Bruno Herrera
Gabriel Koji
Marcelo Honório

Projeto final de Laboratório de Programação VI - Inteligência Artificial

1. Introdução e Métodos

O jogador desenvolvido pelo nosso grupo foi o JogadorBGM, ele é uma implementação da Interface Jogador que contêm os métodos getNome() e jogada(). Foi sobrescrito esse método e adicionado alguns métodos auxiliares. Inclusive alguns que continham na classe **JogadorLinhaQuatro** como o método verificador de jogo finalizado. Esse método não estava estático e achamos anti-ético usarmos uma instância da classe **JogadorLinhaQuatro**.

2. Solução

O algoritmo utilizado pelo nosso projeto foi o MiniMax. As jogadas feitas pelo JogadorBGM são maximizados e as jogadas do adversário são minimizadas. A cada jogada é possível escolher sete colunas distintas desde que uma destas não esteja cheia. Na primeira jogada, onde a árvore se encontra em seu maior tamanho, a profundidade dela é na ordem de 49**.** A profundidade máxima adotada por nosso algoritmo é de sete. Efetuamos alguns teste com uma profundidade maior que sete e o algoritmo mostrou-se lento demais, levando mais do que 10 segundos para efetuar a jogada. Uma jogada com profundidade 8, leva 12 segundos em média para calcular a coluna. Jogadas que são feitas em mais de 10 segundos são canceladas.

2.1 Função Utilidade

A função utilidade utilizada é um heurística que retorna 0, 1 ou 2. Zero é quando nenhum jogador ganha com aquela jogada, 1 o jogador 1 ganha e 2 o jogador 2 ganha.

2.2 Métodos mais importantes

- JogadorBGM(): Construtor da classe, ele cria o vetor de movimentos e seta o tamanho da matriz para a classe.
- jogada(): método principal da classe. Ele é uma sobrescrição do método da interface Jogador. Basicamente nela, chamamos o método minimax() que tenta descobrir a melhor jogada.
- minimax: método que tenta descobrir a melhor jogada baseado em uma função utilidade que foi descrita no item **2.1**
- setColunaAltura()**:** método que cria popula um vetor com a quantidade de jogadas que já possui em todas colunas, isso é para o cálculo da função utilidade. Esse método também serve para que o minimax saiba que aquela coluna não aceita jogadas mais. Se a jogada for feito em uma coluna cheia o programa entre em loop.
- CopiaMatriz(): copia o tabuleiro e faz uma rotação de 90º para a direita. Isso facilita o cálculo do método quatroEmLinha().
- quatroEmLinha(): método que retorna a função utilidade do minimax. Ele verifica se um jogador formou quatro em linha nas colunas, linhas, diagonais esquerdas e diagonais direitia.

2.2. Poda alfa beta

Foi tentado utilizar a poda alfa beta, mas devido ao tempo curto de desenvolvimento desse projeto não conseguimos implementar a tempo uma solução que se mostrasse estável e que realmente trouxesse utilidade para o código.

3. Testes

O JogadorBGM ganhou todas as partidas realizadas com o JogadorAleatorio e JogadorAleatorioFocado. Algumas partidas nosso jogador ganhou jogando somente em uma coluna. Com o JogadorAleatorioFocado foram feitas mais jogadas mas sempre o JogadorBGM ganhou. Fizemos o teste do JogadorBGM contra o próprio JogadorBGM, foi constado que na grande maioria das partidas, quem iniciava o jogo ganhava. Algumas vezes o jogo empatava.

4. Considerações finais

Esse projeto mostrou-se muito interessante para a disciplina de Laboratório VI. A inteligência Artificial implementada no jogador é muito interessante. O que mais nos impressionou é não há nenhuma chamada a um método randômico e mesmo assim o JogadorBGM contra ele mesmo nunca dava a mesma partida. A cada nova partida era um jogo diferente.

5. Bibliografia

http://pt.wikipedia.org/wiki/Minimax
http://en.wikipedia.org/wiki/Minimax
http://www.moodle.univ-ab.pt/moodle/mod/resource/view.php?id=2221
RUSSEL, Stuart. NORVIG, Peter. Inteligência Artificial. Editora Campus. Tradução da segunda edição.