1

Agentes que resolvem problemas através de busca

Capítulo 3
Parte I

2

Agente reativo simples

- também chamados de agentes situados ou agentes estímulo-resposta
- reagem a estímulos do ambiente
- não possuem estado interno
- não é capaz de planejar com antecedência;
- o projetista escolhe e *hard-wire* o goal dentro do agente (em oposição à construir agentes com capacidade de tomada de decisões);
- não sabe o que suas ações podem realiza

3

Agente que Resolve Problemas

- Decide o que fazer (formula o seu próprio goal)
- Conhece os efeitos de suas ações
- Encontra uma sequência de ações que o leve a um estado do mundo desejado (estado goal)

4

Agente que Resolve Problemas

- *Goal:* conjunto de estados do mundo onde o *goal* é satisfeito (especificação parcial do mundo)
- *Ações:* causam transições de um estado para outro

Agente:
- Qual é o estado *goal* que devo atingir?
- Que tipo de ações eu disponibilizo?
- Que mudanças no mundo eu provoco?
- Quais ações me levam ao estado *goal*?

Agente "deliberativo"

5

FORMULAÇÃO
BUSCA
EXECUÇÃO

- *Formulação do goal*: processo de determinação do goal
- *Formulação do problema:* escolha de um conjunto relevante de estados e ações
- *Busca:* "imaginar" ou "deliberar" as sequências de ações que atinjam o goal
- *Solução*: conjunto de passos que atinge o goal
- *Execução* : realização dos passos/ações da solução

Agente "deliberativo"

6

FORMULAÇÃO
BUSCA
EXECUÇÃO

função Agente-Deliberativo-Simples (*p*) **devolve** *ação*
  ***entradas*:** *p,* uma percepção
  estático: *s*, um sequência de estados, inicialmente vazia
    estado, alguma descrição do estado do mundo
    g, um goal, inicialmente nulo
    problema, uma formulação de problema
  estado ← ATUALIZA-ESTADO (*estado, p*)
  se *s* vazia ***então***
    g ← FORMULA-GOAL (*estado)*
    problema ← FORMULA-PROBLEMA (*estado, g*)
    s ← BUSCA (*problema)*
  ação ← RECOMENDA (*s, estado*)
  s ← RESTO (*s, estado)*
  devolve *ação*

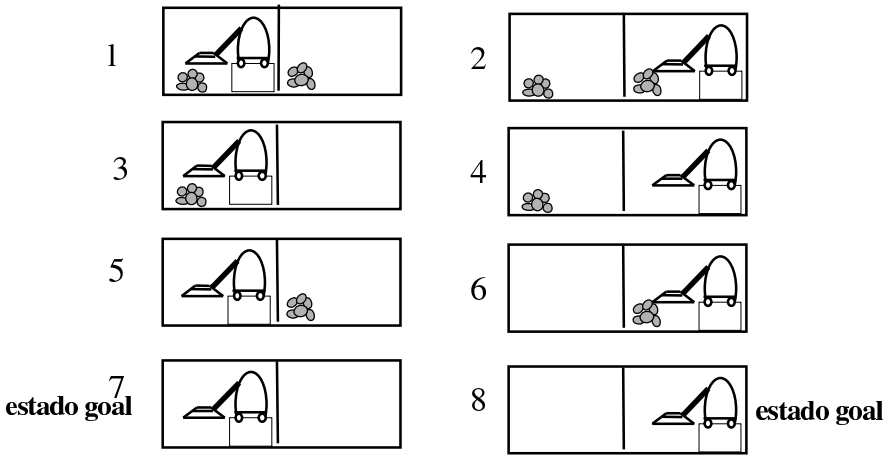
7

Formulação de problemas

- Depende de como o agente está conectado ao seu ambiente através de suas percepções e ações)
- Existem 4 tipos diferentes de problemas:
  - problemas de estado único
  - problemas de multi-estados
  - problemas de contingência
  - problemas de exploração

8

O mundo do aspirador



9

O mundo do aspirador

- possue 2 localizações
- 8 estados possíveis
- 3 ações (efeito conhecido):*Esquerda, Direita e Aspira*
- gol: *Eliminar toda a sujeira dos 2 cômodos*
- tipos de sensores: sensor de posição; sensor local de sujeira
- Tipos de ambiente:
  - inacessível / acesso limitado / acesso total
  - conhecimento total sobre as ações (determinístico)
  - não-determinismo (Lei de Murphy): a ação *Aspira* **algumas vezes** deposita sujeira no carpete mas **somente se não existe sujeira**

10

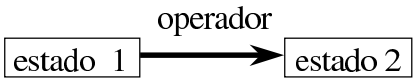
Formulação de problemas:
diferentes tipos de conhecimento que o agente possue

① Problemas de estado simples	• ações: efeitos conhecidos (determinismo) • estado: acesso total (estado exato)
② Problemas de estados múltiplos	• ações: efeitos parcialmente conhecidos (não derminismo) *ou* • estado: acesso limitado • requer raciocínio sobre um conjunto de estados
③ Problemas de contingência	• ações com efeitos condicionais (não-determinismo) • estado: acesso limitado (durante execução) • requer mistura de planejamento e execução
④ Problemas de exploração	• estado: inacessível e não determinístico • execução revelam estados possíveis do mundo • ações: precisa executar para descobrir os efeitos de ações. Busca no mundo real

11

Formulação de Problema

- Envolve o conhecimento que um agente deve usar para resolver o problema:
  - estado inicial (que o próprio agente sabe estar)
  - conjunto de ações (operadores) disponíveis ao agente
    - Operadores modelam as ações: mudam o estado do mundo



  - teste de goal: checa por um conjunto explícito de estados goal ou por uma propriedade do mundo (ex.: xadrez)
  - função custo de caminho (*g* → soma dos custos das ações individuais em um caminho

12

Tipo de Dado Problema

tipo-de-dado Problema

componentes: estado-inicial, operadores, teste-do-goal, função-custo

13

Espaços de problemas

- **Espaço de estados** do problema: conjunto de todos os estados possíveis a partir de qualquer estado inicial.
- **Espaço de busca**: conjunto de todos os estados alcançáveis aplicando-se alguma combinação de operadores ao estado inicial (uma medida do tamanho de complexidade)
- **Representação de espaço de busca**
  - Estados do mundo representados como nós de uma árvore
    - Estado inicial: raiz da árvore
    - Estado Gol: folha(s) da árvore

14

Busca

Objetivo:

- Procurar por uma sequência de ações que transformam o estado inicial do mundo no estado goal

Possibilidade:

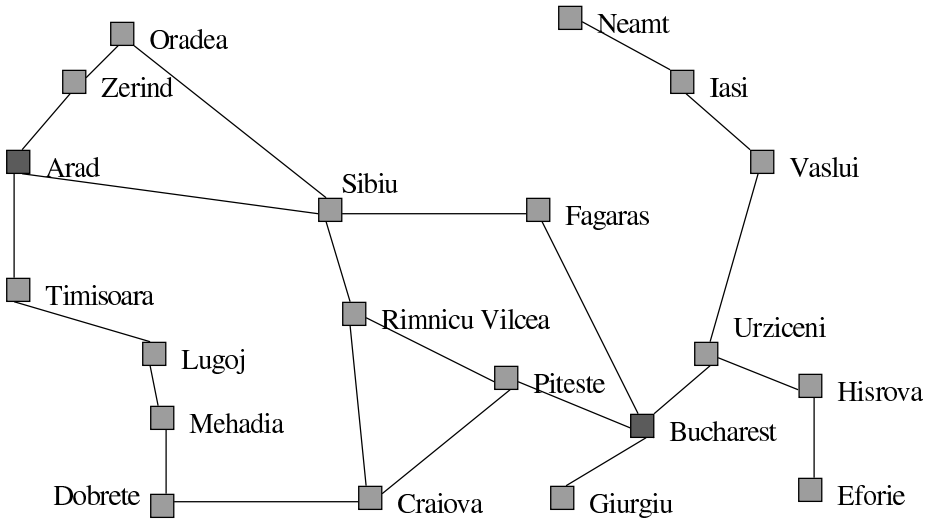
- Encontrar uma solução (qualquer solução)
- Encontrar a solução ótima (custo de caminho baixo)

Custo do caminho x Custo da busca

- compromisso entre encontrar a melhor solução e o tempo gasto em busca

15

Mapa de estradas



16

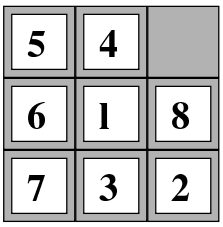
Exemplo: Romênia

Férias na Romênia. No momento em Arad. O vôo sai no dia seguinte para Bucharest.

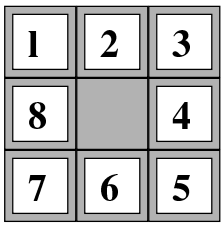
- Formular goal:
  - chegar em Bucharest
- Formular problema:
  - *estados*: várias cidades
  - *ações*: dirigir entre cidades
  - *função custo*: kilometragem total ou tempo de viajem
  - *teste de goal:* estou em Bucharest?
- Encontrar solução:
  - sequência de cidades, por exemplo: Arad, Sibiu, Fagaras, Bucharest

17

8-puzzle ou 15-puzzle



Estado Inicial

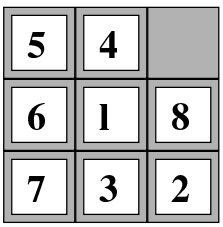


Estado Gol

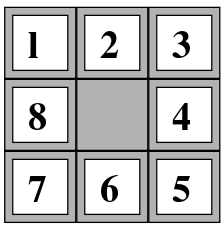
Estados: ??
Operadores: ??
Teste de gol: ??
Função custo: ??

18

8-puzzle ou 15-puzzle



Estado Inicial

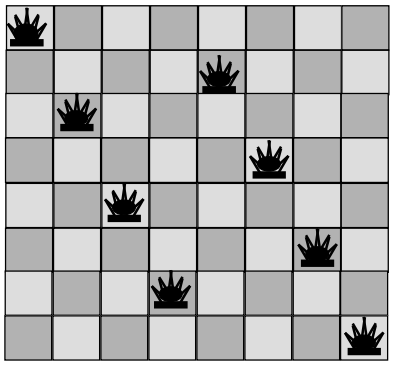


Estado Gol

Estados: posição de cada uma das 8 peças em um dos 9 quadrados
Operadores: quadrado vazio move para a esquerda, direita, para cima ou para baixo
Teste de gol: estado tem que casar com a configuração do Estado Gol
Função custo: cada passo custa 1 ⟶ comprimento do caminho.

36288 estados diferentes

Problema das 8-rainhas

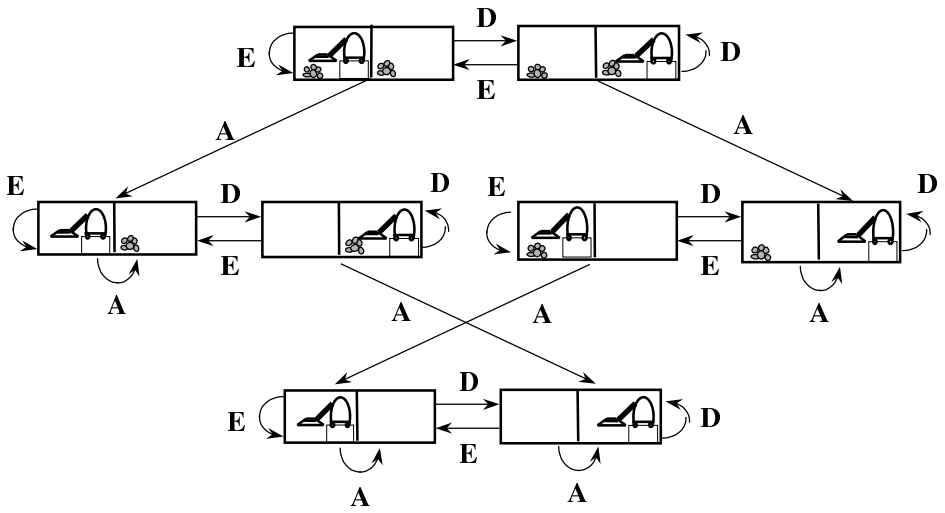


Teste de gol: 8 rainhas no tabuleiro sem perigo de ataque
A formulação correta faz uma grande diferença no tamanho do espaço de busca (64^8 versus 2057 sequências possíveis para serem investigadas)

Problema do aspirador

Estados: um dos 8 estados
Operadores: move para direita (D), move para esquerda (E), aspira (A)
Teste de gol: nenhuma sujeira em nenhum quadrado
Custo do caminho: cada passo custa 1

Espaço de estados do mundo do aspirador



Árvore de busca

- Nós da árvore de busca - componentes:
  - estado correspondente do espaço de estados
  - o nó da árvore de busca que o gerou (*nó pai*)
  - o operador que foi aplicado para gerá-lo
  - o no. de nós no caminho a partir da raiz (*profundidade*)
  - o custo do caminho da raiz até o nó
- função Expande: calcula cada um dos componentes dos nós gerados

23

Árvore de busca em espaço de estados

- Estrutura de dados imposta sobre o espaço de estados
- Nós → estados do mundo
- X é gerado a partir de Y:
  - X → nó filho ou sucessor
  - Y → nó pai ou antecessor
- Raiz → nó sem pai
  - → estado inicial ou estado final **
- Nó folha ou terminal → nó sem filhos

24

O que um algoritmo de busca faz:

- gera sequência de ações
- múltiplas escolhas de caminho
- essência de busca: escolher uma opção e deixar outras de lado p/ mais tarde
- a decisão de quem escolher primeiro define uma estratégia de busca

Algoritmo geral de busca

função Busca-Geral(*problema*, *estratégia*) **devolve** uma solução, ou falha

inicializa a árvore de busca com o estado inicial do *problema*

laço faça

se não há mais candidatos para expandir **então devolve** falha

escolha um nó folha para expandir de acordo com a *estratégia*

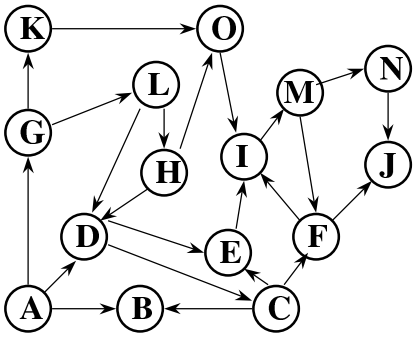
se o nó contém um estado gol **então devolve** a solução

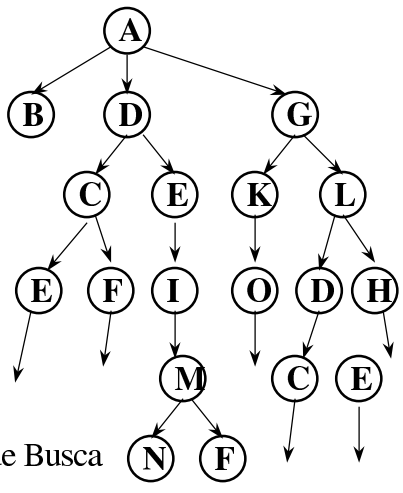
senão expanda o nó e adicione na árvore de busca

fim

Problema geral de busca

Espaço de Estados





Árvore de Busca

Estratégia de Busca - avaliação

- Completude:
  - a estratégia garante encontrar uma solução, se existir uma?
- Qualidade da solução:
  - a estratégia encontra a solução ótima?
- Complexidade de tempo:
  - quanto tempo gasta para encontrar uma solução?
- Complexidade de memória:
  - quanto de memória é preciso?
- Que características deve ter um espaço de busca para que a estratégia seja adequada ?

Estratégias de busca não-informada

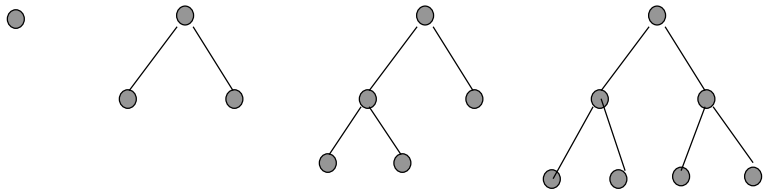
- BUSCA CEGA: não usa informações sobre o domínio. Variam quanto a ordem em que nós são expandidos:
  - Busca em largura - *Breadth first*
  - Busca em profundidade - *Depth first*
  - Busca de custo uniforme
  - Busca em profundidade limitada
  - Busca em profundidade iterativa
  - Busca bidirecional

Não possuem informação sobre o número de passos ou o custo do caminho entre o um nó e o gol.

Busca em largura

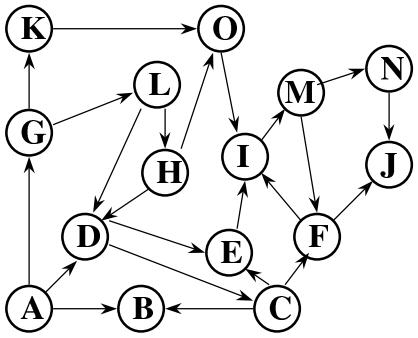
- Expande todos os nós do mesmo nível de pronfundidade (*d* antes de *d+l)*
  - completude: garante encontrar uma solução
  - se existir mais do que uma solução, encontra a mais próxima da raiz primeiro
  - complexidade de tempo e memória: variam exponencialmente com a profundidade

Busca em largura - avaliação



- *b - branching factor, d - depth of the solution*
- máximo no. de nós antes de encontrar a solução:
  $1 + b + b^2 + b^3 + ... + b^d \longrightarrow O(b^d)$
- problema: todos os nós das folhas devem ser mantidos na memória

Busca em largura



Estado Inicial: A

A B D G C E L K E F I D H O

Busca em largura - algoritmo

Cheque todos os nós filhos em um nível antes de expandir a árvore em mais um nível de profundidade. Elimine estados repetidos (não é preciso checá-los novamente)

Algoritmo:

Insira o nó inicial na lista.
Se nó inicial é o gol, FIM.
Se lista vazia, não existe solução.
Senão: selecione o primeiro nó da lista.
Se nó selecionado é o gol, FIM.
Senão expande o nó selecionado e insira todos os seus sucessores no final da lista.
Repita.