TP2: Busca Competitiva em Ligue-4 Utilizando Minimax, Poda Alfa-Beta e Iterative Deepening

Caio Santana Trigueiro
Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
caiosantana@dcc.ufmg.br

Abstract

Este trabalho apresenta a implementação de um agente de inteligência artificial para jogar Ligue-4 (Connect Four) utilizando algoritmos de busca adversarial. Foram implementados os algoritmos Minimax com profundidade limitada, poda Alfa-Beta e Iterative Deepening, combinados com uma função heurística que avalia posições do tabuleiro. Os experimentos demonstram a eficácia da poda Alfa-Beta em reduzir o número de nós explorados e a capacidade do Iterative Deepening em aproveitar melhor o tempo disponível. Os resultados mostram que o agente implementado é capaz de vencer consistentemente contra um jogador aleatório e apresenta desempenho competitivo em diferentes configurações de profundidade e tempo.

Introdução e Objetivo

O objetivo deste trabalho é implementar e avaliar algoritmos de busca adversarial para o jogo Ligue-4 (Connect Four). Especificamente, implementamos:

- Algoritmo Minimax com profundidade limitada e função heurística
- Poda Alfa-Beta para otimização da busca
- Iterative Deepening com limite de tempo
- Função heurística de avaliação de posições

O trabalho visa demonstrar a eficácia desses algoritmos e analisar trade-offs entre profundidade de busca, tempo de execução e qualidade das decisões.

Metodologia

Evolução do Agente

O agente foi desenvolvido incrementalmente, seguindo as seguintes etapas:

Baseline: Agente Aleatório Inicialmente, validamos o ambiente com um agente que escolhe jogadas aleatoriamente entre as colunas válidas.

Minimax com Profundidade Limitada Implementamos o algoritmo Minimax clássico com profundidade máxima configurável. Para estados terminais, atribuímos valores extremos (vitória = $+\infty$, derrota = $-\infty$, empate = 0). Para estados não terminais, utilizamos uma função heurística.

Poda Alfa-Beta Adicionamos a poda Alfa-Beta ao Minimax, mantendo a mesma decisão ótima mas reduzindo significativamente o número de nós explorados.

Iterative Deepening Implementamos Iterative Deepening que explora profundidades progressivamente (1, 2, 3, ...) até atingir a profundidade máxima ou estourar o limite de tempo, mantendo sempre a melhor jogada conhecida.

Função Heurística

Nossa função heurística `evaluate()` combina três componentes:

1. **Valorização do centro**: Peças nas colunas centrais (2, 3, 4) recebem pontos adicionais, com a coluna 3 (central) recebendo maior peso.
2. **Contagem de sequências**: Sequências de 2 peças recebem peso 1, sequências de 3 peças recebem peso 10.
3. **Detecção de ameaças**: Sequências de 3 peças que podem virar 4 (com espaço vazio acessível) recebem peso muito alto (500 para ameaças próprias, -1000 para ameaças do oponente).

Decisões de Projeto

- **Ordenação de jogadas**: Implementamos ordenação que prioriza colunas centrais, melhorando a eficiência da poda Alfa-Beta.
- **Verificação de acessibilidade**: A detecção de ameaças considera apenas espaços vazios acessíveis (que podem receber peças por gravidade).
- **Contadores de estatísticas**: Implementamos contadores para número de nós visitados e nós podados, essenciais para análise experimental.

Experimentos e Resultados

Minimax vs Aleatório

Realizamos experimentos comparando Minimax com diferentes profundidades (2, 3, 4, 5) contra um jogador aleatório.

Table 1: Resultados: Minimax vs Aleatório

Profundidade	Taxa Vitória	Tempo Médio (ms)	Estados Visitados
2	XX%	XX	XX
3	XX%	XX	XX
4	XX%	XX	XX
5	XX%	XX	XX

Alfa-Beta vs Minimax (sem poda)

Comparamos a versão com poda Alfa-Beta contra a versão sem poda, mantendo a mesma profundidade.

Table 2: Resultados: Alfa-Beta vs Minimax (sem poda)

Profundidade	Taxa Vitória	Tempo Médio (ms)	Estados Visitados
2	XX%	XX	XX (AB) / XX (Minimax)
3	XX%	XX	XX (AB) / XX (Minimax)
4	XX%	XX	XX (AB) / XX (Minimax)
5	XX%	XX	XX (AB) / XX (Minimax)

Iterative Deepening vs Alfa-Beta

Avaliamos o Iterative Deepening com limites de tempo de 1s e 2s contra Alfa-Beta com profundidade fixa.

Table 3: Resultados: Iterative Deepening vs Alfa-Beta

Limite Tempo	Taxa Vitória	Tempo Médio (ms)	Prof. Média	Estados Visitados
1s	XX%	XX	XX	XX
2s	XX%	XX	XX	XX

IA vs Jogador Humano

Joguei X partidas contra a IA implementada. Percepções qualitativas:

- **Forças**: A IA demonstra boa capacidade de bloquear ameaças imediatas e criar oportunidades de vitória.
- **Fraquezas**: [A completar após mais partidas]

Discussão

Análise dos Resultados

[Análise crítica dos resultados obtidos, incluindo:

- Impacto da profundidade na qualidade das decisões
- Eficiência da poda Alfa-Beta
- Vantagens do Iterative Deepening
- Trade-offs entre tempo e qualidade

]

Limitações

- A função heurística pode ser melhorada com mais componentes
- Profundidades maiores (5+) são computacionalmente caras
- Não implementamos tabela de transposições (otimização futura)

Conclusão

Este trabalho demonstrou a implementação bem-sucedida de algoritmos de busca adversarial para Ligue-4. A poda Alfa-Beta mostrou-se essencial para viabilizar buscas em profundidades maiores, enquanto o Iterative Deepening permitiu melhor aproveitamento do tempo disponível. A função heurística implementada, embora simples, mostrou-se eficaz para guiar a busca.

Ideias de melhorias futuras:

- Implementar tabela de transposições
- Melhorar função heurística com mais componentes
- Otimizações adicionais para competição