Projeto Nº1: Época Normal

Inteligência Artificial 2025/2026

Docentes: Prof. Joaquim Filipe, Eng. Filipe Mariano

Jogo do Solitário

Manual de Utilizador

Realizado por:

- António Guerreiro (202200160)
- Bruno Leite (202100504)
- Guilherme Cruz (2024129841)

Data: 12 de Dezembro de 2025

Índice

---

1. Introdução

No âmbito da unidade curricular de Inteligência Artificial, foi proposto como primeiro projeto a resolução do **Jogo do Solitário** (*Peg Solitaire*) utilizando métodos de procura em espaço de estados.

O objetivo do programa consiste em encontrar uma sequência de movimentos válidos que reduzam o número de peças no tabuleiro a apenas uma (solução ótima), utilizando algoritmos de procura não informada (BFS, DFS) e informada (A*).

2. Instalação e Utilização

O projeto foi desenvolvido em linguagem Common Lisp com recurso ao ambiente de desenvolvimento **LispWorks**.

Requisitos do Sistema

Para a correta execução do software, é necessário garantir que os seguintes ficheiros se encontram na mesma diretoria:

- `projeto.lisp`: Ficheiro principal contendo o menu, a gestão de ficheiros e a interação com o utilizador.
- `puzzle.lisp`: Módulo do domínio contendo a definição do tabuleiro, peças, operadores, e heurísticas.
- `procura.lisp`: Módulo genérico contendo a implementação dos algoritmos de procura (BFS, DFS, A*) e métricas.
- `problemas.dat`: Ficheiro de dados contendo as configurações iniciais dos tabuleiros a resolver.

3. Configuração

Para iniciar o sistema, siga os passos abaixo no LispWorks:

1. **Iniciar o LispWorks** e abrir o ficheiro `projeto.lisp`.

2. **Definir o Diretório de Trabalho:** Antes de compilar, é necessário indicar ao LispWorks onde estão os ficheiros para que as dependências sejam carregadas corretamente. Execute o seguinte comando no *Listener*, ajustando o caminho para a sua pasta local:

```
(hcl:change-directory "C:\\Caminho\\Para\\O\\Vosso\\Projeto\\")
```

3. **Carregar o Programa:** Compile e carregue o buffer do ficheiro `projeto.lisp`. O código contém instruções para carregar automaticamente os módulos dependentes:

```
(compile-file "puzzle.lisp")
(load "puzzle")
(compile-file "procura.lisp")
(load "procura")
```

4. **Iniciar a Interface:** Para lançar o menu principal, execute a função no *Listener*:

```
(menu-principal)
```

4. Interface da aplicação

A interação com a aplicação é realizada através da consola (Listener).

4.1. Menu Principal

Ao iniciar, é apresentado o menu principal com duas opções. O utilizador deve introduzir o número da opção desejada seguido da tecla *Enter*.

```text
================================================
=  PROJETO SOLITÁRIO - INTELIGÊNCIA ARTIFICIAL =
================================================

Escolha uma opção:
1. Resolver um problema
2. Sair
Opção >
```

Se for a primeira execução, o sistema carregará automaticamente a lista de problemas a partir do ficheiro `problemas.dat`.

4.2. Seleção do Problema

Ao selecionar a opção **1**, o sistema lista a quantidade de problemas disponíveis e solicita a escolha de um deles pelo seu índice numérico.

```text
Escolha o numero do problema (1 a 6):
```

Deve introduzir um número inteiro válido (ex: `1` corresponde ao primeiro tabuleiro definido no ficheiro de dados).

4.3. Escolha do Algoritmo

Após a seleção do tabuleiro, é apresentado o menu de seleção de algoritmos de procura:

```
Algoritmo:
1. BFS (Largura)
2. DFS (Profundidade)
3. A* (Heuristica Base: Mobilidade)
4. A* (Heuristica Extra: Contagem Pinos)
Opcao >
```

Detalhes das Opções:

1. **1.BFS (Largura):** Executa a procura em largura. Garante a solução ótima (caminho mais curto), mas pode ser intensivo em memória.

2. **2.DFS (Profundidade):** Executa a procura em profundidade. Ao selecionar esta opção, será solicitado um parâmetro extra:

   ```
   Profundidade Maxima:
   ```

O utilizador deve definir o limite de profundidade para evitar ciclos infinitos ou caminhos excessivamente longos.

3. *3.A (Heurística Base):** Utiliza a heurística `h1-base`, que calcula o inverso da mobilidade (`1 / (o(x) + 1)`), privilegiando estados com mais movimentos possíveis.

4. *4.A (Heurística Extra):** Utiliza a heurística `h2-extra`, baseada na contagem de pinos restantes (`Pinos - 1`), estimando a distância ao objetivo final.

4.4. Sair

A opção **2** no menu principal encerra a execução da função e retorna o controlo ao Listener.

5. Input/Output

5.1. Ficheiro de Input (`problemas.dat`)

Os tabuleiros são carregados a partir do ficheiro `problemas.dat`. O formato utilizado é uma lista de listas (matriz), onde:

1. `1`: Representa um pino.
2. `0`: Representa uma casa vazia.

3. `nil`: Representa uma posição inválida (fora da cruz).

Exemplo de configuração no ficheiro:

```
(
(nil nil 0 0 0 nil nil)
(nil nil 0 0 1 nil nil)
( 0 0 0 0 1 1 0)
...
)
```

5.2. Output na Consola

Quando uma solução é encontrada, o programa exibe os detalhes completos na consola:

1. **Cabeçalho:** Indicação de sucesso e o número total de passos da solução.
2. **Estatísticas:**
    1. **Gerados:** Total de nós criados.
    2. **Expandidos:** Total de nós visitados.
    3. **Tempo:** Tempo de execução em milissegundos.
    4. **Métricas:** Valores de Penetrância e Ramificação Média.
3. **Passo a Passo:** A visualização gráfica de cada estado do tabuleiro desde o início até à solução, incluindo os valores de custo `g` e heurística `h` para cada nó.

Exemplo de Output:

```
--- SOLUCAO ENCONTRADA ---
Numero de passos: 5
Estatisticas: Gerados: 45 | Expandidos: 12 | Tempo: 0 ms
Penetrancia: 0.111 | Ramificacao: 1.450
Estado (g=0, h=0.500):
nil nil 0 0 0 nil nil
...
--------------------
```