Université du Québec à Montréal

BotGammon: Joueur artificiel de backgammon

Travail présenté à
M. Éric Beaudry

Dans le cadre du cours
INF4230-10 – Intelligence Artificielle

Marc-Étienne Déry DERM06099201
Dominick Latreille LATD19099100
François Bilodeau BILF31089104
Philippe Pépos Petitclerc PEPP03049109
Équipe O4L

15 décembre 2014

1 Introduction

L'application que nous présentons est un joueur artificiel de backgammon. Plusieurs techniques d'intelligence artificielle ont été développées pour résoudre ce genre de jeu, et de nouvelles techniques apparaissent encore dans les temps récents au fur et à mesure que diverses techniques d'IA évoluent. Nous utilisons le logiciel libre GNU Backgammon afin d'avoir une interface nous permettant de tester notre application contre différents niveaux d'intelligence artificiel. Ce logiciel est très complet et comprend une panoplie de fonctions nous permettant de tester différentes situations.

L'idée d'un joueur artificiel pour le jeu de backgammon nous a semblé intéressante tant au niveau du processus de création qu'au niveau du résultat final. La présence du hasard, associée à un bon niveau de stratégie rend ce jeu de table original et stimulant. Le joueur artificiel permet à un joueur apprenti de se pratiquer et d'évaluer ses capacités, en fonction du niveau de l'intelligence de l'agent.

2 Problématique

Le type de problème à résoudre lorsqu'on développe un joueur artificiel de backgammon est un problème de prise de décision avec probabilité. Il s'agit d'évaluer le meilleur coup dans la limite des coups possibles en tenant compte des probabilités des lancers de dés.

Le backgammon est un environnement **entièrement observable**, puisqu'on voit le plateau de jeu en tout temps. Le joueur artificiel fait face a un environnement **stochastique** puisqu'il doit brasser deux dés avant de pouvoir jouer. Il ne sait donc jamais avec certitude la distance que ses pions parcourront, mais il a accès aux probabilités des différents lancers de dés possibles. Ensuite, l'environnement est **multi-agent**, car le backgammon se joue à deux joueurs. Il est également **séquentiel**, puisque chaque décision a un effet sur les décisions futures des deux joueurs. De plus, le backgammon est un environnement **statique**, car aucun changement à l'envionnement ne peut être effectuer par un agent externe et parce qu'il n'y a pas d'horloge, contrairement aux échecs dans certains cas. Pour finir, le joueur artificiel fait face à un environnement **discret**, puisque les endroits où l'on peut déposer les pions sont bien définis et que chaque coup joué est distinct d'un autre.

Le défi dans la conception d'un joueur artificiel de backgammon réside dans le fait que le nombre de position est très élevé, alors la solution naïve qui consiste à vérifier le meilleur coup dans une table de positions n'est pas réalistement faisable. Pour calculer le nombre de positions légales possibles sur le plateau de jeu, posons :

(1) C(n,m) = n!m!(n-m)! qui représente le nombre de façons de sélectionner m pions dans un ensemble de n pions. (2) D(n,m) = C(n+m-1,m) qui représente le nombre de façons de distribuer m pions de la même couleur sur n "flèches"

$$C(n,m) = n!m!(n-m)! \tag{1}$$

représentant le nombre de façons de sélectionner m pions dans un ensemble de n pions et :

$$D(n,m) = C(n+m-1,m) \tag{2}$$

représentant le nombre de façons de distribuer m pions de la même couleur sur n *flèches*

En supposant que les pions noirs occupent m flèches sur les 24 possibles, il y a donc $C(24,m)$ façons de sélectionner les flèches, pour m pions. Le reste des $15-m$ pions peuvent être sur les mêmes flèches que les m précédents, sur le bar ou à l'extérieur du plateau ; le moyen de les distribuer est alors $D(m+2,15-m)$. Les pions noirs ont $26-m$ endroits disponibles, alors ils peuvent être distribués en $D(26-m,15)$ façons. Le nombre total de positions possibles est donc :

$$m = \sum_{m=0}^{15} C(24,m) \times D(m+2,15-m) \times D(26-m,15) \tag{3}$$

$$m = 18528584051601162496 \tag{4}$$

Ajoutons à cela le hasard qu'amènent les dés, le backgammon est loin d'être trivial à jouer pour un ordinateur.

3 Pertinence de la technique d'intelligence artificielle

Parmi les multiples techniques d'intelligence artificielle existantes, celles qui sont les plus utilisées pour le cas du jeu de Backgammon sont la stratégie de prise de décisions expectiminimax et l'apprentissage par réseaux de neurones (comme c'est le cas pour GNU Backgammon). Ces approches sont très intéressantes, mais nous avons décidé d'opter pour l'algorithme expectiminimax avec élagage alpha-bêta et profondeur itérative pour notre IA puisqu'il fonctionne avec les jeux à somme nulle comme le backgammon, en plus d'intégrer des éléments de hasard. C'est aussi une question de temps ; nous désirions obtenir un bon résultat le plus rapidement possible, sans avoir à laisser un algorithme d'apprentissage tourner pendant des heures ou des jours. L'heuristique au backgammon peut être sans cesse améliorée, et c'est donc ce sur quoi nous nous sommes penchés.

L'algorithme expectiminimax est une variation de minimax qui possède en plus du noeud "min" et "max" un noeud "chance". Ce noeud chance représente l'élément de probabilité du problème associé, qui est dans notre cas les deux dés lancés avant chaque coup. Tout comme le minimax, cet algorithme va créer un arbre récursif et alterner entre le joueur "min" pour qui l'on cherche à minimiser son gain et le joueur max que l'on

veut maximiser. Par contre, le noeud chance sera appliqué avant chaque position hypothétique du plateau dans l'arbre, donc avant le joueur "min" et avant le joueur "max", comme indiqué par un cercle sur le schéma ci-contre. Contrairement aux noeuds '"min" et "max" qui prennent les valeurs d'utilité de leurs enfants, le noeud de chance prend comme valeur la probabilité d'avoir une combinaisons de dés quelconques. Pour le backgammon, nous avons une probabilité de 1/18 d'avoir deux dés différents ou 1/36 d'avoir un double. La complexité spatiale d'expectiminimax est O(ndbd) plutôt que O(bd) pour minimax, où n équivaut au nombre de possibilité des résultat de chance.

Pour la profondeur itérative, c'est une méthode qui nous permet de faire une recherche le plus profond possible avec le temps alloué pour l'IA. Dans le cas où l'on ne réussit pas à terminer une itération à une certaine profondeur par manque de temps, on prend le résultat de la précédente itéraion. Nous utilison l'élégage alpha-beta pour notre Expectminimax. Cela nous permet d'éliminer certain noeud qui vont avoir une valeur inférieur que le noeud que l'on vient d'évaluer et donc réduire le nombre de branchement.

Afin de rendre la tâche possible, nous avons pris en considération certaines hypothèses. Tout d'abord, nous considérons qu'il faut 1 point seulement pour gagner(1 partie = 1 point), contrairement aux parties de tournoi qui demandent habituellement entre 3 et 7 points pour gagner. De plus, nous n'utiliserons pas le dé doubleur. Le fait de ne pas pouvoir doubler nous permet d'avoir un gain en rapidité pour chaque coup puisque nous n'avons pas besoin de calculer la probabilité de gagner avant de brasser les dés. clearpage

4 Résultats

Afin d'évaluer l'algorithme implémenté, nous avons effectué xxx parties en jouant des coups légaux aléatoires contre certains niveaux de difficulté de GNU Backgammon. Ce logiciel est considéré comme l'un des meilleurs joueurs artificiels de backgammon et peut utiliser des bases de données de plusieurs Go associées à des réseaux de neurones. Pour ce projet, nous y sommes allés avec un intelligence de base pour comparer au nôtre. Nous comparons ces résultats à yyy parties jouées avec notre algorithme expectiminimax.

Voici les explications pour les différents niveaux de difficulté de GNU Backgammon :

— **Beginner** : utilise un bruit déterministe de 0.060 à l'évaluation.

— **Casual play** : utilise un bruit déterministe de 0.050 à l'évaluation.

— **Intermediate** : utilise un bruit déterministe de 0.030 à l'évaluation.

— **Advanced** : utilise un bruit déterministe de 0.015 à l'évaluation.

Il existe d'autres niveaux d'intelligence (il peut par exemple évaluer à l'avance jusqu'à 7 roulement de dés dans l'arbre des possibilités).

Difficulté	Heuristique	Pourcentage de victoire	Nombre de parties	Profondeur
Beginner	Aucune	1%	1000	1
Beginner	Heuristique v1	30%	1000	1
Beginner	Heuristique v1	0%	20	2

5 Conclusion

Pour conclure, *include resumé des resultats*. Il sera toujours possible d'améliorer notre AI en améliorant l'heuristique puisqu'il est difficile de faire un heuristique parfait pour le backgammon. De plus, il serait possible de faire de l'elagage comme gnubg (a completer) pour eliminer le nombre de branchements.

6 Répartition des tâches

Dominick : Document, présentation, fonctions pour la grille de jeu, expectiminimax. François : Heuristique, mouvement de pion, document, tests et résultats, présentation Marc-Étienne : Expectiminimax, liste des coups possibles, structure du programme, interaction avec GNU Backgammon, tests et résultats Philippe : Document