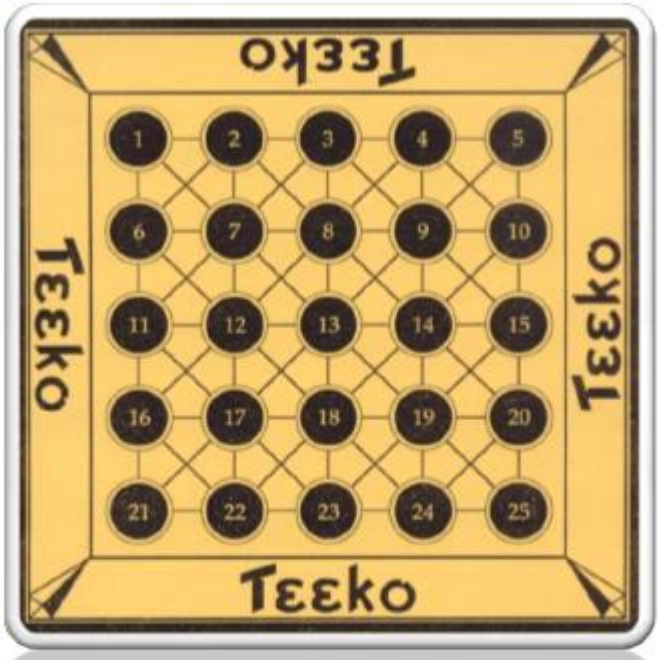
Morgane Le Bris
Salomé Welche
Etienne Edouard

Projet *Teeko*
IA41 P15



Morgane Le Bris
Salomé Welche
Etienne Edouard

Sommaire

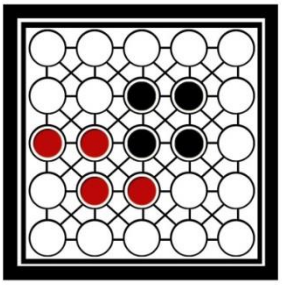
Morgane Le Bris
Salomé Welche
Etienne Edouard

I. Rappel de l’énoncé

Tout d’abord un peu d’histoire : Teeko est un jeu de stratégie combinatoire abstrait inventé par le magicien John Scarne en 1937, revu en 1952, puis une dernière fois au cours des années 60. Le mot " teeko " vient des autres jeux qui ont inspiré l'inventeur : le "T" de tic-tac-toe, le "E" de chess ("échecs"), le "K" de checkers ("damier") et le "O" de bingo. À quoi Scarne rajouta un "E" pour des raisons phonétiques.

Le Teeko se joue à deux et repose sur :

- Un plateau de jeu composé de vingt-cinq cases (cinq par cinq)
- Quatre pions de couleur noire (joueur "Noir")
- Quatre pions de couleur blanc (joueur "Blanc")
- Le but du jeu est d’aligner (horizontal, vertical, diagonal) ses pions ou de les positionner en carré.



Maintenant les règles du jeu :

- Le plateau est vide en début de partie.
- Tour à tour, les joueurs posent un de leurs pions sur une intersection libre (une case libre)
- emplacements de 1 à 25 sur la figure ci-dessus).
- Une case est libre si elle ne contient pas déjà un pion.
- A la fin de cette phase de pose, si aucun joueur n'a obtenu de configuration gagnante, les joueurs déplacent à tour de rôle l'un de leurs pions.
- Un pion déplacé ne peut l'être que sur un emplacement libre adjacent à des pions.
- On peut donc déplacer un pion sur une case dont au moins une de ces cases adjacentes n’est pas vide.
- Dès qu'un joueur réalise une configuration gagnante, la partie s'arrête.

Morgane Le Bris
Salomé Welche
Etienne Edouard

II. Analyse du Problème

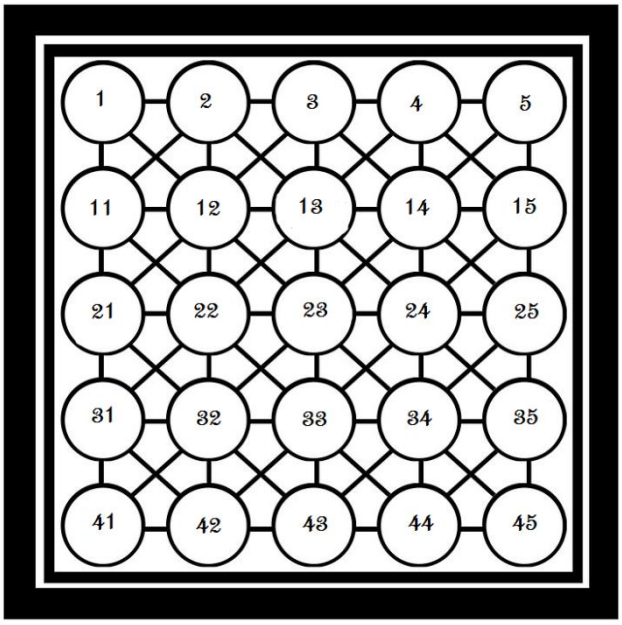
Nous nous fixons comme objectif d'implémenter le jeu du Teeko permettant de jouer une partie humain contre humain et également une partie humain contre ordinateur ou ordinateur contre ordinateur.

On représentera un coup comme une liste de deux éléments [a, b]: a étant le numéro de la case contenant le pion à bouger (0 si on se trouve dans la première phase) et b le numéro de la case dans laquelle il faut placer le pion.

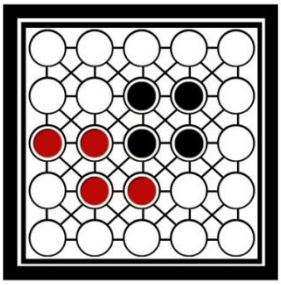
Le problème de l'écriture du programme a d'abord été découpé en sous-problèmes :

- Définir la représentation des données (le plateau de jeu)
- Ecrire la fonction d'affichage
- Ecrire la fonction d'évaluation qui permet d'attribuer un score à tous les coups possibles des joueurs
- Algorithme utilisé pour l'IA : alpha beta

III. Méthodes proposées



Nous avons décidé de représenter notre plateau par 2 deux listes croissantes de quatre chiffres représentants la place des pions de chaque joueurs sur le plateau. Le numéro de la place d'un pion est déterminé par le tableau ci-contre. Par exemple dans le tableau ci-dessous la liste du joueur noire est [13, 14, 23, 24].



Morgane Le Bris
Salomé Welche
Etienne Edouard

Voici les principales fonctions que nous avons faites :

- *play()* qui lance le jeu et donne le choix du mode de jeu

- *display_board(+BoardBlanc,+BoardNoir)* permet l'affichage du plateau. Un B signifie un jeton blanc, N signifie un jeton noir et X une case vide.

```
---------------
B  X  B  X  N
B  X  X  N  X
X  X  N  N  X
X  B  X  X  X
X  X  X  X  X
---------------
```

- *getPossibleMovementL(+Joueur, +Adversaire, -ListeMvmt)* qui retourne tous les mouvements possibles d'un joueur à partir de ses pions posés.

- *ordonner(+L, -O)* qui ordonne la liste L dans l'ordre croissant et donne O

- *deplacement(+A,+B)* qui regarde si le déplacement A-B est bien atteignable par l'un des 8 déplacements possibles.

- *getPossibleMovementX(+Depart, +PionsJoueur, +PionsAdversaire, -ListeMvmtPossibles, +NumeroIteration)* qui retourne la liste de tous les mouvements possible du joueur en fonction de l'endroit où sont ses pions et ceux de son adversaire.

- *move(+Joueur, +From, +To, -JoueurBis)* qui rend le tableau du joueur avec le mouvement effectué (From remplacé par To et le tableau est de nouveau ordonné).

- *score(+BoardBlanc,+BoardNoir,+Init, ?ScoreTotal)* qui calcul la valeur d'une position suivant la position de chaque pion. Pour cela elle prend en compte les 44 possibilités gagnantes. Elle regarde combien de pions amis et de pions ennemis puis attribue un score à chaque possibilité et fait la somme de tous ces résultats pour donner une valeur à une position de tableau.

- *searchBest(+Joueur, +Adversaire, +Mouvement, +Profondeur, +Alpha, +Beta ,+R, +OriginalAdversaire, ?MeilleurDeplacement)* qui à l'aide des fonctions score et getPossibleMovementX retourne le meilleur coup à jouer.

- *win(+Board,-win,+J)* qui regarde si le joueur a une des 44 possibilités gagnantes. Elle retourne J dans win si c'est le cas (4 si le blanc gagne et 5 si c'est le noir) et 0 si ce n'est pas le cas.

- *alphaBeta(+Joueur1,+Adv,+Profondeur, ?TableauDonné, ?ValeurDuCoup)* fonction alphabéta qui calcul le meilleur coup que l'IA peut faire. Le coup sera plus ou moins bien suivant la Profondeur qui représente la taille de l'arbre parcouru, elle est définie par le niveau que l'utilisateur a donné au début.

Morgane Le Bris
Salomé Welche
Etienne Edouard

L'Algorithme AlphaBeta

Pour faire jouer l'ordinateur correctement, nous avons besoin d'évaluer et choisir le meilleur coup. Pour cela deux algorithmes sont à notre disposition : le minimax et l'alphabeta.

Nous avons décidé d'implémenté l'alphabeta car ce dernier est plus optimisé.

❖ Principe du MiniMax :

Cette technique amène l'ordinateur à passer en revue toutes les possibilités pour un nombre limité de coups (une profondeur) et à leur assigner une valeur qui prend en compte les bénéfices pour le joueur et pour son adversaire. Le meilleur choix étant alors celui qui maximise ses bénéfices et minimise ceux de son adversaire.
Cet algorithme s'applique au jeu où s'opposent deux joueurs. De plus, on suppose qu'aucune partie de ce jeu ne comporte un nombre infini de coups, mais qu'à chaque coup il existe un nombre fini de possibilités.
En fait, on parcourt l'arbre de jeu pour faire remonter à la racine une valeur dite valeur de jeu qui est calculé récursivement de la façon suivante (f est la fonction d'évaluation) :
MiniMax (coup) = f (coup) si coup est une position terminale
MiniMax (coup) = max (MiniMax(h1),...,MiniMax(hn)) avec hn les fils du coup
MiniMax (coup) = min (MiniMax(j1),...,MiniMax(jn)) avec jn les fils mais c'est le coup de l'autre joueur

❖ Principe de l'AlphaBeta:

L'alphabeta permet l'élagage de l'arbre. Cet élagage permet de réduire le nombre de nœuds évalués par l'algorithme MiniMax. L'algorithme MiniMax effectue en effet une exploration complète de l'arbre de jeu jusqu'à une profondeur donnée, alors qu'une exploration partielle de l'arbre est généralement suffisante. Lors de l'exploration, il n'est pas nécessaire d'examiner les sous-arbres qui conduisent à des configurations dont la valeur ne contribuera sûrement pas au calcul du gain à la racine de l'arbre. L'élagage AlphaBeta évite d'évaluer des nœuds de l'arbre dont on est sûr que leur qualité sera inférieure à un nœud déjà évalué.
Pour implémenter ceci, nous avons fait une fonction cutBranches qui vérifie si oui ou non on peut couper la branche.

IV. Situations traitées

Nous avons décidé de faire 3 modes de jeu :

- joueur vs joueur
- joueur vs IA
- IA vs IA

L'IA peut avoir 4 niveaux de difficultés que l'utilisateur définira au début (1 étant la plus faible et 4 la plus forte). Mais à partir de 4 l'IA peut mettre plusieurs minutes pour réfléchir à un coup.

V. Résultats

Nous allons vous présenter le mode de jeu d'un joueur contre une IA. Cette dernière sera de niveau 2.

| play.
1: joueur vs joueur
2: joueur vs ia
3: ia vs ia
|: 2.
choisissez une IA d'un niveau entre 1 et 4
|: 2.

```
_______________
 X  X  X  X  X
 X  X  X  X  X
 X  X  X  X  X
 X  X  X  X  X
 X  X  X  X  X
_______________
```

Morgane Le Bris
Salomé Welche
Etienne Edouard

joueur blanc 0 eme tour
entrez le numéro de la ligne voulue
|: 2.
entrez le numéro de la colonne voulue
|: 2.

```
---------------
 X  X  X  X  X
 X  B  X  X  X
 X  X  X  X  X
 X  X  X  X  X
 X  X  X  X  X
---------------
---------------
 X  X  X  X  X
 X  B  X  X  X
 X  X  N  X  X
 X  X  X  X  X
 X  X  X  X  X
---------------
```

joueur blanc 2 eme tour
entrez le numéro de la ligne voulue
|: 2.
entrez le numéro de la colonne voulue
|: 3.

```
---------------
 X  X  X  X  X
 X  B  B  X  X
 X  X  N  X  X
 X  X  X  X  X
 X  X  X  X  X
---------------
---------------
 X  X  X  X  X
 X  B  B  N  X
 X  X  N  X  X
 X  X  X  X  X
 X  X  X  X  X
---------------
```

joueur blanc 4 eme tour
entrez le numéro de la ligne voulue
|: 1.
entrez le numéro de la colonne voulue
|: 2.

```
---------------
 X  B  X  X  X
 X  B  B  N  X
 X  X  N  X  X
 X  X  X  X  X
 X  X  X  X  X
---------------
---------------
 X  B  X  X  N
 X  B  B  N  X
 X  X  N  X  X
 X  X  X  X  X
 X  X  X  X  X
---------------
```

joueur blanc 6 eme tour
entrez le numéro de la ligne voulue
|: 4.
entrez le numéro de la colonne voulue
|: 2.

```
---------------
 X  B  X  X  N
 X  B  B  N  X
 X  X  N  X  X
 X  B  X  X  X
 X  X  X  X  X
---------------
---------------
 X  B  X  X  N
 X  B  B  N  X
 X  N  N  X  X
 X  B  X  X  X
 X  X  X  X  X
---------------
```

joueur blanc 8 tour
entrez le numéro de la ligne
|: 1.
entrez le numéro de la colonne
|: 2.
entrez le numéro de la ligne voulue
|: 1.
entrez le numéro de la colonne voulue
|: 3.

```
---------------
 X  X  B  X  N
 X  B  B  N  X
 X  N  N  X  X
 X  B  X  X  X
 X  X  X  X  X
---------------
---------------
 X  X  B  X  X
 X  B  B  N  N
 X  N  N  X  X
 X  B  X  X  X
 X  X  X  X  X
---------------
```

joueur blanc 10 tour
entrez le numéro de la ligne
|: 4.
entrez le numéro de la colonne
|: 2.
entrez le numéro de la ligne voulue
|: 3.
entrez le numéro de la colonne voulue
|: 1.

```
---------------
 X  X  B  X  X
 X  B  B  N  N
 B  N  N  X  X
 X  X  X  X  X
 X  X  X  X  X
---------------
---------------
 X  X  B  X  X
 X  B  B  N  X
 B  N  N  N  X
 X  X  X  X  X
 X  X  X  X  X
---------------
```

joueur blanc 12 tour
entrez le numéro de la ligne
|: 2.
entrez le numéro de la colonne
|: 3.
entrez le numéro de la ligne voulue
|: 1.
entrez le numéro de la colonne voulue
|: 4.

```
---------------
 X  X  B  B  X
 X  B  X  N  X
 B  N  N  N  X
 X  X  X  X  X
 X  X  X  X  X
---------------
---------------
 X  X  B  B  X
 X  B  N  N  X
 B  X  N  N  X
 X  X  X  X  X
 X  X  X  X  X
---------------
```

L'IA gagne !!
true.

VI. Difficultés

Nous avons essayé de créer une interface en Prolog puis ensuite en C++ mais dans les deux cas nous n'avons pas réussi. Nous avons aussi rencontrés plusieurs difficultés liés à la connaissance du Prolog. Nous nous sommes également posé de nombreuses questions sur la fonction d'évaluation (comment rendre les coups de l'IA intelligents) et sur l'implémentation de l'alphabéta.

Morgane Le Bris
Salomé Welche
Etienne Edouard

VII. Améliorations possibles

Il serait possible d'implémenter une meilleure interface graphique qui représente le plateau de jeu. Le joueur n'aurait qu'à cliquer sur le pion qu'il veut déplacer et sur la case vide adjacente sur laquelle il désire le poser.

On peut aussi améliorer notre IA pour la rendre plus performante, plus rapide mais également lui donner une mémoire des anciens mouvements effectués pour éviter une boucle infini lors du mode de jeu IA vs IA.

Conclusion

Ce projet est une application concrète des notions vues en cours, il nous a permis de solidifier nos connaissances en Prolog. De plus, nous avions tous déjà joué au jeu de société teeko ce qui a rendu ce projet d'autant plus intéressant. Nous avons pu aussi mieux comprendre le principe de l'intelligence artificielle grâce aux recherches effectuées pour implémenter l'AlphaBeta.

Il nous a également permis d'améliorer notre travail en groupe. Et de définir des tâches pour bien avancer dans la même direction.