Faculdade de Engenharia da Universidade do Porto
Mestrado Integrado em Engenharia Informática e Computação
Unidade Curricular de Inteligência Artificial

Implementação do Nine Men's Morris com um jogador inteligente - Relatório Final

Universidade do Porto

Faculdade de Engenharia

FEUP

André Freitas ei10036@fe.up.pt
Mário Aguiar ei10108@fe.up.pt
Daniel Nora ei10030@fe.up.pt

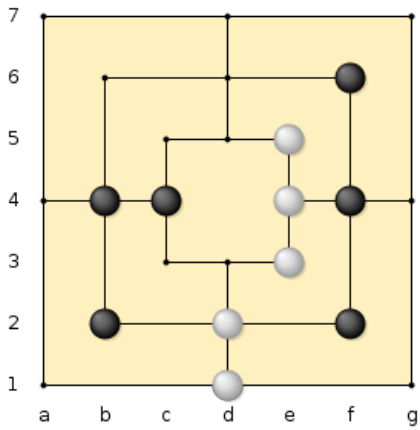
20 de Maio de 2013

Objectivo

Neste trabalho pretendeu-se aplicar conhecimentos sobre algoritmos de pesquisa de adversários no contexto de um jogo de tabuleiro chamado Nine Men's Morris, implementando um jogador automático inteligente capaz de computar as jogadas mais adequadas à situação de jogo. O algoritmo de pesquisa utilizado foi o Mini-Max com cortes alfa e beta.

Especificação do jogo

O Nine Men's Morris é um jogo de tabuleiro onde o objetivo é comer as peças dos adversários quando se consegue colocar três peças na horizontal ou vertical. Existe dois jogadores tendo cada um deles 9 peças. Só se pode pôr peças no tabuleiro em determinadas posições, que são ilustradas na seguinte imagem.



Na situação da imagem acima, como as peças brancas conseguiram um "três vertical", podem comer uma peça preta do adversário. Note-se que neste exemplo existem um "três vertical" tanto para as brancas como para as pretas, o que nunca ocorreria numa situação de jogo, tratando-se então apenas de um exemplo meramente ilustrativo.

O objetivo do jogo é fazer com que o adversário fique reduzido a duas peças ou fique sem jogadas possíveis. Existem padrões e estratégias que garantem a vitória ficando o adversário impossibilitado de dar a volta tal como o *CheckMate* do Xadrez.

Fases de Jogo

O jogo possui 3 fases que ilustram o tipo de jogadas possíveis:

1. A primeira fase é onde se coloca novas peças no tabuleiro. Se colocarmos uma peça e formamos uma linha horizontal ou vertical com mais duas peças, forma-se uma *mill;*
2. Estando esgotadas as peças para colocar no tabuleiro, a segunda fase consiste em mover peças apenas para as posições adjacentes;
3. A terceira fase aplica-se aos jogadores que tiverem apenas três peças, sendo permitido mover a peça para qualquer sítio (mesmo que não seja adjacente).

Descrição

Funcionalidades

A aplicação permite fazer uma partida do jogo Nine Men's Morris, selecionando de entre vários tipos de jogo e modos de dificuldade:

- Modos de jogo: humano-humano, humano-computador e computador-computador.
- Níveis de jogo: fácil, médio e difícil.

Módulos

- **init** - ponto de entrada na aplicação, contendo o método main e fazendo as inicializações necessárias;

- **logic** - package responsável pela representação do estado do jogo (tabuleiro, peças e jogadores). Aqui é também validada a interação do jogador com o tabuleiro. As funções de avaliação e de paragem usadas no Mini-Max também fazem parte deste package, bem como as heurísticas, que são listadas de seguida:
  - Vitoria numa jogada.
  - Nº de peças.
  - Nº de mills possiveis numa jogada.
  - Nº de formações do tipo em que se pode movimentar uma peça entre duas mills, removendo uma peça adversária a cada turno.

- **algorithms** - implementação do algoritmo Mini-Max com cortes alfa e beta no contexto do jogo;

- **graphical** - interface gráfica do programa, geração das jogadas e movimentação das peças do jogador automático, arrastamento de peças pelo rato, etc;

- **images** - gráficos usados ao longo do programa.

Representação do estado do jogo

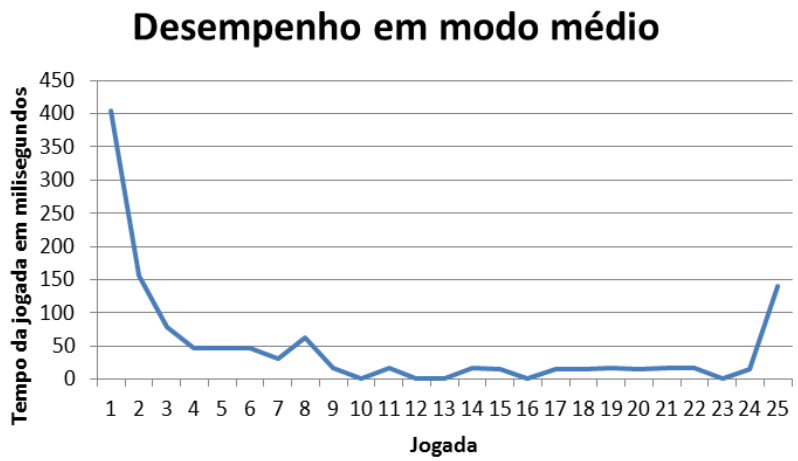
O tabuleiro está representado por uma HashMap de posições do tabuleiro, em que a chave é uma String representando as coordenadas da posição, e o valor é um objeto Piece que representa uma peça ou null caso a posição esteja vazia.

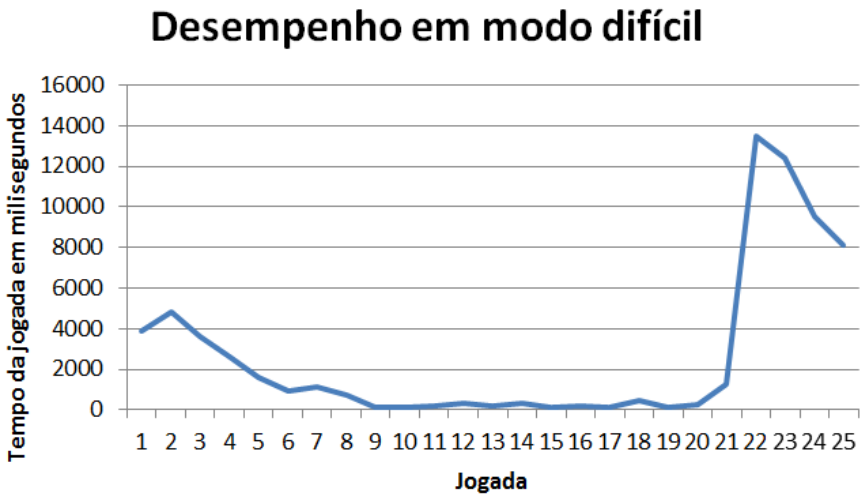
Para optimizar o desempenho da função de avaliação são criadas duas HashMaps auxiliares no início do jogo representando as posições adjacentes e todas as mills.

Análise da complexidade

A utilização de memória durante a computação das jogadas do CPU é constante, na medida que um aumento na profundidade da árvore de pesquisa não implica um aumento na utilização de memória. Em vez de guardar o estado de jogo em cada nó da árvore a jogada é feita e desfeita conforme os avanços e recuos na árvore de pesquisa, executando as jogadas sempre sobre o mesmo tabuleiro. A única informação guardada adicionalmente por cada avanço na árvore são os dados indispensáveis sobre uma jogada (posição inicial, final e peças envolvidas).

Quanto ao tempo gasto por jogada, podemos ver a sua evolução ao longo do jogo de acordo com os modos de dificuldade escolhida nas seguintes figuras:





A complexidade do método *getPossibleMoves()* (computação de todas as jogadas possíveis), é dependente da fase de jogo. Na primeira e na terceira fases a complexidade temporal é linearmente proporcional ao número de espaços livres do tabuleiro (por esse motivo, como podemos ver pelos gráficos, o tempo da jogada é maior no início e fim do jogo). Na segunda fase a complexidade temporal é proporcional ao número de casas adjacentes livres para todas as peças de um jogador. A complexidade para uma determinada jogada é sempre acrescida se resultar na formação de um mill visto ser necessário remover uma peça do adversário do tabuleiro (calculando a opção mais vantajosa para a peça a remover).

Ambiente de desenvolvimento

As especificações da principal máquina na qual foram realizados testes de desempenho são as seguintes:

- Processador: Intel Pentium Dual-Core T4200 2GHz
- Memória: 4GB
- OS: Windows 7 32 bit

Os recursos utilizados podem ser encontrados em anexo.

Avaliação do programa

A avaliação deste programa consistiu na execução de vários jogos entre humanos e CPU e entre dois jogadores do CPU. Foi observado que o CPU não opta pelo caminho mais direto para chegar à vitória mas sim pelo percurso que aumente mais a sua vantagem em relação ao adversário, como era de se esperar por ser usado o algoritmo Mini-Max. Em todos os testes com bots de dificuldades diferentes, foi observado que o bot de dificuldade maior ganha sempre.

Conclusões

O grupo considera muito satisfatório o desempenho do jogador automático, tanto no que toca à sua inteligência como à sua rapidez. Como tal, consideram-se plenamente cumpridos os objetivos estabelecidos.

De forma a maximizar a inteligência do jogador automático poderia ter-se explorado alternativas na sua implementação, tais como a utilização de algoritmos genéticos. Em termos de funcionalidade do ponto de vista do utilizador, teria sido interessante implementar uma versão do jogo com funcionamento em rede, embora tal seja fora do propósito deste projeto.

Recursos

Software

Foi utilizado o Eclipse para o desenvolvimento do projeto em Java 1.7 e o GIT para controlo das versões do código. O plugin WindowBuilder foi usado como auxiliar na construção da interface gráfica.

Referências

1. “Nine Men’s Morris”, Wikipédia, acedido a 9 de Abril, http://en.wikipedia.org/wiki/Nine_Men's_Morris
2. “Minimax with Alpha Beta Pruning“, UCLA Computer Science, acedido a 11 de Março, http://cs.ucla.edu/~rosen/161/notes/alphabeta.html

Anexos

Manual do utilizador

Selecione um modo de jogo da lista localizada no menu. Se selecionar o modo de jogo Humano vs CPU pode escolher entre os modos fácil, médio e difícil. Se selecionar o modo CPU vs CPU, pode selecionar o grau de dificuldade de cada jogador automático individualmente.

Durante o jogo, siga as instruções escritas no ecrã. Quando for o seu turno, conforme a fase de jogo, arraste a peça da sua cor para uma posição livre do tabuleiro ou uma peça do tabuleiro para outra localização. Quando lhe for dada a possibilidade de eliminar uma peça do adversário, basta que clique em cima da peça desejada.

Exemplo de uma execução

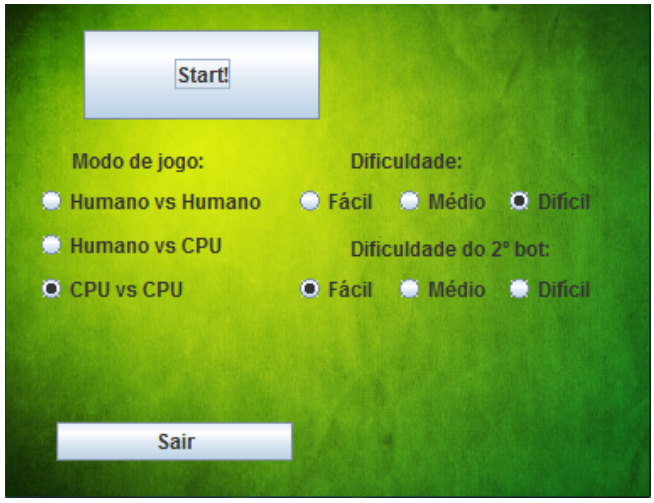


Figura 1: Menu do jogo

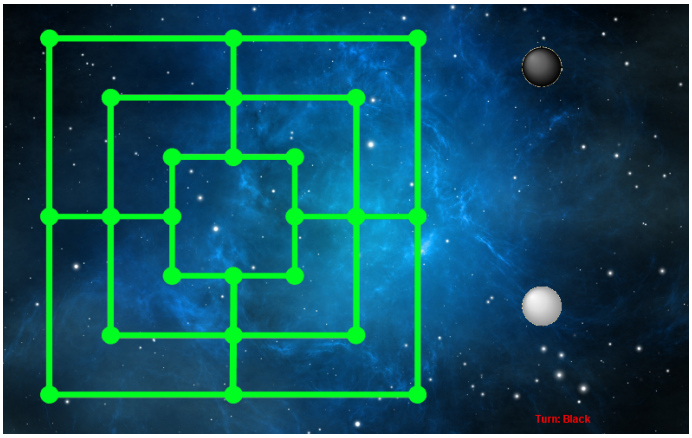


Figura 2: Início de jogo, tabuleiro vazio

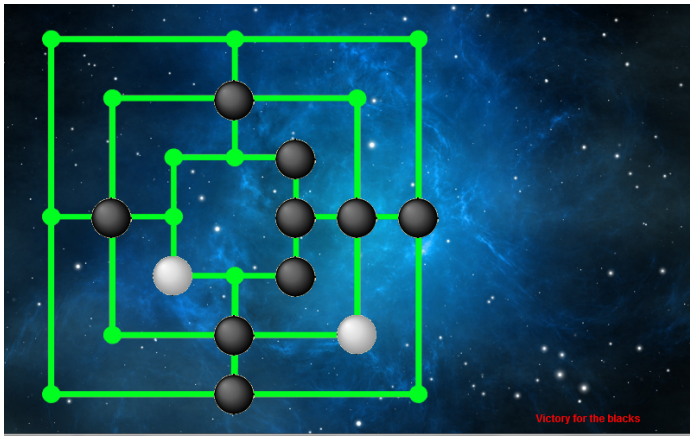


Figura 3: Final de jogo, vencem as pretas