

Étude Pratique : Développement d'une Intelligence Artificielle à base de l'algorithme Monte Carlo Tree Search

Francesco BARIATTI Adrien GASTÉ Mikael LE
Romain LEBouc
Encadrant : Pascal GARCIA

Année scolaire 2015/16

Résumé

Ce document est un compte-rendu du projet d'études pratiques réalisé dans le cadre de la 3^e année de formation à l'INSA de Rennes.

Ce projet consiste à programmer une Intelligence Artificielle (IA) pour jouer à un jeu de plateau (le Jeu du Pingouin), et une interface graphique pour interagir avec le jeu. Le projet s'étale sur toute la durée d'une année scolaire.

1 Introduction

1.1 Le Jeu du Pingouin

Le Jeu du Pingouin est un jeu de plateau confrontant 2 à 4 joueurs sur un plateau de 60 cases hexagonales, sur lesquelles se trouvent de 1 à 3 poissons, comme présenté dans la figure 1.

Chaque joueur place 4 pingouins sur le plateau en début de partie. À chaque tour, il en déplace un dans l'une des 6 directions possibles, en récupérant la case sur laquelle le pingouin se trouvait. Il gagne alors autant de points qu'il y a de poissons dessus.

Les pingouins ne peuvent pas passer à travers des autres pingouins (y compris ceux du même joueur) et des trous créés par les déplacements des pions. Lorsqu'un joueur ne peut pas jouer, ceux pouvant encore jouer continuent.

Le jeu se termine lorsque aucun des pingouins ne peut se déplacer, et le joueur avec le plus de points remporte la partie.

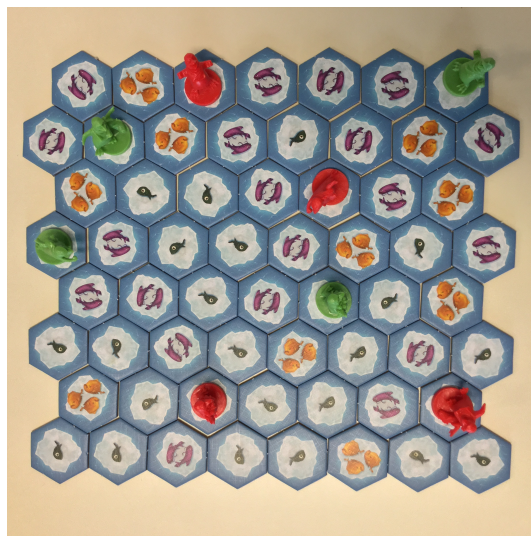


FIGURE 1 – Plateau du Jeu du Pingouin

1.2 L'algorithme Monte-Carlo Tree Search (MCTS)

Le *Monte-Carlo Tree Search* est un algorithme de recherche de décision, utilisé dans les jeux tel que le Go ou encore Ms. Pacman. Son principe repose sur la simulation de plusieurs millions de parties qui permettent de construire progressivement un arbre et d'ensuite choisir le meilleur chemin pour le tour courant du joueur.

Les implémentations peuvent avoir des différences (comme le score donné à une partie gagnante ou perdante, ou les valeurs stockées dans les noeuds), mais le principe de base est toujours le même est il est composé de 4 étapes qui sont répétées pour construire l'arbre du jeu :

- **Sélection** : En considérant un arbre partiellement construit suite à plusieurs simulations, un chemin est alors choisi par un calcul se servant des valuations aux $n_{\frac{1}{2}}$ uds, permettant ainsi d'explorer les meilleurs chemins en priorité, jusqu'à une feuille. Ce principe repose sur le tirage aléatoire pondéré.
- **Expansion** : À partir du $n_{\frac{1}{2}}$ ud considéré, ses enfants sont alors développés puis un est choisi au hasard pour la suite.
- **Simulation** : Une simulation des prises de décision pour chacun des joueurs est alors faite aléatoirement depuis cet enfant jusqu'à la fin du jeu. Le résultat de cette simulation est alors observé.
- **Rétropropagation** : À chaque $n_{\frac{1}{2}}$ ud est associé un score de 2 nombres : le premier est le nombre de parties gagnées par l'IA, le deuxième est le nombre total de parties jouées sur la branche courante. Le score des $n_{\frac{1}{2}}$ uds se trouvant sur le chemin entre la racine et le fils choisi par l'expansion est alors mis à jour pour prendre en compte le résultat de la simulation.

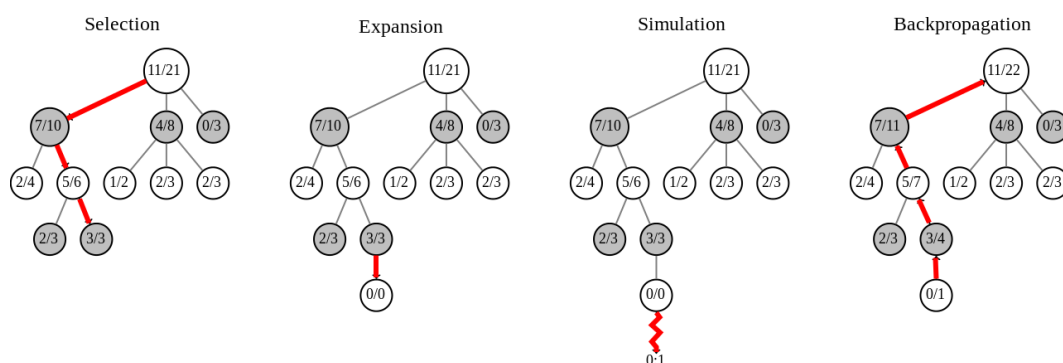


FIGURE 2 – Étapes du MCTS

L'un des avantages indéniables de l'algorithme est qu'il peut être interrompu à tout moment : le choix de la branche optimale sera fait à partir de l'arbre construit jusqu'à ce moment là. De plus, c'est un algorithme sans heuristique, c'est à dire qu'il n'a pas besoin de connaître au préalable les règles du jeu pour bien jouer.

2 Tâche à réaliser

La tâche à réaliser est de programmer le jeu du Pingouin en Langage C++ et y implémenter le MCTS comme IA. Le mode Joueur contre IA est imposé. Il faut également créer une interface utilisateur pour rendre le programme accessible à tous.

2.1 Implémentation du jeu

L'algorithme à implémenter dans le programme est le MCTS. L'algorithme a déjà été programmé par notre encadrant Pascal GARCIA en C++, mais il nécessite d'une représentation efficace du jeu pour pouvoir faire ses simulations.

C'est donc à nous de faire cette représentation de sorte à qu'elle respecte l'interface définie par Pascal GARCIA et qu'elle comporte des calculs rapides pour passer d'une étape à l'autre du jeu.

2.2 Création d'une interface graphique

Pour permettre de rendre l'application facile à utiliser, une interface graphique doit être programmée ; les interactions Homme-Machine se font à la souris. Il n'y a pas de restriction sur la méthode utilisée. Il va de soi que l'interface graphique doit pouvoir lancer le jeu et communiquer les coups du joueur à l'IA et inversement.

3 Réalisation

A chaque séance, nous nous sommes généralement divisés en 2 équipes de 2 afin d'avancer plus rapidement le projet sur deux points différents. Lorsque nous avons l'occasion, nous rencontrons notre encadrant afin qu'il donne son avis ainsi que des conseils pour des problèmes que nous n'arrivions pas à résoudre.

Le projet a été effectué à l'aide de *Git* pour faciliter l'accès aux différentes versions du code.

3.1 Prise en main du MCTS avec le Tic-Tac-Toe

Afin de comprendre et tester le fonctionnement du MCTS, nous avons décidé, pendant le 1er semestre, de l'implémenter sur un jeu simple : le Tic-Tac-Toe. Cela nous a également permis d'apprendre à programmer en C++, langage qu'on ne connaissait pas, et dans lequel le MCTS est codé.

Pascal GARCIA nous a conseillé de représenter la grille sous forme de *bitboards*¹ de 16 bits pour optimiser les calculs, l'un représentant les croix et l'autre les cercles. Les états gagnants étaient des *bitboards*, définis dans le code comme des entiers. Lorsque l'un des *bitboards* satisfaisait (selon certaines opérations bit à bit) un de ces états, la partie se terminait.

3.2 Le Jeu du Pingouin

La deuxième étape du projet consiste à coder le Jeu du Pingouin de telle sorte que l'IA respecte les règles et comprenne la condition de victoire et que les calculs pour effectuer chaque coup soient rapides, pour que le MCTS puisse simuler énormément de parties dans le temps imparti.

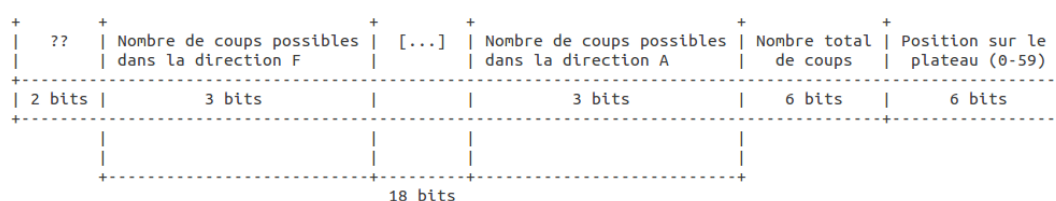
3.2.1 Choix de représentation

Chacun des pingouins a été modélisé par un *bitboard* de 32 bits. Le plateau a été représenté à l'aide de 3 *bitboards* de 64 bits (chacun représentant la présence de 1, 2 ou 3 poissons sur les cases) : la présence d'obstacles sur une case est calculée grâce à des opérations bit à bit (*and*, *or*, *not*)

Il a fallu confronter le problème du déplacement des pions qui n'existait pas dans le Tic-Tac-Toe : en effet, il n'a pas été évident de relier le déplacement sur le plateau (en six directions possibles) avec la représentation du plateau en *bitboard*. La solution retenue a été de numérotter les 60 cases du plateau et de faire correspondre le déplacement de chacune des 6 directions par un calcul arithmétique.

De plus, la modélisation optimale des pingouins a été trouvée difficilement car il a fallu associer plusieurs types d'informations différentes à chacun des pingouins (par exemple, le nombre de déplacements possibles dans une direction).

1. Une bitboard est juste un nombre entier classique, dans lequel chaque bit est interprété de façon particulière. Souvent chaque bit représente la présence, ou l'absence, d'un élément dans une case du plateau

FIGURE 3 – Découpage du *bitboard* pour un pingouin

Une solution envisagée a été de mettre chaque type d'informations dans un *bitboard* en particulier, mais cela s'est révélé trop difficile à gérer. Nous avons alors opté pour stocker toutes les informations concernant un pingouin dans un *bitboard* personnel.

3.2.2 Avantage de la représentation en bitboards

L'enjeu principal pour l'implémentation du jeu était la rapidité : plus un coup était rapide plus de parties le MCTS pourrait simuler pendant son tour. c'était donc nécessaire que pour avoir le contenu d'une case, ou la position d'un pingouin le calcul soit rapide. Il était donc hors de question de stocker les valeurs dans un tableau dans la mémoire, les *bitboards* permettent d'utiliser la puissance des calculs bit à bit(exécutables en un cycle de processeur).

La recherche d'information se fait avec des masques appliqués sur les variables : par exemple l'entier 63 (les 6 bits de poids faible à 1) appliqué sur un pingouin permet d'obtenir sa position. Le *bitboard* des obstacles (un 1 dans les cases où il y a soit de l'eau, soit un pingouin) est calculé dynamiquement à chaque tour en combinant les trois *bitboards* des poissons et les positions des pingouins, ce qui permet d'avoir une valeur en moins à stocker et mettre à jour.

3.3 L'interface graphique

L'IA nécessite d'avoir les coups qu'on veut jouer exprimés comme numéro par rapport aux coups totaux disponibles pour un joueur : cette partie de conversion nécessaire était la raison principale pour le développement d'une interface graphique, vu que c'était un travail fastidieux et très difficile à faire à la main à chaque fois qu'on veut jouer un coup.

Le choix du langage pour implémenter l'interface graphique est tombé vers un langage qu'on maîtrise mieux que le C++ : le Java. elle est donc développée avec la bibliothèque *javafx* qui permet d'avoir un modèle *Modèle*, *Vue*, *Contrôleur* qui va faciliter la reprise dans le futur.

L'interface devait être facile à l'utilisation et "jolie". On n'avait pas de bases ni d'indications particulières, on a donc commencé de zéro pour cette partie.



FIGURE 4 – Interface graphique du Jeu du Pingouin

Communication avec l'IA Les deux programmes étant indépendant, il faut une façon pour que les données du jeu passent du programme en C++, qui gère le jeu et l'IA, à l'interface en Java. Pour ce faire on a décidé d'utiliser une représentation en JSON, et de transmettre l'état entier du jeu après chaque coup. L'interface graphique fait le travail inverse de l'IA : elle convertit les différentes valeurs des *bitboards* de l'état en tableaux et objets qui sont ensuite utilisés pour l'affichage.

Difficultés La principale difficulté dans le développement de l'interface graphique a été la mise en place du plateau et de l'architecture MVC : une période de recherche d'informations sur le javafx a été nécessaire. Une fois les bases mises en place l'implémentation des différentes fonctionnalités (comme le fait que les mouvements possibles apparaissent en une couleur différente) a été plus rapide.

4 Conclusion

Il y a eu quelques difficultés rencontrées au début du projet : le langage utilisé par Pascal GARCIA pour coder l'IA, à savoir le C++, nous était inconnu, et la complexité de transcrire les règles d'un jeu de plateau de tel sorte que le programme puisse les comprendre était importante. Nous avons alors consacré notre 1er semestre à implémenter ce programme dans un jeu plus simple, le Tic-Tac-Toe, afin de se familiariser avec ces concepts. Cette décision nous a permis d'avoir un développement beaucoup plus rapide pour l'IA du jeu du pingouin, ce qui nous a laissé le temps nécessaire pour développer une interface graphique conventionnelle.

Le produit obtenu satisfait entièrement le cahier des charges : le mode Joueur vs IA a été implémentée avec succès, ce dernier possédant le niveau d'un joueur expérimenté. Il serait intéressant de rajouter des fonctionnalités supplémentaires, tel que choisir le niveau de difficulté, voire même utiliser notre expérience acquise sur un autre jeu.

Nous remercions Pascal GARCIA, notre encadrant, pour sa disponibilité et ses conseils.