2ª Fase Projeto



## Unidade Curricular de Inteligência Artificial

Prof. Joaquim Filipe Eng. Filipe Mariano

# Jogo do Cavalo

## Manual Técnico

Trabalho realizado por:

João Nanques - 202002588 Emanuel Martins - 202100104

Turma IG-02

# Indice

. Introdução

. Arquitetura do Sistema

. Entidades e sua implementação

. Algoritmos e sua implementação

. Resultados

. Limitações técnicas e ideias para desenvolvimento futuro

# Introdução

O presente projeto tem como objetivo a aplicação de conhecimentos com a programação do Jogo Do Cavalo, desta forma abordaremos neste relatório o que envolve a implementação do jogo em LISP puro. Neste desafio os algoritmos de busca têm muita importância, o BFS (Procura em Largura), o DFS (Procura em Profundidade) e o A\*. O Jogo Do Cavalo, usa um tabuleiro 10x10 como base para os movimentos, apresenta um cavalo cujo movimentos são restritos e apenas 8 em L. Este movimento singular, composto por oito direções possíveis, é implementado como operadores em LISP.

# Arquitetura do Sistema

O sistema produzido para a resolução do desafio do Jogo Do Cavalo foi implementado em linguagem LISP puro, utilizando o IDE LispWorks.

### A resolução foi divididaem 4 ficheiros:

 interact.lisp - Implementação da lógica dos menus, ficheiro principal responsável pela interação.  jogo.lisp - Implementação dos operadores e lógica de jogo.

 algoritmo.lisp - Implementação do algoritmo Alfa-Beta e Mini-Max com cortes Alfa-Beta.

 log.dat - Ficheiro responsável pelo armazenamento de logs oriundos da execução do programa.

# Entidades e implementação

## Tabuleiro

O tabuleiro é uma representação em forma de lista de listas em LISP, composta por átomos, onde cada átomo representa uma casa com um valor numérico de 0 a 99, este tabuleiro é estruturado por 10 linhas por 10 colunas.

Exemplo de um tabuleiro gerado aleatóriamente :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ((32 | 19 | 47 | 53 | 64 | 11 | 89 | 05 | 70 | 56) |
| (27 | 84 | 91 | 13 | 42 | 28 | 60 | 75 | 39 | 98) |
| (07 | 69 | 53 | 18 | 29 | 46 | 88 | 21 | 04 | 77) |
| (95 | 12 | 68 | 30 | 80 | 57 | 03 | 44 | 65 | 23) |
| (61 | 33 | 22 | 15 | 41 | 73 | 02 | 87 | 94 | 50) |
| (79 | 36 | 51 | 09 | 55 | 16 | 20 | 37 | 96 | 62) |
| (74 | 08 | 40 | 45 | 01 | 58 | 66 | 10 | 24 | 14) |
| (31 | 82 | 85 | 49 | 26 | 06 | 43 | 72 | 38 | 97) |
| (59 | 76 | 81 | 25 | 35 | 90 | 54 | 63 | 93 | 78) |
| (67 | 52 | 86 | 16 | 17 | 34 | 48 | 22 | 71 | 99)) |

## Funções e Regras do Jogo

Nesta secção do documento, vamos descrever as funcões e regras implementadas no projecto sabendo que os movimentos apenas podem se realizar em posições disponiveis do tabuleiro.

### Regra do Simétrico

A regra do Simétrico coloca o valor simétrico da casa visitada no movimento atual como NIL, deixando esta casa de puder ser visitada em futuras jogadas.

### Regra do Duplo

A regra do duplo verifica se o valor da casa visitada no movimento atual é um número duplo (número com os dois algarismos iguais, por exemplo "11", "22", "66"). No caso do número ser um duplo duplo, coloca a casa com o maior número duplo disponível a NIL, deixando esta casa de puder ser visitada em futuras jogadas.

### Regra do Cavalo

Regra do cavalo tem como objetivo colocar o cavalo em jogo, a regra é aplicada sempre que se começa a procura no tabuleiros. A regra começa por validar que o cavalo está colocado numa casa, tambem só se pode colocar o cavalo na primeira linha do tabuleiro.

O resultado da regra é um conjunto de casas disponiveis para a jogada, aplicando as mesmas regras definidas acima (Regra do Duplo e Simetrico).

## Representação de Estados

O Jogo Do Cavalo permite que o tabuleiro obtenha as várias possibilidades de jogadas e os caminhos possíveis até encontrar uma solução, neste caso o problema será equacionado em termos de estados, em que são representados os estados ou uma representação sequencial, operadores que permitem a transição dos estados desde do estado inicial até o final, havendo a possibilidade de utilizar a exploração das árvores.

### Operadores

Considerando as 8 operações possíveis em L criámos 8 operadores que vão realizar esses movimentos, os movimentos podem ser feitos em relação às linhas e às colunas.

A organização da função é (linha , coluna), a linha representa os movimentos para cima/baixo e a coluna os movimentos para a direita/esquerda.

Os operadores são representam os seguintes movimentos: operador-1 (2 1)

operador-2 (1 2)

operador-3 (-1 2)

operador-4 (-2 1)

operador-5 (-2 -1)

operador-6 (-1 -2)

operador-7 (1 -2)

operador-8 (2 -1)

Nó

Composição do Nó

Um nó é composto pelo seguinte:

(Tabuleiro | Posição | Pontos | Profundidade | Pai | Heuristica) Seletores do Nó

Os seletores do nó retornam os atributos que fazem parte do nó. Nomeadamente:

\*no-estado-tabuleiro no-posicao

no-pontos

no-profundidade no-pai

no-custo

### Sucessões

A sucessão de um nó, num contexto de xadrez, representa o conjunto de movimentos disponíveis para um cavalo a partir da sua posição atual no tabuleiro. O cavalo move-se em forma de "L", podendo avançar duas casas numa direção e, em seguida, uma casa perpendicular ou vice-versa. Esta sucessão é crucial em algoritmos de busca, como o alfa-beta, pois ao analisar as opções de movimento, é possível determinar estrategicamente as jogadas mais vantajosas, contribuindo para a tomada de decisões informadas durante o jogo.

# Algoritmos e sua implementação

Neste projeto, abordamos o desafio do Cavalo, cujo objetivo primário é alcançar a pontuação estipulada para cada problema com o menor número possível de jogadas. Para solucionar esse problema, recorremos aos algoritmos discutidos nas aulas, empregando-os na navegação do cavalo ao longo do tabuleiro. O foco está na busca pela solução por meio de movimentos possíveis até que atingir os objetivos não seja mais viável.

## Algoritmo Alfa-Beta

O algoritmo alfa-beta é uma técnica utilizada em árvores de decisão para otimizar a busca em jogos de dois jogadores, como por exemplo o jogo do cavalo. Este algoritmo é uma extensão do algoritmo minimax, que explora todas as possíveis jogadas, mas pode ser bastante ineficiente em termos computacionais. O alfa- beta corta partes da árvore de procura que são garantidamente irrelevantes para a escolha final da jogada. O "alfa" representa a melhor escolha atual para o jogador maximizador, enquanto o "beta" representa a melhor escolha para o jogador minimizador. Durante a pesquisa, se é descoberto que uma determinada escolha é pior do que a escolha atual para o jogador minimizador (beta é menor), ou melhor do que a escolha atual para o jogador maximizador (alfa é maior), a busca nesse ramo pode ser interrompida, economizando assim recursos computacionais.

Integrar este algoritmo num projeto em Common Lisp envolve a implementação eficiente de funções recursivas que avaliam as jogadas possíveis, atualizando os valores de alfa e beta conforme necessário. É importante manter um código claro e bem estruturado, aproveitando as características de Lisp para facilitar a manipulação de árvores e a passagem de funções como argumentos. Ao fazer isso, será possível criar uma implementação robusta do algoritmo alfa-beta em Common Lisp, melhorando assim o desempenho na tomada de decisões em jogos estratégicos.

# Limitações técnicas e ideias para desenvolvimento futuro

Durante o desenvolvimento do projeto, enfrentamos desafios significativos na implementação dos algoritmo Alfa-Beta. Outra das grandes dificuldades foi fazer o trace de alguns erros que foram surgindo no

processo de adaptação do que foi feito na 1ª Fase do projeto. Como foi dito anteriormente o que foi mais

complexo foi a implementação do Alfa-Beta onde destacamos o entendimento do seu funcionamento e a maneira como alterariamos o valor do Alfa ou Beta.