Manual Técnico

IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Projeto: Resolução do Jogo **O Solitário (Peg Solitaire)**

1. Capa

- **Unidade Curricular:** Inteligência Artificial
- **Projeto:** Projeto 01 – Peg Solitaire Solver
- **Ano Letivo:** 2025/2026
- **Linguagem:** Common Lisp
- **Professor:** Filipe Mariano

Autores:

- Dinis Xavier | 202300148
- Rui Monteiro | 202300265
- Daniel Pais | 202200286

2. Arquitetura do Sistema

O sistema foi desenvolvido segundo uma **arquitetura modular**, separando claramente:

1. **Algoritmos de procura (genéricos)**
2. **Domínio do problema (Peg Solitaire)**
3. **Interface com o utilizador e gestão de ficheiros**

2.1 Módulos do Sistema

Módulo	Ficheiro	Responsabilidade
Interface / Front-end	`projeto.lisp`	Interação com o utilizador, menus, leitura/escrita de ficheiros, apresentação de resultados
Procura	`procura.lisp`	Implementação genérica dos algoritmos BFS, DFS, A* e IDA*
Domínio	`puzzle.lisp`	Representação do tabuleiro, operadores, função objetivo e heurísticas

2.2 Fluxo de Informação

1. O utilizador seleciona problema e algoritmo (`projeto.lisp`)
2. O estado inicial é passado ao algoritmo de procura (`procura.lisp`)
3. O algoritmo invoca funções do domínio (`puzzle.lisp`):
    - `gera-sucessores`
    - `objetivo?`

- heurísticas (`h1`, `h2`)

4. O resultado e estatísticas regressam ao front-end

5. A solução é apresentada no ecrã e registada em ficheiro

---

3. Entidades e sua Implementação

3.1 Entidades do Domínio

Entidade do domínio	Descrição
Tabuleiro	Grelha 7x7 em forma de cruz
Peça (pino)	Elemento que pode ser movido
Movimento	Salto horizontal ou vertical com captura
Estado	Configuração completa do tabuleiro

3.2 Entidades Programáticas

Tabuleiro

- Representado como **lista de listas**
- Valores possíveis:
  - `1` – peça
  - `0` – célula vazia
  - `nil` – posição inválida

Nó de Procura

Representado como uma lista:

```
(estado g pai h)
```

Onde:

- `estado` – tabuleiro
- `g` – custo desde a raiz (profundidade)
- `pai` – nó anterior
- `h` – valor heurístico

Funções associadas:

- `no-estado`, `no-g`, `no-pai`, `no-h`, `no-f`

---

4. Algoritmos e sua Implementação

4.1 Algoritmo Geral de Procura

Todos os algoritmos seguem o mesmo esquema base:

1. Inicializar lista de abertos com o estado inicial
2. Repetir enquanto existirem nós abertos:
    - Selecionar nó segundo a estratégia
    - Testar condição de objetivo
    - Gerar sucessores
    - Atualizar estruturas de controlo
3. Recolher estatísticas e tempo de execução

4.2 BFS – Procura em Largura

- Estrutura: fila FIFO
- Garante solução ótima
- Elevado consumo de memória

Características técnicas:

- Lista `abertos` ordenada por nível
- Lista `fechados` para evitar repetições
- Custo uniforme (g incrementado em 1)

4.3 DFS – Procura em Profundidade Limitada

- Estrutura: pilha LIFO
- Utiliza limite de profundidade
- Não garante solução ótima

Vantagens:

- Baixo consumo de memória

Desvantagens:

- Sensível ao valor do limite

4.4 A* – Procura Informada

- Seleção do nó com menor `f(n) = g(n) + h(n)`
- Usa lista de abertos ordenada
- Mantém custos mínimos nos fechados

Heurísticas modulares permitem comparar estratégias de avaliação

4.5 IDA* – Aprofundamento Iterativo com Heurística

- Combina DFS com limites crescentes de `f`
- Muito mais eficiente em memória do que A*
- Pode repetir expansões

---

5. Heurísticas Implementadas

5.1 Heurística h1

```
h1 = 1 / (número de peças móveis + 1)
```

- Não admissível
- Pouco informada
- Favorece estados com maior mobilidade

5.2 Heurística h2

```
h2 = número de peças - 1
```

- **Admissível e consistente**
- Baseada no número mínimo de jogadas necessárias
- Demonstra melhor desempenho em A* e IDA*

---

6. Métricas de Desempenho

O sistema recolhe automaticamente:

- Número de nós gerados
- Número de nós expandidos
- Fator de ramificação médio
- Penetrância
- Tempo de execução

Estas métricas são calculadas de forma genérica em `procura.lisp`.

---

7. Análise Crítica dos Resultados

Para a análise de resultados decidimos basearmo-nos **como exemplo nos resultados obtidos para o Problema 5**, conforme registados no ficheiro de resultados do sistema. Todos os algoritmos encontraram solução com **profundidade 10**, permitindo uma comparação direta e justa do desempenho.

Tabuleiro do Problema 5

```
```

```
  1 2 3 4 5 6 7
1     ○ ○ ○
2     ○ ○ ○
3 ○ ○ ○ ● ● ● ●
4 ○ ○ ○ ○ ● ● ●
5 ○ ○ ○ ● ● ● ●
6     ○ ○ ○
7     ○ ○ ○
```

7.1 Resultados Quantitativos (Problema 5)

Algoritmo	Nós Gerados	Nós Expandidos	Fator Ramificação Médio	Penetrância	Tempo (s)
BFS	1696	715	2.3706	0.005896	0.0780
DFS (lim=10)	445	428	2.1308	0.022472	0.0310
A* (h1)	1696	715	2.3706	0.005896	0.1580
A* (h2)	1038	492	2.1077	0.009634	0.0300
IDA* (h1)	9114	9113	1.0026	0.001097	2.8400
IDA* (h2)	5081	5080	1.0051	0.001968	0.0470

7.2 Análise por Algoritmo

BFS- apresenta o comportamento esperado de uma procura não informada: garante optimalidade, mas com um número elevado de nós gerados e expandidos. O fator de ramificação médio elevado (2.3706) reflete a exploração sistemática do espaço de estados, tornando o algoritmo pouco escalável.

DFS com limite 10- mostra-se bastante eficiente em termos de nós gerados e tempo de execução. No entanto, esta eficiência depende fortemente do limite escolhido. Neste caso, como o limite coincide com a profundidade da solução, o algoritmo encontrou uma solução ótima por coincidência, não sendo esse comportamento garantido em geral.

A* com h1- apresenta exatamente os mesmos valores de nós gerados e expandidos que o BFS, mas com um tempo de execução superior. Este resultado confirma que a heurística h1 é pouco informativa, não conseguindo orientar eficazmente a procura, degradando o desempenho para um comportamento semelhante ao de uma procura não informada.

A* com h2- evidencia uma melhoria significativa. O número de nós gerados (1038) e expandidos (492) é substancialmente inferior ao BFS e ao A*(h1), e o tempo de execução é o menor entre todos os algoritmos testados (0.030s). Estes resultados confirmam que a heurística h2 é admissível, consistente e informada.

IDA* com h1- apresenta um desempenho claramente inferior, com mais de 9000 nós gerados e um tempo de execução muito elevado (2.84s). Tal como no A*, a heurística h1 não fornece informação suficiente, levando a múltiplas iterações pouco eficazes.

IDA* com h2- reduz significativamente o número de nós gerados face ao IDA*(h1) e apresenta um tempo de execução competitivo (0.047s). Apesar de gerar mais nós do que o A*(h2), consome muito menos memória, validando a sua adequação para problemas de maior dimensão.

7.3 Comparação Global

Algoritmo	Ótimo	Eficiência Temporal	Eficiência de Memória
BFS	Sim	Média	Fraca
DFS	Não garantido	Boa	Muito boa
A* (h1)	Não garantido	Fraca	Média
A* (h2)	Sim	Excelente	Média
IDA* (h1)	Não garantido	Muito fraca	Excelente
IDA* (h2)	Sim	Boa	Excelente

7.4 Conclusões da Análise

Os resultados do Problema 5 demonstram claramente que a **qualidade da heurística** tem um impacto determinante no desempenho dos algoritmos informados. A heurística h2 permite obter soluções ótimas com um custo computacional muito inferior, tornando o A*(h2) a melhor opção quando existe memória disponível, e o IDA*(h2) a escolha mais adequada em cenários com restrições de memória.

7.2 Observações

- BFS torna-se impraticável em problemas grandes
- DFS depende fortemente do limite
- A* com h2 apresenta melhor compromisso
- IDA* é a melhor opção quando memória é limitada

---

8. Opções de Implementação Tomadas

- Separação rigorosa entre procura e domínio
- Representação funcional (sem efeitos laterais)
- Estruturas simples (listas) em vez de classes
- Estatísticas integradas nos algoritmos

Estas opções privilegiam clareza pedagógica em detrimento de desempenho máximo.

---

9. Limitações Técnicas e Desenvolvimento Futuro

9.1 Requisitos Não Implementados

- SMA*
- RBFS
- Interface gráfica

- Paralelização da procura

9.2 Melhoramentos Futuros

- Uso de tabelas hash para fechados
- Detecção avançada de simetrias
- Heurísticas mais informadas
- Refactoring para uso de CLOS

---

10. Considerações Finais

Este projeto demonstra uma implementação completa e extensível de algoritmos clássicos de procura aplicados a um problema real. A arquitetura modular facilita a manutenção, experimentação e evolução futura do sistema.

Fim do Manual Técnico