



---

---


---

---


---

---

---



### Plano de Ensino



- Apresentação da Disciplina.
- Introdução à Inteligência Artificial.
- Agentes Inteligentes.
- **Resolução de Problemas.**
- Mecanismos de Busca.
- Formas de Raciocínio Artificial.
- Representação do Conhecimento.
- Redes Semânticas.
- Aquisição de Conhecimento.
- Sistemas Especialistas.
- Sistemas Multiagentes.
- Redes Neurais.
- Mineração de Dados.

---

---


---

---


---

---

---



### Livro-Texto



- Bibliografia Básica:
  - » RUSSELL, Stuart J.. Inteligencia Artificial. 2ª ed. Rio de Janeiro: Campus - Elsevier, 2004.
- Bibliografia Complementar:
  - » LUGER, G.F.. Inteligência Artificial : Estruturas e Estratégias para a Resolução de Problemas Complexos. 4ª ed. Porto Alegre: Artmed, 2004.

---

---

---

---

---

---

---

### 3. Resolução de Problemas – Introdução



- Agentes reativos mapeiam suas ações em mapeamento direto de estados e ações.
  - » Não operam bem em ambientes onde o mapeamento de regras condição-ação é muito grande para se armazenar.
  - » O aprendizado demanda muito tempo.
- Agentes baseados em objetivos chamados de resolução de problemas podem ser construídos.
  - » Decidem o que fazer encontrando sequência de ações que levam a estados desejáveis.

---

---

---

---

---

---

---

### 3. Resolução de Problemas – Introdução



- Agentes inteligentes devem maximizar sua medida de desempenho → devem adotar um objetivo e satisfazê-lo. Exemplo:
  - » agente na cidade de Arad (Romênia) em viagem de férias.
  - » medida de desempenho:
    - melhorar bronzeados,
    - melhorar seu conhecimento do idioma,
    - ver paisagens, etc.
- O problema de decisão é complexo e envolve muitos compromissos e objetivos.
  - » Imagine agora que o agente possui uma passagem não-reembolsável para partir de Bucharest na manhã seguinte.
    - Objetivo → chegar a Bucharest a tempo.
    - Caminhos que não chegam a Bucharest devem ser descartados.
    - Problema de decisão do agente é simplificado.

---

---

---

---

---

---

---

### 3. Resolução de Problemas – Introdução



- *Formulação de objetivos* → primeiro passo para a resolução de problemas; é baseada na situação atual e na medida do desempenho do agente.
- *Formulação de problemas* → processo de decidir que ações e estados devem ser considerados, dado um objetivo.
- *Busca* → processo de procurar pela melhor sequência.
- *Solução* → a melhor sequência de ações encontrada.
- *Execução* → ações são executadas.

**Formular objetivo → buscar → executar**

---

---

---

---

---

---

---

### 3. Resolução de Problemas – Introdução



**função** Agente-de-Resolução-de-Problemas-Simples(*percepção*) **retorna** uma ação  
**entradas:** *percepção*, uma percepção  
**variáveis estáticas:** *seq*, uma sequência, inicialmente vazia  
                                  *estado*, alguma descrição do estado atual do mundo  
                                  *objetivo*, um objetivo, inicialmente nulo  
                                  *problema*, uma formulação de problema  
*estado*  $\leftarrow$  Atualizar-Estado(*estado*, *percepção*)  
**se** *seq* está vazia **então faça**  
          *objetivo*  $\leftarrow$  Formular-Objetivo(*estado*)  
          *problema*  $\leftarrow$  Formular-Problema(*estado*, *objetivo*)  
          *seq*  $\leftarrow$  Busca(*problema*)  
          *ação*  $\leftarrow$  Primeiro(*seq*)  
          *seq*  $\leftarrow$  Resto(*seq*)  
**retornar** *ação*

- Supõe que ambiente é estático, observável, discreto e determinístico.

7

---

---

---

---

---

---

---

---

### 3. Resolução de Problemas – Introdução



- Problemas e soluções bem definidos  $\rightarrow$  um problema pode ser definido formalmente por 4 componentes:
  1. Estado inicial  $\rightarrow$  estado em que o agente começa.
    - Exemplo: cidade de Arad.
  2. Ações ou função sucessor  $S(x)$   $\rightarrow$  dado um estado particular  $x$ , retorna um conjunto de pares ordenados (*ação*, *sucessor*), onde uma *ação* é válida a partir de  $x$  e *sucessor* é um estado alcançável a partir de  $x$ .
    - Exemplo:  $S(\text{Arad}) = \{(\text{Arad} \rightarrow \text{Sibiu}, \text{Sibiu}), (\text{Arad} \rightarrow \text{Zerind}, \text{Zerind}), (\text{Arad} \rightarrow \text{Timisoara}, \text{Timisoara})\}$

8

---

---

---

---

---

---

---

---

### 3. Resolução de Problemas – Introdução



3. Teste de objetivo  $\rightarrow$  determina se um dado estado é objetivo.
    - Explícito  $\rightarrow$  Exemplo: chegar em Bucharest.
    - Implícito  $\rightarrow$  Exemplo: cheque-mate no jogo de xadrez.
  4. Custo do caminho  $\rightarrow$  o caminho possui um custo e o agente escolhe uma função de custo que reflita sua própria medida de desempenho.
    - Exemplo: soma das distâncias, número de ações executadas, etc.
    - $c(x,a,y)$  é o custo do passo, que deve ser sempre  $\geq 0$ .
- *Solução*  $\rightarrow$  uma sequência de ações que levam do estado inicial para o estado objetivo.
  - *Solução ótima*  $\rightarrow$  uma solução com o menor custo de caminho.

---

---

---

---

---

---

---

---

### 3. Resolução de Problemas – Introdução



- O conjunto de todos os estados acessíveis a partir de um estado inicial é chamado de espaço de estados.
  - » Os estados acessíveis são aqueles dados pela função sucessora.
- O espaço de estados pode ser interpretado como um grafo em que os nós são estados e os arcos são ações.
- O mundo real é absurdamente complexo desta forma é necessário remover detalhes de uma representação através da abstração.
  - » Abstração de estado → remoção de detalhes que não sejam relevantes aos estados como: condições da pista, condições do tempo durante a viagem, passageiros, etc.
  - » Abstração das ações → consumo de combustível, geração de poluição, etc.

---

---

---

---

---

---

---

### 3. Resolução de Problemas – Exemplo



- Problema Romênia
  - » de férias na Romênia; atualmente em Arad.
  - » voo marcado para o próximo dia saindo de Bucharest.
- Formular objetivo:
  - » estar em Bucharest em tempo hábil
- Formular problema:
  - » estados: cidades
  - » ações: dirigir entre as cidades
- Encontrar solução:
  - » sequência de cidades: Arad, Sibiu, Fagaras, Bucharest.

---

---

---

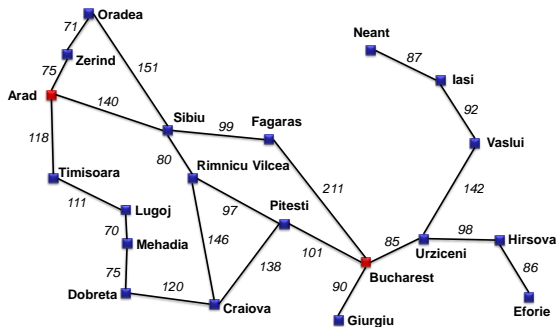
---

---

---

---

### 3. Resolução de Problemas – Exemplo



---

---

---

---

---

---

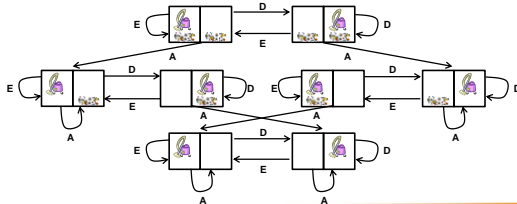
---

### 3. Resolução de Problemas – Miniproblemas



#### ▪ Aspirador de pó

- » **Estados:** o agente está em 1 dentre 2 posições, cada posição podendo conter sujeira ou não, ou seja:  $2 \times 2^2 = 8$  estados.
- » **Estados inicial:** qualquer estado poder ser considerado inicial.
- » **Função sucessor:** estados válidos para as ações (E, D, A).
- » **Teste de objetivo:** todos os quadrados limpos.
- » **Custo do caminho:** cada passo custa 1.




---

---

---

---

---

---

---

---

### 3. Resolução de Problemas – Miniproblemas



#### ▪ Quebra-cabeça de 8 peças

- » **Estados:** a posição de cada uma das peças e do espaço vazio.
- » **Estado inicial:** qualquer estado.
- » **Função sucessor:** estados válidos para as quatro ações (mover espaço vazio para esquerda, direita, acima ou abaixo).
- » **Teste de objetivo:** se o estado corresponde ao estado objetivo.
- » **Custo do caminho:** cada passo custa 1.

7	2	4
5		6
8	3	1

Estado inicial

	1	2
3	4	5
6	7	8

Estado objetivo

---

---

---

---

---

---

---

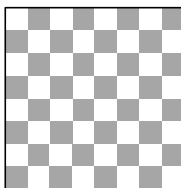
---

### 3. Resolução de Problemas – Miniproblemas

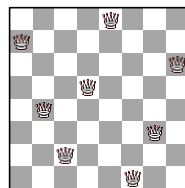


#### ▪ Tabuleiro de Xadrez com 8 rainhas

- » **Estados:** qualquer disposição de 0 a 8 rainhas ( $3 \times 10^{14}$ ).
- » **Estado inicial 1:** nenhuma rainha.
- » **Função sucessor 1:** colocar 1 rainha em qualquer vazio.
- » **Teste de objetivo:** 8 rainhas no tabuleiro, nenhuma atacada.



Estado inicial



Estado final

---

---

---

---

---

---

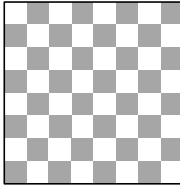
---

---

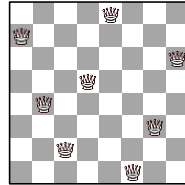
### 3. Resolução de Problemas – Miniproblemas



- Tabuleiro de Xadrez com 8 rainhas
  - » **Estados:** qualquer disposição de 0 a 8 rainhas (2.057).
  - » **Estado inicial 2:** disposições de  $n$  rainhas ( $0 \leq n \leq 8$ ), uma por coluna nas  $n$  colunas mais à esquerda, sem sofrer ataque.
  - » **Função sucessor 2:** colocar 1 rainha em qualquer quadrado na coluna vazia mais à esquerda, sem sofrer ataque.
  - » **Teste de objetivo:** 8 rainhas no tabuleiro, nenhuma atacada.



Estado inicial



Estado final

---

---

---

---

---

---

---

---

### 3. Resolução de Problemas – Mundo Real



- Problema de roteamento → encontrar a melhor rota de um ponto a outro (aplicações: redes de computadores, planejamento militar, planejamento de viagens aéreas).
- Problema de turista → visitar cada ponto pelo menos uma vez.
- Caixeiro viajante → visitar cada cidade exatamente uma vez; encontrar o caminho mais curto.
- Layout de VLSI → posicionamento de componentes e conexões em um chip.
- Projeto de proteínas → encontrar uma sequência de aminoácidos que serão incorporados em uma proteína tridimensional para curar alguma doença.
- Pesquisas na Web → robôs de software que fazem pesquisa na Internet procurando por respostas a perguntas; oportunidades de compra; e informações inter-relacionadas.

---

---

---

---

---

---

---

---

### 3. Resolução de Problemas – Busca de Solução



- Depois de formular alguns problemas, precisamos resolvê-los.
- Devemos realizar buscas em todo o espaço de estados, através de **árvore de busca explícita**, gerado pelo estado inicial e pela função sucessor.
- A raiz da árvore de busca é um nó de busca que corresponde ao estado inicial.
  - » Primeiro passo é testar se o estado inicial é um estado objetivo.
  - » Caso não seja, considerar outros estados, através da expansão do estado atual, gerando então um novo conjunto de estados.
- A **estratégia de busca** é determinar o caminho a seguir e deixar as outras reservadas para mais tarde.

---

---

---

---

---

---

---

---

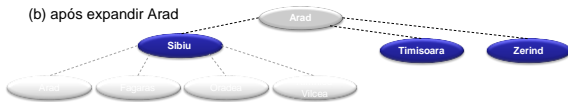
### 3. Resolução de Problemas – Busca de Solução



(a) estado inicial Arad



(b) após expandir Arad



(c) após expandir Sibiu




---

---

---

---

---

---

---

---

### 3. Resolução de Problemas – Busca de Solução



- Descrição informal do algoritmo de busca em árvore.

**função** Busca-Em-Árvore(*problema, estratégia*) **retorna** uma solução ou falha  
inicializar a árvore de busca com o estado inicial do problema

**repita**

- se** não existe candidato para expansão **então retornar** falha
- escolher um nó folha para expansão de acordo com a *estratégia*
- se** nó contém um estado objetivo **então retornar** solução encontrada
- senão** expandir o nó e adicionar os nós resultantes à árvore de busca

---

---

---

---

---

---

---

---

### 3. Resolução de Problemas – Busca de Solução



- No exemplo de localização de rotas da Romênia, problema de direção entre Arad → Bucharest, é importante evitar caminhos infinitos, por exemplo:
  - » Arad → Sibiu,
  - » Arad → Sibiu → Arad
  - » Arad → Sibiu → Arad → Sibiu, ...
- Um bom algoritmo de busca evitaria seguir caminhos repetidos.

---

---

---

---

---

---

---

---

### 3. Resolução de Problemas – Busca de Solução



#### ▪ Representação de nós → estrutura com 5 componentes:

- » *Estado* → o estado no espaço de estados a que o nó corresponde.
- » *Nó-pai* → o nó da árvore de busca que gerou esse nó.
- » *Ação* → a ação que foi aplicada ao pai para gerar o nó.
- » *Custo-do-caminho* → o custo, tradicionalmente denotado por  $g(n)$ , do caminho do estado inicial até o nó.
- » *Profundidade* → o número de passos ao longo do caminho, desde o estado inicial.



---

---

---

---

---

---

---

---

### 3. Resolução de Problemas – Busca de Solução



#### ▪ Borda (*fringe*) → coleção de nós gerados mais ainda não expandidos.

- » Geralmente implementados como uma fila.
- » Cada elemento da borda é chamado de nó-folha, ou seja, um nó sem sucessores na árvore.
- » A maneira como os nós entram na fila determina a estratégia de busca. Operações sobre uma fila:
  - *Cria-fila(elemento, ...)* → cria uma fila com o(s) elemento(s) dado(s).
  - *Vazia(fila)* → retorna V se não existir mais nenhum elemento na fila.
  - *Primeiro(fila)* → retorna o primeiro elemento da fila.
  - *Remover-primeiro(fila)* → retorna *Primeiro(fila)* e remove-o.
  - *Inserir(elemento, fila)* → insere um elemento na fila e retorna a fila resultante.
  - *Inserir-todos(elementos, fila)* → insere um conjunto de elementos na fila e retorna a fila resultante.

---

---

---

---

---

---

---

---

### 3. Resolução de Problemas – Busca de Solução



#### ▪ Medição do desempenho de resolução de problemas: a saída de um algoritmo de resolução de problemas consiste em falha ou em uma solução. O desempenho é avaliado em 4 aspectos:

- » *Completeza* → o algoritmo oferece a garantia de encontrar uma solução quando ela existir?
- » *Otimização* → a estratégia encontra a solução ótima, ou seja, o menor custo de caminho?
- » *Complexidade de tempo* → quanto tempo ele leva para encontrar uma solução?
- » *Complexidade de espaço* → quanta memória é necessária para executar a busca?

---

---

---

---

---

---

---

---



### 3. Resolução de Problemas – Busca de Solução



**função** Busca-Em-Árvore(*problema*, *borda*) **retorna** uma solução ou falha  
*borda*  $\leftarrow$  Inserir(Criar-nó(Estado-Inicial(*problema*)), *borda*)  
**repita**  
  **se** Vazia(*borda*) **então retornar** falha  
  *nó*  $\leftarrow$  Remover-primeiro(*borda*)  
  **se** Testar-Objetivo(*problema*) aplicado a Estado(*nó*) tem sucesso  
  **então retornar** Solução(*nó*)  
  *borda*  $\leftarrow$  Inserir-Todos(Expandir(*nó*, *problema*), *borda*)

**função** Expandir(*nó*, *problema*) **retorna** um conjunto de nós  
*sucessores*  $\leftarrow$  um novo Nó  
**para cada** <ação, resultado> em Sucessor(*problema*)(Estado(*nó*)) **faça**  
  *s*  $\leftarrow$  um novo Nó  
  Estado[*s*]  $\leftarrow$  resultado  
  Nó-pai[*s*]  $\leftarrow$  *nó*  
  Ação[*s*]  $\leftarrow$  ação  
  Custo-do-caminho[*s*]  $\leftarrow$  Custo-do-caminho[*nó*] + Custo-do-passo(*nó*, ação, *s*)  
  Profundidade[*s*]  $\leftarrow$  Profundidade[*nó*] + 1  
  adicionar *s* a *sucessores*  
**retornar** *sucessores*

---

---

---

---

---

---

---

---

### 3. Resolução de Problemas – Busca de Solução



- Complexidade de tempo e espaço são sempre consideradas em relação a alguma medida de dificuldade do problema.
  - » Em Computação Teórica, a complexidade é definida pelo tamanho do grafo.
  - » Em IA, a complexidade de tempo e espaço são definidas em termos de:
    - *b*  $\rightarrow$  fator de ramificação (número máximo de sucessores de qualquer nó);
    - *d*  $\rightarrow$  a profundidade do nó objetivo menos profundo;
    - *m*  $\rightarrow$  o comprimento máximo de qualquer caminho no espaço de estados (pode ser  $\infty$ ).

---

---

---

---

---

---

---

---



Inteligência Artificial

Ciência da Computação  
[clayton.valdo@anhanguera.com](mailto:clayton.valdo@anhanguera.com)



---

---

---

---

---

---

---

---