Dauphine | PSL
UNIVERSITÉ PARIS

Recherche Monte-Carlo et jeux

M. Tristan Cazenave - Master IASD

Recherche Monte-Carlo pour le jeu de dames

Auteurs :
Noé LALLOUET
Marc MEDLOCK

13 juillet 2022

Table des matières

1 Introduction

1.1 Le jeu de dames

Le jeu de dames est un jeu de plateau se pratiquant à deux joueurs. La taille du plateau est de 8×8. Tour à tour, les deux opposants déplacent chacun de leurs 12 pions selon la règle suivante : il est possible de déplacer un pion sur une case diagonale adjacente. Quand un pion se trouve diagonalement adjacent à un pion adverse, il doit le prendre en "sautant par dessus" si la case suivante est libre (voir Figure 1). Plusieurs prises de pièces sont permises dans le même tour si elles sont effectuées par la même pièce. La prise de pièce n'est pas optionelle lorsque le mouvement est possible.

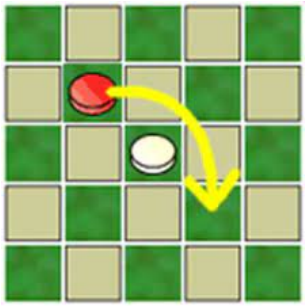

Figure 1 – Prise de pion

Lorsqu'une pièce se retrouve sur la rangée extérieure opposée à sa moitié de départ, elle se transforme en roi. Un roi peut se déplacer d'autant de cases diagonales qu'il le souhaite. Un joueur remporte la partie quand il a pris toutes les pièces de son adversaire. Si plus de 40 mouvements consécutifs sont effectués sans qu'il y ait de prise de pièce, un match nul est déclaré.

Dans l'objectif de la création d'un agent jouant au jeu de dames, l'approche recherche Monte-Carlo est justifiée. En effet, le nombre d'états possibles au cours d'une partie est grand. Soient $0 \leq w \leq 12$ le nombre de pièces blanches et $0 \leq b \leq 12$ le nombre de pièces noires. Le nombre de configurations possibles d'un jeu de dames ayant w pièces blanches et b pièces noires vaut :

$$\binom{32}{w}\binom{32-w}{b}2^{w+b}$$

Ainsi, le nombre total de configurations est :

$$\sum_{b=0}^{12}\sum_{w=0}^{12}\binom{32}{w}\binom{32-w}{b}2^{w+b},$$

soit approximativement 2.3×10^{21}. Il est bien entendu irréaliste de tenter d'explorer tous les états, comme le ferait une recherche minimax, car une évaluation de tous les états possibles serait bien trop gourmande en mémoire. C'est pourquoi il est possible d'utiliser des heuristiques développées par des experts, qui permettent d'estimer la qualité d'un état sans avoir à jouer la partie jusqu'à la fin. Cependant, le développement de telles heuristiques demande le savoir d'experts et n'est pas forcément optimal.

1.2 Recherche Monte-Carlo

La recherche Monte-Carlo permet d'apprendre à un agent à jouer au jeu de dames sans avoir à développer d'heuristique et en un temps relativement court. Le principe de la recherche Monte-Carlo est le suivant : l'agent utilise des parties aléatoires pour construire progressivement un arbre de recherche correspondant aux résultats.

L'algorithme de recherche Monte-Carlo se décompose en quatres étapes principales : la **sélection**,

l'**expansion**, le **playout** et la **rétropropagation**.

Sélection
L'étape de sélection se charge de sélectionner des nœuds de l'arbre de recherche selon une politique donnée jusqu'à atteindre un nœud feuille. Cette politique est le plus souvent fonction du nombre de victoires des nœuds.

Expansion
Lorsqu'un nœud feuille n est atteint, l'étape d'expansion crée des nœuds enfants de n à partir des coups possibles pour cet état.

Playout
L'étape playout joue une partie en effectuant des mouvements aléatoires jusqu'à atteindre un état terminal. Le résultat de la partie est ensuite retourné.

Rétropropagation
Etant donné un résultat de playout r, l'étape de rétropropagation ajoute r aux totaux de victoires (ou défaites) de tous les nœuds traversés pendant l'étape de sélection.

Ces quatre étapes sont répétées pendant un nombre fixé d'itération ou pendant une durée fixée. Ensuite, l'algorithme choisit un mouvement à effectuer parmi les mouvements possibles et en fonction des scores et/ou du nombre d'explorations de chacun.

2 Méthodes

Ici, nous détaillons trois algorithmes de recherche Monte-Carlo que nous avons implémenté dans le cadre de ce projet : **UCT**, **RAVE** et **GRAVE**.

2.1 UCT

UCT est un algorithme de recherche Monte-Carlo très performant, qui utilise la théorie des bandits pour sélectionner les nœuds. En effet, la méthode utilise le procédé *Upper Confidence Bound* et l'applique à l'arbre de recherche.

Plus précisément, UCT définit la politique utilisée dans l'étape de **sélection** et qui sert à parcourir l'arbre à la recherche du nœud qui paraît le plus intéressant à explorer. Ainsi, pendant l'étape de sélection, le nœud est choisi parmi les enfants C d'un nœud de la manière suivante :

$$\text{node} = \text{argmax}_{i \in C}\, \mu_i + k \cdot \sqrt{\frac{\log t}{n_i}}$$

où :

μ_i = pourcentages de victoires après le coup i
k = constante à ajuster
t = nombre de fois que le nœud a été visité
n_i = nombre de visites de l'enfant i.

La politique de sélection permet ainsi de garder un équilibre entre l'exploration et l'exploitation : un noeud sera sélectionné si il aboutit à un bon pourcentage de victoires, mais les noeuds peu explorés seront aussi considérés.

2.2 RAVE

La recherche arborescente de Monte Carlo basique ne recueille pas suffisamment d'informations pour trouver les coups les plus prometteurs qu'après de nombreux tours ; jusque-là, ses coups sont essentiellement aléatoires. Cette phase exploratoire peut être réduite de manière significative dans une certaine classe de jeux en utilisant RAVE (Rapid Action Value Estimation) [1]. Dans ces jeux, les permutations d'une séquence de coups conduisent à la même position. Il s'agit généralement de jeux de plateau dans lesquels un coup implique le placement d'une pièce ou d'une pierre sur le plateau. Dans ces jeux, la valeur de chaque coup n'est souvent que légèrement influencée par les autres coups.

Dans RAVE, pour un nœud d'arbre N donné, ses nœuds enfants C_i stockent non seulement les statistiques des victoires dans les parties commencées au nœud N, mais aussi les statistiques des victoires dans toutes les parties commencées au nœud N et en dessous, si elles contiennent le coup i (également lorsque le coup a été joué dans l'arbre, entre le nœud N et une partie). De cette façon, le contenu des nœuds

de l'arbre est influencé non seulement par les coups joués immédiatement dans une position donnée, mais aussi par les mêmes coups joués ultérieurement.

Lorsqu'on utilise RAVE, l'étape de sélection choisit le nœud pour lequel la formule UCB1 modifiée

$$(1 - \beta(n_i, \tilde{n_i})) \cdot \frac{w_i}{n_i} + \beta(n_i, \tilde{n_i})) \cdot \frac{\tilde{w_i}}{\tilde{n_i}} + c \cdot \sqrt{\frac{\log t}{n_i}}$$

a la valeur la plus élevée. Dans cette formule, $\tilde{w_i}$ et $\tilde{n_i}$ représentent respectivement le nombre de parties gagnées contenant le coup i et le nombre de toutes les parties contenant le coup i, et la fonction $\beta(n_i, \tilde{n_i})$ doit être proche de un pour des n_i relativement petits et proche de zéro pour des grands $\tilde{n_i}$. Une des nombreuses formules pour $\beta(n_i, \tilde{n_i})$, proposée par Gelly et Silver, dit que dans les positions équilibrées on peut prendre :

$$\beta(n_i, \tilde{n_i}) = \frac{\tilde{n_i}}{n_i + \tilde{n_i} + 4b^2 n_i \tilde{n_i}}$$

où b est une constante choisie empiriquement. Ici, nous choisissons b tel que $4b^2 = 1e^{-5}$.

2.3 GRAVE

Le principe de GRAVE [2] est d'utiliser les valeurs AMAF d'un état situé plus haut dans l'arbre que l'état actuel. Il y a un équilibre entre la précision d'une estimation et la conformité à l'état actuel de l'estimation. Un état situé plus haut dans l'arbre a une meilleure précision car il a plus de jeux associés. Cependant, les statistiques concernent un état différent quelques coups auparavant et sont moins précises pour l'état inférieur de l'arbre que les statistiques des états inférieurs. Le principe de GRAVE est de n'utiliser que les statistiques qui ont été calculées avec plus d'un certain nombre de coups. Nous prenons comme état de référence l'état ancêtre le plus proche qui a plus de lancements qu'une constante ref donnée. L'état de référence peut être l'état actuel si le nombre de lancements de l'état actuel est supérieur à ref.

3 Expériences

3.1 Tournoi

Afin d'évaluer les implémentations des trois algorithmes que nous avons présenté en section 2, nous avons décidé d'organiser un tournoi entre les algorithmes. Ce tournoi est au format *round-robin* : il oppose tous les joueurs à tous les autres joueurs. Chaque confrontation est composée de plusieurs parties, afin de tenter de limiter l'impact de l'aléatoire sur les résultats. Nous nous assurons que les deux joueurs jouent autant de parties du côté noir que du côté blanc.

Les résultats du tournoi peuvent être appréciés dans la table 1.

	UCT	RAVE	GRAVE	Total
UCT	-	43.3%	46.4%	44.8%
RAVE	56.7%	-	57.1%	**56.9**%
GRAVE	53.5%	42.9%	-	48.2%

Table 1 – Résultats du tournoi

3.2 Observations

Il est intéressant de remarquer que UCT est l'algorithme le plus faible des trois. Nous voyons que RAVE est l'algorithme le plus performant, alors que RAVE peine à tirer son épingle du jeu. Nous pouvons fournir les observations suivantes :

— **Nombre de parties :** Seules 60 parties ont été jouées entre chaque algorithme, ce qui laisse penser que les résultats affichés ont une variance élevée. Les limitations matérielles dont nous avons été victimes ont prévenu le déroulement d'un nombre de parties plus élevé.

— **Budget d'itérations :** En raison de limitations matérielles, nous avons été contraints de restreindre le nombre de simulations aléatoires à un nombre de l'ordre de la centaine. Nous avons conscience que ce nombre est extrêmement faible : en particulier, il ne permet pas à GRAVE d'opérer normalement car la constante ref n'est soit jamais atteinte, soit de valeur trop faible.

Afin d'obtenir des statistiques plus représentatives, nous avons organisé une revanche de RAVE contre GRAVE, avec un budget d'itérations plus élevé, d'une valeur de 500. La constante ref a été initialisée avec une valeur de 10, après un rapide ajustement empirique. Les deux algorithmes ont joué 6 parties en tant que noir et 6 parties en tant que blanc. A l'issue de ce match, GRAVE affiche un taux de succès de 66% contre RAVE. Ce score, plus réaliste en raison du budget d'itérations plus important, montre que GRAVE est plus performant dans le cadre du jeu de dames. Cependant, le résultat est à remettre dans le contexte du nombre de parties effectuées qui reste faible.

4 Conclusion

Nous avons implémenté trois algorithmes performants de recherche Monte-Carlo dans le contexte du jeu de dames : UCT, RAVE et GRAVE. Nous avons analysé les fonctionnements de chaque algorithme et évalué leurs performances dans le cadre d'un tournoi *round-robin*.

Malgré des contraintes matérielles restreignant le nombre de simulations possibles, nous proposons des résultats servant de comparaison entre les trois méthodes susmentionnées. Dans des travaux futurs, nous pourrions explorer des algorithmes alternatifs de recherche Monte-Carlo, comme SHUSS [3] ou Nested Monte-Carlo Search [4].

Références

[1] Sylvain Gelly et David Silver. « Monte-Carlo tree search and rapid action value estimation in computer Go ». In : (2011). URL : https://www.lamsade.dauphine.fr/~cazenave/mcts-gelly-silver.pdf.

[2] Tristan Cazenave. « Generalized Rapid Action Value Estimation ». In : (2015). URL : https://www.lamsade.dauphine.fr/~cazenave/papers/grave.pdf.

[3] Nicolas Fabiano et Tristan Cazenave. « Sequential Halving Using Scores ». In : (2021). URL : https://www.lamsade.dauphine.fr/~cazenave/papers/SHUSS.pdf.

[4] Tristan Cazenave. « Nested Monte-Carlo Search ». In : (2009). URL : https://www.lamsade.dauphine.fr/~cazenave/papers/nested.pdf.