IPL
escola superior
de tecnologia e gestão
instituto politécnico
de leiria

Inteligência Artificial

Engenharia Informática

2.º ano, 2.º Semestre 2010/2011

Nine Men's Morris

Relatório do Projecto

Documento elaborado por:
Cláudio Esperança – aluno n.º 2070030
Diogo Serra – aluno n.º 2081008

Docente:
Carlos Grilo

Versão 1.0.34
11 de Junho de 2011

Índice de conteúdos

Índice de imagens

Índice de tabelas

1. Introdução

Este documento apresenta o relatório de implementação de uma aplicação capaz de jogar o jogo *Nine Men's Morris*, também conhecido como jogo do Moinho ou Trilha, no âmbito do projeto da unidade curricular de Inteligência Artificial do curso de Engenharia Informática.

1.1. Estrutura do documento

Este documento encontra-se estruturado em diversos capítulos que apresentamos de seguida:

- Apresentação do Jogo - descrição geral das regras e caraterísticas do jogo do Moinho.
- Implementação do Jogo - descrição da aplicação desenvolvida e do modo de implementação da mesma. São abordadas questões como a Interface gráfica, Interface gráfica, Operadores, Função de avaliação de estado, entre outras
- Implementação do “modo de treino” - descrição do modo de treino desenvolvido na aplicação para testes de coeficientes de avaliação. São abordadas questões como a Interface gráfica, bem como os algoritmos utilizados.
- Conclusões - neste capítulo serão apresentadas as conclusões do projeto.

1.2. Definições e acrónimos

Para uma melhor compreensão do documento, aconselhamos a memorização de alguns acrónimos importantes:

Acrónimo	Designação
UC	Unidade Curricular
IPL	Instituto Politécnico de Leiria
ESTG	Escola Superior de Tecnologia e Gestão de Leiria
EI	Curso de Engenharia Informática
DEI	Departamento de Engenharia Informática
IA	Unidade curricular de Inteligência Artificial
SQL	*Structured Query Language*
JVM	*Java Virtual Machine*
JDK	*Java Development Kit*
JRE	*Java Runtime Environment*
MVC	*Model View Controller*
DB	*Database*
GUI	*Graphical User Interface*
SGBD	Sistema de Gestão e Bases de Dados

Tabela 1: Acrónimos utilizados

2. Apresentação do Jogo

O Moinho é um antigo jogo de tabuleiro com três quadrados concêntricos ligados entre si, formando um total de 24 posições de jogo possíveis onde cada um dos dois jogadores do jogo pode colocar um total de 9 peças.

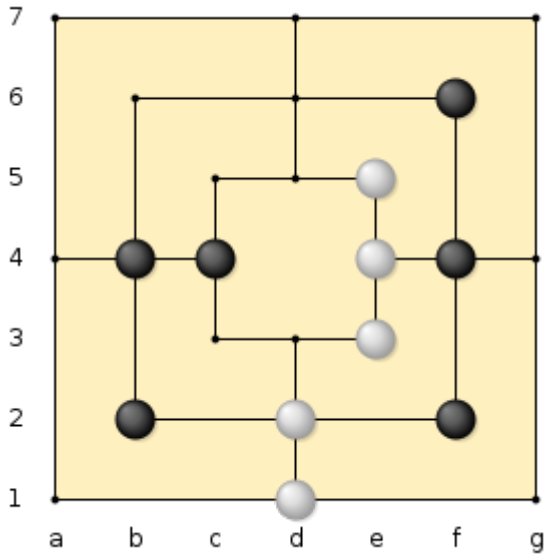

Imagem 1: Tabuleiro do jogo do Moinho

O jogo processa-se de forma alternada e objetivo é conseguir dispor as peças de modo a impedir que o jogador adversário consiga fazer três em linha. Quando o número de peças de um jogador for inferior a três, ou este não conseguir mover as suas peças, o jogo termina com a vitória para o seu adversário. Sempre que um jogador coloca três peças em linha ganha o direito de comer uma peça ao adversário, sendo que a escolha da peça a ser comida deve ser feita preferencialmente pelas peças livres que não façam parte de nenhuma linha de três, podendo estas últimas ser escolhidas em caso da inexistência das primeiras. Depois de uma peça ter sido comida, esta não pode voltar novamente ao jogo.

Existem três fases distintas no jogo:

1. Colocação das peças – onde os jogadores podem distribuir alternadamente as suas 9 peças pelas posições livres que considerem mais promissoras para o seu jogo. Se um jogador colocar 3 peças em linha nesta fase, pode comer uma peça do adversário. Ao colocar as 9 peças, o jogador passa para fase seguinte.

2. Movimentação limitada das peças – nesta fase o jogador pode mover as suas peças ao longo das linhas do tabuleiro para uma posição livre que se encontre à distância de uma posição. Ao fazer 3 em linha, o jogador ganha o direito de comer uma peça do adversário, tal como já foi referido anteriormente. Quando um jogador fica apenas com 3 peças em jogo, passa para a fase seguinte.

3. Movimentação livre das peças – nesta fase as peças do jogador podem mover-se livremente pelas posições livres no tabuleiro de jogo. Se o adversário fizer 3 em linha, uma das peças do jogador será comida e o jogador perderá o jogo. Por sua vez, se o jogador fizer 3 em linha, à semelhança do que acontece nas outras fases, o jogador poderá comer uma das peças do adversário.

3. Implementação do Jogo

A implementação do jogo foi feita em Java e baseou-se nos ficheiros fornecidos pelos docentes da UC. O código fonte fornecido disponibilizava também uma estrutura de dados para representação de um tabuleiro do jogo Moinho, o interface gráfico para a apresentação do mesmo e os controladores necessários para interação com a aplicação.

O projeto consistiu no desenvolvimento da lógica do jogo, integração dos algoritmos de análise e de decisão, que permitiram o desenvolvimento de uma inteligência artificial capaz de jogar o Jogo do Moinho.

Os algoritmos implementados para a resolução do jogo foram o algoritmo AlfaBeta e o MiniMax.

3.1. Interface gráfica

A interface gráfica apresentada na Imagem 2 foi baseada no código fonte fornecido com algumas modificações no sentido de enriquecer a aplicação com as caraterísticas que se acharam pertinentes.

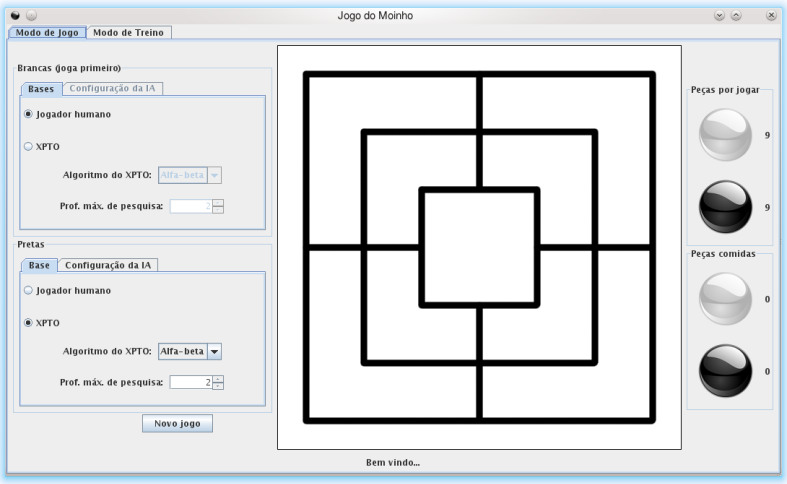

Imagem 2: Interface gráfica da aplicação

Para além dos elementos base fornecidos, tal como a tabela de suporte ao tabuleiro do jogo e os painéis laterais com a seleção do jogador e das suas caraterísticas (algoritmo e profundidade), foram adicionados os seguintes elementos gráficos:

- Separadores “Configuração da IA” – separadores que permitem especificar os coeficientes das funções da avaliação que serão abordados no capítulo Função de avaliação de estado.
- Painel Peças – elemento que contem as peças a colocar no tabuleiro e as peças do adversário comidas
- Barra de estado – onde são apresentadas algumas mensagens informativas sobre o estado do jogo
- Separador “Modo de Treino” – elemento que disponibiliza os elementos gráficos referentes ao modo de treino que será abordado no capítulo Implementação do “modo de treino”.

A interação com a aplicação é feia com recurso exclusivo ao clique para controlar a colocação ou seleção de peças no jogo.

Para colocar uma peça no tabuleiro o jogador deve pressionar sob a posição livre pretendida; caso a jogada seja válida a aplicação irá colocar a peça do jogador

nessa posição e, consoante o resultado da jogada, passar a vez ao jogador seguinte ou, em caso de três em linha, permitir que o utilizador escolha uma peça do adversário para ser retirada do jogo.

Para mover uma peça o jogador deve primeiro clicar sob a peça a mover e clicar novamente sob a posição para onde pretende mover a peça; se a jogada for válida, a peça será movida. Caso o clique seja numa posição inválida para onde a peça não possa ser movida (por estar bloqueada por outras peças), o clique sob será automaticamente ignorado, ficando a aplicação a aguardar por um clique numa posição válida do tabuleiro.

3.2. Adaptação das regras do jogo

De um modo geral, as regras do jogo implementado são as mesmas regras do jogo do Moinho que foram descritas no capítulo da Apresentação do Jogo. No entanto foi adicionada um mecanismo de deteção de ciclos que se traduz numa regra extra: sempre que num dado jogo se chegue a um tabuleiro onde a disposição das peças é exatamente igual à de um tabuleiro anterior, é considerado que o jogo está em ciclo e como tal é decretado um empate entre os dois jogadores em jogo.

Este mecanismo é importante em jogos entre dois jogadores com IA, onde em profundidades reduzidas o mecanismo de decisão considera que o melhor estado é um estado anterior pois não tem alcance suficiente para reconhecer vantagens em outro tipo de jogadas.

3.3. Estrutura do código-fonte

O código-fonte da aplicação está dividido em *packages* que agregam as classes de acordo com as suas caraterísticas e funcionalidades:

Package	Descrição
db	Contém as classes responsáveis pela gestão e persistência da camada de dados da aplicação.
gui	Conjunto de classes que constituem o interface gráfico da aplicação
images	*Package* com os recursos gráficos da aplicação, nomeadamente as imagens que compõem o jogo
jogos	Contém as classes base que suportam a IA do jogo, nomeadamente as classes de representação de um estado genérico, algoritmos de resolução do jogo, entre outras
moinho	Classes especificas para o problema do jogo do Moínho, algumas com relação hierárquica direta com classes da *package* jogos.

Tabela 2: *Packages da aplicação*

3.4. Representação lógica do tabuleiro

O tabuleiro de jogo é representado pela classe `moinho.EstadoJogoMoinho` que por sua vez herda funcionalidades da classe `jogos.Estado`. A representação "física" do tabuleiro ficou a cargo de uma matriz de carateres 7x7, sendo que numa dada posição podem ser ocupada por 1 dos seguintes carateres:

- V, VAZIO – representa o carater espaço o que graficamente se traduz numa

casa desocupada

- X – representa o carater X ou seja, a posição está ocupada por uma peça do jogador brancas
- 0 – representa o carater 0, ou seja, a posição está ocupada por uma peça do jogador pretas

O sistema gráfico fornecido interpreta esta matriz e representa cada uma das posições numa célula de uma `JTable`, fazendo a tradução dos carateres para as respetivas imagens.

Do ponto vista lógico, houve necessidade de complementar a classe `EstadoJogoMoinho` com um sistema que permitisse saber se uma dada posição representa uma posição válida no tabuleiro do jogo. Para isso foi construída uma matriz de posições válidas que é construída e interpretada sempre que um estado é criado, permitindo saber se uma dada posição é válida ou não. Esta classe é uma das mais importantes no sistema, pois contém toda a lógica do próprio jogo, permitindo saber se uma determinada jogada é válida para o estado atual. Permite ainda a interação direta com o estado pela aplicação de operações de colocação, movimento e remoção de peças do tabuleiro. Esta classe ainda a função de avaliação que permite avaliar a qualidade do tabuleiro representado pelo estado para um determinado jogador e que será discutida com mais pormenor no capítulo 3.6.

Foi também criada uma classe `moinho.Posicao` que representa uma dada posição no tabuleiro de jogo. Esta classe tem a particularidade de poder definir relações com outras posições, sendo utilizada neste contexto para definir os seus vizinhos e assim sabermos, com relativa facilidade, se uma dada peça faz parte de um conjunto de três em linha (basta saber se a posição está preenchida com o símbolo do jogador pretendido e se sim, verificar se os vizinhos também contêm peças do jogador). Estas relações entre as posições são construídas quando é construído um novo estado.

3.5. Operadores

Os operadores são as classes base que permitem a IA do jogo analisar e interagir com o tabuleiro. Para este problema do jogo do Moinho foram criados 4 operadores:

Operador	Descrição
`moinho.OperadorMoinho`	Operador base do jogo que, apesar de não ser aplicado diretamente contém uma série de funcionalidades que outros operadores podem herdar e devem implementar
`moinho.OperadorColocarPeca`	Operador composto[1] que estende funcionalidades do `OperadorMoinho` e que permite colocar uma peça no tabuleiro, caso o estado atual o permita
`moinho.OperadorRemoverPeca`	Operador composto que estende funcionalidades do `OperadorMoinho` e que permite remover uma peça do jogador do tabuleiro, caso o estado atual o permita
`moinho.OperadorComerPeca`	Operador que estende funcionalidades do `OperadorMoinho` e que permite remover uma peça do adversário do tabuleiro, caso o estado atual o permita

Tabela 3: *Operadores do jogo*

3.6. Função de avaliação de estado

Um dos elementos mais importantes numa IA para resolução de jogos é a sua

1 Operador composto é um tipo de operador que pode aplicar outros operadores de acordo com as regras estipuladas. No caso deste operador, este pode permitir a aplicação do operador para remover peça, caso a jogada tenha colocado três peças em linha

função de avaliação. Na aplicação desenvolvida optou-se pela implementação de um sistema de avaliação mais flexível onde são utilizadas várias sub-funções de avaliação cujo peso pode ser aumentado, diminuído ou anulado em função de coeficientes que podem ser definidos pelo utilizador.

Foram criadas um total de nove sub-funções de avaliação que avaliam aspetos que vão deste o número de movimentos possíveis ao cálculo das possibilidades de três em linha que um jogador possui.

Função	Descrição
`a1_QuantosTresEmLinha`	Quantos conjuntos de três peças em linha o jogador possui no tabuleiro atual.
`a2_NumeroMovimentosPossiveis`	Número total de movimentos possíveis que as peças do jogador possuem.
`a3_NumeroPecasComMobilidade`	Número de peças que o jogador possui e que se podem movimentar no tabuleiro para, pelo menos, uma posição.
`a4_NumeroPecas`	Número total de peças que um jogador possui em jogo.
`a5_QuantosPossiveisTresEmLinha`	Quantas três em linha o jogador pode fazer na próxima jogada.
`a6_QuantasPecasEmTresEmLinha`	Quantas peças tem o jogador no total em conjuntos de três peças em linha.
`a7_tabuleiroVencedor`	Este estado é um estado vencedor.
`a8_NumeroCasasVizinhasDisponiveis`	Verifica o número de casas vizinhas a peças que o jogador já tenha em jogo e

Função	Descrição
	nas quais o jogador pode colocar peças.
`a9_QuantosPossiveisTresEmLinha`	Quantas possibilidades de três em linha no modo de colocação de peças o jogador possui.

Tabela 4: *Sub-funções de avaliação do jogo*

Estas funções são depois multiplicadas pelos coeficientes associados a cada uma das funções e adicionados a uma pontuação do jogador. As mesmas funções são depois aplicadas mas no contexto do jogo para o adversário, multiplicadas pelos respetivos coeficientes e que são depois subtraídos à função de avaliação. Isto permite que o jogador possa ver o jogo não só na perspetiva de ganhar o jogo, mas também de defender.

A definição dos coeficientes é feita através do separador “Configuração da IA” que é ativo quando se escolhe o jogador “XPTO” no separador Bases.

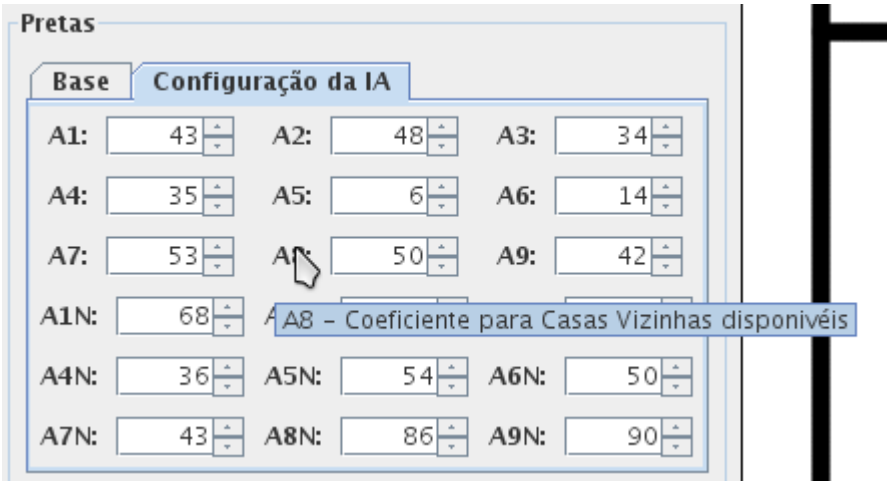

Imagem 3: Coeficientes do jogador XPTO Pretas

Os coeficientes vão de A1 a A9 para as funções que dão pontuação a um tabuleiro e de A1N a A9N a coeficientes para as funções que vão ser utilizadas no cálculo da

pontuação do ponto de vista do jogador adversário.

3.7. Outras otimizações

3.7.1. Impedimento de ciclos

Foram implementados alguns mecanismos de proteção de deteção de ciclos no sentido de evitar jogadas repetidas. Basicamente a cada nova iteração do IA é capturado o estado do tabuleiro atual e guardados numa lista de estados anteriores. Esta lista é depois utilizada para verificar se um estado posterior é igual ao atual e assim declarar o empate. Esta lista foi também utilizada dentro dos algoritmos MiniMax e AlfaBeta para evitar a análise de estados que o tabuleiro já assumiu, ganhando assim alguma vantagem no processamento de cada um dos estados sucessores.

3.7.2. Adição de novas funções de avaliação

Houve alguma preocupação no sentido de manter o sistema de avaliação o mais modular possível. Assim é relativamente simples adicionar novas sub-funções de avaliação, bastando adicionar o respetivo código na função de avaliação do estado e adicionando os coeficientes à classe `db.Peso`. Depois ao ser eliminado o ficheiro com a base de dados, esta será recriada com os novos coeficientes e o interface irá adaptar-se automaticamente a esses novos elementos.

4. Implementação do “modo de treino”

O modo de treino disponibiliza um conjunto de funcionalidades que permite a geração e teste automático de coeficientes para as funções de avaliação. No fundo é um sistema que cria e testa jogadores pela monitorização dos resultados de vários jogos entre estes jogadores.

4.1. Interface gráfica

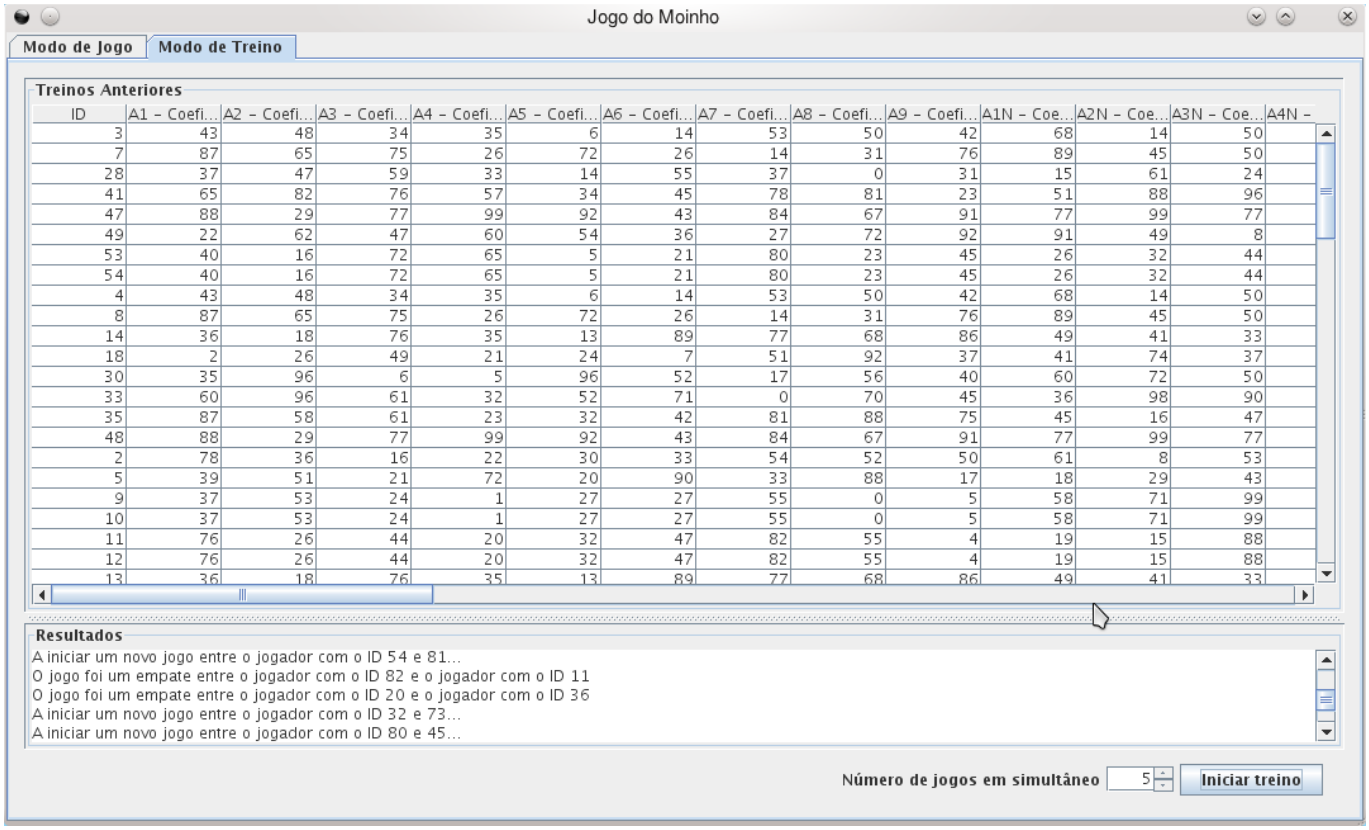

Imagem 4: Interface do modo de treino

A interface gráfica deste modo está disponível no separador “Modo de treino” e contém uma tabela com os resultados estatísticos dos jogos, uma área de texto com a informação do estado, a caixa para definição do número de jogos a executar em simultâneo e um botão para iniciar/para “treinos”.

4.2. Armazenamento e processamento

O armazenamento e processamento dos dados foi feito com base no SGBD SQLite que foi integrado na aplicação através da biblioteca sqlite-jdbc. Foi desenvolvido o código necessário para que o sistema possa criar de forma automática o ficheiro da base de dados de registo da estatística e respetivas tabelas, vistas e *triggers*. Os resultados apresentados na tabela no interface gráfico da aplicação têm como base uma vista que reúne toda a informação estatística disponível.

4.3. Algoritmos

No sentido de adicionar alguma autonomia ao mecanismo de treino e de teste dos jogos foram implementados alguns algoritmos com influências claras nos algoritmos genéticos.

A qualidade dos indivíduos é avaliada tendo como base um modo de torneio onde são escolhidos dois indivíduos de forma aleatória para disputarem um jogo entre si.

Estes algoritmos podem ser consultados na classe `moinho.Treino`.

4.3.1. Geração da população inicial

Caso não existam indivíduos suficientes na base de dados para iniciar um treino, são criados de forma automática um conjunto de indivíduos com coeficientes aleatórios. A partir deste momento é iniciado o torneio entre os diferentes

indivíduos.

4.3.2. Recombinação genética

Para criar novos indivíduos, em cerca de 30% das vitórias é selecionado o hipotético melhor jogador atual para recombinar com o indivíduo vencedor. Em cerca de 1% dos empates o jogador do treino atual é também recombinado com o melhor jogador em jogo.

A recombinação em si é uma recombinação uniforme com 50% de possibilidades de existir troca de material genético (coeficientes de funções de avaliação) entre os dois indivíduos.

4.3.3. Mutação

Em cada recombinação existe 1 possibilidade de mutação do gene em cada 1000 indivíduos. A mutação em si é uma alteração de -10 a 10% do valor do gene em recombinação.

4.4. Adição manual de indivíduos

Sempre que é executado um novo jogo onde participe um jogador com IA, as configurações do jogador são analisadas e, se o jogador não existir na base de dados de estatísticas, este é automaticamente inserido, ficando disponível na lista para ser selecionado para torneios no modo de treino.

5. Análise Crítica

5.1. MiniMax vs AlfaBeta

Com base nos resultados dos jogos que foram efetuados de forma autónoma, pode ser interessante analisar como se comportam cada um dos algoritmos implementados.

Algoritmo	Vitórias	Duração Vitórias	Derrotas	Duração Derrotas	Empates	Jogos
AlfaBeta	173(5,0%)	28	189(5,5%)	28	3073(89,5%)	3435
Minimax	132(5,0%)	34	127(4,8%)	37	2382(90,2%)	2641

Tabela 5: MiniMax vs AlfaBeta

Como se pode verificar pela tabela 5, nesta implementação existe uma elevada percentagem de empates para ambos os algoritmos (cerca de 90%). É de salientar também que nos restantes 10%, apesar da aproximação entre vitórias e derrotas em cada algoritmo, o Minimax têm mais probabilidades de vitória do que de derrota. É também interessante verificar que este algoritmo demora menos tempo a ganhar do que a perder.

Já o AlfaBeta demora o mesmo tempo a ganhar e a perder, e têm mais probabilidade de perder do que ganhar.

5.2. A questão da profundidade...

Alterando o limite de profundidade dos algoritmos conseguimos analisar ainda

melhor estes dados. Como se pode ver na tabela seguinte, com ambos os algoritmos a utilizar a profundidade 2, consegue-se sempre mais vitórias do que derrotas. Para as profundidades 1 e 3, o cenário inverte-se.

Algoritmo	Prof.	Vitórias	Duração Vitórias	Derrotas	Duração Derrotas	Empates	Jogos
AlfaBeta	1	15(2,3%)	13	28(4,3%)	18	615(93,5%)	658
AlfaBeta	2	50(4,3%)	16	33(2,9%)	13	1069(92,8%)	1152
AlfaBeta	3	108(6,6%)	36	128(7,9%)	34	1389(85,5%)	1625
Minimax	1	20(3,1%)	16	30(4,6%)	18	603(92,3%)	653
Minimax	2	50(5,2%)	15	29(3,0%)	16	888(91,8%)	967
Minimax	3	62(6,1%)	56	68(6,7%)	54	891(87,3%)	1021

Tabela 6: Estudo da profundidade

5.3. O sistema de coeficientes

Com o sistema de coeficientes ganhamos a possibilidade de atribuir pesos às funções de avaliação e assim aumentar, diminuir ou anular a importância de um dado fator na função de avaliação. Isto permite que, de forma dinâmica, se “melhore” ou “piore” os jogadores com IA e assim analisar a sua evolução.

Se por exemplo atribuirmos um valor nulo a todos os coeficientes negativos os jogadores não vão defender, ou seja, vão “testar” as jogadas que o favorecem sem olharem a se o adversário pode ou não fazer três em linha na jogada seguinte.

6. Conclusões

O projeto permitiu ter contacto com as dificuldades no desenvolvimento de uma IA que permita a resolução de problemas, neste caso, o problema do jogo do Moinho.

De um modo geral achamos que desenvolvemos um sistema interessante que permite jogar de forma autónoma o jogo do Moinho, mas acreditamos que a aplicação poderia ter beneficiado de uma análise mais aprofundada da questão da aplicação dos coeficientes, da definição de mais sub-funções de avaliação que permitissem um desempenho mais interessante da IA dos jogadores e da otimização do sistema de geração de indivíduos (com a remoção dos indivíduos menos aptos). Infelizmente o tempo não o permitiu.

Foi sem dúvida um projeto interessante e desafiante que valeu pela experiência adquirida.

7. Bibliografia

GASSER, Ralph. Solving Nine Men’s Morris. Games of No Chance, MSRI Publications Volume 29, 1996

GASSER , Ralph. Solving Nine Men’s Morris. Disponível em WWW: <http://en.scientificcommons.org/42902016>

Trilha (jogo). In: Wikipédia, a enciclopédia livre [Em linha]. Flórida: Wikimedia Foundation, 2011, rev. 2 Maio 2011. [Consultado a 11 de Junho de 2011]. Disponível em WWW: <http://pt.wikipedia.org/w/index.php?title=Trilha_(jogo)&oldid=25081415>.

Simona-Alexandra PETCU, Stefan HOLBAN. Nine Men’s Morris: Evaluation Functions. 22-24, 2008. Disponível em WWW: <www.dasconference.ro/papers/2008/**B7**.**pdf** >