PROCURA EM ESPAÇOS DE ESTADOS

Luís Morgado

2024

PROCURA MELHOR-PRIMEIRO (*BEST-FIRST*)

- Utiliza uma função ***f*(*n*)** para **avaliação** de cada nó ***n*** gerado
  - ***f*(*n*)** ≥ 0
  - ***f*(*n*)** representa uma **avaliação** do **custo** da solução através do nó ***n***
    - Quanto menor o valor de *f*(*n*) mais promissor é o nó *n*
- A fronteira de exploração é **ordenada por ordem crescente** de ***f*(*n*)**
- ***f*(*n*)** pode ter diferentes formas
  - Baseada no custo dos nós explorados
  - Baseada em **estimativas** de custo
    - Com base em **heurísticas** – métodos expeditos de estimação de valores ou de resolução de problemas

PROCURA MELHOR-PRIMEIRO (*BEST-FIRST*)

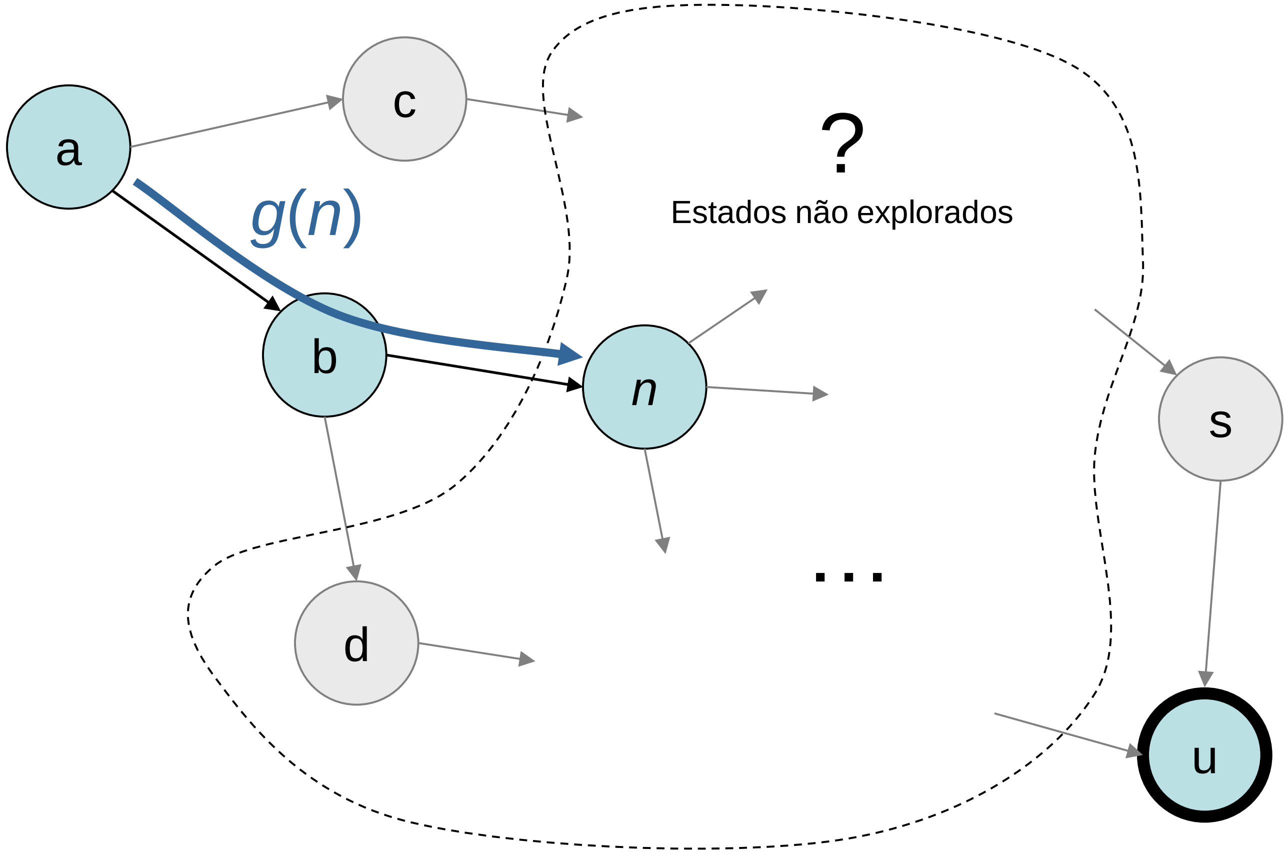
- **Variantes principais de $f(n)$**

  $f(n) = g(n)$

  - **Procura de *Custo Uniforme***
    - Minimização de custo acumulado até cada nó explorado

PROCURA DE CUSTO UNIFORME

f*(*n*) = *g*(*n*)**, representa uma **avaliação** do **custo *g*(*n*)** do percurso até ao nó ***n



g*(*n*)**: custo do percurso até ***n

Custo calculado para a partir do estado inicial atingir o estado ***n***

MÉTODOS DE PROCURA

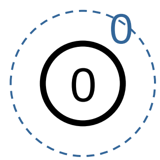
Procura de *Custo Uniforme*

- Estratégia de controlo
  - Explorar primeiro os nós com **menor custo**
    - Fronteira ordenada por custo associado a cada nó: ***f*(*n*) = *g*(*n*)**

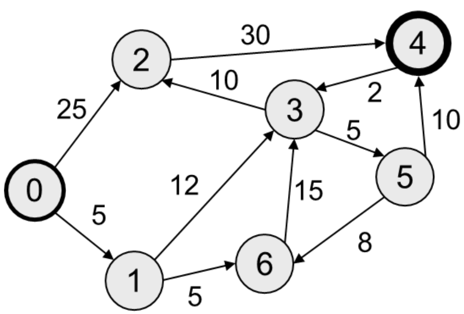
Fronteira de exploração [ ]

[0:0] Estado : Nó

Árvore de Procura



Fronteira de exploração

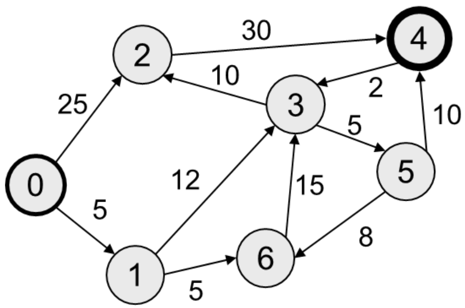


Grafo do
Espaço de Estados

MÉTODOS DE PROCURA

Procura de *Custo Uniforme*

- Estratégia de controlo
  - Explorar primeiro os nós com **menor custo**
    - Fronteira ordenada por custo associado a cada nó



Grafo do
Espaço de Estados

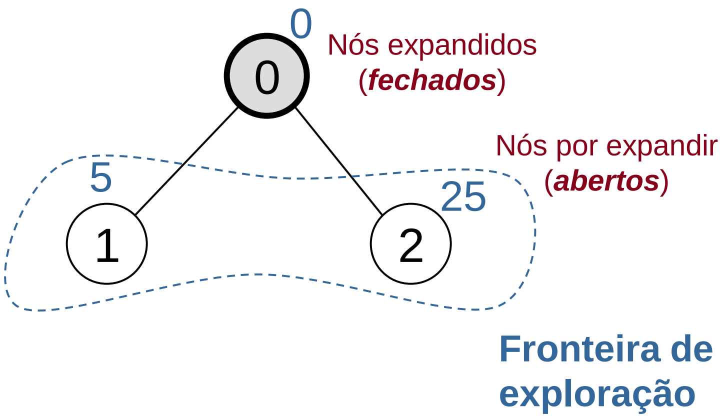
Fronteira de exploração [ ]

[0:0]

0:0 [ ]

[**1:5, 2:25**]

Árvore de Procura



MÉTODOS DE PROCURA

Procura de *Custo Uniforme*

- Estratégia de controlo
  - Explorar primeiro os nós com **menor custo**
    - Fronteira ordenada por custo associado a cada nó

Fronteira de exploração [ ]

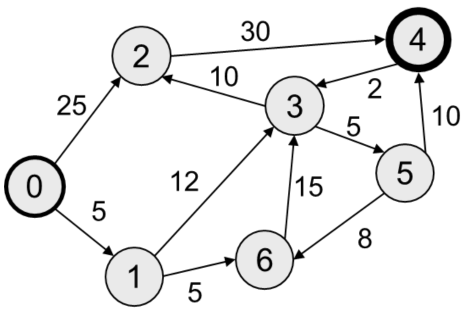
[0:0]

0:0 [ ]

[**1:5, 2:25**]

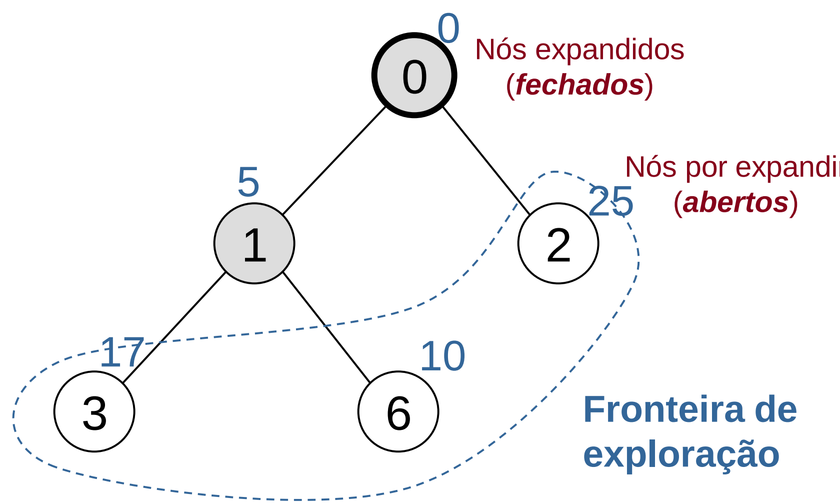
1:5 [2:25]

[**6:10, 3:17,** 2:25]



Grafo do
Espaço de Estados

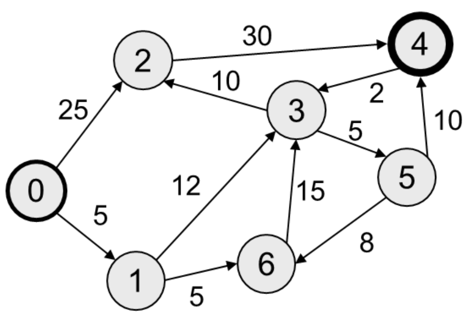
Árvore de Procura



MÉTODOS DE PROCURA

Procura de *Custo Uniforme*

- Estratégia de controlo
  - Explorar primeiro os nós com **menor custo**
    - Fronteira ordenada por custo associado a cada nó



Grafo do
Espaço de Estados

Fronteira de exploração [ ]

[0:0]

0:0 [ ]

[1:5, 2:25]

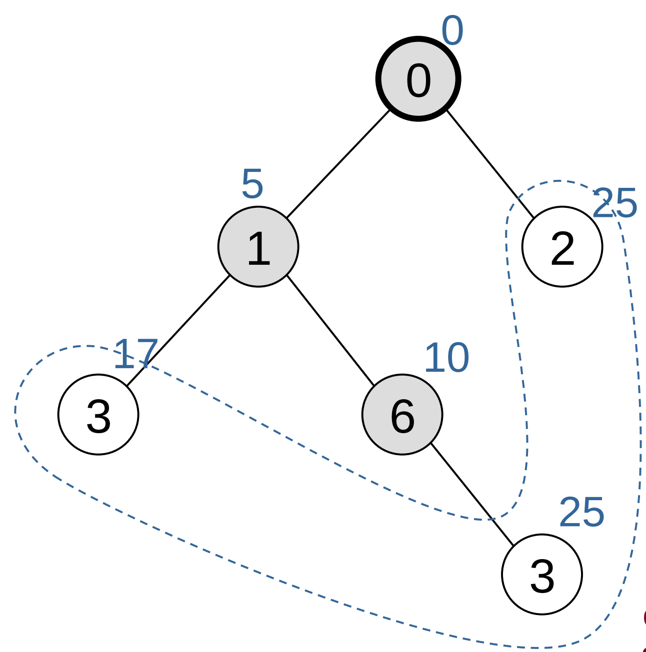
1:5 [2:25]

[**6:10, 3:17,** 2:25]

6:10 [3:17, 2:25]

[**3:17, 3:25,** 2:25]

Árvore de Procura



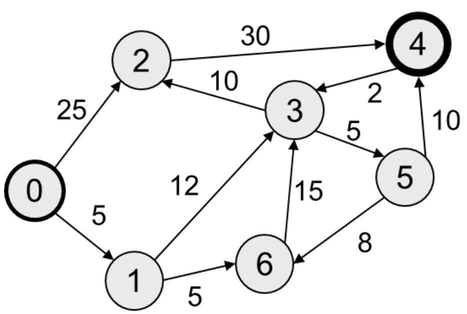
Sem eliminação de nós referentes ao mesmo estado

Fronteira de exploração

MÉTODOS DE PROCURA

Procura de *Custo Uniforme*

- Estratégia de controlo
  - Explorar primeiro os nós com **menor custo**
    - Fronteira ordenada por custo associado a cada nó



Grafo do
Espaço de Estados

Fronteira de exploração [ ]

[0:0]

0:0 [ ]

[**1:5, 2:25**]

1:5 [2:25]

[**6:10, 3:17,** 2:25]

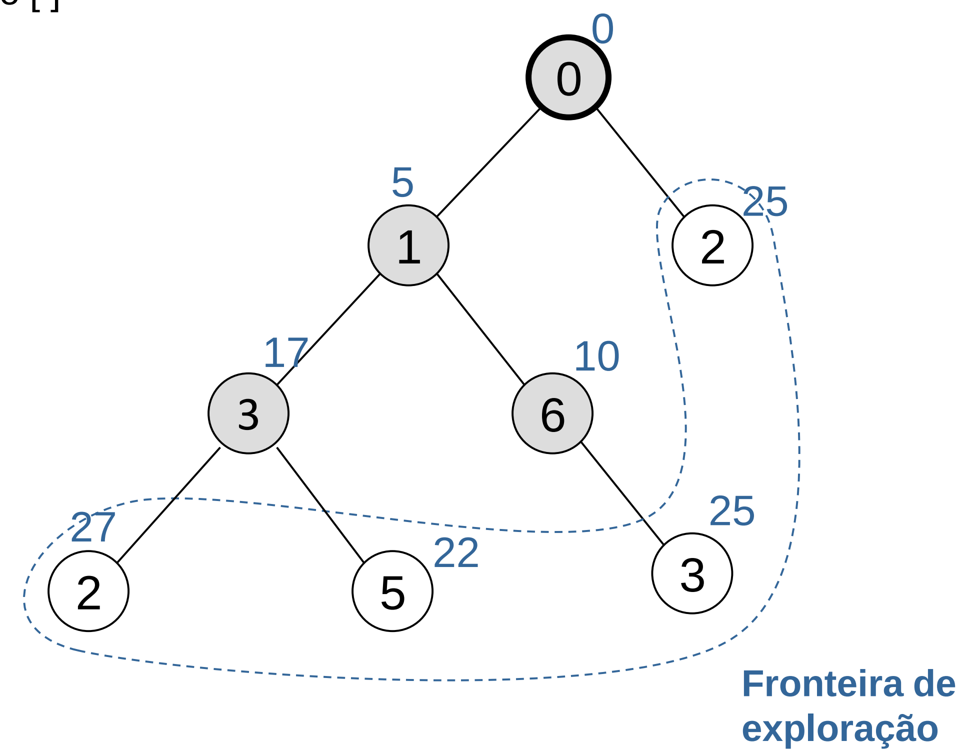
6:10 [3:17, 2:25]

[**3:17, 3:25,** 2:25]

3:17 [3:25, 2:25]

[**5:22,** 3:25, 2:25, **2:27**]

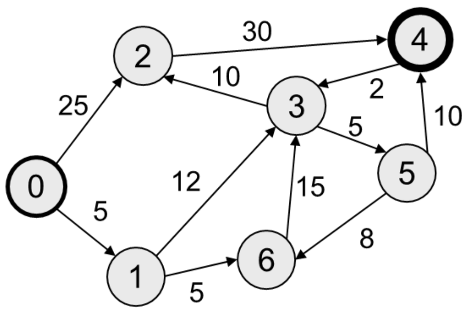
Árvore de Procura



MÉTODOS DE PROCURA

Procura de *Custo Uniforme*

- Estratégia de controlo
  - Explorar primeiro os nós com **menor custo**
    - Fronteira ordenada por custo associado a cada nó



Grafo do
Espaço de Estados

Fronteira de exploração [ ]

[0:0]

0:0 [ ]

[**1:5, 2:25**]

1:5 [2:25]

[**6:10, 3:17,** 2:25]

6:10 [3:17, 2:25]

[**3:17, 3:25,** 2:25]

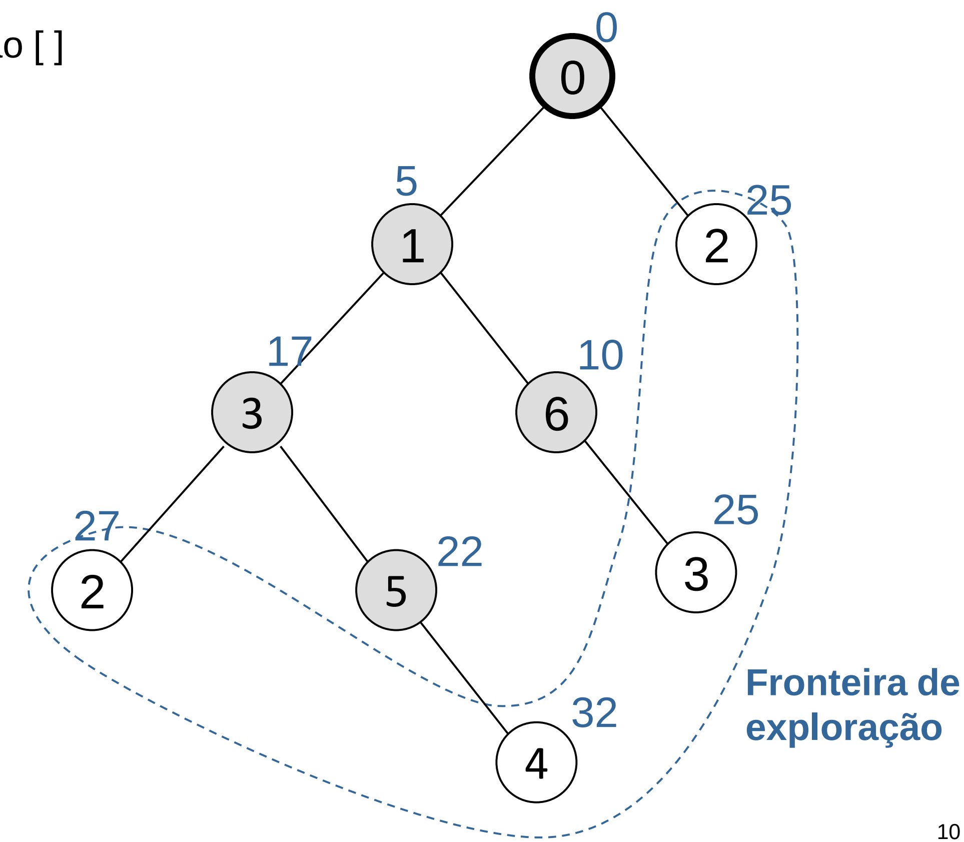
3:17 [3:15, 2:25]

[**5:22, 3:25,** 2:25, 2:27]

5:22 [3:25, 2:25, 2:27]

[3:25, 2:25, 2:27, **4:32**]

Árvore de Procura



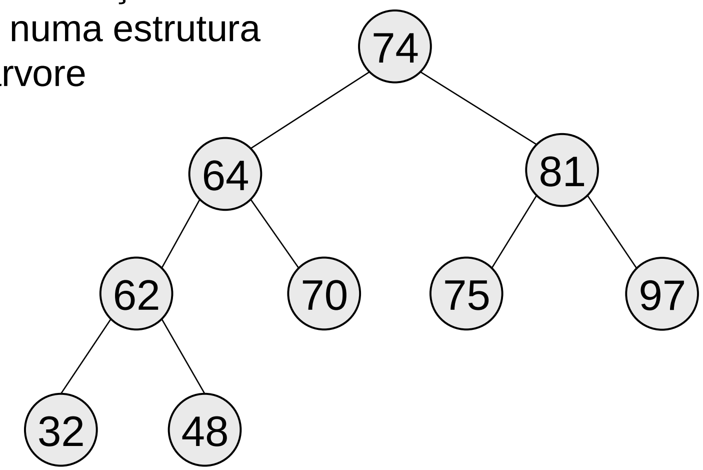
Fronteira com relação de ordem entre nós

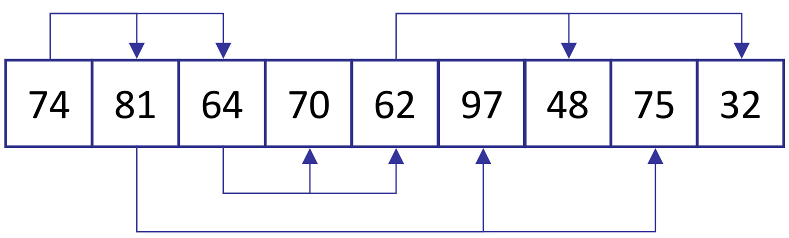
(*Prioridade de processamento*)

Estrutura de dados *PriorityQueue*

(*Fila com prioridade*)

Representação com base numa estrutura em árvore





Representação com base numa lista

Linguagem Python

Biblioteca **heapq (*HeapQueue*)**

Possibilita o acesso a listas com relação de ordem entre os elementos

heapq.**heappush**(*heap*, *item*)

Insere *item* na pilha *heap*

heapq.**heappop**(*heap*)

Remove e retorna da pilha *heap* por ordem de prioridade

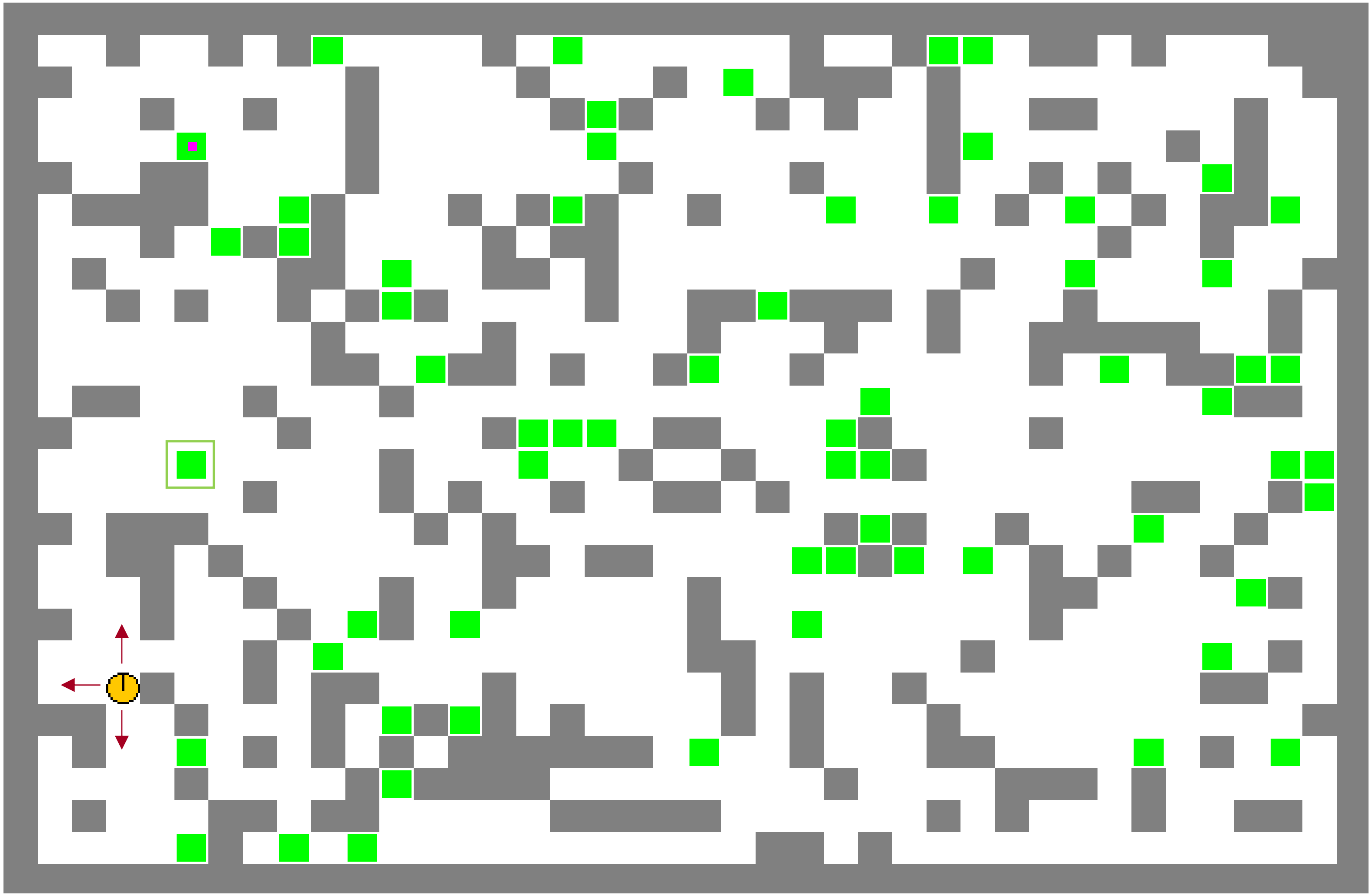
Utiliza a relação de ordem definida para os objectos manipulados

PROCURA EM ESPAÇOS DE ESTADOS

- **Métodos de Procura não Informada**
  - Estratégias de exploração do espaço de estados (*controlo da procura*) não tiram partido de *conhecimento do domínio do problema* para ordenar a fronteira de exploração
  - Procura não guiada
    - Exploração **exaustiva** do espaço de estados

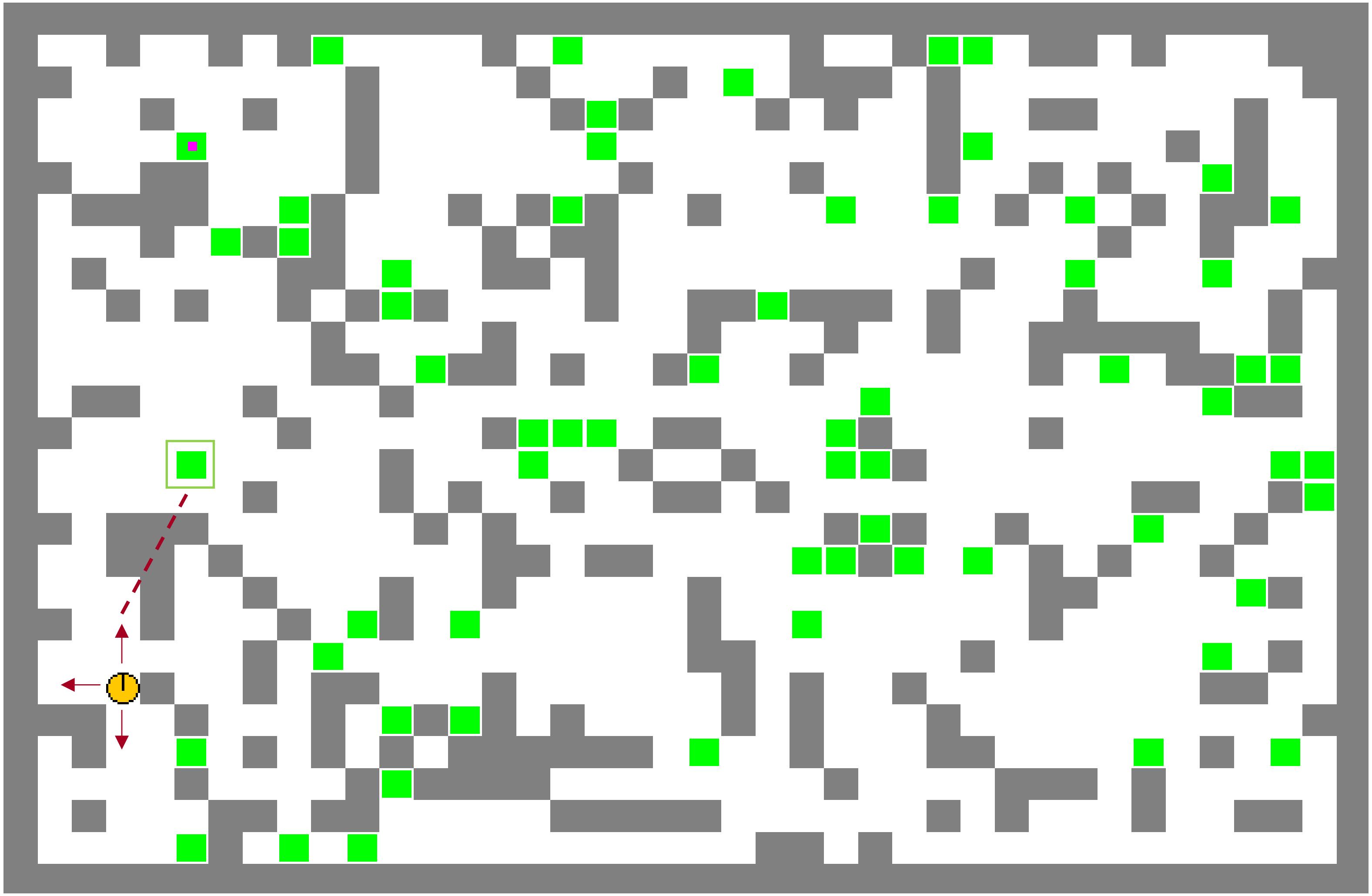
- **Métodos de Procura Informada**
  - Estratégias de exploração do espaço de estados (*controlo da procura*) tiram partido de *conhecimento do domínio do problema* para ordenar a fronteira de exploração
  - Procura guiada
    - Exploração **selectiva** do espaço de estados

Qual a melhor acção a realizar em cada situação?



Qual a melhor acção a realizar em cada situação?

Utilização de conhecimento do domínio do problema, por exemplo, distância ao objectivo



FUNÇÃO HEURÍSTICA *h*(*n*)

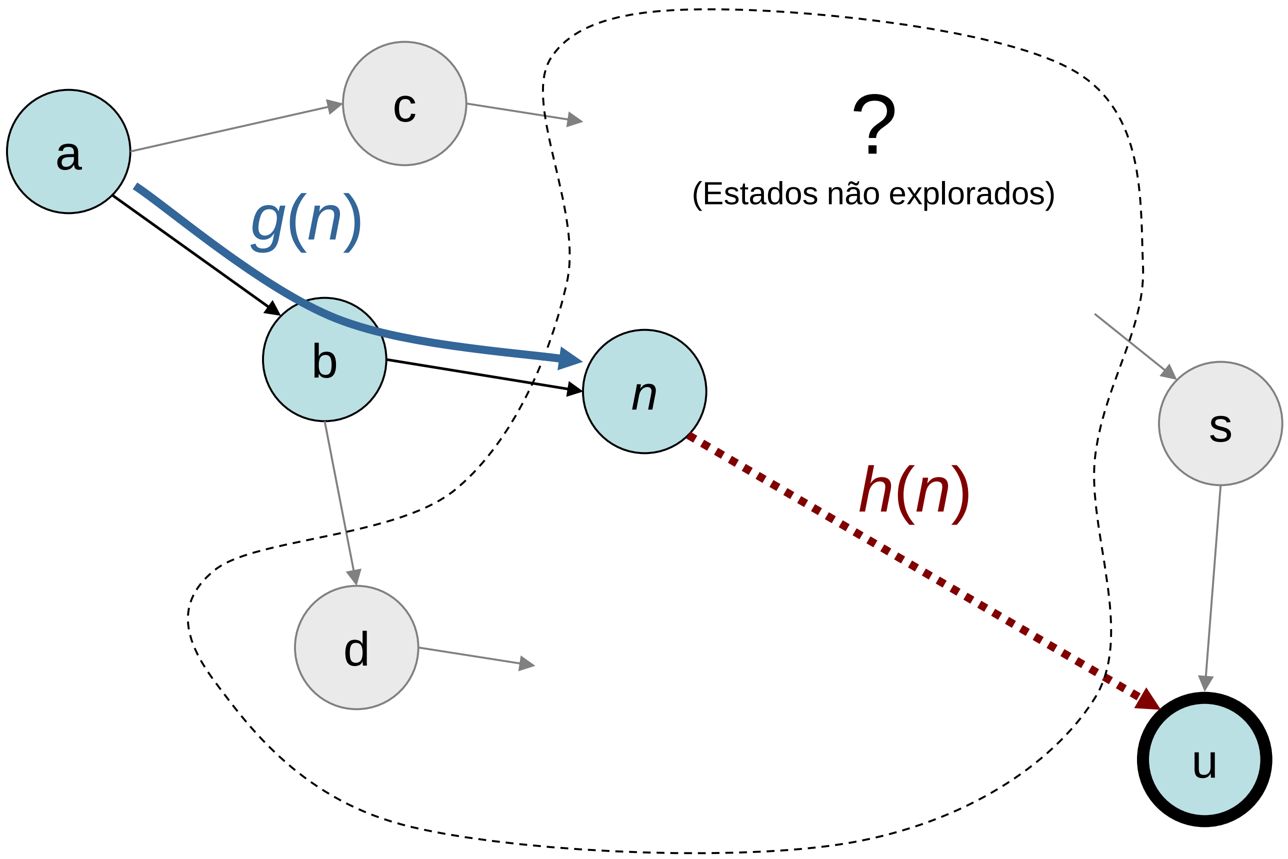
- Representa uma **estimativa** do custo do percurso desde o nó ***n*** até ao nó objectivo
  - **Pode não corresponder ao valor real**

- Reflecte conhecimento acerca do domínio do problema, para guiar a procura

- O seu valor é independente do percurso até ***n***
  - Depende apenas de:
    - **Estado** associado a ***n***
    - **Objectivo**

MÉTODOS DE PROCURA INFORMADA

***f*(*n*)** baseada em dois tipos de funções de avaliação de custo ***g*(*n*)** e ***h*(*n*)**



g*(*n*)**: Custo do percurso até ***n

Custo calculado para a partir do estado inicial atingir o estado ***n***

h*(*n*)**: Estimativa de custo de ***n até ao objectivo

Estimativa de custo (heurística) para a partir do estado ***n*** atingir o estado objectivo ***u*** (sem que o percurso respectivo tenha sido explorado)

PROCURA MELHOR-PRIMEIRO (*BEST-FIRST*)

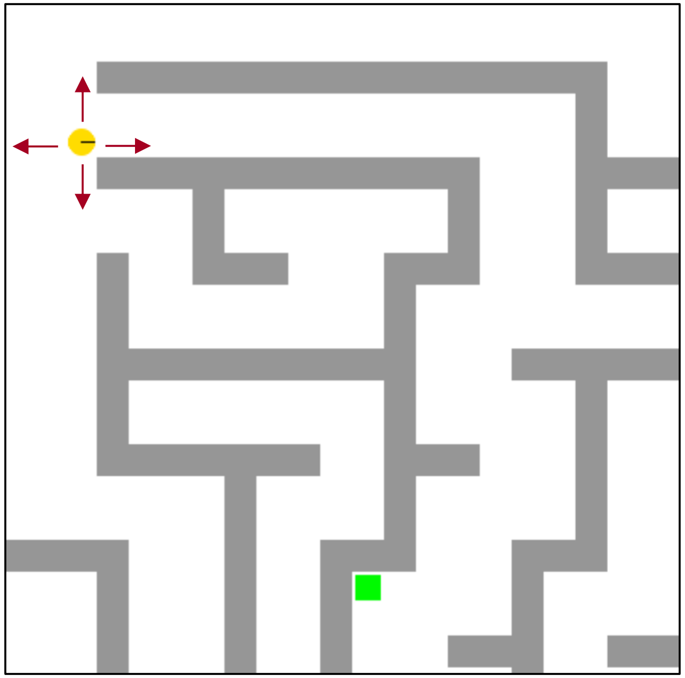
- **Variantes principais de $f(n)$**
  - **$f(n) = g(n)$**
    - **Procura de Custo Uniforme**
      - Minimização de custo acumulado até cada nó explorado
      - Não tira partido de conhecimento do domínio do problema expresso através da função $h(n)$
  - **$f(n) = h(n)$**
    - **Procura Sôfrega (*Greedy Search*)**
      - Minimização da estimativa de custo para atingir o objectivo
      - Não tem em conta o custo do percurso explorado
      - Soluções sub-óptimas
  - **$f(n) = g(n) + h(n)$**
    - **Procura A* (heurística admissível)**
      - Minimização de custo global
        (custo acumulado até ao nó ***n*** + custo estimado até ao objectivo)

MÉTODOS DE PROCURA INFORMADA

Problema

Sendo as *heurísticas* **estimativas** de custo, podem induzir em erro levando a soluções sub-óptimas

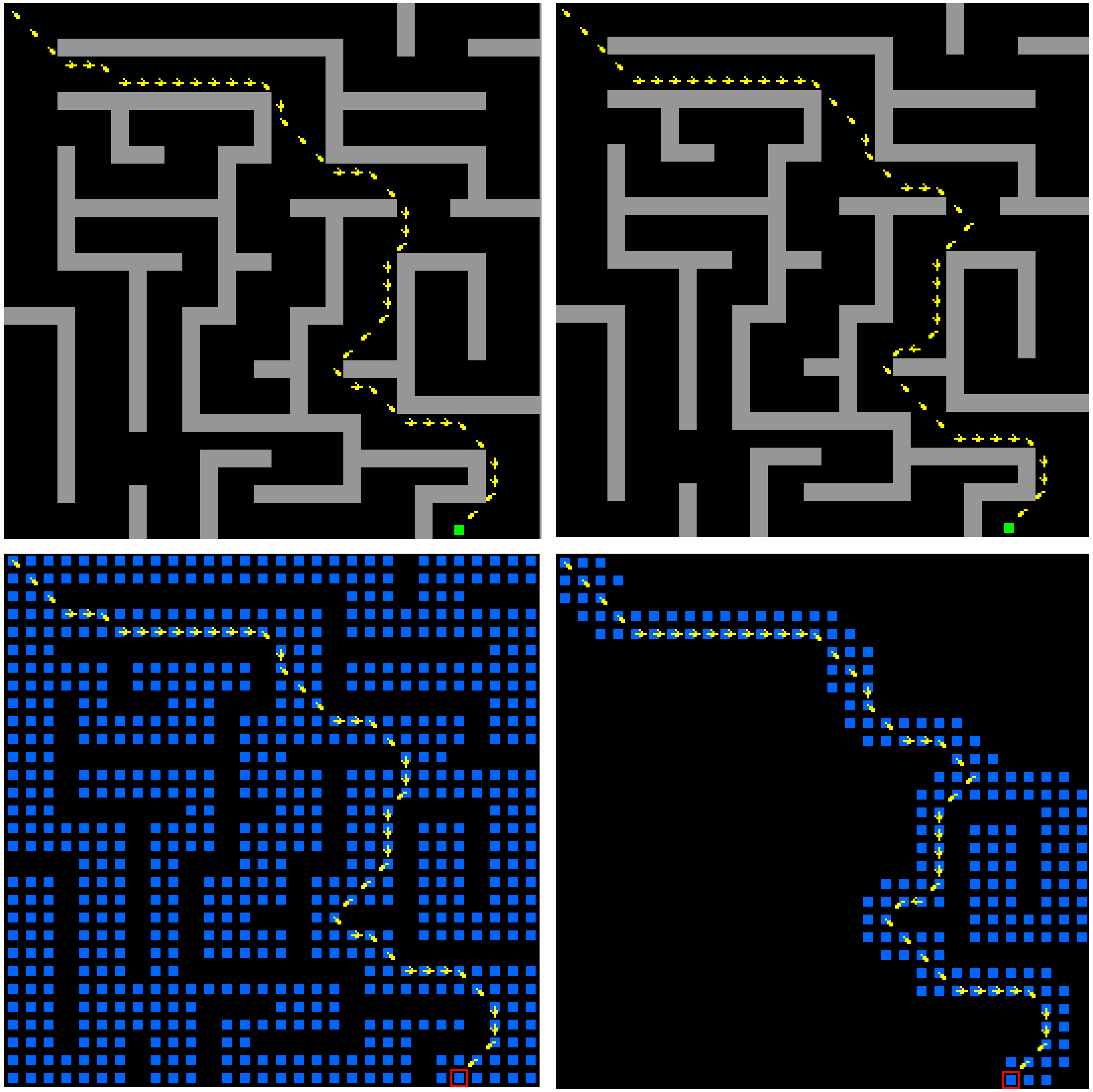
Exemplo: qual a melhor acção a realizar?



PROCURA MELHOR-PRIMEIRO (*BEST-FIRST*)

Procura de custo uniforme

- Solução óptima
- Maior complexidade computacional



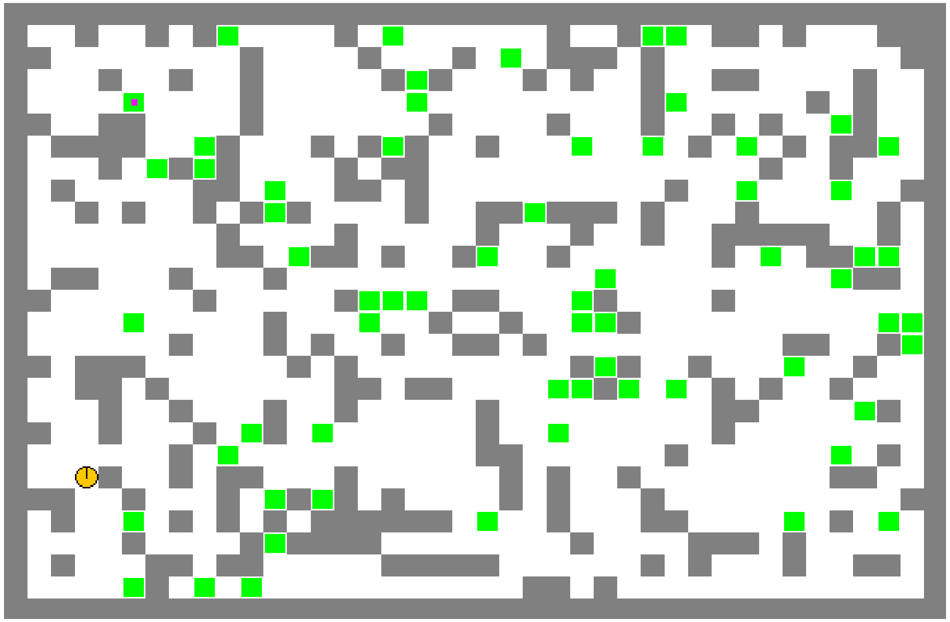
Procura sôfrega

- Solução sub-óptima
- Menor complexidade computacional

PROCURA A*

- **Heurística admissível**
  - $\mathbf{0 \leq \boldsymbol{h}(\boldsymbol{n}) \leq \boldsymbol{h}^*(\boldsymbol{n})}$
  - $\boldsymbol{h}^*(\boldsymbol{n})$
    - Custo mínimo do nó ***n*** até ao objectivo (percurso óptimo)
- Uma heurística **admissível** é ***optimista***
  - A estimativa de custo é **sempre inferior ou igual** ao custo efectivo **mínimo**
  - Para um nó objectivo $\boldsymbol{n}_{\mathbf{obj}}$
    - $\boldsymbol{h}(\boldsymbol{n}_{\mathbf{obj}}) = \mathbf{0}$

PROCURA A*



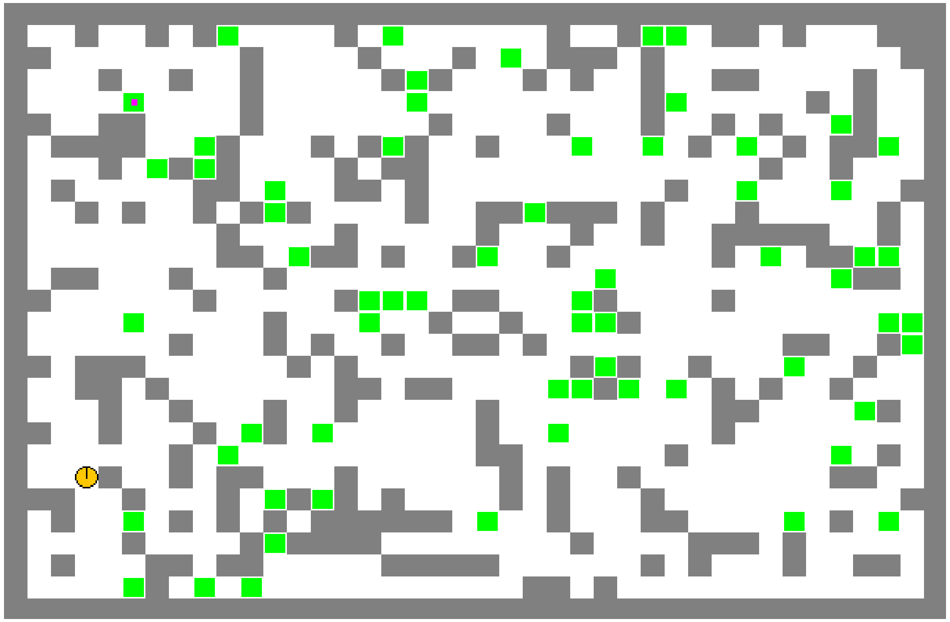
h_1 – Distância Euclidiana

$$h_1(n) = \sqrt{(x_n - x_{obj})^2 + (y_n - y_{obj})^2}$$

Admissível?

SIM

PROCURA A*



h_2 – Distância de Manhattan

$$h_2(n) = \quad |x_n - x_{obj}| + |y_n - y_{obj}|$$

Admissível?

- SIM : Se não forem possíveis movimentos diagonais
- NÃO : Caso contrário

PROCURA A*

- **Como definir uma heurística admissível**
  - No caso geral, uma heurística admissível é obtida através da remoção de restrições associadas ao problema
  - **Exemplo: Navegação autónoma**
    - **h_1 - Distância de Manhattan**
      - Corresponde a retirar a restrição:
        - » Não movimentação através de obstáculos
    - **h_2 - Distância de Euclidiana**
      - Corresponde a retirar as restrições:
        - » Não movimentação através de obstáculos
        - » Não movimentação em diagonal

PROCURA A*

- C^* - Custo da solução óptima
- n - Nó na fronteira de exploração

  $f(n) = g(n) + h(n) \leq C^*$ (se $h(n)$ admissível)

- m - Nó sub-óptimo na fronteira de exploração

  $f(m) = g(m) + h(m)$

- Se m for um nó objectivo

  $h(m) = 0$

  $f(m) = g(m) > C^*$

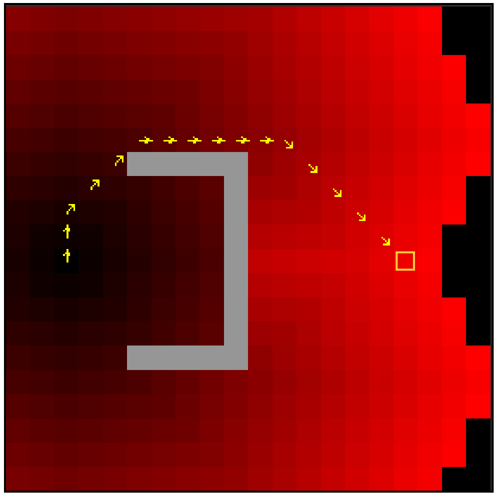
- Então

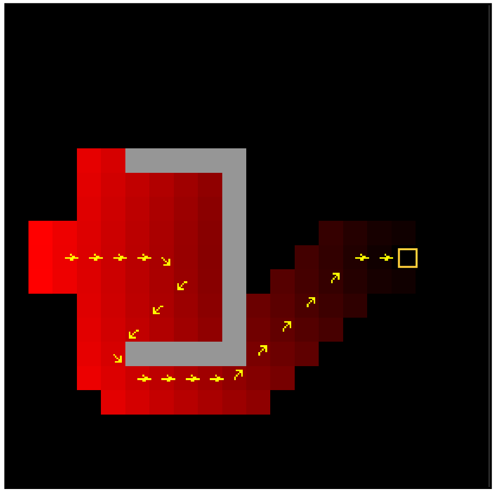
  $f(n) \leq C^* < f(m)$

  m não será expandido e a **solução encontrada será óptima**

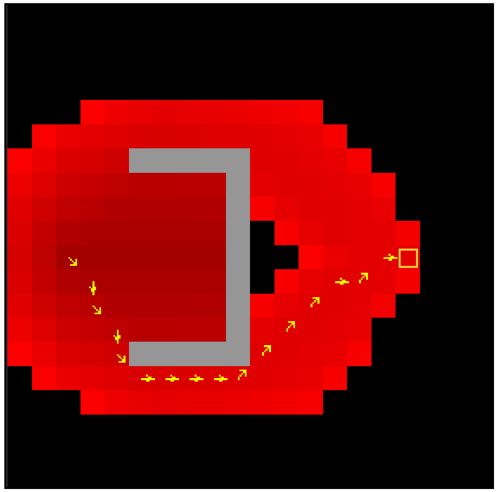
PROCURA EM ESPAÇOS DE ESTADOS
COMPARAÇÃO DE MÉTODOS DE PROCURA

Procura de
custo uniforme
(solução óptima)





Procura sôfrega
(solução sub-óptima)



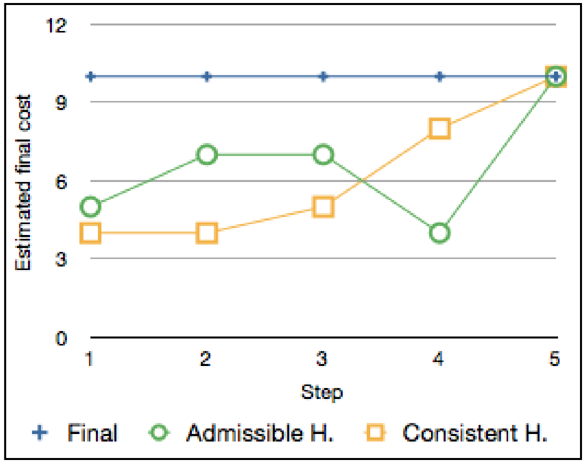
Procura A*
(solução óptima)

PROCURA A*

- **O método de procura A* é**
  - Completo
  - Óptimo
- **Se os nós já visitados forem mantidos**
  - Porque a heurística pode não ser *consistente*
    - O seu valor pode variar ao longo do processo de procura de forma não-monótona, ou seja, **não preserva uma relação de ordem** (crescente ou decrescente)
    - Logo não há garantia de que um nó mais recente seja melhor que nós anteriores, ou seja, que esse nó esteja necessariamente no caminho óptimo, pelo que **é necessário manter os nós anteriormente explorados para comparação**

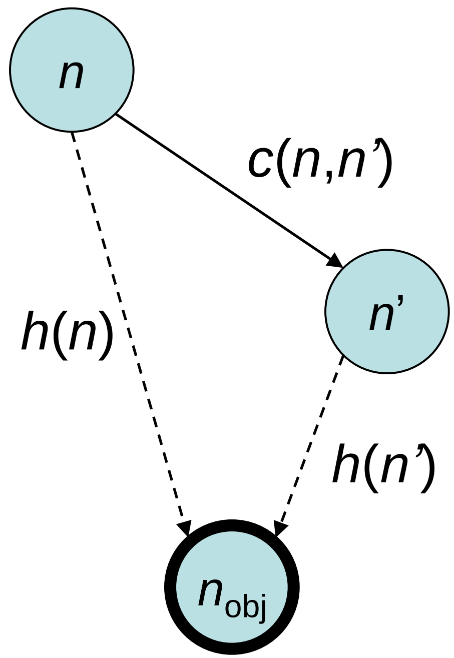
Exemplo de estimativa de custo para atingir o objectivo num processo de procura em espaço de estados, considerando uma **heurística admissível não consistente** e uma **heurística consistente**

Final: Custo final real



PROCURA A*

- **Heurística consistente (ou monótona)**
  - Para cada nó n, seu sucessor n' e custo de transição $c(n,n')$
    - $h(n) \leq c(n,n') + h(n')$
  - Para um nó objectivo
    - $h(n_{obj}) = 0$
- **Uma heurística consistente é também admissível**
- **Uma heurística admissível pode não ser consistente**



PROCURA A*

- Se $h(n)$ for consistente os valores de $f(n)$ nunca diminuem ao longo de um caminho

- Consideremos n' um sucessor de n

  $$g(n) + c(n,n') + h(n') \geq g(n) + h(n)$$

  $$\Downarrow$$

- Qualquer **nó selecionado para expansão tem de estar num percurso óptimo**, pois qualquer outro caminho terá um custo no mínimo igual

PROCURA A*

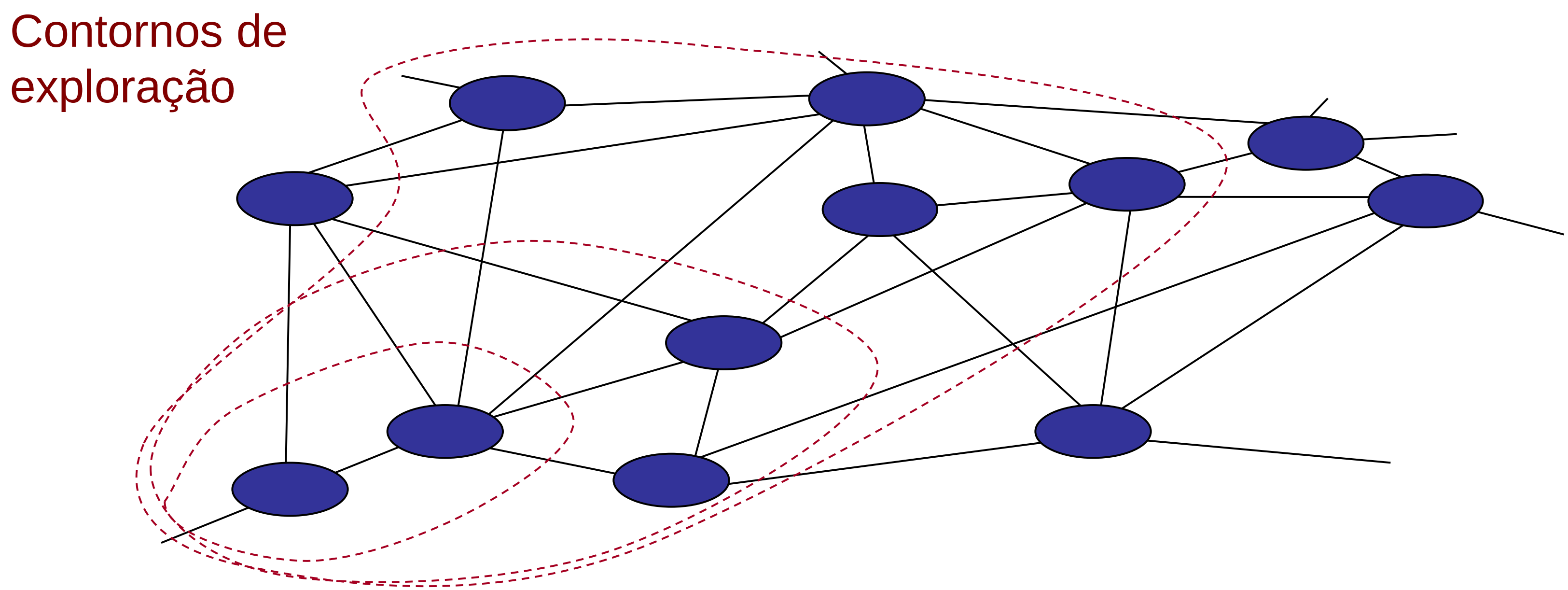
- **Com uma heurística consistente o método de procura A* é**
  - Completo
  - Óptimo
- **Mesmo se os nós já visitados forem eliminados**
  - Redução da complexidade da procura

PROCURA A* COM HEURÍSTICA CONSISTENTE

Ao gerar novo nó sucessor *noSuc*:

- $noSuc \notin Abertos \wedge noSuc \notin Fechados$
  - Inserir *noSuc* em *Abertos*
- $noSuc \in Abertos$
  - Se *noSuc* foi atingido através de um caminho mais curto
    - Remover nó anterior de Abertos
    - inserir noSuc em Abertos
- $noSuc \in Fechados$
  - Eliminar *noSuc*

PROCURA A*



- **Para uma heurística consistente**
  - Sempre que é expandido um nó o percurso desse nó é óptimo
  - São expandidos todos os nós com $f(n) < C^*$
  - São eventualmente expandidos nós com $f(n) = C^*$ antes do nó objectivo

PROCURA A*

- Método de procura de **eficiência óptima** para qualquer função heurística
  - Nenhum outro algoritmo expandirá menos nós, mantendo as características de ser **completo** e **óptimo**, excepto nas situações de escolha entre nós com $f(n) = C^*$

- **No entanto, não resolve o problema da complexidade combinatória**
  - O número de nós expandidos dentro do contorno do nó objectivo contínua a ser uma **função exponencial** da **dimensão do percurso** até ao objectivo
  - Função heurística possibilita a exploração selectiva do espaço de estados
    - Reduz o número de nós explorados
    - Pode não ser suficiente para a resolução prática de problemas de elevada complexidade

REFERÊNCIAS

[Russel & Norvig, 2003]

S. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach", 2nd Edition, Prentice Hall, 2003

[Pearl, 1984]

J. Pearl, "Heuristics: Intelligent Search Strategies for Computer Problem Solving", Addison-Wesley, 1984