Barbier Yann
Razafimbelo Fanoa M2 AMIS

Guidelines for Action Space Definition in Reinforcement Learning-Based Traffic Signal Control Systems

Maxime Treca, Julian Garbiso, Dominique Barth

Introduction

L'article que nous allons étudier a pour objet le contrôle des feux de circulation, appelés Traffic Signal Control (TSC).
La congestion de la circulation est un problème qui peut avoir des répercussions économiques, sociales et environnementales. L'article vise à présenter une étude du contrôle des feux de signalisation par apprentissage par renforcement et à donner des indications sur la définition de l'espace d'action d'un agent contrôlant ces feux. Les systèmes TSC présentent un modèle bien adapté pour l'utilisation de méthodes d'apprentissage par renforcement. En effet, ces systèmes suivent un modèle général qui place un agent, ici le contrôleur des feux de signalisation, dans un environnement. L'agent observe l'environnement, ici les temps d'attente sur une intersection, et encode l'état actuel du trafic dans un état en utilisant les informations observées à l'état actuel. En considérant l'état actuel de l'environnement, l'agent choisit une action qui détermine le prochain état du trafic. Une fois que l'agent a fait son action, l'environnement change et l'agent reçoit une récompense qualifiant l'action précédente compte tenu de l'état du trafic.

Modélisation

Cet article vise à comparer deux types de contrôleur pour la définition de l'espace d'action de l'agent : « step-based » et « phase-based » dont nous expliquerons le fonctionnement par la suite. Tout d'abord la situation que l'article étudie est le cas d'une intersection avec quatre routes provenant de directions différentes : Nord - Sud et Est - Ouest.
Les feux de signalisation de chaque route sont gérés par un contrôleur qui utilise un cycle de signalisation.. Le contrôleur dispose de trois phases $\Psi = \{\psi1, \psi2, \psi T\}$. La phase $\psi1$ met les feux au vert sur les approches Nord - Sud de l'intersection et la phase $\psi2$ sur les approches Est - Ouest. La phase ψT sert de transition entre ces deux phases et met tous les feux au rouge.
Le contrôleur régule le trafic selon un cycle de signalisation Φ, qui suit le modèle suivant $\Phi = \psi1, \psi T , \psi2, \psi T , \psi1\ldots$
Chaque phase de Φ est bornée par une durée minimum et maximum. Les deux types d'agents étudiés dans l'article sont « step-based » et « phase-based ».
Un contrôleur phase-based est un type de contrôleur qui définit la durée de la phase actuelle et ne peut plus la modifier. Le contrôleur lance la phase pour la durée qu'il définit au préalable puis passe automatiquement à la prochaine phase.
Un contrôleur step-based au contraire donne tout d'abord la durée minimale de la phase actuelle puis décide s'il faut prolonger la phase actuelle de n étapes. Le contrôleur peut ensuite prolonger la phase jusqu'à une durée maximum.

Apprentissage par renforcement

L'article utilise une méthode Q-learning pour le contrôle des feux de signalisation.
Agent : L'agent est ici le contrôleur chargé de contrôler les feux de signalisation des approches de l'intersection.
Etat : Un état pour l'apprentissage est défini comme un vecteur de taille 4 $st = (\varphi_i, d_i, c_{1,t}, c_{2,t})$ avec φ_i la phase actuelle, d_i la durée de la phase actuelle, $c_{1,t}$ la congestion actuelle associé à la phase et $\psi1$, $c_{2,t}$ la congestion actuelle associé à la phase $\psi2$.
La congestion d'un état à un temps t est mesurée par la formule :

$$c_{i,t} = (\sum_{a \in \psi_i} n_{a,t}) \bmod 3$$

Où $n_{a,t}$ représente le nombre de véhicule sur l'approche a à l'instant t.

Espace d'action : L'espace d'action est défini différemment selon le type d'agent : pour un agent phase-based, l'espace d'action est défini comme l'intervalle $[0, d_{max} - d_{min}]$ qui correspond à la durée que l'agent peut définir pour la phase actuelle. Pour un agent step-based, l'espace d'action est [0, 1] où 0 correspond à une extension de la phase actuelle de N étapes et 1 fait passer l'environnement à la phase suivante du cycle.

Récompense : La récompense de l'agent à un temps t est définie comme la différence entre le temps d'attente total de chaque véhicule sur chaque approche calculée au dernier point de décision et le temps d'attente total de chaque véhicule sur chaque approche calculée au temps t actuel.
Q-learning : La méthode d'apprentissage utilisée par l'agent est le Q-learning. Cette méthode consiste à mettre à jour les éléments d'une table Q. Chaque couple (état, action) pour l'agent est associé à une valeur qui correspond à la qualité de l'action étant donné l'état actuel. Pour un discount $\gamma \in [0, 1]$ et un taux d'apprentissage $\alpha \in [0, 1]$, la table Q de l'agent est mise à jour selon la règle suivante :

$$Q(s_t, a_t) \leftarrow (1-\alpha_t)Q(s_t, a_t) + \alpha_t(r_t + \gamma \max_a Q(s_{t+1}, a))$$

Dans cette formule, s_t correspond à l'état actuel au temps t, s_{t+1} au prochain état et r_t à la récompense calculée au temps t. L'article utilise un taux d'apprentissage $\alpha = \max(1/v(st, at), 0.05)$ où v(st, at) correspond au nombre de fois où l'agent a visité le couple état/action (S_t, a_t).

L'agent suit une politique ε-gloutonne c'est-à-dire qu'à un point de décision, l'agent choisit une action associée à la plus grande valeur dans la table Q avec une probabilité 1- ε (exploitation), sinon, il choisit une action aléatoire dans son espace d'action avec une probabilité ε (exploration).

Résultats de l'article

a) Environnement de simulation

Un scénario dure 10 000 étapes et représente environ 3 heures de trafic en temps réel. Les durées minimum et maximum d'une phase sont fixées à 5 et 45 respectivement. La phase de transition dure 5 étapes.

La génération du trafic suit deux processus de Poisson. Le premier processus est de taux lambda et est associé aux arrivées sur les quatre approches. Le deuxième processus suit un paramètre τ qui assigne des arrivées sur les approches Nord - Sud de l'intersection. Avoir deux processus permet de simuler des modèles d'intersection composés d'une route principale et de routes secondaires.

b) Analyse des résultats

Le premier résultat de l'article est obtenu en comparant les deux modèles d'agent en fonction du taux τ.

Les résultats de ce tableau montrent pour chaque type d'agent, le temps d'attente moyen des véhicules selon le taux d'arrivées sur les routes Nord - Sud. On peut voir que le contrôleur phase-based est moins efficace par rapport au controller step-based.

τ	Fixed	Phase	Step (Best)	Step (Worst)
0.0	3.617	2.672	2.053	2.473
0.1	4.070	2.746	1.956	2.595
0.2	4.603	3.070	1.977	2.570
0.3	7.773	4.582	2.032	2.531
0.4	6.807	5.773	2.088	2.216
0.5	18.329	3.240	1.994	2.473

Table 1: Average vehicle waiting time after convergence per agent type and traffic parameter τ (in 10^3 seconds).

Le contrôleur step-based est le meilleur type pour tous les scénarios testés et les résultats montrent que ce type d'agent est robuste dans le sens où les performances sont à peu près similaires pour tous les taux d'arrivées.

La comparaison des performances sur la figure 1 montre que les mauvaises performances de l'agent phase-based ne sont pas dûes à son incapacité à apprendre mais plutôt à sa nature moins flexible par rapport à l'agent step-based.

L'article en vient à sa première instruction pour la définition du type d'agent : il faut toujours privilégier un agent de type step-based pour contrôler des feux de signalisation.

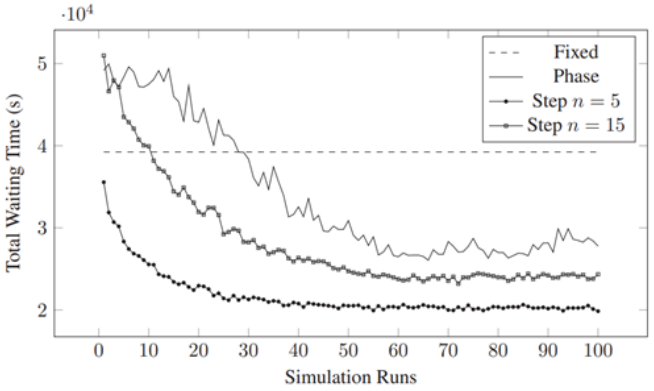


Figure 1: Vehicle waiting time per agent type, $\tau = 0.10$.

Le deuxième résultat obtenu par les auteurs est observable dans ce tableau :

τ	$n = 1$	$n = 5$	$n = 10$	$n = 15$	$n = 20$
0.0	0	4.86	10.29	22.37	27.81
0.1	0	4.17	7.20	23.96	30.15
0.2	0	0.45	5.49	22.68	31.03
0.3	3.04	0	4.38	11.02	26.39
0.4	9.53	0	2.74	11.09	7.84
0.5	22.12	0.56	0	13.60	3.33

Table 2: Percentage difference with respect to the optimum average vehicle waiting time (marked as 0) for step-based methods by action interval value n and traffic scenario τ

On remarque des différences de performances selon la valeur n et τ. Les résultats montrent que la valeur optimale pour la variable n change en fonction du taux de trafic, l'article en conclut qu'une haute interaction de l'agent avec son environnement ne garantit pas forcément de meilleurs résultats. La deuxième conclusion est que les valeurs de n et de τ sont corrélées.

Voici donc la deuxième recommandation : des intervalles de décision courts sont préférable pour un trafic uniforme, dans le cas non uniforme de plus longs intervalles sont conseillés.

Le dernier résultat concerne les valeurs conseillées à choisir pour la valeur de n.

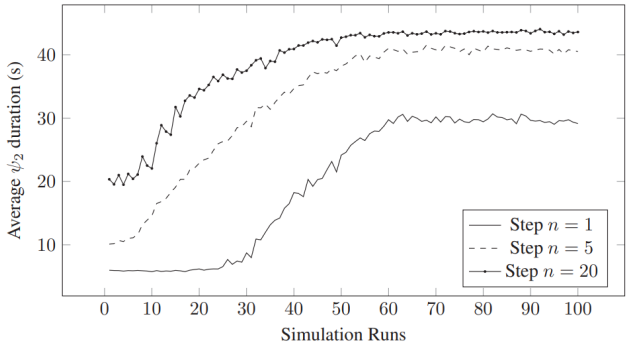


Figure 2: Evolution of phase ψ_2 average duration per simulation iteration, $\tau = 0.50$.

Comme on peut le voir dans ce graphique, l'article se place dans une situation où le trafic est généré avec un taux τ égale à 0.50. Ils observent que dans le cas d'un trafic non uniforme, même si toutes les méthodes apprennent à donner de plus longues durées pour la phase ψ2, les méthodes ayant une valeur de n faible convergent plus lentement que des valeurs n plus grandes. Cela peut être dû à la politique ε-gloutonne de l'agent.

Voici donc la dernière recommandation : il faut privilégier des valeurs de n faibles entre 5 et 10.

Programmation

Notre programme contenu dans le notebook TrafficSignalControl.ipynb vise à reproduire une simulation simplifiée de l'article avec un agent de type step-based. L'implémentation se divise en deux parties : tout d'abord l'implémentation de l'environnement puis l'implémentation de l'agent qui contrôle les feux de signalisation.

a) Environnement

Tout d'abord, l'environnement est constitué de deux classes. La classe Vehicule, caractérisée par une variable waiting_time qui sera mise à jour à chaque pas de temps et qui va par la suite, nous permettre de calculer le temps d'attente. La classe principale de l'environnement est la classe IntersectionEnvironment qui simule l'intersection. Cette classe prend en paramètre la durée minimum et maximum d'une phase et du taux τ pour la génération de trafic. Plusieurs variables sont nécessaires pour faire une implémentation simplifiée de l'environnement de l'article dont l'intervalle n qui correspond au temps que le contrôleur peut ajouter à la phase actuelle.

Le cycle des phases est contenu dans un tableau et la durée initiale de chaque phase est contenue dans un dictionnaire.

Les listes des véhicules sur les 4 routes de l'intersection sont contenues dans un dictionnaire. Chaque fois qu'un véhicule arrive, il sera ajouté à la liste de son approche associée.

La phase actuelle est repérée par son indice qui permettra de le retrouver dans le tableau contenant les cycles des phases.

La méthode step(self, action) de l'environnement prend en paramètre l'action et effectue un pas de temps dans la simulation, comme précisé par l'article la simulation s'arrête lorsque le temps actuel vaut 10 000.

Cette méthode génère le trafic par l'appel à la méthode generate_traffic de l'environnement qui prend en paramètre le taux τ. Cette méthode permet de simuler l'arrivée de voitures dans l'intersection. Comme précisé dans l'article, cette génération suit deux processus d'arrivée poisson, donc on ajoute une voiture dans la liste de l'approche associée en fonction des taux d'arrivées.

La méthode step met à jour le trafic, c'est-à-dire qu'elle s'occupe des départs des voitures. L'article ne précise pas combien de voitures peuvent partir lorsque le feu est vert pour un pas de temps. Nous avons donc initialisé cette variable de nombre de départ par 1 par pas de temps. Cette méthode, selon la phase actuelle, supprime le premier élément de la liste de véhicules sur les approches associées. Par exemple, si la phase actuelle est $\psi 1$, cela signifie que le feu est vert sur les approches Nord - Sud, la méthode enlève donc la première voiture sur la liste des voitures des approches nord et sud. Le nombre de départ de véhicule est incrémenté ce qui nous permet à la fin de calculer le temps d'attente moyen pour chaque phase.

La prochaine étape de la méthode step de l'environnement consiste à mettre à jour le temps d'attente des voitures, si la phase actuelle est $\psi 1$, cela signifie que les feux sur les approches Ouest et Est sont au rouge, donc la variable waiting_time des voitures sur la liste de voitures de ces approches est incrémenté de 1.

Le temps d'attente cumulé entre toutes les approches est calculé à chaque étape ce qui nous permet de calculer la récompense de l'agent. Lorsque l'on se retrouve à un point de décision, l'environnement garde en mémoire le temps d'attente cumulé actuel et, selon l'action de l'agent, modifie la durée de la phase actuelle, donc si l'action de l'agent est 0 la phase est incrémentée de n étapes sinon on passe à la phase suivante du cycle des phases.

Après cette phase de décision, la congestion au temps t est calculée selon la formule présentée par l'article.

La récompense est ensuite calculée comme la différence entre le temps d'attente cumulé à la dernière phase de décision gardé en mémoire par l'environnement et le temps d'attente cumulé actuel au temps t.

Cette méthode retourne ensuite l'état composé de la phase actuelle dans le cycle de phase, la durée actuelle de la phase, et les congestions associées aux phases $\psi 1$ et $\psi 2$.

b) Agent

La classe Agent représente le contrôleur step-based, cette classe est caractérisé par plusieurs variable : les variables alpha, gamma et epsilon en paramètre permettent de mettre à jour la table Q contenue dans un dictionnaire où les clés sont le couple (état, action) et la valeur associée la valeur calculée par la règle de mise à jour de l'article.

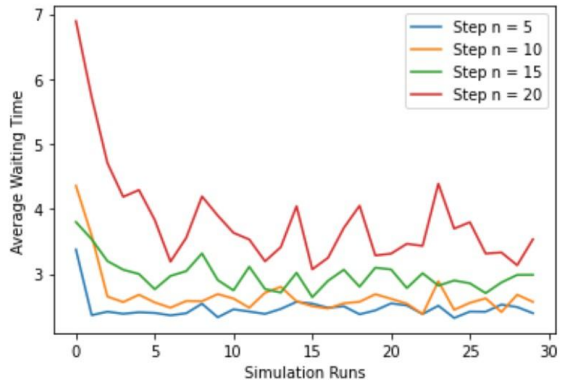
Cette classe Agent possède une méthode action qui prend en paramètre un état. L'agent suit une politique ε-gloutonne c'est-à-dire qu'à un point de décision, l'agent choisit une action associée à la plus grande valeur dans la table Q avec une probabilité 1 - ε ou sinon choisit une action aléatoire dans son espace d'action avec une probabilité ε. Pour obtenir la plus grande valeur dans la table Q selon l'état, on cherche pour l'état actuel l'indice de la plus grande valeur Q dans la table. La méthode updateQ suit la règle de mise à jour de l'article et calcule la nouvelle valeur associée à la clé (etat, action) du dictionnaire en fonction de alpha, gamma et de la récompense.

c) Boucle principale

La méthode simulation permet de lancer la simulation pour un nombre défini d'itérations. Cette méthode initialise tout d'abord l'environnement et l'agent puis lance la boucle principale jusqu'au nombre de simulations défini. Au début de chaque itération, l'environnement est réinitialisé, puis l'agent choisit une action en suivant sa politique ε-gloutonne. En fonction de l'action, l'environnement appelle sa fonction step pour mettre à jour l'environnement d'un pas de temps et cela est répété en boucle jusqu'à ce que t = 10 000, le temps auquel un scénario s'arrête. Le temps d'attente moyen pour chaque phase est ensuite calculé avant de recommencer la simulation pour une nouvelle itération.

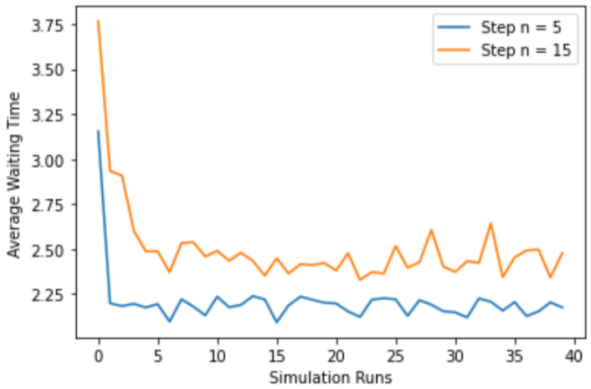
Résultats

Pour alpha = 0.01, gamma = 0.5, epsilon = 0.05 et τ = 0.30 :



Le premier graphique nous permet de comparer les performances de la méthode Q-learning pour différentes valeurs de n step. Nous avons testé les performances pour des valeurs de n step = 5, 10, 15 et 20. Comme décrit dans l'article les n steps entre 5 et 10 présente les meilleures résultats proches de l'optimale

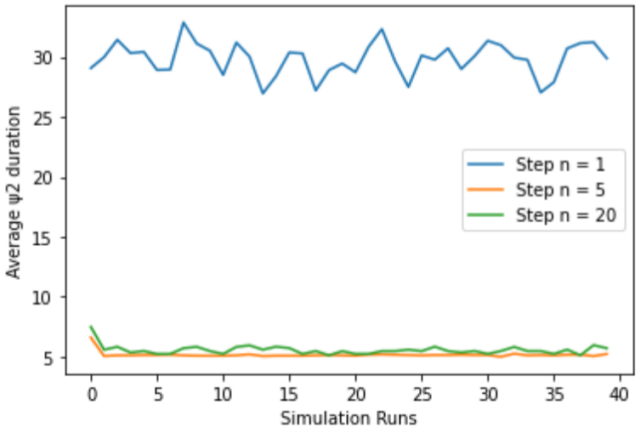
Pour alpha = 0.01, gamma = 0.5, epsilon = 0.05 et τ = 0.10 :



Nous avons ensuite testé la simulation pour un taux d'arrivées plus faible sur les approches Nord - Sud, ici pour τ = 0.10. Comme montré dans l'article pour un τ différents le contrôleur n step est robuste dans la mesure où les performances sont similaires pour des taux d'arrivées différents.

Le dernier résultat dans notre programme est l'analyse de la durée moyenne de la phase ψ2 pour un taux τ = 0.5. Or on sait que d'après l'article λ + τ = 0.5, donc si τ = 0.5, alors la valeur de λ sera nulle. Si λ est nulle cela signifie qu'il n'y a aucune arrivée sur les approches Est et Ouest de l'intersection, on peut donc imaginer que la phase ψ2 se verra toujours attribuer la durée minimale de phase . L'article arrive tout de même à présenter un graphique où la durée ψ2 converge vers la durée maximale d'une phase après plusieurs simulations (voir Figure 2).

En essayant de reproduire ce graphe avec comme paramètres alpha = 0.01,gamma = 0.5, epsilon = 0.05 et τ = 0.50, nous obtenons la figure suivante :



Contrairement à l'article, nous obtenons pour une valeur n step = 1 une moyenne autour de 30. Pour n step = 5 et 20, la durée moyenne de la phase est autour de 5, ce qui confirme notre première intuition. Comme τ = 0.5, on a λ = 0, il n'y a donc aucune arrivée de voitures sur les approches Est et Ouest de l'intersection. On peut supposer que l'agent ne voit donc pas de bénéfice à prolonger cette phase s'il n'y a pas de congestion sur ces approches.

Bibliographie

Lien du repository github : https://github.com/YannBgit/OptimisationQLearningCirculationIntersection

Q-learning based Routing Scheduling For a Multi-Task Autonomous Agent, Omar Bouhamed, Hakim Ghazzai, Hichem Besbes and Yehia Massoud

*Q-Learning for Adaptive Traffic Signal Control Based on Delay Minimization Strategy,*Lu Shoufeng, Liu Ximin, Dai Shiqiang, 2008

Reinforcement Q-Learning from Scratch in Python with OpenAI Gym, Satwik Kansal, Brendan Martin

NumPy https://numpy.org/doc/stable/reference/index.html
matplotlib https://matplotlib.org/stable/api/index