Les algorithmes Minimax et Minimax(α,β)
-
Application au jeu de Dames

Nicolas Bernard
n.bernard@lafraze.net

18 juin 2001

Le programme de dames a été réalisé avec Objective Caml 3.0 et Emacs 20.7 sur un système Linux (kernel 2.4).
Le présent rapport a été réalisé avec LaTeX 2ε.
Les arbres ont été dessinés grâce à l'extension PSTricks.

Table des matières

Introduction

L'intelligence artificielle et le problème du jeu de l'ordinateur

Dans les années quarante, les premiers ordinateurs sont apparus, dont le plus célèbre est sans doute le premier, l'ENIAC. Il s'agissait surtout de calculateurs fonctionnant en logique cablée, des centaines de cables devant être débranchés puis rebranchés lorsque l'on voulait traiter un problème différent. Puis avec l'apparition de ce que von Neumann appelle le "contrôle en mémoire" ([6] [1]), sont vraiment apparues des machines méritant le nom d'ordinateurs). Or le désir de l'homme de créer une machine à son image est antérieure à l'essor de la science-fiction dans les années soixante-dix: que les sceptiques regardent *Métropolis* de Fritz Lang! L'incroyable puissance de calcul (pour l'époque) de ces ordinateurs a suggéré à certains que l'un des problèmes qui semblaient jusque là insurmontables, celui de créer une intelligence comparable à celle de l'homme, pouvait peut-être être surmonté! C'est à cette époque qu'Alan Turing a publié son article désormais célèbre "Les ordinateurs et l'intelligence" dans la revue *Mind* ([7]) et que, dans la foulée, s'est créée une discipline nouvelle: l'intelligence artificielle, que l'on abrège fréquemment en IA ou, en anglais, AI.

Il se trouve que l'un des problèmes semblant mettre en oeuvre l'intelligence sans être trop complexe à formaliser est celui des jeux, en particulier les échecs et les dames, c'est sans doute pourquoi le jeu de dames fut l'un des premiers sujets étudiés par l'IA et notamment dès 1947 par Arthur Samuel. Le but originel de celui-ci était de faire de la publicité afin de récolter de l'argent pour sa discipline. Pour ce faire, il avait décidé de programmer un jeu de dames qui devait remporter le championnat organisé quelques mois plus tard dans une ville voisine. En 1983, il devait écrire "A l'époque, il semblait parfaitement raisonnable de croire que nous allions assembler un ordinateur en quelques mois, et que j'allais rédiger un programme de dames [...] qui

1. voir la bibliographie en fin de document.

défierait et finalement battrait le nouveau champion du monde. Cela nous aurait fourni la publicité dont nous avions besoin. Où s'arrête la naïveté?[2]" Il lui fallut finalement quinze ans pour réaliser un programme de niveau convenable, créant au passage des techniques toujours utilisées aujourd'hui, par exemple par l'ordinateur Deep Blue d'IBM, et mettant en oeuvre un algorithme implémenté dans la quasi totalité des jeux actuels, le **Minimax**[3].

Le jeu de Dames

Si la plupart des gens savent jouer au dames, il faut néanmoins préciser qu'il en existe de multiples règles. Le jeu le plus couramment pratiqué en France est le jeu de dames polonaises, qui se déroule sur un damier de cent cases avec vingt pions par joueur posés sur les cases noires des quatre lignes les plus proches du joueur. Les noirs commencent la partie. Un pion se déplace en diagonale d'une case à la fois. Un pion prend une pièce adverse située à côté (en diagonale!) de lui en passant par dessus à condition qu'une case soit libre derrière et, de là, il peut éventuellement en prendre un ou plusieurs autres dans le même tour. Une prise multiple ainsi réalisée s'appelle une *rafle*. Si un joueur peut prendre un pion, il doit le faire; cependant dans le cas contraire, le joueur adverse peut, à son tour, *souffler* la pièce qui aurait dû prendre puis jouer normalement: c'est la règle du "souffler n'est pas jouer"[4]. Le joueur n'est pas obligé de choisir le coup qui prendra le plus de pièces mais, lorsqu'il a commencé à bouger l'une d'elles, il doit prendre toutes celles qui se présentent. Lorqu'un pion d'une couleur arrive à la ligne de départ des pions adverses, il y a "Dame". Il est alors chevauché par un autre pion de la même couleur. Une dame peut se déplacer dans les deux sens de plusieurs cases à la fois, toujours en diagonale.

Les autres méthodes

Si, comme nous l'avons dit, l'algorithme minimax (et surtout sa variante plus raffinée, le minimax(α,β) (voir plus loin)) est utilisé dans l'immense majorité de ce type de jeux (Dames, Echecs, Backgammon, Go, ...) pour faire jouer l'ordinateur, il existe cependant d'autres méthodes. Ainsi David Fogel

2. Cité dans [4].

3. Le lecteur intéressé pourra consulter [5] pour les principes fondamentaux ainsi que [4] pour l'histoire de l'IA

4. Dans la pratique, cette règle complique de façon significative le programme informatique et son utilisation augmente le nombre de coups possibles diminuant les possibilités d'exploration de l'arbre de jeu et par là le niveau du programme.

et Kumar Chellapilla ont créé un programme de *checkers*[5] avec un réseau neuronal qu'ils ont fait évoluer en utilisant des méthodes darwiniennes. Le programme de la $230^{\text{ième}}$ génération s'est montré capable de battre un humain chevronné, se classant juste derrière les maîtres et les experts ([1]). De telles méthodes sont cependant encore rares...

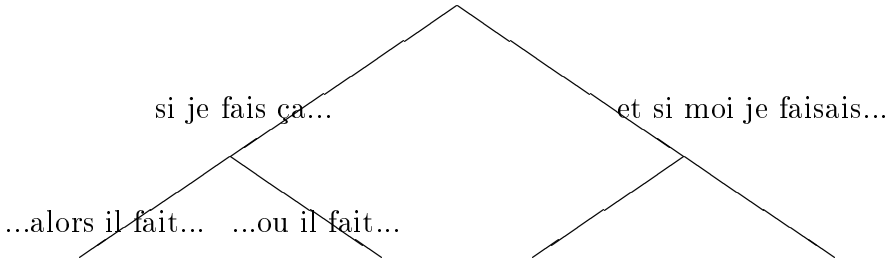
5. Il s'agit de la version américaine des dames: elle se joue sur un damier de soixante-quatre cases avec douze pions par camp.

Chapitre 1

L'algorithme Minimax

Les origines de l'algorithme datent des années 1940. La théorie sous-jacente est la Théorie des Jeux, développée par John von Neumann et Emile Borel vers 1928, dans laquelle on trouve le Théorème du Minimax, et l'algorithme aurait été, si l'on croit [4], inventé en 1947 par Claude Shannon (c'est-à-dire un an avant sa théorie de l'information...).

Cet algorithme est basé sur la possibilité de représenter les suites de coups (les stratégies) de jeu par un arbre: c'est ce que l'on fait quand on joue "si je fais ça, il peut faire ça et je pourrais alors faire... mais s'il fait ...et si moi je faisais..."



Et l'on pourrait poursuivre cet arbre, étudiant tous les coups possibles, jusqu'à la fin du jeu. Le problème, et il est de taille, c'est que, partant du début du jeu jusqu'à sa fin, l'arbre est tellement grand qu'il est impossible de l'explorer entièrement: en imaginant que l'ordinateur le plus puissant que nous puissions construire ait commencé cette exploration au big bang[1], il

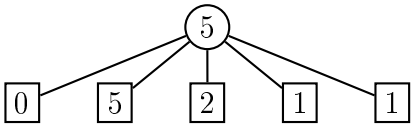
1. Si, comme Fred Hoyle, vous doutez de la théorie du big bang, vous pouvez lire "il y a une quinzaine de milliards d'années"!

n'aurait pas encore terminé son exploration aujourd'hui!!! Et il faut noter qu'il faut aussi construire l'arbre, ce qui peut se faire au fur et à mesure de l'exploration ou préalablement, mais prend dans tous les cas du temps et de la place...

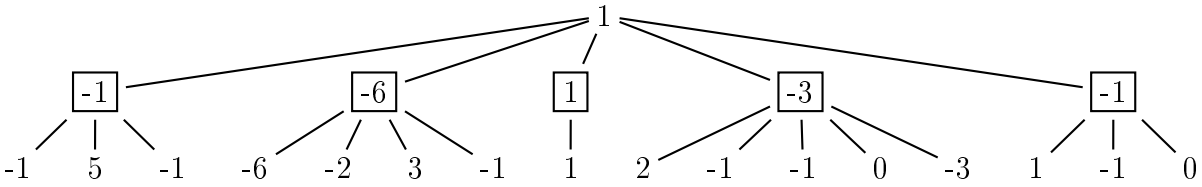
C'est pourquoi l'exploration de l'arbre ne se fait pas en totalité: dans la pratique, on tronque l'arbre, c'est-à-dire que l'on se contente de l'explorer sur une profondeur de quelques coups, le temps mis pour cette exploration augmentant exponentiellement avec le nombre de coups...

Cet algorithme est donc une exploration en profondeur de l'arbre du jeu qui nécessite une fonction qui nous permette de connaître tous les coups possibles (en respectant les règles du jeu bien sûr) à partir d'une position donnée ainsi qu'une fonction qui puisse évaluer une configuration donnée et lui attribuer un score. Le principe de l'algorithme est donc, à partir d'une certaine position du jeu, de construire l'arbre jusqu'à une certaine profondeur puis d'appliquer la fonction d'évaluation aux feuilles (alors qu'il faudrait appliquer la fonction de construction des coups possibles si nous voulions "faire pousser les branches" c'est-à-dire construire l'arbre avec une profondeur d'un coup supplémentaire).

Prenons un exemple très simple: on explore l'arbre avec une profondeur de 1: cela correspond à jouer le coup qui offre le plus de possibilités immédiates (le meilleur score) sans prendre en compte la riposte de l'adversaire. Comme le but dans ce cas est de maximiser notre score, on appelle cela une *étape maximisante*; on peut dire que l'on fait remonter le score maximum.



Cependant, il est bien sûr inutile d'espérer aller très loin avec une telle stratégie (que l'on pourrait qualifier de myope, une stratégie aveugle étant de jouer aléatoirement) dans un jeu comme les dames ou les échecs : en effet, la riposte de l'adversaire pourrait faire perdre tout le bénéfice du coup, alors qu'un autre, moins bon à première vue, aurait permis de limiter les dégâts, comme dans cette extension à une profondeur de 2 de notre arbre précédent, ou l'étape rajoutée est une étape minimisante, car on fait cette fois remonter le minimum "dans" les noeuds intermédiaires avant de choisir, comme avant, le maximum des scores de ces noeuds (donc en fait le maximum des minima).



Bien entendu, pour pouvoir réellement jouer correctement à un jeu tel que le jeu de dames, une exploration à une profondeur de 2 se révèle encore insuffisante d'un point de vue tactique (notons au passage que le minimax est un algorithme de *tactique*. Ce serait un algorithme de stratégie à condition d'explorer une très grande partie, voire la totalité, de l'arbre de jeu) et il faut considérer une profondeur minimale de trois ou quatre comme un minimum. Le problème c'est que, si l'on suppose que l'on peut jouer 30 coups en moyenne (ce qui est tout à fait possible si l'on considère toutes les possibilités de "souffler n'est pas jouer" comme le fait notre programme), c'est qu'à une profondeur de 6 (ce qui correspond à prévoir 3 coups à l'avance), on a alors $30^6 = 729\,000\,000$ cas à évaluer... ce qui est difficile à faire dans un temps suffisamment court pour que le joueur humain devant son écran ne s'impatiente pas trop...

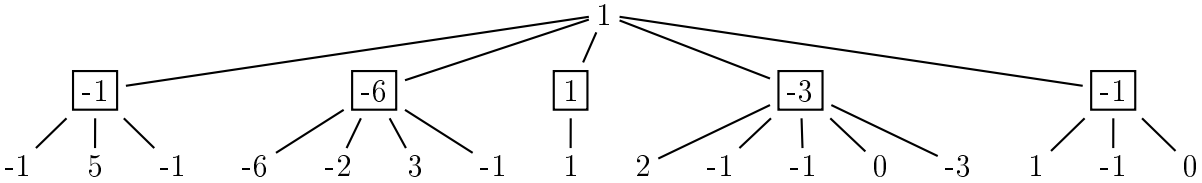
Chapitre 2

L'algorithme Minimax avec coupures α et β

L'algorithme minimax avec coupures alpha et bêta, ou plus simplement, le minimax(α,β), parfois même appelé "algorithme alpha-bêta", est dérivé du précédent. L'idée est "d'élaguer" l'arbre de recherche de façon à diminuer le nombre de possibilités à étudier.

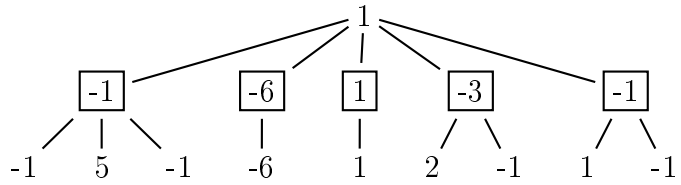
Les origines de cet algorithme sont floues: il semble qu'il ait été connu dans les années 1950 où l'élagage des arbres de jeu fait l'objet de nombreux articles de Samuel, Shaw, Simon, McCarthy, Hart ou Edwars, mais, à l'époque, souvent confondu avec le minimax ordinaire. Une recherche effectuée sur internet associe à sa naissance le nom d'Alan Turing ou parfois de Claude Shannon... Si ces deux pères de l'informatique se sont effectivement penchés sur la question (Shannon aurait, on l'a vu, créé le minimax et Turing a inventé le concept de "position morte") ce n'est, si l'on en croit [2] qu'en 1969 qu'il aurait été décrit précisément pour la première fois dans l'article *Experiments with some programs that search games trees* de Slagle et Dixon.

Son principe découle d'une remarque que l'on peut faire sur l'arbre que l'on a donné en exemple pour le minimax:



Si l'on regarde cet arbre, on s'aperçoit qu'il n'est pas utile de l'explorer entièrement: supposons que l'exploration se fasse en profondeur de gauche à droite: dans la branche principale la plus à gauche remonte le score -1,

par conséquent, comme dans la premeière feuille de la seconde branche on trouve le score -6, il n'est pas la peine de continuer à explorer cette branche, puisque, quel que soit son score, il sera de toute façon inférieur à celui de la première. Cela permet d'élaguer l'arbre en supprimant toutes les autres feuilles de la deuxième branche! On peut répéter le raisonnement pour les autres branches, et l'arbre élagué de cette manière est finalement:



On a réalisé sur cet arbre une *coupure* α. D'une manière plus mathématique, la coupure α est la borne inférieure d'un noeud maximisant. De la même manière, on peut définir la *coupure* β comme la borne supérieure d'un noeud minimisant, ce qui est utile pour les étapes maximisantes...

Dans notre exemple (notez bien, vous pourriez penser que les valeurs des feuilles ont été choisies pour que ça marche, mais ce n'est pas le cas: elles ont été choisies au hasard), sur les seize feuilles originelles, on a ainsi pu en supprimer sept, ce qui n'est pas mal, mais dès que l'arbre est plus grand, ce sont des branches entières qui sont tronçonnées, le nombre de coupures augmentant.

C'est cet algorithme qui est utilisé dans la plupart des jeux d'échecs sur ordinateur et que nous avons implémenté (en langage Caml) dans notre jeu de dames. Voici son pseudo-code (réalisé à partir de deux codes en C trouvés sur les sites du MIT et de CalTech):

$état \leftarrow$ (état actuel du jeu)
$noeudmax \leftarrow$ faux
$reste\_à\_voir \leftarrow$ (profondeur de l'arbre à explorer)
$alpha \leftarrow -\infty$ [1]
$bêta \leftarrow +\infty$ [2]

Si $(reste\_à\_voir = 0) \vee$ plus_de_coup_possible
Alors évaluer $état$
Si $noeudmax \wedge (score > alpha)$
Alors $score \leftarrow alpha$
Si $(\neg noeudmax) \wedge (score < bêta)$
Alors $score \leftarrow bêta$

1. dans la pratique, nous avons utilisé *min_int*.
2. dans la pratique, nous avons utilisé *max_int*.

Sinon pour chaque coup possible
    Générer le coup
    mettre à jour *état*
    Appeler *minimax* récursivement avec $\neg noeudmax$ et $reste\_à\_voir - 1$
    Si $noeudmax \wedge (score\_retourné > alpha)$
        Alors $alpha \leftarrow score\_retourné$
    Si $(\neg noeudmax) \wedge (score\_retourné < bêta)$
        Alors $bêta \leftarrow score\_retourné$
    Si $alpha \geq bêta$
        Alors retourner *score* (et sauter le reste de la boucle et de la fonction[3])

Si *noeudmax*
    Alors retourner *alpha*
    Sinon retourner *bêta*

Remarquons maintenant que si l'exploration de l'arbre avait commencé par la branche du milieu (sur notre dessin), alors on aurait également pu enlever des feuilles à la branche de gauche. Il n'y a pas de méthode "stricte" pour savoir par quelle branche commencer l'exploration, mais on pourrait encore optimiser l'algorithme de façon à ce qu'il "devine" quelle branche il est le plus intéressant d'explorer en premier... C'est le domaine des méthodes heuristiques.

3. en Caml, nous avons utilisé une exception.

Conclusion

Si l'algorithme minimax peut convenir pour un jeu comme le morpion, où l'arbre du jeu est suffisamment petit pour être exploré entièrement, l'explosion combinatoire le rend inutilisable pour un jeu plus complexe tel que les dames ou les échecs. On peut alors utiliser la variante avec coupures alpha et bêta afin d'élaguer l'arbre. Toutefois, si cela suffit à assurer à l'ordinateur un bon niveau dans un jeu où le nombre de coups possibles à chaque étape reste modéré, tels les jeux de *checkers* ou d'othello, et si son niveau reste supérieur à celui d'un joueur occasionnel dans une variante plus complexe comme dans les dames polonaises en tenant compte de toutes les possibilitées de "souffler n'est pas jouer" que nous avons choisi d'implémenter, cet algorithme est encore, en l'état, insuffisant (compte tenu de la rapidité des machines actuelles) face à un joueur expérimenté. Il est alors nécessaire de "l'aider". Pour ce faire, les programmes que l'on peut trouver dans le commerce (ceux d'échecs notamment; les programmes de dames commercialisés sont rares) utilisent des méthodes heuristiques combinées à une base de donnée de parties types.

Quoi qu'il en soit, ce n'est probablement pas ce type d'algorithme qui est utilisé par le cerveau humain pour jouer, ce qui rend intéressant des développements tels que ceux basés sur les réseaux de neurones auxquels nous avons fait allusion dans l'introduction. Néanmoins, une question se pose alors: bien que l'on puisse le reproduire, bien que les règles lui aient été données au départ, peut-on réellement dire que l'on comprend la façon de procéder d'un réseau neuronal ayant subi un processus évolutionniste?

Une chose reste sûre: sur le chemin de la vraie "intelligence articielle", malgré tout le chemin parcouru au cours des cinquante dernières années, comme l'écrivait déjà Alan Turing pour conclure son article de 1950: "Notre vision de l'avenir est limitée, mais du moins nous voyons qu'il reste bien des choses à faire".

Annexe A

Le CD-Rom et notre programme

A.1 Contenu du CD-Rom

Vous trouverez sur le CD-Rom joint la version 3.0 d'Objective Caml pour Linux et Windows. Pour d'autres plateformes, veuillez vous reporter au site officiel de Caml: **http://caml.inria.fr/**. Le cd-rom contient également le code-source de notre programme (version 0.99) ainsi que la version exécutable pour Linux. Vous trouverez également le présent document au format postscript.

A.2 Notre programme de dames

A.2.1 Configuration requise

- Système d'exploitation: Unix avec Xwindow, Windows (MacOS et tout autre système sur lequel fonctionne Caml devraient également convenir). Le programme a été testé avec Linux (kernel 2.2 et 2.4) et Xfree 4.0, Windows 9x. Nous recommandons Unix pour une meilleure esthétique.
- Hardware: Ce programme a été testé et fonctionne correctement sur plusieurs PC, munis de processeur Pentium, K6-2, Pentium 2 et Pentium 3.

A.2.2 Installation et lancement

Pour utiliser la version interprétée du jeu, vous devez avoir une distribution d'Objective Caml installée sur votre machine. Consultez la documentation fournie avec la distribution pour l'installer. Le code-source de différentes

versions d'Objective Caml, ainsi que les exécutables correspondants pour la plupart des plateformes sont disponibles à l'adresse *http://caml.inria.fr*.
Si vous avez une plateforme sous Unix, vous devez créer un *toplevel* incluant les bibliothèques *Graphics* et *Unix* en utilisant la commande

```
ocamlmktop -custom -o montoplevel graphics.cma unix.cma -cclib -lX11
```

ou si votre système place les bibliothèques X11 dans un autre répertoire, comme Linux,

```
ocamlmktop -custom -o montoplevel graphics.cma unix.cma -cclib \
           -L/usr/X11/lib -cclib -lX11
```

Vous devez alors lancer l'interpréteur puis copier-coller le fichier source dans sa fenêtre ou utiliser la commande

```
#use "chemin_et_nom_du_fichier" ;;
```

pour lancer l'interprétation.

La plupart des versions compilées nécessitent simplement d'avoir Objective Caml installé sur le système. Pour un fonctionnement correct, le programme doit pouvoir écrire dans son répertoire.

A.2.3 Utilisation

Lors du lancement apparaît une page de présentation comprenant en bas une série de boutons.

Le premier permet de charger une partie sauvegardée. Vous êtes alors renvoyé à une autre page qui permet de choisir l'un des dix emplacements de sauvegarde. Notez que le choix d'un emplacement ne comprenant pas de sauvegarde provoquera l'arrêt du programme.

Le second bouton permet de jouer à deux sur l'ordinateur.

Le troisième permet de jouer contre l'ordinateur. Vous devez alors choisir votre couleur.

Le dernier bouton permet de quitter le jeu sans jouer.

Comment jouer?

Quand c'est votre tour de jouer, vous devez cliquer sur le pion que vous désirez bouger puis successivement sur les cases sur lesquelles il devra se poser. En cas d'erreur, cliquez sur *Annuler* et rejouez. Quand vous voulez exécuter ce coup, cliquez sur *Fin de Tour*. Si vous voulez souffler un pion, cliquez dessus puis cliquez sur *Souffler*. Notez que si vous désirez souffler, vous devez le faire avant de jouer. Vous pouvez quitter le programme avec le bouton *Abandonner/Quitter*; pensez éventuellement à sauvegarder la partie avant.

Sauvegarder une partie

Vous pouvez sauvegarder une partie lorsque c'est à vous de jouer en cliquant sur *Sauver* puis en sélectionnant un emplacement et en confirmant.

Attention ! Il est possible mais déconseillé de charger des sauvegarde faites avec le programme sous Unix dans la version Windows et *vice-versa*.

Bibliographie

[1] Philip Ball. Quand Darwin joue aux dames avec des ordinateurs neuronaux. *Le Monde*, 14/01/2000.

[2] Frédéric Bayard. Complexité de l'algorithme alpha-béta. Master's thesis, Ecole Normale Supérieure de Cachan, 1996-1997.

[3] Emmanuel Chailloux, Pascal Manoury, and Bruno Pagano. *Développement d'applications avec Objective Caml*. O'Reilly, 2000.

[4] Daniel Crevier. *A la recherche de l'intelligence artificielle*. Champs. Flammarion, 1997.

[5] Jean-Paul Haton and Marie-Christine Haton. *L'intelligence artificielle*. Que sais-je? Presses Universitaires de France, troisième edition, 1993.

[6] John von Neumann. *L'ordinateur et le cerveau*. Flammarion, 1996.

[7] Alan Turing. Computing machinery and intelligence. *Mind*, 1950. Publié en français dans [8], sour le titre Les ordinateurs et l'intelligence.

[8] Alan Turing and Jean-Yves Girard. *La machine de Turing*. Points Sciences. Seuil, 1995.