Etat d'art Hasami Shogi

Siwar Souissi, Mael Houbre, Baptiste Monchicourt

March 2017

1 Introduction

Dans cet état de l'art, nous allons présenter le jeu que nous avons choisi pour le projet de C, le **"Hasami shogi"**. Nous détaillerons par la suite l'origine du jeu, son principe et ses règles, afin de construire une idée générale sur le livrable à fournir à la fin du projet.

2 Les jeux de stratégie combinatoire abstraits

Un jeu de stratégie est un jeu de société faisant appel à la réflexion. L'objectif de ce type de jeu est d'augmenter sa domination spatiale en combattant un ou plusieurs ennemis sur un terrain de jeu. Pour nous, le terrain de jeu est un plateau de 9x9 cases.L'accent est mis sur la planification de l'action (tactique ou stratégique).On parle essentiellement soit de jeux de guerre ou de jeux de simulation, informatisés ou non.

Cependant notre jeu est considéré comme un jeu de stratégie combinatoire abstrait. Un jeu de stratégie combinatoire abstrait comporte un concept de position. Les joueurs jouent à tour de rôle un coup d'une façon définie par des règles, dans le but de remporter la victoire qui est définie par une certaine situation. C'est par exemple le cas du jeu d'échecs. Le vocable " abstrait " est synonyme de " théorique". Pour résumer, ce type de jeu est :

1. Une opposition de deux joueurs, deux équipes que ce soit deux équipes humaines ou une équipe contre une intelligence artificielle .
2. Dans lequel les joueurs ou équipes jouent à tour de rôle. Dans notre jeu où les joueurs jouent avec des pierres de couleurs, le joueur noir commence.
3. Dont tous les éléments sont connus (jeu à information complète) ;
4. Où le hasard n'intervient pas pendant le déroulement du jeu.

On trouve comme exemple de jeux de stratégie combinatoire abstrait notamment les autres jeux proposés pour le projet de C cités en gras:

- **Les jeux Asiatiques:** le jeu de go, le xiangqi, **le shōgi**, **le Bagh Chal**, le congkak..
- **Les jeux Africains:** le Fanorona, **l'Awalé** ou le En Gehé
- **Les jeux modernes:** Othello, **Abalone**, **Hex**, Gounki, Puissance 4, Twixt, Diaballik, Apagos...

3 Le Hasami Shogi

Le Hasami shogi est un jeu de stratégie combinatoire abstrait et une des variantes de shogi (communément appelé "jeu d'échecs japonais"). Ce type de jeux est très populaire au Japon. **Hasami** veut dire "Prise en sandwich" qui décrit la manière dont on capture les pions de son adversaire.

3.1 Les variantes de shogi

De nombreuses variantes de shogi ont été développées au cours des siècles. Certaines faisant partie des plus grands jeux d'échecs jamais joués. Quelques-unes de ces variantes sont toujours jouées actuellement, mais aucune n'est aussi populaire que le shogi lui-même.

Par Ordre d'apparition et Taille :

- **Du 9ème siècle au 12ème siècle:** la plus ancienne description japonaise survivante des règles d'échecs pendant la période des Heian. Les informations pour jouer ce jeu ne sont pas claires mais cela n'a pas empêché le gens de reconstruire cette forme primitive appelée **Heian Shogi**. Le tableau semble avoir été 9 x 8 ou 8 x 8.
- **Au 14ème siècle:**La variante la plus populaire sur un grand plateau est **Chu Shogi**, joué sur une planche 12 x 12.Il existe d'autres variantes du shogi (les tailles de plateau sont indiquées entre parenthèses):
  - Le **Wa Shogi** (11 x 11)
  - Le **Dai Shogi** (15 x 15)
  - Le **Tenjiku Shogi** ou "Shogi Indien" (16 x 16)
  - Le **Dai-dai Shōgi** (17 x 17)
  - Le **Maka dai-dai shōgi** (19 x 19)
  - Le **Tai shogi** (25 x 25)
  - Le **taikyoku shogi** (36 x 36).

- **Au 16ème siècle:** Le jeu a été joué sur une planche 9 x 9 avec la même configuration que dans le shogi moderne, sauf qu'une pièce supplémentaire (un éléphant ivre) se tenait devant le roi, appelée **sho shogi**. L'éléphant ivre a été éliminé par l'empereur Go-Nara (règne 1526-1557).

- **Au 20-21ème siècle :**

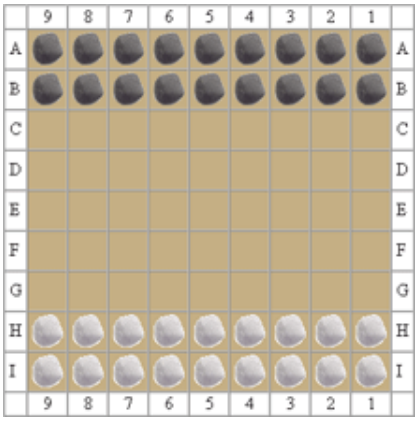
  - **De petite Taille:** Bushi shogi (1x2,année 2000), Nana shogi (3x3,année 1998/2001), Minishogi (5x5, c. 1970), Judkins shogi, Whale shogi (6x6,1981-1998)...
  - **De taille standard:** Sho shogi, Cannon shogi , **Hasami shogi** et le Dai Hasami Shogi (ces deux derniers sont souvent confondus). La seule différence est que dans la première, 9 pierres sont utilisées contre 18 dans la seconde. Le Hand shogi, Annan shogi (9x9).
  - **De grande taille:** Nutty shogi, Cashew shogi (13x13, 2015), Hishigata shogi (1919, 2005)...
  - **A trois dimensions:** Sannin shogi (7x7x7 hexagonal, c. 1930), Space Shogi (9x9x9, 1987).

3.2 Principe du jeu

Remarque:
Les figures suivantes représentent une variante du Hasami Shogi, **"Le Dai Hasami Shogi** . Nous les avons utilisées parcequ'elles expliquent efficacement les règles du jeu qui sont les mêmes pour ces deux variantes de shogi.

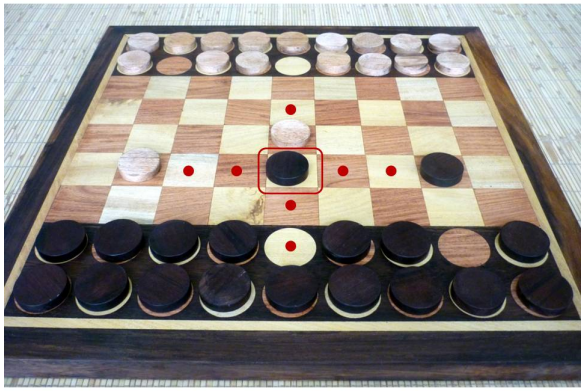
3.2.1 Le plateau de jeu

Le principe du jeu est assez simple. Deux joueurs, Noir et Blanc (ou sente et gote), jouent sur une planche shogi - une grille de 9 rangs (lignes) et 9 fichiers (colonnes). Les carrés sont indifférenciés par marquage ou couleur. Chaque joueur a 9 pierres disposées sur les deux premières lignes du plateau.



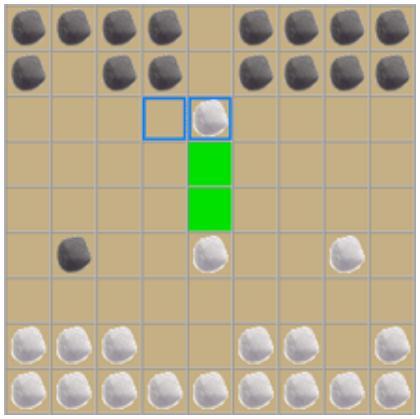
3.2.2 Le déplacement

A chaque tour, le joueur peut déplacer un pion d'autant de cases qu'il veut à l'horizontale ou à la verticale à la manière de la tour dans le jeu d'échec classique. Si un pion adverse adjacent lui bloque le passage et que la case suivante est libre, il peut sauter celui-ci.



3.2.3 Capturer les pierres adverses

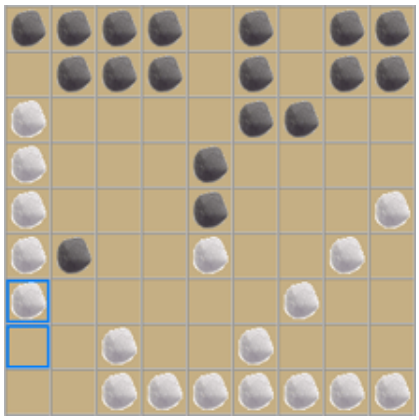
Un pion adverse est pris lorsqu'il est encadré par deux de nos pions,"le Sandwiching". C'est-à-dire lorsqu'un de nos pions est présent à sa droite et un autre à sa gauche (verticalement) ou un devant lui et un derrière lui (horizontalement).



Les cases vertes sont occupées par des pierres Noirs et capturés par les pierres Blanches.

3.2.4 Les conditions de victoire

La partie est gagné si l'un des joueurs a réussi à avoir une ligne de 5 pions à la diagonale ou à la verticale (la ligne à gauche dans la figure suivante) ou bien à l'horizontale si l'alignement est réalisé en dehors de la position de départ. Aucune pierre de cette ligne ne peut se situer sur les rangées de départ des joueurs (la ligne diagonale située à droite de la figure suivante). Une autre façon de gagner est de capturer les pions de l'adversaire jusqu'à ce qu'il n'en reste plus qu'un.



4 Algorithmes utilisés:

4.1 Intelligence Artificielle: IA

Hasami shogi se joue à tour de rôle. On peut représenter une partie par un arbre enraciné en la *position initiale*. Chaque nœud représente une *position possible* et les arêtes d'un même niveau représentent chacune *un coup joué* par le joueur dont c'est le tour. Le but est donc de faire parcourir à l'intelligence artificielle cet arbre afin de trouver la meilleur suite de coups qui la mène à la victoire. On utilise un algorithme de **parcours d'arbre** et **fonction d'évaluation** une qui permet de donner un score à chaque nœud.

4.1.1 Algorithmes de recherches

Le jeu de Hasami Shogi est en quelque sorte un jeu d'échecs où chaque joueur veut capturer le plus de pions adverses pour le faire perdre. On a trouvé que c'est important de détailler les algorithmes de recherches souvent utilisés dans ce type de jeux pour déterminer la meilleure suite de coups à jouer. Ces algorithmes ont permis à un ordinateur de gagner à coup sûr contre un joueur humain.

MinMax

1. **Principe:**

   L'algorithme minimax est un algorithme qui consiste à parcourir l'arbre du jeu à une certaine profondeur n et on prend successivement le maximum puis le minimum des scores obtenus aux feuilles.
   Le but de min-max est de trouver le meilleur coup à jouer à partir d'un état donné du jeu.Max est le joueur qui lance l'algorithme et qui veut gagner et Min est son adversaire. Max souhaite maximiser son score et min souhaite minimiser le score de max. On calcule le score une fois la feuille atteinte. A chaque coup de MinMax on crée une copie du plateau courant pour chaque déplacement de chaque pion.

   - **Pourquoi?** Dans les cas où il est délicat, voire impossible d'annuler un coup, on peut envisager de créer une copie du plateau de jeu avant de simuler un coup, puis de le restaurer ensuite à l'aide de la copie.
   - **Comment ?** Cette copie est effectuée sur un nœud lorsque l'on regarde ses fils, et est supprimée aussitôt l'analyse de ce nœud terminée.
   - **But?** Cette copie permet d'avoir en mémoire un nombre limité de copies à la fois.

2. **Pseudocode:**

   On aboutit à l'algorithme suivant formé par les trois fonctions suivantes :

```
fonction jouer : void

     max_val <- -infini

     Pour tous les coups possibles
          simuler(coup_actuel)
          val <- Min(etat_du_jeu, profondeur)

          si val > max_val alors
               max_val <- val
               meilleur_coup <- coup_actuel
          fin si

          annuler_coup(coup_actuel)
     fin pour

     jouer(meilleur_coup)
fin fonction
```

La fonction qui cherche le minimum est la suivante:

```
fonction Min : entier

     si profondeur = 0 OU fin du jeu alors
          renvoyer eval(etat_du_jeu)

     min_val <- infini

     Pour tous les coups possibles
          simuler(coup_actuel)
          val <- Max(etat_du_jeu, profondeur-1)

          si val < min_val alors
               min_val <- val
          fin si

          annuler_coup(coup_actuel)
     fin pour

     renvoyer min_val
fin fonction
```

La fonction qui cherche le maximum est la suivante:

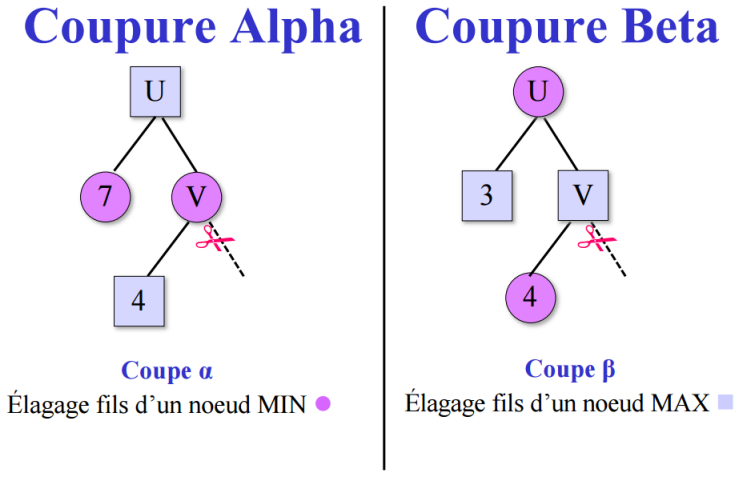
```
fonction Max : entier

     si profondeur = 0 OU fin du jeu alors
          renvoyer eval(etat_du_jeu)

     max_val <- -infini

     Pour tous les coups possibles
          simuler(coup_actuel)
          val <- Min(etat_du_jeu, profondeur-1)

          si val > max_val alors
               max_val <- val
          fin si

          annuler_coup(coup_actuel)
     fin pour

     renvoyer max_val
fin fonction
```

Elagage AlphaBeta

AlphaBeta est une optimisation de MinMax sans modifier le résultat. En effet, MinMaw effectue une exploration complète de l'arbre de recherche jusqu'à un niveau donné. L'élagage permet de parcourir l'arbre partiellemnt en coupant des branches que l'on sait inutile de parcourir, et permettant d'améliorer la profondeur, et donc l'efficacité du parcours de l'arbre enraciné. **une heuristique** (pions du joueur - pions de l'adversaire) est utilisée à chaque noeud afin de trier les branches et accélérer les coupes pour alpha-beta. Cet algorithme permet d'optimiser la place en mémoire.

1. **Principe:**

   - **Coupure Alpha:**
     Si la valeur d'un fils d'un noeud Min est inférieure à la valeur courante d'un noeud Max ancêtre alors les autres frères du fils n'ont pas besoin d'être explorés.
   - **Coupure Beta:**
     Si la valeur d'un fils d'un noeud Max est supérieur à la valeur courante d'un noeud Min ancêtre alors les autres frères du fils n'ont pas besoin d'être explorés.



2. **Pseudocode:** Voici le pseudocode de l'algorithme Alpha-Beta:

```
function alphabeta (p, alpha, beta) /* alpha est toujours inférieur
à beta */
    si p est une feuille alors
        return la valeur de p
    sinon
        si p est un noeud Min alors
            val= infini
            pour tout enfant pi de p faire
                val = Min( val ,alphabeta (pi, alpha, beta))
                si alpha >= val alors /* coupure alpha */
                    return val
                finsi
                beta = Min(beta, val)
            finpour
        sinon
            val= - infini
            pour tout enfant pi de p faire
                val = Max(val,alphabeta (pi, alpha, beta))
                si val >= beta alors /* coupure beta */
                    return val
                finsi
                alpha = Max (alpha, val)
            finpour
        finsi
    return val
    finsi
fin
```

NegaMax

1. **Principe:**
   Negamax est une amélioration de AlphaBeta. Dans cet algorithme on ne cherche qu'à maximiser le score. On fait applel juste au maximum et pour le minimum on donne l'inverse de alpha. Dans cet algorithme, il y a moins de tests effectués de façon à gagner en temps de calcul.

   Dans un jeu à deux joueurs à somme nulle, pour que le score du joueur augmente de 10 $, il faut que l'autre joueur perd 10 $. Au fur et à mesure que la recherche avance dans l'arborescence du jeu, chaque pli alterne le joueur pour se déplacer. Au lieu de suivre explicitement le joueur qui se déplace, on peut utiliser le signe du score de position pour représenter les mouvements alternatifs de chaque joueur.

   Le code negmax retourne l'opposé de la position du score où on est, renvoyé par la recherche récursive. Ce score est ce que l'adversaire aurait trouvé. Donc, si la position a un score élevé, alors c'est bon pour l'adversaire et mauvais pour nous. Si la position que l'adversaire a trouvé a un score bas, alors nous savons que cela doit être bon pour nous. En donnant l'opposé le score de position donne le même effet que manipuler explicitement la maximisation et la minimisation.

2. **Pseudocode:**

```
function Negmax (p, a, b ); // a est toujours inférieur à b
    si p est une feuille alors
        return la valeur de p
    sinon
        Meilleur = - infini
        pour tout enfant pi de p faire
            val = - Negmax (pi, -b, -a)
            si val > Meilleur alors
                Meilleur = val
                si Meilleur > a alors
                    a = meilleur
                    si a >= b alors
                        return Meilleur
                    finsi
                finsi
            finsi
        finpour
        return Meilleur
    finsi
fin
```

Fonction d'évaluation

Dans le cadre de la programmation d'un jeu de réflexion, la fonction d'évaluation devra être conçue de manière à représenter au mieux la situation du joueur. Elle est utilisée au niveau des feuilles de l'arbre de recherche. Il faut donc trouver le bon compromis entre précision de l'évaluation de la position et rapidité d'exécution pour le parcours de l'arbre. S'il existait une fonction d'évaluation parfaite, il ne serait pas nécessaire de prévoir plusieurs coups en avance. D'un point de vue algorithmique, elle est difficile à estimer surtout avec le nombre de possibilités de cas énorme. Il faudra être en mesure de modéliser de manière précise et fidèle à la réalité.
La fonction d'évaluation peut être caculée en utilisant trois critères:

$$\mathbf{f(p)} = \alpha 1\ \mathbf{mobilité(p)} + \alpha 2\ \mathbf{matériel(p)} + \alpha 3\ \mathbf{Hasami(p)} + \alpha 4\ \mathbf{imprévisible(p)}$$

- **mobilité(p):** Ce critère consiste à faire le coup qui permet d'augmenter la mobilité du joueur et en même temps réduire celle de l'adversaire.(C'est la différence de mobilité entre le joueur ayant le trait et le joueur adverse, pour la position p.)
- **matériel(p):** C'est la différence entre le nombre de pions du joueur entrain de jouer et le joueur adverse, pour la position p.
- **Hasami(p):** C'est le coup à faire afin d'avoir le nombre de pions de l'adversaire qu'on peut capturer, en essayant de ne pas laisser la possibilité contraire d'avoir lieu, pour la position p.

- **imprévisible(p):** On peut penser à une façon de réduire le critère prévisiblé des coups à faire par la IA. En effet, dans le cas où on des positions avec le même score, on ne doit pas forcément choisir le premier coup à venir. On peut choisir un coup aléatoire.

Pour le choix des critères sont à définir en lisant des articles sur les stratégies importantes et dans le cas de notre jeu, on va opter pour les critères précédents. Pour le choix des paramètres linéaires, il faut surtout savoir pondérer entre les différents critères afin d'avoir un algorithme plus performant. La fonction d'évaluation décide de la performance de l'intelligence artificielle.

5 Bibliographie

- **Pour la rédaction des règles de Hasami shogi:**

  https://en.wikipedia.org/wiki/Shogi_variant https://fr.wikipedia.org/wiki_de_strat%C3%A9gie
  http://abstractstrategygames.blogspot.fr/2010/10/hasami-shogi.html
  https://fr.wikipedia.org/wiki/Jeu_de_strat%C3%A9gie_combinatoire_abstrait
  https://brainking.com/fr/GameRules?tp=73

- **Pour la rédaction des algorithmes utilisés:**

  - **MinMax**
    https://github.com/pfmonville/Hasami-Shogi https://openclassrooms.com/courses/l-algorithme-min-max
  - **AlphaBeta et Negmax**
    https://fr.wikipedia.org/wiki/%C3%89lagage_alpha-b%C3%AAta
    http://ligmembres.imag.fr/pernet/Enseignements/L3METI_ProgOO/MinMaxAlphaBeta.pdf
    https://github.com/pfmonville/HasamiShogi
    https://interstices.info/jcms/int$_7$0460/$programmationdesechecsetd-autresjeuxhttps : //equilibriumofnothing.wordpress.com/2013/10/15/algorithm - negmax/$
  - **Fonction d'évaluation**
    https://github.com/pfmonville/HasamiShogi
    http://fluminis.free.fr/Rapport$_E checs.pdf$