DEPARTAMENTO
DE CIÊNCIA DE
COMPUTADORES

MARÇO DE 2023

Relatório de Inteligência Artificial

Trabalho Prático n.º 2

REALIZADO POR

Matheus Bezerra
(202101391)
Dante Rotnes
(202105772)

Sumário

1.Introdução 3
2.Minimax 3
3.Alpha-Beta.....4
4.MCTS.....4-5
5.Jogo Connected Four.....5-6
5.Implementações.....7-9
6.Performances e conclusão.....9
7.Referências Bibliográficas 10

Todo Código para esse projeto pode ser encontrado no GitHub pelo link: https://github.com/MatheusRodriguesBezerra/IA/tree/main/trabalho2

Introdução

O objetivo deste relatório será discorrer sobre algoritmos que consigam enfrentar adversários e sejam capazes de conquistar a vitória ou algum objetivo. Os algoritmos que serão estudados serão o minimax, o AlphaBeta e o Monte Carlo Tree Search(MCTS). O jogo a ser abordado é o ConnectFour, um jogo que possui 1 adversário em que é possível vencer, perder e empatar.

Algoritmo Minimax

O objetivo deste algoritmo será escolher a melhor opção de jogada para um determinado jogo de 2 jogadores com informação completa sobre o estado do jogo. A decisão da melhor jogada é feita a partir da criação de jogadas a oriundas do estado atual, e apoiado em um valor de utilidade, escolherá qual jogada deverá ser feita. O princípio princípio base do minimax é que a jogada do adversário será a melhor possível, e a partir disto, o algoritmo tentará anular a jogada do adversário.

Algoritmo Alpha-Beta

O algoritmo Alpha-beta é uma forma de otimizar o algoritmo Minimax, evitando criar descendentes que não afetem a decisão final. Em comparação ao Minimax, este algoritmo possui 2 novas variáveis, o alpha (limite inferior) e o beta (limite superior). Durante a busca, os nós serão avaliados e comparados aos valores de alpha e beta, se o valor de um nó for inferior a alpha ou superior a beta, este nó será descartado. Este formato faz o algoritmo ser mais eficiente e retornar o mesmo resultado e com mesma profundidade do Minimax em menos tempo.

Monte Carlo Tree Search

Este algoritmo tem o objetivo de decidir qual a melhor opção de movimento/escolha e um determinado jogo. O MCTS possui como princípio simular jogadas aleatórias a partir da situação atual do jogo e a partir desta jogada, criar uma árvore de procura com jogadas possíveis. A construção desta árvore possui 4 partes:

- Seleção, será escolhido um nó que não foi ainda explorado, de forma aleatória.
- Expansão, adiciona um novo filho no nó selecionado, este novo nó será uma jogada possível.
- Simulação, utiliza este nó expandido e começa a expandi-lo de forma aleatória até alcançar um resultado final.
- Regressão, será propagado ao nó escolhido na seleção o resultado da simulação.

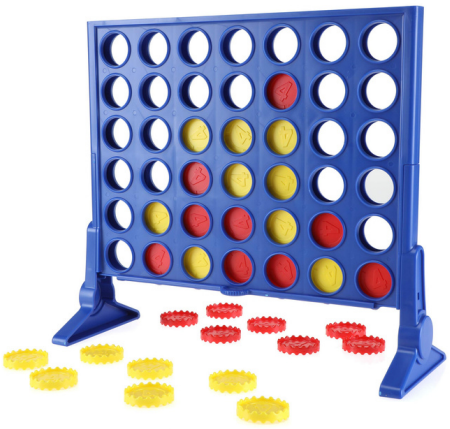
Estas 4 etapas serão repetidas até um certo limite, tempo ou memória. Ao finalizar o ciclo, o algoritmo escolherá o nó filho que possuir a melhor frequência de simulações aleatórias.

O Monte Carlo Tree Search possui certas vantagens em relação ao Minimax e Alpha-Beta, que são:

- Ideal para jogos que possuam uma grande abrangência de movimentos possíveis, como o Go!.
- Capaz de adaptar-se a jogos que não possuam informação completa, como o poker ou outros jogos de baralho.
- Pode ser preparado para jogos com mais de 1 oponente, como Monopoly ou Civilization.

Jogo Connect Four

O jogo Connect Four é um jogo de 2 jogadores que consiste em um grid com 6 linhas e 7 colunas, e o objetivo de cada jogador é formar uma sequência de 4 peças da sua cor, podendo ser na vertical, horizontal ou nas diagonais.



Este jogo possui 3 possibilidades de resultado final, a vitória de alguma das partes ou um empate. O jogo é jogado da forma alternada e é necessário que o jogador da vez faça alguma jogada.

Com o intuito de facilitar a descrição dos oponentes do jogo, haverá o jogador 'X' e o jogador 'O', e que a parir disto, será escolhido que será o BOT.

Para a criação de um valor utilitário para verificar a situação do jogo, será verificado todas as posições que podem haver vitórias, isto quer dizer que será verificado todas as sequências com 4 posições horizontais, verticais e diagonais, que de lá pode ocorrer um resultado final e depois ser adicionado no somatório para a tabela. Para cada sequência será creditado as seguintes pontuações:

-50 para: 3 'O' e sem 'X'
-10 para: 2 'O' e sem 'X'
-1 para: 1 'O' e sem 'X'
0 para: nenhum 'O' e nenhum 'X'
0 para: mais que 1 'O' e mais que 1 'X'
+1 para: 1 'X' e sem '0'
+10 para: 2 'X' e sem '0'
+50 para: 3 'X' e sem '0'

Se ocorrer uma vitória ou ocorrer um empate, deverá ser creditado no valor da tabelas as seguintes pontuações:

-512 se o 'O' vencer
0 se ocorrer empate
+512 se o 'X' vencer

A partir destes critérios será utilizado o valor utilitário das tabelas para ser avaliado a situação de jogo.

Implementações

Todo o codigo do programa foi escrito em Python. Dividimos o programa em 6 arquivos diferentes:

- main.py: Arquivo principal, que roda o programa
- connectFour.py: Possui definições para as funções básicas do jogo
- tabuleiro.py: Possui a definição da classe Tabuleiro, utilizado para criar um tabuleiro do jogo
- minimax.py: Possui a implementação do algoritmo minimax
- alphabeta.py: Possui a implementação do algoritmo alphabeta
- mcts.py: Possui a implementação do algoritmo mcts

Minimax

Para implementar o algoritmo Minimax, utilizamos como referência o algoritmo apresentado pelos professores durante as aulas. O código consiste em 4 funções:

- min_max_decision
- maxValue
- minValue
- play_min_max

Para que fosse possível retornar a jogada escolhida pelo algoritmo, implementamos no *min_max_decision* a primeira chamada do maxValue, senão seria necessário retornar tuplos em cada uma das funções e fazer a chamada de muitas comparações durante as chamadas recursivas do algoritmo.
Além disso, limitamos em profundidade 5 o algoritmo, caso contrário, o tempo de resposta seria muito elevado, tendo possibilidade de estourar o limite de recursividade também.

Alpha-Beta

Para implementar o algoritmo Alpha-Beta, utilizamos nosso algoritmo Minimax e implementamos as diferenças entre os algoritmos. O código consiste em 4 funções:

- alpha_beta_decision
- maxValue
- minValue
- play_alpha_beta

Assim como no Minimax, para que fosse possível retornar a jogada escolhida pelo algoritmo, implementamos no *alpha_beta_decision* a primeira chamada do maxValue e limitamos em profundidade 5 o algoritmo, caso contrário, o tempo de resposta seria muito elevado, tendo possibilidade de estourar o limite de recursividade também.

MCTS

Para implementar o algoritmo Mcts, utilizamos com referência o código para calcular o UCT apresentado nas aulas e, o website liberado no Moodle, referenciado ao final deste documento. Precisamos criar duas classes novas : MCTS e TreeNode. Ademais, o código consiste em 8 funções:

- expand
- is_fully_expanded
- run
- select_node
- simulate
- backpropagate
- get_best_move
- play_mcts

Para que fosse possível implementar a alternação entre os jogadores, criamos um campo na classe TreeNode que salva qual o próximo jogador que deverá jogar a partir daquele Tabuleiro (BOT ou PLAYER). Assim, o algoritmo sabe quais filhos deve criar durante a simulação.

Colocamos o número de iterações para serem feitas em 10000, pois, após vários testes, vimos que é um número que mantém o tempo de resposta relativamente curto e o algoritmo bem abrangente.

Performances e Conclusão

Para verificar a performance de cada algoritmo, precisamos calcular a média de nós gerados por chamada do algoritmo durante um jogo e, o tempo de resposta médio do algoritmo. Assim, obtivemos os seguintes resultados:

- Minimax (limitado em 5 de profundidade):
  - Tempo médio de resposta: 1.728944230079651 segundos
  - Número médio de nós criados: 17660
- Alpha-Beta (limitado em 5 de profundidade):
  - Tempo médio de resposta: 0.5635153225490025 segundos
  - Número médio de nós criados: 3691

Infelizmente não conseguimos verificar a performance do MCTS.

A partir desses dados, e das implementações e respostas de cada algoritmo, pode-se perceber como funciona cada algoritmo perante o problema do Connected Four. O Minimax e o Alpha-Beta, sempre buscam pelo melhor movimento possível a ser feito, o que, apesar de ser o desejado quando se quer um algoritmo para vencer em um jogo, não apresenta características de "pensamento como o humano". Por outro lado, o MCTS, com a certa aleatoriedade implementada, traz a ideia de replicar o pensamento humano e a aprendizagem.

Referências

Imagem Connect four:
https://www.amazon.com/Hasbro-A5640-Connect-4-Game/dp/B00D8STBHY

Algoritmo MCTS:
https://www.analyticsvidhya.com/blog/2019/01/monte-carlo-tree-search-introduction-algorithm-deepmind-alphago/