SAE 1.02 COMPARAISON D'ALGORITHMES

UNE IA POUR ESN WARS

Préparez-vous pour le grand tournoi final des coders en folie

OBJECTIF GÉNÉRAL

L'objectif de ce projet est de concevoir une intelligence artificielle pouvant jouer au jeu ESN Wars programmé dans la SAE 1.01. On réalisera à cette fin une intelligence artificielle (IA) spécialisée dans le jeu à 4 joueurs, qui doit gagner le plus de bitcoins possibles. Un tournoi entre IA aura lieu en public afin de déterminer les meilleures IA !

A vos CLAVIERS !

LE MOTEUR DE JEU

On fournit un moteur de jeu nommé de façon assez originale `moteur_esn_wars.py`. Ce moteur est capable de simuler des parties à un nombre quelconque de joueurs sur des cartes diverses. Il est conseillé de décompresser le dossier fourni tel quel et de travailler directement dedans sans toucher à son architecture. Le jeu est prévu pour tourner sur Linux, mais l'adaptation à Windows ne devrait pas poser problème.

Pour tester le moteur de jeu, vous pouvez exécuter le fichier `main_esn_wars.py` dont le rôle est de lancer la partie. On spécifie le nom des fichiers IA qui vont participer, ainsi que le nom de la map. Tout un tas d'informations décrivant la partie défileront sur la sortie standard ; vous pouvez les rediriger dans un fichier pour les lire. Une variable booléenne RAPPORT du moteur permet d'avoir ces informations qui s'affichent ou non. Vous pouvez aussi retirer les instructions 'input' qui bloquent l'affichage sur certains événements marquants.

Ces informations ont un but de vérification ; votre IA, elle, récupèrera les informations nécessaires sous la forme d'un dictionnaire.

Malgré des tests, il se peut que ce moteur comporte encore certains bugs. Si vous en voyez, vous pouvez le signaler et nous ferons des mises-à-jour. Toutefois, le fonctionnement global du moteur et son interaction avec les IA seront inchangées.

Changement de règles. Afin de rendre le jeu plus intéressant les points suivants ont été changés par rapport à l'énoncé de la SAE 1.01 :

- *les salaires ont augmenté* ! Quand une mission est terminée, elle rapporte maintenant le carré

(SW * D)^2 bitcoins

et non plus SW*D comme précédemment.

- *c'est la période des soldes* ! La PS13 est à seulement 3000 bitcoins en ce moment ! C'est donc le nouvel objectif.

- en revanche, *les salaires des profs ont aussi augmenté* (youpi)... Le facteur multiplicatif du coût des upgrades n'est plus 10, mais 25.

LA CLASSE IA

Dans le dossier IA se trouvent les fichiers contenant les IA. L'IA que vous allez développer sera programmée en écrivant les méthodes d'une classe dont le schéma général est donnée, **IA_ESNW**. Un exemple élémentaire d'IA qui joue complètement au hasard vous est donné.

Pour écrire votre IA, vous modifiez le fichier `IA_aleatoire.py` et le copiez en changeant son nom. Il y a trois méthodes à compléter ou modifier. Surtout, ne pas changer leur nom et leurs paramètres ! Souvenez vous que les méthodes commencent toujours par

le paramètre ‘self’. En dehors de cela vous êtes libres d'ajouter des méthodes, fonctions, variables, etc.

- la méthode `__init__(self, num_joueur : int, game_dict : dict)-> None` est appelée automatiquement en début de partie pour créer l’objet IA de votre classe. Elle numéro du joueur qui lui permet de s'indentifier dans l'ensemble des joueurs, et savoir dans quel ordre on va jouer, ainsi qu'un dictionnaire qui contient les informations sur les missions et les coders en début de partie. Pour plus d'information, affichez ce dictionnaire, les champs ont des noms explicites.

- Si vous voulez ajoutez des données dans votre IA, c’est dans cette méthode que vous pouvez les initialiser, en ajoutant par exemple `self.valeur = 3` ou `self.historique = []` ou tout autre possibilité. Ceci n'impactera pas le bon fonctionnement de votre IA.

- la méthode `action(self, game_dict : dict) -> str` est appelée à chaque tour de jeu afin de prendre une décision. Elle doit répondre dans un délai qui sera de quelques secondes par une str qui doit être une des possibilités suivantes :

- 'N', 'S', 'E','W' pour se déplacer dans une des 4 directions possibles ;

- 'L' ou 'E' pour améliorer son coding Level ou son Energie max.

- 'P' pour passer.

Le programme fait des vérifications sur les actions. Si une action n'est pas possible (déplacement sur un autre joueur, en dehors du plateau, ou tentative d'amélioration en dehors du Job Center), ou bien s'il n'y a pas de réponse ou une réponse n'ayant pas de sens, on considère que le joueur passe.

Cette méthode reçoit un dictionnaire `game_dict`, qui décrit l'état du jeu au début du tour. Ce dictionnaire contient en particulier l'historique de tous les coups joués.

- enfin, la méthode `game_over(self, game_dict: dict) -> None` est appelée en fin de partie. Cette méthode ne sert pas dans la simulation du jeu mais elle peut vous être utile pour effectuer des calculs, afficher des résultats etc en fin de partie.

Globalement, vous êtes libres de modifier tous les fichiers et même le moteur, mais c'est dans ce cadre précis que votre IA doit fonctionner.

MODALITÉS DU TOURNOI

Un match de tournoi s'effectue toujours avec votre IA et trois autres IAs. Le match se déroule suivant les critères prévus (1000 tours, objectif 3000 bitcoins) et on regarde la quantité de bitcoins accumulées lors de la partie. Le score de votre IA sur ce match est le nombre total de bitcoin accumulées. Toutefois, si votre IA atteint 3000 en premier, le score est doublé (et vaut donc au moins 6000). Les matchs se déroulent sur une liste de maps aléatoires inconnues, cependant on peut assurer les points suivants :

- tous les joueurs seront confrontés aux mêmes maps dans les mêmes proportions

- chaque map comporte entre 5 et 10 missions

- les SW et Difficulté des missions sont comprises entre 1 et 5

Dans un premier temps, votre IA sera confrontée à 3 types de matchs contre des IAs standards, les mêmes pour tous les participants.

- match contre trois IA aléatoires

- match contre trois IA simples

- match contre trois IA avancées

Sa performance sur de nombreux matchs contre ces trois types d'IA donnera un score moyen qui sera transformé en une note sur 7.

Dans un second temps, votre IA sera confrontée de nombreuses fois à des IA des autres joueurs, des milliers de matchs seront simulés. Le même principe de comptage de points sera appliqué et on calculera le score moyen de votre IA et donnera une autre note sur 7. Un classement total entre toutes les IAs sera effectué.

Les 6 points restants viendront du rapport (voir plus loin).

COMMENT PEUT FONCTIONNER UNE TELLE IA ?

L'IA que vous concevez doit savoir jouer à plusieurs, et elle sera testée contre différentes autres IA pour voir son score moyen obtenu sur un grand nombre de parties. Dans un premier temps, vous pourrez la tester contre l'IA alétoire fournie, ensuite si vous « élevez » plusieurs IA vous pourrez les faire jouer entre elles avant de garder la meilleure.

Voici quelques idées pour concevoir une IA :

Une IA « faite à la main »

Avant de se lancer dans les choses compliquées, il y a quand même des choses simples à programmer… Quelques exemples :

1) repérer les missions intéressantes

2) utiliser une stratégie simple, choisir une mission, y aller, puis revenir au JC si besoin.

3) Enfin, être attentif à ce que font les autres : si un joueur laisse une mission avec très peu de workload pour aller au JC, c'est peut-être le moment d'aller lui "voler" la mission !

Et ainsi de suite, vous pouvez déjà implémenter ce genre de règles dans une IA avant de faire quelque chose de plus complexe. Il est possible de faire une IA assez complexe en utilisant ce genre de règles dites « heuristiques » en prévoyant de nombreux cas de figure.

Quelques idées pour des IA plus générales

Dans ce type de jeu, les idées de base sont simples (choisir une mission, s'y rendre) mais tout le détail est dans l'optimalité des choix effectués ! Ainsi, on est vide amenés à écrire des sortes d'IA avec choix basés sur des paramètres heuristiques (exemple : si workload * difficulté > distance * 0.345, faire ceci, etc), les détails de la formule étant basés sur des paramètres qu'il faut optimiser. Il existe différents moyens d'optimiser ces paramètres une fois que vous les avez identifiés, par exemple un générant de nombreuses IA au hasard et en gardant les meilleures, ou même en utilisant de la *programmation génétique* !

LE RAPPORT

Cette SAE étant assez libre pour vous dans la façon de procéder, il est très important que vous documentiez votre analyse, vos idées, votre façon de développer votre IA. Nous vous demandons donc d'écrire un rapport (pdf entre 4 et 10 pages) contenant :

- vos réflexions, vos analyses, vos idées pour développer une IA qui puisse jouer convenablement au jeu
- le travail que vous avez accompli, comment vous avez procédé pour créer cette IA, les tests qui ont été effectués, les modifications
- la façon dont l'IA finale que vous rendez fonctionne, de manière générale et éventuellement dans les détails

Le but de ce rapport est de pouvoir apprécier la qualité de votre travail même si les performances de votre IA ne sont pas forcément les meilleures. Ne le négligez pas.

VOTRE TRAVAIL

PENDANT LA SAÉ

- Travail en **binôme obligatoire** (exceptionnellement par 3 suivant la parité).
- Toutes les fonctions doivent être documentées avec des spécifications (docstring), et raisonnablement commentées. Les noms des fonctions et des variables doivent être judicieusement choisis. Comme précisé plus haut, les spécifications des fonctions et docstrings servent à développer intelligemment votre programme, et il est dommage voire inutile de les faire à la fin, on doit s'appuyer dessus pour bâtir le programme !
- S'aider d'un groupe à l'autre est possible, recopier un code tel quel est interdit. Nous avons des moyens informatiques de comparaison des codes (JPlag, entre autres). Le code de votre binôme doit être le vôtre, et chacun.e des membres du binôme doit être capable de l'expliquer intégralement, que ce soit « votre partie » ou non.
- Le projet doit être fonctionnel **SUR LES MACHINES DE L'IUT**. À vous de tester votre projet sur ces machines avant de le rendre.
- Vous déposerez votre projet dans la **rubrique Travaux de l'espace Initiation au Développement sur eCampus**, sous la forme d'une **archive** au format **`NOM1_NOM2.zip`** où NOM1 et NOM2 sont les noms des deux membres du binôme (ajoutez le troisième nom pour un trinôme…)
- Cette archive contiendra :
    - un fichier python nommé IA_NOM1_NOM2.py
    - le rapport nommé RAPPORT_NOM1_NOM2.pdf

Attention, le nom respect des formats ci-dessous entraîne une diminution de la note finale.

DEADLINE : vendredi 12 janvier à 23h59