UE Projet

Jonathan GRAFF

Janvier 2020

Sommaire

I) Introduction

I.1) Contexte

Altran est une entreprise spécialisée dans le conseil, dans l'ingénierie et dans la recherche et développement. Altran accompagne les entreprises du concept à l'industrialisation, pour développer de nouveaux produits et services, et intervient depuis plus de 35 ans auprès des grands acteurs de nombreux secteurs : Automobile, Aéronautique, Spatial, Transport, Energie, Industrie, Communications, Electronique, Logiciel, Internet, Finance, Secteur Public ... Le groupe est présent dans plus de 30 pays et compte plus de 50000 employés.

Mon projet se situe dans la branche nommée Altran Research. Altran Research est un département recherche interne à la société Altran. Le projet de recherche Sinbad se focalise sur l'efficacité énergétique dans l'énergie et fait partie du programme Future of Energy, un des sept programmes de recherche du département. Le projet Anagreen est hébergé au sein du projet Sinbad, c'est une collaboration entre Altran et ArcelorMittal financé par l'Ademe. L'objectif du projet est de développer un outil d'aide à la décision pour la récupération de la chaleur fatale dans l'industrie.

I.2) Objectifs

Mon objectif personnel ce semestre, dans le cadre de l'Unité d'Enseignement Projet, est dans un premier temps de m'approprier le sujet, de comprendre les enjeux du problème des réseaux de chaleur, ce domaine m'étant inconnu. Dans un second temps, de me familiariser avec les algorithmes à renforcement profond, afin d'en implémenter un exemple sur un problème-jouet.

Ce rapport comportera donc une première partie théorique sur le problème des réseaux d'échangeurs de chaleur, de leur utilité, ainsi que des difficultés rencontrées. Dans une deuxième partie, j'expliquerai comment fonctionnent les algorithmes à renforcement profond, et dans une troisième partie, celle-ci pratique, je parlerai de l'implémentation d'un tel algorithme.

II) Problèmes des réseaux d'échangeur thermiques

II.1) Motivation

La chaleur fatale est la chaleur résiduelle issue d'un procédé et non utilisée par celui-ci. Une étude de l'ADEME de 2017 [1] a montré que cette source d'énergie, si elle n'était pas perdue,

représenterait 16% de la consommation énergétique nationale et représente 36% de la consommation industrielle. Mais d'après ce rapport, ce nombre est sûrement sous-évalué car ce nombre a été obtenu en ne comptant que les procédés les plus énergivores (comme les fours, séchoirs et chaudières), les rejets les plus accessibles, et les niveaux de température les plus efficaces (au-delà de 100 ° C). Les industries possédant le plus de chaleur fatale sont les industries chimiques, les métaux et matériaux, et l'agro-alimentaire. Il s'agit donc d'un enjeu majeur, aussi bien environnemental qu'économique, que de ne pas gaspiller cette énergie.

Historiquement, jusque dans les années 1970, les entreprises se sont développées au fur et à mesure des années sans se préoccuper de cette source d'énergie perdue, le coût de l'énergie étant particulièrement favorable à l'époque, mais aussi la préoccupation écologique et notamment du réchauffement climatique étant alors très faible. Aujourd'hui, il y a une prise de conscience plus importante de ce sujet, l'énergie est plus chère qu'auparavant, il y a des taxes sur les rejets polluants, et des fonds d'aide sont alloués aux entreprises souhaitant investir dans ce domaine. Pour toutes ces raisons, les entreprises se sont donc progressivement intéressées à l'importance de la récupération de la chaleur fatale.

Des années de développement interne sans tenir compte de ce problème font qu'aujourd'hui il est très difficile de revenir en arrière sur ce sujet, la configuration des complexes industriels ne s'y prêtant pas toujours. De plus, de nouvelles installations nécessiteraient parfois des coupures dans les chaînes de production, ce qui pour certaines compagnies n'est tout simplement pas envisageable, car synonyme de perte de chiffre d'affaires trop importante.

C'est dans ce contexte qu'Anagreen propose de développer un outil d'aide à la décision pour la récupération de la chaleur fatale. L'outil propose une méthodologie qui permet de recycler la chaleur interne d'un procédé pour réduire sa demande en énergie externe. Dans un premier temps, la méthodologie permet d'identifier les gisements de chaleur disponible, puis dans un deuxième temps de construire un réseau d'échangeurs pour permettre le transfert de chaleur entre flux chauds et flux froids.

II.2) Généralités sur les échangeurs thermiques

Afin de récupérer de la chaleur fatale pour une utilisation en interne, il est nécessaire de placer des échangeurs thermiques entre les flux chauds, ceux nécessitant d'être refroidis, et les flux froids, ceux nécessitant d'être réchauffés. Un échangeur thermique est un dispositif permettant de transférer de la chaleur d'un flux chaud vers un flux plus froid, sans les mélanger. Même si cela ne rentre pas directement dans le cadre du rapport, il est intéressant de comprendre quelques

bases sur les échangeurs.

Types d'échangeurs

Il existe trois principaux modes de transfert de chaleur :

- à co-courant : les deux flux ont la même direction, dans ce cas le flux froid aura nécessairement en sortie une température moins élevée que celle du flux chaud. La figure 1 représente le profil de l'évolution des températures des flux en fonction de la surface d'échange.

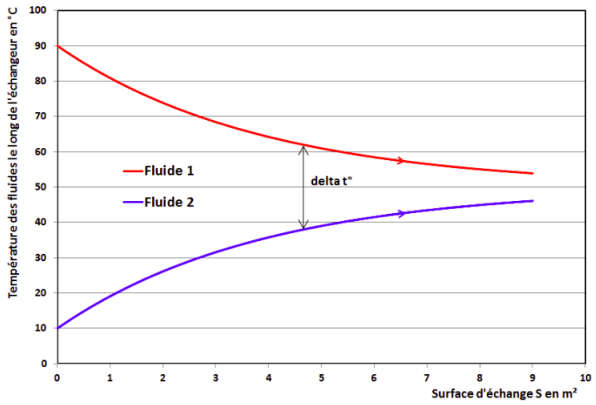


Figure 1: Evolution des températures des flux lors d'un transfert à co-courant

- à contre-courant : les flux circulent parallèlement mais dans des sens opposés, dans ce cas, le flux froid peut avoir une température de sortie supérieure à elle du flux chaud. En général, ces échangeurs sont plus efficaces. La figure 2 montre le profil de l'évolution des températures en fonction de la surface d'échange dans ce cas :

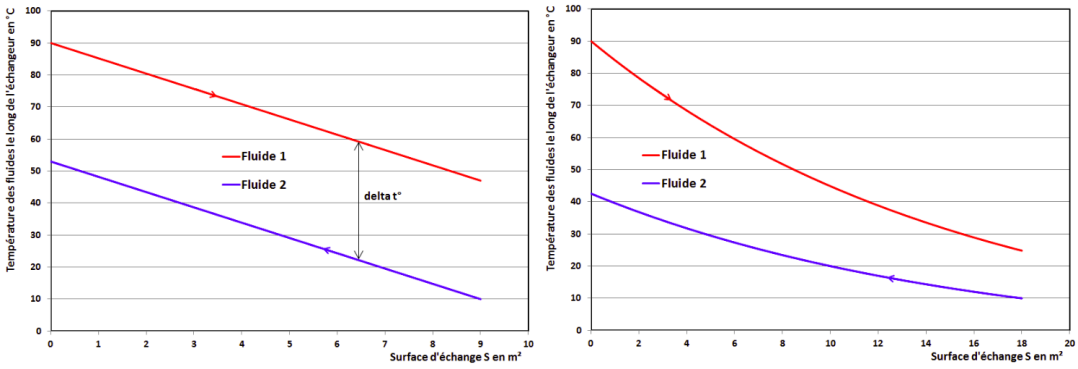


Figure 2: Evolution des températures des flux lors d'un transfert à contre-courant

- à courant croisés : les deux flux ont des directions perpendiculaires.

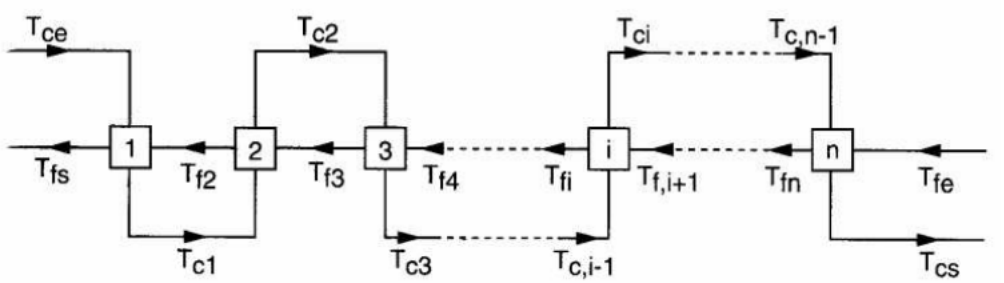
Il existe différents types d'échangeurs : à tube, à faisceau tubulaire, à spirales, à plaques, à baïonnettes, à ailettes..., et pour chacun différentes tailles et surfaces d'échange.

Chaque échangeur a également un coût différent, lié à sa surface d'échange, et donc à sa performance. Le problème du dimensionnement des échangeurs est un sujet de recherche en soi dont nous ne nous préoccuperons pas ici.

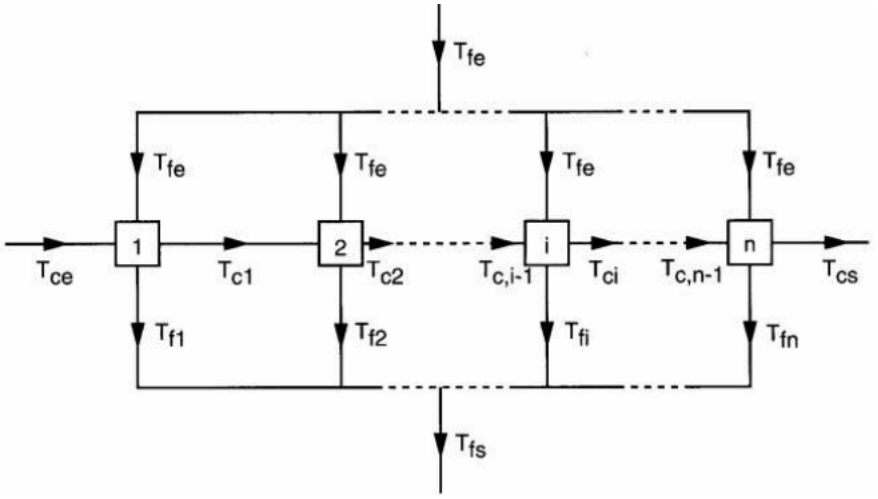
Montage des échangeurs

Ces échangeurs peuvent être montés de quatre façons différentes :

- en série :

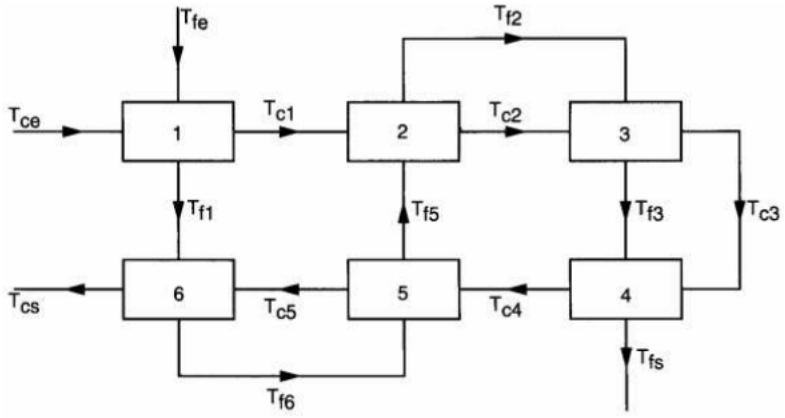


- un en parallèle et l'autre en série :



- les deux en parallèles. Cette méthode est moins utilisée que les autres pour des raisons de performances moindres.

- en maillage :



II.3) Réseaux d'échangeurs thermiques

Une entreprise possédant en général plusieurs flux chauds et plusieurs flux froids, il faudrait don placer plusieurs échangeurs thermiques afin d'optimiser le rendement énergétique. Toute la question est de savoir quels échangeurs placer (problèmes de coût et de maintenance), où les placer (endroits accessibles et optimaux), et quand les placer (l'installation pouvant nécessiter un arrêt des machines, celui-ci n'étant en général pas souhaitable).

Ce problème d'apparence simple est en fait un problème combinatoire des plus complexes. Il s'agit en fait, comme l'ont montré Furman et Sahinidis [4], d'un problème NP-difficile, c'est-à-dire que le nombre d'opérations pour résoudre ce problème croit exponentiellement en fonction du nombre de paramètres d'entrée, mais que même la vérification d'une solution est un problème de complexité exponentielle.

A titre d'exemple, supposons qu'on ait 10 flux chauds, et 10 flux froids, et qu'on souhaite placer au maximum 15 échangeurs thermiques dans notre réseau. Le premier échangeur a donc $10 \times 10 = 100$ possibilités pour être placé. Le deuxième a $11 \times 11 = 121$ possibilités pour être placé, car chaque flux sur lequel on a placé le premier échangeur est divisé en deux parties, une avant l'échangeur, l'autre après. En continuant ainsi, et en divisant à la fin par 15! pour ne pas compter les permutations des échangeurs comme une autre solution, on trouve à l'aide d'un algorithme, qu'il y a de l'ordre de 2.295×10^{24} solutions possibles. Bien sûr, tout ceci sans tenir compte d'autres paramètres, comme le type d'échangeurs, leur surface...

II.4) Etat de l'art

Pour essayer de s'approcher de la solution optimale à ce problème, plusieurs méthodes sont aujourd'hui utilisées :

- Tout d'abord il y a la méthode empirique, s'appuyant sur des avis d'experts pour déterminer une solution. Cette méthode ne prend souvent pas le problème dans son ensemble, et même si cette solution est souvent satisfaisante, elle ne garantit pas d'être optimale.
- La méthode du pincement, consiste à diviser le problème en deux sous-problèmes plus simples. Elle repose sur des principes thermodynamiques, est utilisée depuis plus de 30 ans et fait encore aujourd'hui l'objet de nombreuses études. Elle fournit en général de meilleurs résultats que la méthode empirique, mais ne trouve pas non plus nécessairement l'optimum global.
- Enfin, d'autres méthodes plus récentes, consistent à trouver des algorithmes mathématiques pour résoudre le problème. Parmi ces problèmes, on peut citer les algorithmes gloutons, des algorithmes de programmation linéaire ou non-linéaire mixte entière (MILP ou MINLP), des algorithmes génétiques, des méthodes stochastiques, de recuit simulé...

III) Proposition d'une nouvelle solution : l'apprentissage par renforcement profond

L'objectif du projet est de vérifier si une méthode utilisant les algorithmes d'apprentissage par réseaux de neurones à renforcement profond, pourrait résoudre le problème d'optimisation combinatoire.

Dans un article, l'équipe de Google Brain [2] a obtenu des résultats intéressants sur le problème combinatoire du trajet à suivre par le voyageur de commerce et l'objectif du projet Anagreen est de se demander si ces mêmes résultats pourraient être transposables sur ce problème de réseaux d'échangeurs de chaleur.

Dans le cadre de ce projet tutoré, nous allons détailler comment fonctionne ce type d'apprentissage et nous allons voir son fonctionnement sur des exemples simples.

III.1) Apprentissage par renforcement

L'apprentissage par renforcement est un procédé d'apprentissage où un agent (un robot par exemple) apprend par lui-même le comportement à adopter face à un problème pour le résoudre.

L'agent est placé dans un environnement et peut avoir plusieurs états. Par rapport à chaque état, il peut effectuer un ensemble d'actions.

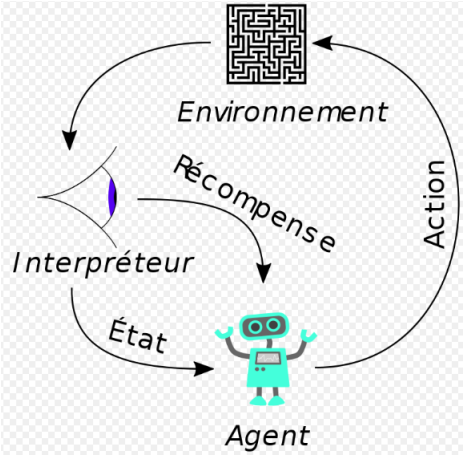


Figure 3: Schéma représentant l'apprentissage par renforcement

L'action ou les probabilités d'effectuer chaque action selon l'état dans lequel l'agent se trouve est une fonction appelée la politique. A chaque action effectuée l'agent va recevoir une "récompense", modélisée sous la forme d'un nombre réel, positif ou nul. Pendant la phase d'apprentissage, l'agent va alterner entre deux comportements :

- une phase d'exploration où l'agent va tester aléatoirement de nouvelles actions, afin de voir si celles-ci sont bonnes ou mauvaises, et actualiser sa politique en fonction.
- une phase d'exploitation, où la politique actuelle est éprouvée.

L'agent alterne entre ces deux comportements aléatoirement, et au cours de l'apprentissage, la probabilité d'exploration est en général diminuée au fur et à mesure que la politique s'approche de la politique optimale.

Il existe plusieurs variations pour ces types d'algorithmes à renforcement : basé sur un modèle ou non, à différence temporelle, Q-learning, méthodes acteurs-critiques...

III.2) Modélisation mathématique

Notations

On note $S = \{s_i\}_{i \in I}$ l'ensemble des états dans lequel l'agent peut se trouver, et $A_s = \{a_i\}_{i \in I_s}$ l'ensemble des actions possibles en se trouvant à l'état s. On note $A = \cup_{s \in S} A_s$ l'ensemble

des actions possibles. Un état est dit **terminal** si l'expérience se termine à ce moment-là (par exemple perdre ou gagner un jeu...).

L'environnement renvoie à l'agent, en fonction de son état s et de son action a, un autre état s' et une récompense r, qui est un nombre réel. Mathématiquement, env est une fonction

$$\begin{array}{rccc} env: & S \times A & \to & S \times \mathbb{R} \\ & (s,a) & \mapsto & (s',r) \end{array}$$

Dans le cas d'un environnement stochastique, celui-ci ne renvoie pas directement un nouvel état, mais un vecteur de probabilités d'aller dans chaque état.

Le but est de trouver une suite de paires état-action (s,a) qui maximise la récompense G, où G vérifie :

$$G = \sum_t \gamma^t r_t$$

Si l'environnement est stochastique, on veut alors maximiser le nombre $E(G)$.

γ est le **facteur de réduction**, c'est un réel de $[0,1]$. Il sert à pondérer les récompenses selon le moment où elles sont reçues :

- si $\gamma = 0$, seule la récompense immédiate compte. Ce n'est pas un cas intéressant, puisque l'agent ne prendra jamais le risque d'aller voir dans un autre état, moins intéressant à court terme, mais peut-être plus à long terme.
- si $\gamma = 1$, toutes les récompenses valent la même chose, peu importe le temps où elles sont reçues. Ce n'est pas très intéressant, puisque si le jeu peut être infini, la récompense sera infinie.
- pour γ dans $]0,1[$, la somme converge. Plus γ est proche de 1, plus le gain à long terme est intéressant. En pratique, on prend en général γ entre 0.9 et 1 exclu.

On note :

$$Q^*(s,a) := \max_{(s_t,a_t)_{t\geq 0}} \left(\sum_{t\geq 0} \gamma^t r_t \ | s_0 = s, a_0 = a \right)$$

La fonction Q^* est appelée la Q-fonction optimale. Il s'agit de la fonction qui renvoie la récompense maximale pouvant être obtenue à partir d'un état et d'une action de départ. Evidemment, cette fonction n'est en général pas connue. On va chercher à l'approcher, car une fois cette fonction connue, on connaitra aussi la trajectoire à suivre pour obtenir ce maximum :

$$a_1 = \text{argmax}_{a\in A_{s_1}} Q^*(s_1,a),\ a_2 = \text{argmax}_{a\in A_{s_2}} Q^*(s_2,a) \text{ etc.}$$

On définit également à partir de Q et Q^* les fonctions de valeur V et V^* par :

$$V(s) = \max_{a \in A} Q(s,a) \text{ et } V^*(s) = \max_{a \in A} Q^*(s,a)$$

Cette fonction représente la valeur maximale pouvant être atteinte en se trouvant à l'état s. Dans l'algorithme du Q-learning, on va chercher à approximer Q^* directement. Dans celui du $V-$learning, on va plutôt approximer V.

Théorème de Bellmann

Soient (s,a) un couple état-action, et posons $(s',r) = env(s,a)$. La fonction Q^* vérifie:

$$Q^*(s,a) = r + \gamma \max_{a' \in A_{s'}} Q^*(s',a')$$

Ce théorème va nous permettre d'approcher la fonction Q^* au moyen d'un algorithme itératif. En effet, si une fonction Q vérifie "à peu près" la relation précédente, alors Q sera proche de Q^*.

Algorithme du Q-learning

En pratique :

1) on initialise Q à 0, ϵ à 1

2) on choisit un état s aléatoirement

3) on choisit une action a selon le principe suivant : on choisit aléatoirement une action avec une probabilité de ϵ : c'est la phase d'**exploration**. Et avec une probabilité de $1-\epsilon$, on choisit l'action qui maximise $Q(s,a)$ (avec le Q courant) : c'est la phase d'**exploitation**.

4) on calcule $s', r = env(s,a)$ et on met à jour :

$$Q(s,a) = r + \gamma \max_{a'} Q(s',a')$$

5) on diminue ϵ.

6) si l'état s n'est pas terminal, on recommence à l'étape 3 avec $s = s'$. S'il l'est, on recommence à l'étape 2.

7) On arrête l'algorithme selon un critère d'arrêt, par exemple au bout d'un nombre fixé d'itérations

III.3) Apprentissage par renforcement profond

Le problème de la méthode ci-dessus est que le nombre d'états est en général immense, souvent même infini. Il est alors impossible de construire la fonction Q de façon à ce que tous les couples états-actions soient pris en compte. On utilise alors à la place un réseaux de neurones pour pouvoir prédire la valeur de la fonction Q, en général un réseau de neurones à plusieurs couches cachées, d'où le nom "profond".

En pratique, on choisit une taille de batch, on initialise les poids ω de notre réseau de neurones, on part d'un couple état-action initial (s, a), et on construit un batch d'états-action en suivant la même stratégie que précédemment, selon ϵ.

Une fois ce batch construit, on l'utilise pour modifier les poids ω du réseau de façon à ce que la distance entre les $Q(s, a)$ et les $r + \gamma \max_{a'} Q(s', a')$ soit minimale (où s' est l'état renvoyé après l'action a).

Dans le cadre du projet tutoré, nous allons nous concentrer sur l'apprentissage par renforcement standard.

IV) Implémentation d'un exemple

Dans cette partie, l'objectif est d'illsustrer le fonctionnement de l'algorithme du Q-learning vu plus tôt. Nous allons donc voir les performances de l'entraînement auprès d'un agent. L'environnement, et le programme sont inspirés de [5].

L'environnement

Comme environnement, nous allons prendre une grille dans laquelle nous allons placer l'agent dans la case en-haut à gauche. Dans la case du bas à droite, on y place un "trésor" et dans celle juste au-dessus, une "bombe".

Le but du jeu est pour l'agent de se déplacer selon quatre directions (haut, bas, gauche, ou droite) pour aller chercher le trésor, en évitant bien sûr la bombe, et ceci le plus vite possible. Certaines cases seront des obstacles. S'il se trouve sur un bord de la grille, et que la direction choisie est vers l'extérieur de la grille, l'agent restera sur place.

Les états terminaux sont donc les deux cases "bombe" et "trésor".

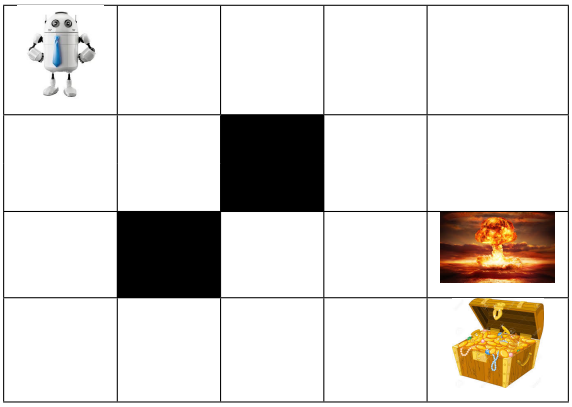


Figure 4: L'environnement GridWorld pour une grille 4×5 avec deux obstacles

La récompense

Pour l'entraînement, l'agent recevra les récompenses suivantes :

- +10 pour un déplacement sur le trésor
- -10 pour un déplacement sur la bombe
- -1 pour chaque déplacement

Voici une figure représentant les récompenses pouvant être gagnées :

-1	-1	-1	-1	-1
-1	-1	0	-1	-1
-1	0	-1	-1	-10
-1	-1	-1	-1	10

Figure 5: Les récompenses obtenues dans Gridworld

Les résultats

Le programme a donc tourné deux fois, une fois avec un agent qui ne choisit son chemin qu'aléatoirement, et une autre fois avec un agent qui apprend.

Voici les récompenses obtenues par ces deux agents sur 500 essais :

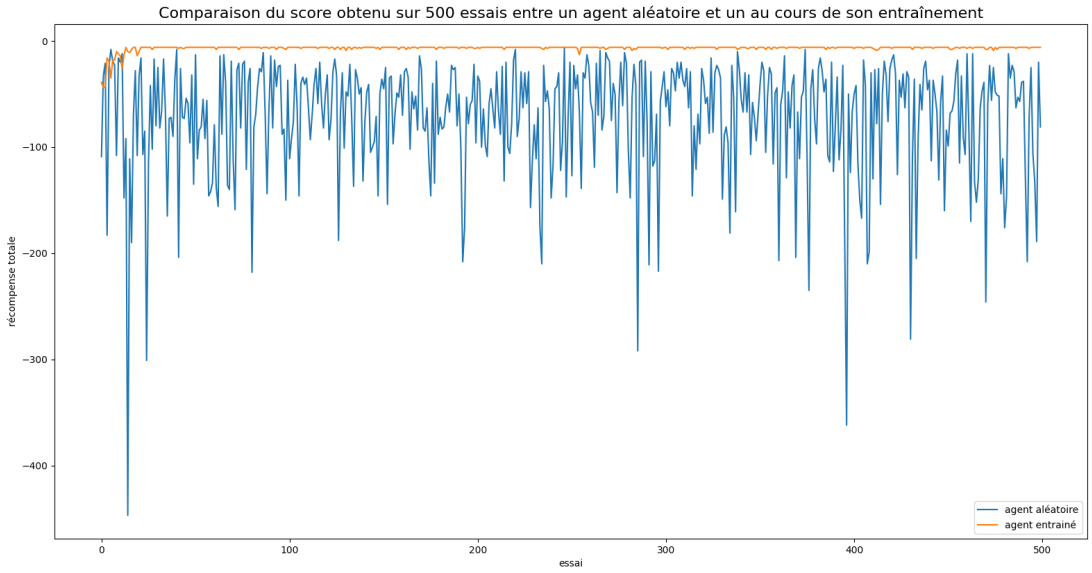


Figure 6: Comparaison des récompenses

On remarque que l'agent qui apprend est très vite beaucoup plus efficace que l'agent aléatoire qui lui peut prendre des valeurs jusqu'à -500. Le programme fonctionne donc bien. En regardant les résultats de l'agent entraîné :

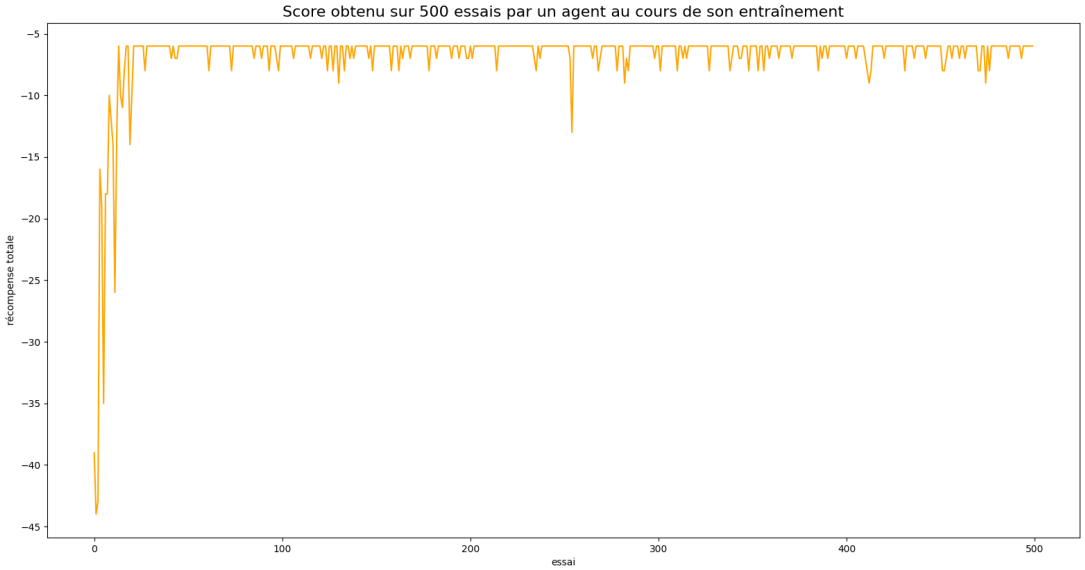


Figure 7: Récompenses de l'agent au cours de son entraînement

On voit bien que les résultats varient beaucoup au début, puis de moins en moins au fur et à mesure de l'apprentissage de l'agent.

La Q-table

La Q-table est composée, pour chaque case de la grille, d'un dictionnaire. Ses quatre clés sont les quatre actions (haut, gauche, droite et bas) et ses valeurs sont celles de $Q(etat, action)$. Au début, toutes ses valeurs sont égales à 1. Voici sur un exemple 3×3, comment, à l'aide de la formule de Bellmann la Q-table est actualisée. Le A rouge représente la position de l'agent, le B et le T la bombe et le trésor.

0 0 A 0 0	0 0 0 0	0 0 0 0
0 0 0 0	0 0 0 0	0 0 B 0 0
0 0 0 0	0 0 0 0	0 0 T 0 0

Figure 8: La Q-table au début

Au début, l'agent est dans une phase d'exploration. Il choisit au hasard une direction. Imaginons qu'il choisisse d'aller à droite. Il recevra une récompense de -1. Donc :

$$Q((0,0), "droite") = -1 + \max_{a'} Q((0,1), a') = -1 + 0 = -1$$

Supposons qu'ensuite, il aille à nouveau droite, le même calcul nous donne :

$$Q((0,1), "droite") = -1$$

Si à l'étape suivante, il choisit d'aller vers le bas, alors la récompense sera de -10, et donc :

$$Q((0,2), "bas") = -10 + \max_{a'} Q((0,1), a') = -10 + 0 = -10$$

A la fin du premier épisode, on aura donc :

0 0 − 1 0	0 0 − 1 0	0 0 0 −10
0 0 0 0	0 0 0 0	0 0 BA 0 0
0 0 0 0	0 0 0 0	0 0 T 0 0

Figure 9: La Q-table à la fin du premier épisode

Imaginons un deuxième épisode où le chemin choisi sera "droite bas bas droite". Alors $Q((0,0),"droite") = -1$, $Q((0,1),"bas") = -1$, $Q((1,1),"bas") = -1$ et $Q((2,1),"droite") = 10$.

A la fin du deuxième épisode, on aura donc :

0 0 − 1 0	0 0 − 1 −1	0 0 0 −10
0 0 0 0	0 0 0 −1	0 0 B 0 0
0 0 0 0	0 0 10 0	0 0 TA 0 0

Figure 10: La Q-table à la fin du deuxième épisode

Si à un troisième épisode, le chemin choisi est "bas droite bas droite". Alors $Q((0,0),"bas") = -1$, $Q((1,0),"droite") = -1$, $Q((1,1),"bas") = -1 + 10 = 9$ et $Q((2,1),"droite") = 10$.

A la fin du troisième épisode, on aura donc :

0 0 − 1 −1	0 0 − 1 −1	0 0 0 −10
0 0 − 1 0	0 0 0 9	0 0 *B* 0 0
0 0 0 0	0 0 10 0	0 0 *TA* 0 0

Figure 11: La Q-table à la fin du troisième épisode

Avec deux autres épisodes on obtiendra donc la Q-table suivante :

0 0 − 1 7	0 0 − 1 −1	0 0 0 −10
0 0 8 0	0 0 0 9	0 0 *B* 0 0
0 0 0 0	0 0 10 0	0 0 *TA* 0 0

Figure 12: La Q-table à la fin du cinquième épisode

Voici la Q-table de l'agent entraîné à la fin de son apprentissage, avec des valeurs arrondies à 10^{-2}. En rouge est notée la valeur maximale :

−2.02 −1.07 4.00 −3.30	0.60 0.55 5.00 −2.59	2.11 −0.57 6.00 0.60	1.41 1.71 − 1.21 7.00	−0.8 −0.80 − 0.80 −0.10
−2.56 −2.60 − 2.54 −2.48	1.29 2.51 − 2.57 −2.58	0.00 0.00 0.00 0.00	4.07 6.46 − 0.40 8.00	−0.46 3.46 − 0.50 −0.47
−1.78 −1.80 − 1.80 −1.20	0.00 0.00 0.00 0.00	−0.10 0.04 5.14 −0.10	1.56 1.35 − 4.69 9.00	0.00 0.00 0.00 0.00
−1.20 −1.12 1.03 −1.10	−0.59 −0.56 4.31 −0.60	−0.10 −0.34 8.13 −0.10	6.28 2.25 10.00 4.70	0.00 0.00 0.00 0.00

Figure 13: La Q-table de l'agent à la fin

On voit bien que la valeur maximale dirige bien vers le bon chemin. Si le chemin ne varie pas (si $\epsilon = 0$ par exemple), le chemin choisi sera donc le suivant :

→	→	→	↓	
		■	↓	
	■		↓	bombe
			→	trésor

Figure 14: Chemin choisi sans tenir compte des mouvements aléatoires

V) Perspectives

Dans ce projet, les bases ont été posées pour comprendre le problème des réseaux d'échangeurs de chaleur, ainsi que le fonctionnement de l'algorithme du Q-learning. Il resterait ensuite à s'approprier le principe de l'algorithme du deep Q-learning, puis, ce qui prendrait plus de temps, de l'appliquer au problème des réseaux d'échangeur afin de voir si l'algorithme d'apprentissage à renforcement profond peut servir à résoudre ce problème efficacement ou non. Il faudrait

ensuite comparer les résultat obtenus avec ceux déjà existants avec les nombreux algorithmes et méthodes décrits plus haut.

VI) Remerciements

Je tiens à remercier l'entreprise Altran de m'avoir permis d'effectuer ce projet tutoré, et plus particulièrement mon encadrant, Ufuk Halisdemir, pour sa sympathie, sa disponibilité, pour le temps qu'il m'a accordé, ainsi que la qualité et la clarté de ses explications.

Bibliographie

[1] ADEME. La chaleur fatale. Septembre 2017.

[2] Irwan Bello, Hieu Pham, Quoc V.Le andMohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. Conference Paper, Janvier 2017.

[3] Wallonie Energie. Différents types d'échangeurs de chaleur. https://energie.wallonie.be/fr/differents-types-d-echangeurs-de-chaleurs.html?IDC=8049&IDD=97759, Mis à jour le 08 avril 2015.

[4] Kevin C. Furman and Nikolaos V. Sahinidis. Computational complexity of heat exchanger network synthesis. Computers and Chemical Engineering, pages 1371 – 1390, Février 2001.

[5] Michael Tinsley. Gridworld with q-learning reinforcement learning. https://github.com/michaeltinsley/Gridworld-with-Q-Learning-Reinforcement-Learning-, 23 février 2018.