## Programming project

## Artificial Intelligence

Part I: MINIMAX (45%)

First program: MiniMaxOpening.java

The first program plays a move in the opening phase of the game with MiniMax algorithm.

Second program: MiniMaxGame.java

The second program plays in the midgame/endgame phase with MiniMax algorithm.

Part II: ALPHA-BETA (35%)

First program: ABOpening.java

The first program plays a move in the opening phase of the game with ALPHA-BETA pruning algorithm.

Second program: ABGame.java

The second program plays in the midgame/endgame phase with ALPHA-BETA pruning algorithm.

Part III: PLAY A GAME FOR BLACK (10%)

First program: MiniMaxOpeningBlack.java

The first program plays a move in the opening phase of the game with MiniMax algorithm for black.

Second program: MiniMaxGameBlack.java

The second program plays in the midgame/endgame phase with MiniMax algorithm for black.

Part IV: STATIC ESTIMATION (10%)

First program: MiniMaxOpeningImproved.java

Program plays a move in the opening phase of the game with MiniMax algorithm with improved static estimation function.

Second program: MiniMaxGameImproved.java

The second program plays in the midgame/endgame phase with MiniMax algorithm with improved static estimation function.

**Improvised static estimation function:**

As per the static estimation function discussed in class, we evaluate static estimation as the difference between black and white mens. For mid game/endgame estimate we estimate based on no of mens left for white and black and the number of black moves possible.

We can improvise the static estimation by taking account of the depth of search and mills that can be formed for a specific position. For opening static estimation is calculated in staticEstimationOpeningImproved() as

**return** ((numWhitePieces - numBlackPieces)\**ply*) + mills;

For staticEstimationMidgameEndgameImproved()

**return** ((1000 \* (numWhitePieces - numBlackPieces) \* depth) - blackMoveCount + (100 \* mills));

Considering mills that can be formed to evaluate static estimation will increase in efficient estimation.

Examples of the program output:

**Sample Test 1: Output of all programs for an input below:**

Input: xxxWxxxBxWxxxxxxBxxxBWW  
depth: 3

**Outputs:**

MiniMaxOpening:

Board Position: WxxWxxxxxWxxxxxxBWxxBWW

Positions evaluated by static estimation: 3696

MINIMAX estimate: 4

ABOpening:

Board Position: WxxWxxxxxWxxxxxxBWxxBWW

Positions evaluated by static estimation: 1241

AB estimate: 4

MiniMaxOpeningImproved:

Board Position: WxxWxxxxxWxxxxxxBWxxBWW

Positions evaluated by static estimation: 3696

MINIMAX estimate: 12

MiniMaxOpeningBlack:

Board Position: BxxWxxxxxWxxxxxxBWxxBWW

Positions evaluated by static estimation: 3528

MINIMAX estimate: 2

MiniMaxGame:

Board Position: WxxxxxxxxWxxxxxxBWxxBWW

Positions evaluated by static estimation: 509

MINIMAX estimate: 10000

ABGame:

Board Position: WxxxxxxxxWxxxxxxBWxxBWW

Positions evaluated by static estimation: 130

AB estimate: 10000

MiniMaxGameImproved:

Board Position: WxxxxxxxxWxxxxxxBWxxBWW

Positions evaluated by static estimation: 509

MINIMAX estimate: 10000

MiniMaxGameBlack:

Board Position: xxxWxxxxxWxBxxxxxWxxBWW

Positions evaluated by static estimation: 170

MINIMAX estimate: 10000

**Sample Test Cases 2: examples in which alpha-beta produces savings over MINIMAX.**

**Example 1:**

Input: BWBWBWBWBWWWWxWBWBxxBxB

depth: 2

**Outputs:**

MiniMaxOpening:

Board Position: BWBWBWBWBWWWWWWBWBxxBxx

Positions evaluated by static estimation: 136

MINIMAX estimate: 2

ABOpening:

Board Position: BWBWBWBWBWWWWWWBWBxxBxx

Positions evaluated by static estimation: 37

AB estimate: 2

MiniMaxGame:

Board Position: BWBWBWBWBWWWWxWBxxxWBxB

Positions evaluated by static estimation: 48

MINIMAX estimate: 1981

ABGame:

Board Position: BWBWBWBWBWWWWxWBxxxWBxB

Positions evaluated by static estimation: 42

AB estimate: 1981

**Example 2:**

Input: xxxxxxxxxWxxxxxxBxxxxxx

depth: 3

**Outputs:**

MiniMaxOpening:

Board Position: WxxxxxxxxWxxxxxxBxxxxxx

Positions evaluated by static estimation: 8056

MINIMAX estimate: 1

ABOpening:

Board Position: WxxxxxxxxWxxxxxxBxxxxxx

Positions evaluated by static estimation: 1145

AB estimate: 1

MiniMaxGame:

Board Position: xxxWxxxxxxxxxxxxBxxxxxx

Positions evaluated by static estimation: 42

MINIMAX estimate: 10000

ABGame:

Board Position: xxxWxxxxxxxxxxxxBxxxxxx

Positions evaluated by static estimation: 16

AB estimate: 10000

**Sample Test Cases 3: Examples to compare the static estimates given by static estimation and improved static estimation.**

* Opening:

Input: xxxWxxxBxWxxxxxxBxxxBWW

Depth:3

* 1. MiniMaxOpening:

Board Position: xxxWxxxxxWxxxxxxBWxxBWW

Positions evaluated by static estimation: 5747

MINIMAX estimate: 3

* 1. MiniMaxOpeningImproved:

Board Position: xxxWxxxxxWxxxxxxBWxxBWW

Positions evaluated by static estimation: 5747

MINIMAX estimate: 9

* Midgame/Endgame phase of the game:

Input:

Depth:

* 1. MiniMaxGame:

Board Position: BWBWBWBWBWWWWxWBxxxWBxB

Positions evaluated by static estimation: 48

MINIMAX estimate: 1981

* 1. MiniMaxGameImproved:

Board Position: BWBWBWBWBWWWWxWBxBxWBxx

Positions evaluated by static estimation: 701

MINIMAX estimate: 5996

**From the above results we can conclude that:**

1. Alpha-Beta Pruning Algorithm evaluates less nodes compared to MiniMax Algorithm.

2. The improvised version of the static estimation function optimizes MiniMax Algorithm for better performance.