Projeto 3

Beatriz Juliato Coutinho
cc22121@g.unicamp.br

Marcos Godinho Filho
cc22142@g.unicamp.br

1 Introdução

O "Jogo da Velha"é um jogo de dupla cujo objetivo é formar uma linha de três símbolos idênticos em um tabuleiro 3x3. O jogo pode ser representado como um problema de busca, no qual um agente recebe um estado inicial e um estado meta, e precisa encontrar uma sequência de ações que o leve a alcançar esse objetivo. Nesse contexto, o "Jogo da Velha"pode ser modelado como uma busca competitiva, um tipo de problema que envolve múltiplos agentes com objetivos conflitantes [1].

Um problema de busca é composto pelos seguintes elementos: agente, estado, estado inicial, ações, modelo de transição, conjunto de estados alcançáveis, solução e solução ótima. Ao tratar o "Jogo da Velha"como um problema de busca competitiva, é possível resolvê-lo utilizando algoritmos como o Minimax. Nele, o

2 Busca Competitiva

A busca competitiva ocorre em cenários onde múltiplos agentes (entidades que percebem seu ambiente e agem sobre ele) competem entre si, com objetivos conflitantes, tornando necessário considerar as ações dos oponentes e como elas impactam nas decisões da I.A. Este cenário é chamado de jogo, e podem ser facilmente resolvidos como um problema de busca [2]. Nesse algoritmo, como os oponentes são imprevisíveis, deve-se avaliar todas as possíveis jogadas do adversários para tomar uma boa decisão [5]. Normalmente, é determinado uma restrição de tempo, exigindo que o algoritmo retorne uma resposta detro do limite estabelecido, mesmo não sendo a solução ótima [7]. Em Inteligência Artificial, os jogos competitivos possuem características específicas, sendo elas [3]:

- são determinísticos: cada ação leva a um resultado conhecido;
- jogadas ocorrem de forma alternada entre dois jogadores;
- soma zero: o ganho de um jogador significa a perda do outro;
- informações perfeitas: todos os jogadores conhecem o estado atual do jogo.

O conjunto de estados alcançáveis é represetado por um árvore de busca. O oponente opera na mesma árvore de busca que o algoritmo, de forma competitiva, de forma que ele evita que o minimax atinja a solução. Como o oponente controla a busca em certos momentos, não é possível ter controle sobre o próximo estado em todos os níveis da árvore. Por isso, é necessário levar em conta todos os possíveis movimentos do adversário.

3 Algoritmo Minimax

O Minimax é um algoritmo de busca competitiva que assume que ambos os jogadores jogam de forma ideal. Um jogador busca maximizar sua pontuação (maximizador) e o outro tenta minimizá-la (minimizador). O algoritmo prevê todos os estados possíveis do jogo em uma árvore de busca, atribuindo valores positivos para vantagem do maximizador e negativos para o minimizador, usando heurísticas específicas para cada jogo. A partir dos estados terminais, esses valores são propagados até a raiz, onde o jogador escolhe a melhor jogada possível, alternando os papéis de maximizador e minimizador a cada turno. [6]

O pseudocódigo da função Minimax pode ser visto abaixo: [4]:

- Dado o estado atual do jogo, verifique se ele terminou. Se sim, retorne a pontuação associada àquele estado.
- Dado o papel do jogador, comece com uma melhor pontuação valendo –infinito, caso seja maximizador, ou +infinito, caso seja minimizador.
- Comece com um conjunto de ações possíveis a partir do estado atual.
- Para cada uma das ações:
  - Crie um novo estado a partir da ação atual.
  - Calcule uma nova pontuação a partir desse estado. Para fazê-lo, chame a função Minimax recursivamente, fornecendo a ela o novo estado e o papel do jogador oposto ao atual (isto é, se ele é maximizador, forneça minimizador e vice-versa).
  - Se o papel do jogador for maximizador e a nova pontuação for maior que a melhor pontuação, ou se o papel do jogador for minimizador e a nova

pontuação for menor que a melhor pontuação, faça a melhor pontuação valer a nova pontuação.

- Retorne a melhor pontuação.

3.1 Como o Minimax foi implementado no Jogo da Velha

Considerando o algoritmo do Minimax, para implementá-lo no jogo da velha precisou-se criar as seguintes funcionalidades específicas do jogo:

- Verificar se o jogo terminou (para cada linha, coluna e diagonais verifica-se se há um vencedor; caso não haja, verifica-se o caso de empate, isto é, todo o tabuleiro estar preenchido).
- Calcular a pontuação do tabuleiro (se "O"ganhou, atribui-se o valor -1, já que ele é o minimizador; se "X"ganhou, atribui-se o valor 1, já que ele é o maximizador; se houve empate, atribui-se o valor 0, já que ele não é vantajoso para nenhum dos jogadores).
- Determinar as ações possíveis a partir do tabuleiro atual (cada jogador pode escolher todas as casas vazias).

Além disso, no momento de utilizar o Minimax para encontrar as melhores jogadas possíveis, utilizamos o seguinte algoritmo:

- Dado o papel do jogador, comece com uma melhor pontuação valendo -infinito, caso seja "X", ou +infinito, caso seja "O".
- Para cada casa vazia do tabuleiro:
  - Crie uma cópia do tabuleiro preenchendo a casa atual com "X"ou "O", de acordo com o jogador.

– Calcule a pontuação desse tabuleiro com base no Minimax, fornecendo à função o estado do jogo atual e o papel do jogador. O papel do jogador será de maximizador, se jogador for "O"(já que o "X"é o próximo a jogar), ou de minimizador, se o jogador for "X"(já que o "O"é o próximo a jogar).
– Atribua à melhor pontuação o valor máximo entre ela e a pontuação atual.

Referências

[1] Lucas da Silva, Rodrigo Lyra, Valdir José Corrêa, and Alisson Steffens Henrique. Minimax no jogo da velha. *UNIVALI*, 2023.

[2] Fabrício Olivetti de França and Denis Fantinato. Inteligência artificial. *UFABC*, 2018.

[3] Rosalvo Ferreira de Oliveira Neto. Busca competitiva. *UNIVASF*.

[4] Ahmed A Elnaggar, Mahmoud Gadallah, Mostafa Abdel Aziem, and Hesham El-Deeb. A comparative study of game tree searching methods. *International Journal of Advanced Computer Science and Applications*, 5(5):68–77, 2014.

[5] Paulo Martins Engel. Inteligência artificial - busca competitiva - jogos. *Informática - UFRGS*.

[6] Stanford University. Algorithms - minimax. *Stanford University*, 2003.

[7] Jacques Wainer. Busca competitiva. *Instituto de Computação - UNICAMP*, 2014.