Projeto № 1
Época normal

PROBLEMA DO PUZZLE DOS PONTOS E DAS CAIXAS

O puzzle é constituído por um tabuleiro de $n * m$ caixas. Cada caixa é delimitada por 4 pontos entre os quais é possível desenhar um arco. Quando os quatro pontos a volta de uma caixa tiverem conectados por 4 arcos, a caixa é considerado fechada. O espaço da solução é portanto constituído por $n * m$ caixas, $(n + 1) * (m + 1)$ pontos e $((n + 1) * m) + (n * (m + 1))$ arcos. A figura 1 apresenta um exemplo do puzzle com 20 caixas (n=4 e m=5), com 15 arcos conectados e 4 caixas fechadas.

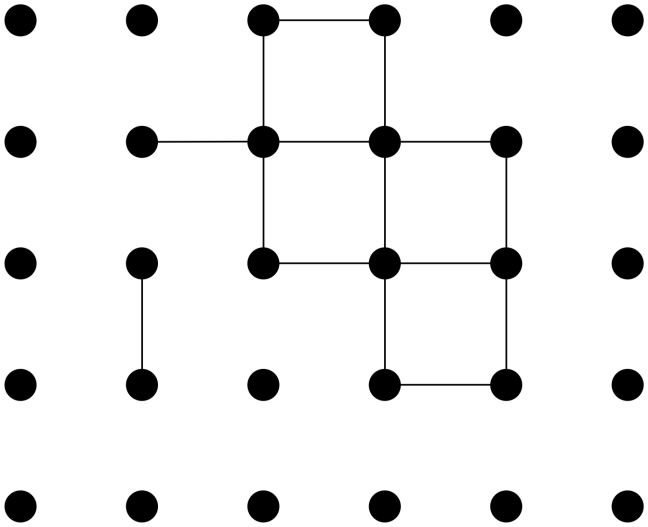


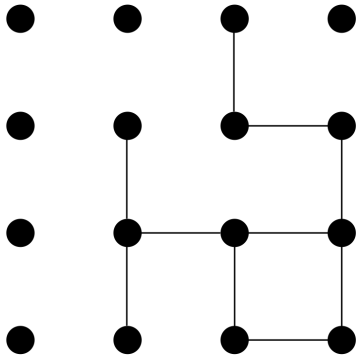
Figura 1: Ilustração de um exemplo do problema dos pontos e das caixas

Objetivo do puzzle

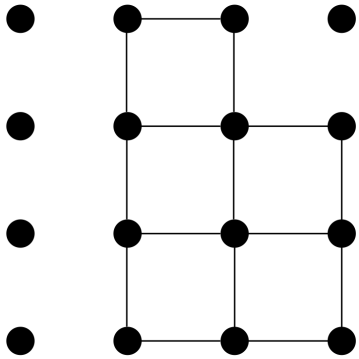
O objetivo do puzzle é fechar um determinado número de caixas a partir de uma configuração inicial do tabuleiro. Para atingir este objetivo, é possível desenhar um arco entre dois pontos adjacentes, na horizontal ou na vertical. Quando o número de caixas por fechar é atingido, o puzzle está resolvido. A resolução do puzzle consiste portanto em executar a sucessão de traços que permite chegar a um estado onde o número de caixas por fechar é alcançado (ver exemplo da figura 2).

OBJETIVOS DO PROJETO

Pretende-se desenvolver um programa, em LISP, para indicar a sequência de passos que conduzem de uma posição inicial do problema até a uma posição final em que o número de caixa fechadas do



(a) Ilustração de um estado inicial do problema, com um objetivo de 5 caixas fechadas por alcançar



(b) Estado final para o mesmo problema com 5 caixas fechadas

Figura 2: Exemplo de problema e a sua resolução

problema foi alcançado. A figura 2 apresenta um exemplo com o estado inicial do puzzle (a) com um objetivo de 5 caixas fechadas e a solução do mesmo (b). Neste exemplo, foram fechadas 4 caixas, além da caixa já fechada no estado inicial. Para resolver este puzzle, foi preciso colocar 5 arcos: dois na vertical e três na horizontal.

OPERADORES

Colocação

Existem dois operadores para a resolução do problema:

1. a colocação de um arco entre dois pontos do tabuleiro na horizontal e
2. a colocação de um arco entre dois pontos do tabuleiro na vertical.

Os operadores podem colocar arcos entre pontos adjacentes (apenas). Estes operadores podem ser aplicados em qualquer lugar do tabuleiro onde não existe ainda um arco.

NOTAÇÃO DO PROBLEMA

Tabuleiro

O tabuleiro é representado sob a forma de uma lista composta por 2 listas.

- A primeira lista representa os arcos horizontais. Este lista é composta por $n + 1$ listas, cada uma delas composta por m elementos. O valor destes elementos é [NIL] se não existir traço nessa posição e [T] se existir um arco.
- A segunda lista representa os arcos verticais. Esta lista é composta por $m + 1$ listas, cada uma delas composta por n elementos. O valor destes elementos é [NIL] se não existir traço nessa posição e [T] se existir um arco.

A Figura 3 ilustra uma representação em Lisp de tabuleiro vazio de dimensão 3 x 3.

```
1  (
2   ((NIL NIL NIL) (NIL NIL NIL) (NIL NIL NIL) (NIL NIL NIL))
3   ((NIL NIL NIL) (NIL NIL NIL) (NIL NIL NIL) (NIL NIL NIL))
4  )
```

Figura 3: Exemplo de representação em LISP do problema do puzzle dos pontos e das caixas, com um tabuleiro vazio de dimensão 3 x 3.

A Figura 4 ilustra uma representação em Lisp de um tabuleiro de dimensão 3 x 3 com a caixa no centro do tabuleiro fechada por 4 arcos.

```
1  (
2  ((NIL NIL NIL) (NIL T NIL) (NIL T NIL) (NIL NIL NIL))
3  ((NIL NIL NIL) (NIL T NIL) (NIL T NIL) (NIL NIL NIL))
4  )
```

Figura 4: Exemplo de representação em LISP do problema do puzzle dos pontos e das caixas, com um tabuleiro de dimensão 3 x 3 com uma caixa fechada.

Movimentos

Um movimento de colocação é considerado possível se a posição de destino pertence ao tabuleiro E se ainda não existe um arco nessa posição.

Cada movimento de colocação é composto por 3 parâmetros:

- Para a colocação do arco na horizontal:
  - o primeiro parâmetro representa o número da linha onde desenhar o arco (entre 1 e n + 1),

  - o segundo representa o número da coluna onde desenhar o arco (entre 1 e m),
  - o terceiro representa o tabuleiro ao qual aplicar o operador.

- Para a colocação do arco na vertical,
  - o primeiro parâmetro representa o número da coluna onde desenhar o arco (entre 1 e m + 1),
  - o segundo representa o número da linha onde desenhar o arco (entre 1 e n),
  - o terceiro representa o tabuleiro ao qual aplicar o operador.

Por exemplo, se pretendemos colocar um arco na vertical no lado direito da terceira caixa da primeira linha no tabuleiro da figura 4, o operador a aplicar será:

```
1  (arco-vertical 4 1 '(
2     ((NIL NIL NIL) (NIL T NIL) (NIL T NIL) (NIL NIL NIL))
3     ((NIL NIL NIL) (NIL T NIL) (NIL T NIL) (NIL NIL NIL))
4     ))
```

Figura 5: Exemplo de aplicação do operador **arco-vertical** ao estado do problema da figura 4

O estado do problema resultante está representado na figura 6.

```
1     (
2     ((NIL NIL NIL) (NIL T NIL) (NIL T NIL) (NIL NIL NIL))
3     ((NIL NIL NIL) (NIL T NIL) (NIL T NIL) (T NIL NIL))
4     )
```

Figura 6: Resultado da aplicação do operador **arco-vertical** para a 4ª coluna e 1ª linha do estado do problema representado na figura 4.

Estrutura do programa

O programa deverá estar dividido em três partes, cada uma num ficheiro diferente:

1. uma parte para implementar os métodos de procura independentes do domínio de aplicação,
2. outra para implementar a resolução do problema, incluindo a definição dos operadores e heurísticas, específicos do domínio de aplicação,
3. e a terceira parte para fazer a interação com o utilizador e para proceder à escrita e leitura de ficheiros.

Enquanto a primeira parte do programa deverá ser genérica para qualquer problema que possa ser resolvido com base no método de procura selecionado, a segunda parte é específica do caso particular em causa.

O projeto deverá apresentar um estudo comparativo do comportamento dos três métodos: **procura em largura**, **procura em profundidade** e **A***. Para além destas três formas de procura cada grupo deve programar, aplicar e estudar uma das seguintes estratégias **Simplified Memory A* (SMA*)**, **Interactive Deepening A* (IDA*)** ou **Recursive BestFirst Search (RBFS)** consoante indicação prévia do algoritmo que cabe a cada um.

No caso dos métodos informados, o programa deverá utilizar funções heurísticas modulares, i.e. que possam ser colocadas ou retiradas do programa de procura como módulos. As heurísticas não devem estar embebidas de forma rígida no programa de procura. Exige-se a utilização de duas heurísticas, uma fornecida no fim do presente documento e outra desenvolvida pelos alunos. O projeto deverá incluir a implementação de cada um dos métodos, de forma modular, permitindo que o utilizador escolha qualquer um deles, conjuntamente com os seus parâmetros (heurística, profundidade...) para a resolução de um dado problema.

EXPERIÊNCIAS

Pretende-se que o projeto estude, para cada problema dado, o desempenho de cada uma das heurísticas, apresentando, em relação a cada ensaio, dados estatísticos sobre a sua eficiência, nomeadamente o fator de ramificação média, o número de nós gerados, número de nós expandidos, a penetrância e o tempo de execução. Os projetos deverão apresentar as soluções de TODOS os problemas abaixo indicados nos anexos, mediante um ficheiro de resultados produzido automaticamente pelo programa, sendo descontado 0,5 valor por cada problema não resolvido. No caso de ser apresentada a solução, mas não o estudo de desempenho das heurísticas o desconto é de apenas 0,2 valor por cada caso. Estes problemas deverão estar num ficheiro **problemas.dat**, utilizando a notação atrás indicada. O último problema (identificado pela letra [g]) será apresentado durante a avaliação oral e inserido no ficheiro **problemas.dat** para verificar o funcionamento do projeto.

HEURÍSTICAS

Sugere-se usar como heurística de base, uma heurística que privilegia os tabuleiros com o maior número de caixas fechadas. Para um determinado tabuleiro x:

$$h(x) = o(x) - c(x) - 1 \tag{1}$$

em que:

- $o(x)$ é o objetivo para esse tabuleiro: o número de caixas a fechar no tabuleiro x,
- $c(x)$ é o número de caixas já fechadas do tabuleiro x.

Por exemplo, no estado do problema apresentado na figura 7, $o(x) = 3$, $c(x) = 1$ e $h(x) = 1$, enquanto que no estado do problema apresentado na figura 8, $o(x) = 7$, $c(x) = 5$ e $h(x) = 1$.

Esta heurística pode ser melhorada para refletir de forma mais adequada o conhecimento acerca do puzzle e assim contribuir para uma maior eficiência dos algoritmos de procura informados. Como referido anteriormente, além da heurística acima sugerida, deve ser definida pelo menos uma segunda heurística que deverá melhorar o desempenho dos algoritmos de procura informados em relação a primeira fornecida.

GRUPOS

Os projetos deverão ser realizados em grupos de duas pessoas, sendo contudo sempre sujeitos a avaliação oral individual e a um teste individual para confirmação da capacidade de compreensão e desenvolvimento de código em LISP. Em casos excepcionais poderá haver trabalhos realizados individualmente.

Segunda Parte do projeto

O projeto incluí uma segunda parte opcional que permite atribuir um bónus aos alunos que conseguirem implementar esta segunda parte. Consiste em implementar o mesmo projeto, usando uma nova estrutura de dados para representar o estado do problema. A estrutura a utilizar só será comunicada no dia seguinte ao dia de entrega do projeto (14/12) e terá de ser entregue 48h depois da sua comunicação (16/12). Quem conseguir replicar o funcionamento da primeira versão do projeto terá um bónus de até 3 valores na nota do projeto. Estes 3 valores estão limitados a 30% do valor dado aos primeiros 4 critérios da grelha de avaliação, i.e. ao código produzido na primeira parte. Recomenda-se fortemente a utilização de tipos abstratos de dados, por forma a minimizar a dimensão das alterações causadas pela modificação da sua representação interna.

DATAS

Entrega da primeira parte do projeto : 14 de dezembro de 2016, até as 08:00 da manhã.
Entrega da segunda do projeto : 16 de dezembro de 2016, até as 08:00 da manhã.

DOCUMENTAÇÃO A ENTREGAR

A entrega do projeto e sua documentação deverá ser feita através do Moodle, na zona do evento "entrega do primeiro projeto". Todos os ficheiros a entregar deverão ser devidamente arquivados num ficheiro comprimido (zip com um tamanho máximo de 5Mb), até à data acima indicada. O nome do arquivo deve seguir a estrutura seguinte:
$nomeAluno1\_numeroAluno1\_nomeAluno2\_numeroAluno2\_P1$
Para além de entregar os ficheiros de código, é necessário elaborar 2 manuais (o manual de utilizador e o manual técnico) com a estrutura habitual, que deverão constar em dois ficheiros escritos em formato PDF. Não será preciso entregar uma versão impressa dos relatórios.

Ficheiros com Código

Os ficheiros de código devem ser devidamente comentados e organizados da seguinte forma:

projeto.lisp Carrega os outros ficheiros de código, escreve e lê de ficheiros e trata da interação com o utilizador.

puzzle.lisp Código relacionado com o problema.

procura.lisp Deve conter a implementação de:

1. Algoritmo genérico de procura
2. Algoritmo de procura de Largura Primeiro
3. Algoritmo de procura do Profundidade Primeiro
4. Algoritmo de procura do Melhor Primeiro
5. Um dos algoritmos SMA*; IDA* ou RBFS

Manuais

ManualTecnico.pdf O Manual Técnico deverá conter o algoritmo geral e por partes devidamente comentado; descrição dos objetos que compõem o projeto, incluindo dados e procedimentos; identificação das limitações e opções técnicas. Deverá ser apresentada uma análise critica dos resultados das execuções do programa onde deverá transparecer a compreensão das limitações do projeto. Deverão usar uma análise comparativa do conjunto de execuções do programa para cada algoritmo e cada problema, permitindo verificar o desempenho de cada algoritmo e das heurísticas. Deverá, por fim, apresentar a lista dos requisitos do projeto (listados neste documento) que não foram implementados.

ManualUtilizador.pdf O Manual do Utilizador deverá conter a identificação dos objetivos do programa, e descrição geral do seu funcionamento; explicação da forma como se usa o programa (acompanhada de exemplos); descrição da informação necessária e da informação produzida (ecrã/teclado e ficheiros); limitações do programa, do ponto de vista do utilizador, de natureza não técnica.

AVALIAÇÃO

Tabela 1: Grelha de classificação.

Procura em profundidade e largura	4
Procura com A* e heurística dada	3
Implementação de nova heurística	1
Implementação do algoritmo adicional	2
Resolução dos problemas a) a g)	3
Qualidade da Codificação	1
Relatórios (Utilizador e Técnico)	2
Avaliação escrita de LISP	4
Total	20
Bónus	até 3

O valor máximo da nota é 20 valores, já com a integração do valor do bónus. O valor do bónus - 3 valores - está limitado a 30% do valor dado aos primeiros 4 critérios da grelha de avaliação.
A avaliação do projeto levará em linha de conta os seguintes aspectos:

- Data de entrega final – Existe uma tolerância de uma semana em relação ao prazo de entrega, com a penalização de um valor por cada dia de atraso. Após uma semana de atraso a nota do projeto será zero.
- Correção processual da entrega do projeto - (Moodle; manuais no formato correto). Anomalias processuais darão origem a uma penalização que pode ir até 3 valores.
- Qualidade técnica - Objetivos atingidos; Código correto; Facilidade de leitura e manutenção do programa; Opções técnicas corretas.
- Qualidade da documentação - Estrutura e conteúdo dos manuais que acompanham o projeto.
- Avaliação oral - Eficácia e eficiência da exposição; compreensão das limitações e possibilidades de desenvolvimento do programa. Nesta fase poderá haver lugar a uma revisão total da nota de projeto.
- Avaliação escrita de LISP - Confirmação da capacidade de compreensão e desenvolvimento de código em LISP.

Com este projeto pretende-se motivar o paradigma de programação funcional. A utilização de variáveis globais, de instruções de atribuição do tipo set, setq, setf, de ciclos, de funções destrutivas ou de quaisquer funções com efeitos laterais é fortemente desincentivada dado que denota normalmente uma baixa qualidade técnica.

ANEXOS - PROBLEMAS

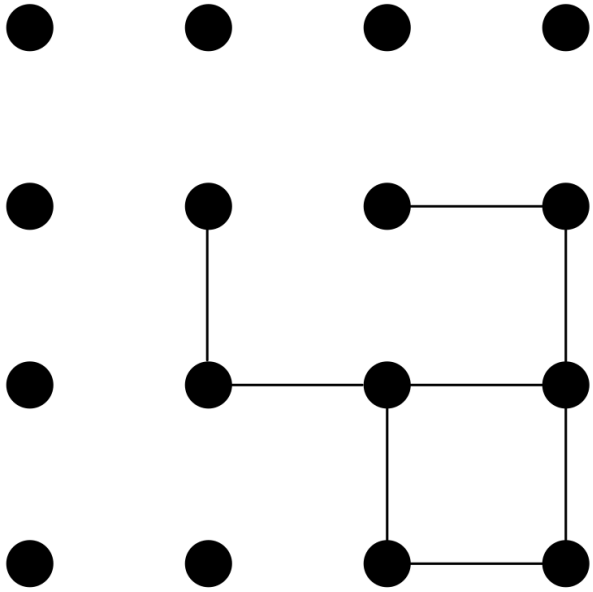


Figura 7: Problema a) com um objetivo de 3 caixas

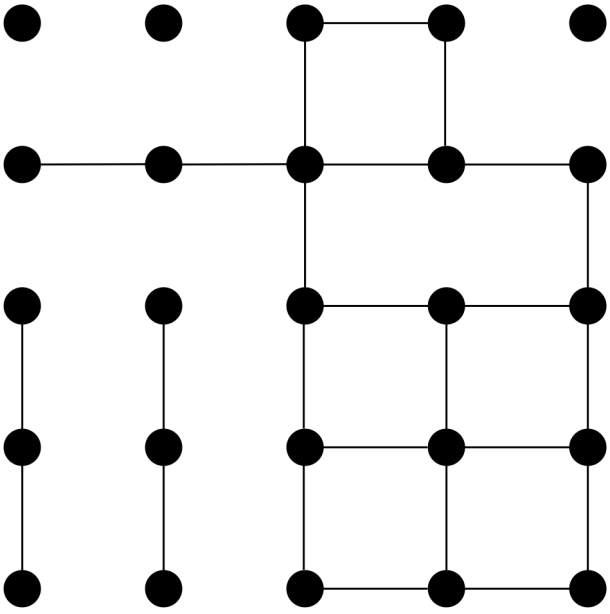


Figura 8: Problema b) com um objetivo de 7 caixas

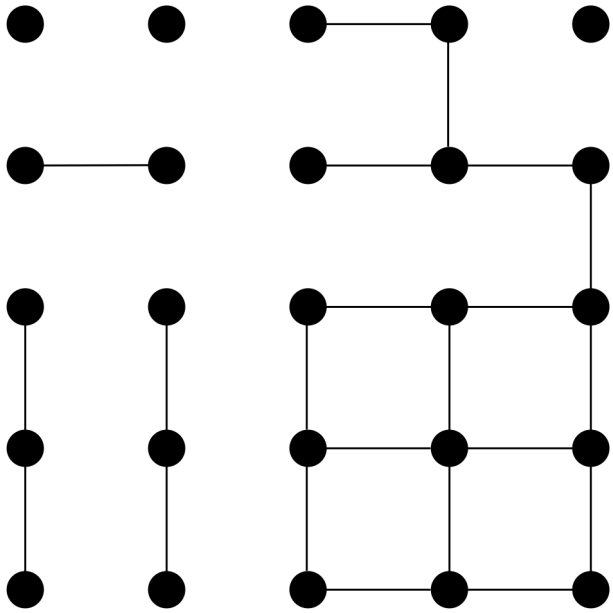


Figura 9: Problema c) com um objetivo de 10 caixas

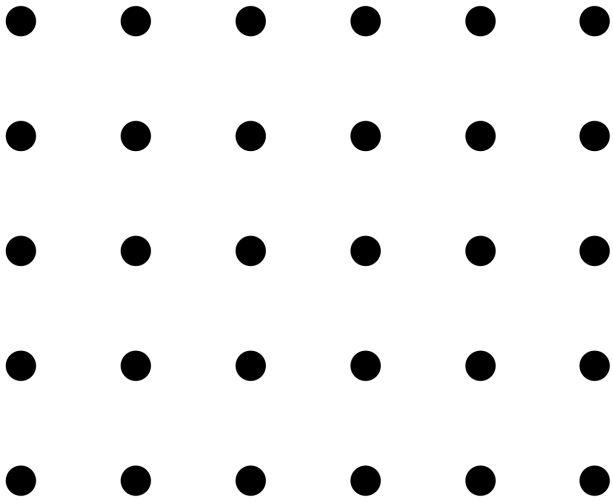


Figura 10: Problema d) com um objetivo de 10 caixas

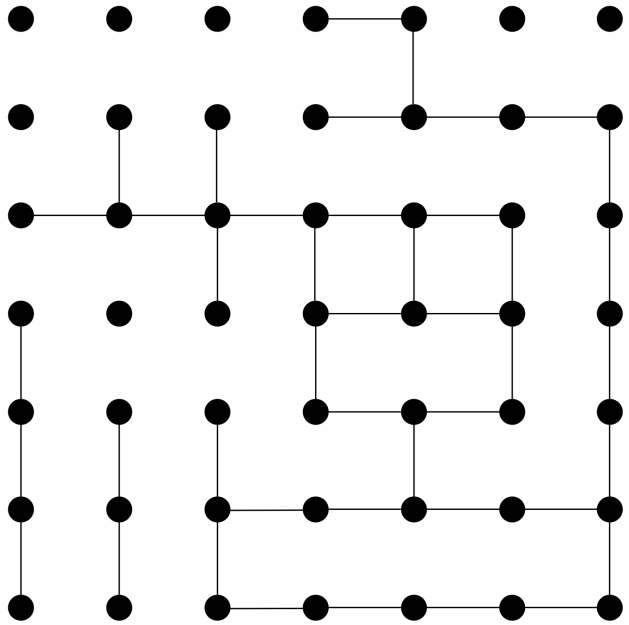


Figura 11: Problema e) com um objetivo de 20 caixas

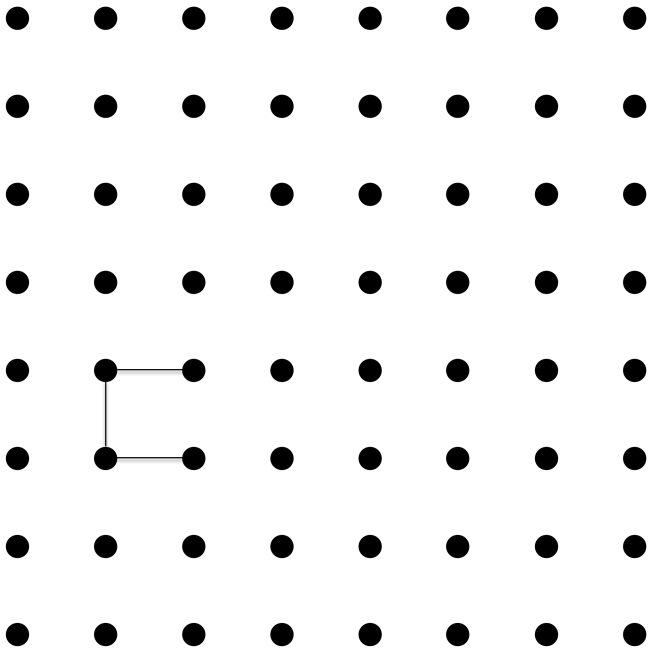


Figura 12: Problema f) com um objetivo de 45 caixas