# Création de différentes IA en utilisant une architecture I2A

## Dans le contexte du jeu Sokoban

Thomas Guyomard - Jérémy Tremblay

Université du Littoral Côte d'Opale

*Encadrant : Jérôme Buisine*

2024/06/10

# Sommaire

# Références


Sébastien Racanière, Théophane Weber, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, Demis Hassabis, David Silver, Daan Wierstra.
Imagination-Augmented Agents for Deep Reinforcement Learning.
*DeepMind*, arXiv:1707.06203v2 [cs.LG], 14 Feb 2018.

Max-Philipp B. Schrader.
gym-sokoban.
GitHub repository, GitHub, 2018.
`https://github.com/mpSchrader/gym-sokoban`.

# Présentation du jeu Sokoban

Jeu de réflexion. Le but : le joueur doit ranger des caisses sur des cases cibles.

- Le personnage a la capacité d'effectuer des déplacement dans chacune des 4 directions possibles. Le niveau est validé une fois toutes les caisses rangées.

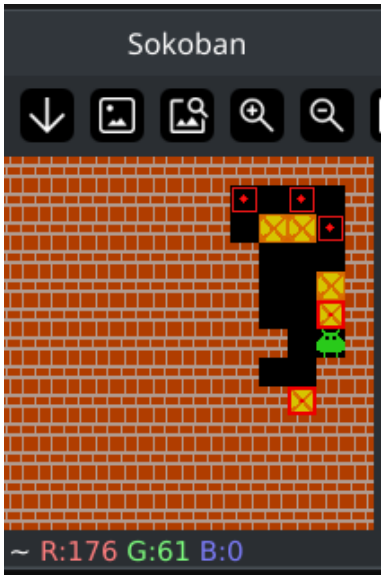



Figure: Capture d'écran du jeu Sokoban.

# IA Q-learning avec une représentation en tableau

## Definitions

L'implémentation combine le Q-learning avec un modèle interne I2A pour résoudre le problème complexe du Sokoban, un jeu de réflexion.

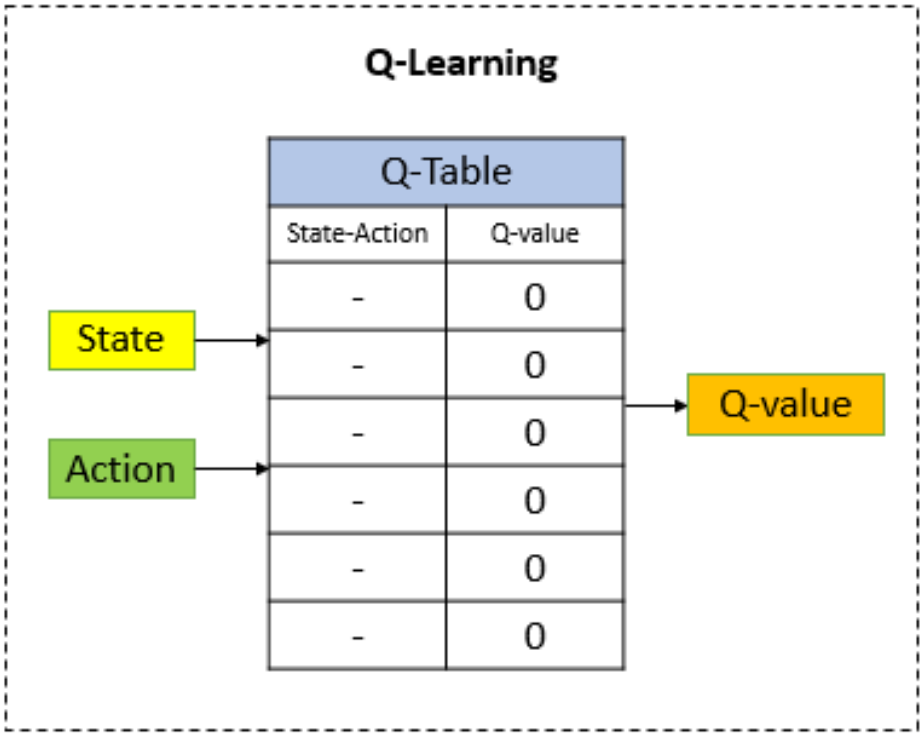



Figure: Schéma du fonctionnement du Q-Learning.

# Implémentation du Q-learning

- **Q-table** Contient les états / actions du jeu, elle est initialisée avec des zéros.
- **Boucle d'apprentissage** Itérations sur des épisodes pour mettre à jour la Q-table.
- **Système de récompenses** Intégration des récompenses conformément à la politique définie.
- **Mise à jour de la Q-table** Prise en compte de la récompense immédiate et de l'estimation de la récompense future.
- **Exploration et Exploitation** Equilibre les connaissances actuelles et permet davantage d'apprentissage.

## Contenu de la Q-table

```
11468484326015042000:  array([-0.05, 2.77913742,
1.69800543, 3.03067097, 2.54018055, 2.77913742,
2.77913742, 3.03067097, 2.54018055])
```

# Contexte et environement d'exécution

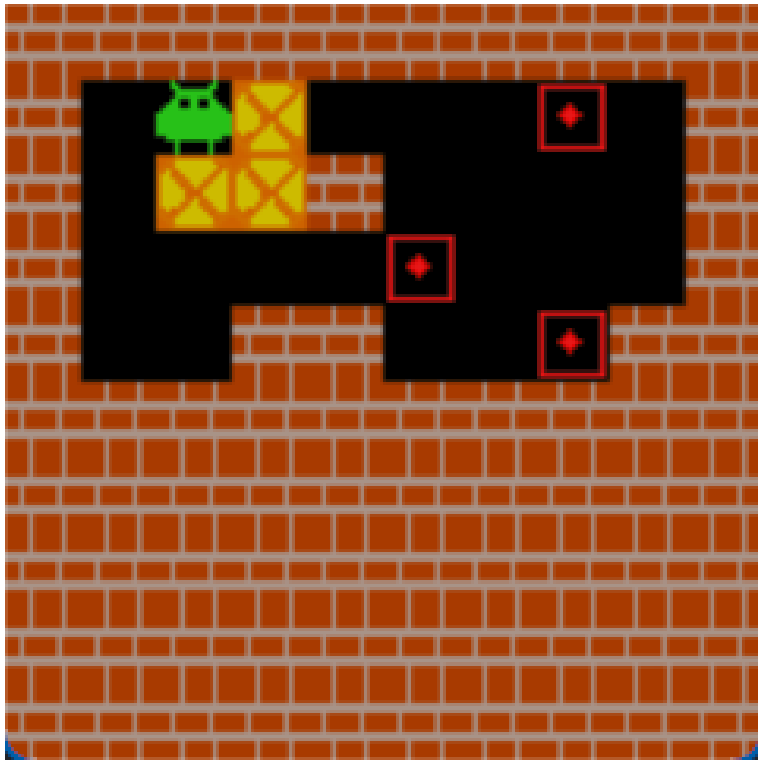


Figure: Évolution des récompenses totales obtenues par épisode.

**Contexte de jeu :**

- Niveau de taille moyenne
- Jeu de 10x10 cases
- 3 caisses à pousser

**Contexte d'exécution :**

- 2000 parties réalisées/algorithme
- Utilisation du Q-learning
- 40 min de temps d'exécution au total/algorithme

# Premiers résultats du Q-learning

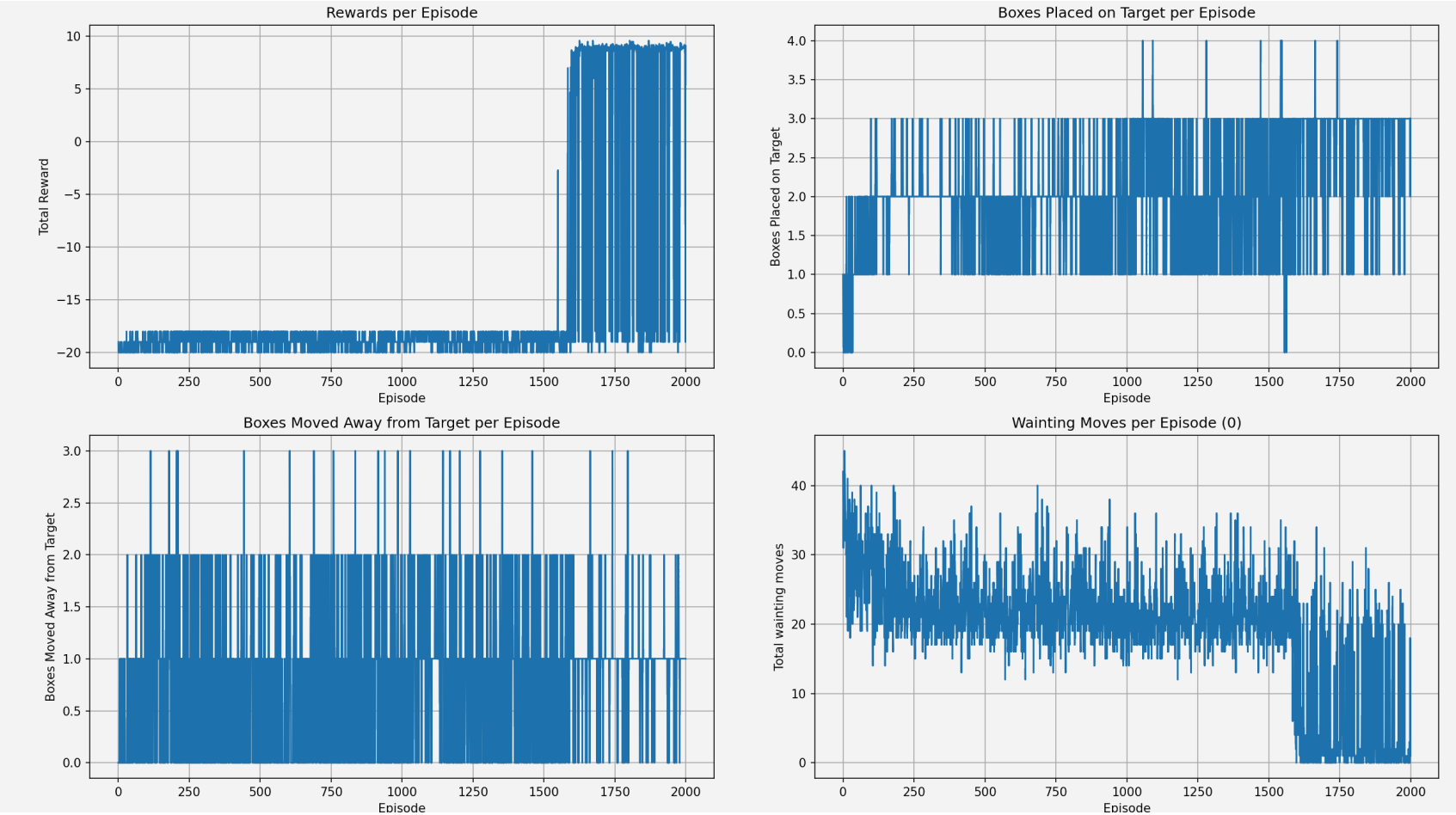



Figure: Résultats du Q-learning sur 2000 épisodes.

# Problème

- **Meilleures performances :** Optimiser l'algorithme de jeu pour détecter plus rapidement les situations de blocage.
- **Pénalités :** Réduire les erreurs de détection et minimiser les faux positifs.

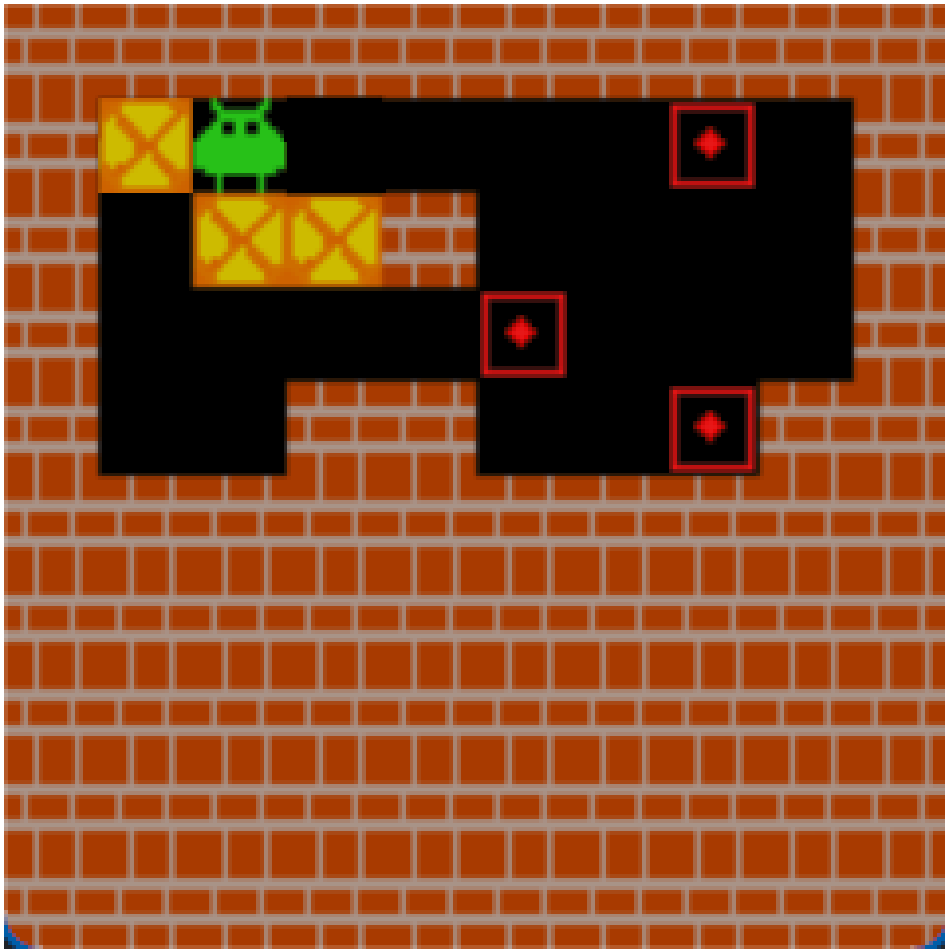
## Problématique

Comment détecter efficacement une partie perdue dans le jeu Sokoban ?

# Cas Simple

## Observation des résultats

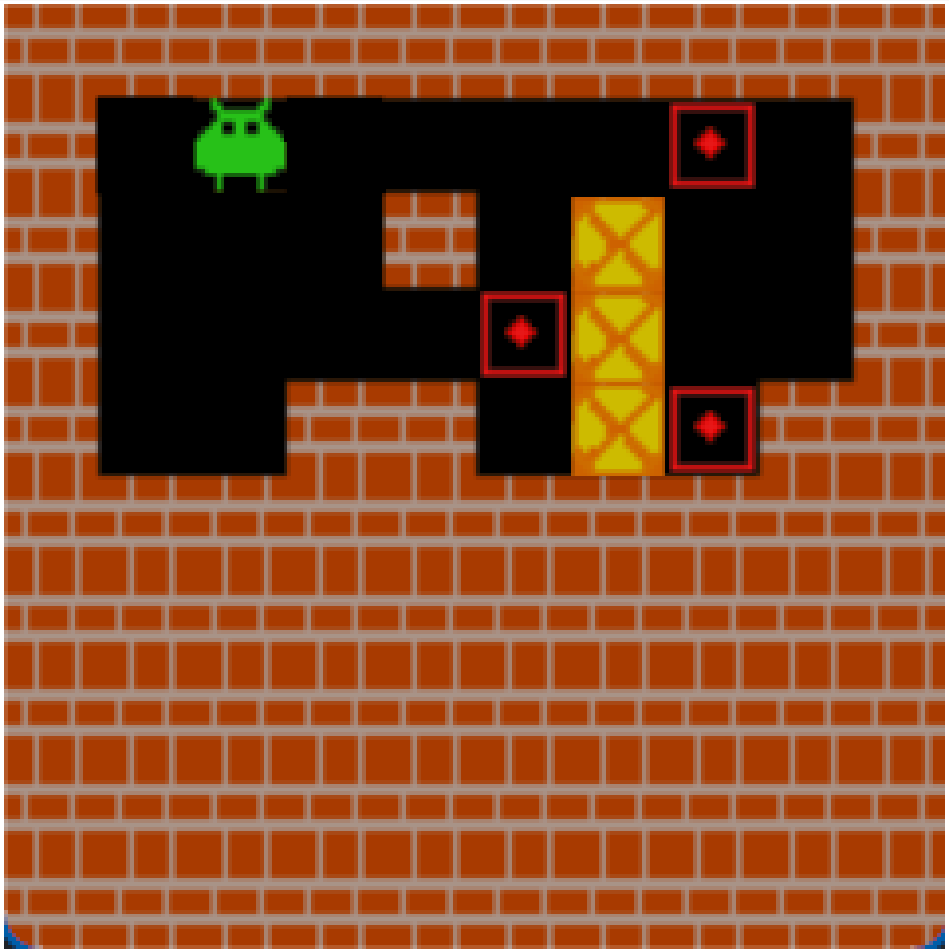
L'agent a bloqué la caisse contre le mur.



# Cas plus complexe

## Observation des résultats

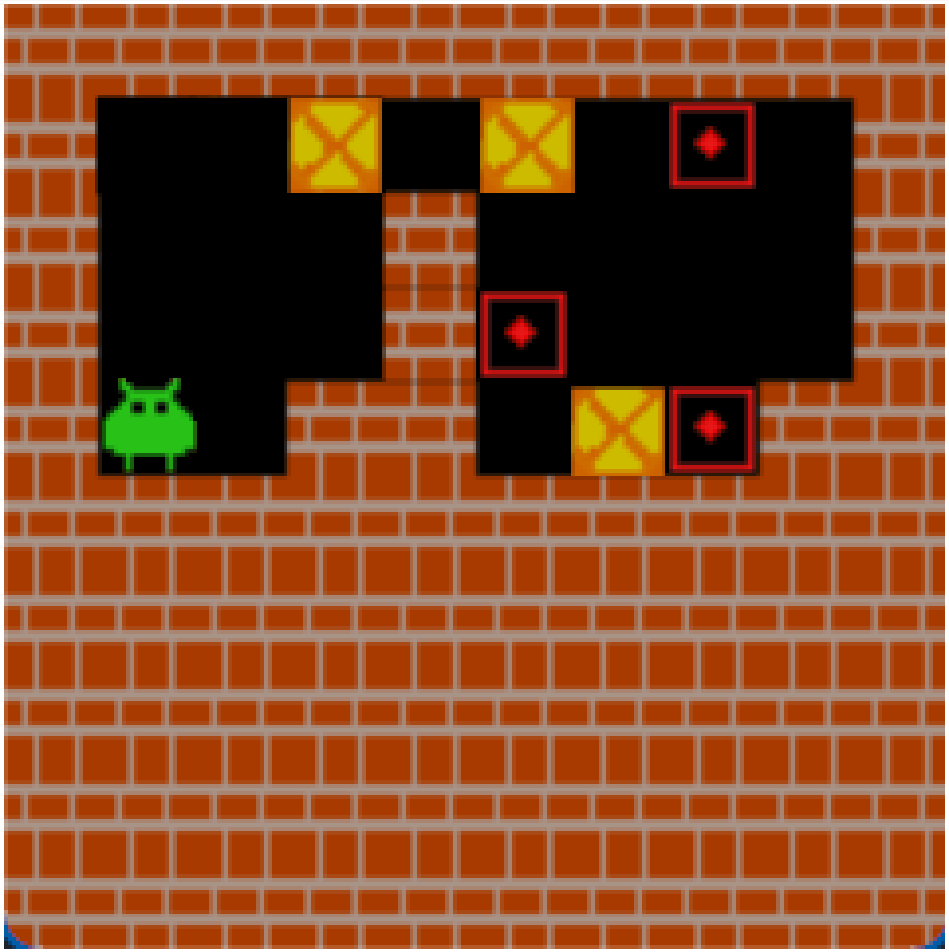
L'agent a bloqué une caisse entre deux autres caisses .



# Cas très complexe

## Observation des résultats

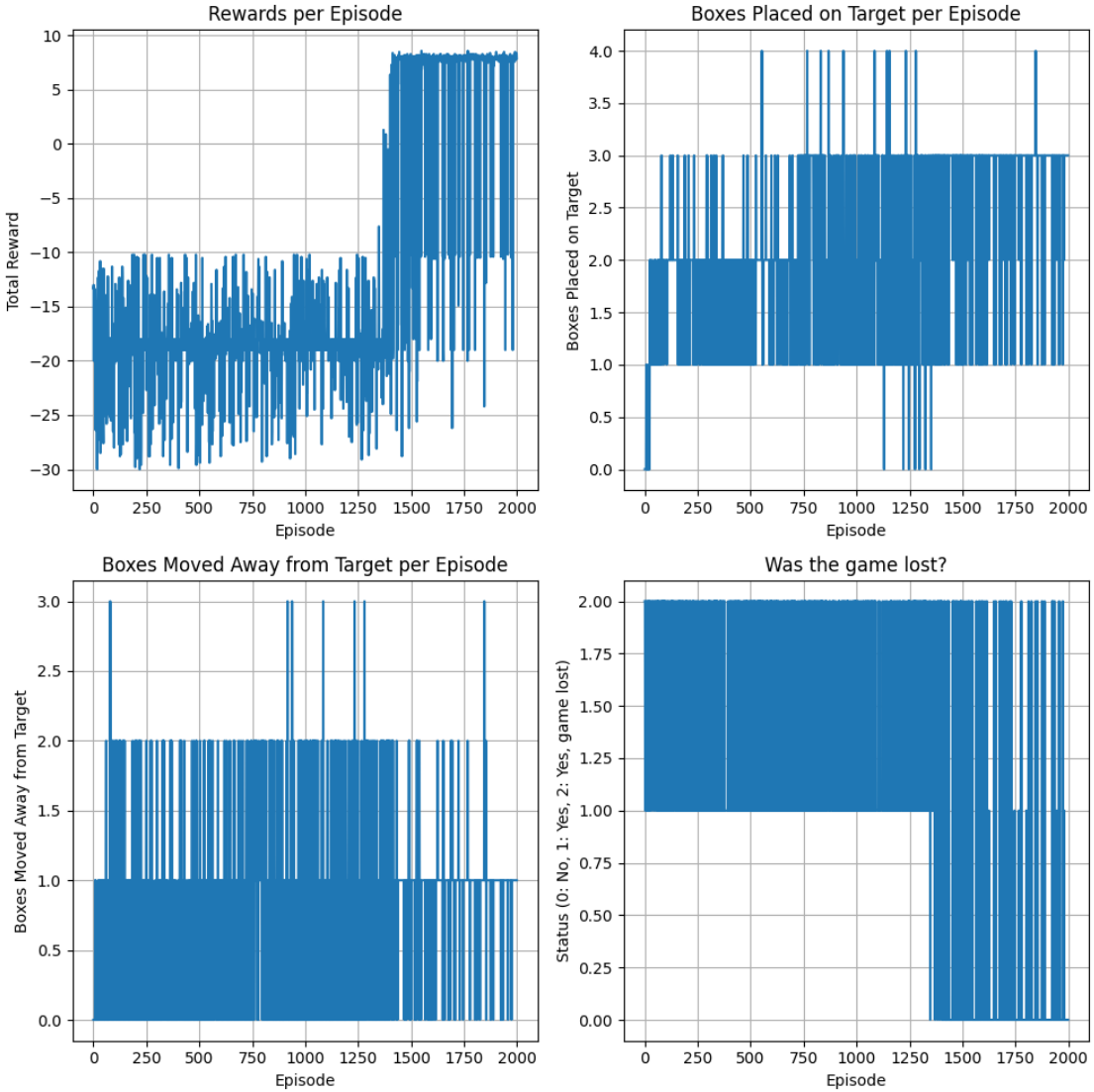
Aucune caisse n'est bloqué pour autant la partie est perdue.



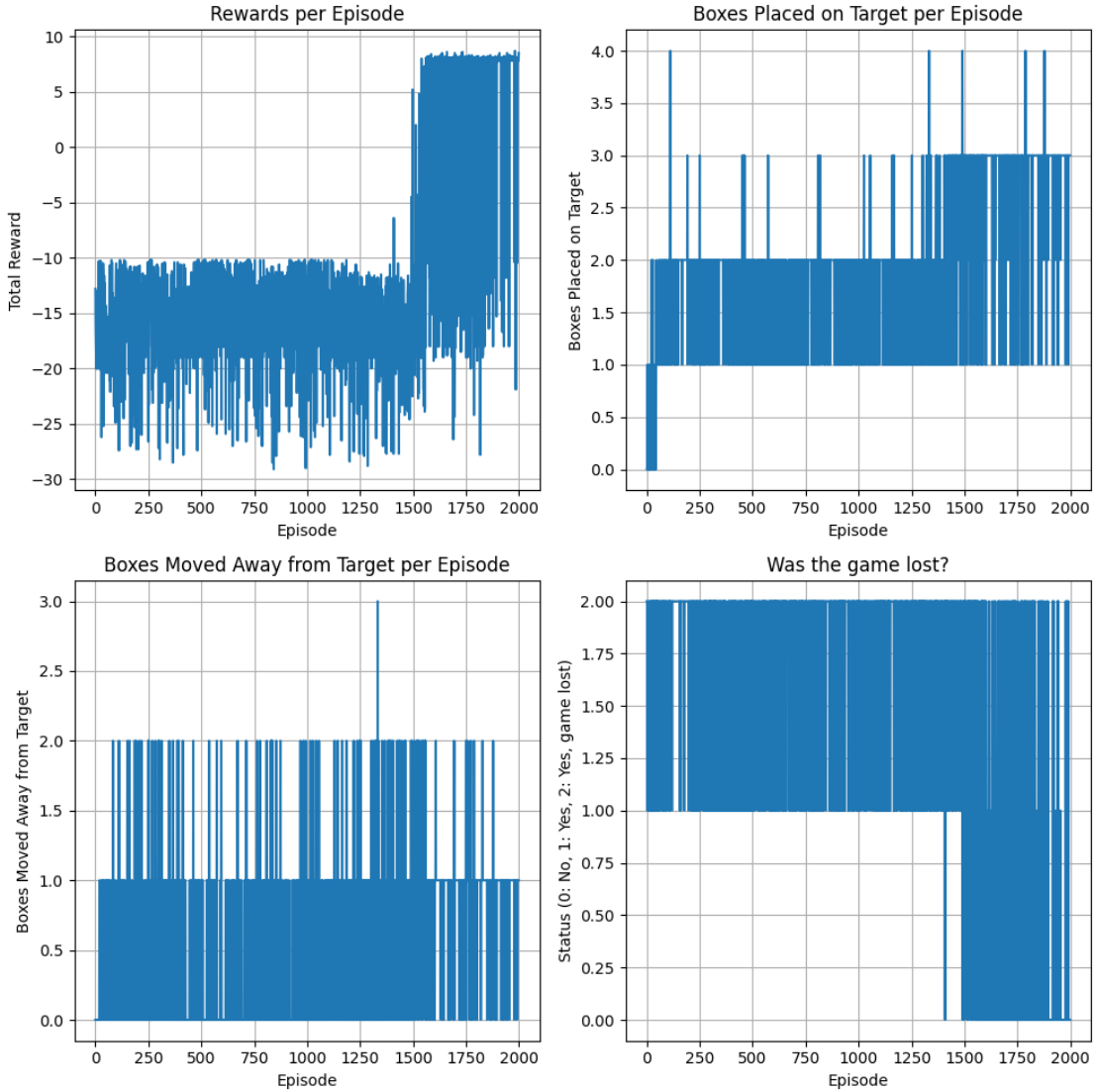
# Fonction : vérifier_fin_de_partie

1. Pour chaque case du plateau :
   - Si c'est une caisse :
     - Si la caisse est bloquée dans un coin:
     - La partie est perdue.
   - Vérifier si la caisse peut être poussée :
     1. Pour chaque direction (haut, bas, gauche, droite) :
     2. Si la case derrière la caisse et celle devant sont libres :
     3. La caisse peut être poussée. La partie n'est pas perdue.
   - Si la caisse ne peut pas être poussée :
     - Rechercher les caisses adjacentes.
     - Si aucune caisse adjacente n'est trouvée :
     - La partie est perdue.
     - Si des caisses adjacentes sont trouvées :
     - Appliquer récursivement la fonction vérifier_fin_de_partie pour chaque caisse adjacente.
2. Si on est arrivé ici alors la caisse peut être poussée, renvoyer vrai.

# Résultats du Q-learning avec détection des murs (1/4)



# Résultats du Q-learning avec détection des murs avec actions obligatoires (2/4)



# Résultats du Q-learning avec pénalité en cas d'actions inutiles (3/4)

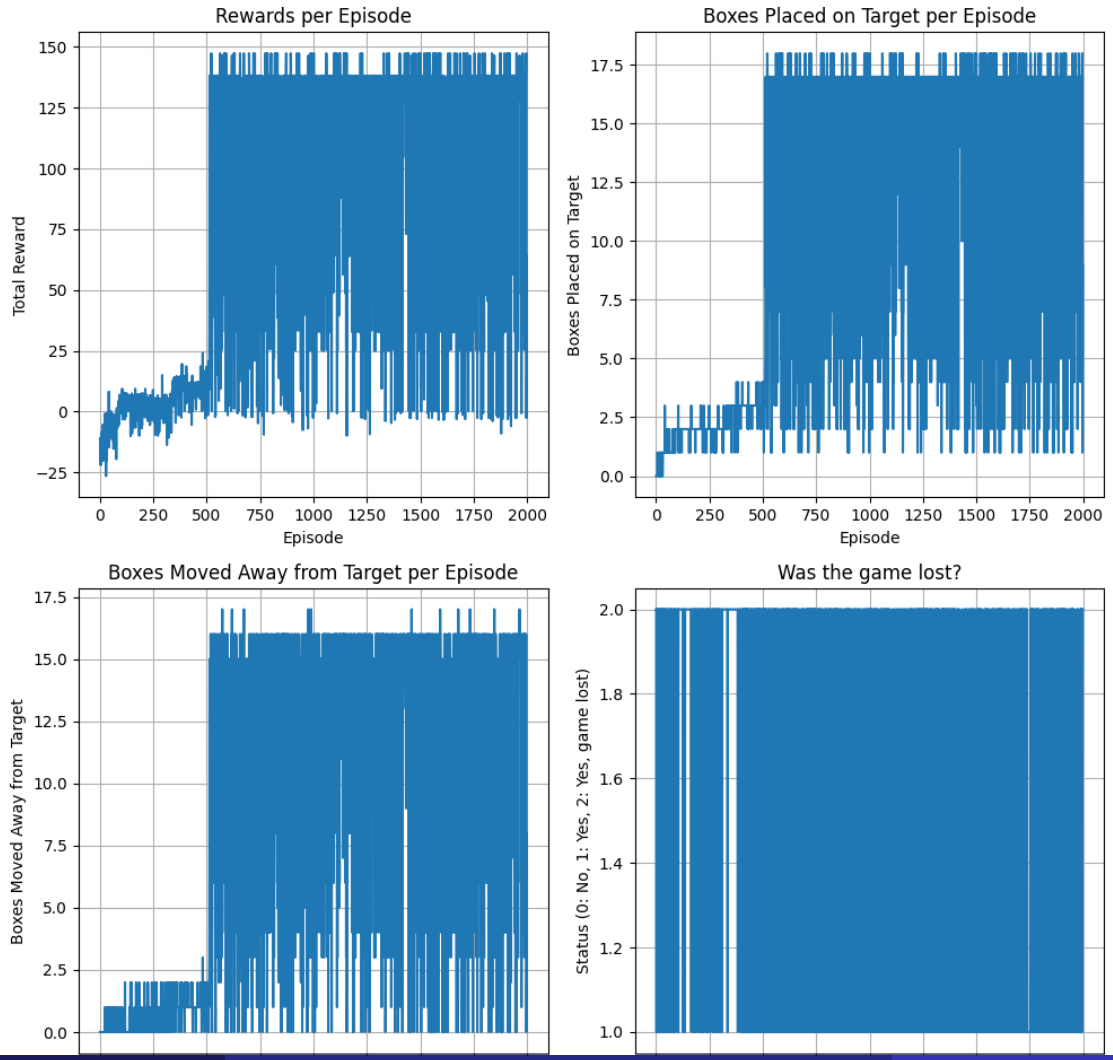
## Idée :

Mettre une pénalité en cas d'action inutile (foncer dans un mur, essayer de pousser une caisse invisible...)

## Résultats :

L'agent ne termine jamais le niveau, malgrés plusieurs valeurs de malus (de -1 à -0.2)

# Résultats du Q-learning avec détection des murs, actions obligatoires et mise à jour des récompenses (4/4)



# Recherche des meilleurs paramètres pour $\alpha$, $\gamma$ et $\epsilon$

- Alpha ($\alpha$) : Le taux d'apprentissage
- Gamma ($\gamma$) : Le facteur de remise
- Epsilon ($\epsilon$) : Le taux d'exploration

## Exécution d'un algorithme de recherche par optimisation (arbre de recherche)

Problème : durée d'exécution de 16 heures pour 20 itérations.
Utilisation des valeurs récupérées : résultat peu concluant, le nombre d'itérations n'est probablement pas significatif.

## Hyperparamètres

Meilleurs paramètres : {'alpha': 0.30000000000000004, 'gamma': 0.95, 'epsilon': 0.30000000000000004}

# Recherche des meilleurs paramètres pour le système de récompense

## Exécution d'un algorithme de recherche par optimisation (arbre de recherche)

Durée d'exécution : 12 heures 20 minutes sur 20 instances.

## Meilleures valeurs de récompense trouvées

- Récompense placement correct de boîte : 8
- Récompense placement invalide de boîte : 0
- Récompense de victoire : 0
- Récompense de défaite : -7
- Récompense de mouvement invalide : -1
- Récompense à chaque itération : -0.1497212470641976

## Résultats

L'agent pousse la boîte sur un emplacement à l'infini...

# Suite logique de ces travaux de recherche

| | |
|---|---|
| 1 | Réaliser un algorithme de détection de fin de partie et l'optimiser à l'aide d'un autre langage (C++), et / ou recoder la bibliothèque du jeu. |
| 2 | Rechercher les meilleurs paramètres possibles pour le système de récompense et pour l'exploration. |
| 3 | Tester l'agent sur des nouveaux niveaux. |
| 4 | Essayer de nouveaux modèles tel que le Deep Q-learning et les réseaux de neurones convolutionnels et les comparer. |

Merci pour votre attention.