Universidade do Porto
Faculdade de Engenharia
FEUP

Pesquisa com Adversários: Jogo de Tabuleiro – Hex

Relatório Intercalar

Inteligência Artificial
3º ano do Mestrado Integrado em Engenharia Informática e Computação

Elementos do Grupo:
Carlos Frias – 040509116 – ei04116@fe.up.pt
José Semedo – 070509178 – ei07178@fe.up.pt

Porto, 11 de Abril de 2011

1. Objetivos:

O presente relatório pretende documentar o desenvolvimento, até ao momento, do projeto da cadeira de Inteligência Artificial do curso Mestrado Integrado em Engenharia Informática e Computação na Faculdade de Engenharia da Universidade do Porto.

O objetivo do trabalho é desenvolver uma aplicação que simule o jogo de tabuleiro Hex, jogado com dois jogadores. A aplicação permite jogar humano contra humano, humano contra computador e computador contra computador, com três níveis de dificuldade: fácil, médio e difícil.

Na implementação do trabalho serão usadas as seguintes técnicas de Inteligência Artificial: o algoritmo minimax, o algoritmo minimax com cortes alfa-beta e variações destes.

2. Descrição:

2.1. Especificação:

O jogo a desenvolver é o jogo de tabuleiro Hex que é jogado num tabuleiro tipicamente 11x11, podendo haver as variações 13x13 e 19x19, com uma grelha hexagonal.

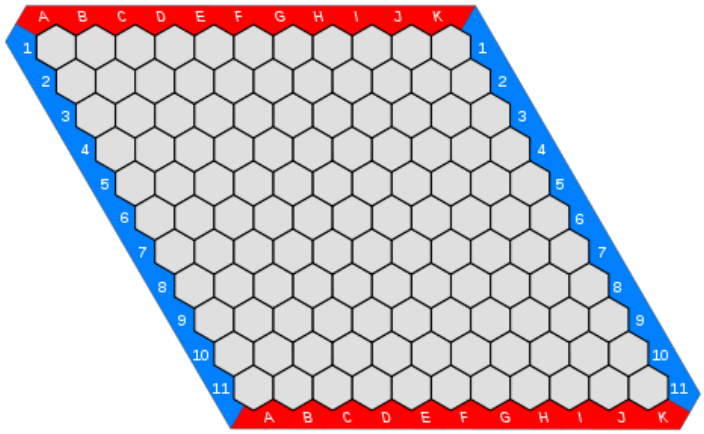

Figura 1: Tabuleiro de jogo 11x11

A cada jogador é atribuída uma cor, branca ou preta. O jogadores colocam alternadamente uma peça da sua cor numa das posições ainda não ocupadas do tabuleiro. O objetivo é formar um caminho de peças da sua cor que ligue dois lados opostos do tabuleiro. O primeiro jogador a completar o caminho ganha o jogo. As quatro casas dos cantos do tabuleiro pertencem a ambos os lados adjacentes. Como o primeiro jogador tem uma distinta vantagem, o segundo jogador pode escolher se quer ou não trocar de posições com o primeiro jogador após este ter feito a primeira jogada.

A aplicação permite as seguintes variedades de jogo: jogador humano vs jogador humano,

jogador humano vs jogador computador e jogador computador vs jogador computador, com três níveis de dificuldade: fácil, médio e difícil.

A aplicação é implementada em JAVA, e terá uma interface gráfica 2D com utilização da API swing do JAVA.

Os algoritmos usados são o minimax, minimax com cortes alfa-beta e variações destes com diferentes funções de avaliação (heurísticas).

Para o nível de dificuldade fácil, o jogador computador irá calcular a sua jogada com o algoritmo minimax com cortes alfa-beta, mas ocasionalmente irá ocorrer uma jogada ao acaso, simulando uma jogada errada ou mal calculada, para criar um certo grau de facilidade ao nível.

No nível de dificuldade médio, o jogador computador irá calcular as suas jogadas baseando-se unicamente no algoritmo de pesquisa com adversários minimax com cortes alfa-beta, com uma razoável função de avaliação.

Para o nível de dificuldade difícil será otimizada a função heurística para o cálculo das jogadas do jogador computador.

Ainda não estudamos a fundo o problema da função heurística, mas parece-nos que devem ser tidos em conta os seguintes fatores:

- número mínimo de casas (em linha reta, horizontal ou vertical) a preencher para completar o caminho;
- número mínimo de casas (em linha reta, horizontal ou vertical) a preencher para o adversário completar o caminho;
- a eventual existência de pontes, isto é, caso uma jogada ligue dois caminhos e promova um menor número de casas a preencher para completar o caminho.

Em termos de representação de conhecimento iremos utilizar um array de inteiros de dimensão dois para representar o tabuleiro em cada um dos seus estados de jogo. Cada elemento do array, em cada instante, apenas pode ter os valores 0, 1 ou 2: o valor 0 representa uma casa vazia, o valor 1 representa uma casa preenchida com uma peça do jogador um e o valor 2 representa uma casa preenchida por uma peça do jogador dois. Os estados do jogo serão calculados pelas possíveis jogadas do respetivo jogador a cada momento. Como inicialmente o tabuleiro está vazio, significa que na representação da árvore de pesquisa irá haver uma enorme ramificação nos níveis superiores, pelo que a boa implementação do algoritmo minimax com cortes alfa-beta é fulcral para eliminar ramos da árvore que não conduzem a uma solução ótima. Uma pesquisa em largura neste caso em concreto está fora de questão.

2.2. Trabalho realizado:

Neste momento temos implementada a interface gráfica da aplicação, no entanto iremos efetuar alguns melhoramentos.

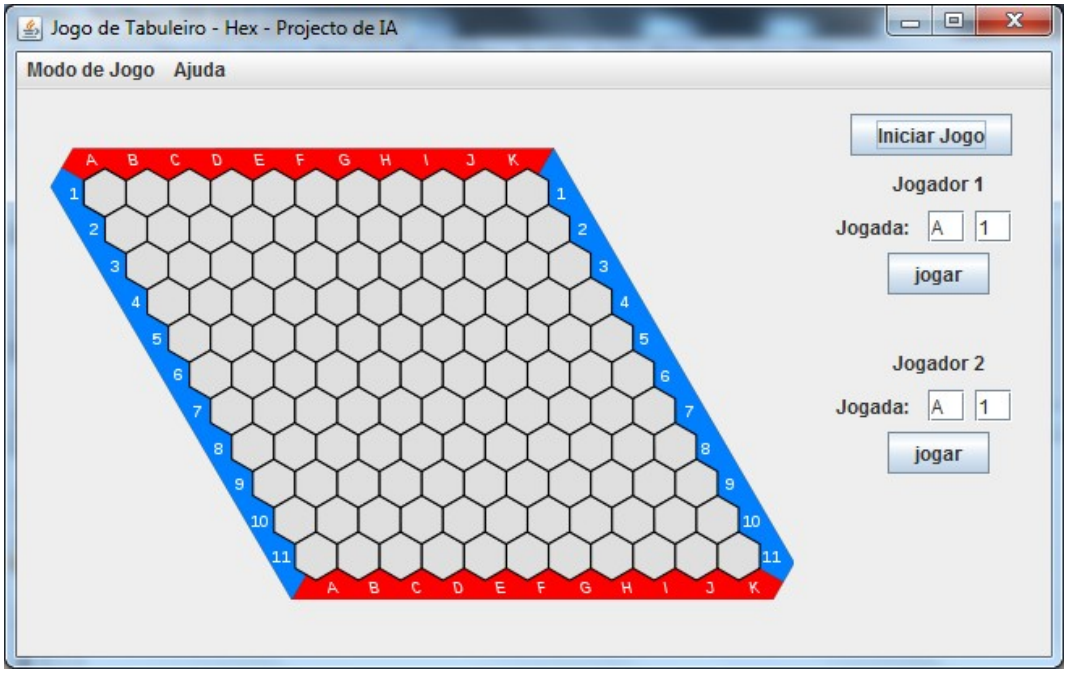

Figura 2: Janela de visualização de jogo

Os melhoramentos a realizar prendem-se com a jogabilidade, seria conveniente ao jogador poder arrastar uma das suas peças para a posição do tabuleiro com o rato (tipo drag and drop), em vez de digitar as coordenadas da sua jogada.

Falta ainda implementar as jogadas, isto é, ir colocando as peças brancas ou pretas nas posições do tabuleiro assim que seja realizada a jogada.

A janela de visualização apresenta uma barra de menus, onde se pode escolher o modo de jogo e grau de dificuldade.

Figura 3: Menu Modo de Jogo

No menu ajuda são apresentadas as regras do jogo e as informações sobre a aplicação desenvolvida.

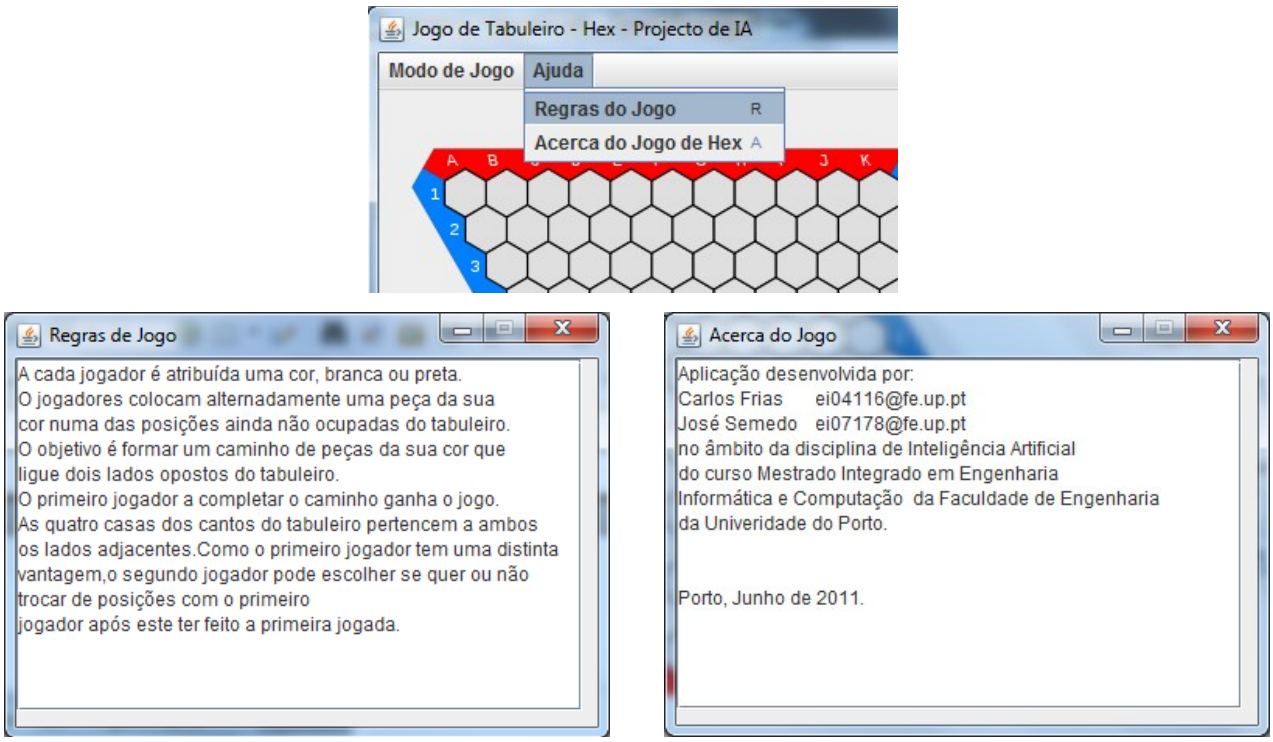

Figura 4: Menu Ajuda e Janelas: Regras de Jogo e Acerca do Jogo.

Para além da interface gráfica estamos a desenvolver a classe Tabuleiro que cada sua instância representa um dos estados do jogo, nessa classe estarão os métodos que permitem validar jogadas, alterando o array de inteiros que representa o tabuleiro, os métodos que permitem determinar os caminhos feitos a dado momento e os métodos para calcular os valores da função avaliação (heurística) para esse estado do jogo. Existe um método toString() que é utilizado para representar em modo de texto o tabuleiro a dado momento, este método é utilizado principalmente em fase de testes, para compreender melhor a implementação do código e verificar/corrigir erros ao longo da realização do trabalho.

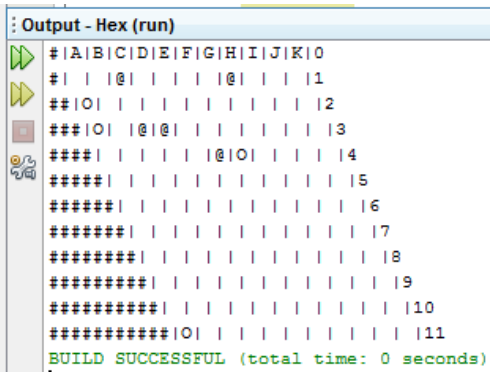

Figura 5: Representação do tabuleiro de jogo em modo de texto após nove jogadas.

2.3. Resultados esperados e forma de avaliação;

Até ao momento debruça-mo-nos principalmente pela interface gráfica, a qual fomos testando até obter um resultado aceitável, se bem que ainda não o pretendido.

Quanto à implementação interna da aplicação apenas fizemos alguns métodos que fomos testando e vendo os resultados em modo de texto no output do editor de JAVA. A validação das jogadas está implementada. Estamos a desenvolver o método "as_path" que a cada momento verifica se existe caminho entre duas posições do tabuleiro, este método recursivo será útil no futuro.

Relativamente à implementação dos algoritmos minimax e minimax com cortes alfa-beta ainda não começamos a implementar, apenas pensamos empiricamente em possíveis funções de avaliação.

3. Conclusões:

Consideramos útil a elaboração do trabalho na medida em que possibilita a familiarização com técnicas de Inteligência Artificial, nomeadamente com o algoritmo de pesquisa com adversários minimax, o qual já conhecia-mos ao de leve da cadeira de Programação em Lógica.

Em virtude de termos tido algum trabalho em outras cadeiras do curso ainda não demos a atenção devida nem o tempo necessário para um maior desenvolvimento do projeto até ao momento.

4. Recursos:

- Artificial Intelligence - A Modern Approach 3rd ed - S. Russell, P. Norvig (Prentice-Hall, 2010);
- Inteligência Artificial, apontamentos da cadeira, disponível em: http://paginas.fe.up.pt/~eol/IA/ia1011.html. Último acesso em 10 de Abril de 2011;
- Hex (board game), disponível em: http://en.wikipedia.org/wiki/Hex_%28board_game%29. Último acesso em 10 de Abril de 2011;
- NetBeans IDE 6.9.1, disponível em: http://netbeans.org/;
- Java SE Development Kit 6u24, disponível em: http://www.oracle.com/.