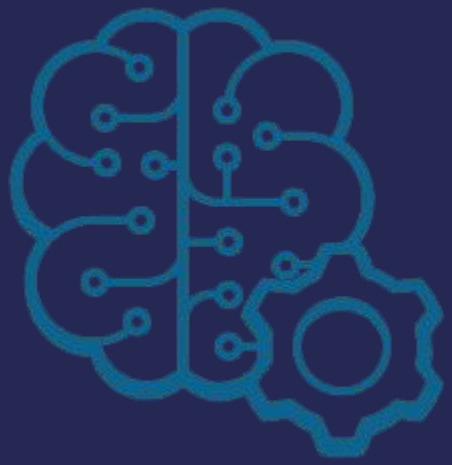
Prof. Filipe Dwan Pereira



INTELIGÊNCIA ARTIFICIAL

Aula 03 - Resolução de problemas - busca

Livro base:

Inteligência Artificial, Russell & Norvig, Editora Campus. 3ª Edição, 2013.

Aula baseada no capítulo 03

Agentes de resolução de problemas

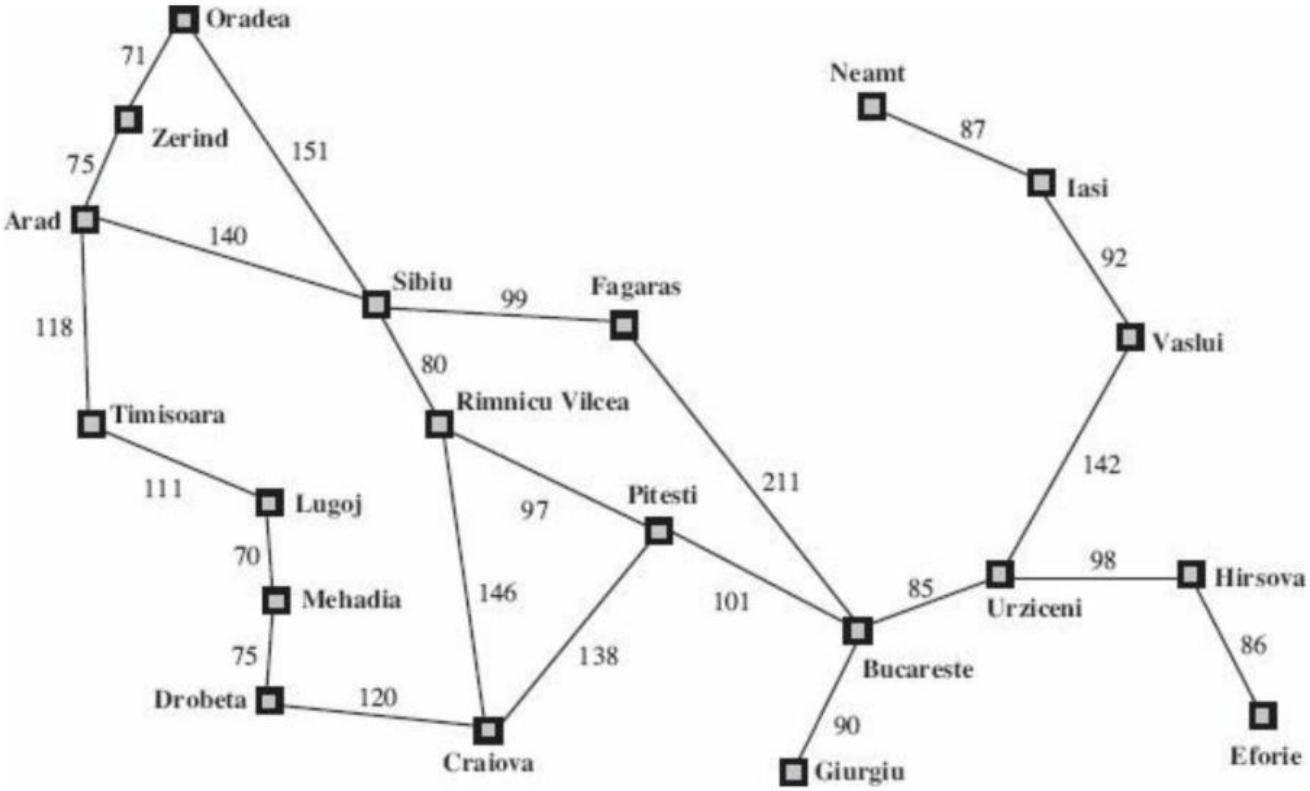
- Agentes de resolução de problemas utilizam representações **atômicas ->** os estados do mundo são considerados como um todo.
- Veremos diversos algoritmos de **busca sem informação —** algoritmos para os quais não se fornece nenhuma informação sobre o problema a não ser sua definição
  - busca sem informação -> solução não eficiente.

Agentes de resolução de problemas

- Imagine um agente que tem uma passagem não-reembolsável para partir de Bucareste na manhã seguinte.
  - Nesse caso, faz sentido para o agente adotar o objetivo de chegar a Bucareste.
  - objetivos ajudam a organizar o comportamento, limitando os objetivos que o agente está tentando alcançar.
  - A **formulação de objetivos**, baseada na situação atual e na medida de desempenho do agente, é o primeiro passo para a resolução de problemas.

Agentes de resolução de problemas

- A tarefa do agente é descobrir que sequência de ações o levará a um estado objetivo.
- Esse processo de procurar por tal sequência é chamado **busca**.



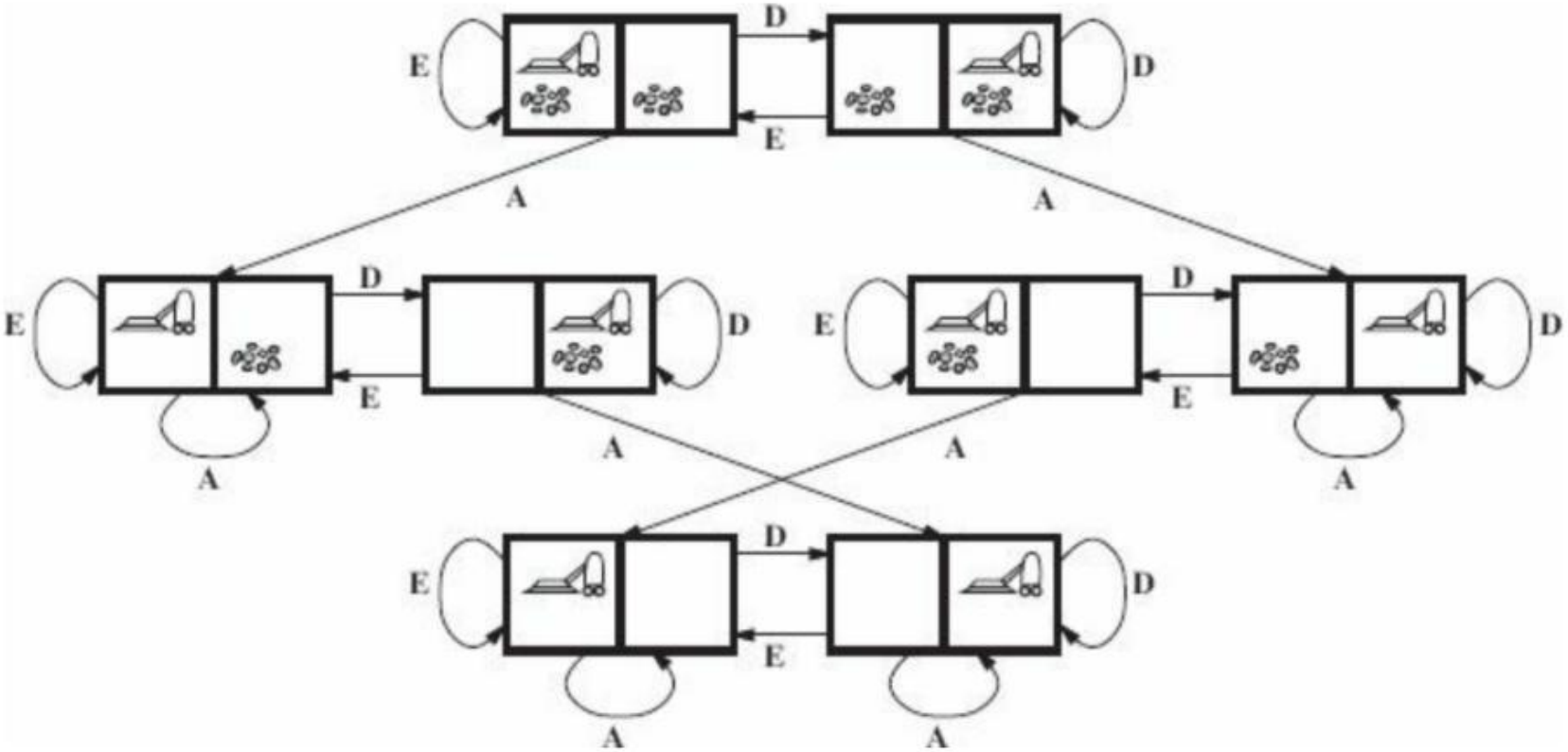
Agentes de resolução de problemas

- Um problema pode ser definido formalmente por quatro componentes:
- 1 - O estado inicial em que o agente começa. Por exemplo, o estado inicial do nosso agente na Romênia poderia ser descrito como Em(Arad).
- 2 - Uma descrição das ações possíveis que estão disponíveis para o agente. A partir do estado Em(Arad), as ações aplicáveis são: {Ir(Sibiu), Ir(Timisoara), Ir(Zerind)}.
- 3 - O teste de objetivo, que determina se um dado estado é um espaço objetivo - {Em(Bucareste)}.
- 4 - Uma função de custo de caminho que atribui um custo numérico a cada caminho.

Ex.: mundo do aspirador de pó

- Vamos examinar o mundo do aspirador de pó:
- **Estados**: O agente está em uma entre duas posições, cada uma das quais pode conter sujeira ou não.
- **Estado inicial**: Qualquer estado pode ser designado como o estado inicial.
- **Modelo de transição**: Gera os estados válidos que resultam da tentativa de executar as três ações (Esquerda, Direita e Aspirar).
- **Teste de objetivo**: Verificar se todos os quadrados estão limpos.
- **Custo de caminho**: Cada passo custa 1, e assim o custo do caminho é o número de passos do caminho.

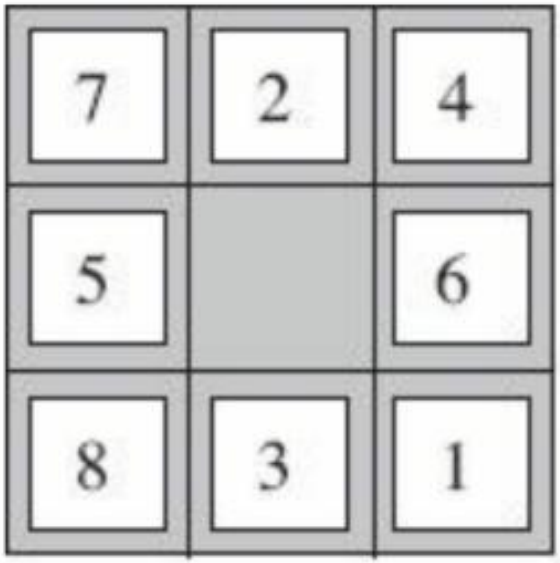
Ex.: mundo do aspirador de pó



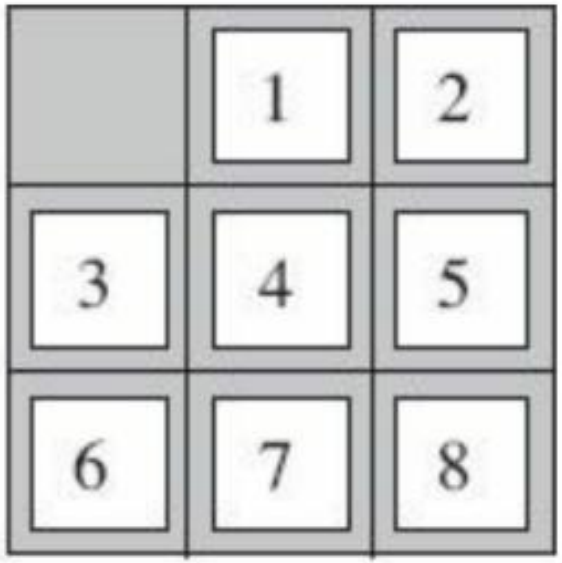
Ex.: quebra cabeça de oito peças

- Formulação padrão:
- **Estados**: Uma descrição de estado especifica a posição de cada uma das oito peças e do espaço vazio em um dos nove quadrados.
- **Estado inicial**: Qualquer estado pode ser designado como o estado inicial.
- **Modelo de transição**: Dado um estado e ação, ele devolve o estado resultante.
- **Teste de objetivo**: Verifica se o estado corresponde à configuração de estado objetivo.
- **Custo de caminho**: Cada passo custa 1, e assim o custo do caminho é o número de passos do caminho.

Ex.: quebra cabeça de oito peças



Estado inicial

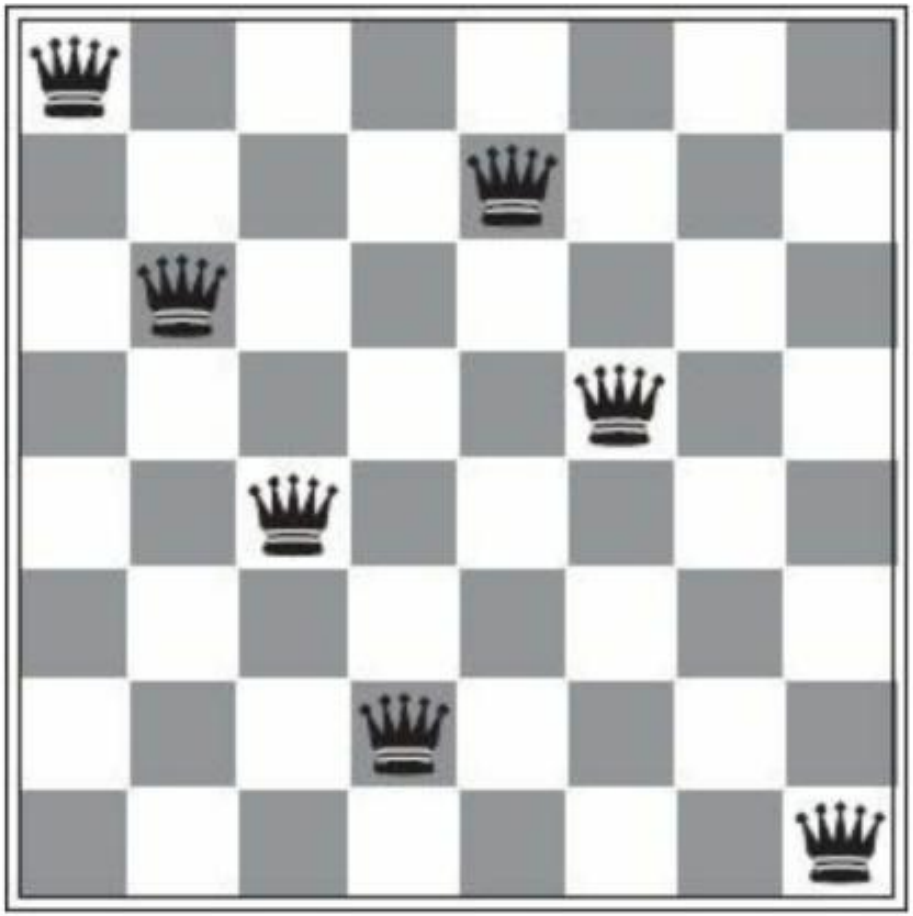


Estado objetivo

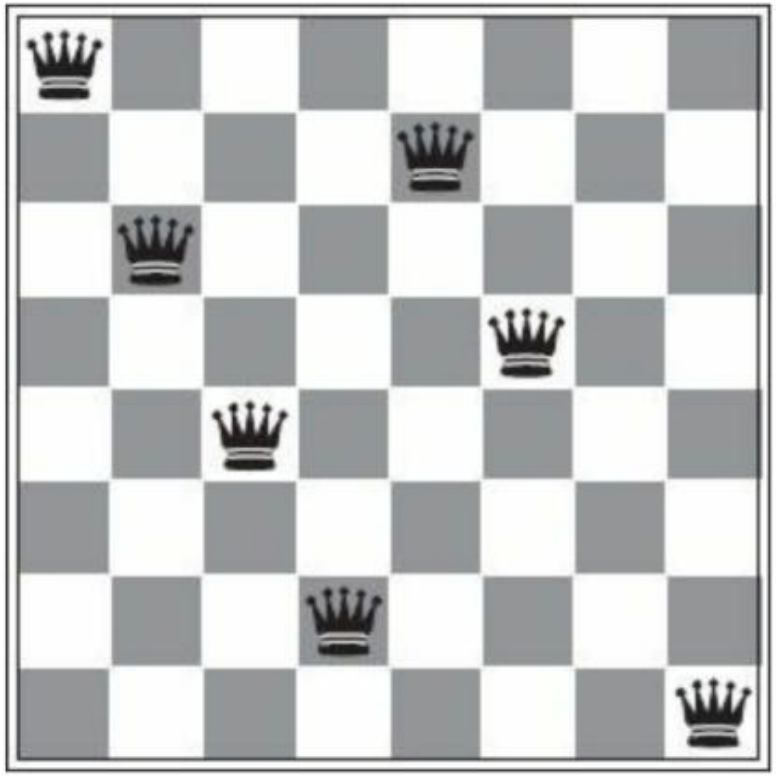
Ex.: problema das 8 rainhas

- Formulação padrão:
- **Estados**: Qualquer disposição de 0 a 8 rainhas no tabuleiro é um estado.
- **Estado inicial**: nenhuma rainha no tabuleiro.
- **Modelo de transição**: Colocar uma rainha em qualquer quadrado vazio.
- **Teste de objetivo**: 8 rainhas estão no tabuleiro e nenhuma é atacada.
- **Custo de caminho**: Cada passo custa 1, e assim o custo do caminho é o número de passos do caminho.

Ex.: problema das 8 rainhas



Ex.: problema das 8 rainhas

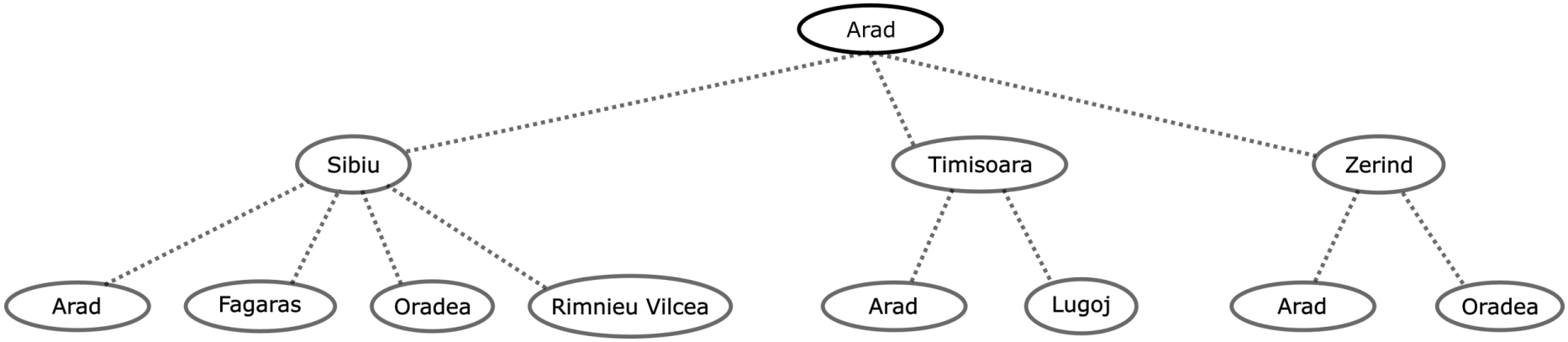


Solução do problema das N rainhas

Em busca de soluções

- Em geral, podemos ter um grafo de busca em lugar de uma árvore de busca, quando o mesmo estado pode ser alcançado a partir de vários caminhos.
- Mostrando algumas das expansões na árvore de busca para encontrar uma rota de Arad até Bucareste. A raiz da árvore é um nó de busca correspondente ao estado inicial, Em(Arad).

Árvore de busca parcial para localização de uma rota de Arad para Bucareste



Estratégia de busca

função BUSCA-EM-ÁRVORE(*problema*) **retorna** uma solução ou falha
inicializar a borda utilizando o estado inicial do *problema*
repita
se *borda vazia* **então retornar** falha
escolher um nó folha e o remover da borda
se o nó contém um estado objetivo **então retornar** a solução correspondente
expandir o nó escolhido, adicionando os nós resultantes à borda

função BUSCA-EM-GRAFO(*problema*) **retorna** uma solução ou falha
inicializar a borda utilizando o estado inicial do *problema*
inicializar o conjunto explorado tornando-o vazio
repita
se *borda vazia* **então retornar** falha
escolher um nó folha e o remover da borda
se o nó contiver um estado objetivo **então retornar** a solução correspondente
adicionar o nó ao conjunto explorado
expandir o nó escolhido, adicionando os nós resultantes à borda
apenas se não estiver na borda ou no conjunto explorado

Medição de desempenho de resolução de problemas

- Podemos avaliar o desempenho do algoritmo em quatro aspectos:
- **Completeza**: O algoritmo oferece a garantia de encontrar uma solução quando ela existir?
- **Otimização**: A estratégia encontra a solução ótima
- **Complexidade de tempo**: Quanto tempo ele leva para encontrar uma solução?
- **Complexidade de espaço**: Quanta memória é necessária para executar a busca?

Medição de desempenho de resolução de problemas

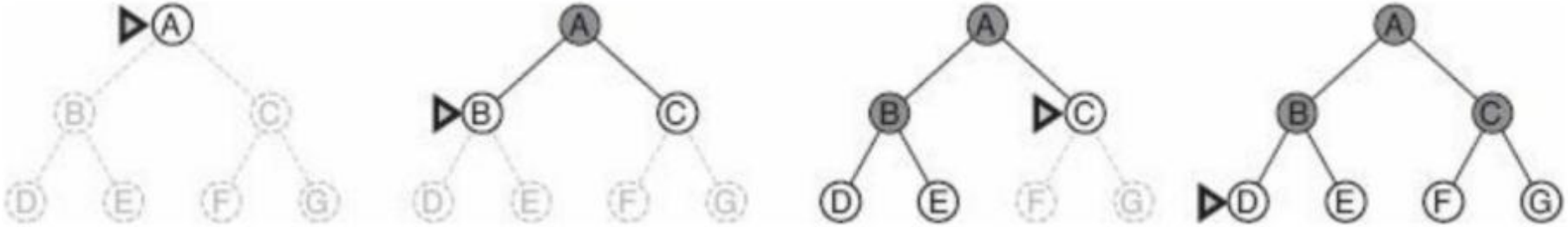
- Em inteligência artificial, a complexidade é expressa em termos de três quantidades:
  - **b**, o fator de ramificação ou número máximo de sucessores de qualquer nó;
  - **d**, a profundidade do nó objetivo menos profundo (ou seja, o número de passos ao longo do caminho da raiz até o estado objetivo mais próximo); e
  - **m**, o comprimento máximo de qualquer caminho no espaço de estados.

Infraestrutura para algoritmos de busca

- n.ESTADO: o estado no espaço de estado a que o nó corresponde;
- n.PAI: o nó na árvore de busca que gerou esse nó;
- n.AÇÃO: a ação que foi aplicada ao pai para gerar o nó;
- n.CUSTO-DO-CAMINHO: o custo, tradicionalmente denotado por g(n), do caminho do estado inicial até o nó, indicado pelos ponteiros para os pais.

Estratégias de busca sem informação

Busca em Largura

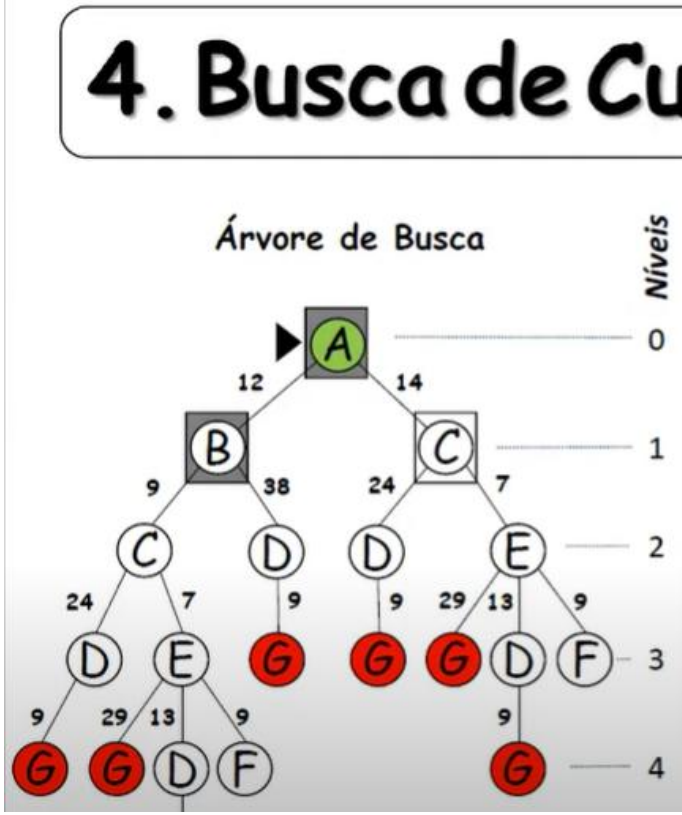


Estratégias de busca sem informação

```
função BUSCA-EM-LARGURA(problema) retorna uma solução ou falha
  nó ← um nó com ESTADO = problema.ESTADO-INICIAL, CUSTO-DE-CAMINHO = 0
  se problema.TESTE-DE-OBJETIVO(nó.ESTADO) senão retorne SOLUÇÃO(nó),
  borda ← uma fila FIFO com nó como elemento único
  explorado ← conjunto vazio
  repita
    se VAZIO?(borda), então retorne falha
    nó ← POP(borda) / * escolhe o nó mais raso na borda */
    adicione nó.ESTADO para explorado
    para cada ação em problema.AÇÕES(nó.ESTADO) faça
      filho ← NÓ-FILHO(problema, nó, ação),
      se (filho.ESTADO)não está em explorado ou borda então
        se problema.TESTE-DE-OBJETIVO(filho.ESTADO) então retorne SOLUÇÃO(filho)
        borda ← INSIRA(filho, borda)
```

Estratégias de busca sem informação

4. Busca de Custo Uniforme



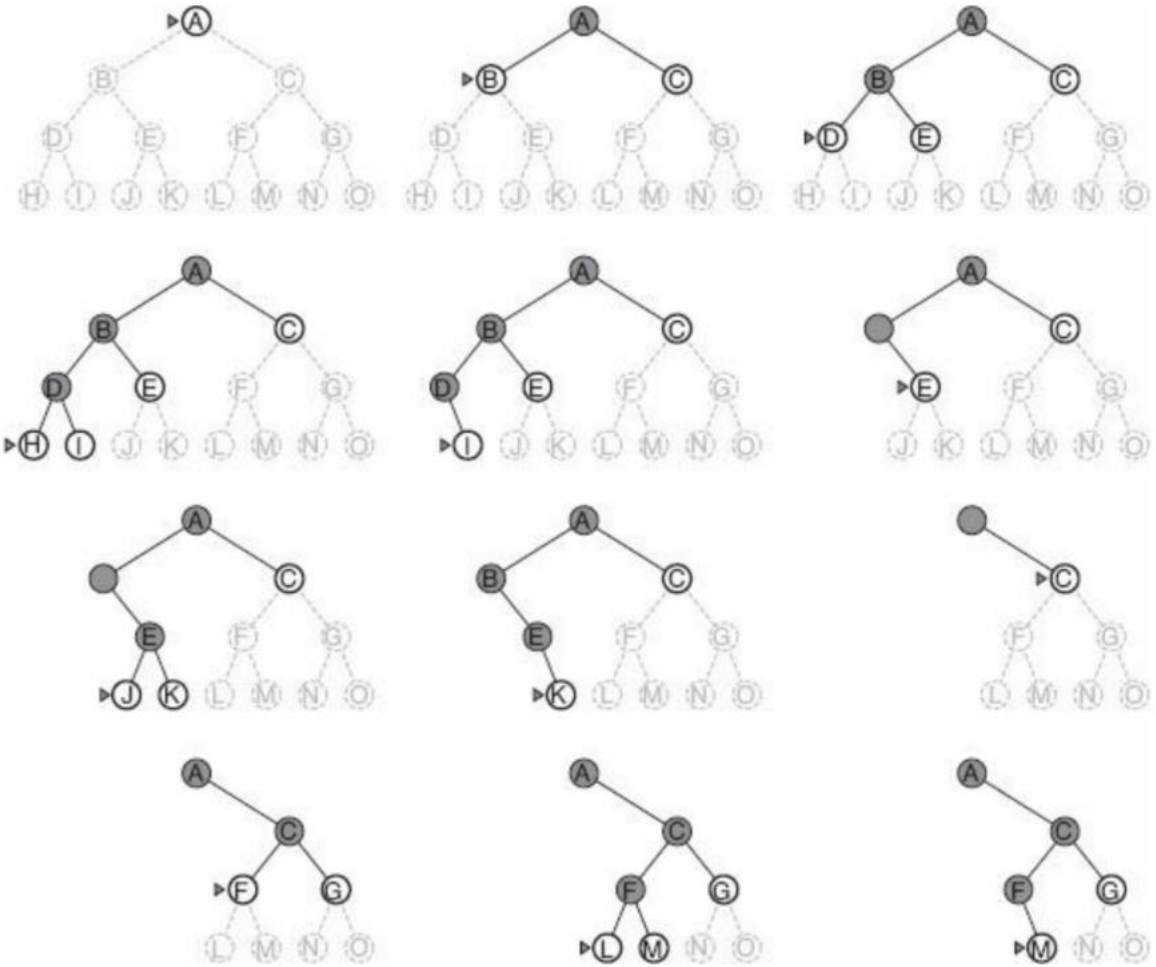
Passos	Nó atual	LISTA	Nó atual é o objetivo?	Qual o menor Caminho?
1°	A	A-C (14)	Não	A-B (12)

Estratégias de busca sem informação

```
função BUSCA-DE-CUSTO-UNIFORME(problema) retorna uma solução ou falha
  nó ← um nó com ESTADO = problema.ESTADO-INICIAL, CUSTO-DE-CAMINHO = 0
  borda ← fila de prioridade ordenada pelo CUSTO-DE-CAMINHO, com nó como elemento
único
  explorado ← um conjunto vazio
  repita
    se VAZIO?(borda), então retornar falha
    nó ← POP(borda) / * escolhe o nó de menor custo na borda */
    se problema.TESTE-OBJETIVO(nó.ESTADO) então retornar SOLUÇÃO(nó)
    adicionar (nó.ESTADO) para explorado
    para cada ação em problema. AÇÕES(nó.ESTADO) faça
      filho ← NÓ-FILHO (problema, nó, ação)
      se (filho.ESTADO) não está na borda ou explorado então
        borda ← INSIRA (filho, borda)
      senão se (filho.ESTADO) está na borda com o maior CUSTO-DE-CAMINHO então
        substituir aquele nó borda por filho
```

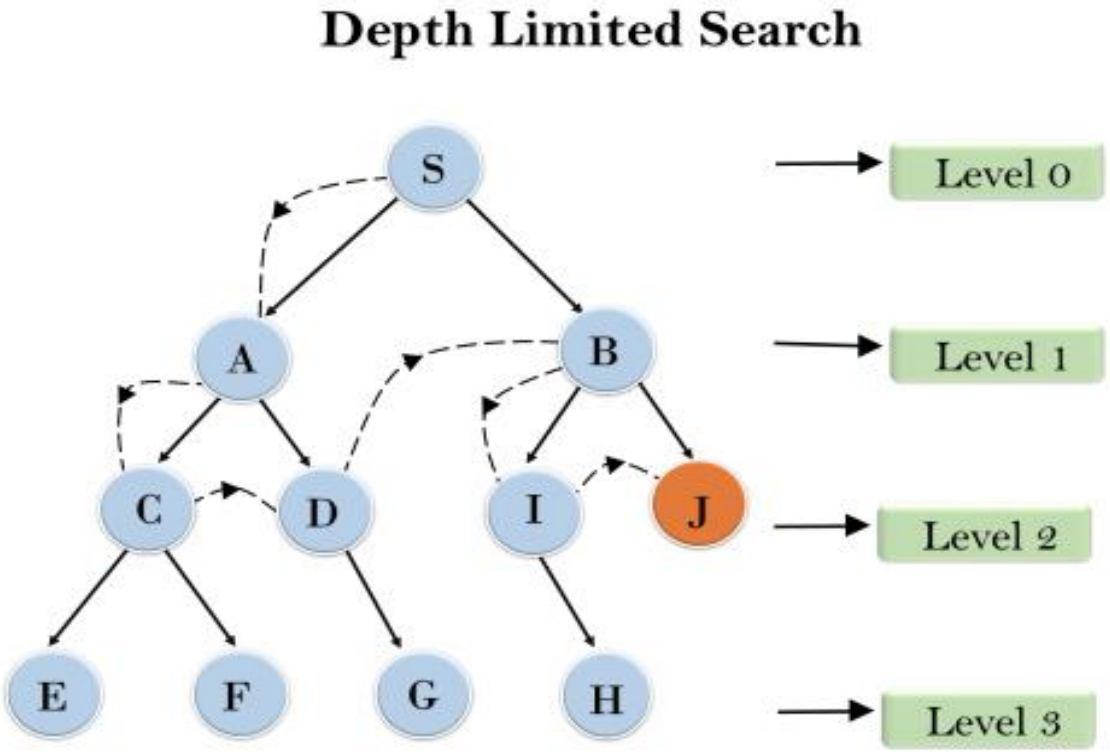
Estratégias de busca sem informação

Busca em profundidade



Estratégias de busca sem informação

Busca em profundidade limitada



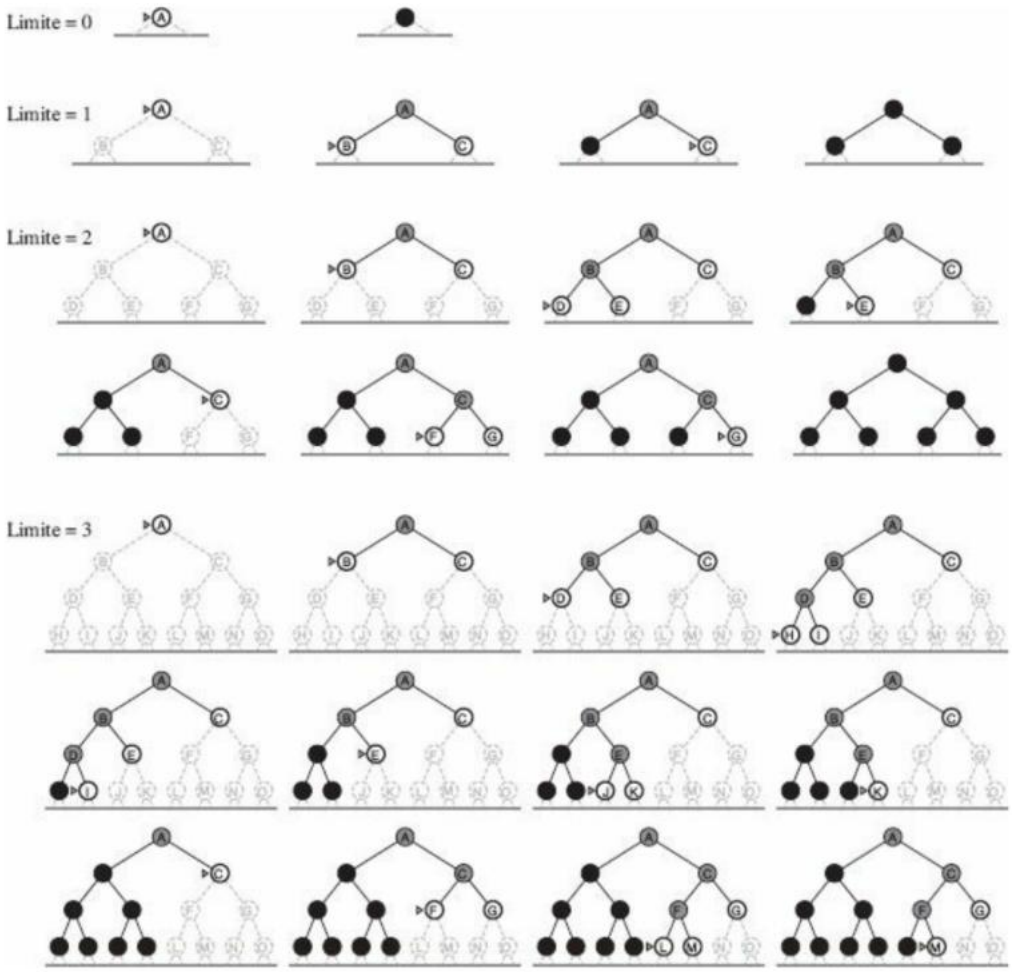
Estratégias de busca sem informação

```
function DEPTH-LIMITED-SEARCH(problem, limit) returns a solution, or failure/cutoff
  return RECURSIVE-DLS(MAKE-NODE(problem.INITIAL-STATE), problem, limit)

function RECURSIVE-DLS(node, problem, limit) returns a solution, or failure/cutoff
  if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
  else if limit = 0 then return cutoff
  else
      cutoff_occurred? ← false
      for each action in problem.ACTIONS(node.STATE) do
          child ← CHILD-NODE(problem, node, action)
          result ← RECURSIVE-DLS(child, problem, limit − 1)
          if result = cutoff then cutoff_occurred? ← true
          else if result ≠ failure then return result
      if cutoff_occurred? then return cutoff else return failure
```

Estratégias de busca sem informação

Busca de aprofundamento iterativo



Estratégias de busca sem informação

função BUSCA-DE-APROFUNDAMENTO-ITERATIVO(*problema*) **retorna** uma solução ou falha
para profundidade = 0 **até** ∞ **faça**
resultado ← BUSCA-EM-PROFUNDIDADE-LIMITADA(*problema*, *profundidade*)
se *resultado* ≠ *corte* **então retornar** *resultado*

Comparando estratégias de busca não informada

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

Figure 3.21 Evaluation of tree-search strategies. b is the branching factor; d is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: [a] complete if b is finite; [b] complete if step costs $\geq \epsilon$ for positive ϵ; [c] optimal if step costs are all identical; [d] if both directions use breadth-first search.

Veículos Científicos: Journals

- Journal of Machine Learning Research www.jmlr.org
- Machine Learning
- IEEE Transactions on Neural Networks
- IEEE Transactions on Pattern Analysis and Machine Intelligence
- Annals of Statistics
- Journal of the American Statistical Association
- ...

Veículos Científicos: Conferences

- International Conference on Machine Learning (ICML)
- European Conference on Machine Learning (ECML)
- Neural Information Processing Systems (NIPS)
- Computational Learning
- International Joint Conference on Artificial Intelligence (IJCAI)
- ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)
- IEEE Int. Conf. on Data Mining (ICDM)

Sources

- "Statistical Pattern Recognition: A Review"
  - Jain, Anil. K; Duin, Robert. P.W.; Mao, Jianchang (2000). "Statistical pattern recognition: a review". *IEEE Transtactions on Pattern Analysis and Machine Intelligence* **22** (1): 4-37
- "Machine Learning" Online Course
  - Andrew Ng
  - http://openclassroom.stanford.edu/MainFolder/CoursePage.php?course=MachineLearning
- "Machine Learning" Course
  - Kilian Weinberger
  - http://www.cse.wustl.edu/~kilian/cse517a2010/

"Sejamos a mudança que queremos ver no mundo". Gandhi

filipedwan@gmail.com

;-)

skype filipedwan

twitter @filipedwan