SAE 1.02 COMPARAISON D'ALGORITHMES

UNE IA POUR BOMBERBUT

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 BomberBUT programmé dans la SAE 1.01. On réalisera à cette fin une intelligence artificielle (IA) qui doit maximiser son score. Votre IA jouera quelques parties en solo avant d'affronter en tournoi les IA adverses !

A vos CLAVIERS !

LE MOTEUR DE JEU

On fournit un moteur de jeu composé de deux fichiers nommés `bb_ia_start.py` et `bb_modele.py`. Le premier fichier permet de lancer une partie où les joueurs sont des IAs. Le deuxième fichier contient toute la logique du jeu.

Ce moteur est capable de simuler des parties à un nombre quelconque de joueurs sur des cartes diverses, avec l'ensemble des règles. Il est conseillé de décompresser le dossier fourni tel quel et de travailler directement dedans sans toucher à son architecture.

Pour tester le moteur de jeu, vous pouvez exécuter le fichier `bb_ia_start.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/scénario. 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.

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 assez fortement que ce moteur comporte encore certains bugs. Si vous en voyez, vous pouvez le signaler et nous ferons des mises-à-jour.

LE RÉPERTOIRE MAPS

Ce répertoire des fichiers "training" qui sont prévus pour un seul joueur, et des fichiers "battle" prévus pour 4 joueurs.

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_Bomber**. 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 deux méthodes à compléter et 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. dans votre fichier.

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

des noms explicites. Les paramètres timerglobal et timerfantome correspondent au règles du jeu (nombre total de tours, et fréquence d'apparition des fantômes).

- 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 (return) dans un délai qui sera de quelques secondes par une str qui doit être une des possibilités suivantes :

- 'H', 'B', 'G','D' pour se déplacer dans une des 4 directions

- 'X' pour poser une bombe

- 'N' pour ne rien faire et passer.

Le programme fait des vérifications sur les actions. Si une action n'est pas possible (déplacement non valide etc), on considère que le joueur passe.

Cette méthode reçoit un dictionnaire `game_dict`, qui décrit l'état du jeu au début au moment de décider l'action.

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

MATCHS SOLO

Dans un premier temps, votre IA sera lancée sur des maps à un joueur, ressemblant aux scénarios "training", de difficulté croissante.

- une map sans fantômes, de petite taille, avec beaucoup de temps, où il faut récupérer quelques minerais

- une map sans fantômes de plus grande taille, où il faut récupérer un maximum de minerai en temps limité

- une map avec quelques fantômes

- une map avec beaucoup de fantômes

Le score obtenu par votre IA sur ces maps donnera une première partie de la note.

MATCHS À QUATRE JOUEURS

Ensuite, votre IA sera lancée en tournoi avec 3 joueurs adverses. De nombreux matchs auront lieu entre les IA de tous les participants et un classement sera obtenu, pour une autre partie de la note.

LE MINI 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 1 et 3 pages) contenant :

- vos réflexions, vos analyses, vos idées pour développer une IA qui puisse jouer convenablement au jeu
- 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.