Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Relatório Jogo Conecta 4

Aluno(s):
Tainara Marina Gonçalves Morais e Tatiane Vitória de Oliveira

Julho
2023

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Relatório

Relatório do trabalho prático de implementação e avaliação de um agente de Inteligência Artificial (IA) para o jogo Conecta-4 .

Aluno: Tainara Marina Gonçalves Morais e Tatiane Vitória de Oliveira

Professor: Talles H Medeiros

Julho
2023

Conteúdo

1 Resumo **1**

2 Apresentação **2**

3 Descrição de atividades **3**
3.1 Minimax com poda Alfa Beta 3
3.2 Profundidade 3
3.3 A Heurística 4

4 Análise dos Resultados **8**

5 Trabalhos Futuros **12**

Bibliografia **12**

1 Resumo

Este artigo tem como objetivo aprimorar o entendimento sobre a busca competitiva em jogos de adversários, utilizando o jogo Connect-4 como estudo de caso. Para isso, um agente de Inteligência Artificial (IA) será implementado e avaliado, considerando as restrições de desempenho impostas pelo tamanho do tabuleiro.

A implementação do agente de IA envolve a criação de uma função de avaliação heurística para tomar decisões em tempo real. Dado que o jogo pode se tornar pesado quando o tabuleiro é grande, a heurística deve ser capaz de tomar decisões de qualidade mesmo sem explorar todas as possibilidades no tabuleiro. Isso permitirá ao agente tomar decisões eficientes dentro de um limite de tempo.

Além disso, será aplicada a técnica de poda alfa-beta para otimizar o algoritmo Minimax. Essa técnica permite ao agente explorar mais profundamente na árvore de jogo dentro do mesmo período de tempo, reduzindo o número de nós a serem visitados.

Para avaliar o desempenho do agente, será utilizado um tabuleiro maior, por exemplo, com 15 linhas e 16 colunas. O número total de nós visitados durante a busca será contabilizado como métrica de desempenho, permitindo comparar as diferentes versões do agente.

Os resultados e análises obtidos neste estudo contribuirão para o aprimoramento da compreensão sobre busca competitiva em jogos de adversários, bem como para o desenvolvimento de estratégias eficientes em jogos do tipo Connect-4, considerando a limitação de recursos computacionais.

2 Apresentação

O Connect 4 é um jogo de tabuleiro em que ganhar ou perder depende de movimentos realizados no passado. O jogo consiste em 2 jogadores e dois tipos diferentes de discos atribuídos a cada jogador. Há um tabuleiro tipo matriz composto por seis linhas e sete colunas, e cada jogador, turno a turno, coloca seus respectivos discos em qualquer uma das sete colunas maximizando sua chance de ganhar. Ganhar é definido quando um jogador consegue alinhar 4 mesmos tipos de discos na horizontal, vertical ou diagonal. Aproximadamente 4,5 trilhões de configurações de tabuleiro são possíveis para este jogo e são categorizadas como uma complexidade moderada

Para a implementação de um agente de Inteligência Artificial (IA) no Connect 4, utilizamos o algoritmo Minimax, amplamente empregado em jogos de computador para dois jogadores. Esse algoritmo gera uma árvore de jogo, em que os nós representam situações no jogo e as arestas correspondem a movimentos possíveis.

A nossa implementação do algoritmo Minimax, disponível no código fornecido na seção seguinte, considera a poda alfa-beta. Essa técnica é importante para otimizar o desempenho do algoritmo, permitindo que o agente explore a árvore de jogo de forma mais eficiente. A poda alfa-beta reduz o número de nós a serem visitados, eliminando ramificações que não afetam a decisão final.

Além disso, desenvolvemos uma heurística baseada na teoria dos jogos de soma-zero. Nesse tipo de jogo, os ganhos de um jogador são sempre correspondentes às perdas do outro jogador, resultando em uma soma total de pontos igual a zero. No Connect 4, cada jogador ganha ou perde pontos a cada turno, sendo que vencer resulta em ganhar x ponto e perder resulta em perder x ponto. Essa relação de soma-zero entre os jogadores é utilizada na heurística para avaliar o tabuleiro em tempo real e tomar decisões estratégicas. Assim, considerando todas as possíveis partidas de Connect 4, a soma total dos pontos ganhos pelos jogadores será sempre igual a zero. Essa característica define o Connect 4 como um jogo de soma zero, estabelecendo a base teórica para o desenvolvimento do agente de IA e sua estratégia de jogo.

3 Descrição de atividades

O código inicial fornecido pelo professor serviu como base para o início de toda a implementação. Começamos com a implementação da poda alfa-beta no algoritmo Minimax. Essa otimização permite que o agente explore mais profundamente a árvore de jogo no mesmo intervalo de tempo.

3.1 Minimax com poda Alfa Beta

Nosso algoritmo Minimax com poda alfa-beta, foi implementado da seguinte forma: A função verifica se o jogo foi vencido ou empatado e retorna a pontuação correspondente se for o caso. Se a profundidade máxima foi atingida, avalia o tabuleiro usando uma heurística específica para a IA. Em seguida, obtém as localizações válidas para a próxima jogada. Se for a vez do jogador maximizador, percorre as colunas válidas e realiza as seguintes etapas: cria uma cópia do tabuleiro, faz o movimento na coluna atual, chama recursivamente a função minimax para o próximo estado, atualiza o contador de nós visitados, atualiza a pontuação e a coluna da melhor jogada se a pontuação obtida for maior do que a atual, atualiza o valor de alfa com o máximo entre alfa e a nova pontuação e verifica se a condição de poda alfa-beta é atendida. Para o jogador minimizador, o processo é semelhante, mas atualiza o valor de beta com o mínimo entre beta e a nova pontuação. Por fim, a função retorna a coluna da melhor jogada, a pontuação associada a ela e a quantidade total de nós visitados.

3.2 Profundidade

Vale ressaltar que implementamos nosso algoritmo analisando uma estimativa da profundidade ótima:

Seja C o número de colunas do tabuleiro, e D a profundidade máxima permitida ao algoritmo percorrer. Em uma implementação de minimax sem poda alfa-beta, o número de nós-folha visitados é: C elevado a D.

Em uma implementação com poda alfa-beta, a otimização é, aparentemente, dada por um fator F no expoente, como a seguir: C elevado a D * F.

Esse fator, de acordo com a performance esperada do algoritmo no melhor e no pior caso (ver Alpha–beta pruning - Wikipedia), está entre 0.5 e 1.

Isso significa que podemos definir um limite L de quantos nós queremos que o algoritmo analise, e descobrir a profundidade ideal para permanecermos dentro deste limite. A equação seria:

$$C^{F \cdot D} \leq L$$

No entanto, antes de calcularmos o limite de profundidade D por definitivo, precisamos levar em consideração o fato de que a função de analisar o tabuleiro e a função de heurística são ambas (CR) (sendo R o número de linhas). Portanto, precisamos multiplicar o número de nós visitados por RC, para obtermos um valor proporcional ao custo (tempo) de visitar cada nó (e assim alcançarmos uma performance mais previsível).

$$\begin{aligned} C^{FD} \cdot CR &\leq L \\ C^{FD} &\leq \frac{L}{CR} \\ FD &\leq \log_C \left(\frac{L}{CR} \right) \\ D &\leq \frac{\log_C \left(\frac{L}{CR} \right)}{F} \end{aligned}$$

A profundidade em uma árvore de busca representa o número de jogadas à frente que o algoritmo analisará. Quanto maior a profundidade, mais jogadas o algoritmo considerará e mais precisa será a avaliação da posição atual do jogo. No entanto, aumentar a profundidade também aumenta a complexidade computacional, tornando o algoritmo mais lento.

A escolha adequada da profundidade depende do jogo específico, da capacidade computacional disponível e das restrições de tempo. Uma profundidade maior geralmente leva a decisões melhores, mas também requer mais recursos computacionais. É importante encontrar um equilíbrio entre a precisão das decisões e o desempenho do algoritmo.

3.3 A Heurística

Além disso, uma função de avaliação heurística foi desenvolvida para tomar decisões em tempo real. A heurística foi projetada levando em consideração o desempenho do jogo em tabuleiros de tamanho grande. Como o jogo pode se tornar lento com um tabuleiro extenso, a heurística é capaz de tomar decisões de qualidade mesmo sem explorar todas as possibilidades do tabuleiro. Isso significa que o agente pode tomar decisões estratégicas eficientes, mesmo com uma análise parcial do tabuleiro.

Inicialmente pedimos ao chat gpt para gerar uma heuristica aleatória para que pudéssemos entender como ela melhorou o codigo do professor. A heuristica disponibilizada não era eficaz, já que conseguimos ganhar do agente muitas vezes (essa constatação foi feita com base nas vezes que jogamos

contra o agente com essa heuristica). Esta função de jogo não proporcionou uma experiência interessante, e a inteligência artificial não demonstrou ser suficientemente "esperta". Ficou evidente que nessa função não foi atribuída importância ao caso em que o oponente possui quatro peças alinhadas, e, portanto, o robô não se esforçava tanto para impedir que isso ocorresse.

```
if player_piece_count == 4:
score += 100
elif player_piece_count == 3 and empty_count == 1:
score += 5
elif player_piece_count == 2 and empty_count == 2:
score += 2

if opponent_piece_count == 3 and empty_count == 1:
 score -= 4
```

Em seguida, encontramos um artigo que apresentava uma heurística considerada eficiente quando combinada com o algoritmo Minimax:

```
if player_piece_count == 4:
 score += 1_000_000
elif player_piece_count == 3:
 score += 1
elif opponent_piece_count == 3:
 score -= 100
elif opponent_piece_count == 4:
 score -= 1_000_000
```

Com essa função, o comportamento da IA era confuso, pois não reconhecia as "ameaças"construídas pelo oponente e não evitava que o jogador humano formasse três peças alinhadas. Isso tornava fácil enganar a inteligência artificial até o último momento e vencer o jogo facilmente.

Outro problema era a falta de consideração por peças contíguas com espaços vazios. Ou seja, se houvesse três peças contíguas, mas uma peça

do oponente bloqueando de cada lado, essa posição não era vantajosa, mas a função não levava isso em conta.

Após a leitura do documento "Heuristic Connect4 player with Alpha Beta Pruning", identificamos uma estratégia interessante para o design de uma função heurística mais eficaz. Essa estratégia envolve atribuir pesos crescentes exponencialmente para cada caso (2 peças contíguas, 3 peças contíguas e 4 peças contíguas). Além disso, corrigimos o problema de consideração dos espaços vazios mencionado anteriormente.

Tendo como base essas duas heurísticas anteriores, começamos a criar a nossa heurística, que leva em conta as ameaças do oponente, tome decisões mais inteligentes e seja capaz de identificar posições vantajosas, mesmo levando em consideração os espaços vazios entre as peças.

Além disso, para definir nossa heurística, optamos por seguir a ideia dos jogos de soma-zero. Nesses jogos, os ganhos de um jogador são sempre correspondentes às perdas do outro jogador, resultando em uma soma total de pontos igual a zero.

No Connect 4, cada jogador ganha ou perde pontos a cada turno, onde ganhar resulta em +1 ponto e perder resulta em -1 ponto. Em caso de empate, nenhum dos jogadores ganha ou perde pontos. Utilizamos essa relação de soma-zero para avaliar o tabuleiro em tempo real e tomar decisões estratégicas. Assim, considerando todas as possíveis partidas de Connect 4, a soma total dos pontos ganhos pelos jogadores será sempre igual a zero.

Com base nessa abordagem, criamos uma heurística em que as vitórias e as derrotas são simétricas, refletindo a relação de soma-zero. Essa heurística foi implementada para melhorar o desempenho do agente de IA no jogo Connect 4, permitindo que ele tome decisões estratégicas mais eficientes durante as partidas.

A heurística implementada no código é usada para avaliar o tabuleiro e determinar a pontuação de uma determinada posição. Essa pontuação é usada pelo algoritmo Minimax para tomar decisões estratégicas durante o jogo Connect 4.

A heurística utiliza várias sequências de peças no tabuleiro para calcular a pontuação. Ela percorre o tabuleiro em busca de sequências horizontais, verticais e diagonais de comprimento 4. Para cada sequência encontrada, a heurística atribui uma pontuação com base no número de peças do jogador atual (player-piece), peças do oponente (opponent-piece) e espaços vazios (empty-count) na sequência.

A heurística passa por várias iterações, cada uma com uma abordagem diferente para a atribuição de pontuações. Na última iteração, a pontuação é atribuída da seguinte forma:

Se o jogador atual tiver 4 peças na sequência, a pontuação é incrementada

em 1.000.000.

Se o jogador atual tiver 3 peças na sequência e houver 1 espaço vazio, a pontuação é incrementada em 100.

Se o jogador atual tiver 2 peças na sequência e houver 2 espaços vazios, a pontuação é incrementada em 10.

Se o jogador atual tiver 1 peça na sequência e houver 3 espaços vazios, a pontuação é incrementada em 1.

Se o oponente tiver 1 peça na sequência e houver 3 espaços vazios, a pontuação é decrementada em 1.

Se o oponente tiver 2 peças na sequência e houver 2 espaços vazios, a pontuação é decrementada em 10.

Se o oponente tiver 3 peças na sequência e houver 1 espaço vazio, a pontuação é decrementada em 100.

Se o oponente tiver 4 peças na sequência, a pontuação é decrementada em 1.000.000.

Essa heurística é utilizada pelo algoritmo Minimax para avaliar o tabuleiro em cada nó da árvore de jogo, permitindo que o agente tome decisões estratégicas com base na pontuação calculada.

A combinação da poda alfa-beta com a heurística de avaliação permite ao agente fazer escolhas mais informadas e melhorar o desempenho do jogo. A poda alfa-beta reduz o número de nós visitados na árvore de jogo, acelerando o processo de busca, enquanto a heurística fornece uma estimativa da qualidade de uma determinada situação do jogo.

Essas implementações visam otimizar o desempenho do agente de Inteligência Artificial no Connect 4, permitindo que ele tome decisões estratégicas de forma mais rápida e eficiente, mesmo em tabuleiros maiores. Dessa forma, o agente é capaz de jogar o Connect 4 de maneira competitiva e oferecer uma experiência desafiadora para os jogadores.

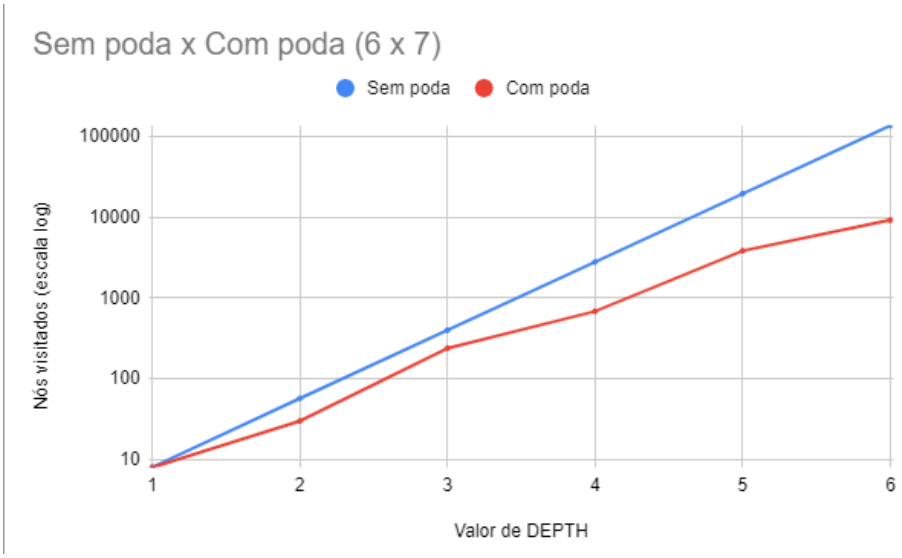
4 Análise dos Resultados

Nesta seção, vamos apresentar uma análise dos resultados obtidos ao longo deste trabalho. Mostraremos os resultados obtidos ao executar nosso algoritmo de Connect 4 com diferentes níveis de profundidade, tanto com o uso do algoritmo de poda alfa-beta quanto sem ele. Além disso, faremos uma comparação com outras heurísticas encontradas na internet e avaliaremos seu desempenho em relação à nossa heurística, que se baseia em jogos de soma-zero, nos discos posicionados no tabuleiro e nas vagas disponíveis para colocar discos.

Primeiramente, rodamos para o valor padrão do tabuleiro, 6x7, e obtivemos os seguintes resultados:

Valores para um jogo de COLUMNS = 7	Número de nós visitados		
Valor de DEPTH	Sem poda	Com poda	Quantas vezes a poda é melhor que sem
1	8	8	1
2	57	30	1.9
3	400	239	1.673640167
4	2801	685	4.089051095
5	19608	3847	5.096958669
6	137257	9229	14.87235887

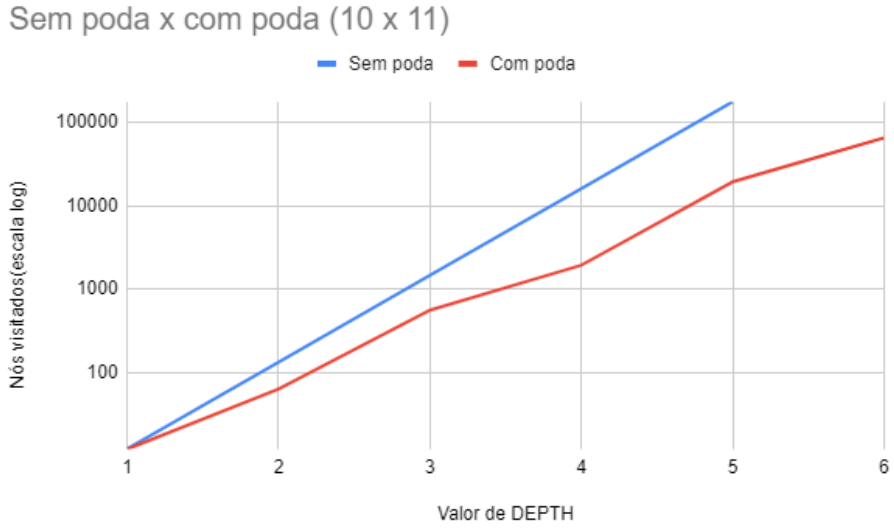
O gráfico correspondente:



Como podemos ver nesse tamanho de tabuleiro, quanto maior a profundidade(depth), menor é a quantidade de nós visitados pelo algorítmo, já que ele consegue ignorar nós inalcançáveis/desnecessários para encontrar a melhor solução. Apresentamos também quantas vezes o algoritmo com poda se mostra mais eficaz que o algoritmo sem a poda. Além disso, é possível perceber que: com o aumento da profundidade, o aumento dos nós visitados pelo algoritmo sem poda cresce em intervalos muito maiores do que com a poda, ou seja, a consistência da quantidade de nós visitados com a poda é maior.

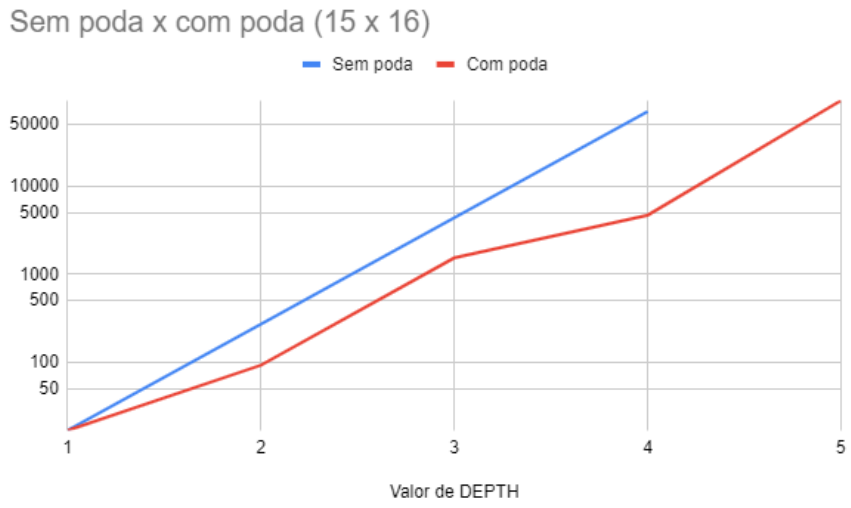
Apresentaremos a seguir gráficos do desempenho do algorítmo em tabuleiros cada vez maiores, como o 10x11 e 15x16.

Valores para um jogo de C = 11, R = 10	Número de nós visitados		
Valor de DEPTH	Sem poda	Com poda	Quantas vezes a poda é melhor que sem
1	12	12	1
2	133	63	2.111111111
3	1464	557	2.628366248
4	16105	1924	8.370582121
5	177156	19327	9.166244114
6		65465	0



Também realizamos o teste no tabuleiro de 15x16, conforme consta no gráfico a seguir:

Valores para um jogo de C = 16, R = 15	Número de nós visitados		
Valor de DEPTH	Sem poda	Com poda	Quantas vezes a poda é melhor que sem
1	17	17	1
2	273	93	2.935483871
3	4369	1537	2.842550423
4	69905	4674	14.95614035
5		93840	



Como podemos ver através dos gráficos, em todos os tamanhos de tabuleiro, a utilização da poda alfa-beta reduziu o número de nós visitados em comparação à execução sem poda e também, demonstrou ser uma técnica eficaz para reduzir o número de nós visitados durante a busca no Connect 4, resultando em melhor desempenho computacional, especialmente em tabuleiros maiores e com maiores profundidades de busca.

5 Trabalhos Futuros

Como trabalho futuro ou melhoria no algoritmo atual, sugerimos realizar uma análise detalhada sobre a influência da paridade da profundidade na eficácia da poda alfa-beta. Durante a análise dos dados obtidos, observamos indícios de que a poda alfa-beta pode se beneficiar da paridade da profundidade, o que consideramos um aspecto relevante a ser explorado.

Nossa proposta é investigar mais a fundo essa relação entre a paridade da profundidade e o desempenho da poda alfa-beta. Pretendemos realizar experimentos adicionais, comparando o desempenho da poda alfa-beta em diferentes configurações de profundidade, tanto em profundidades pares quanto ímpares. Essa análise detalhada nos permitirá compreender melhor como a paridade da profundidade influencia a eficiência da poda alfa-beta e, assim, obter insights valiosos para aprimorar o algoritmo.

Além disso, também consideramos importante explorar outras técnicas de otimização e aprimoramento do algoritmo de Connect 4. Podemos investigar a aplicação de heurísticas mais avançadas, explorar estratégias de busca mais eficientes ou até mesmo considerar abordagens de aprendizado de máquina para aprimorar a tomada de decisões do algoritmo.

Em resumo, como trabalho futuro, propomos uma análise detalhada da influência da paridade da profundidade na poda alfa-beta, a fim de compreender melhor sua relação com o desempenho do algoritmo. Essa análise nos permitirá aprimorar o algoritmo atual e explorar outras possíveis melhorias, como a aplicação de heurísticas avançadas ou estratégias de busca mais eficientes.

Bibliografia

Alpha–beta pruning - Wikipedia,
Disponível em: https://en.wikipedia.org/wiki/Alpha–beta-pruning

Heuristic Connect4 player with Alpha Beta Pruning,
Disponível em: http://www.cs.cornell.edu/boom/2001sp/Anvari/Anvari.htm

Introdução à Teoria dos Jogos Algorítmica,
Disponível em: http://www1.eeg.uminho.pt/economia/caac/pagina(Porcentagem)20pessoal
/Disciplinas/Disciplinas(Porcentagem)2004/jogos.pdf

SOLVING CONNECT 4 USING OPTIMIZED MINIMAX AND MONTE CARLO TREE SEARCH,
Disponível em: hhttps://www.mililink.com/upload/article/279817393aams$_{v}ol_{2}16_{a}pril_{2}022_{a}25_{p}3303-3313_{k}avita_{s}heoran,_{e}t_{a}l..pdf$