# 

# Sistemas de Inteligentes

**Relatório: Análise e Comparação de Algoritmos para o Jogo do Galo**

**1. Descrição do Jogo**

O Jogo do Galo (conhecido também como Tic Tac Toe) é um jogo de tabuleiro para dois jogadores. Num tabuleiro de 3×3, os jogadores, alternadamente, marcam os espaços com os seus símbolos (normalmente "X" ou "O"). O objetivo é alinhar três símbolos consecutivos na horizontal, vertical ou diagonal. Se todas as casas forem preenchidas sem que nenhum jogador consiga essa sequência, o jogo termina empatado. Na implementação apresentada, o jogo permite modos de jogo humano versus computador e computador versus computador, possibilitando ainda uma análise comparativa dos algoritmos de tomada de decisão.

**2. Descrição dos Algoritmos Implementados**

**2.1 Algoritmo Minimax**

**Implementação:**  
O algoritmo **Minimax** é uma técnica recursiva utilizada em jogos de dois jogadores (zero-sum) para determinar a jogada ótima. No código, a função minimax avalia todas as jogadas possíveis a partir do estado atual do tabuleiro e atribui uma pontuação que reflete vitória, derrota ou empate.

* **Casos Base:**
  + Se o jogador vencer, retorna uma pontuação positiva (10 subtraído da profundidade, de forma a valorizar vitórias mais céleres).
  + Se o adversário vencer, retorna uma pontuação negativa (a profundidade subtraída de 10, enfatizando derrotas mais tardias).
  + No caso de empate, retorna zero.
* **Recursividade:**
  + Para o jogador que maximiza, a função seleciona a jogada com a maior pontuação obtida.
  + Para o jogador minimizador, seleciona-se a jogada com a menor pontuação.

**2.2 Algoritmo Alpha-Beta**

**Implementação:**  
O algoritmo **Alpha-Beta** constitui uma otimização do Minimax que reduz o número de nós a serem avaliados através de uma técnica de poda. A função alpha\_beta recebe, para além dos parâmetros do Minimax, dois valores: **alpha** e **beta**.

* **Alpha:** Valor mínimo que o jogador maximizador está garantido a obter.
* **Beta:** Valor máximo que o jogador minimizador assegura.  
  Durante a recursividade, se se verificar que o valor atual não pode produzir uma decisão melhor para o jogador, a subárvore é descartada, economizando tempo e recursos computacionais, sem comprometer a escolha da jogada ótima.

**3. Discussão das Principais Características dos Algoritmos**

**3.1 Optimalidade**

* **Minimax:**  
  Garante a escolha da jogada ótima ao explorar por completo a árvore de decisões, assegurando que se maximiza a probabilidade de vitória (ou se minimiza a probabilidade de derrota).
* **Alpha-Beta:**  
  Mantém a mesma garantia de optimalidade que o Minimax, uma vez que apenas elimina ramos que não influenciam a decisão final, preservando a integridade da escolha ótima.

**3.2 Completude**

Ambos os algoritmos são completos em jogos com um espaço de estados finito, como é o caso do Jogo do Galo. Isto significa que, ao explorar todas as possibilidades (ou ao podar ramos irrelevantes sem afetar a decisão final), os algoritmos encontram sempre uma solução se esta existir.

**3.3 Complexidade**

* **Minimax:**  
  A complexidade do Minimax é exponencial face à profundidade da árvore de decisão, da forma geral representada por O(bd)O(b^d)O(bd), onde bbb é o fator de ramificação (número de jogadas possíveis por movimento) e ddd é a profundidade da árvore. Embora seja adequado para jogos simples, torna-se inviável à medida que o espaço de estados aumenta.
* **Alpha-Beta:**  
  A poda alpha-beta reduz drasticamente o número de nós avaliados. Em condições ideais, a complexidade pode ser reduzida para O(bd/2)O(b^{d/2})O(bd/2), o que permite uma execução significativamente mais rápida sem comprometer a qualidade da decisão.

**4. Estudo do Custo de Tempo e Memória**

Na implementação, foram incluídas funções para medir o desempenho de ambos os algoritmos, nomeadamente:

* **Tempo de Execução:**  
  A função obter\_jogada\_computador mede o tempo que cada algoritmo demora para calcular a melhor jogada. Ensaios realizados com diferentes dimensões de tabuleiro (3x3 e 4x4) demonstram que a poda alpha-beta apresenta tempos de execução inferiores em relação ao Minimax convencional, especialmente quando o número de estados possíveis aumenta.
* **Contagem de Nós Visitados:**  
  A função contar\_nos\_visitados permite contabilizar o número de nós explorados durante a execução de cada algoritmo. Os resultados obtidos evidenciam que o algoritmo alpha-beta explora consideravelmente menos nós, refletindo uma utilização mais eficiente da memória e dos recursos computacionais.

Os gráficos gerados (guardados como comparacao\_algoritmos.png e comparacao\_algoritmos\_jogo.png) proporcionam uma visualização clara do desempenho dos algoritmos, demonstrando a eficácia da técnica de poda em reduzir o tempo de execução e o custo computacional.

**5. Discussão dos Resultados**

Com base nos testes e nas visualizações gráficas:

* **Desempenho:**  
  O algoritmo alpha-beta mostrou-se mais eficiente, tanto em termos de tempo de execução como na redução do número de nós visitados, comparativamente ao Minimax. Esta vantagem torna o alpha-beta particularmente adequado para jogos ou cenários com espaços de estados mais extensos.
* **Eficácia da Poda:**  
  A diminuição significativa do número de nós explorados valida o emprego da técnica de poda alpha-beta, que permite ao algoritmo agir de forma mais célere e com menor utilização de memória, sem sacrificar a optimalidade da decisão.
* **Aplicabilidade:**  
  Embora ambos os algoritmos sejam viáveis para o Jogo do Galo, num contexto de jogos com maior complexidade, a estratégia de poda torna-se essencial para garantir uma tomada de decisão em tempo útil.

**6. Conclusões Principais**

* **Optimalidade e Completude:**  
  Tanto o Minimax como o Alpha-Beta asseguram a escolha da jogada ótima e são completos para jogos com um número finito de estados.
* **Eficiência do Alpha-Beta:**  
  A técnica de poda alpha-beta reduz significativamente a complexidade computacional (reduzindo o número de nós visitados e o tempo de execução), sem comprometer a decisão ótima.
* **Custo de Tempo e Memória:**  
  O Minimax, apesar de ser simples de implementar, pode levar a um consumo excessivo de tempo e memória com o aumento da profundidade da árvore de decisões. Em contrapartida, o Alpha-Beta apresenta uma otimização notória, tornando-o preferencial em contextos de maior complexidade.
* **Possíveis Melhorias Futuras:**  
  A abordagem atual pode ser expandida para tabuleiros de dimensões superiores ou para jogos de maior complexidade. Adicionalmente, poderão ser exploradas técnicas complementares, como heurísticas de ordenação de jogadas, que contribuam para uma otimização ainda maior dos algoritmos.