Inteligência Artificial

Fabrício Enembreck

PPGIA – Programa de Pós-Graduação
em Informática Aplicada

Definições de Inteligência Artificial

- "estudo de como fazer os computadores realizarem coisas que, atualmente, as pessoas fazem melhor." Rich e Knight
- Quatro definições de Russel e Norvig:
  - Sistemas que pensam como humanos
  - Sistemas que pensam racionalmente
  - Sistemas que agem como humanos
  - Sistemas que agem racionalmente

Pensando Humanamente

- Pesquisar como pessoas pensam
  - Empiricamente
  - Baseando-se em psicologia cognitiva
- Resolvedores gerais de problemas
- Modelos Comportamentais (observação)
- **Modelos Cognitivos** (estímulos, crenças, metas, raciocínio, ações)

3

Pensando Racionalmente

- Aristóteles: pensar racionalmente é um processo de raciocínio irrefutável.
- Aristóteles propôs os padrões de argumentos que representam raciocínios corretos. Esses **Silogismos** fazem parte da **Lógica**.
- Sócrates é um homem. Todos os homens são mortais. Logo, Sócrates é mortal.

4

Agindo Humanamente

- Turing 1950: comportamento inteligente é a capacidade de alcançar a performance humana em todas as tarefas cognitivas, suficientes para enganar um interrogador
- Seis capacidades básicas de um computador inteligente:
  - Processamento de linguagem natural
  - Representação de conhecimento
  - Raciocínio automatizado
  - Aprendizagem de máquina
  - Visão computacional
  - Robótica

5

Agindo Racionalmente

- Um agente deve agir para alcançar sua meta
- Percepção e ação
- **Agentes racionais**
- Inteligência pode ser um fenômeno social, não apenas um modelo de raciocínio intrínseco ao indivíduo

6

Um pouco de história

- 1943-1956 (O início)
  - McCulloch e Pitts (1943): Modelo de neurônios cerebrais e redes de neurônios para representar conectivos lógicos e o processo de aprendizagem
  - Shannon e Turing (1950): Programas de xadrez para computadores de von Neumann
  - Minsky e Edmonds (1951): Construíram a primeira rede neural para computador em Princetown (3000 válvulas para simular 40 neurônios)

7

Um pouco de história (cont.)

- John McCarthy (1955): Desenvolvimento da primeira linguagem funcional (LISP) para prova de teoremas. Convenceu Minsky e colegas a trabalhar em **Inteligência Artificial**.
- Workshop de Dartmouth (1956) reuniu pesquisadores em teoria dos autômatos, redes neurais e estudo da inteligência.
- Newell e Simon apresentaram um programa de raciocínio baseado em Lógica
- MIT, CMU, Stanford e IBM.

8

Um pouco de história (cont.)

- 1952-1969 (Entusiasmo)
  - Limitação da tecnologia
  - Newell e Simon: Provador geral de teoremas para puzzles com estratégias de raciocínio
  - Samuel (1952): Provador de teoremas para jogo de damas
  - McCarthy (1958): Mudou-se para o MIT, criou o LISP, criou o time-sharing, criou o Advice Taker
  - McCarthy (1963): Mudou-se para o MIT e aprimorou o Advice Taker com o método de resolução introduzido por Robinson
  - Minsky e os micro-mundos

9

Um pouco de história (cont.)

- 1966-1974 (Realismo)
  - Apesar das aplicações potenciais, os sistemas "inteligentes" da época eram muito especializados e problemas muito "pequenos"
  - Problema: Aplicações não utilizavam conhecimento, mas apenas substituições sintáticas (weak-methods)
  - Weizenbaum 1965 – ELIZA
  - Teoria dos problemas NP-completos
  - Friedberg (1959) – Estudos sobre algoritmos genéticos
  - Representações muito limitadas de comportamento inteligente (Minsky e redes neurais)

10

Um pouco de história (cont.)

- 1969-1979: Sistemas a base de conhecimento
  - Suprir a necessidade de conhecimento para aplicações de domínios específicos
  - Buchanan et al. (1969): DENDRAL – Dada uma fórmula molecular + massas o sistema previa todas as fórmulas derivadas + massas quando a fórmula era bombardeada por um elétron usando um conjunto de regras
  - Feigenbaum, Buchanan e Shortliffe (1972): **Sistema Especialista** MYCIN – diagnóstico de infecções sanguíneas c/ **tratamento de incertezas**

11

Um pouco de história (cont.)

- 1969-1979: Sistemas a base de conhecimento
  - Duda et al. (1979): PROSPECTOR - Sistema especialista para descoberta de jazidas de chumbo
  - Shank, Alberson, Riesbeck, Dyer: Conhecimento é necessário para se construir sistemas que compreendem linguagem natural
  - Woods (1973): LUNAR – Primeiro sistema de PLN que respondia questões sobre rochas trazidas da Lua
  - Esquemas de representação de conhecimento: Prolog (1972), Frames (Minsky, 1975), Redes Semânticas (Woods), Grafos Conceituais (Shank), etc.

12

Um pouco de história (cont.)

- Primeiras aplicações comerciais (1980-1988)
  - Sistemas especialistas
  - Sistemas de visão computacional
- O retorno de redes neurais (1986 - ...)
- Hoje em dia
  - Reconhecimento de Padrões (voz, imagem,som)
  - Raciocínio Incerto (Fuzzy, Probabilista)
  - Processamento em Linguagem Natural
  - Mineração e aquisição de conhecimento a partir de dados
  - Inteligência Artificial Distribuída – Agentes Inteligentes
  - Programação Genética/Algoritmos Genéticos
  - Redes de sensores/coordenação de entidades autônomas
  - Tecnologias da Informação – Pesquisa/Filtragem/Classificação
  - Jogos Inteligentes, ...

13

Resolução de Problemas e Busca

Capítulo 2

14

Construção de um Sistema para Solução de um Problema

- Definir o problema precisamente
- Determinar o tipo de conhecimento necessário para resolver o problema.
- Criar um esquema da representação para tal conhecimento
- Escolher (ou desenvolver) técnicas de solução adequadas para solução do problema.

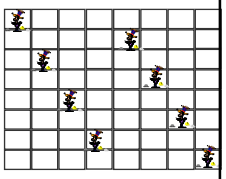
15

Definição do Problema

- Considerações:
  - Representação Computacional do problema
  - Objetivo (o que se pretende alcançar)
  - Onde Iniciar
  - Como modificar os estados
  - Como identificar modificações úteis na solução do problema

16

Problemas Difíceis



- Entendimento de Inglês.
- Jogar Xadrez.
- Resolver Integrais Indefinidas.
- Prever o clima.
- Prever mudanças no estoque de uma loja
- Organizar uma linha de produção
- Acomodar objetos dentro de um espaço físico limitado

17

Generalização do Problema

- É útil desenvolver métodos gerais para solução de problemas.
- Dessa forma torna-se fácil comparar problemas (especialmente o tamanho).
- Pode ser desenvolvido um conjunto de estratégias para solução de problemas e qualquer uma pode ser aplicada.
- É necessário, porém, escolher a metodologia mais adequada.

18

Definição do Problema como um Espaço de Estados (Cont.)

- Uma possível estratégia para solução de problemas é listar todos os estados possíveis.
- A solução do problema consiste em percorrer o espaço de estados a partir do estado inicial até o estado meta.
- É necessário desenvolver um conjunto de operadores que modifique um estado para um outro estado.

19

Definição do Problema como um Espaço de Estados (Cont.)

- Conjunto de Estados possíveis.
- Conjunto de operações possíveis que modifiquem um estado.
- Especificação de um estado *inicial(s)*.
- Especificação de um estado *meta(s)*.

20

Espaço de Estados

- Geralmente não é possível listar todos os espaços possíveis:
  - utilização de abstrações para descrever estados válidos.
  - pode ser mais fácil descrever estados inválidos.
  - algumas vezes é útil fornecer uma descrição geral do espaço de estados e um conjunto de restrições.

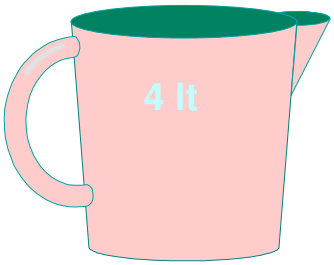
21

Operações

- O sistema de solução se move de um estado para outro de acordo com operações bem definidas.
- Geralmente estas operações são descritas como *regras*.
- Um sistema de controle decide quais regras são aplicáveis em um dado estado e resolve conflitos e/ou ambiguidades.

22

Exemplo: Problema dos Jarros de Água



Sem limite de Água

Objetivo: 2 litros no jarro 4 lt

23

Espaço de Estados do Problema dos Jarros de Água

O espaço de estados pode ser representado por dois inteiros x e y:

x = litros no jarro de 4 litros

y = litros no jarro de 3 litros

Espaço de Estados = (x,y)

tal que $x \in \{0,1,2,3,4\}$, $y \in \{0,1,2,3\}$

24

Início dos Jarros de Água e Estados Meta

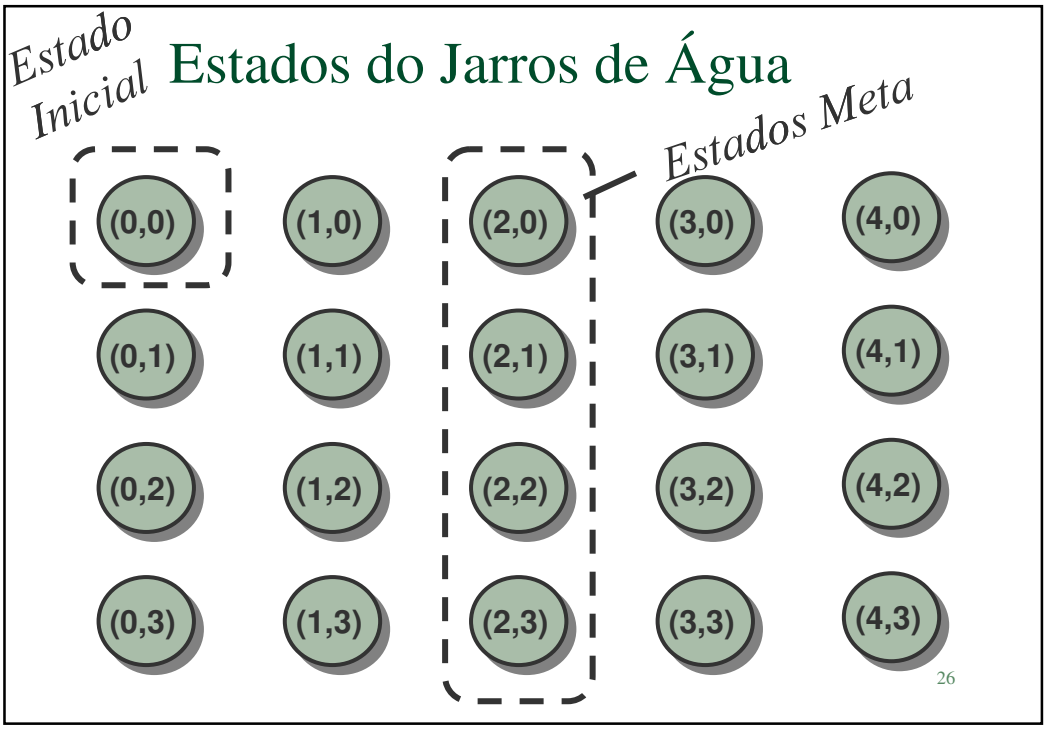
- O Estado Inicial ocorre quando ambos os jarros estão vazios:

  (0,0)

- O Estado Meta é qualquer estado que possua 2 litros de água no jarro de 4 litros:

  (2,n) para qualquer n

25



26

Operações com Jarros de Água

- Colocar 3 lt. no jarro 3
- Colocar 4 lt. no jarro 4
- Esvaziar jarro 3
- Esvaziar jarro 4
- Coloca o conteúdo do jarro 3 no jarro 4
- Outros ???

(x,y) -> (x,3)
(x,y) -> (4,y)
(x,y) -> (x,0)
(x,y) -> (0,y)
(0,y) -> (y,0)

27

Restrições

- Não é possível colocar água em um jarro cheio.
- Restrições são associadas para que uma operação possa ser aplicada sobre um estado

28

Regras de Produção

- Uma operação e as condições que devem ser satisfeitas (restrições) antes da operação poder ser aplicada é chamada de *regra*.
- Tipicamente é necessário mesclar regras gerais e regras específicas.
- Informação prévia sobre a solução tende a produzir regras específicas e aumentar a velocidade da busca.

29

Domínio de Regras Específicas

- Para o problema dos Jarros de Água :
  - (0,2) -> (2,0)
  - (x,2) -> (0,2)

- Utilizando estas regras soluções podem ser encontradas rapidamente!

30

Sistema de Produção

- Um *Sistema de Produção* é formado por:
  - Um conjunto de regras.
  - Algum mecanismo para representar o estado do sistema (uma base de dados).
  - Uma estratégia de controle que controla a ordem na qual regras da base de dados são comparadas e a solução de conflitos.
  - Um mecanismo que aplica as regras.

31

Estratégias de Controle

- Pré-requisitos Gerais:
  - Movimentação:
    - aplicação da mesma regra sucessivamente não é provavelmente útil.
    - É necessário mudar um estado para fazer progresso.
  - Sistemática
    - opera à medida que explora novas regiões no espaço de estados.
    - Evita busca circular (loops).

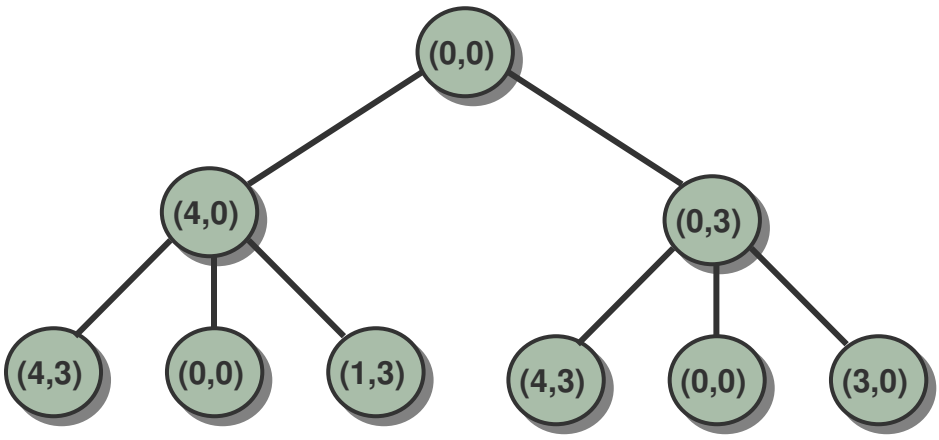
32

Estratégias de Controle para os Jarros de Água

- Lista ordenada de regras - aplica a primeira regra adequada para determinada situação.
- Escolhe qualquer regra (randomicamente) adequada para determinada situação.
- Aplica todas as regras adequadas, e armazena o caminho resultante de todos os estados obtidos. No próximo passo faz o mesmo para todos os estados.

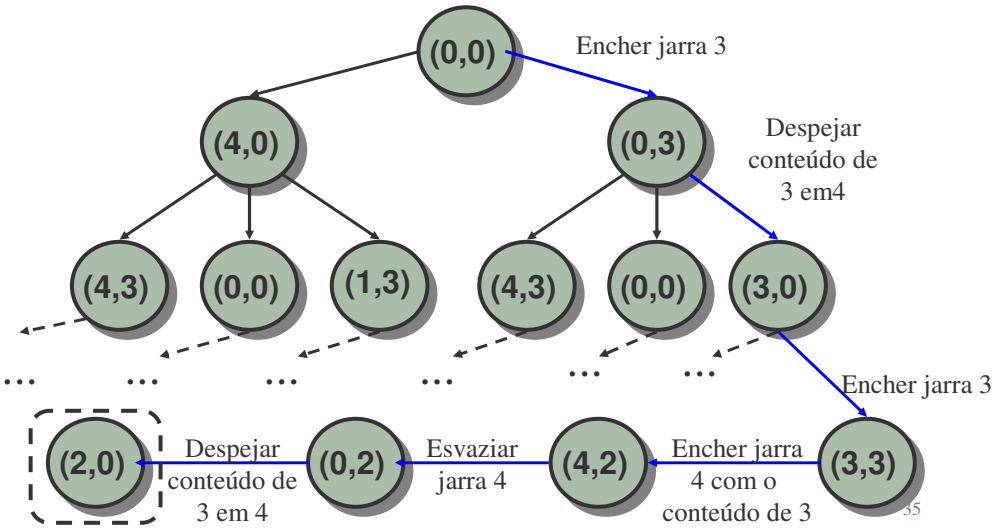
33

Árvore dos Jarros de Água



34

Árvore dos Jarros de Água – Exemplo de uma solução



35

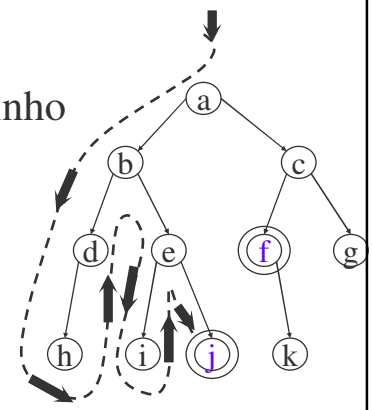
Busca em Profundidade

- Algoritmo (BP)
  - Estabeleça alguma ordenação às regras;
  - **Enquanto** existem regras aplicáveis
    - Aplique a próxima regra e gere um novo estado;
    - Faça uma busca em profundidade (BP) no novo estado.
  - **fim_enquanto**

36

Características da Busca em Profundidade

- Não necessita armazenar o caminho de uma grande lista de estados.
- Pode encontrar uma solução muito rapidamente.
- "Poda" é possível
  Exemplo: utilização de heurísticas
- Pode facilmente encontrar problemas com ciclos (loops).



37

Busca em Profundidade em Prolog

```
profundidade(A,[A]) :-  % se o estado é meta, então pára
    meta(A).
profundidade(A,[A|B]) :-  % senão continua a busca no novo estado
    novo_estado(A,C),
    profundidade(C,B).
```

* *meta(A)* verifica se *A* é uma solução

Obviamente, o algoritmo acima não implementa o tratamento de ciclos, porém, ele pode ser facilmente modificado, armazenando o caminho percorrido, de forma a tratar o problema adequadamente.

38

Busca em Profundidade - Exercício

Baseado no programa Prolog de BP apresentado anteriormente, crie uma outra versão que resolva o problema dos ciclos.

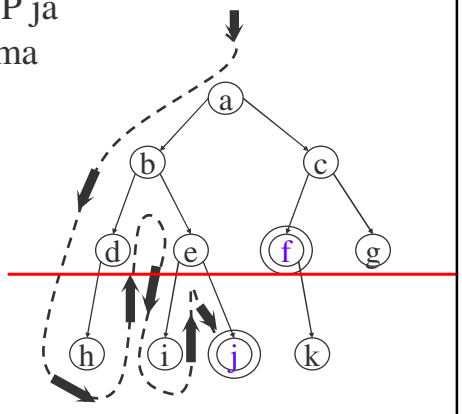
Solução:

```
profundidade(No,Solucao) :-
        profundidade(No,[],Solucao).
profundidade(No,_,[No]) :-
        meta(No).
profundidade(No,Caminho,[No|Solucao]) :-
        \+ pertence(No,Caminho),
        novo_estado(No,Novo),
        profundidade(Novo,[No|Caminho],Solucao).
```

39

Busca em Profundidade - Exercício

Considerando os programas de BP já apresentados, escreva um programa Prolog que realize busca em profundidade com limitação de profundidade (fornecida como parâmetro)



40

Busca em Profundidade - Resolução do Exercício Anterior

```
profundidade(No,Solucao,Prof) :-
    profundidade(No,[],Solucao,Prof).
profundidade(No,_,[No],_) :-
          meta(No).
profundidade(No,Caminho,[No|Solucao],Prof) :-
     Prof  > 0,
     Prof1 is Prof -1,
    \+ pertence(No,Caminho),
    novo_estado(No,Novo),
     profundidade(Novo,[No|Caminho],Solucao,Prof1).
```

41

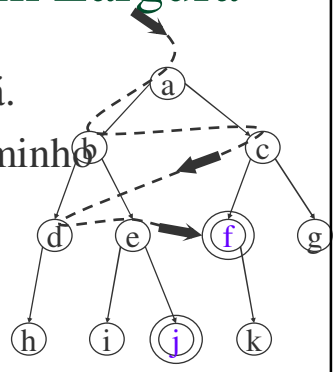
Busca em Largura

- A estratégia de árvore para o problema dos jarros de água é um exemplo de *busca em largura.*
- Algoritmo geral para BL:
- crie *lista_nós* e a inicialize com o estado inicial;
- **enquanto** um estado meta não é encontrado **ou** *lista_nós* != { };
  - remova o primeiro elemento de *lista_nós, primeiro_nó*;
  - aplique todas as regras possíveis em *primeiro_nó* e adicione os estados resultantes em *lista_nós*;
- **fim_enquanto**

42

Características da Busca em Largura

- Se há uma solução, BL a encontrará.
- Encontrará a solução mínima (o caminho mais curto até a solução).
- Não terá problemas com ciclos.
- Requer espaço disponível para armazenar ***lista_nós,*** que pode ser muito grande!!!



43

Busca em Largura - Exercício

Escreva um programa Prolog que realize busca em largura num grafo semelhante à figura do grafo anterior

Exemplo:

?- largura([a],[f,j],Cam).

Cam = [a,b,c,d,e,f]

yes

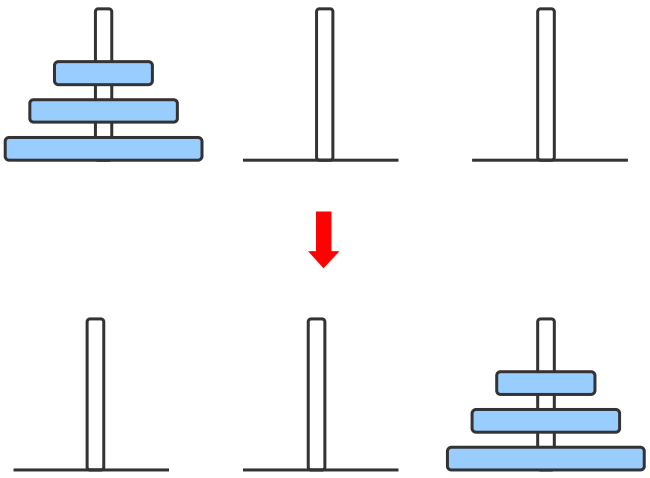
44

Busca em Largura - Resposta ao Exercício Anterior

```
largura([No|_],L,[No]) :-
   pertence(No,L),!.
largura([No|Lista],N,[No|L]) :-
   filhos_sem_repeticoes(No,Lista,Filhos),
   concatena(Lista,Filhos,Nos),
   largura(Nos,N,L).
```

* *filhos_sem_repeticoes(No,Lista,F)* retorna em *F* os filhos de *No* que não pertencem a lista *Lista*

Outro problema: As torres de Hanói

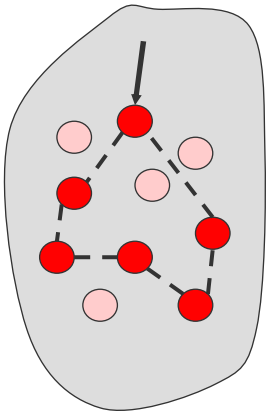


Exercício

- 1) Construa a árvore de busca para o problema das torres de Hanói
- 2) Responda a questão: Qual método encontrará a solução mais rapidamente (largura ou profundidade)? Demonstre usando a árvore da questão anterior.

47

Outro Problema



- Problema do Caixeiro Viajante
  - Lista de cidades para visitar
  - Lista de distâncias entre cada cidade
  - Visite cada cidade apenas uma vez
  - Encontre o menor trajeto
- Descreva estratégias de busca BL and BP que podem ser utilizadas neste problema

48

Busca Heurística

Busca Heurística

- Muitos problemas possuem espaços de busca que são muito grandes para serem examinados completamente.
- É possível construir estratégias que não prometem a melhor solução, mas que encontram uma “boa” resposta rapidamente.