Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Trabalho Prático
Inteligência Artificial

Integrantes: Rander Gabriel de Oliveira Rodrigues - 2017014561
Renato Sérgio Lopes Júnior - 2016006875

Disciplina: Desenvolvimento de Jogos Digitais

Belo Horizonte
Novembro/2018

Introdução

Neste trabalho foi desenvolvido um programa para jogar o jogo Reversi. Este programa conta com uma inteligência artificial que joga contra o jogador humano. O Reversi é um jogo de tabuleiro no qual dois jogadores (branco e preto) tentam preencher o maior número de posições com suas peças.

Implementação

O trabalho foi implementado na linguagem de programação Java. Em seguida, há uma descrição de cada classe que compõe o programa.

GameController: Classe que irá conectar os outros componentes do jogo, como a IA e a GUI. Ela irá inicializar esses componentes e chamar os métodos deles quando necessário.

GameBoard: Classe que representa o tabuleiro do Jogo. Além disso, essa classe possui os métodos que lidam com a movimentação das peças e as regras do jogo.

AI: Classe que contém os métodos relacionados à inteligência artificial do jogo. Mais informações sobre a implementação da IA na seção seguinte.

GUI: Classe que lida com a interface gráfica. Foram utilizados os componentes da biblioteca java.awt para a criação dessa interface.

Position: Classe auxiliar que representa uma posição i,j do tabuleiro. Foi criada para facilitar a passagem de parâmetro e o retorno nos outros métodos.

Reversi: Classe auxiliar que contém o método principal que, basicamente, cria uma instância do GameController e a inicializa.

Para implementar o jogo de Reversi, o tabuleiro foi representado como uma matriz. Cada posição da matriz pode estar com uma peça branca, uma peça preta ou vazia.

Os métodos mais importantes para o jogo são canMoveFromPosition, que verifica se pode colocar uma peça em determinada posição de acordo com as regras do jogo e makeMove, que coloca a peça em determinada posição e atualiza o tabulerio. No primeiro, para verificar se uma peça pode ser colocada em determinada posição é verificado se na vizinhança, em alguma direção, há pelo menos uma peça da outra cor e em seguida uma peça da mesma cor, em sequência. Se não houver nenhuma, então a peça não pode ser colocada naquela posição. No segundo, para atualizar o tabuleiro é verificado em todas as 8 direções da vizinhança da peça se há peças que devem mudar de cor.

Inteligência Artificial

Para implementar a Inteligência Artificial, foi utilizado o Algoritmo Minimax e a estratégia da Poda Alfa-Beta. Esse algoritmo e essa estratégia são apresentados em pseudo-código a seguir.

```
function ALPHA-BETA-SEARCH(state) returns an action
    inputs: state, current state in game
    v ← MAX-VALUE(state, −∞, +∞)
    return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
    inputs: state, current state in game
            α, the value of the best alternative for MAX along the path to state
            β, the value of the best alternative for MIN along the path to state
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s, α, β))
        if v ≥ β then return v
        α ← MAX(α, v)
    return v
```

```
function MIN-VALUE(state, α, β) returns a utility value
    inputs: state, current state in game
            α, the value of the best alternative for MAX along the path to state
            β, the value of the best alternative for MIN along the path to state
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← +∞
    for a, s in SUCCESSORS(state) do
        v ← MIN(v, MAX-VALUE(s, α, β))
        if v ≤ α then return v
        β ← MIN(β, v)
    return v
```

O método correspondente à função Max-Value é o maxValue da classe AI. O correspondente à Min-Value é o minValue, da mesma classe. Já o método equivalente à função Alpha-Beta-Search é o computeNextMove, que retorna a posição que a IA irá jogar.

O estado é dado pelo tabuleiro, que é uma matriz NxN, onde N é o tamanho do lado do tabuleiro (Neste trabalho N=8). Para representar -∞ e +∞, foram usadas as constantes da

classe Double, Double.NEGATIVE_INFINITY e Double.POSITIVE_INFINITY, respectivamente.

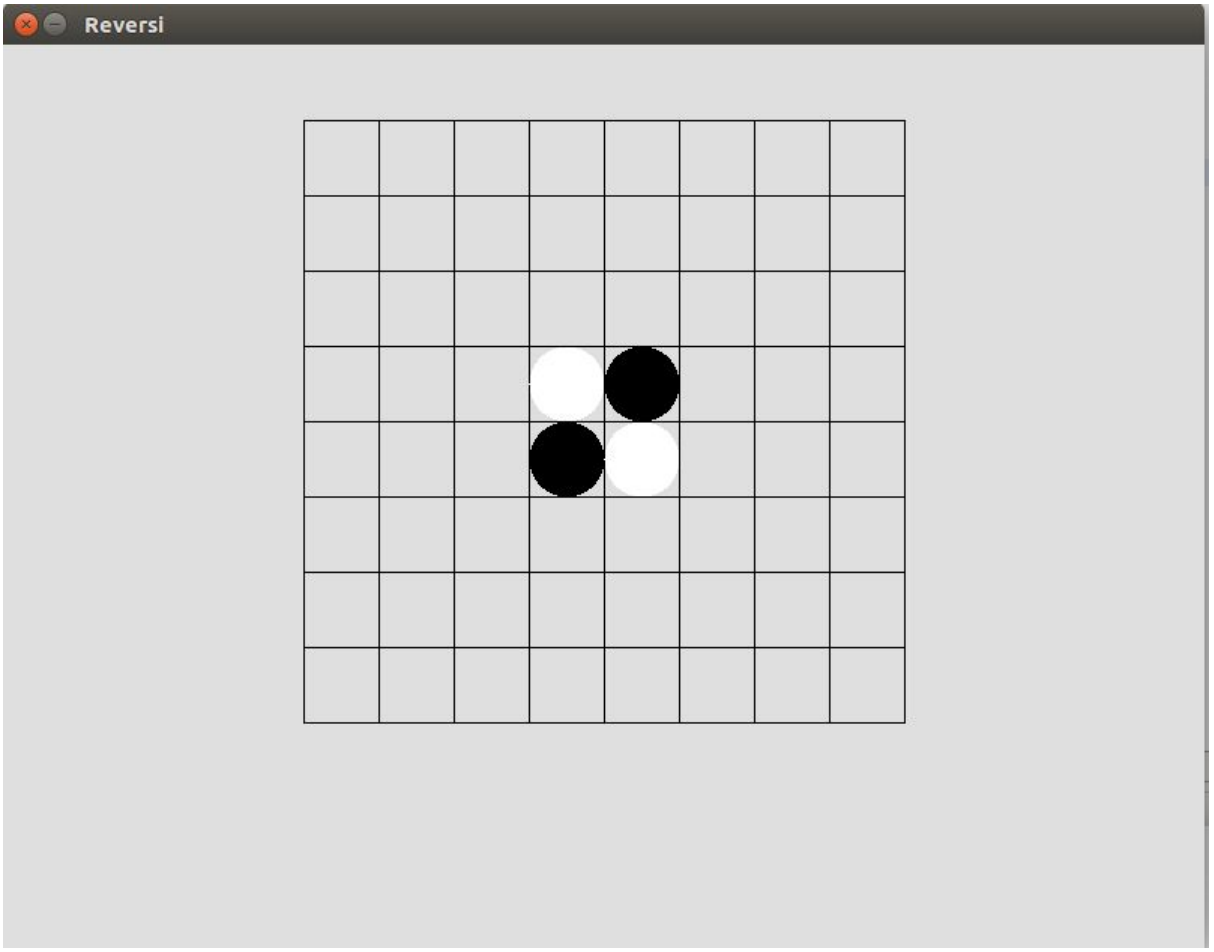
A função de avaliação é implementada pelo método evaluateGameBoard: a utilidade é dada pelo número de peças da cor da IA (Neste trabalho, peças brancas) subtraído pelo número de peças da cor do jogador (Neste trabalho, peças pretas). A IA sempre irá buscar maximizar esse valor, ou seja, aumentar a quantidade das suas peças com relação a quantidade das peças do adversário.

A profundidade máxima da busca é dada pela constante MAX_TREE_DEPTH. Esse valor foi fixado em 4, após a realização de alguns testes.

Também foi definido um tempo máximo para a IA executar, que é dado pelo atributo timeout. Quando esse tempo limite é ultrapassado, a IA para de fazer mais buscas na árvore e retorna a melhor opção já encontrada até o momento.

Interface Gráfica

Foi implementada uma interface gráfica simples para o usuário poder interagir com o jogo. O tabuleiro é apresentado e basta clicar onde se deseja colocar a peça. Caso a posição seja válida, a IA joga e o jogo continua. Se a posição for inválida, é exibido um quadrado vermelho e o jogador deve escolher uma outra posição. O usuário joga com as peças pretas enquanto a IA, com as brancas.



Execução

Para compilar o trabalho e gerar o .jar, execute o seguinte comando em um terminal na pasta src:
$ make

Para executar o arquivo .jar, execute o seguinte comando em um terminal:
$ java -jar reversi.jar

Conclusão

Assim, com este trabalho foi possível construir um jogo funcional de Reversi. A IA funcionou da maneira esperada. Algumas dificuldades foram encontradas durante o desenvolvimento da IA e da GUI, mas foram solucionadas após pesquisa nas referências bibliográficas.

Referências Bibliográficas

https://en.wikipedia.org/wiki/Reversi. Acesso em 15/11/2018.
https://othello-reversi.com/play/index.html. Acesso em 15/11/2018.
https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/. Acesso em 15/11/2018.
https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning. Acesso em 15/11/2018.
Slides disponibilizados na turma no Moodle Minha UFMG Virtual.