Dossier de validation Intelligence Artificielle pour le jeu Avalam

Groupe 2 :
Mollin Florian
Suzan Jérémie
Nadji Karim
Csomor Barnabas
Vincent Alexandre
Seguret Aymeric

Semestre 6 - Licence 3 INFO - 2013/2014

Table des matières

1 Introduction

Dans le cadre du projet PROG6, nous devions créer une version informatique du jeu de stratégie Avalam et développer une intelligence artificielle afin qu'un utilisateur puisse jouer contre l'ordinateur.

1.1 Les premiers choix

Tout d'abord nous avons réfléchi à ce que nous voulions faire pour nos IA. Nous voulions que l'IA facile ne pose aucun problème aux débutants, que pour battre l'IA moyenne, l'utilisateur doive commencer à réfléchir à certaines stratégies, et que l'IA difficile soit quasiment imbattable.
Nous avons donc réfléchi aux différents algorithmes pour créer nos IA :

— Pour l'IA facile : effectuer un coup aléatoire tiré dans l'ensemble des coups possibles suivant la configuration actuelle du plateau.

— Pour l'IA moyenne et difficile : utiliser un algorithme $\alpha\beta$ avec une heuristique évaluant l'intérêt du coup et une profondeur déterminée.

Après quelques tests et réflexions, nous avons constaté que l'IA facile était trop simple à battre, nous sommes donc parti sur une version de l'IA facile avec un algorithme $\alpha\beta$ de profondeur égale à 1.
Nous avons dans un premier temps, développé l'IA facile afin de la faire tester aux utilisateurs pour connaître leurs avis.

1.2 Les premiers avis des utilisateurs

Nous avons élaboré un questionnaire afin de connaître l'avis des utilisateurs sur notre jeu (sur l'IHM, sur l'IA, ...).
Nous avons donné ce questionnaire à un panel d'utilisateurs de tout âge (19 à 55ans) et de domaine professionnel différents, car nous voulions développer notre jeu pour n'importe quel type d'utilisateur et non pour des joueurs avancés.
Grâce à ce questionnaire, nous avons récupéré des résultats à propos de l'IA facile (car l'IA difficile et moyenne n'étaient pas encore implémentées).
Voici certains avis à propos de l'IA que nous avons reçu :

— « Mettre le niveau facile plus facile »

— « Mettre l'ordinateur facile un peu moins fort »

— « L'ordinateur facile est trop compliqué »

Grâce à ces résultats et aux observations que nous avons faites lorsque les utilisateurs jouaient, nous avons décidé de nous inspirer de l'IA facile pour créer notre IA moyenne et de modifier son algorithme afin de la rendre plus facile.

2 Tester la cohérence des IA

Pour tester la cohérence des IA, nous avons développé un paquetage permettant de faire s'affronter nos IA l'une contre l'autre, et d'en tirer des statistiques pour tester leur cohérence.
Notre paquetage effectue les tests sur un nombre de "matchs" donnés. Grâce à ce paquetage nous pouvons vérifier la cohérence des IA suivant leur pourcentage de victoire face aux autres IA.

3 L'algorithme alpha-bêta

3.1 Une première version de l'algorithme

Nous avons réfléchi à une première version de l'algorithme $\alpha\beta$ à implémenter qui est la suivante :

Fonction ValeurAlphaBeta (première version)

```
ValeurAlphaBeta(p : Plateau, j : Joueur, alpha : entier, beta : entier, estMax :
    booleen, pmax : entier, h : heuristique) : entier
    //Calcul la valeur de p pour le joueur j, selon que p estMax ou non.
    //Et avec pmax la profondeur maximale
    Si partieGagnante(p, j) alors renvoyer +∞
    Sinon si partiePerdante(p, j) alors renvoyer −∞
    Sinon si partieNulle(p, j) alors renvoyer 0
    Sinon si pmax = 0 alors renvoyer h(p, j)
    Sinon si estMax alors
        Pour s parmi les successeurs de p faire
            alpha = max(alpha, ValeurAlphaBeta(s, j, alpha, beta, !estMax, pmax-1))
            Si alpha >= beta alors
                Retourner alpha
        Retourner alpha
    Sinon
        Pour s parmi les successeurs de p faire
            beta = min(beta, ValeurAlphaBeta(s, j, alpha, beta, !estMax, pmax-1))
            Si beta <= alpha alors
                Retourner beta
        Retourner beta
```

Fonction DecisionAlphaBeta (première version)

```
DecisionAlphaBeta(p : Plateau, j : Joueur, pmax : entier) : coup
    //decide du meilleur coup a jouer par le joueur j avec le plateau p
    alpha = −∞
    Pour chaque coup de coupsJouables(p) faire
        val = ValeurAlphaBeta(appliquer(coup, p), j, alpha, +∞, faux, pmax)
        Si val > alpha alors
            action = coup
            alpha = val
    Retourner action
```

Mais cette version de l'algorithme $\alpha\beta$ s'est révélée beaucoup trop lente, nous avons donc modifier cette version afin de l'accélérer.

3.2 La version finale de l'algorithme

La première version de l'algorithme est trop longue pour plusieurs raisons. Tout d'abord, à chaque appel de la fonction `ValeurAlphaBeta`, une copie du plateau est faite, ce qui prend beaucoup de temps pour la copie. Pour palier ce problème, nous allons créer qu'une seule instance du plateau de jeu, et jouer des coups à chaque appel de la fonction `ValeurAlphaBeta`, et annuler le dernier coup joué à la fin de celle-ci.
De plus, pour améliorer encore les performances de l'algorithme, nous allons appliquer un filtre aux successeurs du plateau afin de supprimer les coups "bêtes" et de ne garder que les coups "intelligents", ce qui permettra de réduire le nombre d'appel dans l'arbre de l'algorithme et ainsi de réduire son temps d'exécution.

Fonction ValeurAlphaBeta

```
ValeurAlphaBeta(p : Plateau, j : Joueur, alpha : entier, beta : entier, estMax :
    booleen, pmax : entier, h : heuristique, f : filtre, c : Coup) : entier
    //Calcul la valeur de p pour le joueur j, selon que p estMax ou non.
    //Et avec pmax la profondeur maximale
    Si partieGagnante(p, j) alors renvoyer +∞
    Sinon si partiePerdante(p, j) alors renvoyer −∞
    Sinon si partieNulle(p, j) alors
        annulerCoup(p, c)
        renvoyer 0
    Sinon si pmax = 0 alors renvoyer h(p, j)
    Sinon si estMax alors
        coupsJouables = coupsJouables(p)
        coupsJouables = f(coupsJouables, j)
        Pour chaque coup de coupsJouables faire
            appliquerCoup(p, coup)
            alpha = max(alpha, ValeurAlphaBeta(p, j, alpha, beta, !estMax, pmax-1,
                f, coup))
            Si alpha >= beta alors
                annulerCoup(p, c)
                Retourner alpha
        annulerCoup(p, c)
        Retourner alpha
    Sinon
        coupsJouables = coupsJouables(p)
        coupsJouables = f(coupsJouables, j)
        Pour chaque coup de coupsJouables faire
            appliquerCoup(p, coup)
            beta = min(beta, ValeurAlphaBeta(p, j, alpha, beta, !estMax, pmax-1, f,
                coup))
            Si beta <= alpha alors
                annulerCoup(p, c)
                Retourner beta
        annulerCoup(p, c)
        Retourner beta
```

4 Les IA finales

4.1 L'IA facile

4.1.1 Le principe

Le principe de l'IA facile est de jouer des coups aléatoires tirés parmi tous les coups jouables hormis les coups ayant pour destination une tour de sa propre couleur.

4.1.2 L'algorithme

L'algorithme de cette IA est donc l'algorithme $\alpha\beta$ avec une profondeur de 1.

4.1.3 L'heuristique

Pour cette IA, il n'y a pas d'heuristique car il s'agit de coups aléatoires.

4.1.4 Le filtre

Ici le filtre consiste à éliminer tous les coups "bêtes" : les coups ayant pour destination un pion de la couleur de l'IA.

4.2 L'IA moyenne

4.2.1 Le principe

Le principe de l'IA moyenne est de poser des difficultés à l'utilisateur sans pour autant le battre à tous les coups. Il ne possède pas de filtre, mais possède une heuristique qui augmente suivant l'écart du nombre de points entre l'IA et son adversaire.

4.2.2 L'algorithme

L'algorithme de cette IA est l'algorithme $\alpha\beta$ avec une profondeur de 2.

4.2.3 L'heuristique

Pour cette IA, l'heuristique attribut des points selon la différence de score entre les deux joueurs.

4.2.4 Le filtre

Il n'y a pas de filtre pour cette IA, car elle est assez rapide avec une telle profondeur.

4.3 L'IA difficile

4.3.1 Le principe

Le principe de l'IA difficile est de battre quasiment tout le temps un joueur expérimenté. Son heuristique varie au cours de la partie.

4.3.2 L'algorithme

L'algorithme de cette IA est l'algorithme $\alpha\beta$ avec une profondeur de 3.

4.3.3 L'heuristique

Pour cette IA, l'heuristique varie au cours de la partie. Cette heuristique attribut des points suivant la différence de score entre les deux joueurs, et le nombre de tours isolées ou de tours de 5. Suivant l'avancement de la partie, l'heuristique attribut un facteur plus ou moins important à la différence de score entre les deux joueurs et aux nombres de tours isolées ou de tours de 5.

4.3.4 Le filtre

Ici le filtre consiste à ne garder que les tours ayant un nombre de voisins inférieur ou égal à 4, et de ne garder que les coups rapportant des tours de 5 ou des tours isolées.

5 La cohérence des IA

Grâce au paquetage que nous avons créé, nous avons obtenu un tableau récapitulant les matchs des IA. Pour ces statistiques, 100 matchs ont été fait entre chaque IA. Voici les résultats :

Matchs	Scores cumulés	Pourcentages victoire
Facile - Moyen	665 - 694	25% - 75%
Facile - Difficile	499 - 809	0% - 100%
Moyen - Difficile	503 - 807	0% - 100%

Grâce aux résultats apportés par notre programme de test, nous pouvons voir la cohérence des IA : que l'IA moyenne gagne contre la facile, et que l'IA difficile gagne contre la facile et la moyenne.

6 Les temps d'exécution

En plus de vérifier la cohérence des IA, nous devions vérifier leurs temps d'exécution, en effet, les temps de réflexion des IA ne doivent pas être trop long pour l'utilisateur. Voici les temps d'exécution des IA :

IA	Temps maximum	Temps moyen
Facile	80ms	4ms
Moyen	2330ms	16ms
Difficile	3300ms	199ms

D'après les tests effectués auprès des utilisateurs, le temps d'attente est convenable pour chacune des IA.

7 Conclusion

Au final, nous avons des IA cohérentes et avec un temps d'exécution rapide (donc acceptable pour les utilisateurs), de plus nous avons respecté ce que nous voulions faire dès le début, que l'IA facile permette à l'utilisateur comprenant les règles de le battre, que l'IA moyenne gagne contre les utilisateurs novice, et que l'IA difficile gagne les utilisateurs les plus avancés.