RIVERSAIL ET C-UCRL2

PROJET DE REINFORCEMENT LEARNING

Antoine d'Andigné **François-Marie Frank**

ABSTRACT

L'utilisation des propriétés d'équivalence dans l'espace d'états d'un Processus de Décision Markovien (MDP) a été étudiée dans plusieurs études. Ce travail porte sur la structure d'équivalence dans le cadre du *Reinforcement Learning* au sein de l'environnement **RiverSail** où un agent doit naviguer sur différents canaux fluviaux tout en collectant des récompenses et où un vent aléatoire sur chaque canal fait que les distributions de transitions ne sont plus connues. Nous introduisons des classes d'équivalence entre les probabilités de transition de diverses paires état-action de l'environnement. L'exploitation de ces propriétés d'équivalence existant au sein de l'espace d'états de l'environnement permet de faire mieux que l'approche classique UCRL-2.

L'implémentation en Python de notre projet peut être consultée à l'adresse suivante

`https://github.com/antoinedandi/RL_Project_RiverSail`

***K*eywords** Reinforcement Learning · MDP · Modeling · State-Action Equivalence

1 Introduction et présentation de l'environnement RiverSail

Un environnement RiverSail (voir l'image 1 pour une description visuelle d'un canal fluvial) est un Processus de Décision Markovien (MDP) discret où un agent doit naviguer sur différents canaux fluviaux tout en collectant des récompenses (récompense 1 à la fin de chaque canal, en rouge). En sortant d'un canal (en entrant dans les états en pointillés), l'agent est envoyé au hasard pour entrer dans un autre canal. Dans chaque canal fluvial, la navigation est similaire, sauf pour la présence d'un vent constant tiré aléatoirement, indiqué dans le coin supérieur gauche de chaque canal, dont la direction est inconnue : Lorsqu'un bateau en position rose se déplace dans la direction de la flèche, il se retrouve dans les états gris, ombrés selon leur niveau de probabilité. Tous les états bleu foncé d'une même région se comportent de la même manière. Ici, les masses exactes de probabilité de l'état suivant sont inconnues du marin en raison du vent inconnu, mais il en connaît parfaitement la structure.

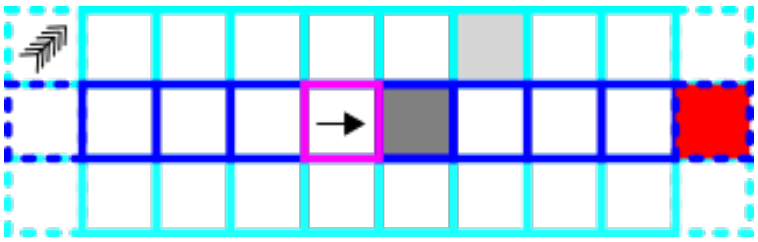


Figure 1: Illustration de l'environnement RiverSail

La tâche consiste à tester différentes approches pour minimiser le regret au sens "average return" dans un tel environnement. Ici la composante "vent" inconnu créé une difficulté à surmonter (en terme de modélisation, et d'apprentissage). L'enjeu est de réussir à exploiter efficacement la similarité des transitions au sein d'une rivière pour diminuer la complexité du problème et augmenter la performance de l'agent.

Travaux connexes Ce travail est basé sur le papier [1] de Mahsa Asadi, Mohammad Sadegh Talebi, Hippolyte Bourel et Odalric-Ambrym Maillard qui introduit notamment une notion de similarité entre les probabilités de transition des

différentes paires état-action d'un MDP et qui introduit l'algorithme **C-UCRL2**, qui est un prolongement naturel de UCRL2 dans un contexte de "undiscounted MDP" et qui permet d'utiliser efficacement les propriétés d'équivalence entre les états du MDP.

2 Modélisation de RiverSail et classes d'équivalence

2.1 Choix de modélisation pour l'environnement RiverSail

Voici les choix qui ont été effectués pour modéliser l'environnement RiverSail:

- **direction du vent** - Dans chaque canal fluvial, la navigation est similaire, sauf pour la présence d'un vent constant tiré aléatoirement, indiqué dans le coin supérieur gauche de chaque canal, dont la direction est inconnue. Les directions possibles du vent sont: *Nord*, *Nord-Est*, *Est*, *Sud-Est*, *Sud*, *Sud-Ouest*, *Ouest*, *Nord-Ouest* et *Pas de vent*. La force du vent est constante entre les différents canaux et égale à 0.3.
- **effet du vent sur la navigation** - Sur l'exemple de la figure 1, pour l'action "aller à droite", il y a une forte probabilité (0.7) d'aller sur la case immédiatement à droite (gris foncé), et une plus faible égale à la force du vent (0.3) d'atteindre une case située dans la direction du vent à partir de cette case (gris clair).
- **bords de la rivière** - Pour toute case, si la transition fait sortir de la rivière, alors on modifie la transition pour faire venir le bateau sur la case de la rivière qui lui est immédiatement adjacente.
- **similarité entre les états** - D'une manière générale, l'important n'est pas tant le détail des transitions que le fait qu'elles soient toutes similaires au sein d'une rivière.

2.2 Approximation de la politique optimale par *Value Iteration*

Nous avons utilisé l'algorithme de *Value Iteration* pour calculer une approximation de la politique optimale. Ceci est nécessaire pour calculer le regret par la suite. l'image ci-dessous détaille la politique optimale trouvée pour un canal de longueur 10.

L'ordre des actions sur les lignes de la matrice est: haut, droite, bas, gauche.

```
Computing an estimate of the optimal policy (for regret)...
[[0.25 0.25 0.25 0.25]
 [0.   0.   0.   1.  ]
 [0.   1.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.   0.   1.   0.  ]
 [0.25 0.25 0.25 0.25]
 [0.25 0.25 0.25 0.25]
 [0.   0.   0.   1.  ]
 [0.   1.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.25 0.25 0.25 0.25]
 [0.25 0.25 0.25 0.25]
 [0.   0.   0.   1.  ]
 [0.   1.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [1.   0.   0.   0.  ]
 [0.25 0.25 0.25 0.25]]
**********************************************
```

Figure 2: Optimal policy trouvée par value iteration for un canal de longueur 10

Intuitivement, on peut se satisfaire de cette politique trouvée par l'algorithme de *Value Iteration*: D'une manière générale, le marin à intérêt à naviguer vers la droite, sauf quand il se rapproche du bout du canal et dans ce cas il

faut qu'il revienne sur la ligne du milieu où se situe la récompense avant de continuer vers la droite. Pour les états en pointillés, l'action choisie n'a pas de conséquence (d'où les proba 0.25) puisque le marin change de canal fluvial.

2.3 Similarité & classes d'équivalence

Nous présentons maintenant la notion de structure d'équivalence telle que introduite dans [**1**]. Nous introduisons d'abord une notion de similitude entre les paires état-action du MDP :

Définition: ***Paires état-action similaires*** La paire (s', a') est dite ϵ-similaire à la paire (s, a), pour $\epsilon = (\epsilon_p, \epsilon_\mu) \in \mathbb{R}_+^2$, si $\|p(\sigma_{s,a}(\cdot)|s, a) - p(\sigma_{s',a'}(\cdot)|s', a')\|_1 \leq \epsilon_p$ et $|\mu(s, a) - \mu(s', a')| \leq \epsilon_\mu$,
où $\sigma_{s,a} : \{1, \ldots, S\} \to \mathcal{S}$ désigne une permutation entre les états telle que $p(\sigma_{s,a}(1)|s, a) \geq p(\sigma_{s,a}(2)|s, a) \geq \ldots \geq p(\sigma_{s,a}(S)|s, a)$. On appelle $\sigma_{s,a}$ la ***permutation de profils*** pour (s, a), et on appelle $\sigma = (\sigma_{s,a})_{s,a}$ l'ensemble des permutations de profils de toutes les paires.

La notion de similarité introduite ci-dessus donne naturellement lieu à une *partition* de l'espace $\mathcal{S} \times \mathcal{A}$ comme le montre la définition suivante :

Définition: ***Classe d'équivalence*** La $(0, 0)$-similarité est une relation d'équivalence et induit une partition canonique de $\mathcal{S} \times \mathcal{A}$,. Nous appelons une telle partition canonique ***classes d'équivalence*** $\mathcal{C}$. Nous définissons aussi $C := |\mathcal{C}|$.

2.4 Définition des classes d'équivalence au sein de RiverSail

La figure 3 montre les classes d'équivalence dans un environnement RiverSail, où les paires état-action d'une même couleur sont équivalentes à une permutation près au sens de la définition ci-dessus.

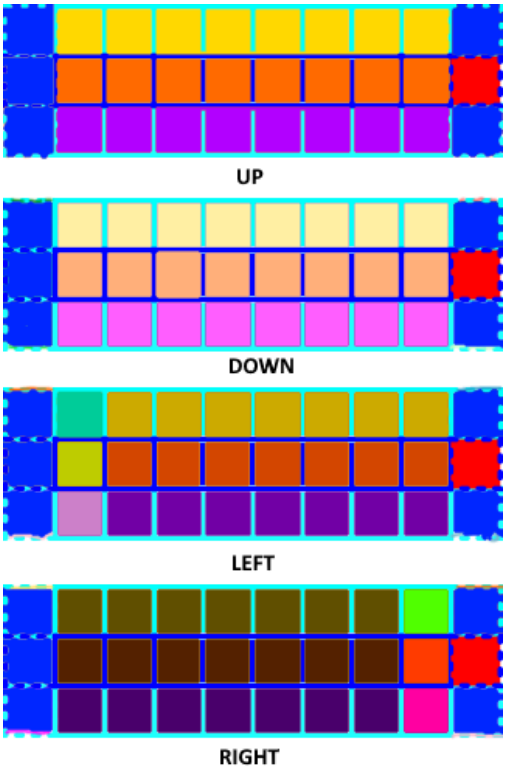


Figure 3: Classes d'équivalence état-action au sein de RiverSail

Il y a au total **20 classes d'équivalence**, et ce même si la longueur de la rivière augmente ! La partie suivante présente et applique l'algorithme **C-UCRL2** introduit dans [1] afin de mettre à profit la similarité des transitions au sein d'une rivière.

3 C-UCRL2

3.1 UCRL2

La méthode C-UCRL2 s'appuie - comme son nom l'indique assez bien - sur UCRL2 proposé par Jaksch et al. (2010) [2], lui-même une variante de UCRL Auer and Ortner [3] (2007), adoptant le principe d'optimisme face à l'incertitude. En effet, considérant les observations faites, on peut définir un ensemble $\mathscr{M}$ des MDP plausibles, puis choisir un MDP optimiste dans cet ensemble pour en tirer une politique optimale.

Soit $g_\pi^{\mathcal{M}}$ le gain moyen pour une politique π dans un MDP $\mathcal{M}$, c'est à dire la limite à l'infini du rapport entre le gain engendré par cette politique et le temps. On chercher alors :

$$\pi^+ = \underset{\pi:\mathcal{S}\to\mathcal{A}}{\operatorname{argmax}} \max\{g_\pi^{\mathcal{M}} : \mathcal{M} \in \mathscr{M}\}$$

Évidemment, l'ensemble des MDP plausibles lui-même varie au cours du calcul, on n'a donc plus $\mathscr{M}$, mais $\mathscr{M}_t$:

$$\mathscr{M}_t = \Big\{(\mathcal{S}, \mathcal{A}, \tilde{p}, \tilde{\mu}) : \forall (s,a) \in \mathcal{S}\times\mathcal{A},\ \ |\mu_{N_t(s,a)}(s,a) - \tilde{\mu}(.|s,a)| \le \tilde{b}_t^H(s,a,\frac{\delta}{2SA})$$
$$\textbf{et}\ \ \|p_{N_t(s,a)}(.|s,a) - \tilde{p}(.|s,a)\|_1 \le \tilde{b}_t^W(s,a,\frac{\delta}{2SA})\Big\}$$

où μ_n et p_n sont les récompense moyenne et distribution empiriques construit à partir de n observations, $N_t(s,a)$ le nombre d'observations de la paire (s,a) au temps t, en enfin $\tilde{b}_t^H$ et $\tilde{b}_t^W$ sont deux bornes s'appuyant sur les inégalités d'Hoeffding et Weissman.

Le calcul de π_t^+ est effectué à l'aide de la méthode EVI (Extended Value Iteration), non pas à tout temps t mais au début d'épisodes, un épisode commençant lorsque que suffisamment de nouvelles observations ont été faites depuis l'épisode précédent. On joue ensuite en suivant π_t^+ jusqu'au début de l'épisode suivant.

3.2 EVI

La procédure d'itération de valeur définit pour chaque état s une suite de valeurs $(u_n)_n$ et une suite de politiques suivant une logique gloutonne par rapport à ces valeurs. En commençant avec $u0 = 0$, la relation de récurrence est définie comme suit :

$$u_{n+1}(s) = \max_{a\in\mathcal{A}} \mu(s,a) + \sum_{s'\in\mathcal{S}} p_{N_t(s,a)}(s'|s,a)\ u_n(s')$$

$$\pi_{n+1}(s) \in \underset{a\in\mathcal{A}}{\operatorname{argmax}}\, \mu(s,a) + \sum_{s'\in\mathcal{S}} p_{N_t(s,a)}(s'|s,a)\ u_n(s')$$

3.3 C-UCRL

La version de C-UCRL que nous avons implémentée suppose les classes d'équivalence $\mathcal{C}$ ainsi que les permutations de profils $\sigma_{s,a}$ sont connues, mais il existe également des versions qui les estiment au vol.

L'idée de base est de regrouper les observations des toutes les paires état-action faisant partie d'une même classe, à commencer par le nombre d'observations : $N_t(c) = \sum_{s,a\in c} N_t(s,a)$.

On peut alors définir la distribution empirique :

$$p^\sigma_{N_t(c)}(s'|c) = \frac{\sum_{s,a\in c} N_t(s,a) p_{N_t(s,a)}(\sigma_{s,a}(s')|s,a)}{\max(1, N_t(c))}$$

qui se calcule très facilement à partir des transitions observées.

On fait de même avec le gain :

$$\mu_{N_t(c)}(c) = \frac{\sum_{s,a\in c} N_t(s,a)\mu_{N_t(s,a)}(s,a)}{\max(1, N_t(c))}$$

On redéfinit ensuite l'ensemble de MDP à parcourir pour trouver une politique :

$$\mathscr{M}_t = \Big\{(\mathcal{S}, \mathcal{A}, \tilde{p}, \tilde{\mu}) : \forall c \in \mathcal{C} \forall (s,a) \in c, \ \ |\mu_{N_t(c)}(c) - \tilde{\mu}(.|s,a)| \leq \tilde{b}^H_{N_t(c)}(\frac{\delta}{2C})$$
$$\textbf{et} \ \ \|p_{N_t(c)}(\sigma^{-1}_{s,a}(.)|c) - \tilde{p}(.|s,a)\|_1 \leq \tilde{b}^W_{N_t(c)}(\frac{\delta}{2C})$$
$$\textbf{et} \ \ \forall (s,a),(s',a') \in c, \tilde{\mu}(.|s,a)| = \tilde{\mu}(.|s',a'), \ \tilde{p}(.|s,a) = \tilde{p}(.|s',a')|\Big\}$$

On se ramène donc à des bornes par classe, et on chercher des distributions constantes au sein d'une même classe.

Le temps de changement d'épisode est aussi modifié pour prendre en compte les classes :

$$t_{k+1} = \min\Big\{t > t_k; \exists c \in \mathcal{C} : n_{t_k:t}(c) \geq \max(N_{t_k}(c), 1)\Big\}$$

où $n_{t:t'}$ est le nombre d'observations entre t et t'.

Pour expliciter les bornes :

$$\tilde{b}^H_{N_t(c)}(\delta) = \sqrt{\frac{(1+1/n)\log(2\sqrt{n+1}/\delta)}{N_t(c)}}$$

$$\tilde{b}^W_{N_t(c)}(\delta) = \sqrt{\frac{2(1+1/n)\log(2\sqrt{n+1}(2^K-2)/\delta)}{N_t(c)}}$$

avec $K \geq |Support(p(.|s,a))|$, en l'occurrence K = 2.

Dans le cas où σ n'est pas connu en avance, on peut utiliser à la place une estimation en prenant n'importe quelle permutation de profils correspondant aux observations.

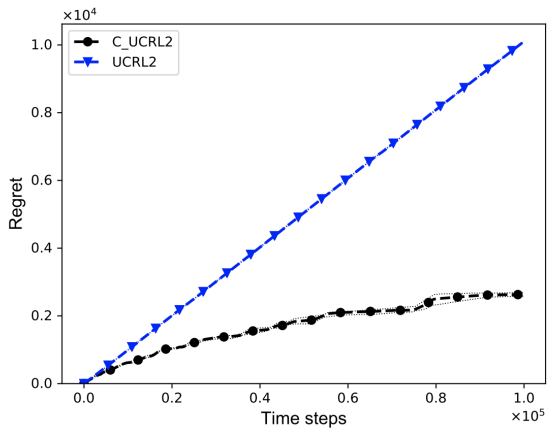
Enfin, si on ne connaît pas non plus les classes, il est possible de les estimer par clustering, ce qui prend évidemment du temps. C'est pourquoi cette méthode doit être réservée aux cas où $C << SA$.

4 Expérience

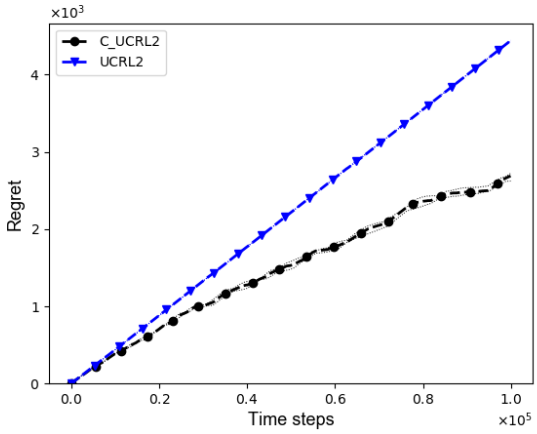
Nous avons choisi de comparer l'efficacité de UCRL2 à C-UCRL2 afin de mettre en lumière l'intérêt de regrouper les couple état-action similaires par classe d'équivalence. Les paramètres de l'expérience réalisés sont les suivants:

- **Longueur des canaux** - Nous avons choisi de générer des canaux de taille raisonnable - *20* - mais néanmoins suffisamment longs pour que le bénéfice d'utiliser des classes d'équivalence puisse se faire ressentir.
- **Horizon temporel** - Nous avons choisi 100,000 itérations comme horizon de temps. Cela permet de faire les simulations dans un temps raisonnable tout en obtenant de bons résultats
- **Paramètre delta** - Nous avons choisi de garder le paramètre $delta$ à sa valeur par défaut qui est de 0.05

On observe effectivement une amélioration nette du regret en utilisant C-UCRL2.



(a) RiverSail de longueur 10



(b) RiverSail de longueur 20

References

[1] Mahsa Asadi et al. "Model-Based Reinforcement Learning Exploiting State-Action Equivalence". In: *ACML*. 2019.

[2] Peter Auer, Thomas Jaksch, and Ronald Ortner. "Near-optimal Regret Bounds for Reinforcement Learning". In: *Advances in Neural Information Processing Systems 21*. Ed. by D. Koller et al. Curran Associates, Inc., 2009, pp. 89–96. URL: `http://papers.nips.cc/paper/3401-near-optimal-regret-bounds-for-reinforcement-learning.pdf`.

[3] Peter Auer and Ronald Ortner. "Logarithmic Online Regret Bounds for Undiscounted Reinforcement Learning". In: *Advances in Neural Information Processing Systems 19*. Ed. by B. Schölkopf, J. C. Platt, and T. Hoffman. MIT Press, 2007, pp. 49–56. URL: `http://papers.nips.cc/paper/3052-logarithmic-online-regret-bounds-for-undiscounted-reinforcement-learning.pdf`.