**INF 420 – Inteligência Artificial**

Relatório trabalho 3 – Implementação de Minimax para o jogo de Lig 4

Marcelo de Matos Menezes - 75254

Hevellin Ferreira Aguiar e Ferraz - 81826

Felipe de Souza Dias - 81841

**1 Introdução**

O trabalho 3 teve como objetivo a implementação do algoritmo de Minimax utilizando a heurística de Monte Carlo para jogar Lig4. Além disso, foi desenvolvida uma heurística otimizada, a partir da Monte Carlo, e implementada a técnica de poda Alpha-Beta para melhorar o desempenho do programa. Foram então feitos testes para comparação da eficiência dos algoritmos em relação ao número de vitórias alcançadas por cada um.

**2 Implementação**

Aproveitando o tema do trabalho, o grupo resolveu estender a ideia e fazer com que o jogo pudesse ser jogado entre dois computadores, jogador contra o computador, ou entre dois jogadores, opções que podem ser escolhidas no menu inicial. O trabalho em si, consiste no processamento da jogada pelo computador que é descrito a seguir.

É passado um tempo limite para o programa, que durante esse tempo utiliza a técnica de aprofundamento iterativo para percorrer os nós da árvore de busca gerada pelo algoritmo Minimax. Quando a profundidade máxima da iteração é alcançada, o programa utiliza a heurística de Monte Carlo como função de avaliação para os nós não expandidos, fazendo com que o computador opte pela melhor jogada de acordo com o tempo e o número de simulações feitas pela heurística. Além disso podem ser executadas, através de opções escolhidas pelo jogador, a técnica de poda Alpha-Beta e a utilização de uma heurística otimizada.

A compilação do código é feita pelo comando g++ main.cpp Jogada\_t.cpp caso esteja utilizando a GNU Compiler Collection;

**3 Resultados**

Foram definidas características padrão para um computador, a fim de tornar todas as comparações justas. O jogo foi executado com o computador padrão contra quatro diferentes configurações de adversários, sendo 20 execuções para cada oponente alternando-se o jogador inicial. As configurações são apresentadas na tabela a seguir:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Tempo máximo por jogada | Função heurística | Número de simulações da heurística | Utiliza poda Alpha-Beta |
| Computador Padrão | 1.5 | Monte Carlo | 50 | Não |
| Oponente teste 1 | 1 | Monte Carlo | 25 | Não |
| Oponente teste 2 | 1.5 | Monte Carlo | 50 | Não |
| Oponente teste 3 | 4 | Monte Carlo | 100 | Não |
| Oponente teste 4 | 3 | Otimizada | 50 | Sim |

\*O oponente teste 4 utiliza a heurística otimizada (Parte 3 do trabalho) que consiste numa melhoria do Monte Carlo: São verificadas as jogadas possíveis para a simulação, e caso haja uma vitória, a jogada recebe a maior prioridade. Caso contrário, se na próxima jogada o oponente consiga ganhar, o impedimento dessa jogada se torna a maior prioridade. Caso as condições acima não existam, é calculada uma jogada aleatória.

Os resultados podem ser visualizados no arquivo resultado.txt

**3 Conclusão**

De acordo com os resultados obtidos pelos testes, somado aos conhecimentos adquiridos em sala de aula, podemos concluir que a escolha de valores muito altos ou muito baixos para o número de simulações da heurística de Monte Carlo é ruim. Isso ocorre devido ao fato de que a heurística é lenta, portanto ao aplicar valores altos sobra menos tempo para que o Minimax expanda mais nós da árvore de busca, e, ao aplicar valores baixos, Minimax terá mais tempo para realizar a expansão dos nós, embora com um valor heurístico não tão bom.

Percebemos também que a heurística de Monte Carlo não é muito segura para garantir uma boa jogada visto que ela cria simulações 100% aleatórias para os nós, logo seria necessário um custo elevado de tempo para gerar boas jogadas.

A poda Alpha-Beta permite uma busca mais profunda, uma vez que ela evita a expansão desnecessária de nós da árvore. Por outro lado, ao encontrar uma derrota ela a adianta, o que é uma decisão ruim contra jogadores ruins, já que ele provavelmente não seguira por este caminho. O algoritmo de Minimax é um bom método contra bons jogares, já que ele tenta prever a jogada do adversário e supõe que o mesmo fará boas jogadas.

A heurística de Monte Carlo Otimizada por sua vez, garante a melhor jogada quando o assunto é vencer o jogo ou impedir a derrota, entretanto ela é mais lenta que a função original, fazendo com que menos nós sejam expandidos.