Projet IAMR 2006:
Conception et réalisation d'un jeu de Puissance 4 avec gestion d'une Intelligence Artificielle

Juliet Eichen, Charles Letaillieur, Patrick Sterlin

Le problème traité consiste à gérer une partie de Puissance 4. Dans ce jeu, chaque joueur place alternativement un pion dans un plateau vertical soumis à la gravité (on ne peut poser un pion qu'en bas du plateau ou directement sur un autre pion). Le but du jeu est d'aligner 4 pions de la même couleur horizontalement, verticalement, ou en diagonale. La programmation a été réalisée en LUSH (Lisp graphique)

1. Conception et réalisation de l'environnement du jeu

Pour une portabilité maximale, nous avons choisi de n'utiliser que les fonctions de base de Lush, et en particulier pas les librairies graphiques telles que Ogre ou SDL qui posent des problèmes sur certaines plateformes. Les divers paramètres graphiques (taille du plateau, couleur des pions, décalages, couleur des menus …) sont stockés dans des variables pour être facilement changeables. La principale difficulté de conception consiste en l'élaboration d'un système de gestion des évènements (clicks souris), la seule fonction fournie par LUSH pour cela étant une fonction « get-click », bloquante, qui retourne les coordonnées du click. Il n'est donc pas évident de réaliser des éléments cliquables (boutons, plateau de jeu) disponibles à tout instant et dans n'importe quel ordre.

Il faut pour cela simuler un traitement multi-threads. Ceci a été réalisé grâce à la fonction « clic », qui est appelée chaque fois que le programme ne calcule pas. Elle est la seule à faire appel à « get-click », et connaît la position de tous les boutons, ainsi que du plateau de jeu. Selon le mode de jeu (1 : humain contre humain, 2 : humain contre ordinateur, 3 : ordinateur contre ordinateur), un clic sur le plateau de jeu ne provoque pas la même action (dans les modes 1 et 2, il permet de placer un pion dans la colonnes sélectionnée), alors que dans le mode 3, il sert à faire avancer d'un tour. Ces actions sont gérées par des appels aux fonctions « attencClic1 » et « attencClic3 ».

Quand les menus s'affichent, on n'appelle plus la fonction « clic », mais une fonction spécifique : « clicMenuJoueurs » pour le choix du mode de jeu (« MenuJoueurs »), « clicMenuNiveau » pour le choix du niveau de l'IA (« MenuNiveau »), et « clicMenuCommence » pour le choix du premier joueur (« MenuCommence »). Chacun des trois modes de jeu est géré par une fonction (jouer1, jouer2 et jouer3). Les deux dernières font appel à la fonction « jouerIA » qui permet à l'IA de jouer un tour.

2. Conception de l'Intelligence Artificielle (IA) du jeu

2.1 Implémentation de l'algorithme min/max

L'algorithme de jeu de l'IA repose sur l'algorithme d'élagage min/max.
Un tour de jeu de l'IA se déroule comme suit :

Tout d'abord, si l'IA est la première à jouer, elle se place au milieu du plateau.
Sinon, elle lance l'algorithme min/max :

L'algorithme min/max fonctionne suivant le principe suivant : on commence par construire un arbre des possibilités à partir de la situation courante. Chaque branche représente un coup (l'ordre des branches correspond à l'ordre des colonnes où l'on joue). L'idéal serait de construire l'arbre jusqu'à ce que la grille soit complètement remplie mais pour des raisons de taille mémoire et de temps de calcul, on limite la profondeur, ce qui donne une « vision » de quelques coups d'avance à l'IA. Cette profondeur est directement liée au niveau de l'IA choisie par l'utilisateur au début de chaque partie (novice, moyen, expert).

Ensuite, on évalue toutes les feuilles de l'arbre en fonction du nombre et de la taille des alignements de pions de l'ordinateur et de son adversaire.

La fonction qui réalise cette construction, par récursivité, est appelée « Possibilite »

Puis, on évalue tous les noeuds en remontant vers la racine : si la branche correspond à un tour où c'est à l'ordinateur de jouer, on choisit pour un père le maximum des évaluations de ses fils, et si la branche correspond à un tour où c'est à son adversaire, on choisit le minimum (on suppose ainsi que l'adversaire utilise la même stratégie que l'IA). La fonction d'évaluation exacte choisie permet de rendre compte, à la fois de la situation de l'ordinateur et de son adversaire. Il s'agit de :

- Si l'ordinateur gagne : **Coeff4**
- Si l'ordinateur perd : **- Coeff4**
- Sinon :

  (nbre d'alignements de 3 pions du joueur
  - nbre d'alignements de 3 pions de l'adversaire)*coeff3
  + (nbre d'alignements de 2 pions du joueur
  - nbre d'alignements de 2 pions de l'adversaire)*coeff2

Coeff4 = 10 000
Coeff3 = 1 000
Coeff2 = 100

De cette façon, toute situation gagnante sera automatiquement choisie, et toute fonction « suicidaire » sera éliminée.
Seules les alignements libres (qui laissent la possibilités d'être complétés) sont comptabilisés.

Nous avons ensuite optimisé cette construction de l'arbre en interrompant la recherche des possibilités d'alignement dans les 3 cas suivants :

- L'ordinateur gagne
- L'ordinateur perd
- L'évaluation donne 0. On évite ainsi de s'acharner sur des solutions qui ne font pas évoluer les choses (quel que soit le joueur qui l'a joué). Comme on ne considère que les alignements libres, cette condition peut-être vérifiée aussi bien au début qu'à la fin de la partie.

2.2 Calcul du nombre d'alignements libres

Fonction d'évaluation

Une partie du travail réside dans l'évaluation de la grille.
En effet, à chaque nouveau coup, il faut pouvoir examiner la situation, dans le cadre de :

-le jeu homme/homme : repérer la fin d'une partie, ie quand la pose d'un pion, permet l'alignement de 4 pions d'une même couleur. C'est la fonction Vérifier qui s'en charge, en interaction avec les fonctions graphiques.
-le jeu machine/machine : là encore, il faudra évidemment pouvoir repérer l'alignement de 4 pions d'une même couleur (fonction vérifier), mais il faudra aussi pouvoir évaluer la pertinence de telle ou telle situation pour le joueur machine, de façon à l'aider à faire ses choix.
A partir du choix d'une fonction d'évaluation que nous espérons efficace, il nous a donc fallu repérer les alignements de 2, 3 et 4 pions de nous comme de l'adversaire. Une situation est d'autant meilleure que le nombre de nos combinaisons est grand, et que le nombre de combinaisons de l'adversaire est petit. La fonction Evaluer compte ainsi, dans le cadre de l'algorithme d'élagage expliqué ci-dessus, le nombre de combinaisons de 2, de 3 et de 4 dans certaines situations.

3. Réalisation de l'Intelligence Artificielle du jeu

3.1 Fonctions générales

L'état du plateau de jeu est stocké dans une liste de listes « pions », qui contient des 0, 1 ou 2 selon qu'il n'y a pas de pion (0), un pion rouge (1) ou un pion bleu (2) dans la case correspondante. L'accès et la modification se fait facilement grâce aux fonctions « nieme » et « setnieme » pour les listes simples, et « getij » et « setij » pour les listes de listes. La liste « colonnes » contient le nombre de pions placé dans chaque colonne. La fonction « placeJeton » place graphiquement un jeton dans une colonne et met à jour le tableau « pions ».

3.2 Fonction « Possibilité »

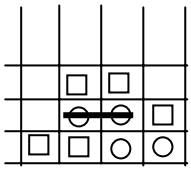
Pour i de 1 à longueur
- mettre un pion ami dans la colonne i (si elle n'est pas pleine)
  - Si l'ordinateur gagne, mettre coeff 4 dans l'arbre
  - sinon, pour j de 1 à longueur
    - mettre un pion adverse dans la colonne j (si elle n'est pas pleine)
    - Si on a atteint la profondeur cherchée
      - mettre l'évaluation de la situation dans l'arbre
      - enlever le pion dans la colonne j
    - Sinon, on applique cet algorithme récursivement.
- enlever le pion ami dans la colonne i

Pour la profondeur, on utilise une variable globale. On l'incrémente quand on met un pion. On la décrémente quand on enlève le pion ou qu'on s'arrête dans la construction.

3.3 Evaluation

Voilà comment se passe l'évaluation : on parcourt les colonnes, pour chacune, on regarde le pion le plus haut. Si ce pion est de la couleur de joueur, on regarde s'il forme une combinaison de 3 avec les pions voisins (7 directions).
On fait ensuite la même chose avec les combinaisons de 2, en prenant soin de ne pas compter celles qui participent déjà à une combinaison de 3.
La fonction évaluer renvoie une liste de 2 chiffres. Le premier est le nombre de combinaisons de 3, le deuxième celui de 2.
On prend évidemment soin de ne pas compter plusieurs fois les mêmes combinaisons.

C'est comme cela que se passaient nos évaluations, quand nous nous sommes rendu compte que cela ne prenait pas en compte les combinaisons dont aucun pion n'était à la surface.
Nous avons donc rajouté deux boucles, une pour les combinaisons de 3 et une pour celles de 2. On parcourt les colonnes, et quand 2 colonnes successives ont une différence de hauteur supérieure à 2, on observe les combinaisons que forme le pion de la cote.

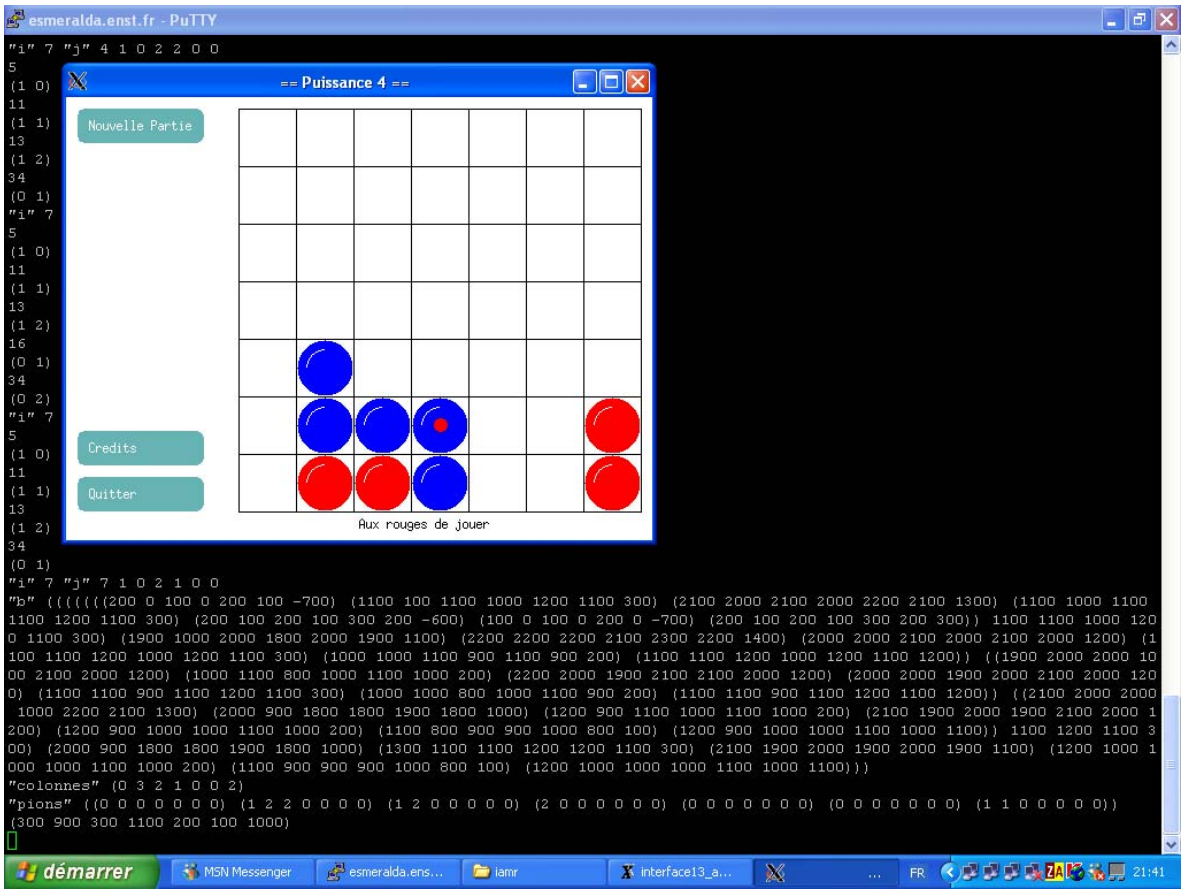


Couple que l'on peut remarquer en parcourant les côtes

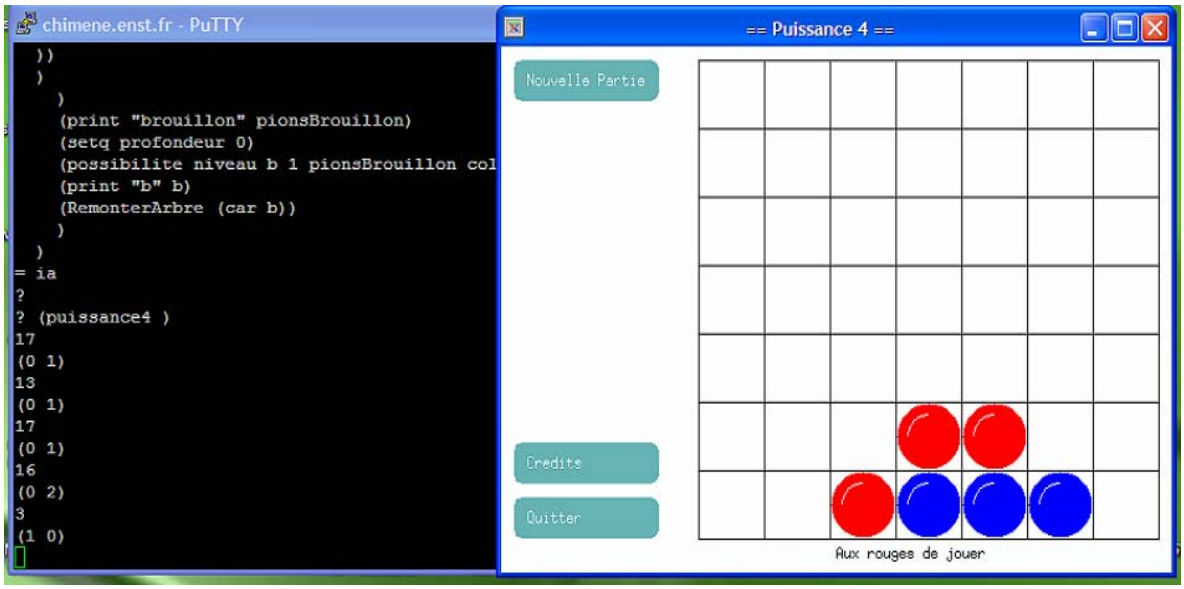
Nous avons aussi fait attention à ne pas prendre en compte les combinaisons n'offrant aucune perspective d'expansion : si la combinaison est encadrée de part et d'autre par un pion adverse ou par le mur.

4. Tests

4.1 Construction de l'arbre



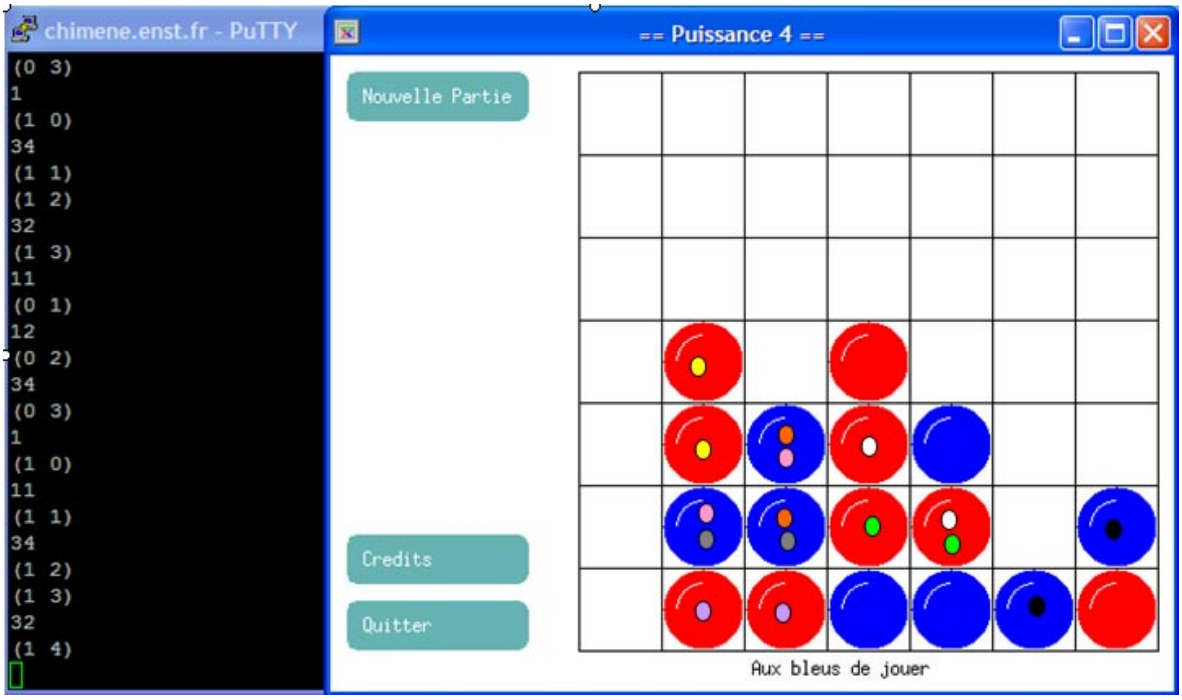
4.2 Evaluation



Les bleus viennent de jouer.

Ce qu'on lit dans la fenêtre : dernière ligne : (1 0) c'est ce que renvoie la fonction d'évaluation : une combinaison de 3 pions alignés et aucune de 2 ne participant pas déjà à celle de 3.

Avant-dernière ligne : évaluation des rouges avant que les bleus jouent. Aucune combinaisons de 3, et 2 couples (un en diagonale et un horizontal)



Le joueur rouge vient de jouer.

La fonction renvoie (1 4) : 1 combinaisons de 3 pions bleus alignés (verticale) et 4 de 2 (couples à tâches blanches,, vertes, violette et jaunes). On remarque que le couple vert est enfoui (aucun des 2 pions à la surface).

Pour les bleus, la fonction renvoie (0 3) : aucune combinaison de 3 : on remarque qu'il n'a pas pris en compte la combinaison horizontale en bas à droite, car celle-ci ne sera jamais exploitable pour créer une combinaison de 4. Il y a 3 combinaisons de 2 pions (les couples à tâches grises, orange et rose). On remarque que le couple à tâches noires (en bas à droite, en diagonal) n'est pas pris en compte, car il n'offre aucune perspective d'extension.

4.2 Commentaires généraux

L'IA est un peu lente à jouer mais les parties sont intéressantes. Elle reste battable même en niveau expert.

5. Problèmes rencontrés

5.1 Détermination de la fonction d'évaluation

Pendant longtemps, nous avons réfléchi à la meilleure façon d'évaluer une bonne ou mauvaise situation. Cela nous paraissait très compliqué. Nous voulions tout pendre en compte, de la même façon qu'un humain analyse une situation globale sans même s'en rendre compte. Finalement, nous avons trouvé dans la littérature des exemples assez simples de fonction d'évaluation qui étaient décrites comme « marchant bien » . Nous avons donc juste repéré nos alignements de 2 et 3 pions. Nous ne faisons alors intervenir que nos pions à nous, le but étant d'être dans la meilleure configuration possible.

Ce n'est que plus tard que nous avons eu une idée pour améliorer cette évaluation : en prenant aussi comptes de la configuration des pions de l'adversaire. Il faut maintenant à la

fois que notre situation soit la meilleure possible ET que celle de l'adversaire soit la moins bonne possible.

5.2 Complexité du programme

Dans le but d'éviter les redondances et de ne prendre en compte que les combinaisons exploitables (non entourées par l'adversaire ou le mur), nous devions multiplier les conditions sur la position des pions, pour ne jamais avoir à citer des cas externes à la grille (provoquant l'arrêt brutal du programme !!!)

Nous avons donc créé des variables locales qui représentent la grille mais à laquelle on a rajouté des colonnes et des lignes vides tout autour, pour ne pas butter contre les parois. Cela a permis d'alléger notablement le programme.

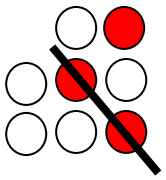
```
-1 –1 –1 –1 –1 –1 –1 –1 –1
-1  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0 -1
-1  0  0  0  0  0  0  0 -1
-1 –1 –1 –1 –1 –1 –1 –1 –1
```

Voici une liste de liste représentant la grille entourée d'un « coussin de protection »

6. Perspectives

On observe qu'en cas de situations équivalentes, l'IA joue toujours à gauche (ordre de parcours de l'arbre). Pour rendre les parties plus intéressantes, on pourrait introduire un facteur aléatoire.

Nous nous sommes rendu compte juste au moment de rendre notre programme d'un petit détail que nous n'avions pas pris en compte : certaines combinaisons ne sont pas détectées : quand aucun pion n'est à la surface, et aucun pion n'est sur la côte.



Cette combinaison de 2 pions n'est pas prise en compte !
Pourtant, elle est exploitable !