HEX - Documentation technique

Rédacteur : Vincent Dugat

10 février 2017

Il y a de nombreuses études, articles, thèses, algorithmes sur les jeux en général, et Hex en particulier. Ce documents présente quelques aspects utiles pour le projet.

1 Principe général de la résolution de jeux

Un jeu de plateau à deux joueurs peut se formaliser par un quintuplet $\{C, c_0, M, L, S\}$, où :

- C est l'ensemble des configurations possibles du plateau de jeu,
- $c_0 \in C$ est la configuration initiale (aucun pion posé au début de la partie par exemple),
- $M : C \to \wp(C)$ une fonction successeur qui associe à chaque configuration de jeu, l'ensemble des configurations accessibles en un coup
- $L \subset C$ un ensemble de configurations finales (celles où on ne peut plus jouer : un joueur gagne ou match nul). Formellement : $L = \{c \in C | M(c) = \emptyset\}$
- $S : L \to R$ une fonction d'évaluation qui attribue un score (nombre) à une configuration finale.

La partie est donc un ensemble $\{c_0, c_1, ..., c_m\}$ où $c_i \in M(c_{i-1}), 1 \leq i \leq m$ et $c_m \in L$

2 Outils formels pour le jeu de HEX

La configuration du plateau de jeu de HEX sera modélisée par un graphe.

Un **graphe** $G = (V, E)$ est une structure de donnée abstraite formée d'un ensemble V : les sommets (vertices), et d'un ensemble E : les arêtes (edges). Les arêtes relient deux sommets. Un graphe est donc un sous-ensemble du produit cartésien. Dans notre cas il y a un sommet par hexagone et deux sommets u et v sont reliés par une arête si les hexagones correspondants sont adjacents (se touchent) sur le plateau de jeu. On peut noter une arête entre u et v par (u, v).

Au graphe G représentant le plateau de jeu on ajoute quatre sommets : $w1$, $w2$, $b1$, $b2$, figurant les bords de chaque couleur.

Chaque sommet du graphe peut être coloré de trois couleurs : blanc, noir, ou transparent suivant que la case est libre, occupée par un pion blanc ou par un pion noir.

Nous avons besoin de quelques notions pour aller plus loin :

Une **chaine** est une suite de sommets $u_0, u_1, ..., u_k-$ tels que $\forall i \in [0, k-1], (u_i, u_{i+1}) \in E$.

Si $u_0 = w1$ et $u_k = w2$ on a une chaine gagnante pour les blancs, et bien sûr la même chose pour les noirs avec b_1 et b_2.

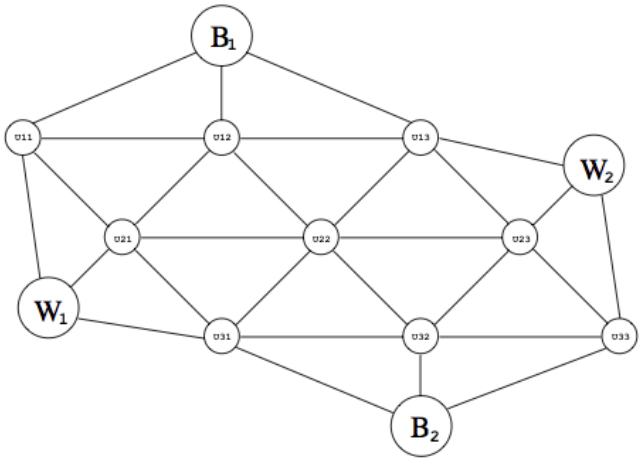

FIGURE 1 – Le graphe d'un plateau 3x3

La stratégie pour chaque joueur va être de faire des chaines pour aboutir à une chaine gagnante. En fait, on va construire des choses un peu plus compliquées que de simples chaines.

Dans la suite on pourra parler indistinctement de sommets ou d'hexagones, ces deux notions étant liées.

Un **groupe** R est un ensemble maximal d'hexagones (sommets) qui sont tous occupés par un pion du joueur P (noir ou blanc) et tel que pour toute paire de sommets u et v de cet ensemble, il existe une chaîne reliant u et v n'utilisant que des sommet de R.

Si on mémorise une liste de groupes par joueur, la fin du jeu arrive dès qu'un groupe gagnant existe dans la liste ($w1$ et $w2$ appartiennent à un groupe par exemple).

Un hexagone du plateau est **adjacent (touche) au groupe** s'il est adjacent à un quelconque hexagone du groupe.

Deux sommets sont dits **directement connectés** s'ils appartiennent au même groupe.

La **distance** d entre un hexagone p et un hexagone q est le nombre d'hexagones qui les séparent par le chemin le plus court. On la note $d(p, q)$.

Par extension, la distance entre un hexagone p et un groupe P est la plus courte distance entre p et les hexagones qui composent P.

Une **connexion forcée** entre deux sommets u et v de couleur P existe si dans toutes les séquences de coups où $\bar{P}$, l'adversaire de P commence, P pourra connecter directement u et v.

Une connexion forcée est associée à un ensemble de sommets A qui rassemble les sommets impliqués dans la connexion forcée. A est l'ensemble associé à la connexion forcée et u et v sont **F-connectés**.

Exemple : le pont (bridge)
Soit u et v deux sommets joués par P. Il y a un **pont** entre u et v si U, l'ensemble des sommets adjacents à u, et V, l'ensemble des sommets adjacents à v, ont une intersection non vide de plus de deux éléments.

FIGURE 2 – Un pont

$card(U \subset V) \geq 2$. Le cas le plus commun est quand $card(U \subset V) \geq 2$ (figure).

Note : $U \subset V$ est l'ensemble associé à la connexion forcée de u et v.

On se limitera ici aux connexions forcées correspondant à des ponts.

Un **graphe réduit** $\hat{G}$ d'un graphe de jeu G est le graphe obtenu en remplaçant les groupes de G par des sommets uniques. Ces sommets reçoivent la même couleur que le groupe qu'ils remplacent. Les sommets transparents ou les sommets noir et blancs isolés restent tels quels dans $\hat{G}$.

Il y a une arête entre un sommet u et un sommet-groupe de $\hat{G}$ si u était adjacent au groupe dans G. Le graphe obtenu est plus simple mais on perd l'adéquation entre le graphe et le plateau.

3 Numérotation des hexagones

Il existe différentes façons de s'y prendre. En voici une qui a le mérite de respecter la logique des tableaux du C :

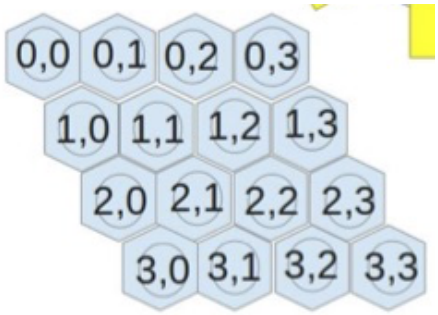

FIGURE 3 – Numérotation avec $N = 4$

4 L'algorithme du minimax

Le principe est de passer en revue toutes les pour un nombre exhaustif ou limité de coups et à leur assigner une valeur qui prend en compte les bénéfices pour le joueur et pour son adversaire. Le meilleur choix est alors celui qui minimise les pertes du joueur tout en supposant que l'adversaire cherche au contraire à les maximiser (le jeu est à somme nulle).

L'algorithme minimax est très simple : on visite l'arbre de jeu pour faire remonter à la racine une valeur (appelée « valeur du jeu ») qui est calculée récursivement de la façon suivante :

- minimax(p) = $f(p)$ si p est une feuille de l'arbre où f est une fonction d'évaluation de la position du jeu
- minimax(p) = MAX(minimax(O_1), minimax(O_2), ..., minimax(O_n)) si p est un nœud du joueur P (blanc par exemple) et où O_1, O_2,..., O_n sont les configurations suivantes de p
- minimax(p) = MIN(minimax(O_1), minimax(O_2), ..., minimax(O_n)) si p est un nœud du $\overline{(P)}$ et où O_1, O_2,..., O_n sont les configurations suivantes de p

On prendra $f(p) = 1$ pour une feuille où le joueur noir gagne et $f(p) = -1$ pour une feuille où le blanc gagne.

5 Travail à faire pour chaque version

1. **V1 (langage C) :**
   - Construire une SD dynamique pour implémenter le graphe représentant le plateau de jeu pour une taille de plateau N quelconque (la formule générale des sommets reliés à un sommet $u_{i,j}$ dans le graphe de jeu peut s'avérer utile).
   - Calculer les groupes et la détection d'un groupe gagnant.
   - Gérer dynamiquement la liste des groupes de chaque joueur.
   - programmer une SD pour le graphe réduit, s'en servir pour les calculs du jeu.
2. **V2 (langage C) :**
   - Construire une SD dynamique pour représenter un arbre de jeu gérant les coefficients minimax,
   - programmer le minimax exhaustif sur un plateau de taille N (implique : gestion des groupes, groupe gagnant),
   - le tester pour l'arbre de jeu d'un plateau 2×2.
   - Parcourir l'arbre pour trouver une stratégie gagnante.
   - essayer l'algo sur des N plus grands et commenter dans le document de bilan chiffres à l'appui.
3. **V3 (Java ou C) :**
   - Détecter les ponts.
   - Gérer les groupes de groupes séparés par des ponts
   - Mettre à jour ces SD à chaque tour de jeu pour chaque joueur
   - programmer un comportement de l'ordinateur cherchant à connecter ses groupes et couper ceux de l'adversaire.

6 Annexes

6.1 Interface Java / C

Il est possible d'appeler des fonctions C depuis Java. L'interface Java/C est décrite par une classe Java comportant une méthode statique par fonction C. Le mot clé native indique que la méthode est en fait implantée en C :

```
class InterfaceAvecC {
  ...
  public static native int m1(String s, int i);
  public static native String m2(String s);
}
```

L'en-tête du fichier C implantant les méthodes est automatiquement généré en utilisant la commande javah après compilation du source Java :

```
  javac InterfaceAvecC.java
  javah -jni InterfaceAvecC
```

Avec l'exemple précédent, le profil généré (dans un .h) est le suivant :

```
JNIEXPORT jint JNICALL Java_InterfaceAvecC_m1
  (JNIEnv *env, jclass cl, jstring js, jint ji);

JNIEXPORT jstring JNICALL Java_InterfaceAvecC_m2
  (JNIEnv *env, jclass cl, jstring js);
```

Les arguments de la méthode se retrouvent en position 3 et suivantes. Le fichier (.c) implantant les interfaces décrites dans le .h doit convertir les types Java non natifs en types C :

```
#include <stdlib.h>
#include <string.h>
#include "InterfaceAvecC.h"

JNIEXPORT jint JNICALL
Java_InterfaceAvecC_m1 (JNIEnv *env, jclass cl, jstring js, jint ji) {
  /* conversion de la chaîne Java en chaîne C */
  const char *s = (*env)->GetStringUTFChars(env, js, 0);
  return strlen(s) * ji;
}

JNIEXPORT jstring JNICALL
Java_InterfaceAvecC_m2(JNIEnv *env, jclass cl, jstring js) {
  /* conversion de la chaîne Java en chaîne C */
  const char *s = (*env)->GetStringUTFChars(env, js, 0);
  /* traitement en C */
  char *res = (char*) malloc(strlen(s) * 2 + 1);
  strcpy(res, s);
  strcat(res, s);
  /* conversion du résultat en chaîne Java */
  jstring jres = (*env)->NewStringUTF(env, res);
  /* libération mémoire */
  (*env)->ReleaseStringUTFChars(env, js, s);
  free(res);
  return jres;
}
```

Le fichier C doit ensuite être compilé en une libraririe partagée dont le nom doit commencer par `lib`. Il est nécessaire d'indiquer le chemin d'accès à l'installation de Java. Il faudra donc adapter les chemins indiqués ci-dessous à votre installation.

```
## Linux/gcc
gcc -I/usr/java/jdk1.5.0_01/include/
    -I/usr/java/jdk1.5.0_01/include/linux -c InterfaceAvecC.c
gcc -shared -o libInterfaceAvecC.so InterfaceAvecC.o

## Mac/gcc
gcc -I/Library/Java/Home/include/
    -I/Library/Java/Home/include/darwin -c InterfaceAvecC.c
gcc -shared -o libInterfaceAvecC.dylib InterfaceAvecC.o

## Windows/gcc
gcc -c -I"C:\j2sdk1.4.2_02\include"
       -I"C:\j2sdk1.4.2_02\include\win32"
       -o InterfaceAvecC.o TestJNI2.c
gcc -shared -o InterfaceAvecC.dll InterfaceAvecC.c
InterfaceAvecC.c InterfaceAvecC.def

## solaris
cc -G -I/opt/JAVA/jdk1.5.0_01/include -I/opt/JAVA/jdk1.5.0_01/include/solaris InterfaceAvecC.c
   -o libInterfaceAvecC.so
```

Cette librairie doit être chargée lors de l'exécution du programme Java. Il faut pour cela ajouter au code de la classe interface l'instruction de chargement :

```
class InterfaceAvecC {
  static {
    System.loadLibrary("InterfaceAvecC");
  }
  public static native int m1(String s, int i);
  public static native String m2(String s);
}
```

Les méthode écrites en C sont maintenant accessibles depuis n'importe quelle classe Java :

```
public class Test {
    public static void main(String args[]) {
        int i = InterfaceAvecC.m1("test",2);
        String s = InterfaceAvecC.m2("test");
        System.out.println("i="+i+", s="+s);
    }
}
```

Toutes les informations sont disponibles à l'adresse suivante :

https://www.jmdoudoux.fr/java/dej/chap-jni.htm

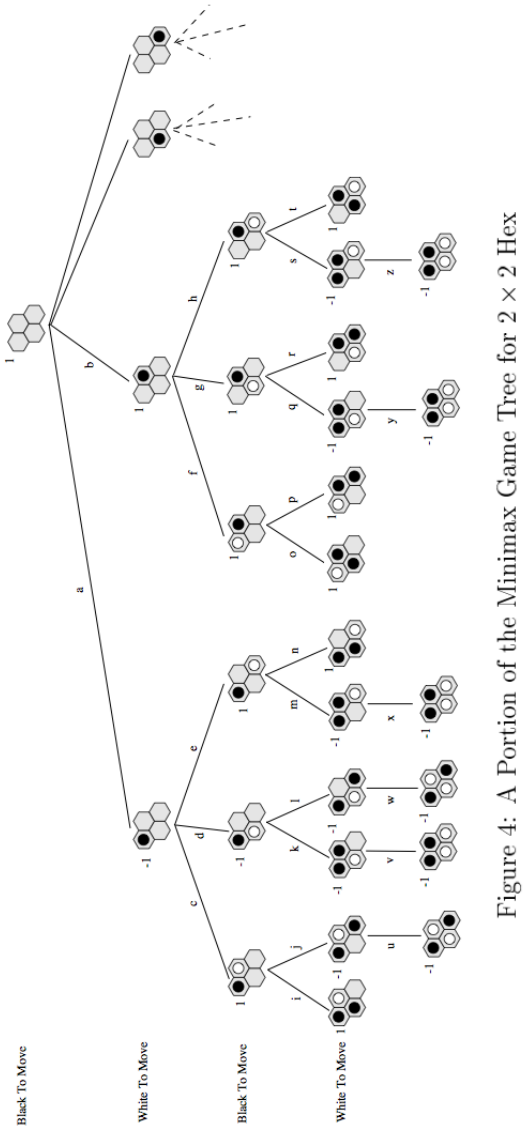

Figure 4: A Portion of the Minimax Game Tree for 2×2 Hex

FIGURE 4 – Le graphe d'un arbre de jeu 2x2