**Institut de la Francophonie pour l'Informatique**


# RAPPORT DE TRAVAUX PRATIQUES 1
# INTELLIGENCE ARTIFICIELLE

Subjet : JEU DE GOMOKU

**Encadrement :** NGUYEN Manh Hung

**Étudiant :** NGUYEN Van Tho
**Promotion :** 17

Hanoï, Juin 2013

# 1 Introduction

L'objectif de ce TP est d'implémenter le jeu de gomoku qui peut joue automatiquement avec les autres programmes dans le concours de jeu de la classe. En fait ce jeu est une modification du jeu gomoku original : la séquence pour gagner a une longueur de 6 pièces au lieu de 5 pièces de celle originale.

Dans ce TP, j'ai implémenté le jeu avec une interface graphique, une heuristique pour choisir les bonnes cellules. J'ai choisit d'implémenter le programme en langage Java. L'interface est conçu grâce à bibliothèque Swing.

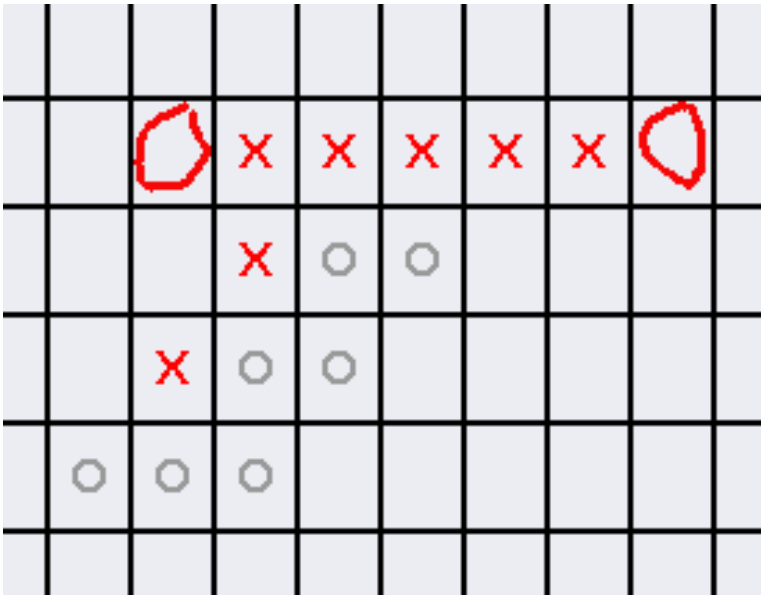
# 2 L'algorithme heuristique

Pour choisir une cellule j'utilise une heuristique pour calculer le poids de chaque cellule. Le programme défend quand il y a une menace, en revanche, il essaie de attaquer. La description d'heuristique est présenter ci-dessous :

## 2.1 Heuristique concernant la longueur de séquence de symboles consécutives

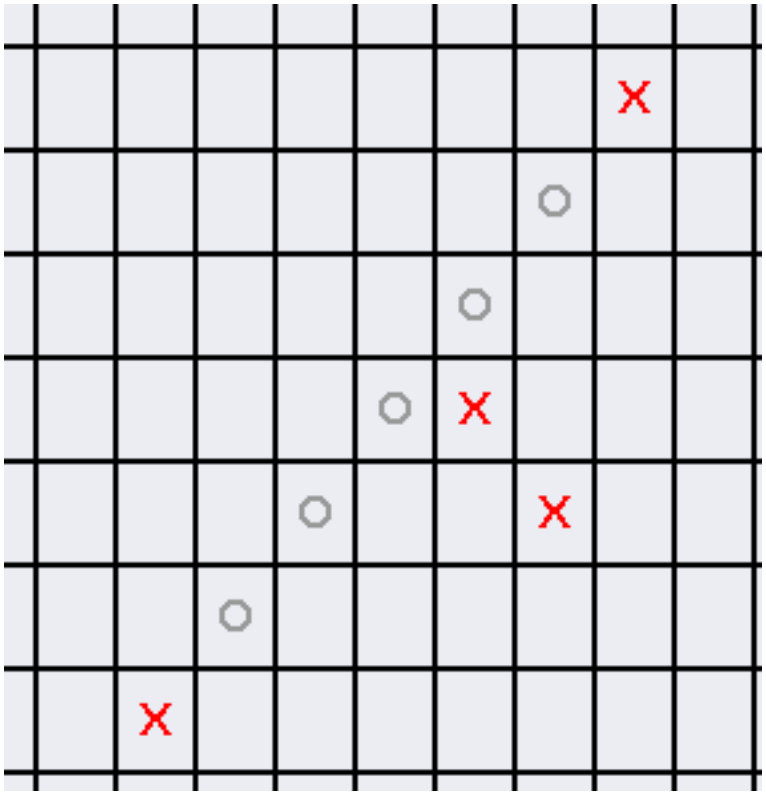
### 2.1.1 Une séquence de 6 symboles consécutives

Si le programme détecte une séquence de 6 symboles consécutives (ouvert ou fermé ou mi-fermé). La séquence peut être horizontale, verticale ou diagonale. Il va attaquer car il va gagner le jeu. Dans mon programme, cette séquence est assigné la plus grande priorité.



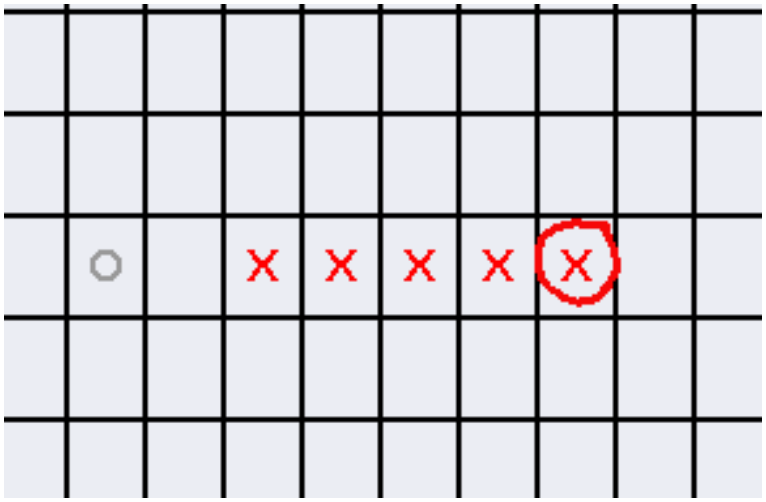
### 2.1.2 Une séquence de 5 symboles consécutives mi-fermé d'adversaire

Cette situation est une menace danger. Il faut donc occuper la cellule vide si non l'adversaire va gagner sauf qu'on peut créer une séquence de 6 symboles consécutives de notre symbole. Cette séquence est assigné une grande priorité égale la situation ci-dessus.



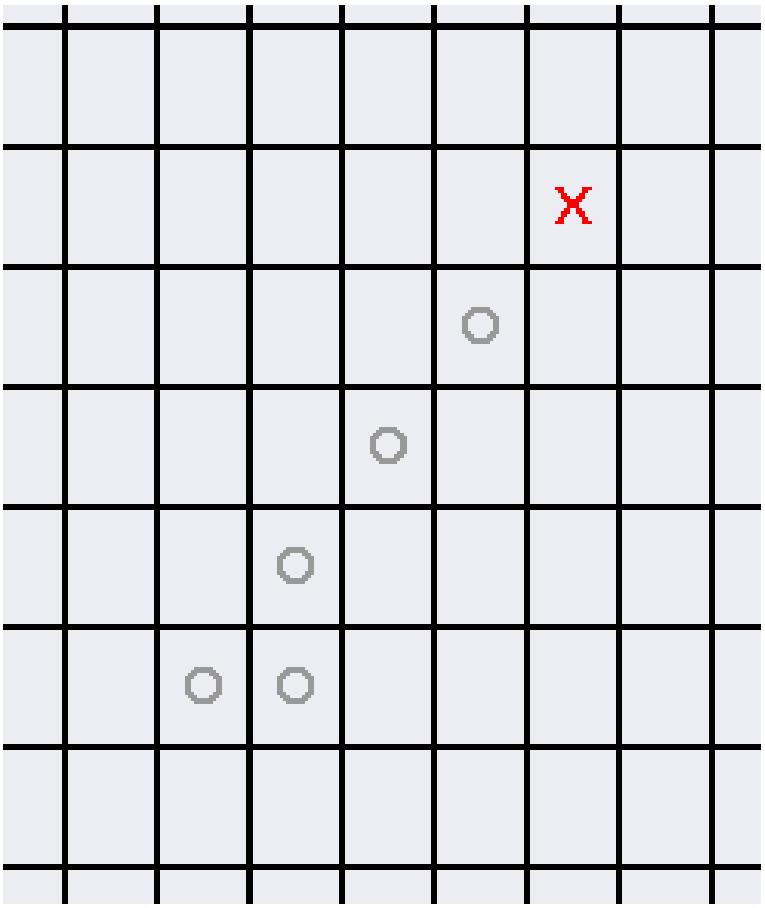
## 2.2 Une séquence ouverte de 5 symboles consécutives

Si on a une séquence de 4 ouverte symboles (les deux bouts ne sont pas occupés) on peut créer une séquence de 5 symboles et gagner le jeu car l'adversaire ne peut pas défendre. Je assigne donc une grande priorité pour cette situation :



### 2.2.1 Une séquence ouverte de 4 symboles consécutives d'adversaire

Cette situation est similaire à celle de la partie ci-dessus mais les symboles sont d'adversaire. Il faut donc occuper une de deux cellules aux bouts de cette séquence sinon l'adversaire va gagner. La priorité de cette situation est donc égale celle de partie ci-dessus :



### 2.2.2 Les autres séquences

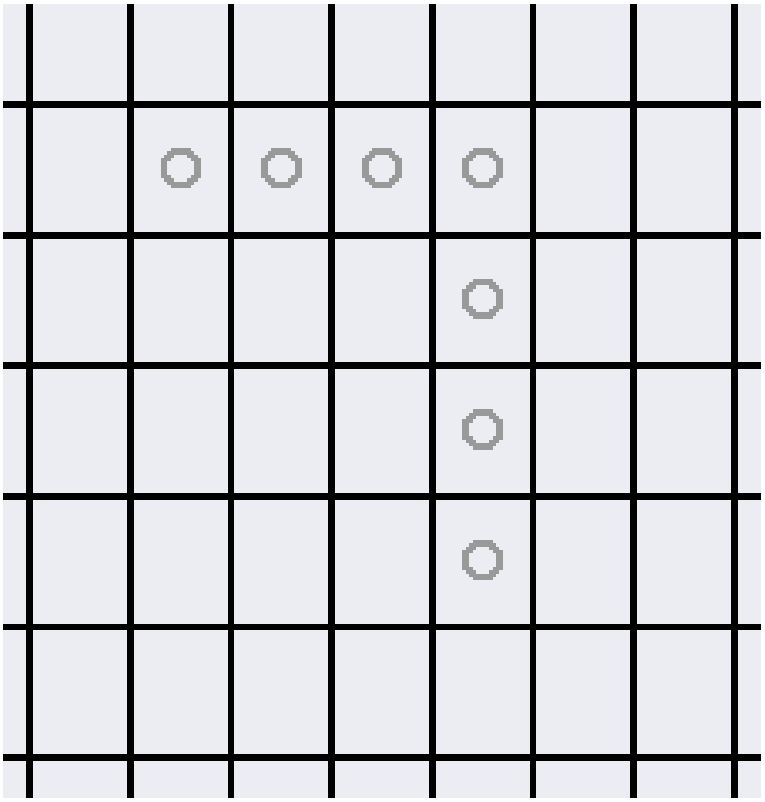
Pour les autres séquences (séquence de 3, de 2, de 1), la même heuristique est appliquée pour calculer le poids d'une cellule. Cependant, la priorité (le score) est moins que les situations ci-dessus.

## 2.3 Quelques heuristiques avancées

Les heuristiques je présente ci-dessus ne sont pas difficiles de défendre. J'ajoute donc quelques heuristiques avancées pour créer des situations qui sont difficiles de défendre ou menacer ces situations d'adversaire.

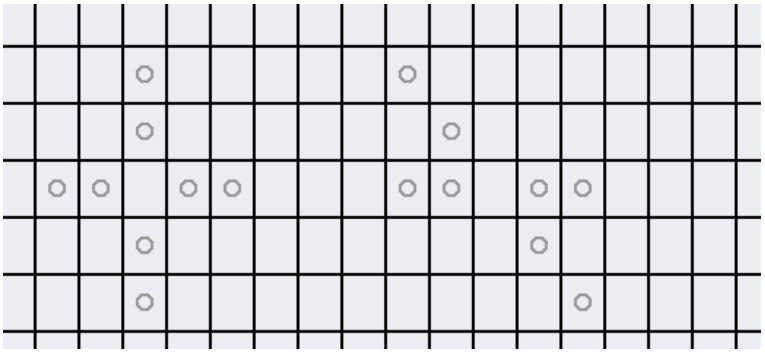
### 2.3.1 Deux séquences ouvertes de longueur 4

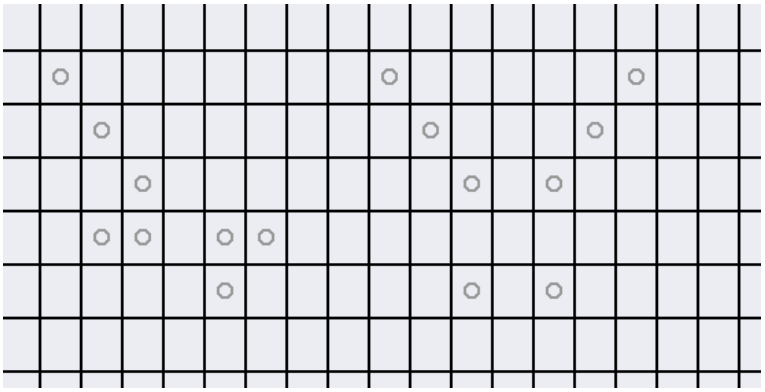
Si on peut créer ces séquences, on va gagner ou si l'adversaire crée ces séquences. Donc on va essayer de créer ces séquences ou menacer l'adversaire de les créer. Il y a plusieurs situations qui se composent par les séquences horizontales, verticales, diagonales. Une priorité égale à celle de la séquence ouverte de 5 symboles.



### 2.3.2 Deux séquences ouvertes de longueur 5 avec une cellule vide au milieu

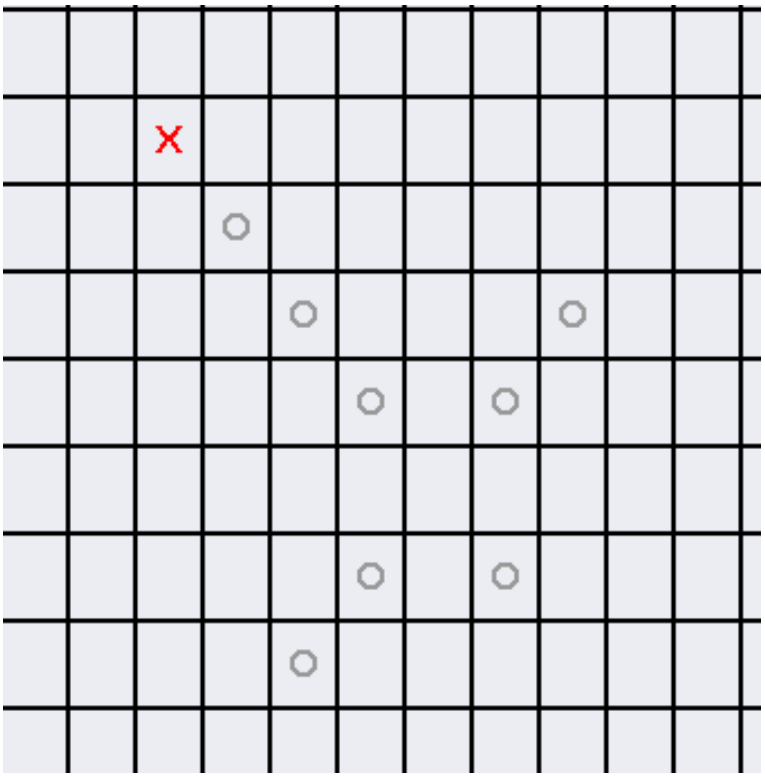
Il y a plusieurs configurations pour cette situation. Elle a la même priorité que Deux séquences ouvertes de longueur 4. Ces images présentent quelques configurations :

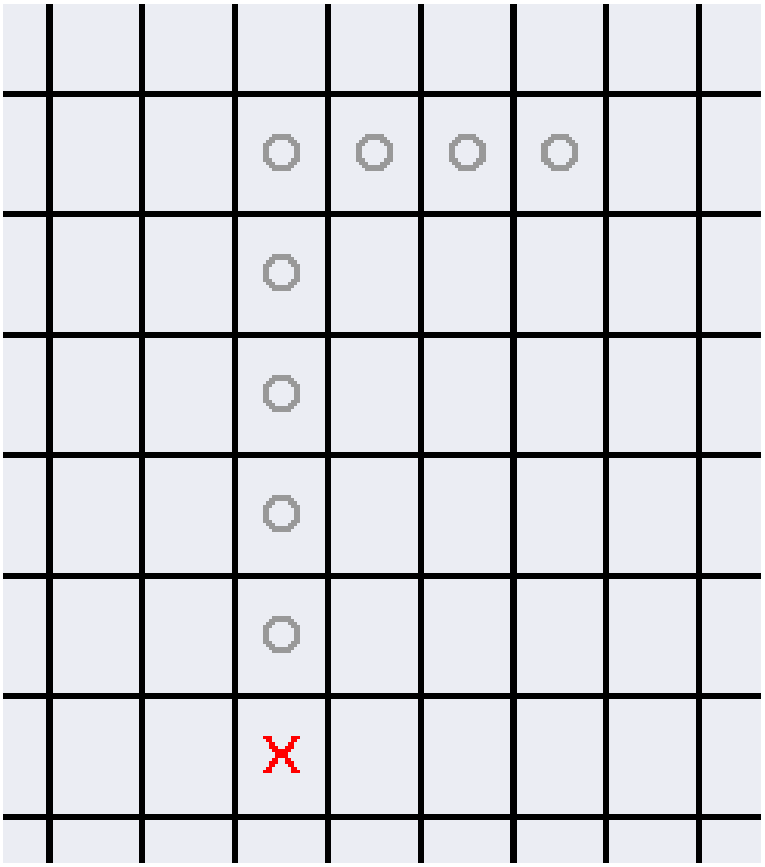




### 2.3.3 Autres situations considérées

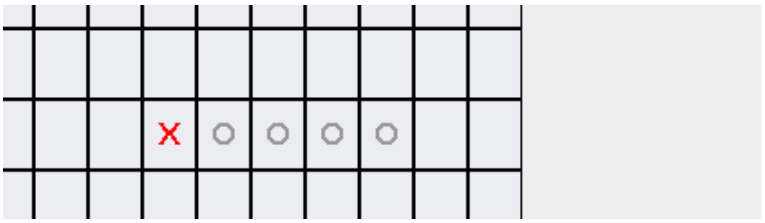
J'ajoute aussi quelques autres heuristiques qui peut gagner le jeu. Elles sont ressembles les deux heuristiques ci-dessus. . Ces images présentent quelques configurations :





### 2.4 Plus haute priorité pour les cellules au centre du plateau

Quand il y a deux cellules possibles. L'algorithme va choisir la cellule la plus proche du centre :

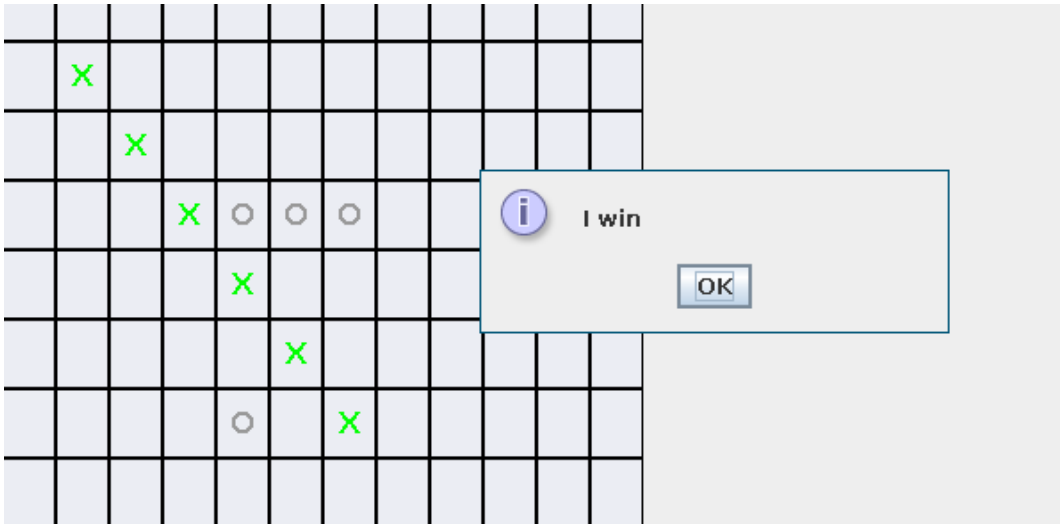


## 3 Le programme

Le programme est conçu avec une interface simple, satisfaire les exigences ci-dessous :

– Un plateau de 20x20 cellules

– Lire ou créer le fichier d'état de jeu gomoku.txt
– Le programme peut bien marcher sur linux ou windows. Sur windows il cherche le fichier d'état à C :
  IFI
  gomoku.txt. Sur linux, il le cherche au répertoire courent.
– Le premier joueur est le joueur X qui a les cellules qui vaux 2, le joueur corresponde aux cellules qui vaux 1. Les cellules vide sont 0.
– Vérifier le gagnant et souligner la séquence de gagne :




– Supporter plusieurs modes de jouer : Jouer automatiquement ou manuellement avec les autre programmes ou avec humain.

# 4 Conclusion et perspective

Dans ce TP, j'ai implémenté une version de jeu gomoku. J'utilise plusieurs heuristiques pour choisir la plus bonne cellule. Le programme défende bien grâce aux bonnes heuristiques. Cependant, le programme examine seulement les mouvement possibles avec la profondeur de 1. Il est donc très limite de créer les configurations dangers définissant par les heuristiques. L'implémentation d'une algorithme avec la profondeur plus grande comme minimax, negamax (avec alpha-beta) est considéré comme un travail dans le futur.