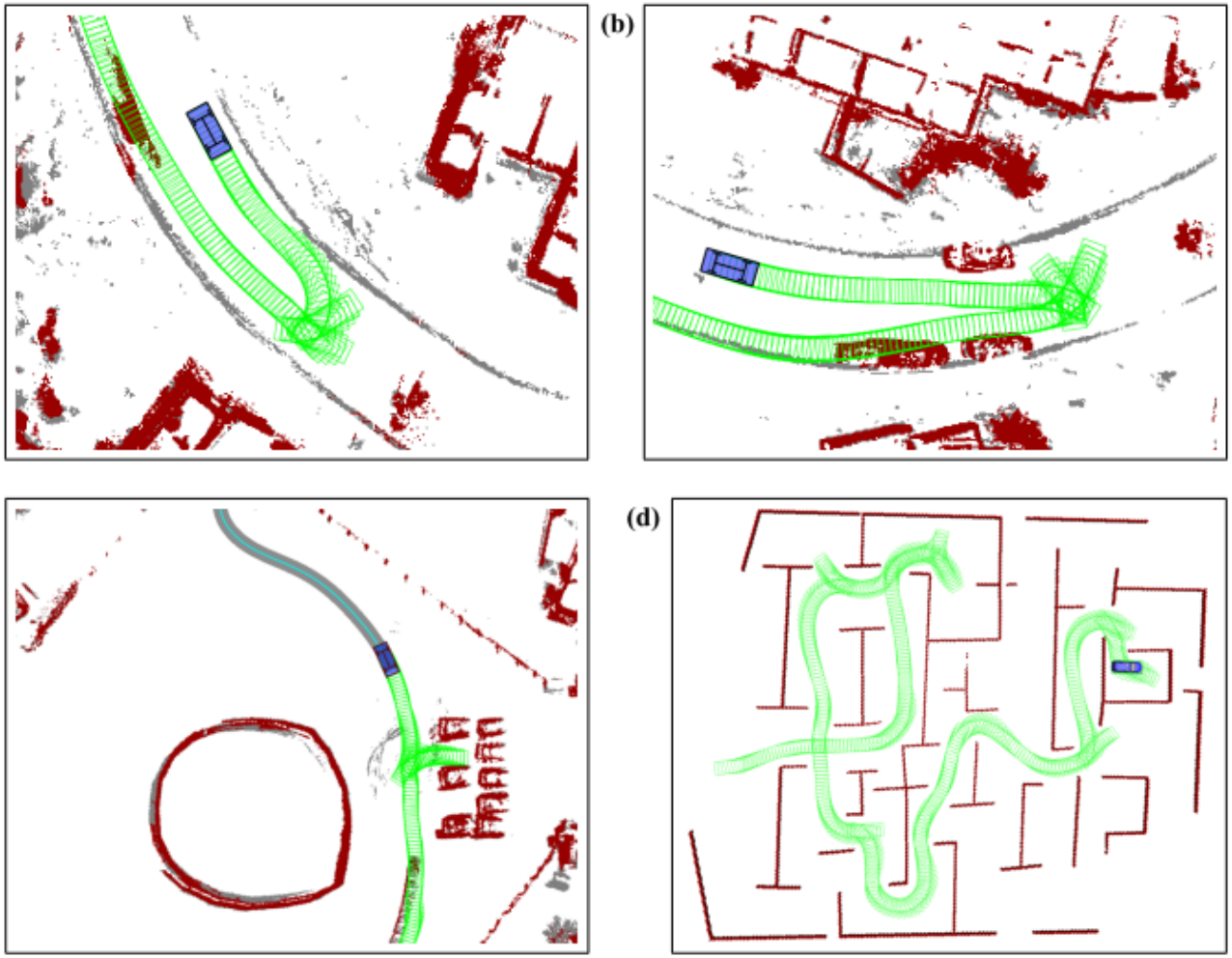
RACIOCÍNIO AUTOMÁTICO PARA PLANEAMENTO

Luís Morgado

ISEL-ADEETC

PROBLEMAS DE PLANEAMENTO

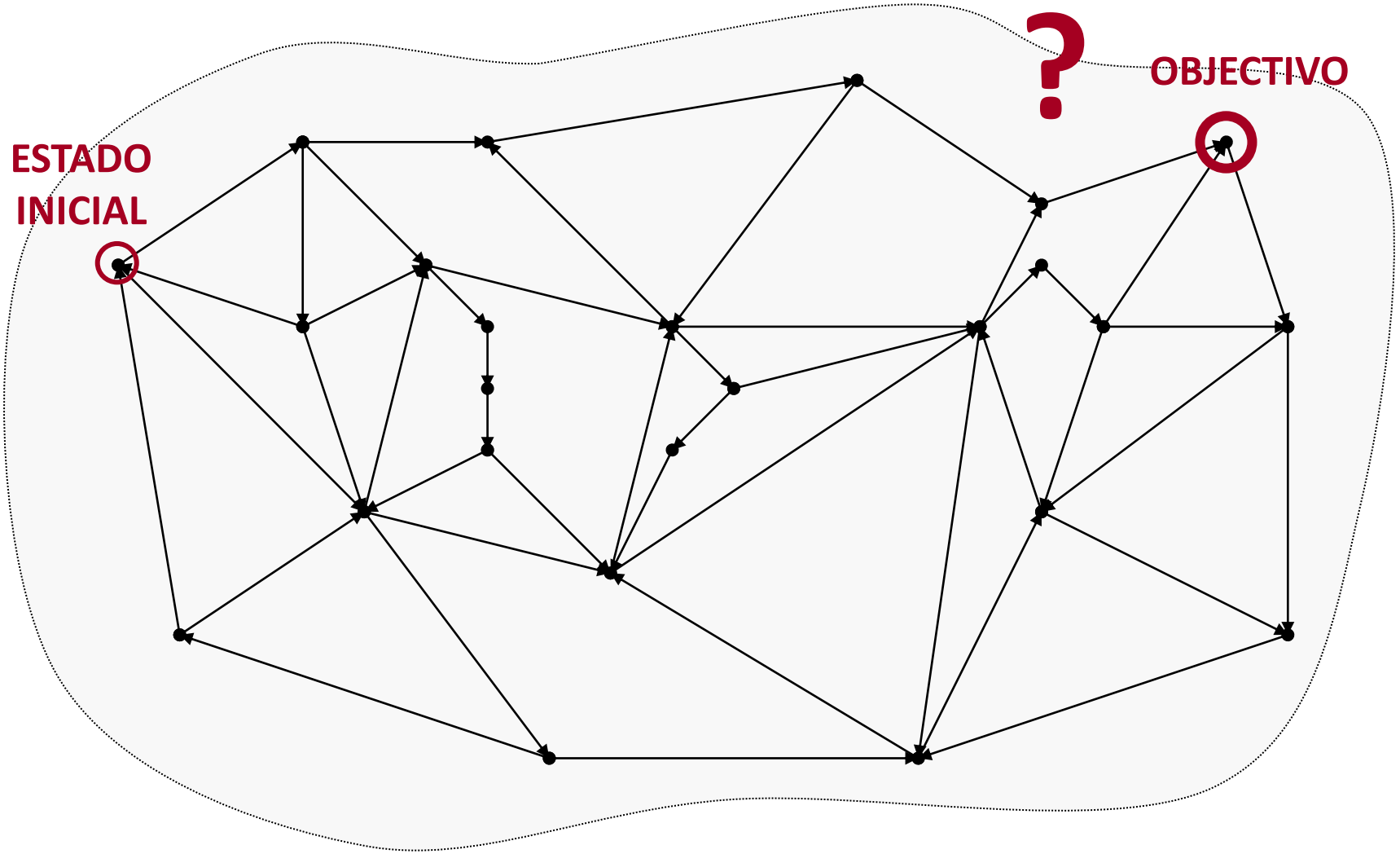
Exemplo: planeamento de trajectos em veículos autónomo



[Montemerlo, 2008]

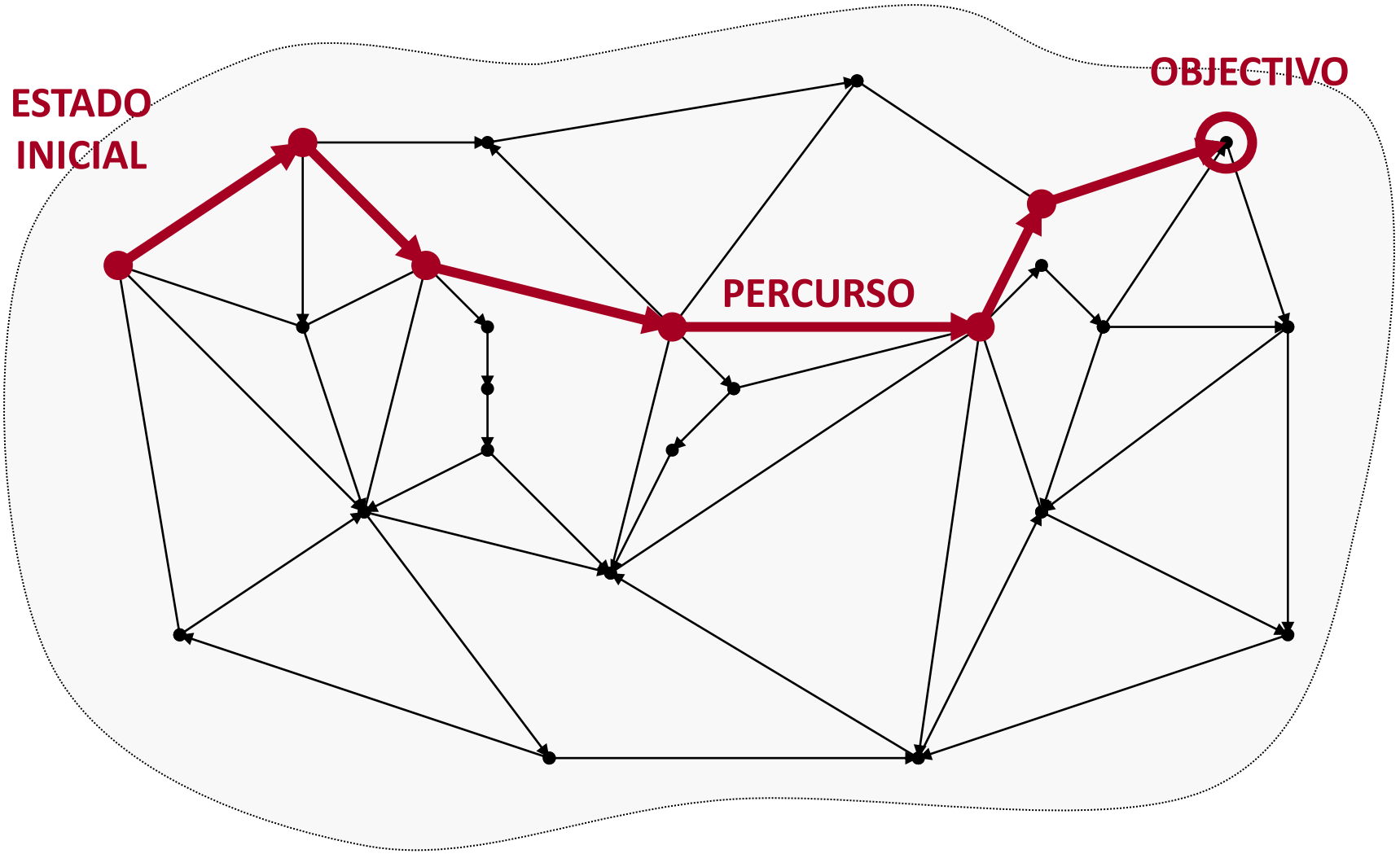
RACIOCÍNO ATRAVÉS DE PROCURA

PROBLEMAS DE PLANEAMENTO



RACIOCÍNO ATRAVÉS DE PROCURA

PROBLEMAS DE PLANEAMENTO

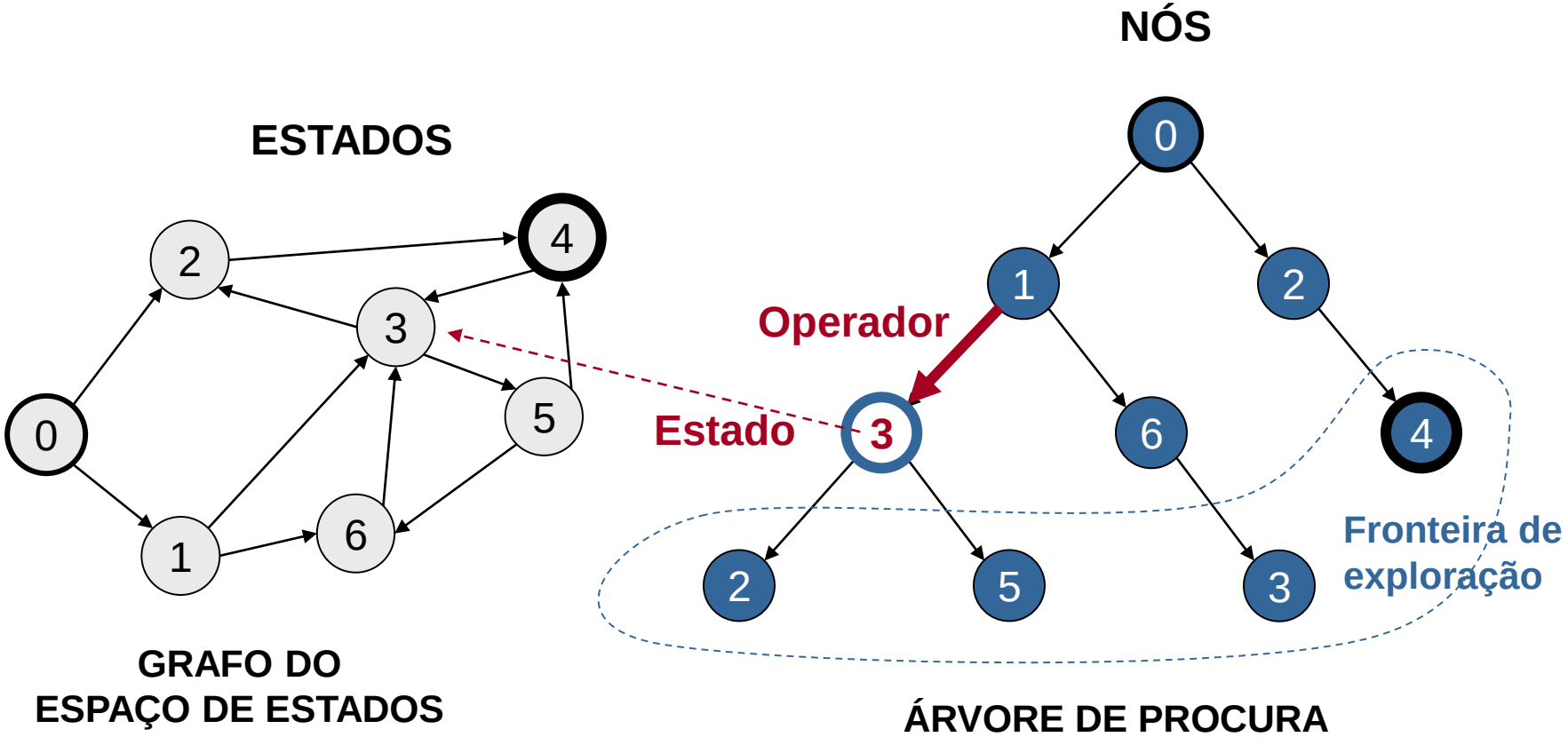


MÉTODOS DE PROCURA EXAUSTIVA

- **Procura não informada**
  - Estratégias de exploração do espaço de estados (controlo da procura) **não tiram partido de conhecimento do domínio do problema para guiar a procura**
  - **Procura exaustiva**
  - **Mantêm percursos explorados**
    - Árvore de procura
  - **Controlo da procura**
    - Fronteira de exploração
    - Relação de ordem de exploração de estados

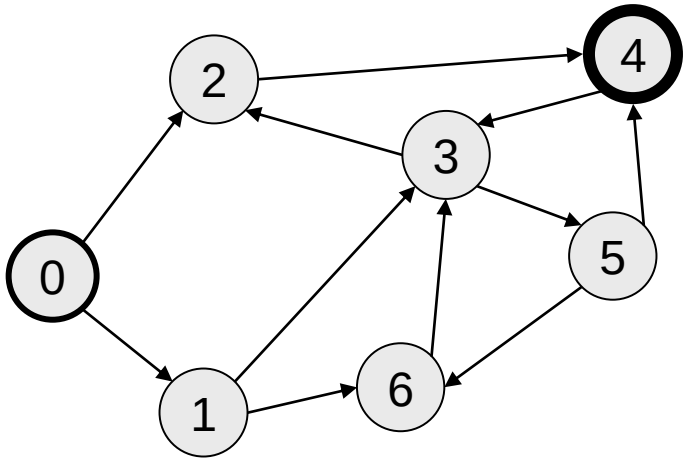
ÁRVORE DE PROCURA

- Resulta da exploração sucessiva do espaço de estados
- Etapa de procura: **Nó** (estado, operador)
- Raiz: **Nó** (*Estado inicial*)

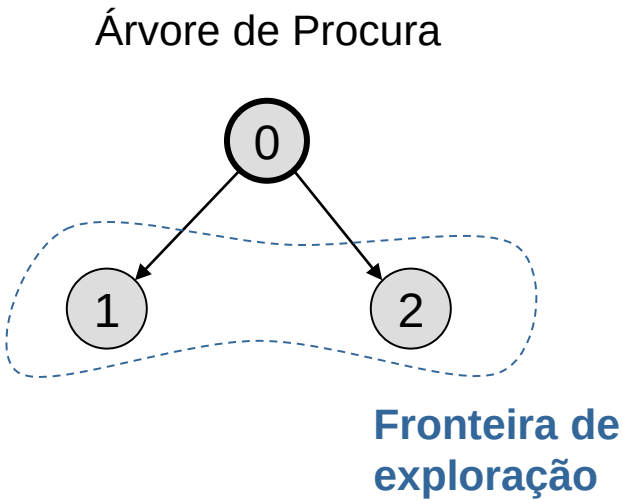


FRONTEIRA DE EXPLORAÇÃO

- **Nós**
  - Gerados (não expandidos)
  - Expandidos
- **Fronteira de exploração**
  - Mantém nós gerados e não expandidos
  - Define limite (contorno) de exploração



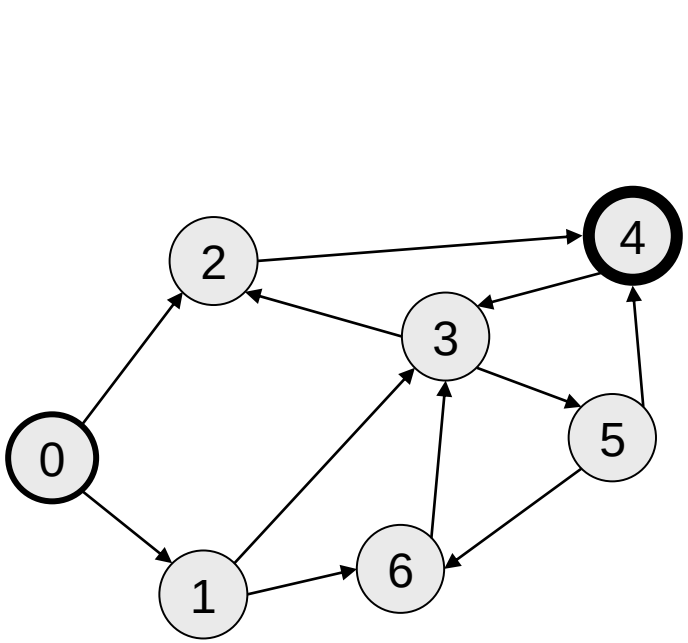
Grafo do
Espaço de Estados



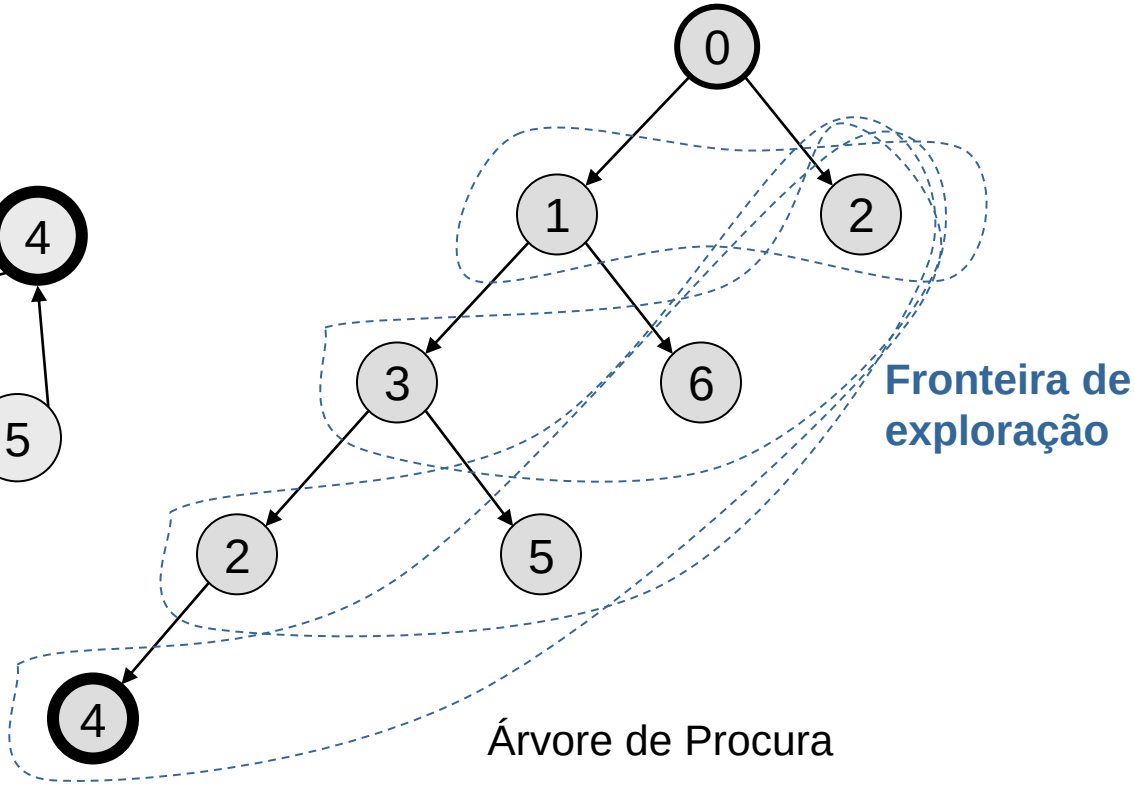
MÉTODOS DE PROCURA

PROCURA EM PROFUNDIDADE

- Estratégia de controlo
  - Explorar primeiro os nós mais **recentes**

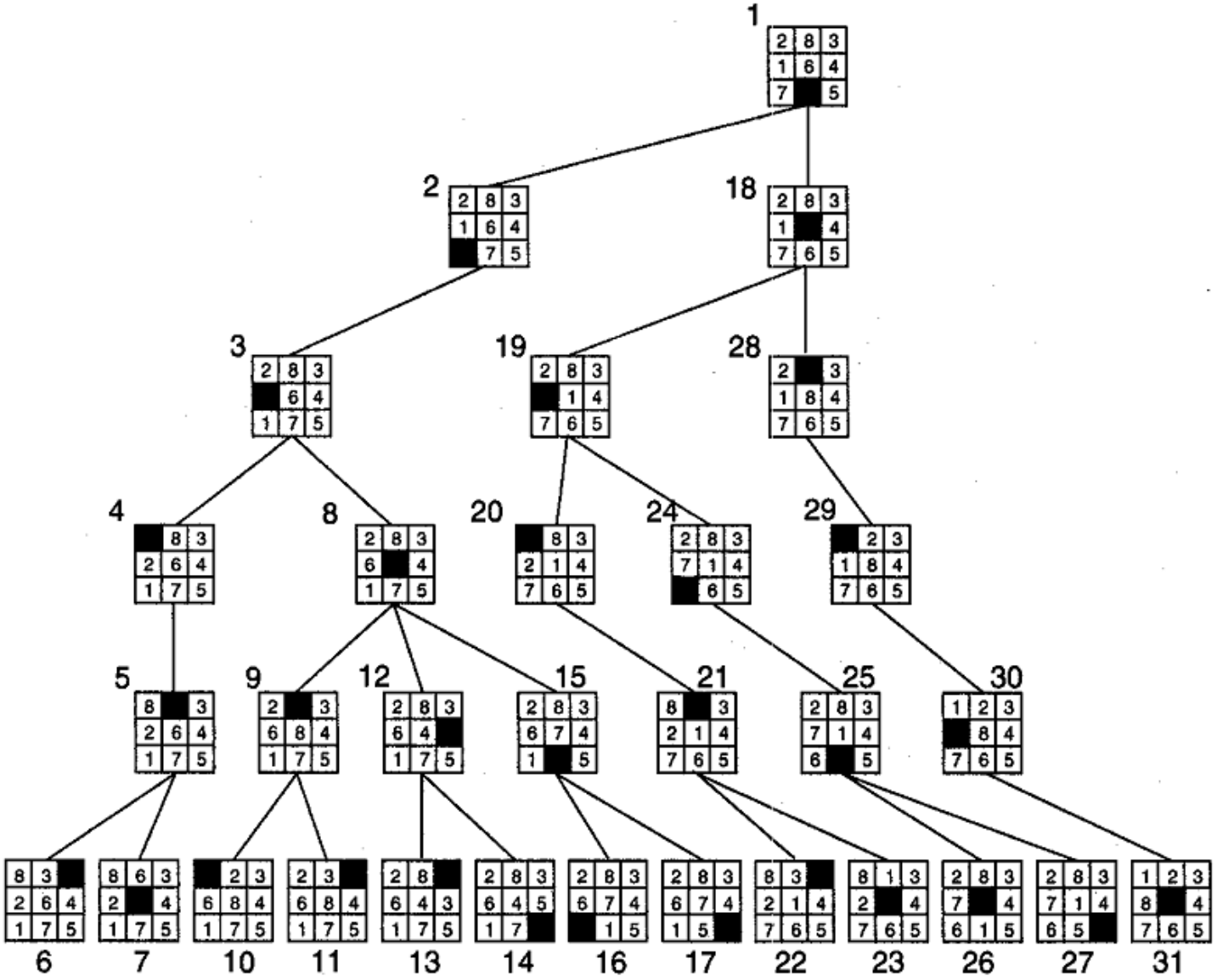


Grafo do
Espaço de Estados



Árvore de Procura

PROCURA EM PROFUNDIDADE

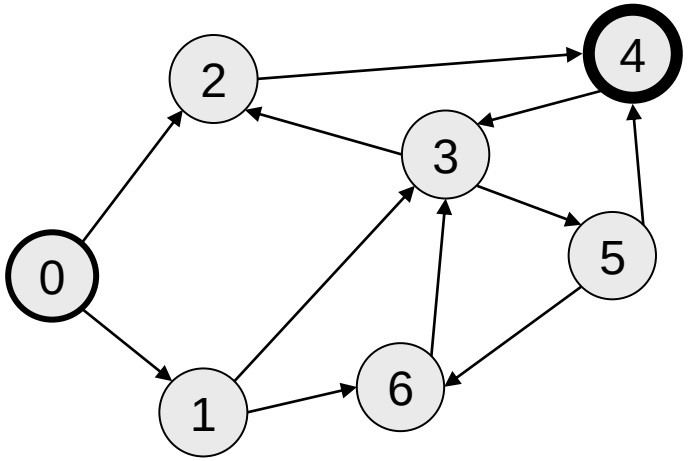


[Luger, 2009]

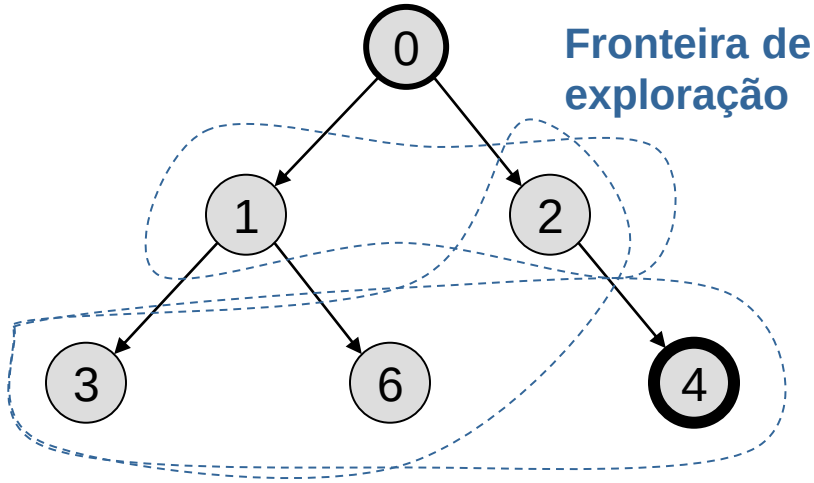
MÉTODOS DE PROCURA

PROCURA EM LARGURA

- Estratégia de controlo
  - Explorar primeiro os nós mais **antigos**

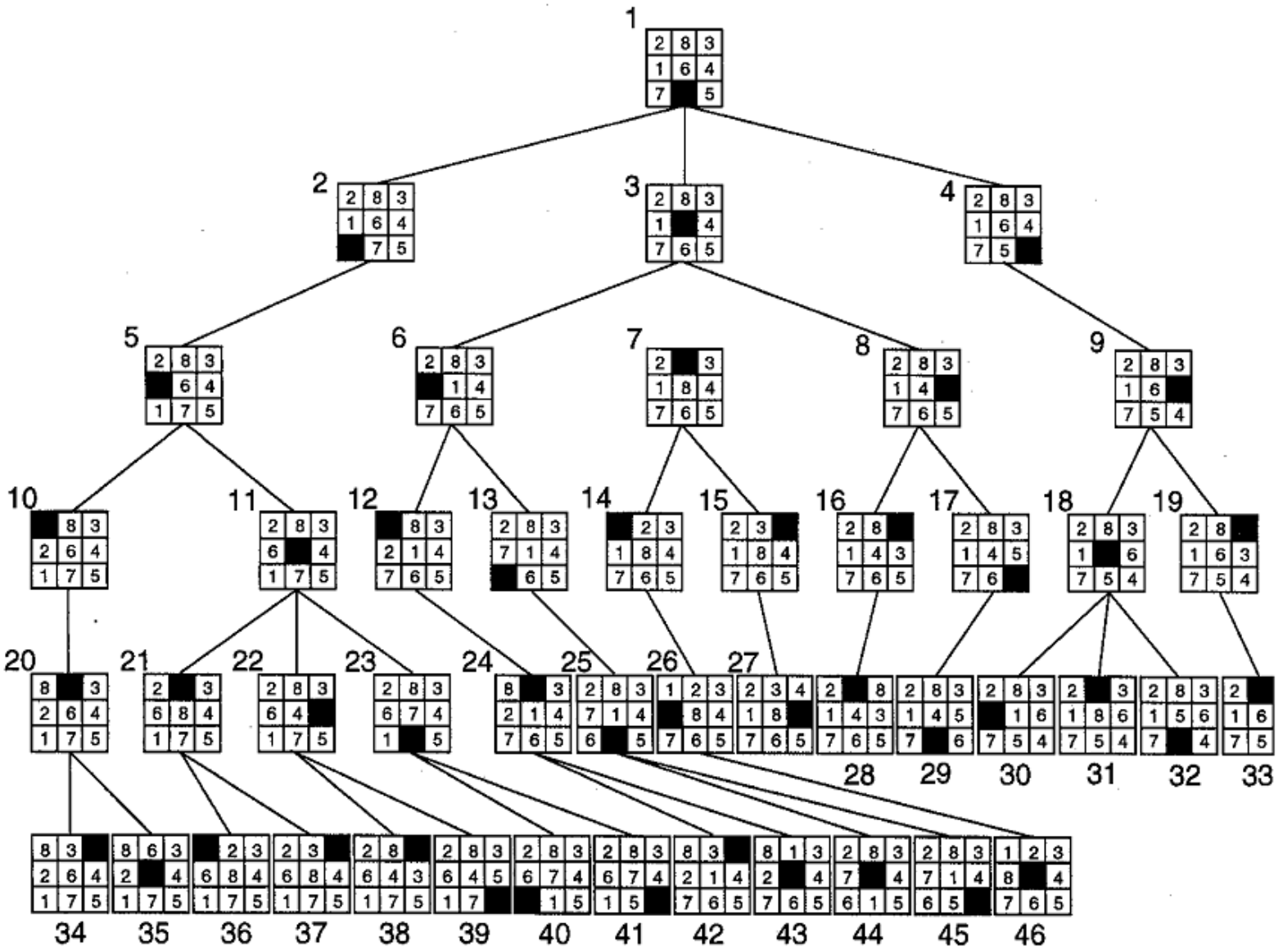


Grafo do
Espaço de Estados



Árvore de Procura

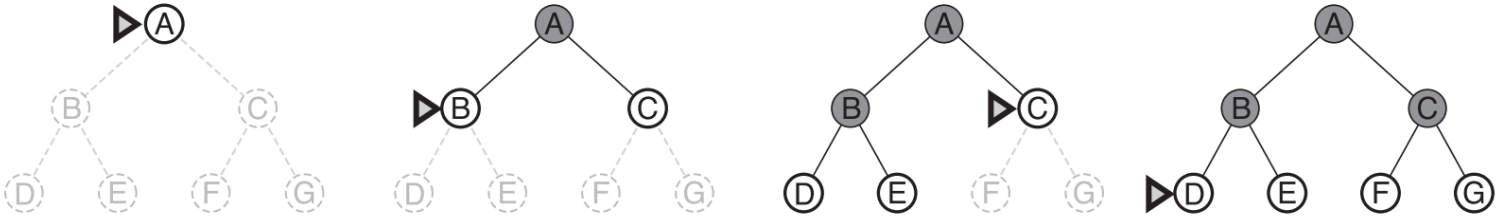
PROCURA EM LARGURA



[Luger, 2009]

COMPLEXIDADE COMPUTACIONAL

Complexidade combinatória



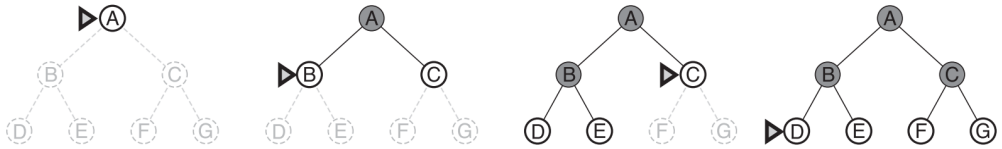
Depth	Nodes	Time	Memory
2	110	.11 milliseconds	107 kilobytes
4	11,110	11 milliseconds	10.6 megabytes
6	10^6	1.1 seconds	1 gigabyte
8	10^8	2 minutes	103 gigabytes
10	10^{10}	3 hours	10 terabytes
12	10^{12}	13 days	1 petabyte
14	10^{14}	3.5 years	99 petabytes
16	10^{16}	350 years	10 exabytes

Figure 3.13 Time and memory requirements for breadth-first search. The numbers shown assume branching factor $b = 10$; 1 million nodes/second; 1000 bytes/node.

[Russel & Norvig, 2010]

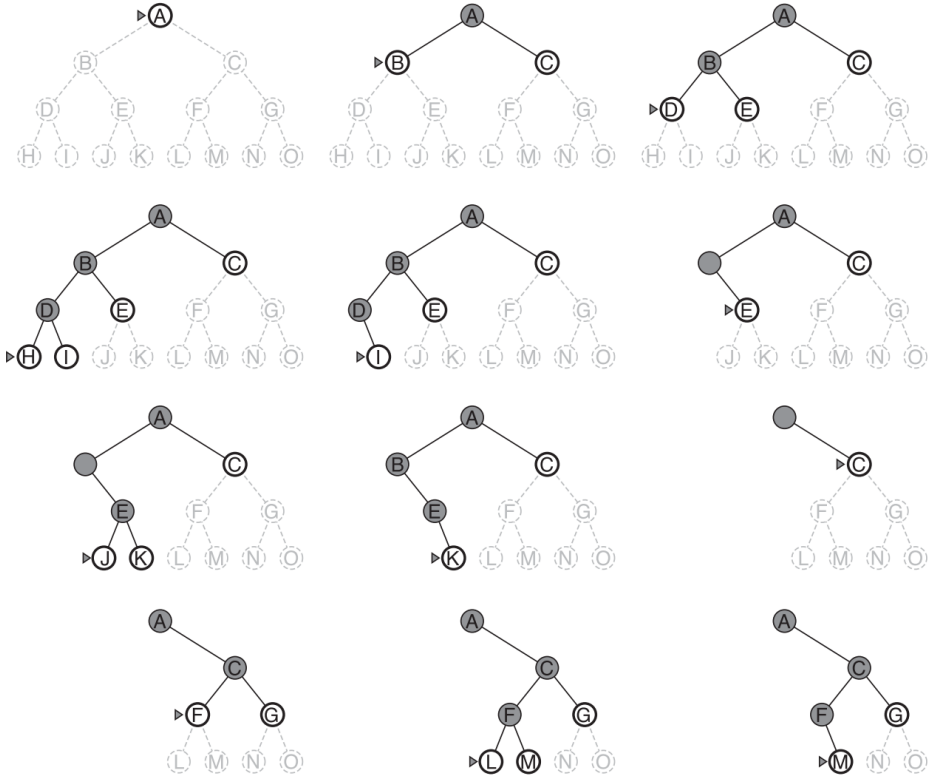
COMPLEXIDADE COMPUTACIONAL

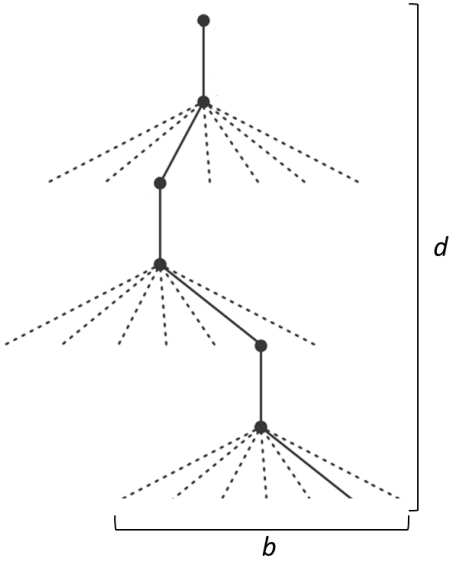
Procura em largura



Tempo	Espaço
$O(b^d)$	$O(b^d)$

Procura em profundidade





Tempo	Espaço
$O(b^d)$	$O(bd)$

COMPLEXIDADE COMPUTACIONAL

Método de Procura	Tempo	Espaço	Óptimo	Completo
Profundidade	$O(b^m)$	$O(bm)$	Não	Não
Largura	$O(b^d)$	$O(b^d)$	Sim	Sim

b – factor de ramificação
d – dimensão da solução
m – profundidade da árvore de procura
C^* – Custo da solução óptima
ε – Custo mínimo de uma transição de estado ($\varepsilon > 0$)

Notação *O*

$g(n)$ é de ordem $O(f(n))$ se existirem duas constantes positivas k e N tal que:

$\forall (n > N) : g(n) \leq kf(n)$

VARIANTES DA PROCURA EM PROFUNDIDADE

PROCURA EM PROFUNDIDADE LIMITADA
(*Depth-Limited Search*)

- Limitar procura a uma profundidade máxima

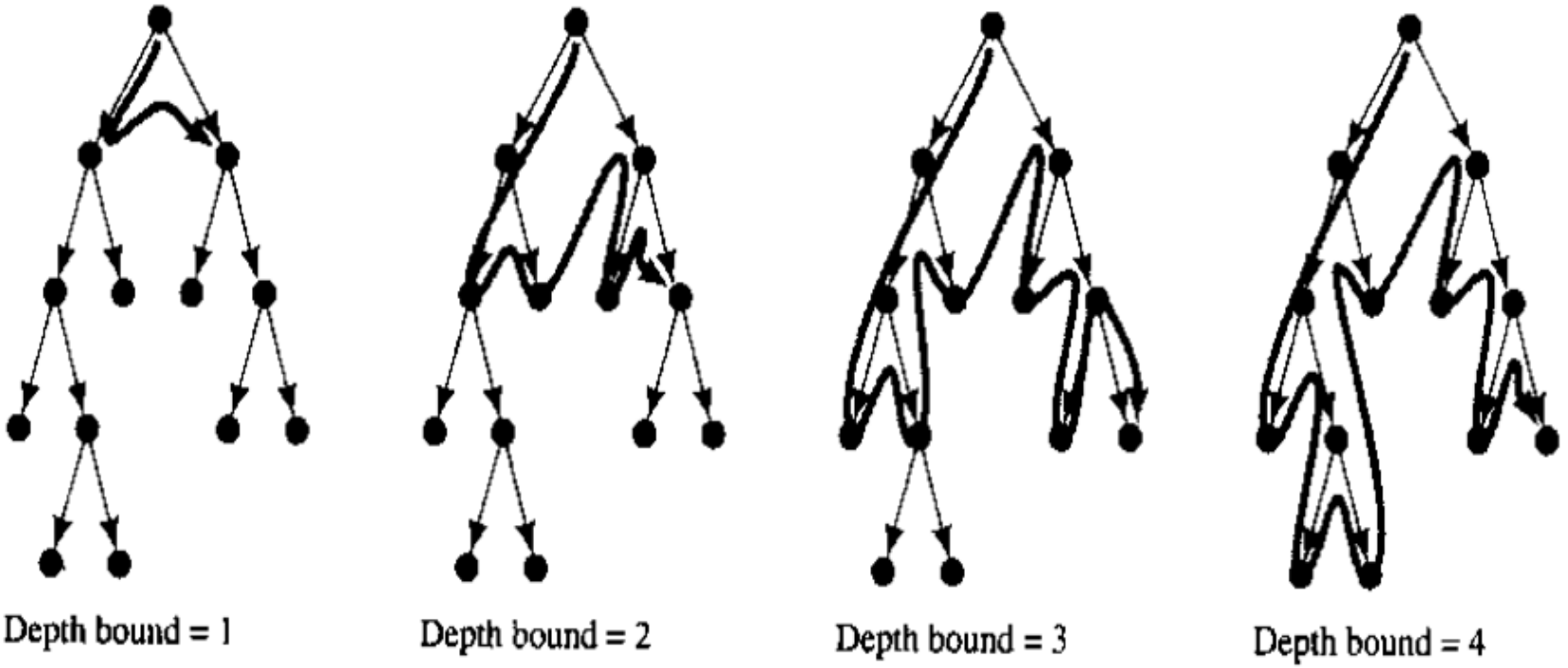
PROCURA EM PROFUNDIDADE ITERATIVA
(*Iterative Deepening Search*)

```
function ITERATIVE-DEEPENING-SEARCH(problem) returns a solution, or failure
    for depth = 0 to ∞ do
        result ← DEPTH-LIMITED-SEARCH(problem, depth)
        if result ≠ cutoff then return result
```

Figure 3.18 The iterative deepening search algorithm, which repeatedly applies depth-limited search with increasing limits. It terminates when a solution is found or if the depth-limited search returns *failure*, meaning that no solution exists.

[Russel & Norvig, 2010]

PROCURA EM PROFUNDIDADE ITERATIVA



[Nilsson, 1998]

Número de nós a expandir para encontrar uma solução de dimensão ***d***

$(d+1) + (d)b + (d-1)b^2 + \ldots + 2b^{d-1} + 1b^d$

Complexidade espacial: $O(bd)$

Complexidade temporal: $O(b^d)$

PROCURA EM GRAFOS COM CICLOS

ESTADOS REPETIDOS NA ÁRVORE DE PROCURA

- Acontece quando as **acções** correspondentes às transições de estado são **reversíveis**
- o grafo do espaço de estados apresenta **ciclos**

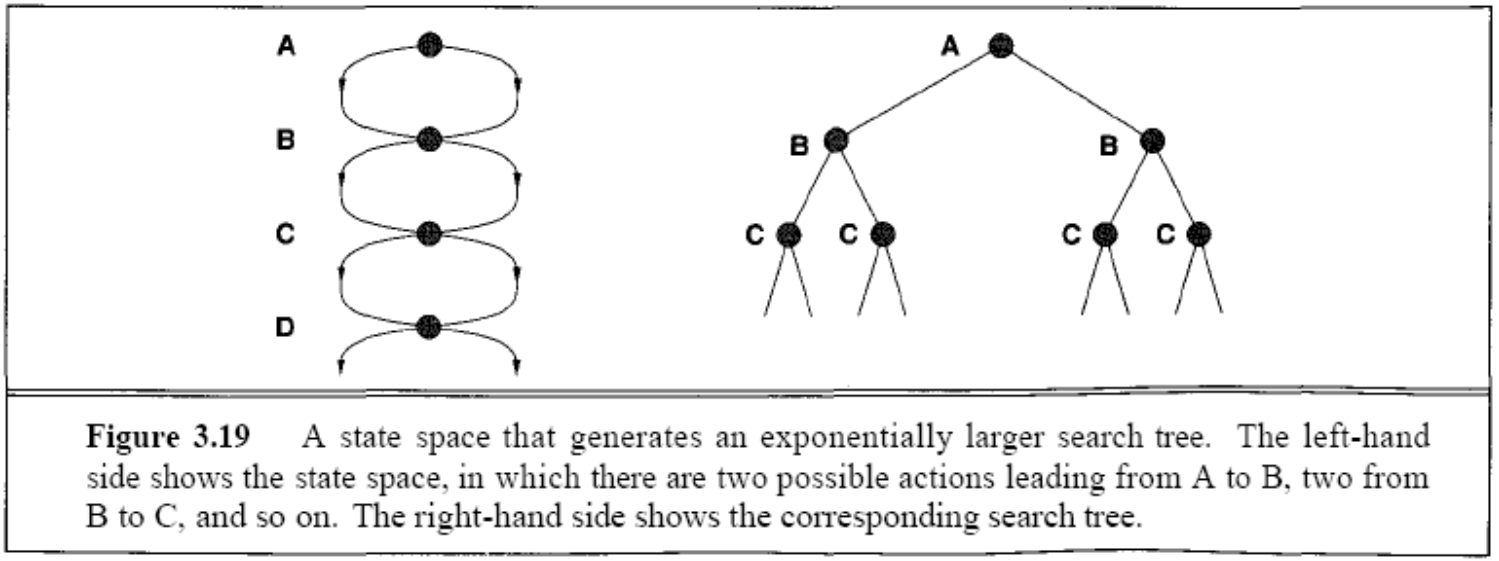


Figure 3.19 A state space that generates an exponentially larger search tree. The left-hand side shows the state space, in which there are two possible actions leading from A to B, two from B to C, and so on. The right-hand side shows the corresponding search tree.

[Russel & Norvig, 2003]

EXPANSÃO DE ESTADOS JÁ ANTERIORMENTE ANALISADOS

- **Desperdício** de recursos (**tempo**, **memória**)

PROCURA EM GRAFOS COM CICLOS

MEMÓRIA DE NÓS PROCESSADOS

- Nós gerados mas **não expandidos** (**fronteira** de exploração - ***fringe***)
  - **ABERTOS**
- Nós **expandidos**
  - **FECHADOS**

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  initialize the explored set to be empty
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set

[Russel & Norvig, 2010]

PROCURA GERAL EM GRAFOS

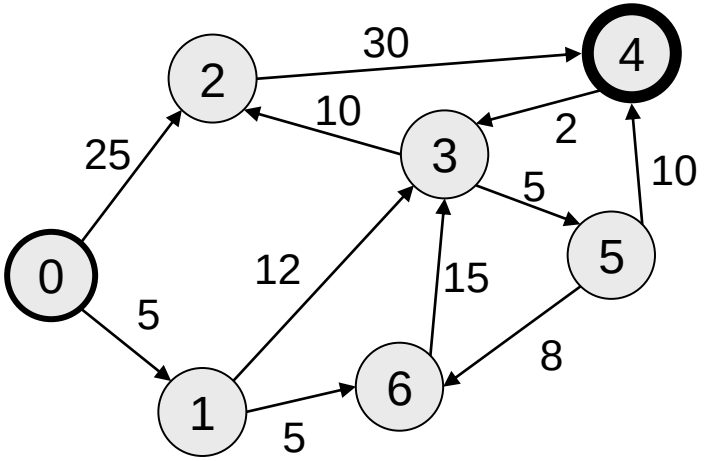
- Ao gerar novo nó sucessor *noSuc* é necessário considerar:
  - $noSuc \notin Abertos \wedge noSuc \notin Fechados$
    - Inserir *noSuc* em *Abertos*
  - $noSuc \in Abertos$
    - Se *noSuc* foi atingido através de um caminho mais curto (com menor custo)
      - Remover nó anterior de *Abertos*
      - inserir *noSuc* em *Abertos*
  - $noSuc \in Fechados$
    - Se *noSuc* foi atingido através de um caminho mais curto (com menor custo)
      - Remover nó anterior de *Fechados*
      - inserir *noSuc* em *Abertos*

PROCURA MELHOR-PRIMEIRO (*BEST-FIRST*)

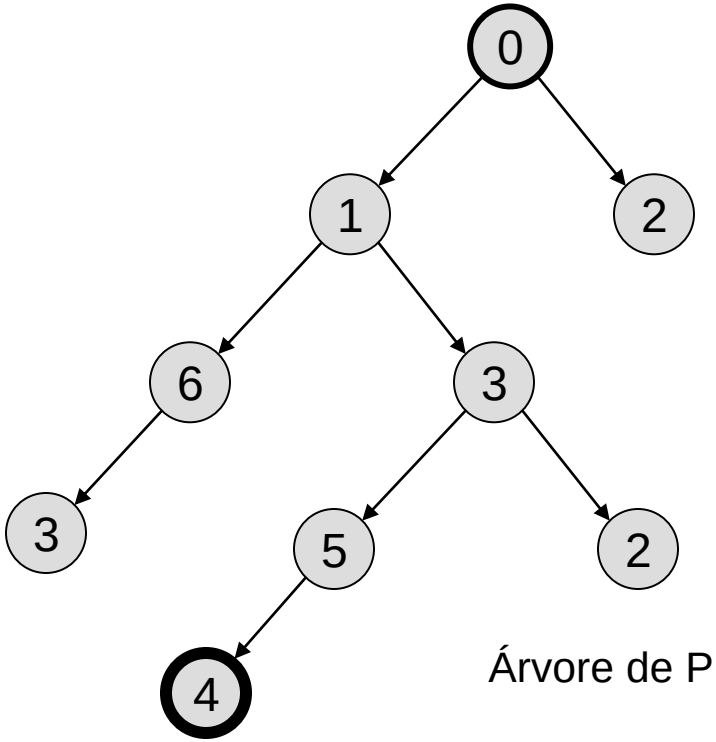
- Tira partido de uma avaliação do estado
  - **Custo vs. Utilidade**
- Utiliza uma função *f* para avaliação de cada nó *n* gerado
  - $f(n) \geq 0$
  - $f(n)$ pode ser uma estimativa do custo da solução através do nó *n* (procura informada)
    - **Quanto menor o valor de $f(n)$ mais promissor é o nó *n***
- A fronteira de exploração é ordenada por ordem crescente de $f(n)$

PROCURA DE CUSTO UNIFORME

- Estratégia de controlo
  - Explorar primeiro caminhos com menor custo
  - Custo de transição $\geq \varepsilon > 0$



Grafo do
Espaço de Estados



Árvore de Procura

MÉTODOS DE PROCURA NÃO INFORMADA

ESTRATÉGIAS DE CONTROLO DA PROCURA

Procura em profundidade

- Explorar primeiro nós com **maior profundidade**

Procura em largura

- Explorar primeiro nós com **menor profundidade**

Procura de custo uniforme

- Explorar primeiro nós de **menor custo**

RELAÇÕES DE ORDEM DE PROCURA

PROCURA EM PROFUNDIDADE $\prec_P$

$$(\forall n_1, n_2 \in G)\; n_1 \prec_P n_2 \Leftrightarrow \text{prof}(n_1) > \text{prof}(n_2)$$

PROCURA EM LARGURA $\prec_L$

$$(\forall n_1, n_2 \in G)\; n_1 \prec_L n_2 \Leftrightarrow \text{prof}(n_1) < \text{prof}(n_2)$$

PROCURA DE CUSTO UNIFORME $\prec_C$

$$(\forall n_1, n_2 \in G)\; n_1 \prec_C n_2 \Leftrightarrow \text{custo}(n_1) < \text{custo}(n_2)$$

COMPLEXIDADE COMPUTACIONAL

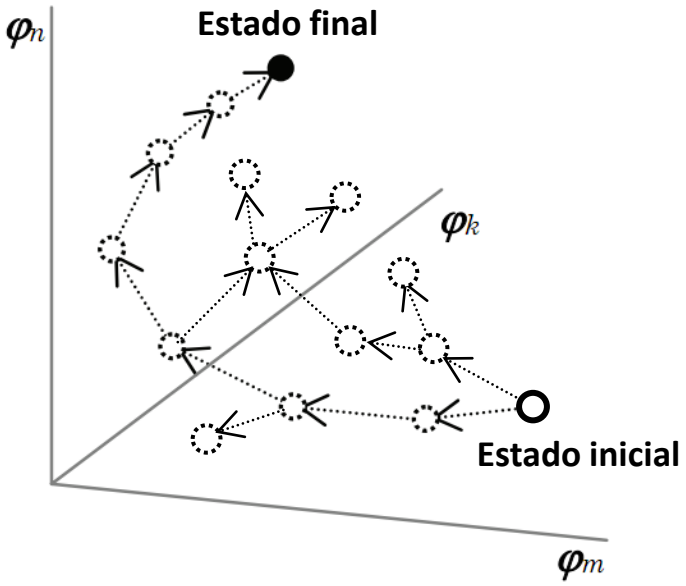
Método de Procura	Tempo	Espaço	Óptimo	Completo
Profundidade	$O(b^m)$	$O(bm)$	Não	Não
Largura	$O(b^d)$	$O(b^d)$	Sim	Sim
Custo Uniforme	$O(b^{[C^*/\varepsilon]})$	$O(b^{[C^*/\varepsilon]})$	Sim	Sim
Profundidade Limitada	$O(b^l)$	$O(bl)$	Não	Não
Profundidade Iterativa	$O(b^d)$	$O(bd)$	Sim	Sim

b – factor de ramificação
d – dimensão da solução
m – profundidade da árvore de procura
l – limite de profundidade

C^* – Custo da solução óptima
ε – Custo mínimo de uma transição de estado ($\varepsilon > 0$)

RACIOCÍNIO ATRAVÉS DE PROCURA

- **ESTADO**
  - Define situação
- **TRANSIÇÃO**
  - Define transformação de estado
  - **Operador**
    - Representação de **acção**
- **PROBLEMA**
  - Estado inicial
  - Estado final ou função objectivo
  - Operadores
  - Função de avaliação (custo)
- **SOLUÇÃO**
  - Percurso no espaço de estados



BIBLIOGRAFIA

[Russel & Norvig, 2003]

S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd Edition, Prentice Hall, 2009

[Russel & Norvig, 2010]

S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd Edition, Prentice Hall, 2010

[Nilsson, 1998]

N. Nilsson , *Artificial Intelligence: A New Synthesis*, Morgan Kaufmann 1998

[Luger, 2009]

G. Luger , *Artificial Intelligence: Structures and Strategies for Complex Problem Solving* , Addison-Wesley, 2009

[Jaeger & Hamprecht, 2010]

M. Jaeger, F. Hamprecht, *Automatic Process Control for Laser Welding*, Heidelberg Collaboratory for Image Processing (HCI) , 2000

[Pearl, 1984]

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

[Montemerlo, 2008]

M. Montemerlo *et al.*, *Junior: The Stanford Entry in the Urban Challenge*, Stanford Artificial Intelligence Lab, 2008