}^{n} r_l$ est équivalent à la récompense cumulée ajoutée d'une énergie initiale positive, $\omega_0 = +20$. Le niveau d'énergie critique est fixé à $\omega_{crit} = +10$, ainsi des niveaux d'énergie inférieurs à ce seuil représentent une situation d'alerte. Le défi pour l'agent est de trouver la politique optimale en restant en vie.

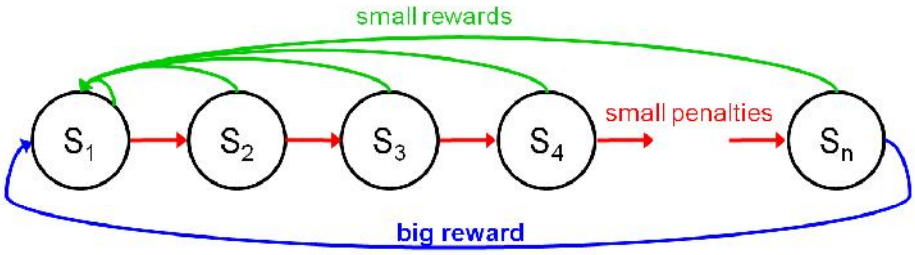

FIGURE 4 – Expérience de la châine en boucle

Dans des résultats préliminaires, AEC a démontré une performance intéressante. Concernant les C-MDPs, il bat la méthode ε-*greedy*, vu que cette dernière s'appuie sur une exploration non-dirigée, qui n'est pas adaptée aux problèmes de décision séquentielle en chaîne. Lorsque comparé à une méthode *optimiste face à l'incertitude*, AEC présente l'avantage de gérer les coûts d'exploration. Le tableau 1 montre les performances moyennes des trois méthodes sur 20 simulations. Pour la méthode ε-*greedy* nous avons réglé $\varepsilon = 0.99$ (exploration presque complètement aléatoire), sinon elle mettrait un temps démesurément long pour arriver à la fin de la chaîne, perturbée par les petites récompenses. La méthode optimiste en face à

TABLE 1 – Résultats expérimentaux

METHOD	TIME TO BIG REWARD	MINIMUM ENERGY	SURVIVAL RATE
ε-Greedy	≈ 1200	≈ -400	0%
Optimistic-Greedy	≈ 60	≈ -30	0%
Average Engaged Climber	≈ 180	≈ 0	100%

l'incertain a été implémentée simplement par une initialisation du modèle de récompenses avec des valeurs très positives. Cette méthode peut rapidement atteindre la grande récompense, vu qu'elle fera une exploration dirigée et intensive au début de la simulation, mais justement, cette façon d'explorer ne prend pas en compte les coûts, amenant l'agent systématiquement à la mort. Ainsi, due à la gestion des coûts d'exploration, AEC nécessite plus de temps que la méthode optimiste pour atteindre la grande récompense, mais il évite d'atteindre des niveaux d'énergie négatifs.

En outre, la manière standard pour le calcul de l'utilité, fondée sur la somme des récompenses actualisées, peut amener l'agent à une politique non-optimale. Dans l'expérience proposée, lorsque $\gamma = 0.8$, l'utilité de rester dans le premier état devient supérieure à l'utilité de suivre la chaîne jusqu'au bout. Le problème est que la grande récompense est trop loin, compte tenu du facteur de réduction. La méthode AEC calcule l'utilité en estimant la récompense moyenne, n'étant donc pas vulnérable à ce piège.

4 Conclusion et Perspectives

Quand l'agent est en mesure d'estimer la valeur de l'information, en comparant son importance à la valeur des récompenses à venir, il peut mieux équilibrer le temps consacré à l'exploration ou à l'exploitation. En MBRL, les méthodes standards suivent l'une des deux procédures : (a) soit l'agent réévalue la stratégie à chaque cycle (c'est le cas des méthodes de la famille ε), (b) soit l'agent construit une politique unique où l'utilité de l'exploration et l'utilité de l'exploitation sont confondues. La méthode proposée dans cet article sépare explicitement le processus de prise de décision en deux étapes : (1) choisir la stratégie (d'explorer ou d'exploiter), et persister dans le choix fait, et, à partir de ce premier choix, (2) sélectionner l'action à effectuer suivant la politique relative à la stratégie choisie.

L'utilisation des bonus d'exploration pour les états moins visités (l'optimisme face à l'incertitude) fait une compensation inhérente entre la valeur des récompenses futures et la valeur des apprentissages possibles. Des méthodes qui suivent ce principe mélangent l'*utilité de récompense* avec l'*utilité d'apprentissage* afin de pouvoir les comparer sur la base d'une "monnaie commune". Cependant, l'optimisme face à l'incertitude provoque généralement une longue phase d'exploration préliminaire. Selon le scenario, il n'est pas possible de se le permettre, par exemple, lorsqu'il y a des enjeux de sécurité, ou des risques à prendre en compte, ou des coûts d'exploration qui doivent respecter un certain budget.

Des expériences préliminaires ont montré que les choix faits par l'algorithme AEC peuvent constituer une voie prometteuse de recherche pour le cas des C-MDPs, même si des expériences dans de scenarios plus complexes sont encore nécessaires, ainsi qu'une analyse théorique plus approfondie de l'algorithme, pour les aspects de complexité et de robustesse.

Références

ASHBY W. (1956). *Introduction to Cybernetics*. Chapman & Hall.

AUER P., CESA-BIANCHI N. & FISCHER P. (2002). Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, **47**(2-3), 235–256.

BRAFMAN R. & TENNENHOLTZ M. (2002). R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, **3**, 213–231.

FEINBERG E. & SHWARTZ A. (2002). *Handbook of Markov Decision Processes : methods and applications*. Kluwer.

KAELBLING L., LITTMAN M. & MOORE A. (1996). Reinforcement learning : A survey. *J. Artif. Int. Res.*, **4**(1), 237–285.

KEARNS M. & SINGH S. (2002). Near-optimal reinforcement learning in polynomial time. *Mach. Learn.*, **49**(2-3), 209–232.

MEULEAU N. & BOURGINE P. (1999). Exploration of multi-state environments : Local measures and back-propagation of uncertainty. *Mach. Learn.*, **35**(2), 117–154.

PEROTTO F. (2015). Looking for the right time to shift strategy in the exploration/exploitation dilemma. *Schedae Informaticae*, **24**. to appear.

PUTERMAN M. (1994). *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. Wiley, 1 edition.

PUTERMAN M. & PATRICK J. (2010). Dynamic programming. In C. SAMMUT & G. WEBB, Eds., *Encyclopedia of Machine Learning*, p. 298–308. Springer.

O. SIGAUD & O. BUFFET, Eds. (2010). *Markov Decision Processes in Artificial Intelligence*. iSTE - Wiley.

SUTTON R. & BARTO A. (1998). *Introduction to Reinforcement Learning*. MIT Press.

TADEPALLI P. (2010). Average-reward reinforcement learning. In C. SAMMUT & G. WEBB, Eds., *Encyclopedia of Machine Learning*, p. 64–68. Springer.

TADEPALLI P. & OK D. (1998). Model-based average reward reinforcement learning. *Artif. Int.*, **100**(1-2), 177–224.

TOKIC M. (2010). Adaptive epsilon-greedy exploration in reinforcement learning based on value difference. In *Proc. of the 33rd KI*, p. 203–210 : Springer-Verlag.

TOKIC M. & PALM G. (2011). Value-difference based exploration : Adaptive control between epsilon-greedy and softmax. In *Proc. of the 34th KI*, p. 335–346 : Springer-Verlag.

UTHER W. (2010). Markov decision processes. In C. SAMMUT & G. WEBB, Eds., *Encyclopedia of Machine Learning*, p. 642–646. Springer.

VALIANT L. (1984). A theory of the learnable. *Commun. ACM*, **27**(11), 1134–1142.

WIERING M. & OTTERLO M. (2012). Reinforcement learning and markov decision processes. In *Reinforcement Learning : State-of-the-Art*, p. 3–42. Springer.