Deep Reinforcement Learning appliqué à la planification de vidage de satellites

Auteur
Youva Addad

Encadrant
Jean-Noël Vittaut

Table des matières

1 Introduction **2**

2 Définition **2**

3 Difficulté de l'optimisation **3**
3.1 les Contraintes . . . 3
3.2 Environnement/Agent . . . 3
3.3 Fonction objective . . . 3

4 Approche de Conception **4**
4.1 DQN . . . 4
4.2 OpenAI Gym . . . 4

5 Modélisation **5**
5.1 Simulateur Global . . . 5
5.2 Image . . . 6
5.3 Task . . . 6
5.4 Satellite . . . 6
5.5 Station . . . 6

6 Implémentations pratique **7**
6.1 Modèles . . . 7
6.2 Espace d'observations . . . 7
6.3 Espace d'actions . . . 8
6.4 Récompense . . . 8

7 Experiences **8**

8 Évaluation **8**
8.1 Glouton . . . 9
8.2 Binary Model . . . 10
8.3 Multiple selection . . . 11

9 Conclusion **11**

1 Introduction

Le contexte général de ce projet concerne l'application de modèles de Deep Reinforcement Learning à la problématique de la transmission au sol d'acquisitions prises par constellation de satellites.
Les constellations de satellites d'observation acquièrent des images dans le monde entier, couvrant jusqu'à plusieurs millions de kilomètres carrés chaque jour, La décision de SpaceX d'augmenter de 30.000 le nombre de satellites pour son projet de méga-constellation Starlink, pour ne cité qu'eux.
Étant donné que le contact direct avec un satellite est limité à de brèves périodes sur les stations au sol, des calendriers des tâches au préalablement définie.
La planification de télé-déchargement de ces images vers les stations est un problème difficile[1,3,4].
Nous souhaitons démontrer que l'apprentissage par renforcement profond pourrait être bien adapté à la planification du vidage.
Le DRL s'est avéré d'une grande valeur puisque ces algorithmes ont maîtrisé plusieurs jeux tels que Pong sur Atari 2600 (Mnih et al.2013), Go with AlphaGo (Silver et al.2017) et plus récemment Starcraft (Arulkumaran, Cully, et Togelius 2019). Le processus de livraison d'images suit un composé de[2] :

1. Réception par des centres opérateurs de requêtes client pour l'observation de zones spécifiques.
2. Découpage des zones en mailles pouvant être acquises par les satellites.
3. Planification des acquisitions des mailles par les satellites.
4. **Planification du vidage des images acquises vers des stations terrestres réceptrices.**
5. Validation des images et livraison aux clients.

2 Définition

— **Plan d'acquisition** : Un planning de l'ensemble des taches qui doivent être effectué
— **Plan de vidage** : Un planning de vidage des images qui doivent être télé-déchargé.
— **Q-learning** : Un algorithme de reinforcement learning, Q pour quality.
— **Maille** : la portée d'un satellite (le périmètre maximum qu'un satellite utilise pour faire une acquisition).

3 Difficulté de l'optimisation

Résoudre ce problème de planification de vidage d'acquisition sans le DRL reviendra à tester toutes les combinaisons possibles un problème qu'on qualifie de NP-Hard[3,4]

$$\begin{array}{rl} \min & Perte \\ \text{s.t.} & \min \quad Connexion \\ & \min \quad expiration \\ & \max \quad stockage \end{array} \tag{1}$$

Avec perte c'est l'ensemble des acquisitions perdu, connexion est le coût pour se connecter a une station, stockage c'est la capacité d'un satellite.

Prendre en compte tous les satellites et toutes ses contraintes afin de trouver le meilleur plan de vidage est difficile, rajouté un très grand nombre de satellites, le problème deviendra impossible sans outil puissant.

3.1 les Contraintes

Comme nous l'avons souligné précédemment nous cherchons à minimiser la perte d'information, toute en maximisant le bon déroulement du plan d'acquisition, et donc le bon déroulement du plan de vidage, toutes en respectant les contraintes suivantes :

- Faire le minimum de connexions possibles.
- Respecter la deadline de validité des acquisitions.
- Respecter la priorité des acquisitions.
- maximisé l'espace de stockage d'un satellite donc privilégier le vidage des acquisitions volumineuse.
- Éviter le plus possible une pertes d'acquisitions.
- le choix de partage d'un station par un satellite.

3.2 Environnement/Agent

Le choix de l'environnement, des actions entreprises par un agent et de la fonction de récompense sont des éléments essentiels pour une bonne performance de vidage. En effet nous travaillons sur un model-free c'est-à-dire une famille d'algorithme qui n'utilise pas la distribution de probabilité de transition (et la fonction de récompense) associée au processus de décision de Markov (MDP), mais utilise une fonction 'Q' pour quality, nous détaillerons cette dernière dans la section suivante.

3.3 Fonction objective

Notre objectif est de maximiser cette fonction[5] :

$$Q^{\text{new}}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Big(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \Big)}^{\text{new value (temporal difference target)}} \overbrace{- \underbrace{Q(s_t, a_t)}_{\text{old value}}}^{\text{temporal difference}}$$

Avec r_t la récompense reçue lors de la transition vers l'état s_t, α est le taux d'apprentissage $(0 < \alpha \leq 1)$.
Donc de manière général $Q^{\text{new}}(s_t, a_t)$ est la somme de :

— $(1-\alpha).Q(s_t, a_t)$ la valeur actuelle pondérée par le taux d'apprentissage. Les valeurs du taux d'apprentissage proches de 1 accélèrent les changements de Q.
— $\alpha.r_t$ la récompense a obtenir si l'action a_t a été entreprise dans l'état s_t pondéré par le taux d'apprentissage.
— $\alpha.\gamma \max_a Q(s_{t+1}, a)$ la récompense maximale qui peut être obtenue de l'état pondéré par le taux d'apprentissage.

4 Approche de Conception

4.1 DQN

Nous avons donc décidé d'utiliser un DQN (Deep Q-Network) pour approximer Q(s, a) la fonction objective définie en **(3.3)**.
Cet algorithme combine l'algorithme Q-Learning (l'optimisation de la fonction objective de (3.3)) avec des réseaux de neurones profonds (DNN). Ainsi, les DNN sont utilisés pour approximer la fonction Q, remplaçant le besoin d'une table pour stocker les valeurs Q[6].

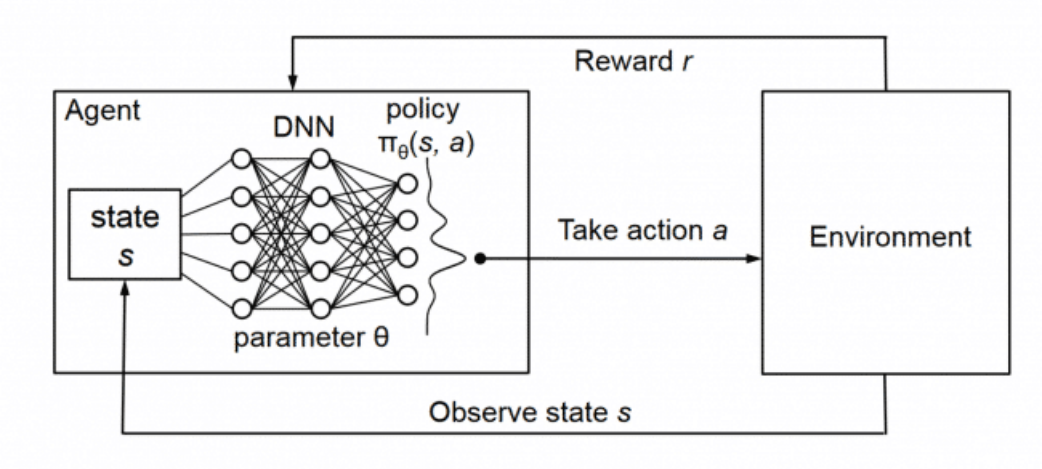

FIGURE 1 – L'apprentissage du DQN

le DQN est capable d'obtenir de bonnes performances dans un environnement en ligne pour une variété de jeux ATARI, directement en apprenant des pixels[6].

4.2 OpenAI Gym

Nous avons en outre adapté notre environnement pour qu'il soit conforme à OpenAI Gym.

OpenAI Gym est un toolkit python pour le développement d'agents de renforcement, il fournit une interface facile à utiliser pour interagir avec une variété d'environnements. Il contient plus de 100 implémentations open source d'environnements d'apprentissage par renforcement. OpenAI Gym accélère le développement d'agents d'apprentissage de renforcement en gérant tout du côté de la simulation d'environnement, nous permettant de nous concentrer sur les agents et leurs algorithmes d'apprentissage.

Le noyau de l'interface de gym est env, pour pouvoir l'utiliser il faudra donc redéfinir les méthodes suivantes :

- reset(self) : Réinitialisez l'état de l'environnement. Renvoie une observation.
- step(self, action) : Marche de l'environnement d'un pas de temps. Renvoie une observation, récompense, fait(done), info.
- render(self, mode='human') : rend une image de l'environnement. Le mode par défaut fera quelque chose de convivial, comme ouvrir une fenêtre.

5 Modélisation

5.1 Simulateur Global

Notre simulateur ce repose sur une représentation 2D de la terre qui est donc notre environnement. Sur cette **Environnement** on y trouve des **satellites** nos agents qui font des orbites suivant un plan d'acquisition prédéfinie (le plan d'acquisition est un ensemble de **Task**) effectuant des prises d'**images**.

La figure ci-dessous nous montre les différents éléments de ce simulateur les points Bleus foncés sont les Stations qui sont fixes, les autres points représentent les satellites qui se déplacent sur différentes orbites, les losanges sont les acquisitions à faire pour chaque satellite qu'on peut distinguer avec leur couleur sur la figure.

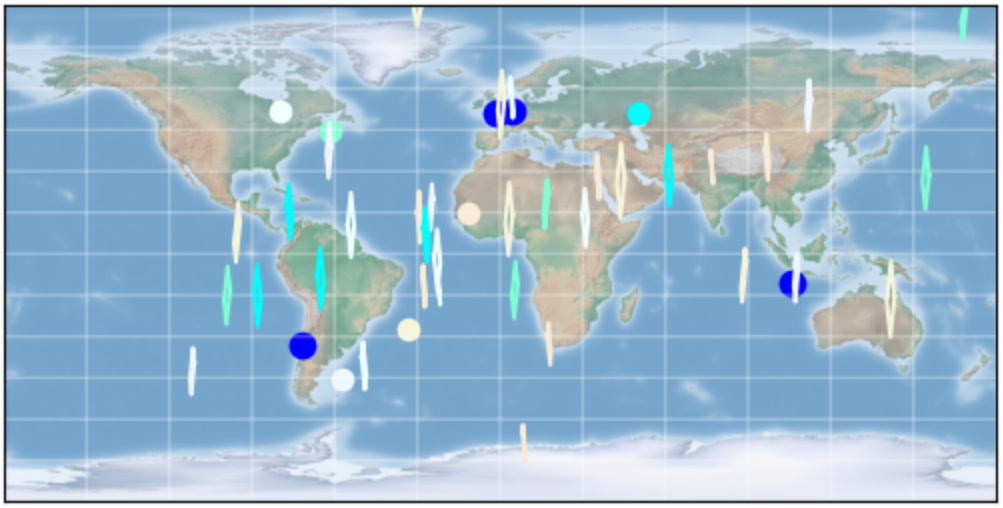

FIGURE 2 – Représentation 2D du simulateur

Pour pouvoir simuler une exécution réelle, la notion de temps n'est pas utilisée telle quelle, elle est modélisée par la mise à jour des positions des satellites dans l'environnement.

5.2 Image

L'objet image représente donc une image capturer par un satellite, elle a comme attribut :

- Un **id** unique pour chaque image
- Un **taille** les acquisitions ont des tailles différentes.
- Une **priorité** pour pouvoir donnée une importance lors du vidage.
- Une **date d'expiration** une limite à ne pas franchir, elle diminue au fur et à mesure de l'avancement du satellite en arrivant a 0 on pénalise la récompense.

5.3 Task

L'objet représentant une tâche a effectué, une tache est donc une demande de prise d'image, générée donc aléatoirement dû à un manque de donnée concrète. l'objet task a pour attribut en plus de ceux d'image :

- une **position de commencement** de l'acquisition.
- une **longueur** de l'acquisition.

5.4 Satellite

Cet objet représente les satellites, elle a donc un plan d'acquisition préalablement définie, elle fait une orbite en suivant une direction , elle se charge de stocker les images acquises pour pouvoir les vider en recevant une autorisation de la part de la station proche (l'autorisation est la liberté de la station). elle comporte donc entre autre :

- Une **capacité maximum de stockage** pour ne pas accepter au-delà de ces capacités, elle est donc mise à jour soit par une nouvelle acquisition soit par vidage d'une image acquise, elle est initialisé aléatoirement.
- Un **ensemble de taches** à effectuer, à l'initiale elle a un nombre fixe de tache (plan d'acquisition au préalablement définie et mis à jour au fur et à mesure).
- Un **ensemble d'images prise** les acquisitions qui doivent être transmises.

5.5 Station

La station permet donc de faire des interactions entre satellites, dés lors qu'un satellite approche d'une station et qu'il décide de décharger des acquisitions il demande un accès c'est à dire regarde si elle est libre, alors la station lui répondra favorablement ou défavorablement suivant l'état courant. Si le satellite a eu un avis favorable (station libre) alors il y aura une connexion, si il a finit de décharger une acquisition et si à l'état suivant il décide de ne pas décharger alors il se déconnecte de la station.
Si la station est visible pour plusieurs satellites, dans ce cas les satellites se partageront la station, mais il ne pourra y avoir qu'un seul satellite transmetteur. On trouvera donc les attributs suivant :

- le **débit de transfert** par changement d'état.
- une **distance limite** pour être dans le réseau de cette station (champ de visibilité de la station par les satellites).
- un **temps de latence** pour établir une connexion entre station et satellite.

6 Implémentations pratique

Pour la réalisation du DQN, nous avons utilisé Keras-Rl(tensorflow).
Nous avons créé deux environnements (c'est à dire deux modèles différent) qui fonctionnent tous les deux de la même manière, où une étape, c'est l'existence d'un satellite qui a une station visible et pourrait décharger au moins une acquisition. Nous avons utilisé une "Queue" lors des mises à jour pour pouvoir ajouter tous les satellites qui peuvent effectuer une action.

6.1 Modèles

Pour pouvoir faire marcher le DQN nous avons simuler deux modèles (deux environnements) :

— **Modèle de décision binaire** : On présente la représentation du satellite courant (l'espace D'observations) contenant une acquisition à décharger choisit au préalable par maximum de priorité. Les actions possible sont donc de décharger ou de ne pas décharger l'acquisition.

— **Modèle multiple choix de sélection** : On présente la représentation du satellite courant (l'espace D'observations) sans le choix de l'acquisition à décharger. Les actions possibles sont de décharger avec un critère de sélection (maximum priorité, maximum taille, date d'expiration la plus proche), ou de ne pas décharger.

6.2 Espace d'observations

L'espace d'observation (l'état courant) dans les deux modèles, est l'état du satellite qui pourra effectuer la prochaine action, l'état est donc un vecteur des différentes caractéristiques du satellites courant, pour les prochaines taches et les acquisitions sauvegarder **la priorité**, **l'expiration**, et **la taille**. Pour pouvoir utiliser le DQN il nous faut une entrée de taille fixe, c'est pour cela la représentation précédente.

En résumer, nous avons cet représentation commune au deux modèles une concaténation de toutes ses caractéristiques, à rajouter au modèle binaire un choix d'acquisition (choisis par maximum de priorité) :

- l'état mémoire et de la connexion
  — Si il est connecté.
  — Le nombre de satellites pouvant se connecter la même station.
  — La capacité mémoire.
  — Taux de saturation de sa mémoire.
- Les acquisitions sauvegardées et celles qui peuvent être déchargées
  — Le nombre total des acquisitions.
  — Le nombre d'acquisitions qui peuvent être déchargées.
  — Représentation d'une agrégation sur les acquisitions (faire une moyenne sur toutes les acquisitions suivant **la priorité**, **l'expiration**, et **la taille**) et celles possible a décharger comme des vecteurs de taille (1,3).
- Les prochaines tâches à effectuer
  — Le nombre des prochaines tâches.
  — Estimation du taux de saturation et de la capacité mémoire (si on fait l'acquisition de ces tâches).
  — Représentation d'une agrégation de ces tâches comme un vecteur de (1,3).

6.3 Espace d'actions

— **Modèle (1)** à pour action :
- **0** : Ne pas décharger, se déconnecter de la station s'il est déjà connecté.
- **1** : Se connecter si ce n'est pas déjà le cas et procéder au déchargement de l'acquisition choisie.

— **Modèle (2)** à pour action :
- **0** : Ne pas décharger, se déconnecter de la station s'il est déjà connecté.
- **1** : se connecter si ce n'est pas déjà le cas et procéder au déchargement de l'acquisition de plus grande pertinence.
- **2** : Se connecter si ce n'est pas déjà le cas et procéder au déchargement de l'acquisition de plus grande taille.
- **3** : Se connecter si ce n'est pas déjà le cas et procéder au déchargement de l'acquisition de plus petite date de validité.

6.4 Récompense

La récompense est commune pour les deux modèles :
— Toutes acquisitions déchargées nous recomposent par sa priorité.
— Chaque connexion a une station coûte le temps de connexion.
— Chaque acquisition sauvegardée expirée nous coûte sa priorité.
— Chaque tâche qu'on n'a pas pu réaliser dû à une saturation mémoire, nous coûte sa priorité.

7 Experiences

Pour expérimenter l'efficacité de nos modèles de vidage nous créons un environnement lié au simulateur :
— Les stations sont choisies aléatoirement parmi 700 stations terrestres existantes, avec un débit de transmission et un temps de connexion aléatoires.
— Les satellites sont initialisé avec des positions aléatoires et des directions permettant d'une manière à atteindre toutes les zones. Le plan d'acquisitions d'un satellite est initialisé avec 350 tâches aléatoires, de nouvelle tache ce rajoute par la suite.
— Les taches sont initialisées aléatoirement avec des priorités $\{1, 2, 3, 4\}$. La date de validité est initialisée proportionnellement à la distance à parcourir entre le satellite et la position générée.

8 Évaluation

Pour l'évaluation, nous avons sauvegardé le déroulement du processus d'apprentissage les gains de chaque satellite, la perte des acquisitions en mémoire, la perte des acquisitions qui ne sont pas encore faites, et le nombre de connexions de ce satellite ce qui nous permet de calculer la moyenne du gain.

8.1 Glouton

Le modelé glouton a les mêmes état que le modèle binaire, en plus si la mémoire est supérieure a 75%, si la mémoire estimé sur les prochaines taches est supérieur a 1%, la validité des acquisitions possible a décharger doit être inférieur a la de toutes les acquisitions en mémoire.

Nous avons utilisé le nombre de visibilités (le nombre de stations rencontrées par un satellite) comme paramètre pour le calculer de l'amélioration de la récompense.

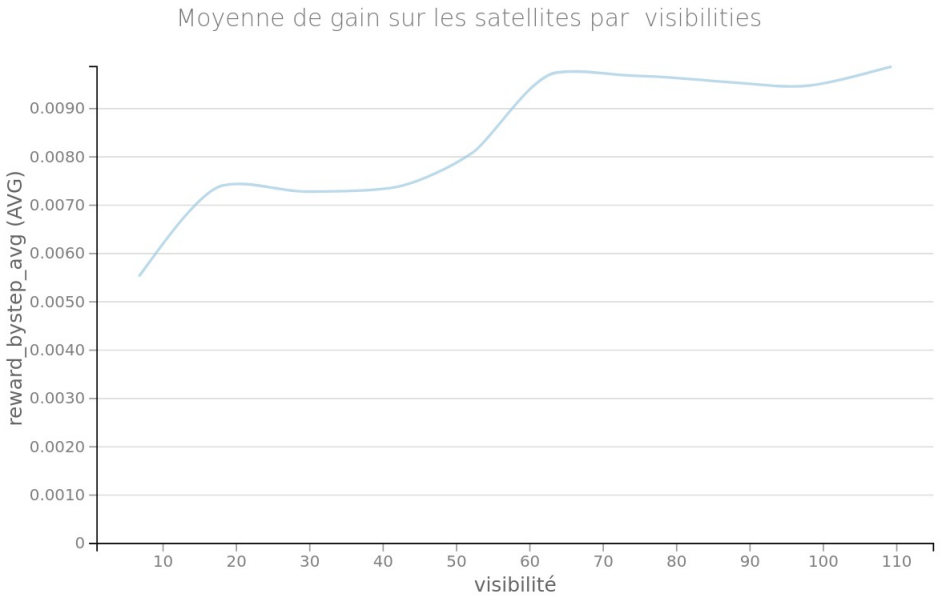

FIGURE 3 – Courbe de la moyenne du gain sur tout les satellites en fonction du nombre de visibilités station

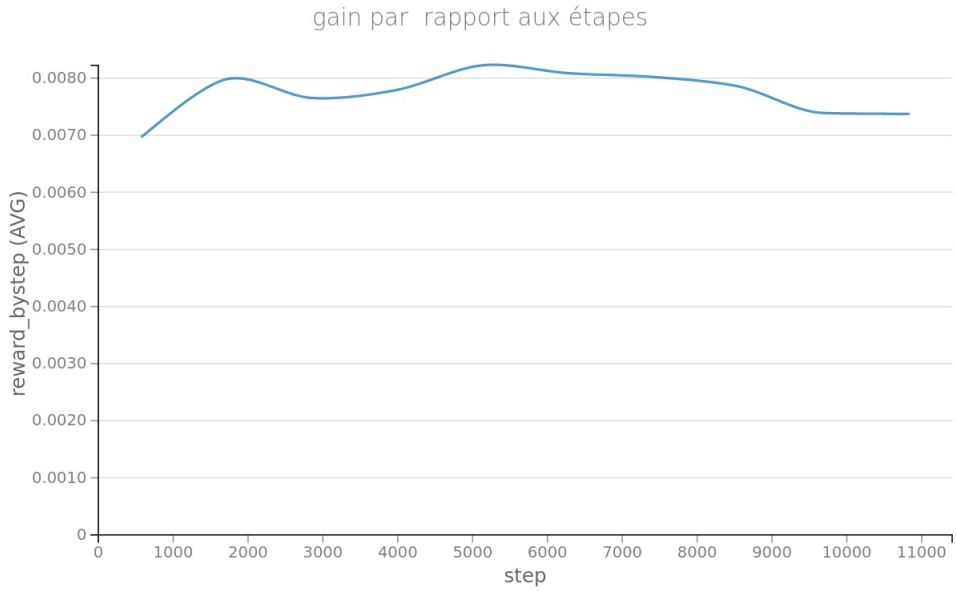

FIGURE 4 – Courbe du gain en fonction du nombre d'étapes d'apprentissage

8.2 Binary Model

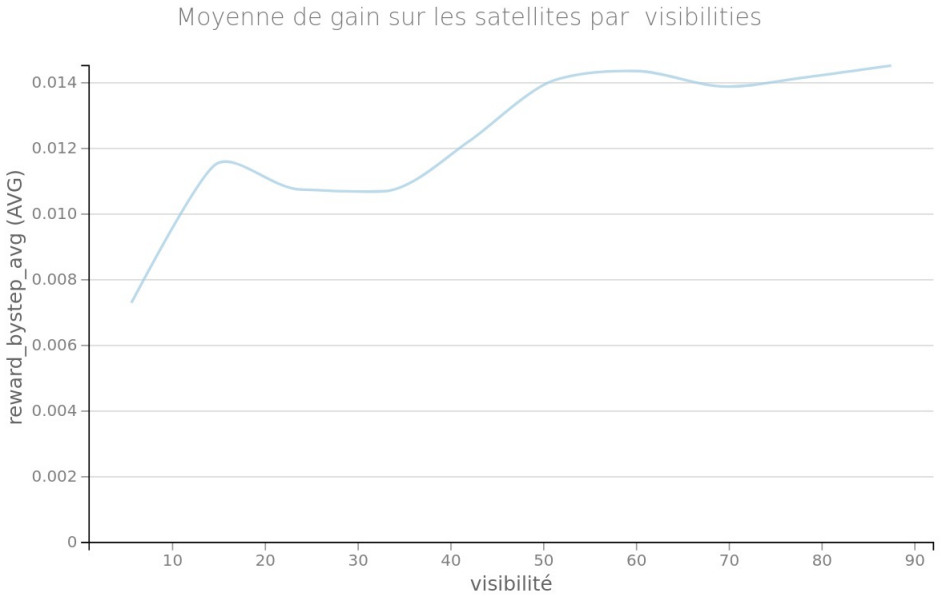

FIGURE 5 – Courbe de la moyenne du gain sur tout les satellites en fonction du nombre de visibilités station

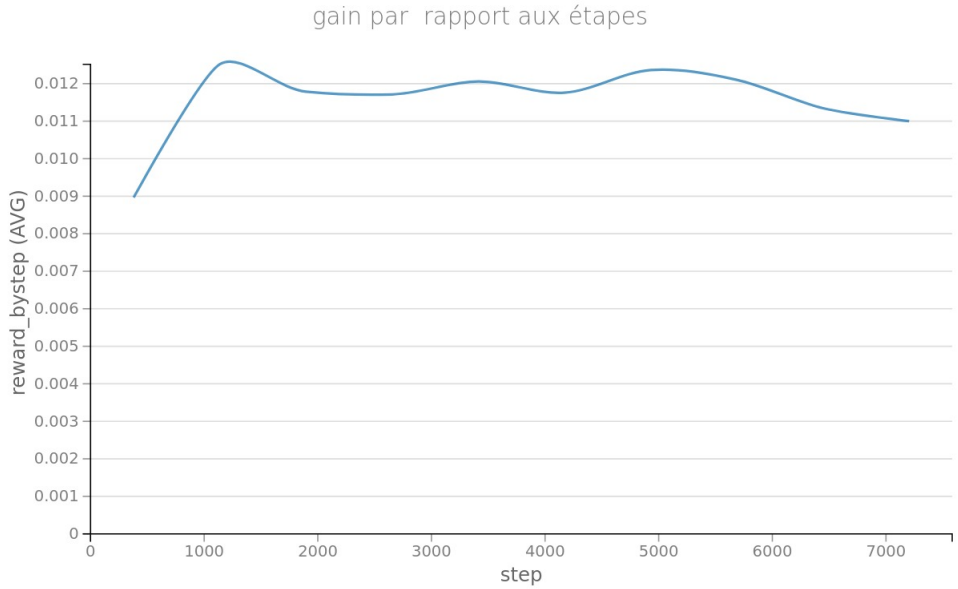

FIGURE 6 – Courbe du gain en fonction du nombre d'étapes d'apprentissage

8.3 Multiple selection

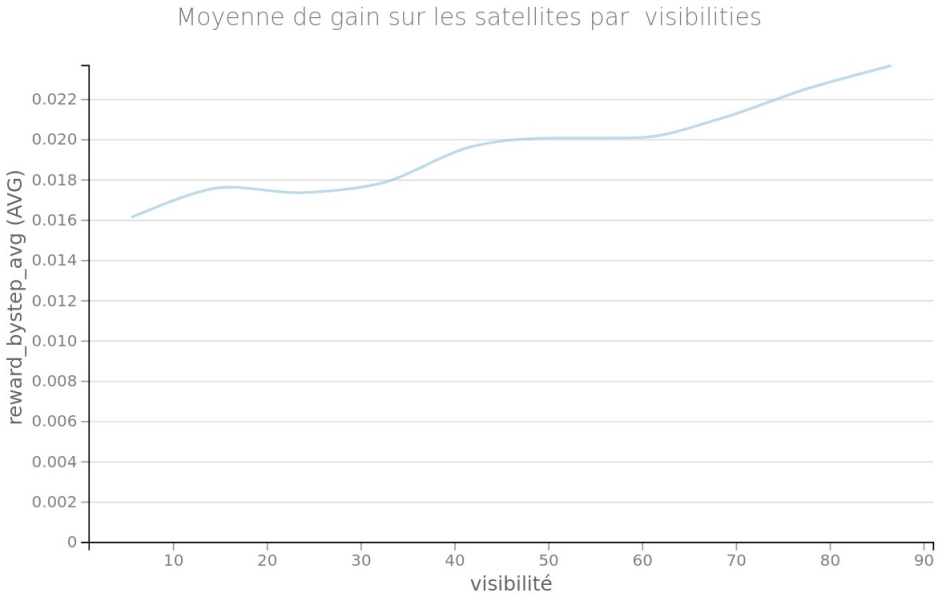

FIGURE 7 – Courbe de la moyenne du gain sur tout les satellites en fonction du nombre de visibilités station

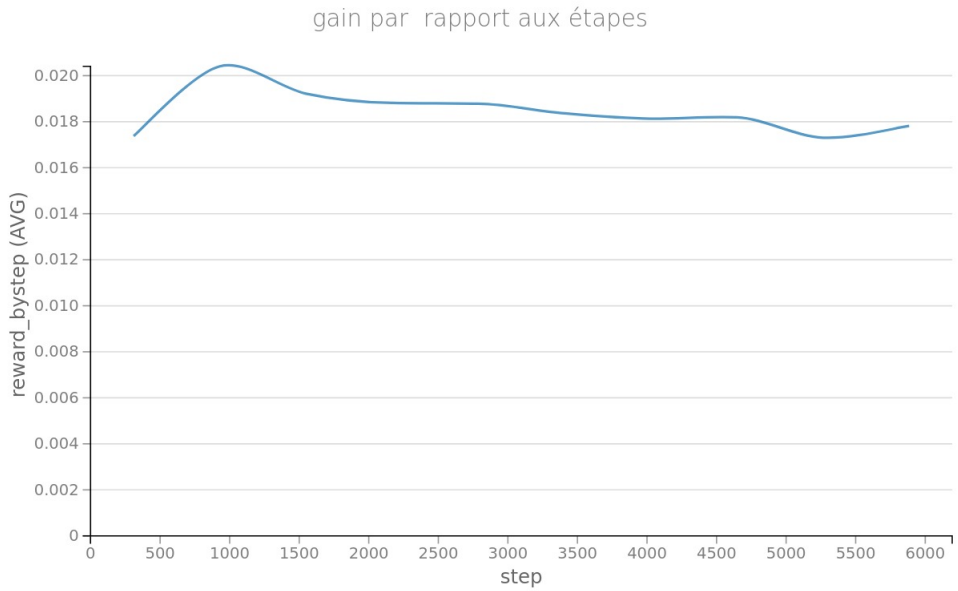

FIGURE 8 – Courbe du gain en fonction du nombre d'étapes d'apprentissage

9 Conclusion

Dans ce projet nous avons présenté une approche deep reinforcment learning pour la résolution du probléme de vidage d'images satellitaires.
Le deep reinforcment learning est un outil très efficace pour pouvoir traité tous type de problème d'ordonnancement, et nous montre que c'est un outil efficace pour la planification de vidage de satellite.

Références

[1] Pavlov, Alexander Anatolievich, et al. « NP-Hard Scheduling Problems in Planning Process Automation in Discrete Systems of Certain Classes ». Advances in Computer Science for Engineering and Education, édité par Zhengbing Hu et al., Springer International Publishing, 2019, p. 429-36. Springer Link, doi :10.1007/978-3-319-91008-6_43.

[2] Jammot, T., et al. « Planification de Vidage d'images Satellitaires Par Systèmes Multi-Agents Auto-Adaptatifs (Présentation Courte) ». Undefined, 2018, https ://www.semanticscholar.org/paper/Planification-de-vidage-d

[3] Garey, Michael R., et David S. Johnson. Computers and Intractability ; A Guide to the Theory of NP-Completeness. W. H. Freeman Co., 1990.

[4] Barbulescu, Laura, et al. « Scheduling Space–Ground Communications for the Air Force Satellite Control Network ». Journal of Scheduling, vol. 7, no 1, janvier 2004, p. 7-34. Springer Link, doi :10.1023/B :JOSH.0000013053.32600.3c.

[5] Francois-Lavet, Vincent, et al. « An Introduction to Deep Reinforcement Learning ». Foundations and Trends® in Machine Learning, vol. 11, no 3-4, 2018, p. 219-354. arXiv.org, doi :10.1561/2200000071.

[6] Mnih, Volodymyr, et al. « Human-Level Control through Deep Reinforcement Learning ». Nature, vol. 518, no 7540, février 2015, p. 529-33. www.nature.com, doi :10.1038/nature14236.