

Rapport du Projet de théorie de Graphe

Sacha LIGUORI, Nicolas SURBAYROLE

14 mars 2017

Table des matières

1	Introduction	2
2	Compréhension et modélisation	2
2.1	Programme de jeu d'échec	2
2.2	Réseau Bayésien	2
2.3	Extension aux tâches en coût inconnu	4
3	Ordonnancement séquentiel	5
3.1	Tri topologique	5
4	Ordonnancement parallèle	5
4.1	Amélioration de l'ordonnancement avec heuristique de priorité	5
5	Extension aux tâches et ressources hétérogènes	5

1 Introduction

2 Compréhension et modélisation

2.1 Programme de jeu d'échec

L'algorithme qui est utilisé dans le cas du jeu d'échec est un algo de minimax. Le but de cet algorithme est de choisir le moins mauvais choix, en considérant que l'on joue en permanence le meilleur coup pour nous et que l'adversaire va jouer le plus mauvais coup pour nous. Le score calculé au nœud terminé est donné par un autre algorithme qui doit être capable d'évaluer un score pour le plateau proposé.

Les nœuds intermédiaires sont scorés en fonction du prochain joueur. Si le nœud correspond à un demi-coups de l'adversaire, on va lui donner le score minimum que l'on peut trouver parmi ses fils. Dans le cas de nos propres demi-coups, il va affecter le score du meilleur des fils. Le choix du coup solution sera fait à la racine en choisissant le coup ayant le meilleur score.

L'algorithme du minimax a le défaut de demander beaucoup de ressource (algorithme en $O(e^x)$ en fonction du nombre de possibilité par demi-coup). Dans le cas d'un calcul à une profondeur de 12 avec 30 choix par demi-coups, on obtient donc environ $30^{12} \simeq 531 \times 10^{15}$ cas terminaux. Il peut être intéressant de mettre en place une optimisation de mémorisation afin d'éviter de recalculer les coups déjà calculés. Il peut aussi être envisagé de faire une présélection des coups les plus intéressants en exécutant l'algorithme avec une profondeur plus faible et en supprimant les coups qui retournent à une position passée.

2.2 Réseau Bayésien

D'après les lois de probabilité, on sait que $P(M) = P(M \wedge A \wedge P) + P(M \wedge A \wedge \neg P) + P(M \wedge \neg A \wedge P) + P(M \wedge \neg A \wedge \neg P)$

Pour le jeu de Monty Hall, le premier choix du candidat est indépendant de la position de la clef (aucune information n'est fournie au candidat). De la même manière, la clef est placée dans la boîte avant que le candidat effectue son choix. Il y a donc indépendance entre C_1 et B_G . Le présentateur va donc choisir une des boîtes restantes vides. La position de la clef et le premier choix du candidat va donc amener la probabilité de B_R à 1 dès que le candidat

ne choisit pas la boîte contenant la clef lors de son premier choix. B_R est donc fortement lié à C_1 et B_G . C_2 est aussi lié à B_R car il est impossible que $B_R = C_2$.

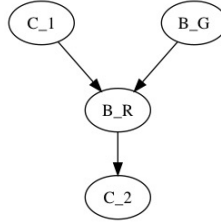


FIGURE 1 – Dépendance des variables aléatoires dans le jeu de Monty Hall

On simplifie le problème en supposant que le candidat choisit la boîte rouge et décide de changer de choix.

C_2 / B_G	Rouge	Vert	Bleue	Probabilité marginale
Rouge	1/6	1/6	1/6	1/2
Vert	1/6	1/6	1/6	1/2
Bleu	0	0	0	0
Probabilité marginale	1/3	1/3	1/3	1

$$P(C_2 = B_G) = P(C_2 = \text{Rouge} \wedge B_G = \text{Rouge}) + P(C_2 = \text{Vert} \wedge B_G = \text{Vert})$$

$$P(C_2 = B_G) = \frac{1}{3} + \frac{1}{3} = \frac{2}{3}$$

Le jeu peut être simplifié de la manière suivante : peut importe le premier choix du candidat, la boîte qu'il choisit à une probabilité de $\frac{1}{3}$ de contenir la clef contre $\frac{2}{3}$ pour le groupe de boîtes non choisies. Le présentateur mène alors une action qui change uniquement la cardinalité du groupe non choisi par le candidat. La seule boîte qui reste donc dans ce groupe à donc désormais une probabilité de $\frac{2}{3}$ de contenir la clef. L'action de changement de boîte a donc $\frac{2}{3}$ de probabilité d'être fructueuse.

De manière plus formelle :

$$P(C_1 = B_G) = \frac{1}{3} \quad (1)$$

$$P(C_2 = B_G | C_1 \neq C_2 \cap C_1 = B_G) = 0 \quad (2)$$

$$P(C_2 = B_G | C_1 \neq C_2 \cap C_1 \neq B_G) = 1 \quad (3)$$

$$P(C_2 = B_G | C_1 \neq C_2) = \frac{2}{3} \quad (4)$$

2.3 Extension aux tâches en coût inconnu

La modélisation sous forme d'un DAG le temps d'attente à la correction d'un partiel ou le temps de traitement d'un dossier dans l'administration peut mener à des tâches au coût inconnu.

L'ordonnancement s'effectue en faisant les a_N et N_x premières tâches en même temps puis b_N et N_x dernières, il en résulte donc un temps de $T_2(N) = 2N$.

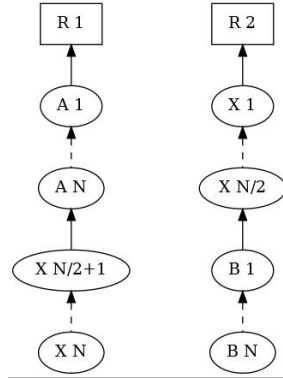


FIGURE 2 – Graphe d'ordonnement

3 Ordonnancement séquentiel

3.1 Tri topologique

Y contient les noeuds non numérotés mais dont tous les prédécesseurs le sont, Z contient tous les noeuds numérotés et $X = \mathcal{V} \setminus (Y \cup Z)$. L'ordre topologique total est garanti par un parcours en commençant par la source et il est total car Y est modélisé par une file (FIFO)

Pour chacun des noeud il va être appelé Succ, et des q'il y a un arc il est appelé Precc. $\mathcal{C}(c, n, m) = (n + m) \times c$

L'utilisation d'une pile induit un parcours en **profondeur** alors que celui d'une file se fait en **largeur**.

4 Ordonnancement parallèle

La borne inférieure du temps total d'exécution est $t_{min} = \lceil \frac{\sum_{i=1}^n}{r} \rceil$. Si cette borne est atteinte alors l'utilisation des ressources est maximale.

La contrainte de ressources limitées sur les listes se traduit par un nombre maximal de r tâches en parallèle.

4.1 Amélioration de l'ordonnancement avec heuristique de priorité

cf. cour

5 Extension aux tâches et ressources hétérogènes

La prise en compte de l'hétérogénéité dans problème conduit à un temps d'exécution réel de $\alpha \times t$