

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

BARIATTI Francesco LE Mikael LEBOUC Romain
GASTÉ Adrien

2015 - 2016

Contents

1	Introduction	1
1.1	Le Jeu du Pingouin	1
1.2	L'algorithme Monte-Carlo Tree Search (MCTS)	1
2	Étude Pratique	4
2.1	Tâche à réaliser	4
2.2	Implémentation du MCTS	4
2.3	Création d'une interface graphique	4
3	Réalisation	5
3.1	Prise en main du MCTS avec le Tic-Tac-Toe	5
3.2	Création de l'IA pour le Jeu du Pingouin	5
3.3	Intégration de l'interface graphique pour le jeu	6

Introduction

Les études pratiques sont des projets réalisés chaque année pas les élèves du département Informatique de l'INSA de Rennes qui s'étalent sur 10 mois.

Cette étude pratique se présente sous la forme d'une Intelligence Artificielle (IA) à créer pour un jeu de plateau, qui sera jouable via une interface graphique.

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 1.1.

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.

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.

La construction de cet arbre est composée de 4 étapes :

- la sélection : En considérant un arbre partiellement construit suite à plusieurs simulations, un chemin est alors choisi par un calcul se servant



Figure 1.1: Plateau du Jeu du Pingouin

des valuations aux noeuds, permettant ainsi d'explorer des choix moins bons, jusqu'au dernier noeud qui est une feuille; ce principe repose sur le tirage aléatoire pondéré.

- l'expansion : À partir du noeud considéré, il développe ses enfants puis on en choisit un au hasard.
- la simulation : Il simule alors des prises de décision pour chacun des joueurs aléatoirement depuis cet enfant (la feuille courante) jusqu'à la fin du jeu. Il observe ensuite quel joueur a gagné la simulation.
- la rétropropagation : À chaque noeud est associé un score de 2 nombres : le premier est le nombre de parties gagnées par l'IA, le 2ème est le nombre total de parties jouées sur la branche courante. Après l'étape précédente, on met à jour le score de chaque noeud de l'arbre en remontant du noeud courant à la racine.

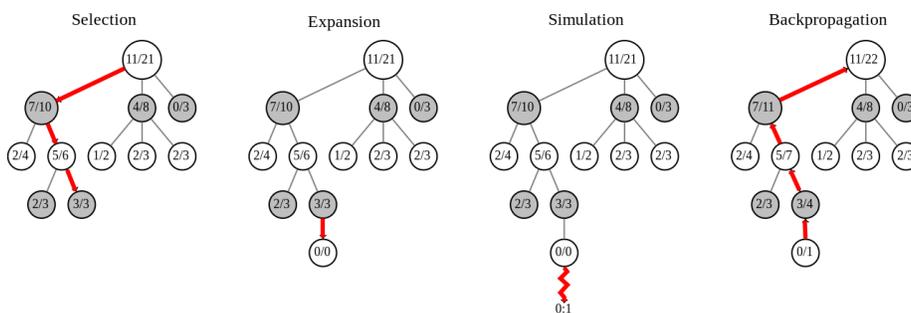


Figure 1.2: Étapes du MCTS

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 déjà construit. 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 être bon.

Étude Pratique

2.1 Tâche à réaliser

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

2.2 Implémentation du MCTS

L'algorithme à implémenter dans le programme est le MCTS. Il a déjà été programmé par notre encadrant Pascal GARCIA en C++. C'est donc à nous de le faire interagir avec le programme du jeu afin que l'IA choisisse la meilleure solution en fonction du coup fait par le joueur humain.

2.3 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.

Réalisation

A chaque séance, nous nous sommes généralement divisés en 2 équipes de 2 afin d'avancer plus rapidement le projet sur 2 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, à savoir le Tic-Tac-Toe. Cela nous a également permis d'apprendre à programmer en C++, le langage utilisé pour coder l'algorithme.

Pascal GARCIA nous a conseillé de représenter la grille sous forme de *bitboards* de 16 bits pour optimiser les calculs, l'un représentant les croix et l'autre les cercles. Les états gagnants étaient sous forme d'entiers et lorsque l'un des *bitboards* satisfaisait un de ces états, la partie se terminait.

3.2 Création de l'IA pour 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.

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) que nous avons ensuite complété avec la position des pingouins.

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 déplacement sur le plateau et déplacement et sa représentation en *bitboard*. La solution retenue a été de numéroter 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).

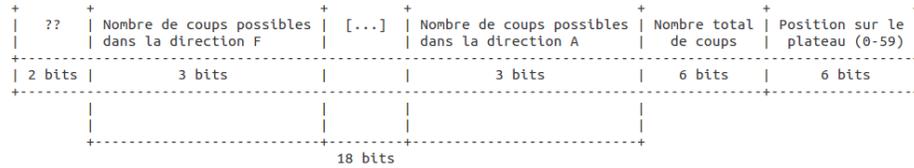


Figure 3.1: Découpage du *bitboard* pour un pingouin

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 de stocker toutes les informations concernant un pingouin dans un *bitboard* personnel.

3.3 Intégration de l'interface graphique pour le jeu

Conclusion

Remerciements