The map coloring game
Final report

Arnaud Audoin Clément Badiola Samuel Da Silva
Alexandre Perrot Camille Ritlewski

Client : Paul Dorbec

24 janvier 2013

Table des matières

Chapitre 1

Introduction

1.1 Graph coloring

Graph coloring is not a new problem in mathematics and computer science. The first results in the field date back to 1852, when Francis Guthrie postulated the famous four color theorem.

Coloring a graph consists of assigning a color to each vertex of the graph, such that two adjacent vertices are of different colors. A such coloring is called a proper coloring. There are other forms of coloring, such as edge or face coloring, but they always reduce to a vertex coloring on another graph.

A graph is said to be k-coloriable if it admits a proper coloring with k colors. The minimal number of colors needed to achieve a proper coloring of a particular graph is called the chromatic number of that graph, noted $\chi(g)$. Deciding if a graph is k-coloriable for a given value of k is NP-complete.

La coloration de graphe est un problème qui date du XIXème siècle. Il s'agit d'assigner à chaque sommet d'un graphe des couleurs afin que deux sommets adjacents (reliés par une arête) ne soient pas de la même couleur. Il existe des variantes consistant à colorier les arêtes ou les faces d'un graphe, mais elle se ramènent toutes à une coloration de sommets. Le nombre minimal de couleurs pour obtenir une bonne coloration sur un graphe donné est appelé le nombre chromatique de ce graphe. Un graphe est dit k-coloriable si son nombre chromatique est <=k. Décider si un graphe est k-coloriable est un problème NP-complet.

1.2 Map-coloring game

Le sujet de l'article est une variante de la coloration de graphe impliquant deux joueurs. La règle de coloration est la même, chaque sommet ne peut avoir la même couleur qu'un de ses voisins.

Le premier joueur essaie de colorier correctement le graphe. Le second essaie de l'en empêcher sans pour autant pouvoir outrepasser la règle de coloration. Au début du jeu, on définit un nombre N de couleur qui pourrons être utilisé lors de la partie. La partie s'arrête si tous les sommets sont coloriés ou alors si il n'est plus possible de colorier le graphe sans ajouter une N+1 eme couleur, ce qui serait contraire aux régles.

On définit le "game chromatic number" comme le nombre minimal de couleurs avec lesquelles le premier joueur peut réussir à achever une coloration correcte du graphe quelque soit la stratégie du second.

1.3 Exemple d'une partie

1.4 Applications

Le jeu du coloriage de graphe n'a pas d'application directe, mais il aide a avoir une meilleure compréhention du domaine de la coloration de graph. La coloration de graphe à des applications diverse parmis lesquelles : - l'allocation mémoire, - gestion d'occupation de salle, - gestion d'emploie du temps....

Chapitre 2

Théorie

Chapitre 3

Travail

Sommaire

3.1 Monte Carlo

Monte-Carlo methods are a set of randomized algorithms. The general idea is to use random sampling rather than computing to help solve a problem. Such algorithms are mostly used when computing a solution to the problem is not feasible. Because of the use of random sampling, they are heuristical by nature : their results are not always guaranteed to be exact. Throughout the years, Monte-Carlo methods have been successfully used in physics, mathematics and computer science.

In this report, we will be interested in a special case of Monte-Carlo method : Monte-Carlo tree search, aka MCTS. It applies Monte-Carlo principles to tree exploration. The main field of application is artificial intelligence, especially games with very large move space where a complete tree exploration is not possible in reasonable time, such as Chess and Go.

MCTS is composed of four steps : selection, expansion, simulation and backpropagation. Thoses steps are repeated until time runs out, the maximum number of simulated games has been played, a satisfying result is found or whatever condition is relevant. Generally, the more iterations are run, the more accurate is the result.

The selection step chooses a node to play a simulation from. Several algorithms exist for this stage, depending on the context. The expansion step produces childs of the selected node if not already generated. This step makes the search tree grow in height. The simulation step plays the game until the end following nodes selected with a selection algorithm which can be the same as the one used in the selection step or a different one. Finally, th backpropagation step uses the outcome of the simulated game to modify the estimated value of

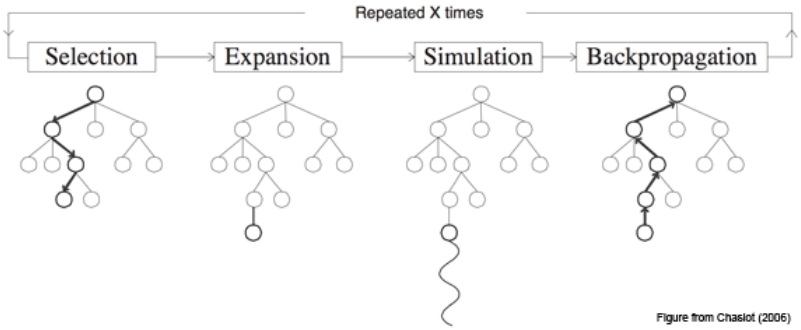

FIGURE 3.1 – Etapes de Monte-Carlo tree search

the traversed nodes. This is the crucial step to know what has been played and if the selected moves good choices or not.

According to Guillaume Chaslot [2] : "MCTS is applicable if at least the following three conditions are satisfied : (1) the payoffs are bounded (the game scores are bounded), (2) the rules are known (complete information), and (3) simulations terminate relatively fast (game length is limited)". Those three criterias are verified in the case of the map-coloring game. We will use Monte-Carlo tree search to implement an AI capable of playing Alice or Bob.

3.2 How our AI works

Dans Monte Carlo, on va chercher à jouer beaucoup de parties pour un graph donné (que nous appellerons le graph étudié) afin de pouvoir jouer les meilleurs coups possibles. Pour chaque graph étudié, nous générons un arbre des coups joués où chaque nœuds, sauf la racine, correspondent à des coups joués sur ce graph.

Ce que contient un nœud de l'arbre généré :

- le nœud joué dans le graph étudié
- la couleur jouée sur ce nœud
- le nombre de partie jouées en passant par ce nœud
- le nombre de partie gagné par Alice en passant par ce nœud

Au début l'arbre ne contient que la racine et les premiers coups possibles. Nous n'avons aucune connaissance des coups qui amènent à colorier proprement le graph étudié ni de ceux qui nous mettrons dans une situation bloquante.

Afin de limiter le nombre de fils, nous allons éliminé des coups équivalents : nous considérons que les couleurs sont numérotées et qu'il est impossible de jouer une couleur n, si la couleur n-1 n'a pas encore été jouée. En effet il est équivalent d'introduire la couleur n-1 ou la couleur n.

D'autre part nous allons généré le graph au fur et a mesure des parties et des coups joués. À chaque fois que l'on est dans un nœud (i.e. on viens de jouer le coup correspondant dans le graph étudié), si il n'a pas de fils, on regarde l'état de la partie :

- soit la partie est finie (Alice a gagné ou Bob à gagné), on remonte alors jusqu'à la racine en mettant à jour tous les nœuds sur le chemin. Si Alice a

gagné (i.e. le graph étudié est entièrement colorié) on incrémente le nombre de partie joué et le nombre de partie gagné. Si Alice n'a pas gagné (i.e. le graph étudié n'est pas entièrement colorié) on incrémente juste le nombre de partie joué.
- soit on peut continuer (auquel cas on génère les fils et on continue la simulation récursivement).

Quand on a joué dans un nœud, on a un pourcentage de victoire pour le coup correspondant, mais il n'est pas forcément représentatif. Le fait de jouer plus de parties va permettre de diminuer l'intervalle de confiance de ce ratio de victoire.

Pour éviter de toujours passer par les mêmes nœuds, nous allons essayé passer par tous les frères d'un nœud déjà visité avant de pouvoir le revisiter pour explorer ces fils. Ce qui compte, ce sont surtout les coups au premier niveau (le niveau que l'on est en train d'étudier, les coups à jouer immédiatement). Les coups de niveau inférieur ne pourront pas être tous explorés, mais seront regardés de plus près quand on avancera dans le jeu et qu'ils arriveront au premier niveau. La partie délicate de Monte-Carlo est la sélection de nœud pour l'exploration. Dans un premier temps nous avons considéré qu'il faut en priorité visiter le nœud qui a le moins de parties jouées, car il a été visité moins de fois que ses « frères ». Mais jouer celui le moins regardé pour l'instant n'est clairement pas la bonne solution sur le long terme. Il faut une solution adaptative, qui va jouer les coups moins regardés au début, puis qui va jouer les meilleurs pour vérifier que ce sont bien les meilleurs. Nous allons donc utiliser les algorithmes de Multi-armed bandit, plus particulièrement UCB1, qui donne de bons résultats. (ressource en anglais : http://www.chrisstucchio.com/blog/2012/bandit_algorithms_vs_ab.html) (ressources en français : http://researchers.lille.inria.fr/~munos/master-mva/lecture03.pdf, http://fr.slideshare.net/cornec/bandits-algo-klucb-par-garivier) La seconde partie de Monte Carlo est l'exploitation. Une fois qu'on a effectué assez de simulations, on choisit un nœud pour le jouer "pour de vrai". Là encore, il y a plusieurs solution : on peut en effet prendre le nœud avec le meilleur ratio brut, mais on peut aussi prendre le nœud avec le meilleur ratio au mieux (plus l'intervalle de confiance) ou encore celui avec le meilleur ratio au pire (moins l'intervalle de confiance). Sinon, pour la différence entre Alice et Bob, on se base sur une sélection min-max classique. Donc pour Alice on choisit le coup qui a le plus grand nombre de partie gagnée. Et pour Bob on choisit celui qui a le plus petit nombre de partie gagnée.

3.3 Mise en oeuvre de l'experience

Chapitre 4

Bilan

Sommaire

4.1 Bilan

Texte ici.

4.1.1 Échecs

Texte ici.

4.2 Si c'était à refaire...

Si c'était à refaire, nous aborderions le problème différemment...

4.3 Perspectives

Annexe A

Diagrammes

Annexe B

Manuel de maintenance

Annexe C

Extraits de code

Bibliographie

[1] G. Chaslot, J. Saito, B. Bouzy, J. Uiterwijk et H. Van Den Herik : Monte-carlo strategies for computer go. *Proceedings of the 18th BeNeLux Conference on Artificial Intelligence, Namur, Belgium*, pages 83–91+, 2006.

[2] Guillaume Chaslot : *Monte-Carlo Tree Search*. Thèse de doctorat, Université de Maastricht, 2010. http://www.unimaas.nl/games/files/phd/Chaslot_thesis.pdf.

Table des figures

Liste des tableaux

Rapport de projet d'études et de recherche.
The map coloring game.
Master 2, 2012

Les sources de ce rapport sont disponibles librement sur