

Inteligência Artificial

Joaquim Gonçalves

1

“Os que suficientemente loucos para pensar que podem mudar o mundo são os que o fazem”

Steve Jobs

2

Nota

Alguns slides que aqui se apresentam são adaptados dos apontamentos do Prof. Luís Paulo Reis (U. Minho) e do livro, Artificial Intelligence: A Modern Approach de Russell and Norvig.

Alguns slides são originais.

3

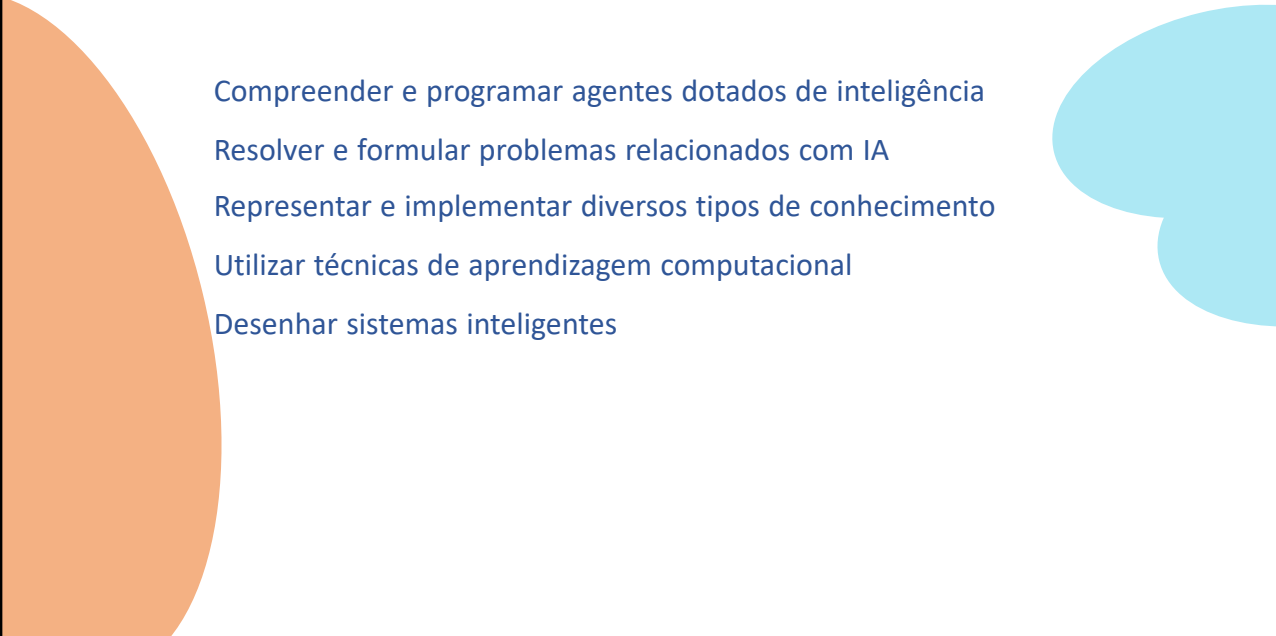
Objectivos

Esta unidade curricular pretende fornecer uma introdução técnica alargada e uma revisão dos principais conceitos de inteligência artificial

Visa dotar os alunos de competências básicas de pensamento abstrato e resolução de problemas teóricos complexos.

4

Competências



Compreender e programar agentes dotados de inteligência
Resolver e formular problemas relacionados com IA
Representar e implementar diversos tipos de conhecimento
Utilizar técnicas de aprendizagem computacional
Desenhar sistemas inteligentes

5

Programa previsto

01

Introdução e motivação do tema da IA

02

Noções básicas de IA

03

Pesquisa de Soluções

Informada
Não informada
Com adversário

04

Aprendizagem computacional

Não supervisionada
Por reforço
Supervisionada

6

Pré Requisitos

Programação

Desenvolvimento de algoritmos, linguagens de programação.

Matemática

Lógica, Cálculo, Teoria de Conjuntos, Probabilidade e Estatística.

Estruturas de dados

Listas, Árvores e Grafos

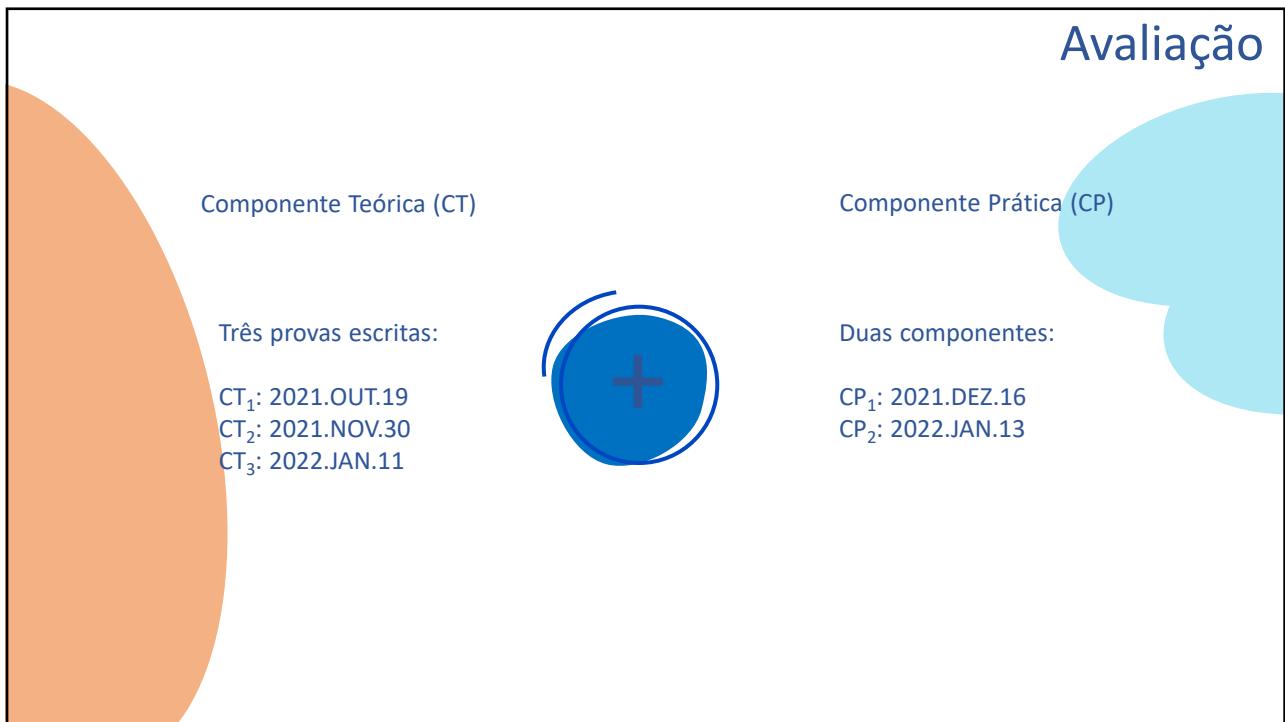
7

Bibliografia

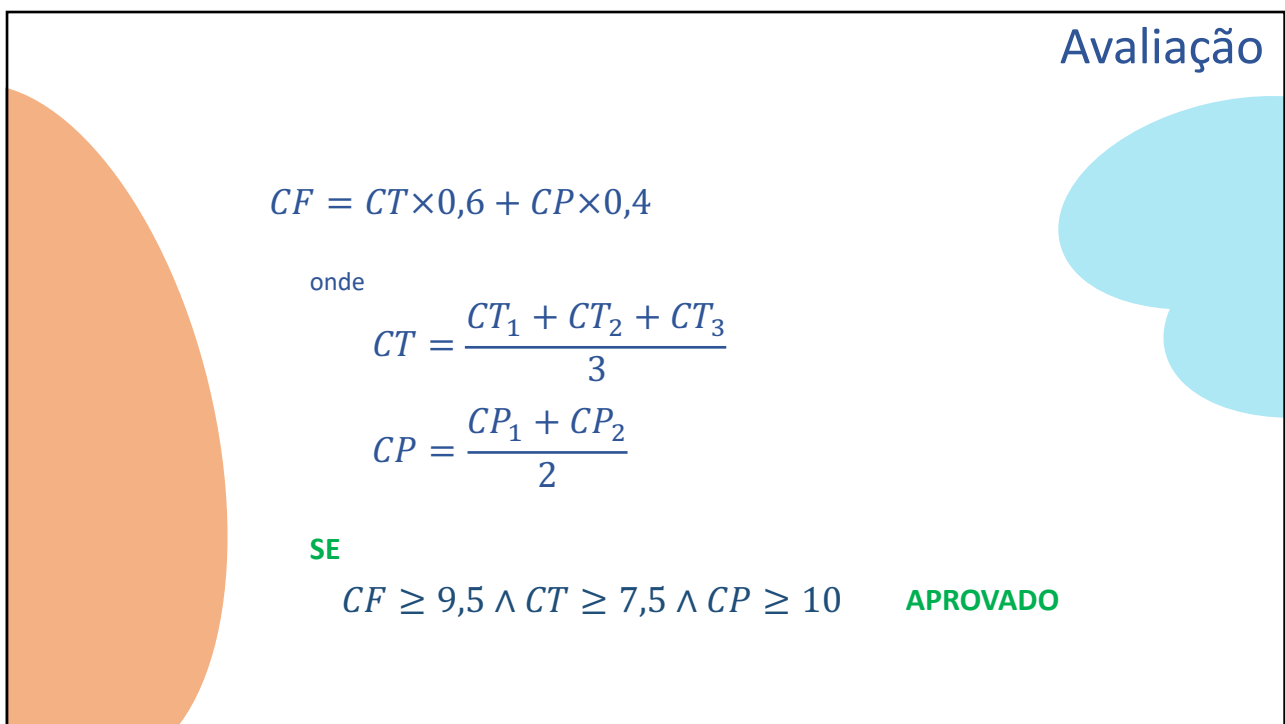
Russell and P. Norvig, Artificial Intelligence: A Modern Approach
Third Edition, Pearson Education 2014,
ISBN 10: 1-292-02420-8
ISBN 13: 978-1-292-02420-2

Luiz Eduardo Borges. Python para Desenvolvedores.
2ª edição. Edição de autor (disponível em <http://ark4n.wordpress.com/python/>)

8



9



10



Gabinete nº 6

jgoncalves @ ipca.pt

11



Introdução

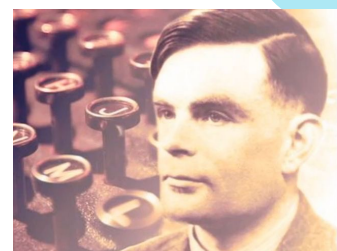
12

História...

Surge na década de 50 com o objectivo de desenvolver sistemas que resolvam problemas sem solução algorítmica tradicional

Alan Turing, matemático britânico, propõe em 1950 um teste (teste de Turing) para avaliar a inteligência da máquina

O teste tem como base a impossibilidade de distinguir duas entidades inteligentes (seres humanos), perante uma determinada questão. Se não for possível distinguir a proveniência das respostas quando responde um ser humano e um computador, então este é dotado de inteligência

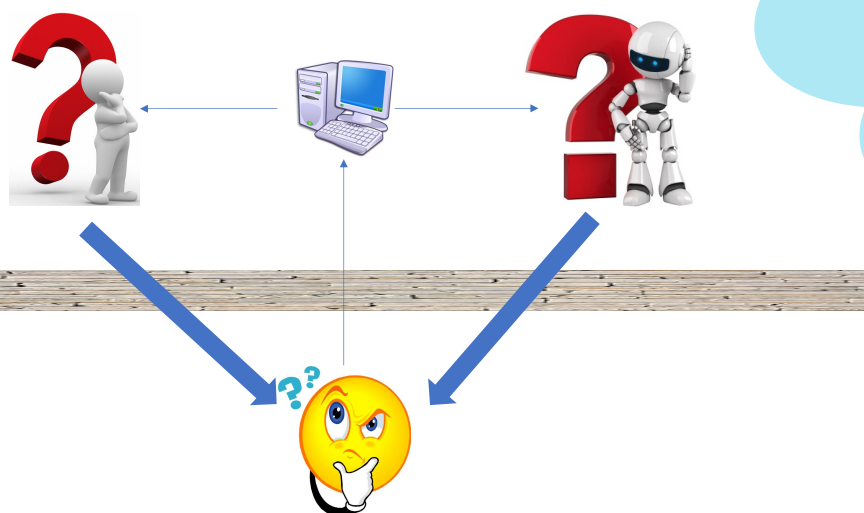


Alan Turing

13

História...

Teste de Turing



14

Aplicações

- Automação de sistemas complexos
- Sistemas de controlo
- Previsão
- Marketing
- Jogos
- Redes distribuídas
- ...

15

O que é a Inteligência?

- Capacidade para utilizar o raciocínio e a lógica?
- Capacidade de escrever e falar de forma expedita?
- Capacidade para resolver os problemas colocados?
- Entender o motivo pelo qual se cometeu um erro?
- Compreender a sociedade e agir de acordo com as expectativas?

16

Inteligência

Capacidade mental que envolve a habilidade de raciocinar, planejar, resolver problemas, pensar de forma abstrata, compreender ideias complexas, e aprender com a experiência.



Capacidade para se adaptar, moldar ou seleccionar o ambiente

Capacidade para julgar e compreender

Capacidade para compreender e lidar com pessoas, objectos e símbolos

Capacidade para agir intencionalmente e pensar racionalmente em função do ambiente

17

Três aspectos da Inteligência

Analítica

Resolução de problemas teóricos, geralmente com recurso à abstracção

Creativa

Introdução de novos conceitos e ideias ou perspectivas sobre um determinado problema

Prática

Seleccionar as melhores soluções para os problemas do quotidiano

18

Algumas definições

É o estudo das ideias que, implementadas no computador, lhes permitam realizar os mesmos objetivos que fazem as pessoas parecer inteligentes.

Mais especificamente a IA tenta que os computadores sejam mais úteis e ao mesmo tempo estuda os princípios que tornam a inteligência possível.

Patrick Winston, ex-director do Lab. de IA do M.I.T.

Ambiguidade: a definição contém o definido

Redundância: "computadores mais úteis"

É o estudo dos processos que possibilitam aos computadores realizar tarefas para as quais, no momento, as pessoas são mais aptas.

E. Rich , Universidade do Texas

Definição vaga e incompleta.

19

Inteligência vs Inteligência artificial

Inteligência

- Capacidade de aprender e perceber como lidar com novas situações
- Utilização habilidosa do conhecimento.

Inteligência Artificial

- Ciência para construir máquinas para fazer coisas que requerem inteligência quando feitas pelo homem.
- Estudo das ideias que possibilitam aos computadores serem inteligentes.
- Estudo das computações que tornam possível a percepção, raciocínio e acção.

20

Inteligência Artificial

Como tornar Sistemas Computacionais mais “inteligentes” e mais úteis usando a racionalidade?

SE

existem processos simbólicos identificáveis na base do raciocínio

ENTÃO

esses processos podem ser estudados e simulados, sendo o Hardware (pessoa ou computador) um DETALHE de Implementação

21

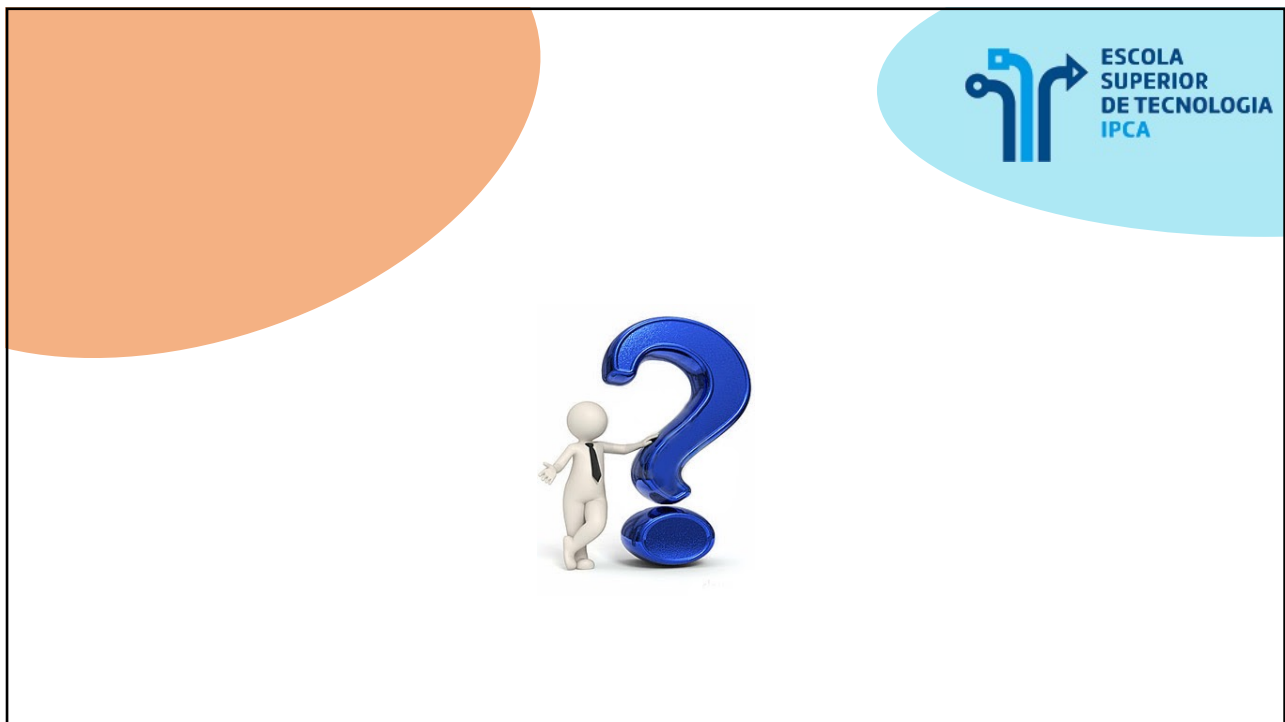
Comportamento inteligente

- Até que ponto a inteligência é aprendida?
- Como ocorre a aprendizagem?
- O que é a criatividade?
- O que é a intuição?
- A inteligência é observável a partir do comportamento?
- Como é o conhecimento representado nos neurónios?
- O que é auto-consciência? Que papel têm na inteligência?

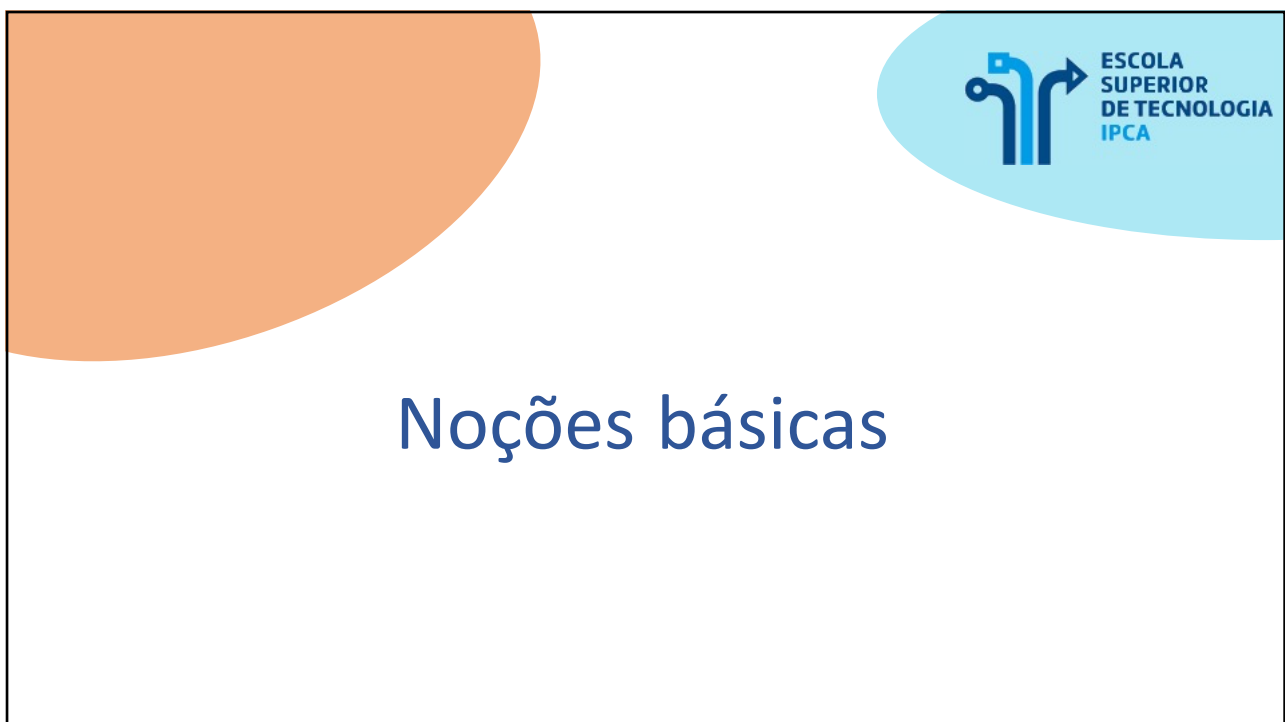
É possível obter inteligência num computador?

Necessitamos de um mecanismo biológico?

22



23



24

Abordagens

Devemos modelar o pensamento ou o comportamento?

Devemos modelar o ser humano ou um ser ideal?

Os sistemas deverão

- Pensar como humanos
- Agir como humanos
- Pensar racionalmente
- Agir racionalmente



25

Agentes e Sistemas multiagentes

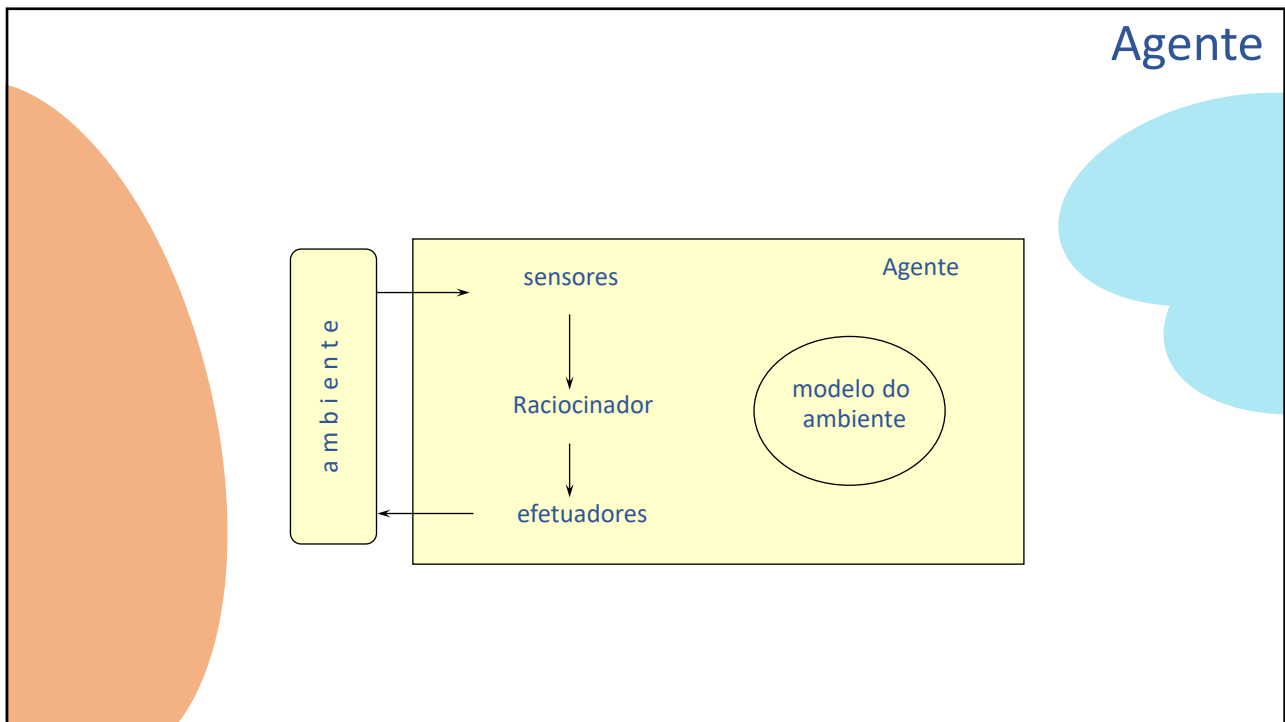
Agente é qualquer entidade que:

- percebe seu ambiente através de sensores (ex. câmeras, microfone, teclado, mensagens de outros agentes,...)
- age autonomamente sobre ele através de actuadores (ex. vídeo, auto-falante, impressora, braços, ftp, mensagens para outros agentes,...)

Sistema multiagente contém dois ou mais agentes que interagem para trabalharem de forma cooperativa

Deve estar associada uma medida de desempenho que é o critério que define o grau de sucesso das ações

26



27

Exemplos de Agentes

- Um agente humano
 - Sensores: olhos, ouvidos, ...
 - Actuadores: mãos, pernas, boca, ...
- Um agente robótico
 - Sensores: câmeras, detectores da faixa de infravermelho, ...
 - Actuadores: motores, ...
- Um agente de software
 - Sensores: teclas digitadas, conteúdo de arquivos, pacotes de redes, ...
 - Actuadores: exibição de algo na tela, gravação de arquivos, envio de pacotes de rede, ...

28

Agentes

Exemplos

Agente	Dados Perceptivos	Acções	Objectivos	Ambiente
Diagnóstico Médico	Sintomas, Exames, Respostas	Perguntar, Prescrever exames	Saúde do doente, Minimizar custos	Doente, Consultório
Filtrador de emails	Mensagens	Aceitar ou rejeitar mensagens	Diminuir consumo de tempo ao utilizador	Utilizadores, Mensagens
Motorista	Imagem, Velocímetro, Sons	Andar, Parar, Ultrapassar	Segurança, Rapidez, Conforto	Ruas, Automóveis, Peões, sinais de trânsito

29

Agentes Inteligentes - Agentes Racionais

Agente Racional é aquele que faz a ação correta!

Qual a ação correta?

❖ Aquela que o faz ser mais bem sucedido!

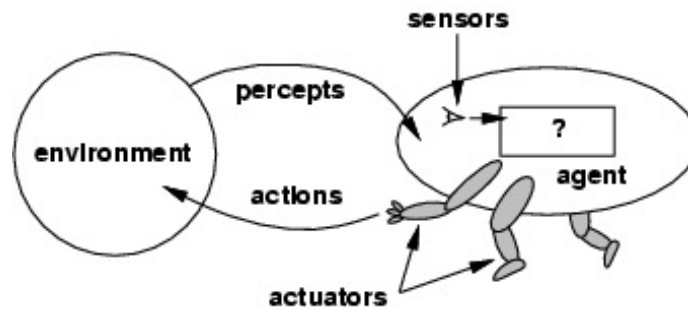
Como e quando avaliar esse sucesso? (medida do sucesso)

Agente Racional Ideal:

- Para cada sequencia de perceções, faz a acção que é esperada.
- Maximiza a sua medida de desempenho com base no conhecimento que tem.

30

Agente Inteligente



A função agente mapeia a história das percepções para as ações:

$$[f: P^* \rightarrow A]$$

Programa executa a arquitetura física para produzir f

31

Estrutura dos Agentes Inteligentes

Agente exibe um comportamento

acção que é executada depois de uma dada sequência de percepções!

Tarefa da IA:

Projectar o programa e a arquitectura para o Agente



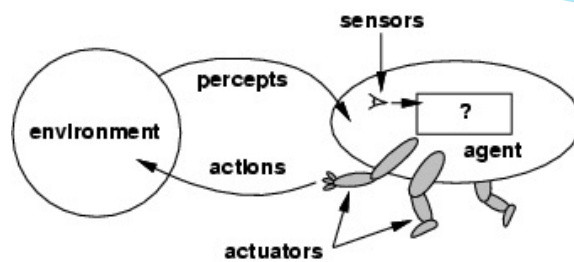
32

Programa de um Agente / Tipos de Agentes

Estruturas de Dados internas são atualizadas usando percepções e usadas para tomar a decisão das ações a executar (melhor ação)

Tipos de Agentes (Russel e Norvig):

- Agentes reflexos simples
- Agentes com representação do mundo
- Agentes baseados em objetivos
- Agentes baseados em utilidade
- Agentes com Aprendizagem

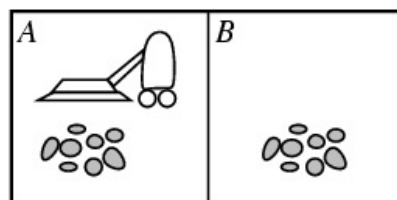


33

Agentes

Exemplo: O Mundo do Aspirador

- Percepções: local e conteúdo
 - Exemplo: [A, sujo]
- Ações: Esquerda, Direita, Aspirar, Não operar



34

Estrutura dos Agentes

Para o desenvolvimento de um agente inteligente devem ser considerados 4 aspectos:

- Performance Measure (medida de desempenho)
- Environment (ambiente)
- Actuators (actuadores)
- Sensors (sensores)

PEAS

35

PEAS

Exemplo: Condutor de Taxi

Medida de desempenho:

Viagem sem violações às leis de trânsito, segura, rápida, confortável para os passageiros, maximizar lucros, ...

Ambiente:

Ruas, estradas, outros veículos, peões, clientes, ...

Actuadores:

Direção, acelerador, travão, embraiagem, caixa de velocidades, buzina, ...

Sensores:

Câmara, sonares, localizador laser, velocímetro, GPS, odómetro, sensores do motor, ...

36

PEAS

Exemplo: Sistema de Diagnóstico Médico

Medida de desempenho:
Nível de recuperação do doente, qualidade de vida do doente, minimizar custos, ...

Ambiente:
Doente, hospital, equipa médica, ...

Actuadores:
Écran para exibir perguntas, testes, diagnósticos, tratamentos, análises, ...

Sensores:
Entrada pelo teclado de sintomas, resultados, respostas do doente, ...

37

PEAS

Exemplo: Robot de selecção de peças

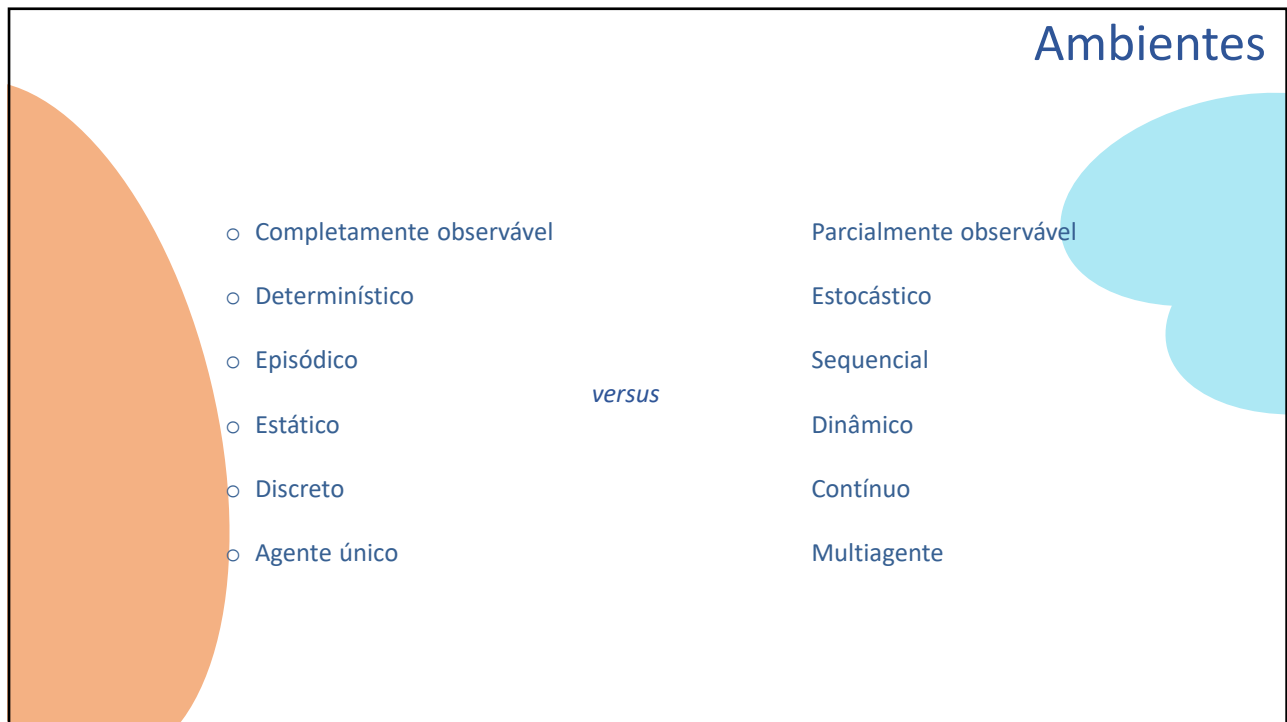
Medida de desempenho:
Proporção de peças bem seleccionadas, ...

Ambiente:
Tapete transportador de peças, bandejas para colocar peças seleccionadas, ...

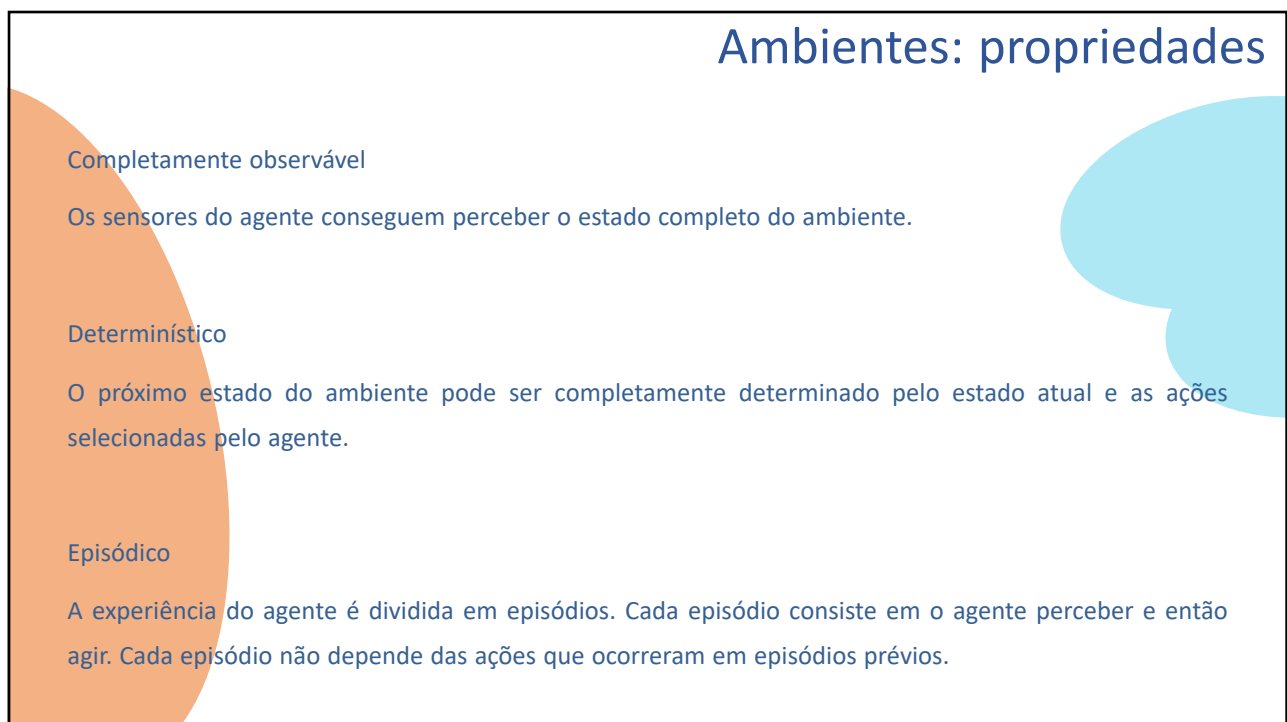
Actuadores:
Braço e mão robóticos (motores), ...

Sensores:
Camara, sensores de movimento das juntas, sensores de proximidade, ...

38



39



40

Ambientes: propriedades

Estático

Ambiente não muda enquanto o agente está a seleccionar a acção a realizar.

Semi-dinâmico

O ambiente não muda enquanto o agente decide, mas o "score" do agente muda.

Discreto

Existe um número distinto e claramente definido de percepções e ações em cada turno.

Contínuo

Percepções e acções mudam num espaço contínuo de valores.

41

Propriedades dos Ambientes

Ambiente de tarefa	Observável	Determinístico	Episódico	Estático	Discreto	Agentes
Jogo de palavras cruzadas	Completamente	Determinístico	Seqüencial	Estático	Discreto	Único
Xadrez com um relógio	Completamente	Estratégico	Seqüencial	Semi	Discreto	Multi
Pôquer	Parcialmente	Estratégico	Seqüencial	Estático	Discreto	Multi
Gamão	Completamente	Estocástico	Seqüencial	Estático	Discreto	Multi
Direção de táxi	Parcialmente	Estocástico	Seqüencial	Dinâmico	Contínuo	Multi
Diagnóstico médico	Parcialmente	Estocástico	Seqüencial	Dinâmico	Contínuo	Único
Análise de imagens	Completamente	Determinístico	Episódico	Semi	Contínuo	Único
Robô de seleção de peças	Parcialmente	Estocástico	Episódico	Dinâmico	Contínuo	Único
Controlador de refinaria	Parcialmente	Estocástico	Seqüencial	Dinâmico	Contínuo	Único
Instrutor interativo de inglês	Parcialmente	Estocástico	Seqüencial	Dinâmico	Discreto	Multi

O tipo de ambiente e a respectiva tarefa determina a base da construção do agente

Mundo: Parcialmente observável, estocástico, sequencial, dinâmico, contínuo e multi-agente

42

Tipos Básicos de Agentes

Reativo Simples

Reativo Baseado em modelos

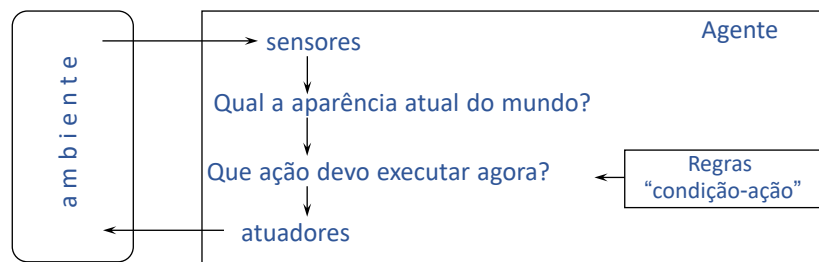
Baseados em Objetivos

Baseado na Utilidade

Com Aprendizagem

43

Agente Reativo Simples



Regras condição-acção (regras se-então) fazem uma ligação directa entre a percepção actual e a acção

O agente funciona apenas se o ambiente for completamente observável e a decisão correcta puder ser tomada com base apenas na percepção actual

44

Agente Reativo Simples

Seleciona ações com base na percepção actual, ignorando o restante do histórico de percepções.

Vantagens e Desvantagens

Regras condição-ação: representação inteligível, modular e eficiente

ex. Se velocidade > 60 então multar

Não pode armazenar uma sequência de percepções, pouca autonomia

Funcionará somente se a decisão correta puder ser tomada com base apenas na percepção actual.

Ambiente completamente observável

45

Agente Reativo Simples

Exemplo

Agente aspirador de pó, porque sua decisão baseia-se apenas na posição actual e no facto dessa posição conter ou não lixo.

Função AGENTE-ASPIRADOR-REATIVO(posição,estado) retorna uma ação

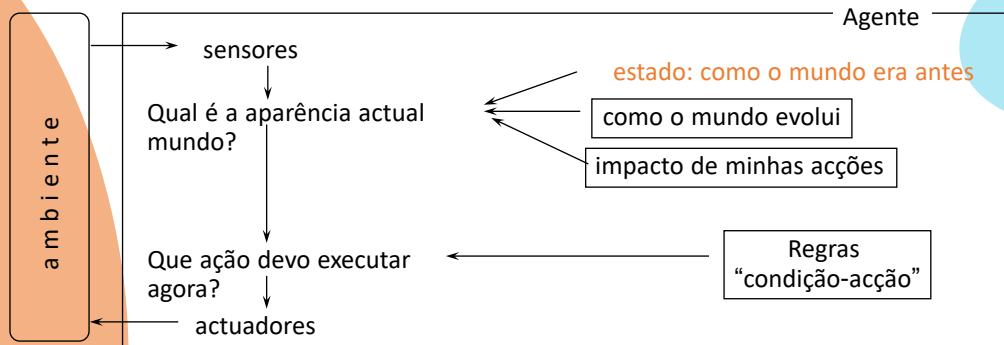
se estado = Sujo então retorna Aspirar

senão se posição = A então retorna Direita

senão se posição = B então retorna Esquerda

46

Agente Reativo Baseado em Modelos



Mantêm um estado interno - representação do mundo

47

Agente Reativo Baseado em Modelos

O agente deve manter um estado interno que dependa do histórico de percepções e reflita os aspectos não observados no estado actual

Dois tipos de conhecimento são necessários para actualizar o estado interno do agente (modelo do mundo):

- Como o ambiente evolui independente do agente

Um carro que está a ultrapassar, em geral estará mais perto do que estava um instante anterior

- Como as ações do próprio agente afectam o mundo

Se o agente virar o volante à direita, o carro irá virar para a direita

48

Agente Reativo Baseado em Modelos

Um agente que utiliza o modelo de mundo

Desvantagem: pouca autonomia

não tem objetivo, não encadeia regras

Ambientes: determinístico e pouco complexo

49

Agente Reactivo baseado em Modelos

Função AGENTE-REATIVO-BAS-MOD(percepção) retorna uma ação

Variáveis estáticas:

estado, uma descrição do estado atual do mundo

regras, um conjunto de regras condição-ação

ação, a ação mais recente, inicialmente vazio

estado \leftarrow ACTUALIZA-ESTADO(estado, ação, percepção)

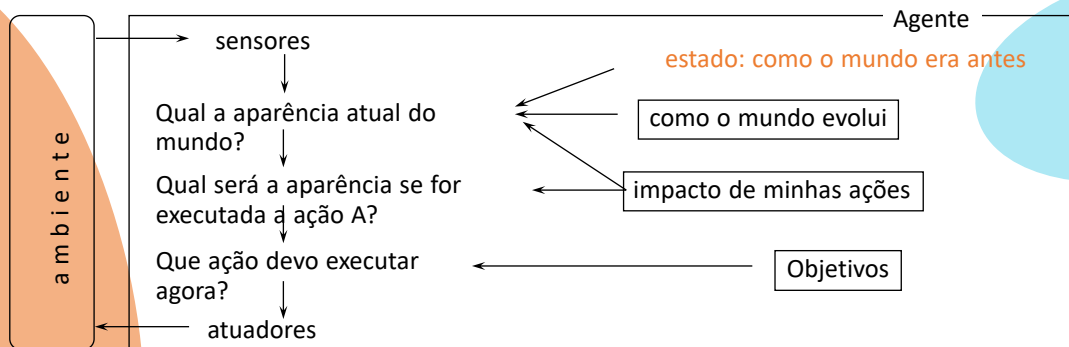
regra \leftarrow DETERMINA_REGRA(estado, regras)

ação \leftarrow ACÇÃO-DA-REGRA(Regra)

retornar ação

50

Agente Baseado em Objectivos



Descrição do estado do mundo e do objectivo a atingir

Exemplo: Chegar a Lisboa

Resolução de problemas por Pesquisa, Planeamento

51

Agente Baseado em Objectivos

Agente combina seu objetivo com as informações sobre os resultados de acções possíveis a fim de escolher acções que alcancem os seus objectivos

Exemplo

Táxi num entroncamento de estradas: virar à esquerda, à direita ou ir em frente?

Necessidade de busca e planeamento: áreas da IA dedicadas a encontrar sequências de acções que alcançam os objectivos do agente.

52

Agente Baseado em Objectivos

O agente precisa de algum tipo de informação sobre o seu objectivo

Objectivos descrevem situações desejáveis. Ex: estar no destino

Combinando informações sobre o objectivo do agente e os resultados de suas acções, o agente pode escolher acções que permitam alcançar o objectivo

A selecção da acção baseada em objectivo pode ser:

Directa: quando o resultado de uma única acção atinge o objetivo

Mais complexa: quando é necessário longas sequências de acções para atingir o objectivo

53

Agente Baseado em Objectivos

Para encontrar sequências de acções que alcançam os objectivos utilizam-se algoritmos de Busca e Planeamento

A tomada de decisão envolve a consideração do futuro (distinta das regras de condição-acção)

“O que acontecerá se eu fizer isso ou aquilo?”

“O quanto isso me ajudará a atingir o objetivo?”

Agentes reativos: reacção -> travar quando carro da frente travar

Agentes baseado em objetivo: raciocínio -> carro da frente trava -> carro da frente diminui velocidade -> objetivo: não atingir outros carros -> acção para atingir objetivo: travar

54

Agente Baseado em Objetivo

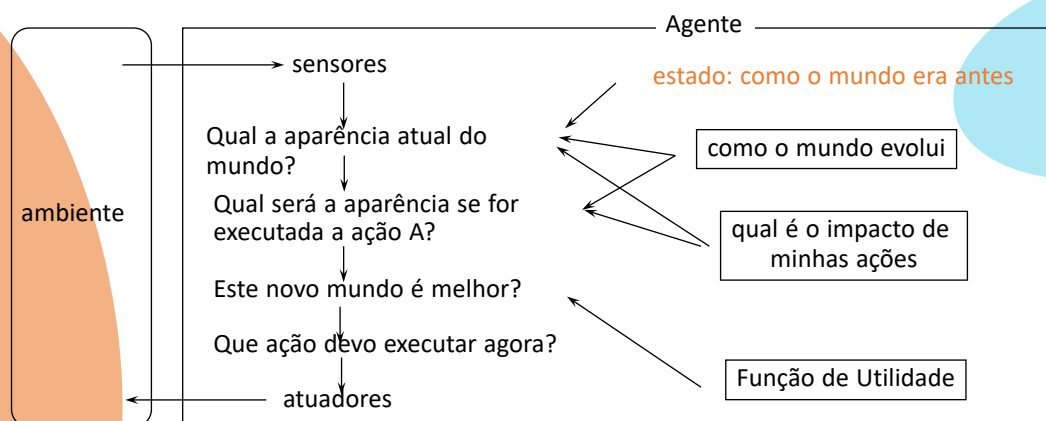
Vantagens e desvantagens:

Mais complicado e “ineficiente”, porém mais flexível, autónomo
Não trata objetivos que criem conflito

Ambientes: determinístico

55

Agente Baseado em Utilidade



Utilidade: Espécie de grau de felicidade do agente!

Mapeia o estado actual num valor!

56

Agente Baseado em Utilidade

Existem muitas sequências de ações que levam o agente ao seu objetivo. Algumas mais rápidas, mais seguras, mais económicas, etc.

Agentes baseados em utilidade utilizam uma medida de desempenho (função de utilidade) que permite uma comparação entre diferentes estados do mundo, permitindo selecionar a sequência de ações

Ambiente: sem restrição

Desvantagem: não tem adaptabilidade

57

Agente Baseado em Utilidade

Se um estado do mundo é mais desejável que outro, então ele terá maior utilidade para o agente

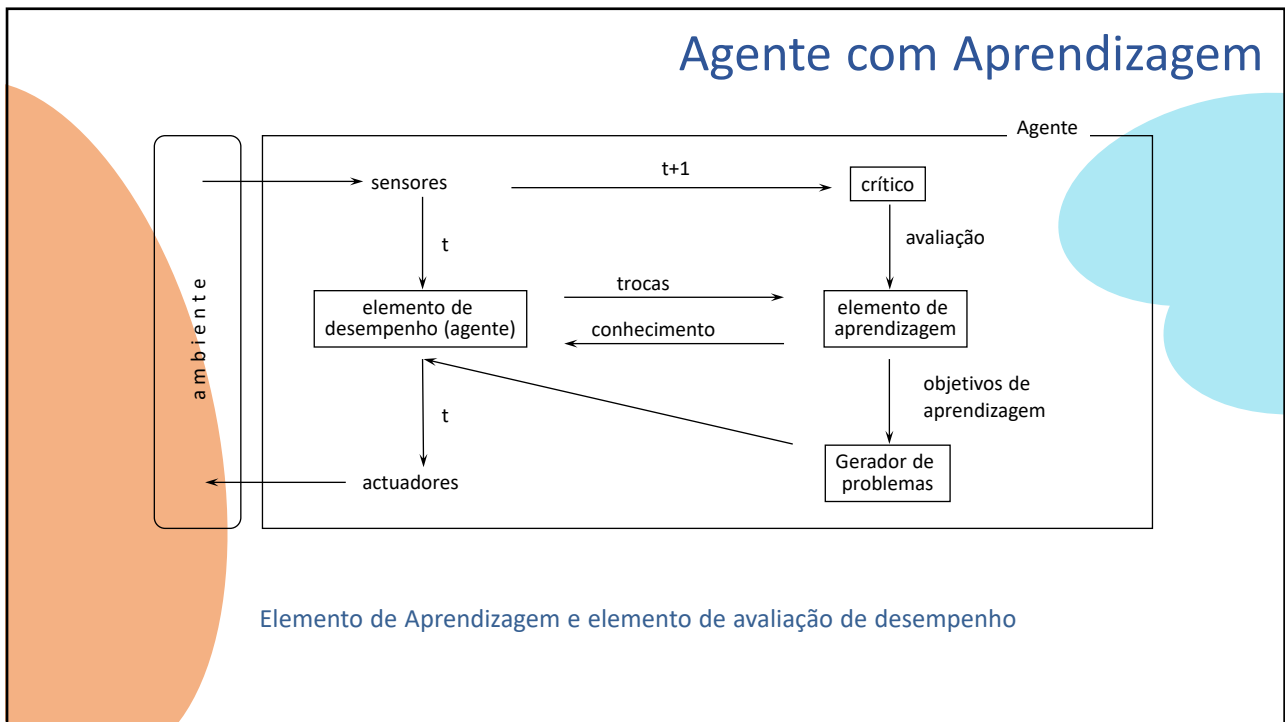
Utilidade é uma função que transforma um estado num número real que representa o grau de satisfação desse estado

Especificação completa da função de utilidade – decisões racionais em dois tipos de casos:

Quando existem objetivos conflituosos (velocidade x segurança) a função de utilidade especifica o compromisso apropriado

Quando existem vários objetivos que se deseja alcançar e nenhum deles pode ser atingido com certeza – ponderar a importância dos objetivos

58



59

Agente com Aprendizagem

Em agentes sem aprendizagem tudo o que o agente sabe foi determinado por quem o desenvolveu

Turing propõe construir máquinas com aprendizagem e depois ensiná-las

Aprendizagem também permite ao agente actuar em ambientes totalmente desconhecidos e tornar-se mais competente do que o seu conhecimento inicial poderia permitir

Um elemento de aprendizagem utiliza realimentação sobre como um agente está a funcionar e determina de que maneira o elemento de desempenho deve ser modificado para funcionar melhor, no futuro.

60

Exercício

TempControl: Agente para Controlar a Temperatura de uma Sala

Apresente um diagrama e o pseudo-código para um agente reactivo simples que permita controlar a temperatura de uma sala. Suponha que dispõe das percepções T1 e T2 correspondentes à temperatura da sala e à temperatura exterior e as ações AQ ligar o ar quente, NAQ – Desligar o ar quente, AC – ligar o ar frio, NAC – Desligar o ar frio, AJ – abrir as janelas, NAJ – fechar as janelas.

Pretende-se que a temperatura da sala esteja entre os 22 e os 24 graus. Sempre que seja possível usar as janelas para controlar a temperatura (não desperdiçando energia), tal deve ser efetuado. Sempre que a temperatura esteja mais de 2 graus afastada da banda desejada (inferior a 20 ou superior a 26 graus), devem ser fechadas as janelas e usar o aquecedor ou ar frio para repor a temperatura.

61

Tópicos de Resolução do exercício

Percepções:

T1 – Temperatura Interior
T2 – Temperatura Exterior

Ações:

AQ ligar o aquecedor
NAQ – Desligar o aquecedor
AC – ligar o ar frio
NAC – Desligar o ar frio
AJ – abrir as janelas
NAJ – fechar as janelas

Objetivo:

Manter a temperatura da sala entre os 22 e os 24 graus

Interpretação da Percepção:

M_QUENTE = $T1 > 26$
QUENTE = $T1 > 24$ e $T1 \leq 26$
NORMAL = $T1 \geq 22$ e $T1 \leq 24$
FRIO = $T1 \geq 20$ e $T1 < 22$
M_FRIO = $T1 < 20$
FORA_UTIL = $T2 < 24$ e QUENTE ou $T2 > 22$ e FRIO

Regras Condição-Ação:

Se NORMAL
Então NAQ; NAC; NAJ

Se (QUENTE ou FRIO) e FORA_UTIL
Então NAQ; AJ; NAC

Se QUENTE e não(FORA_UTIL) ou M_QUENTE
Então NAQ; NAJ; AC

Se FRIO e não(FORA_UTIL) ou M_FRIO
Então AQ; NAJ; NAC

62

Tópicos para resolução do exercício

Agente mais inteligente:

Câmaras para analisar quantas e quais as pessoas no interior da sala

Ajuste da temperatura em função dos gostos das pessoas

Utilização de previsões meteorológicas da Internet

...

63

Agentes: metodologia de desenvolvimento

Decompõe problema em: percepções, ações, objetivos e ambiente (e outros agentes)

Decompõe tipo de conhecimento em:

Quais são as propriedades relevantes do mundo?

Como o mundo evolui?

Como identificar os estados desejáveis do mundo?

Como interpretar suas percepções?

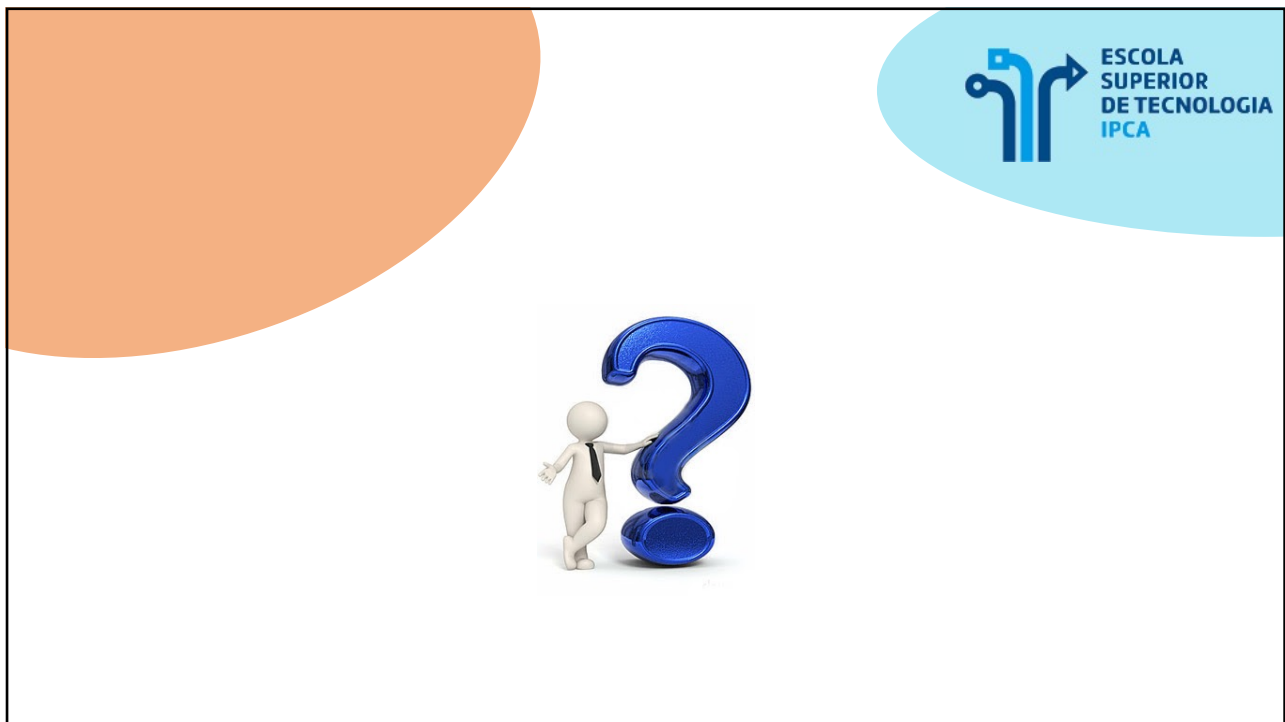
Quais as consequências de suas ações no mundo?

Como medir o sucesso de suas ações?

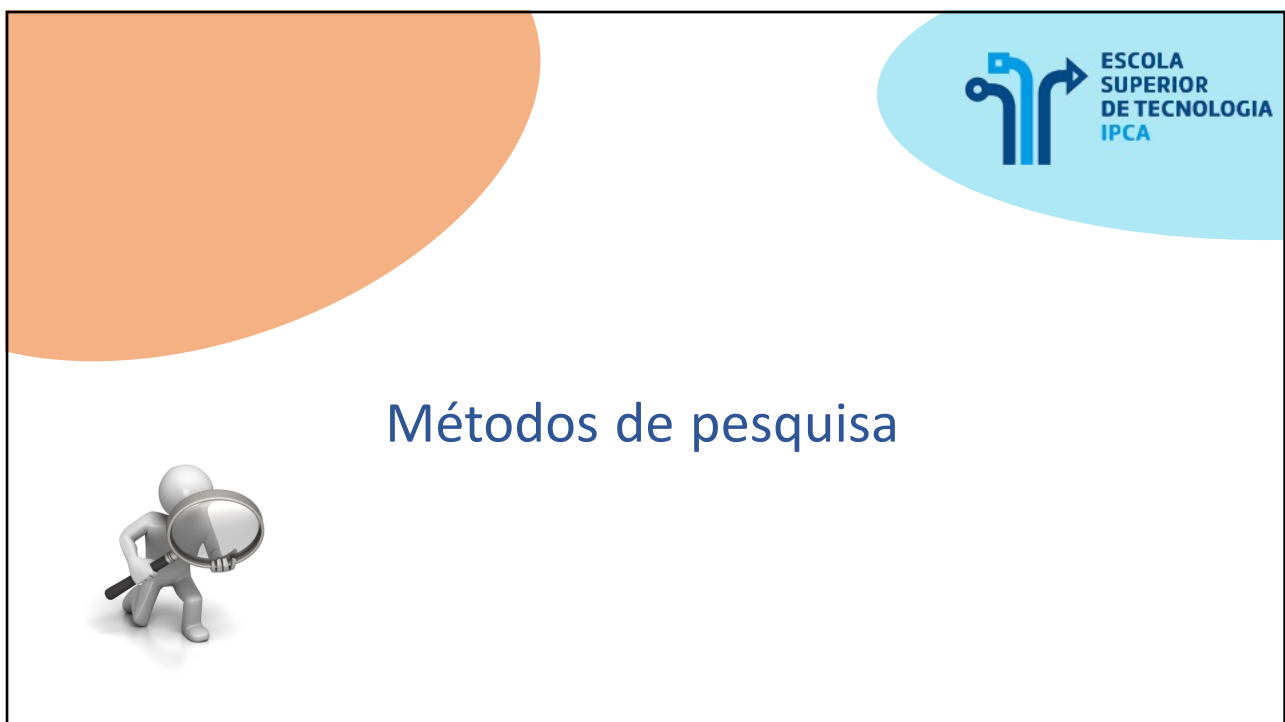
Como avaliar seus próprios conhecimentos?

Indica arquitetura e método de resolução de problema

64



65



66

Resolução de Problemas por Pesquisa

- Formulação de Problemas
- Pesquisa Não Informada
- Pesquisa Informada
- Algoritmos Iterativos
- Pesquisa com Adversários (Jogos)

67

Resolução de Problemas por Pesquisa

Como é que um agente pode agir, estabelecendo objetivos e considerando possíveis sequências de ações para atingir esses objetivos!

Resolução de Problemas

- Formulação de um problema como um problema de pesquisa
- Pesquisa não Informada (estratégias de pesquisa)
- Pesquisa Informada (pesquisa gulosa, algoritmo A*)
- Algoritmos de Melhoria Iterativa
- Jogos (em que é incluído um agente hostil!)

68

Agente para Resolução de Problemas

“Problem Solving Agent”

Procura encontrar a sequência de ações que leva a um estado desejável!

Formulação do Problema



Quais as ações possíveis? (qual o seu efeito sobre o estado do mundo?)
Quais os estados possíveis? (como representá-los?)
Como avaliar os Estados

Problema de pesquisa



Solução: sequência de ações

Execução

69

Problemas de Pesquisa

Muitos problemas em ciências da computação podem ser definidos como:

- Um conjunto S de ESTADOS (possivelmente infinito)
- Um estado INICIAL $s \in S$
- Uma relação de TRANSIÇÃO T ao longo deste espaço de estados
- Um conjunto de estados FINAIS (objetivos): $O \in S$

Problema pode ser

- representado por um GRAFO, onde os nodos representam estados e os arcos os pares da relação de transição
- resolvido através de pesquisa e um caminho entre o estado inicial e um estado objetivo

70

Agente para Resolver Problemas

Formulação do Problema:

Representação de um Estado

Definição do espaço de estados

Estado Inicial (Actual)

Estado(s) objectivo (final) – pode ser conhecido ou não

Teste Objectivo (avaliar se estamos num estado objectivo)

Operadores (Nome, Pré-Condições e Efeitos)

Custo da Solução

71

Formulação do Problema

Qual o conhecimento do agente sobre o estado do mundo e sobre as suas ações?

Quatro tipos de problemas distintos:

Problemas de estado único (ambiente determinístico e acessível)

Problemas de múltiplos estados (ambiente determinístico mas inacessível)

Problemas de contingência (ambiente não determinístico e inacessível, é necessário usar sensores durante a execução, solução é uma árvore ou política)

Problemas de exploração (espaço de estados desconhecido)

72

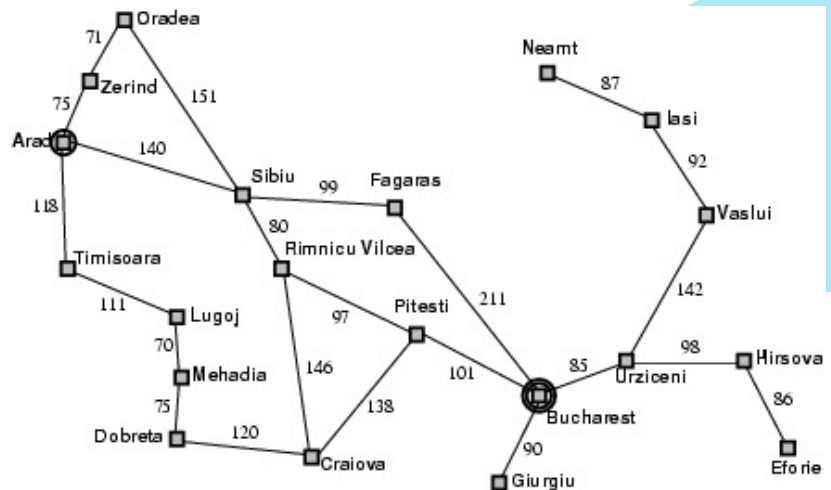
Exemplo: Mapa de Estradas da Roménia

Estado Inicial:

Arad; Estado Objectivo: Bucharest

Operadores:

Arcos com respectivos custos



73

Problema do Puzzle-8

Estados, Operadores, Teste Objetivo, Custo da Solução ?

7	2	4
5		6
8	3	1

Estado inicial

	1	2
3	4	5
6	7	8

Estado final

74

Problema do Puzzle-8

Estados

Especifica a posição de cada uma das peças e do espaço vazio (várias representações são possíveis)

Estado inicial

Representado na figura

Operadores - sucessores:

Gera os estados válidos que resultam da execução. São as quatro ações (mover espaço vazio para esquerda, direita, cima ou abaixo)

Teste de objetivo

Verifica se o estado corresponde à configuração objetivo (representado na figura)

Custo da solução

Cada passo custa 1, sendo o custo da solução o número de passos para resolver o problema

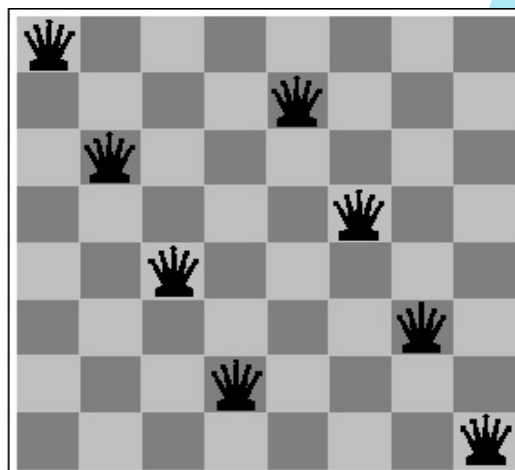
7	2	4
5		6
8	3	1

	1	2
3	4	5
6	7	8

75

Problema das N-Rainhas

Estados, Operadores, Teste Objetivo, Custo da Solução



76

Problema das N-Rainhas

Teste Objetivo: 8 Rainhas no tabuleiro sem nenhum ataque

Formulação 1:

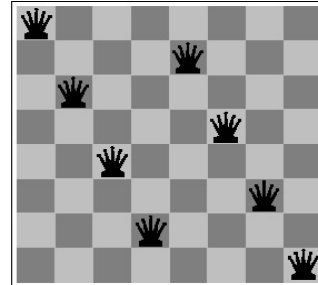
- Estado: Qualquer arranjo de 0 a 8 Rainhas no tabuleiro
- Operador: Adicionar uma rainha em qualquer quadrado
 - Temos 64^8 sequências possíveis!

Formulação 2:

- Estado: Arranjos de 0 a 8 Rainhas, uma em cada coluna, sem ataques!
- Operador: Adicionar uma rainha na coluna mais à esquerda que estiver vazia, sem atacar nenhuma outra
 - Temos 2057 sequências possíveis!

Formulação 3:

- Estado: Arranjos de 8 Rainhas no tabuleiro, uma em cada coluna!
- Operador: Movimentar rainha atacada para casa da mesma coluna



77

Criptogramas

Encontrar dígitos (todos diferentes), um para cada letra de forma a que a soma seja correta!

Estados: Puzzle com algumas letra substituídas por números

Operadores: Substituir todas as ocorrências de uma letra por um dígito

Teste Objetivo: Puzzle só contém dígitos e a soma está correta!

Custo da Solução: 0 (todas as soluções são iguais)

F	O	R	T	Y	
	T	E	N		
+		T	E	N	
<hr/>					
	S	I	X	T	Y

S	E	N	D	
+	M	O	R	E
<hr/>				
M	O	N	E	Y

78

Criptogramas

Soluções dos Criptogramas:

- Criptograma 1: F=2, O=9, R=7, T=8, Y=6, E=5, N=0, S=3, I=1, X=4
- Criptograma 2: S=9, E=5, N=6, D=7, M=1, O=0, R=8, Y=2

<div> FORTY TEN + TEN ----- SIXTY </div>	<div> 29786 850 + 850 ----- 31486 </div>	<div> SEND + MORE ----- MONEY </div>	<div> 9567 + 1085 ----- 10652 </div>
--	--	---	---

Mais soluções?

79

Problemas do Mundo Real

Encontrar o melhor caminho de um ponto a outro

aplicações: google maps, redes de computadores, planeamento militar, viagens aéreas

Visitar cada ponto pelo menos uma vez num dado espaço



80

Exercício: Missionários e Canibais

Problema dos Missionários e Canibais

3 missionários e 3 canibais estão numa das margens do rio com um barco que só leva 2 pessoas. Encontrar uma forma de levar os 6 para a outra margem do rio sem nunca deixar mais canibais do que missionários numa das margens durante o processo!

Formular este problema como um problema de pesquisa, definindo a representação do estado, estado inicial, os operadores (e respetivas pré-condições e efeitos), o teste objetivo e o custo da solução.

Resolver o problema através de uma pesquisa em árvore

81

Exercício: Problema dos Baldes

Dois baldes, de capacidades c_1 e c_2 , inicialmente vazios, sem qualquer marcação intermédia. As únicas operações que pode realizar são:

- esvaziar um balde
- encher (completamente) um balde
- despejar um balde para o outro até que o segundo fique cheio
- despejar um balde para o outro até que o primeiro fique vazio

O objetivo consiste em determinar quais as operações a efetuar de modo a que o primeiro balde contenha n litros (exemplo: 2 litros)?

Formule o Problema como um problema de pesquisa

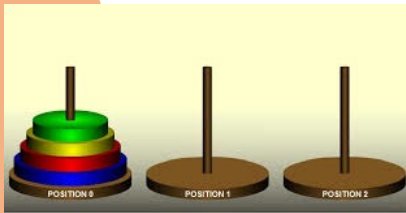
82

Exercício: Torres de Hanoi

Problema

3 torres (A, B e C) e 4 discos (D1 a D4).

Inicialmente os discos encontram-se na torre A e o objetivo é transferi-los para a torre C.



Em cada jogada, o jogador pode deslocar um disco de uma torre para outra torre, desde que não coloque esse disco sobre um disco menor.

Formule uma versão genérica do problema (N discos em M torres) como um problema de pesquisa.

83

Pesquisa da Solução

Método para realizar a Pesquisa da Solução:

1. Começar com o estado inicial
2. Executar o teste do objetivo
3. Se não foi encontrada a solução, usar os operadores para expandir o estado atual gerando novos estados - sucessores (expansão)
4. Executar o teste objetivo
5. Se não tivermos encontrado a solução, escolher qual o estado a expandir a seguir (estratégia de pesquisa) e realizar essa expansão
6. Voltar a 4)

84

Pesquisa da Solução - Árvore de Pesquisa

Árvore de pesquisa composta por nós. Nós folhas, ou não têm sucessores ou ainda não foram expandidos!

Importante distinguir entre a árvore de pesquisa e o espaço de estados!

Nó da Árvore (cinco componentes):

- Estado a que corresponde
- Nó que lhe deu origem (pai)
- Operador aplicado para o gerar
- Profundidade do nó
- Custo do caminho desde o nó inicial

Fronteira: Conjunto de nós à espera de serem expandidos

(Representada como uma fila)

85

Estratégias de Pesquisa

CrITÉRIOS de Avaliação:

- Completude: Está garantido que encontra a solução?
- Complexidade no Tempo: Quanto tempo demora a encontrar a solução?
- Complexidade no Espaço: Quanta memória necessita para efectuar a pesquisa?
- Optimalidade: Encontra a melhor solução?

Estratégias de Pesquisa:

- Pesquisa Não-Informada (cega)
- Pesquisa Informada (heurística)

86

Representação de um problema

O modelo criado para representar o problema é determinante no esforço requerido para a sua resolução.

O modelo permite

- Visualizar o problema
- Especificar as estruturas
- Controlar o processo de resolução

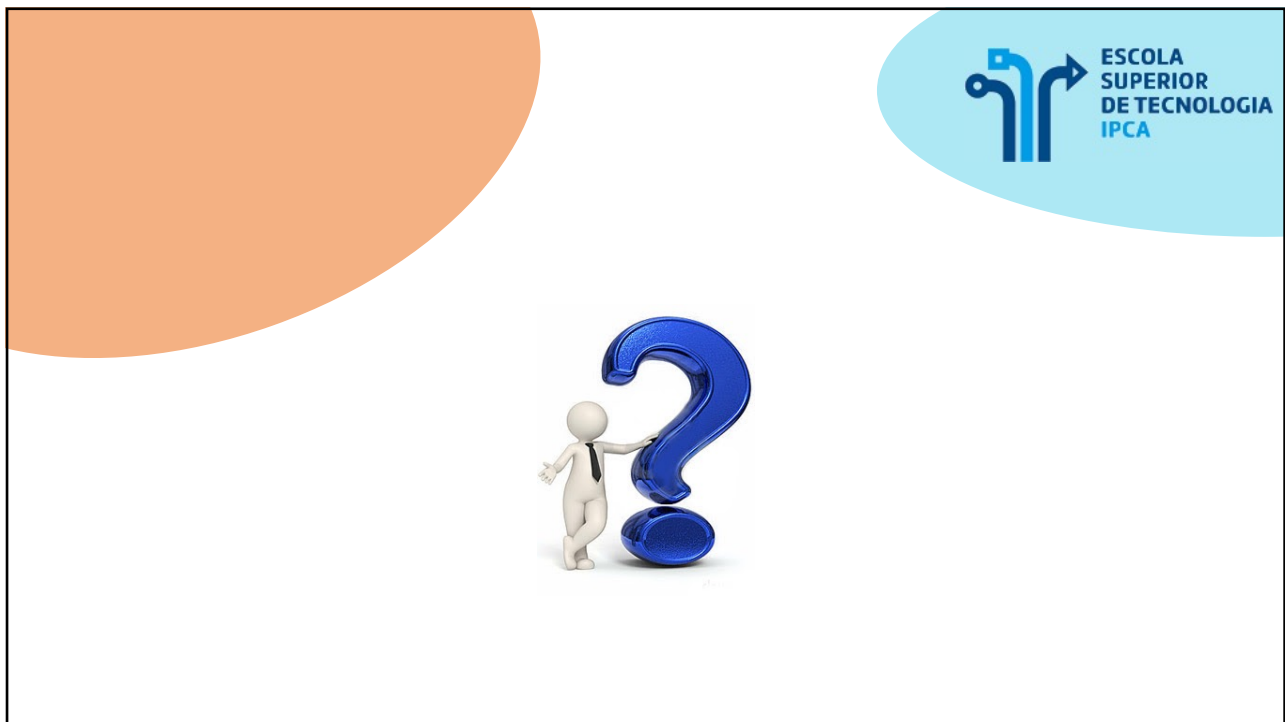
87

Modelo

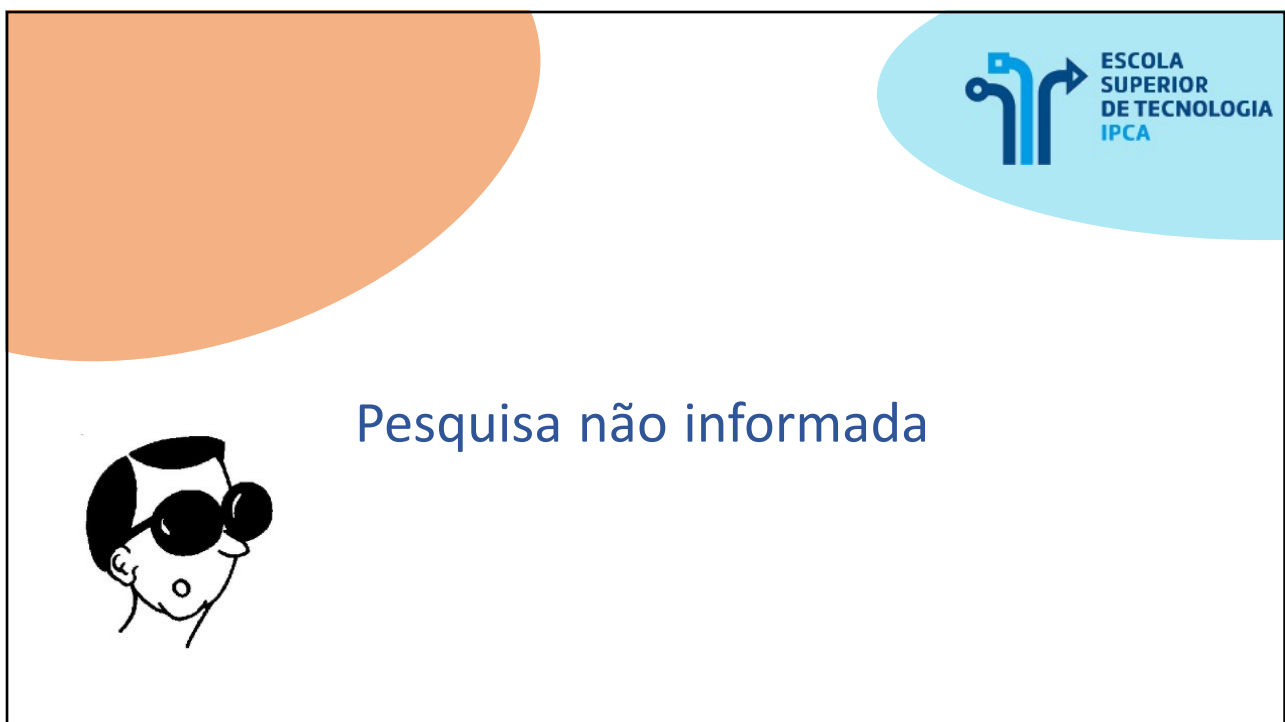
Um bom modelo do problema deverá possuir as seguintes características:

- Clareza – o modelo e o problema deverão estar relacionados de forma evidente
- Exactidão – o modelo deve ser fiel à realidade nas características relevantes do problema
- Completude – o modelo deve representar todos os aspectos relevantes para a resolução do problema
- Eficiência – a representação deve poder ser utilizada de forma eficiente
- Conciso – devem ser representados apenas os aspectos relevantes
- Utilidade – deve ser avaliada a capacidade do modelo para a resolução do problema

88



89



Pesquisa não informada

90

Métodos

Breath-First (Largura Primeiro)

Uniform-Cost (Custo Uniforme)

Depth-First (Profundidade Primeiro)

Depth-Limited (Profundidade Limitada)

Depth-Iterative (Profundidade iterativa)

Bidireccional

91

Notação

Sigla	Significado
b	Factor de ramificação
d	Profundidade da solução
m	Profundidade máxima pesquisa
l	Limite de profundidade

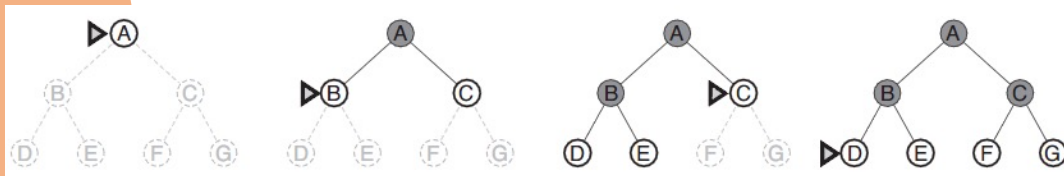
92

Primeiro em Largura

Breadth-first search

Estratégia: Todos os nós de menor profundidade são expandidos primeiro

- Vantagem: Pesquisa muito sistemática
- Desvantagem: Normalmente demora muito tempo e sobretudo ocupa muito espaço



93

Primeiro em Largura

Se existe solução será encontrada – completo e ótimo

A solução é encontrada no nó de menor profundidade

Solução com profundidade d exige

$$1 + b + b^2 + \dots + b^d$$

Complexidade exponencial no espaço e no tempo: $O(b^d)$

Em geral só pequenos problemas podem ser resolvidos assim!

94

Custo Uniforme

Estratégia

Semelhante ao Breath-First mas a expansão é efectuada sempre com o nó com menor custo da fronteira (medido pela função de custo da solução)

Se garantirmos que $g(\text{sucessor}) \geq g(n)$ então a primeira solução encontrada é a mais barata.

↓
Custo de ir da raiz até ao nó n

Pesquisa Primeiro em Largura e Pesquisa de Custo Uniforme são iguais se
 $g(n) = \text{Depth}(n)$

95

Primeiro em Profundidade

Depth-First Search

Estratégia

Expandir sempre um dos nós (o mais à esquerda) mais profundos da árvore, quando se chega a um nó sem sucessor expande-se o seguinte com maior profundidade

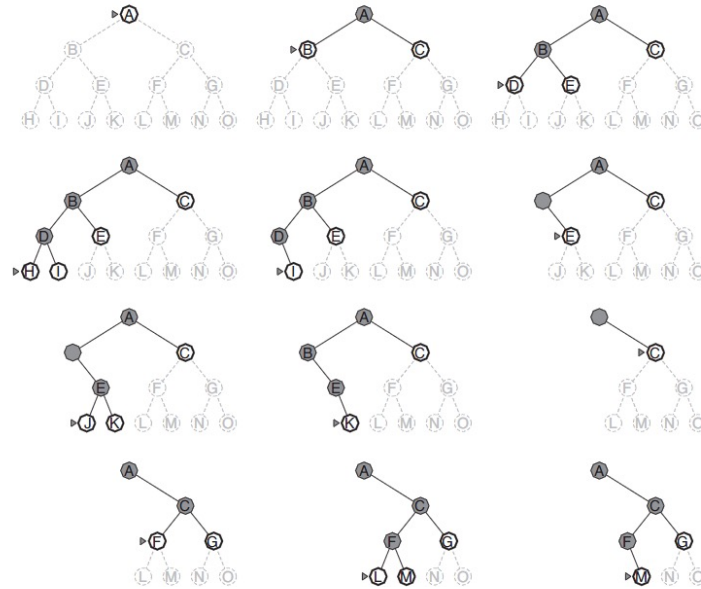
- Vantagem: Pouca memória necessária, bom para problemas com muita soluções
- Desvantagem: Não pode ser usada para árvores com profundidade infinita, pode ficar presa em ramos errados

Se for definida uma profundidade limite transforma-se em

Pesquisa com Profundidade Limitada

96

Primeiro em Profundidade



97

Primeiro em Profundidade

A estratégia não é completa nem ótima (no caso de termos profundidade infinita)

Solução com profundidade d exige

$$1 + b + b^2 + \dots + b^d$$

Complexidade no espaço: $b \times m$

Complexidade no tempo: $O(b^m)$

Para problemas com diversas soluções pode tornar-se muito mais rápido do que a pesquisa em largura

Inapropriado para problemas com que gerem árvores muito profundas ou ilimitadas

98

Profundidade Limitada

Coloca-se um limite à profundidade na estratégia da pesquisa Profundidade Primeiro.

Resolve o problema da profundidade ilimitada.

A ideia passa por admitir que uma solução não deverá estar muito afastada do nó inicial, assim, despreza-se as partes do grafo que não estão suficientemente perto do nó inicial.

A estratégia não é completa nem ótima.

99

Profundidade Iterativa

Estratégia: Executar pesquisa em profundidade limitada, iterativamente, aumentando sempre o limite da profundidade

A estratégia é completa e ótima

Complexidade no tempo: $O(b^d)$

Complexidade no espaço: $O(b \times d)$

Em geral é a melhor estratégia para problemas com um grande espaço de pesquisa e em que a profundidade da solução não é conhecida

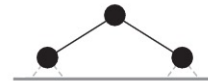
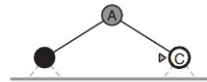
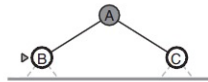
100

Profundidade Iterativa

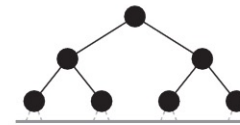
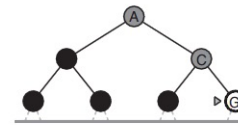
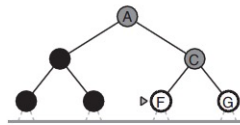
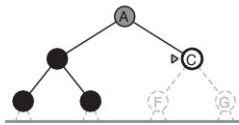
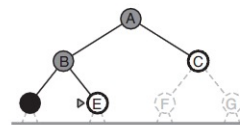
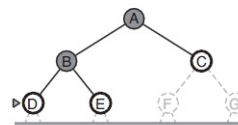
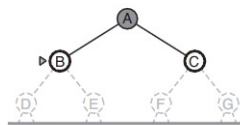
Limit = 0



Limit = 1



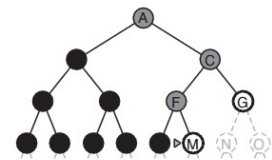
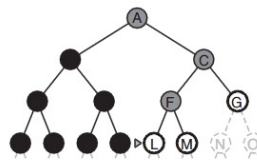
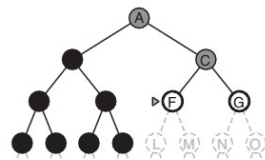
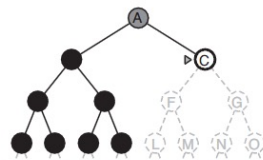
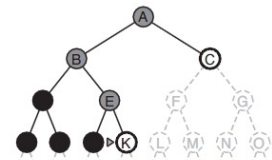
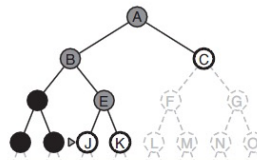
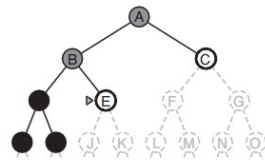
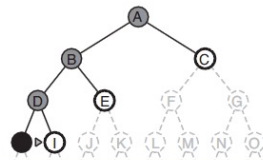
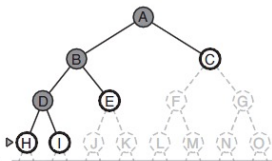
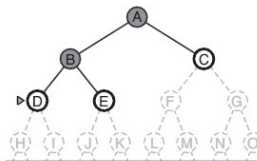
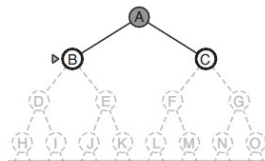
Limit = 2



101

Profundidade Iterativa

Limit = 3



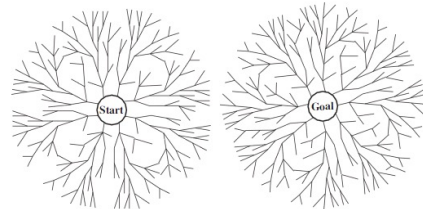
102

Bidireccional

Estratégia

Executar pesquisa para a frente desde o estado inicial e para trás desde o objetivo, simultaneamente

- Vantagem: Pode reduzir enormemente a complexidade no tempo
- Desvantagem: Será possível gerar os predecessores?
- E se existirem muitos estados objetivo?
- Como fazer o “matching” entre as duas pesquisas?
- Que tipo de pesquisa fazer nas duas metades?



103

Comparação entre Estratégias de Pesquisa

Avaliação das estratégias de pesquisa

Método	Largura primeiro	Custo uniforme	Profundidade primeiro	Profundidade limitada	Profundidade iterativa
Completeness	Completo	Completo	Não completo	Não completo	Completo
Tempo	$O(b^d)$	$O(b^d)$	$O(b^d)$	$O(b^d)$	$O(b^d)$
Espaço	$O(b^d)$	$O(b^d)$	$O(b \times m)$	$O(b \times l)$	$O(b \times d)$
Optimality	Ótimo	Ótimo	Não Ótimo	Não Ótimo	Ótimo

b – factor de ramificação; d – profundidade da solução; m – profundidade máxima; l – limite da profundidade

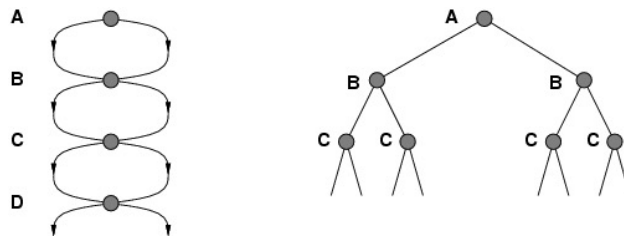
104

Estados Repetidos

Falha em detetar estados repetidos pode tornar um problema linear num problema exponencial!

Evitar Estados Repetidos

- Não voltar ao estado anterior
- Não criar ciclos (não voltar a estados por onde passou na sequência)
- Não usar nenhum estado repetido (será possível? Qual o custo computacional?)



105

Exercícios - Pesquisa

Formular como problemas de pesquisa e resolver utilizando as várias estratégias de pesquisa estudadas

- Missionários e Canibais
- Problema dos Baldes
- Torres de Hanoi
- Criptogramas
- 8-Rainhas e N-Rainhas
- 8-Puzzle e N-Puzzle

Nota: Para todos os problemas tente que a solução seja o mais genérica possível de modo a permitir resolver versões com dados diferentes

106

Exercícios - Pesquisa

Mapa: Colorir um mapa plano, usando 4 cores de forma a que dois vértices adjacentes não tenham a mesma cor.

Corrente: Juntar um conjunto de elos de corrente para formar uma corrente completa. Os operadores podem abrir um elo ou fechar um elo. Na forma mais usual, o estado inicial é composto por quatro correntes com três elos e o objetivo por uma corrente circular com 12 elos.

Jogo do Solitário: O objetivo é capturar todas as peças ficando uma peça no final no tabuleiro.

Solitário de Cartas: Resolver a paciência de cartas “solitário” supondo que todas as cartas estão descobertas no início.

107



108

Pesquisa informada



109

Pesquisa Informada

Utiliza informação do problema para “orientar” o algoritmo de pesquisa

Estratégia de Pesquisa:

Definida escolhendo a ordem de expansão dos nós!

Pesquisa do Melhor Primeiro (Best-First Search)

Utiliza uma função de avaliação que retorna um número indicando o interesse de expandir um nó

Pesquisa Gulosa (Greedy-Search)

$f(n) = h(n)$ – estima distância à solução

A*

$f(n) = g(n) + h(n)$ – estima custo da melhor solução que passa por n

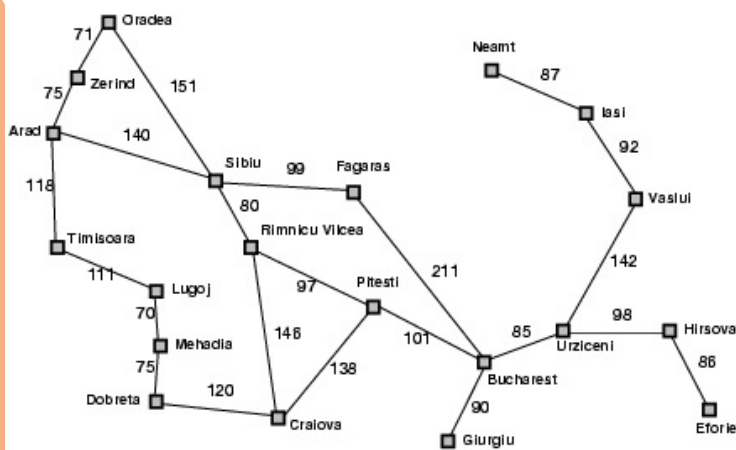
110

Pesquisa Informada

Estado Inicial: Arad | Objetivo: Bucharest

$h(n)$ = distância em linha reta

Informação adicional



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

111

Pesquisa Gulosa (Greedy-Search)

Estratégia:

Expandir o nó que parece estar mais perto da solução

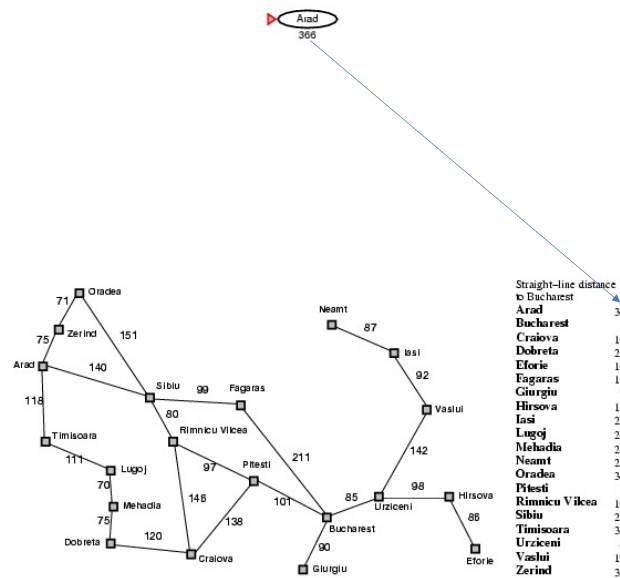
$h(n)$ = custo estimado do caminho mais curto do estado n para o objetivo (função heurística)

Exemplo:

$h(n)$ = distância em linha reta entre n e o objetivo

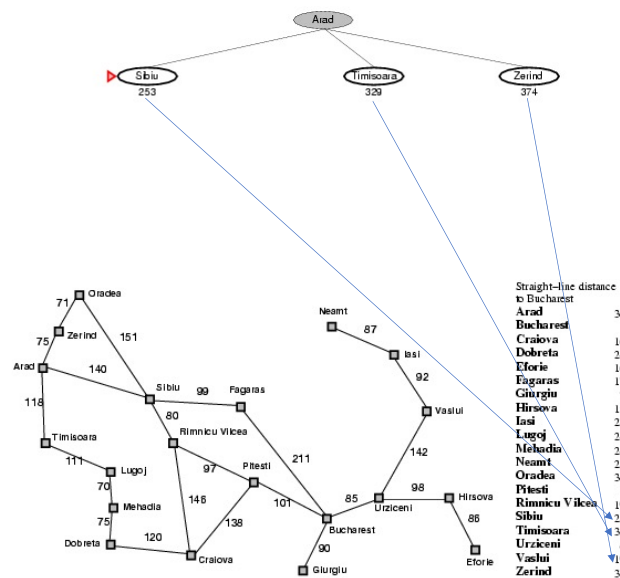
112

Pesquisa Gulosa (Greedy-Search)



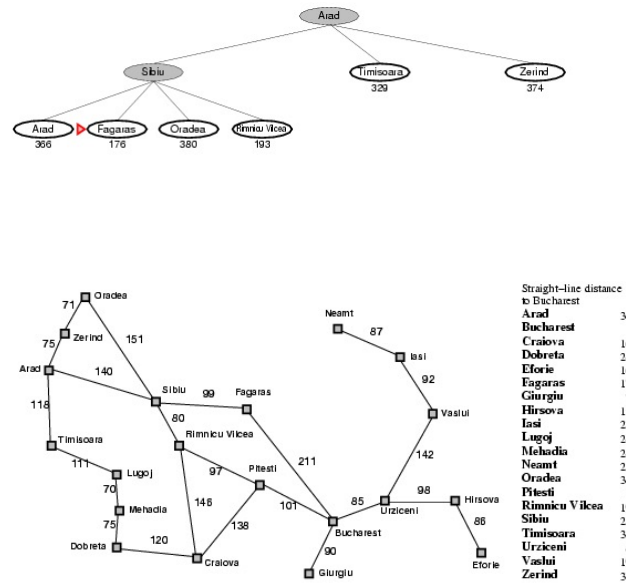
113

Pesquisa Gulosa (Greedy-Search)



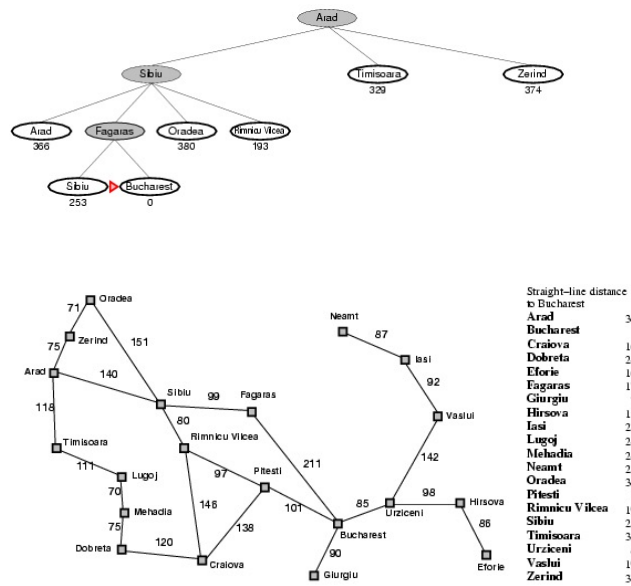
114

Pesquisa Gulosa (Greedy-Search)



115

Pesquisa Gulosa (Greedy-Search)



116

Pesquisa Gulosa (Greedy-Search)

Propriedades:

Completa? Não!

pode entrar em ciclos! (ex.: lasi → Neamt → lasi → Neamt → lasi ...)

Suscetível a falsos começos

Complexidade no tempo? $O(b^m)$

Mas uma boa função heurística pode diminuir consideravelmente o tempo

Complexidade no espaço? $O(b^m)$

Mantém todos os nós na memória

Semelhante à procura em profundidade mas
mais exigente no espaço

Ótima? Não! Não encontra sempre a solução ótima!

Necessário detetar estados repetidos!

117

A*

Estratégia:

O algoritmo A* combina a pesquisa gulosa com a de custo uniforme minimizando a soma do caminho já efetuado com o mínimo previsto que falta até a solução

A ideia é evitar a expansão de nós cujo custo é muito elevado

118

A*

Função de avaliação: $f(n) = g(n) + h(n)$

$g(n)$ = custo total, até agora, para chegar ao estado n

$h(n)$ = custo estimado para chegar ao objetivo

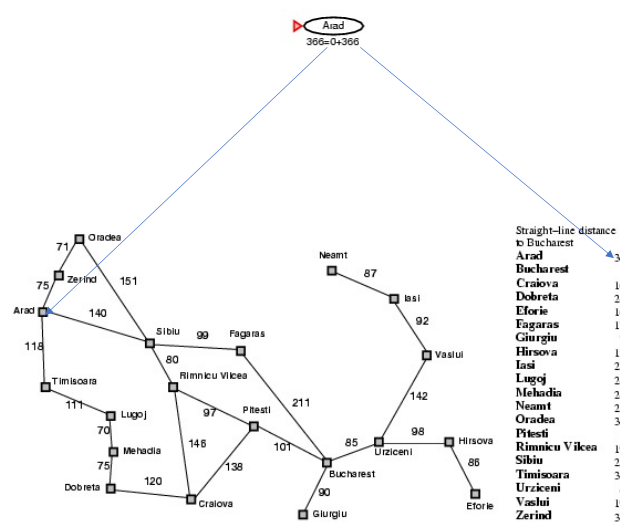
$f(n)$ = custo estimado da solução mais barata que passa pelo nó n

119

A*

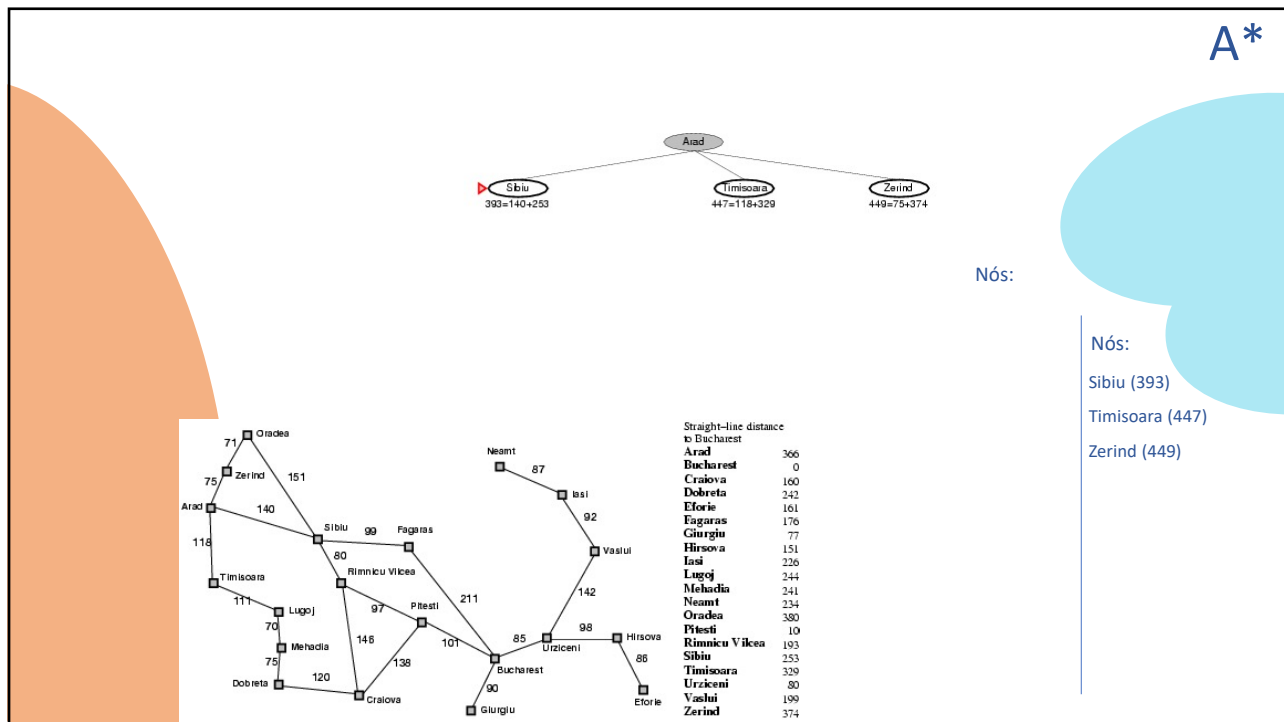
Estado Inicial: Arad; Objetivo: Bucharest; $f(n) = g(n) + h(n)$

$g(n)$ = custo até n ; $h(n)$ = distância em linha reta ao objetivo

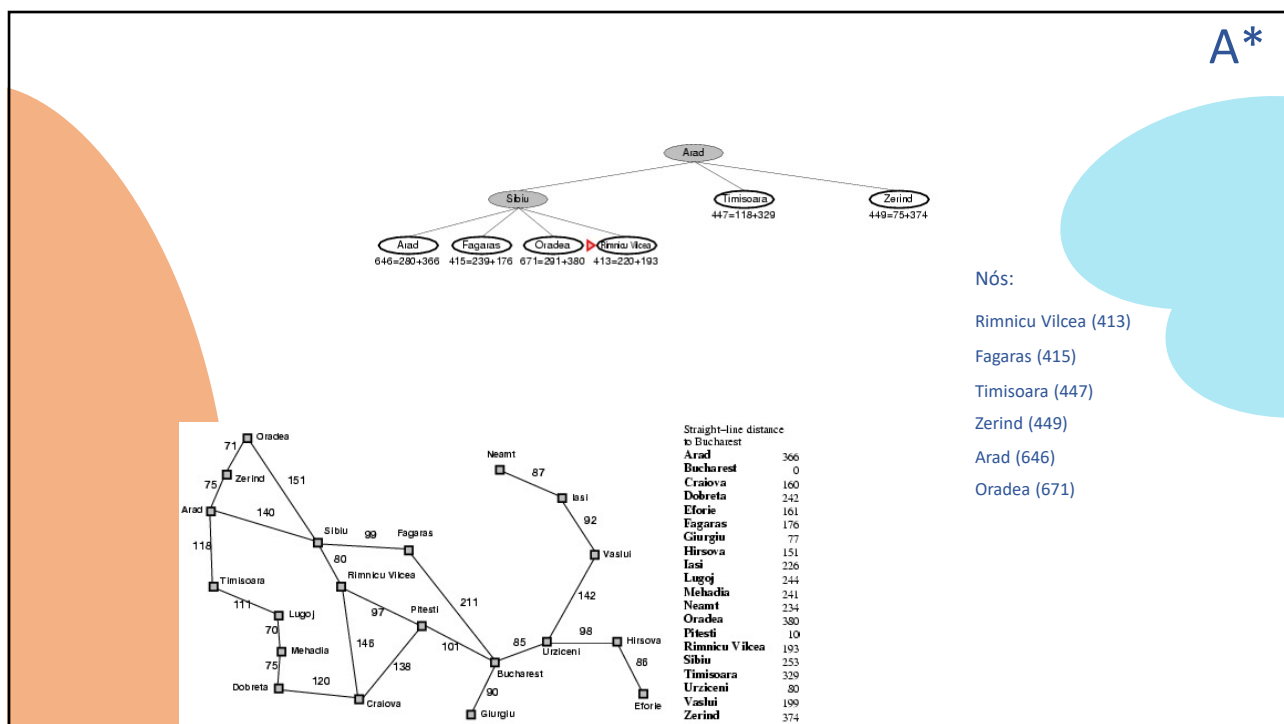


Nós:
Arad (366)

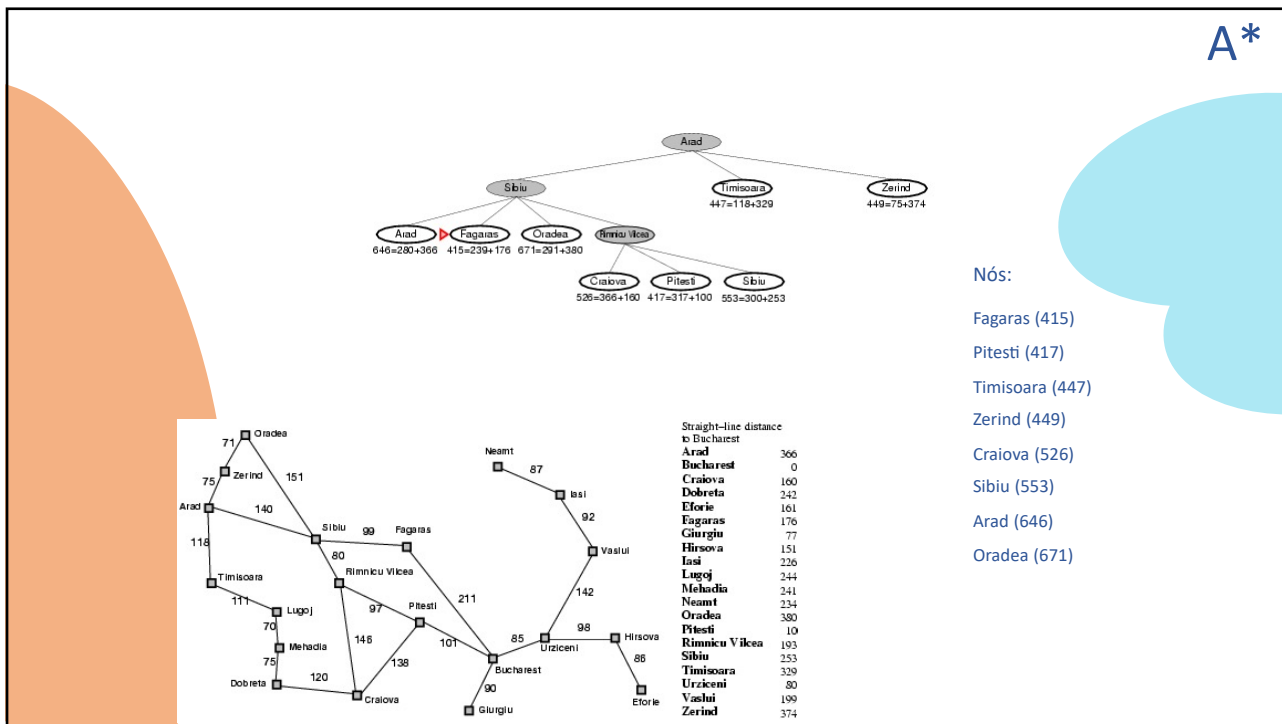
120



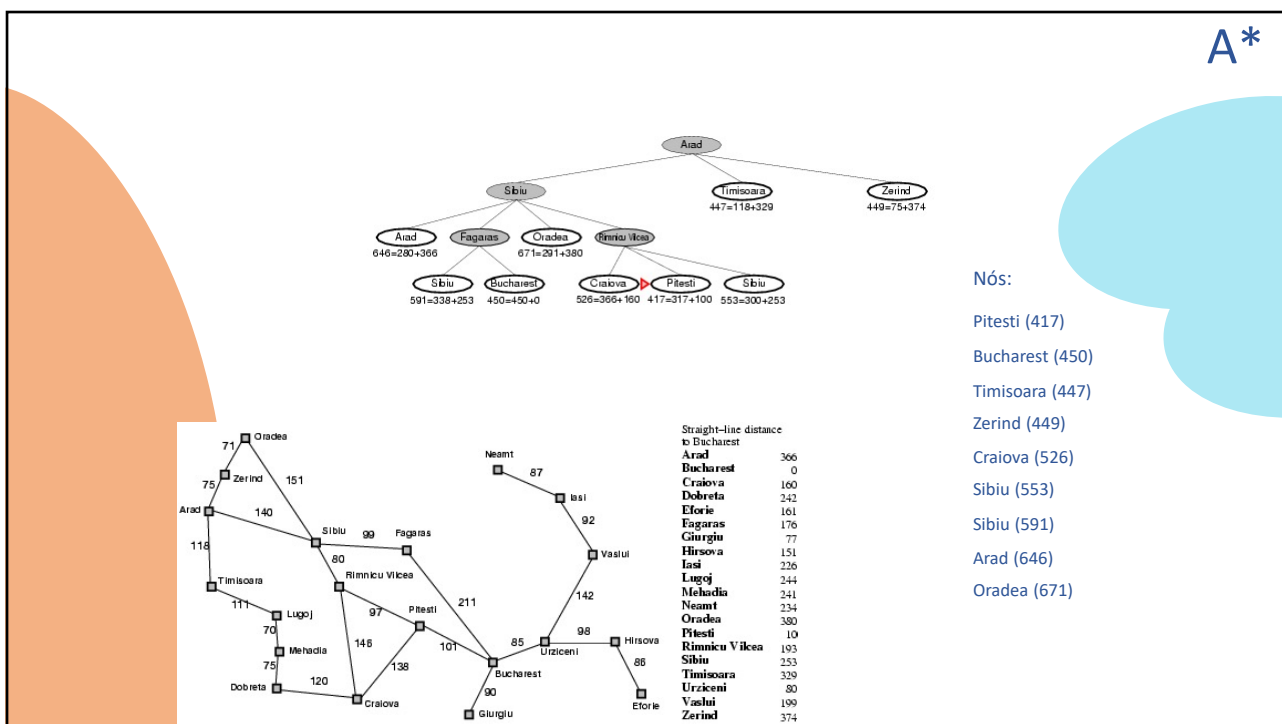
121



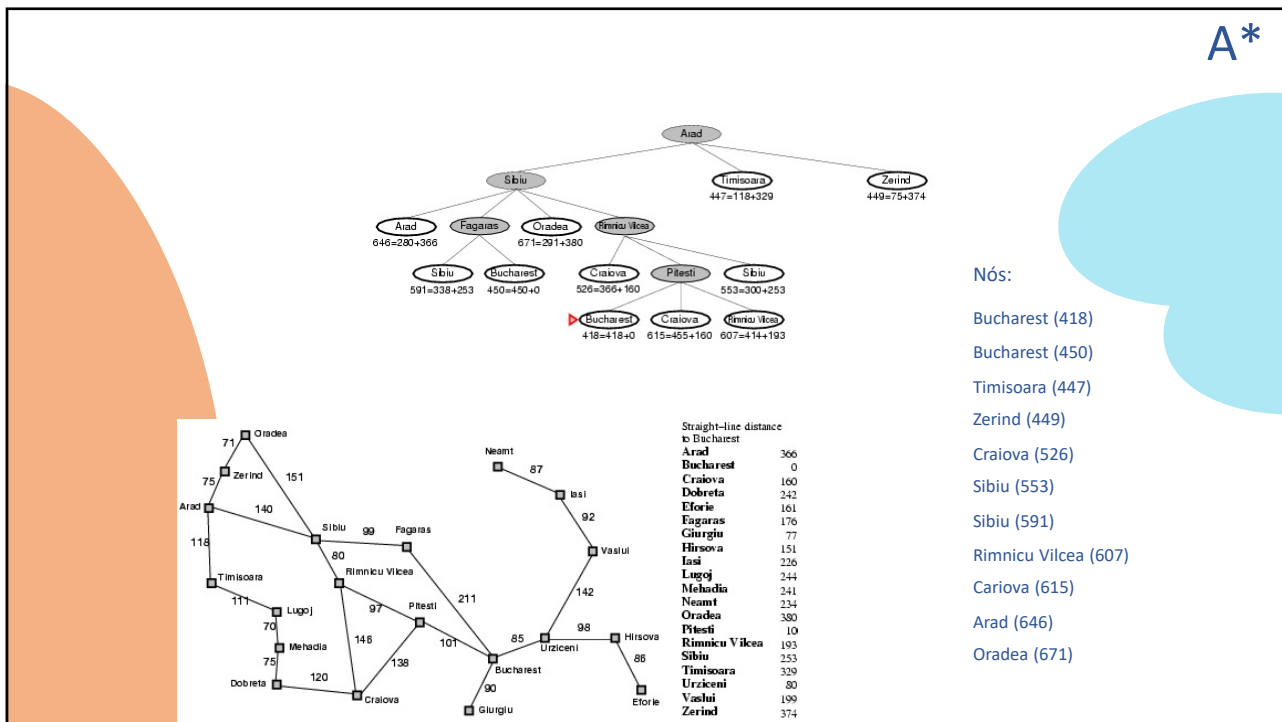
122



123



124



125

A*

Heurísticas admissíveis

Uma heurística $h(n)$ é admissível sse para cada nó se verifica

$$h(n) \leq h^*(n)$$

onde $h^*(n)$ é o custo real do caminho desde n até ao objectivo.

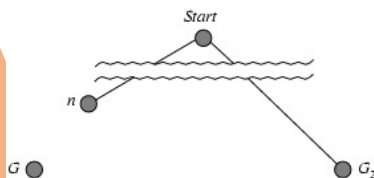
Uma heurística admissível nunca sobrestima o custo de atingir o objectivo - é realista ou optimista

Teorema: se $h(n)$ é admissível, então a procura em árvore A* é ótima

126

Prova da Optimalidade do A*

Consideremos um nó objectivo não óptimo G_2 que já foi gerado mas não expandido (nó folha).
Seja um nó folha tal que n está no menor caminho para um nó objectivo óptimo G (com custo C^*)

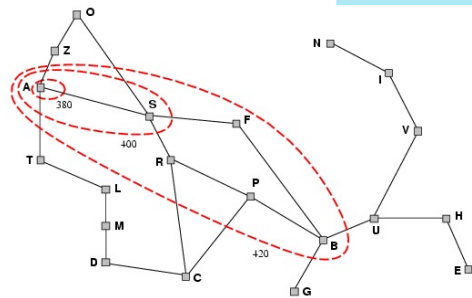


$h(G_2)=0$ – porque G_2 é objectivo

$f(G_2) = g(G_2) + h(G_2) = g(G_2) > C^*$ - porque G_2 não é óptimo

$f(n) = g(n) + h(n) \leq C^*$ - porque h é uma heurística admissível

Logo $f(n) \leq C^* < f(G_2)$ e assim G_2 não será analisado antes de n - o algoritmo A* nunca escolherá G_2 para expansão!



127

Pesquisa A*

Se $h(n)$ é uma heurística admissível então

Algoritmo A* será ótimo e completo ?

Se existir mais do que um caminho que passe por um nó n que está no caminho óptimo o melhor caminho pode ser descartado porque aquele nó já foi visitado a partir de outro caminho

É necessária uma heurística consistente!!!

128

Pesquisa A*

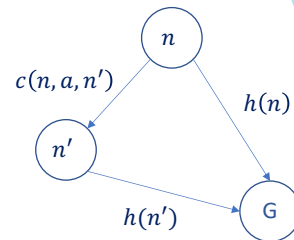
Uma heurística é consistente se para cada sucessor de n (n') gerado por uma determinada acção a

$$h(n) \leq c(n, a, n') + h(n')$$

Onde $c(n, a, n')$ é o custo de ir de n a n' através de a

Se h é consistente então

$$\begin{aligned} f(n') &= g(n') + h(n') = g(n) + c(n, a, n') + h(n') \geq \\ &\geq g(n) + h(n) \geq f(n) \end{aligned}$$



É ótima e completa excepto de o número de nós com $f \leq f(G)$ for infinito

Complexidade no tempo: $O(b^m)$

Complexidade no espaço: $O(b^m)$ - Mantém todos os nós em memória!

129

Pesquisa IDA*

Iterative Deepening A* (IDA*)

Semelhante ao A* mas em cada iteração é incrementado o valor limite da profundidade

- Em cada iteração o valor limite de $f(n)$ é incrementado
- Se o valor de $f(n)$ for superior ao limite o nó n não é explorado
- Em cada iteração o valor limite é actualizado com o menor valor de $f(n)$ para os nós não explorados na iteração anterior

Requer menos espaço do que o algoritmo A*

130

Pesquisa SMA*

Simplified Memory Bounded A* (SMA*)

Alteração do algoritmo A* quando não existe memória disponível

SMA* utiliza toda a memória disponível, evitando estados repetidos

Estratégia: Quando necessita de gerar um sucessor e não tem memória, esquece um nó da fila que pareça pouco prometededor (com um custo alto).

131

Pesquisa RBFS

Recursive Best First Search (RBFS)

Semelhante à procura em profundidade

- Para cada nó explorado mantém o registo do caminho alternativo com menor valor de f
- Se o valor de f para o nó actual for superior ao valor do nó em memória (do menor valor), recuperamos o caminho alternativo
- Cada alteração corresponde a uma iteração do IDA*

132

Funções Heurísticas - 8 Puzzle

Solução típica em 20 passos com factor de ramificação médio: 3

Número de estados: $3^{20} = 3.5 \cdot 10^9$

Nº Estados (sem estados repetidos) = $9! = 362880$

Heurísticas:

$h1$ = Nº de peças fora do sítio

$h2$ = Soma das distâncias das peças até às suas posições correctas

7	2	4
5		6
8	3	1

Estado inicial

	1	2
3	4	5
6	7	8

Estado final

133

Funções Heurísticas - 8 Puzzle (2)

Relaxação de Problemas como forma de inventar heurísticas:

Peça pode-se mover de A para B se A é adjacente a B e B está vazio

Relaxamento

- a) Peça pode-se mover de A para B se A é adjacente a B
- b) Peça pode-se mover de A para B se B está vazio
- c) Peça pode-se mover de A para B

134

Exercício

Suponha o seguinte jogo (para 1 jogador) em que o tabuleiro é constituído por 6 casas e 6 fósforos. O objetivo do jogo é colocar um fósforo em cada um dos quadrados.

Para tal, em cada jogada, o jogador pode:

- Movimentar 1 fósforo de uma casa para outra casa que esteja à sua direita
- Movimentar 2 fósforos de uma casa para outra casa que esteja à sua esquerda
- Movimentar 2 ou 3 fósforos de uma casa para outra que esteja acima ou abaixo dela

- A. Formule o problema como um problema de pesquisa.
- B. Partindo dum determinado estado, desenhe as árvores de pesquisa utilizando as estratégias primeiro em largura e primeiro em profundidade (considere primeiro os movimentos a partir do quadrado 1, depois a partir do 2 e assim sucessivamente).
- C. Represente graficamente o espaço de estados (ignore estados em que um quadrado tenha mais de 3 fósforos).
- D. Defina duas funções heurísticas que lhe permitam aplicar o algoritmo A* ao problema. Explique em que consiste o algoritmo A* e aplique-o para resolver o problema.

135

Algoritmos de Melhoria Iterativa

Em muitos problemas de otimização, o caminho para o objetivo é irrelevante! O objetivo é ele mesmo a solução!

Espaço de Estados = conjunto das configurações completas!

Algoritmos de melhoria iterativos partem de um estado (corrente) e tentam melhorá-lo!

Estratégia: Começar como uma solução inicial do problema e fazer alterações de forma a melhorar a sua qualidade

136

Algoritmos de Melhoria Iterativa

Hill-Climbing Search

- Escolher um estado aleatoriamente do espaço de estados
- Considerar todos os vizinhos desse estado
- Escolher o melhor vizinho
- Repetir o processo até não existirem vizinhos melhores
- O estado corrente é a solução

Simulated Annealing

- Semelhante ao Hill-Climbing Search mas admite explorar vizinhos piores

137

Algoritmos de Melhoria Iterativa

Tabu Search

- Explora os estados vizinhos mas elimina os piores (vizinhos tabu)

Particle Swarm Optimization

- Vários estados de partida (enxame)
- Explora-se a vizinhança e guarda-se, a melhor solução e o melhor estado
- Movimentam-se os estados na direcção da melhor solução encontrada até ao momento
- A velocidade de movimentação depende das distâncias à melhor solução e melhor estado e da posição do estado

138

Algoritmos de Melhoria Iterativa

Ant Colony Optimization

Vários estados (colônia formigas) de partida

Determina-se a probabilidade de um caminho ser melhor a partir do número de “formigas” que passa por ele

Genetic Algorithms

Definição do estado como um cromossoma

Gerar soluções (cromossomas) a partir de uma população de estados inicial

139



140

Pesquisa com adversário



141

Jogos como Problemas de Pesquisa

Agente Hostil (adversário) incluído no mundo!

Oponente Imprevisível => Solução é um Plano de Contingência

Tempo Limite => Pouco provável encontrar objetivo! É necessário uma aproximação

Xadrez:

- Todos consideram que é necessário inteligência para jogar
- Regras simples mas o jogo é complexo
- Mundo totalmente acessível ao agente
- Factor de ramificação médio de 35, partida com 50 jogadas => 35^{100} folhas na árvore de pesquisa (embora só existam 10^{40} posições legais)

Conceito de corte na árvore de pesquisa e função de avaliação!

142

Tipos de Jogos

Tipos de Jogos:

- Informação:
 - Perfeita: Xadrez, Damas, Go, Otelo, Gamão, Monopólio
 - Imperfeita: Poker, Scrabble, Bridge, King
- Sorte/Determinístico:
 - Determinístico: Xadrez, Damas, Go, Otelo
 - Jogo de Sorte: Gamão, Monopólio, Poker, Scrabble, Bridge, King

Plano de “Ataque”:

- Algoritmo para o jogo perfeito
- Horizonte finito, avaliação aproximada
- Cortes na árvores para reduzir custos

143

Algoritmo MiniMax

Jogo: Problema de pesquisa com:

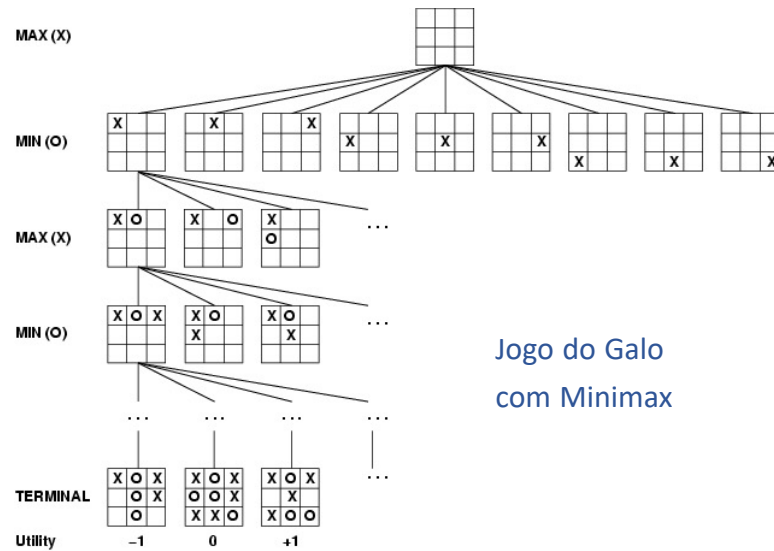
- Estado Inicial (posição do tabuleiro e qual o próximo jogador a jogar)
- Conjunto de Operadores (que definem os movimentos legais)
- Teste Terminal (que determina se o jogo acabou ou seja está num estado terminal)
- Função de Utilidade (que dá um valor numérico para o resultado do jogo, por exemplo 1-vitória, 0-empate, -1-derrota)

Estratégia do algoritmo Minimax:

- Gerar a árvore completa até aos estados terminais
- Aplicar a função utilidade a esses estados
- Calcular os valores da utilidade até a raiz da árvore, uma camada de cada vez
- Escolher o movimento com o valor mais elevado!

144

Minimax - Exemplo para o Jogo do Gato

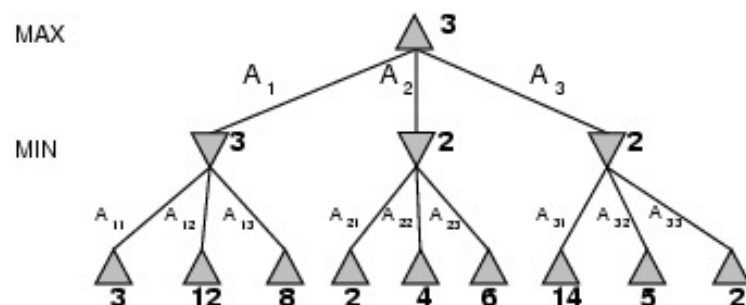


145

Minimax - Exemplo Geral

Estratégia:

Escolher o movimento que tem o maior valor minimax: melhor que se pode conseguir contra as melhores respostas do adversário!



146

Propriedades do Minimax

Completo? Sim se a árvore for finita!

Ótimo? Sim contra um adversário ótimo! Senão?

Complexidade no Tempo? $O(b^m)$

Complexidade no Espaço? $O(b \times m)$ (exploração primeiro em profundidade)

Problema: Inviável para qualquer jogo minimamente complexo

Exemplo: Xadrez ($b=35$, $m=100$), $b^m = 35^{100} = b^m = 2,5 \times 10^{154}$

Supondo que são analisadas 450 milhões de hipóteses por segundo são necessários 2×10^{138} anos para chegar à solução!

147

Recursos Limitados

Supondo que temos 100 segundos e exploramos 10^4 nós/segundo, podemos explorar 10^6 nós por movimento

Aproximação usual:

- Teste de Corte: Profundidade Limite
- Função de Avaliação: Utilidade (interesse) estimada para a posição

Exemplo (Xadrez):

- Teste de Corte: Profundidade de Análise n
- Função de Avaliação simples = soma dos valores das peças brancas em jogo menos a soma dos valores das peças negras em jogo!
- Função de avaliação só deve ser aplicada a posições estáveis (em termos do seu valor). Por exemplo, posições com possíveis capturas devem ser mais exploradas...
- Outro Problema: Problema do horizonte!

148

Cortes à Pesquisa

MinimaxCutoff é idêntico ao MinimaxValue excepto:

- Terminal-Test é substituído por Cutoff
- Utility é substituída por Evaluation (que calcula uma avaliação da posição atingida)

Será que funciona na prática?

- Se $b^m = 10^6$ com $b=35 \Rightarrow m=4$

Jogar de xadrez com profundidade 4 é absolutamente miserável!

- Profundidade 4 \Rightarrow Jogador Novato
- Profundidade 8 \Rightarrow PC, Muito bom jogador humano
- Profundidade 12 \Rightarrow Deep Blue, Kasparov

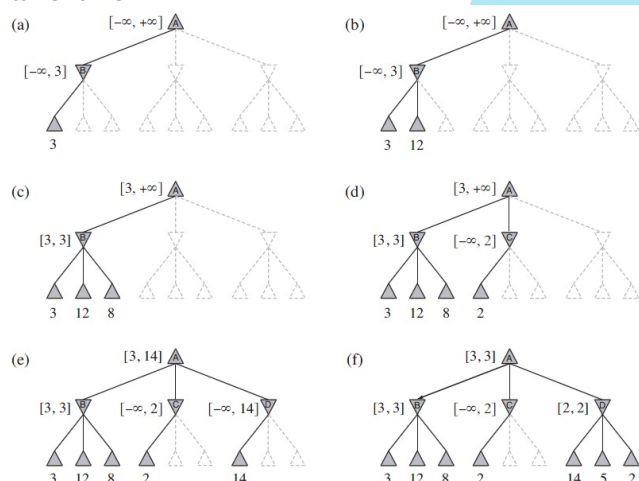
149

Cortes Alfa – Beta

α é o melhor valor (para Max) encontrado até agora no caminho corrente

Se V for pior do que α , Max deve evitá-lo \Rightarrow cortar o ramo

β é definido da mesma forma para Min



150

Cortes Alfa – Beta

Cortes Alfa-Beta não afetam o resultado final

Boa ordenação melhora a eficiência dos cortes

- Essencial raciocinar sobre a ordenação

Com ordenação perfeita: Complexidade no Tempo = $O(b^{m/2})$

- Duplica a profundidade de pesquisa
- Profundidade 8 => Bom jogador de Xadrez

Bom exemplo do valor de raciocinar sobre que computações são relevantes:

- Isto é essencial em Sistemas Inteligente/Sistemas Baseados em Conhecimento

151

Jogos Determinísticos

Damas:

Chinook acabou com o reinado de 40 anos do campeão humano Marion Tinsley em 1994. Usava uma base de dados para finais de partida definindo a forma perfeita de vencer para todas as posições envolvendo 8 ou menos peças (no total de 443748401247 posições).

Hoje em dia é um problema resolvido.

Xadrez:

Deep Blue derrotou o campeão do mundo humano Gary Kasparov num jogo com 6 partidas em 1997.

Pesquisa 200 milhões de posições por segundo e usa uma função de avaliação extremamente sofisticada e métodos para estender algumas linhas de pesquisa para além da profundidade 40!

152

Jogos Determinísticos

Othello:

Campeão do mundo Takeshi Murakami foi derrotado em 1997 pelo Logistello

Campeões humanos recusam-se a competir com computadores pois não têm qualquer hipótese! (b entre 5 e 15) -

Go:

Campeões humanos foram recentemente derrotados pelos computadores

153

Jogos de Azar

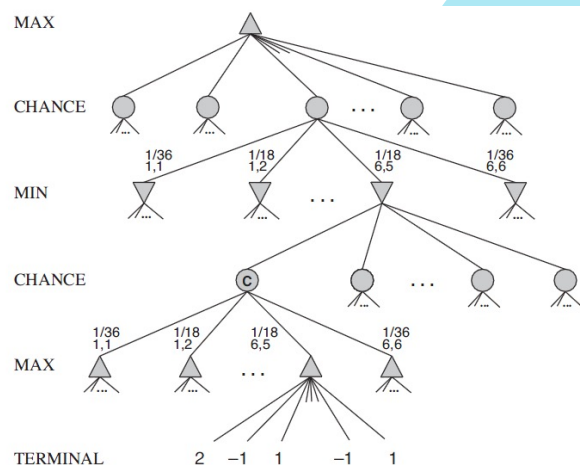
Em muitos jogos, ao contrário do xadrez, existem eventos externos que afetam o jogo, tais como tirar uma carta ou lançar um dado!

Exemplos: Jogos de cartas, Gamão, Scrabble, ...

Árvore de pesquisa deve incluir nós de probabilidade!

Decisão é efetuada com base no valor esperado!

ExpectiMiniMax



154

Jogos

Trabalhar com jogos é extremamente interessante

- Fácil testar novas ideias!
- Fácil comparar agentes com outros agentes
- Fácil comparar agentes com humanos!

Jogos ilustram diversos pontos interessantes da IA:

- Perfeição é inatingível => é necessário aproximar!
- É boa ideia pensar sobre o que pensar!
- Incerteza restringe a atribuição de valores aos estados!

Jogos são para a IA um tubo de ensaio

155

Exercícios - Jogos

Formular um dos seguintes jogos de tabuleiro para 2 jogadores como um jogo e projetar um agente inteligente capaz de o jogar:

1. Xadrez (Chess)
2. Shogi (Xadrez Japonês)
3. Damas (Checkers)
4. 4 em Linha (Connect 4)
5. Attaxx
6. Damas Chinesas
7. Othello (Reversi)
8. Abalone
9. Hex
10. Jogo do Galo 3D (4x4)
11. Diplomacy
12. Jogo do Futebol
13. Quarto (Gigamic)
14. Quixo (Gigamic)
15. Quoridor (Gigamic)
16. Sahara (Gigamic)
17. Pentaminós (8x8)
18. Estratégia (Stratego)
19. Link Five
20. Mancala
21. Fanorana
22. Nine Mens Morris
23. Alquerque

24. Tablut
25. Surakarta
26. Terrace
27. Go
28. Dots and Boxes
29. Dots and Hexagons
30. Amazons
31. Scrabble (pec. visíveis, letr.conh.)
32. Jogo do Galo (normal, memória e movimento)
33. Dominós (peças visíveis e sort. conhecido)
34. Gamão (sort. conhecido)
35. Paper & Pencil Racing
36. Arimaa
37. Gimp
38. Lines of Action
39. Mancala 4x8
40. Connections
41. Omega Chess
42. Tori Shogi
43. Fanorama
44. Hexxagon
45. Jungle Game (J.da Selva)
46. Seega (Tab. dim. Var.)

47. Halma (2 jog., Tab. dimensão variável)
48. Quits (Gigamic) (Tab. dimensão variável)
49. Pylos (Gigamic)
50. Tantrix (2 Jog., 7 Peças, 3 Cores)
51. Ticket to Ride (2 jog., ordem cartas conhecida)
52. Carcassone (2 jog. Sort. peças prévio)
53. Settlers of Catan (2 jog. sem trocas de recursos)
54. Blokus
55. Sputnik (Gigamic)
56. Katamino (Gigamic)
57. Gobblet (Gigamic)
58. Quads (Gigamic)
59. Quoridor Kid (Gigamic)

156

Exercícios

Implementar o algoritmo Minimax (sem e com cortes alfa-beta)

Descrever e/ou implementar descrições do estado, geradores de movimentos e funções de avaliação para os seguintes jogos:

Gamão, Monopólio, Scrabble.

Construir um agente capaz de jogar os jogo de cartas:

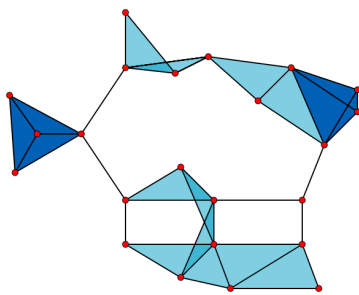
Hearts, King, Sobe e Desce, Poker, Lerpa e Bridge.

157



158

Algoritmos genéticos



159

Ideia

Natureza



Modelo
Biológico

Teorias Darwinianas

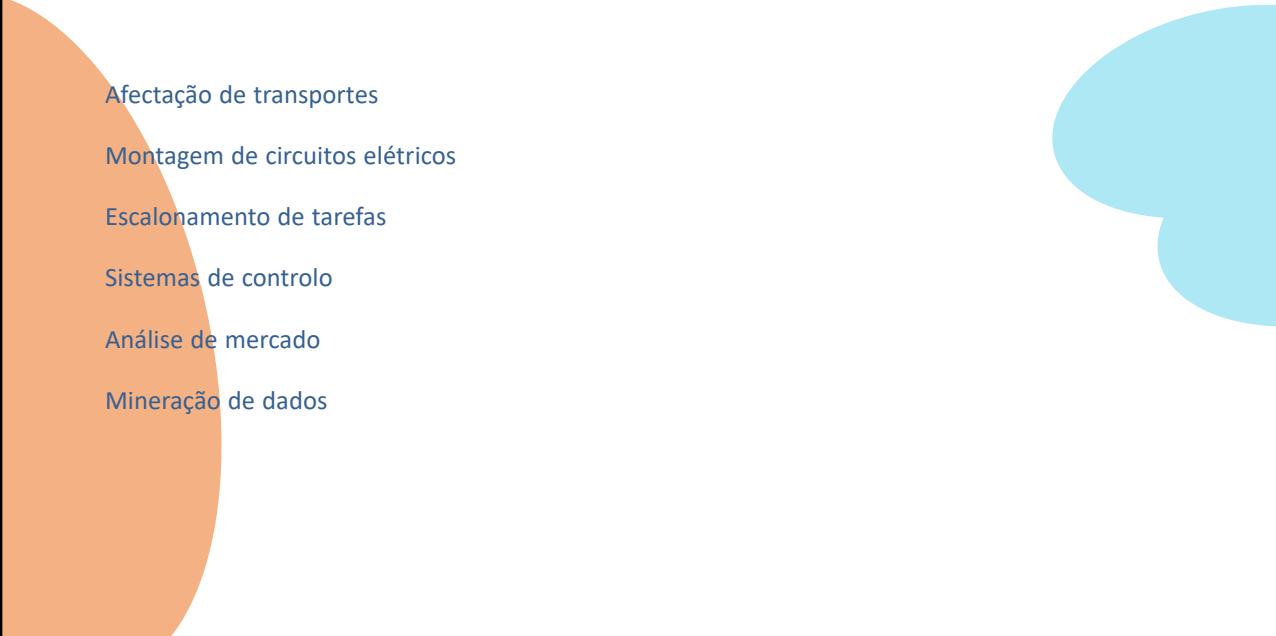


Teoria de Computação Evolutiva

Modelo
Computacional

160

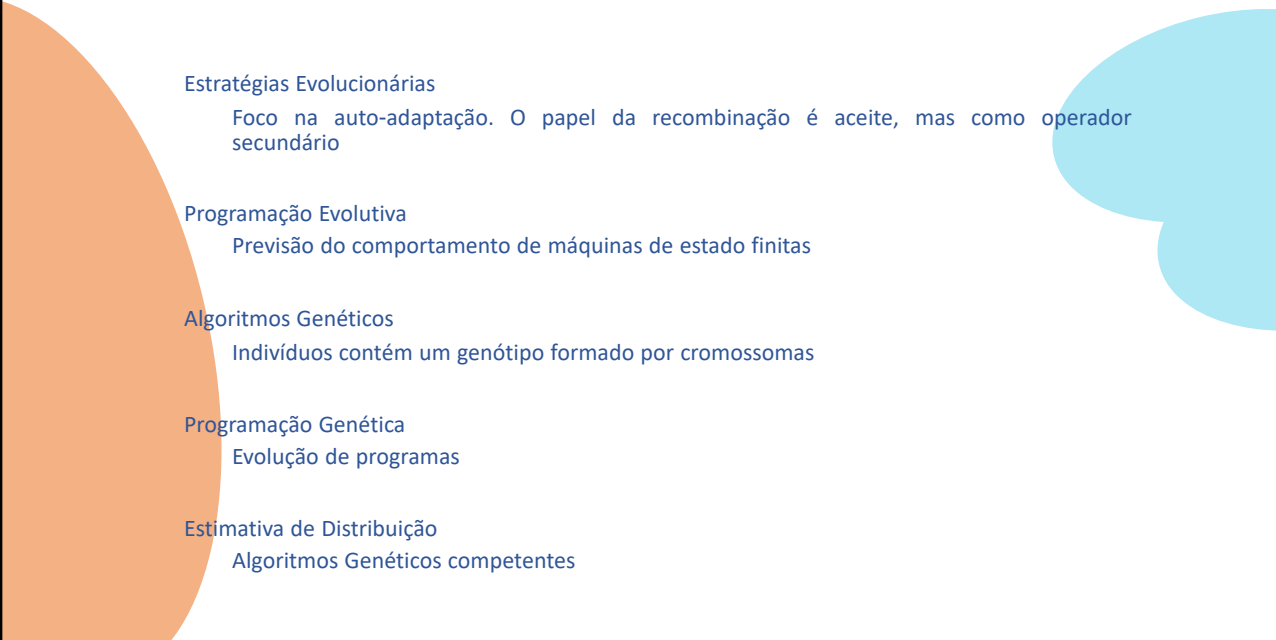
Exemplos de aplicação



Afectação de transportes
Montagem de circuitos eléctricos
Escalonamento de tarefas
Sistemas de controlo
Análise de mercado
Mineração de dados

161

Ramos



Estratégias Evolucionárias
Foco na auto-adaptação. O papel da recombinação é aceite, mas como operador secundário

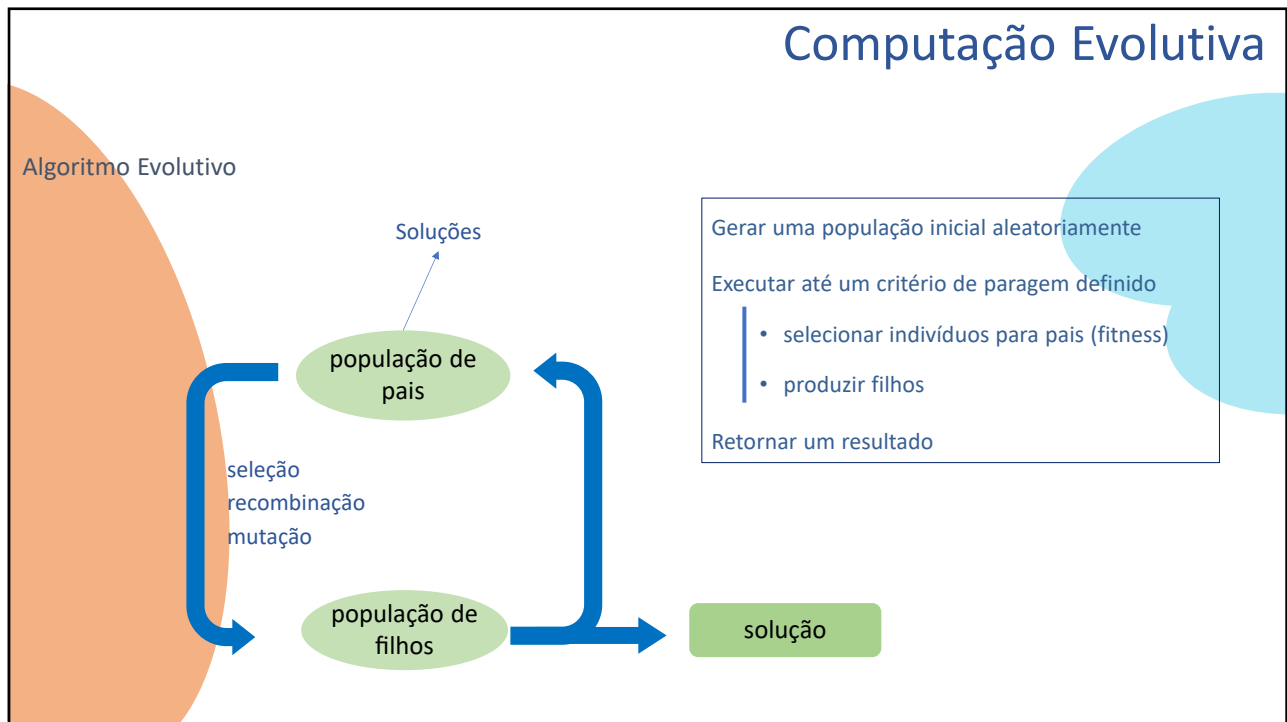
Programação Evolutiva
Previsão do comportamento de máquinas de estado finitas

Algoritmos Genéticos
Indivíduos contêm um genótipo formado por cromossomas

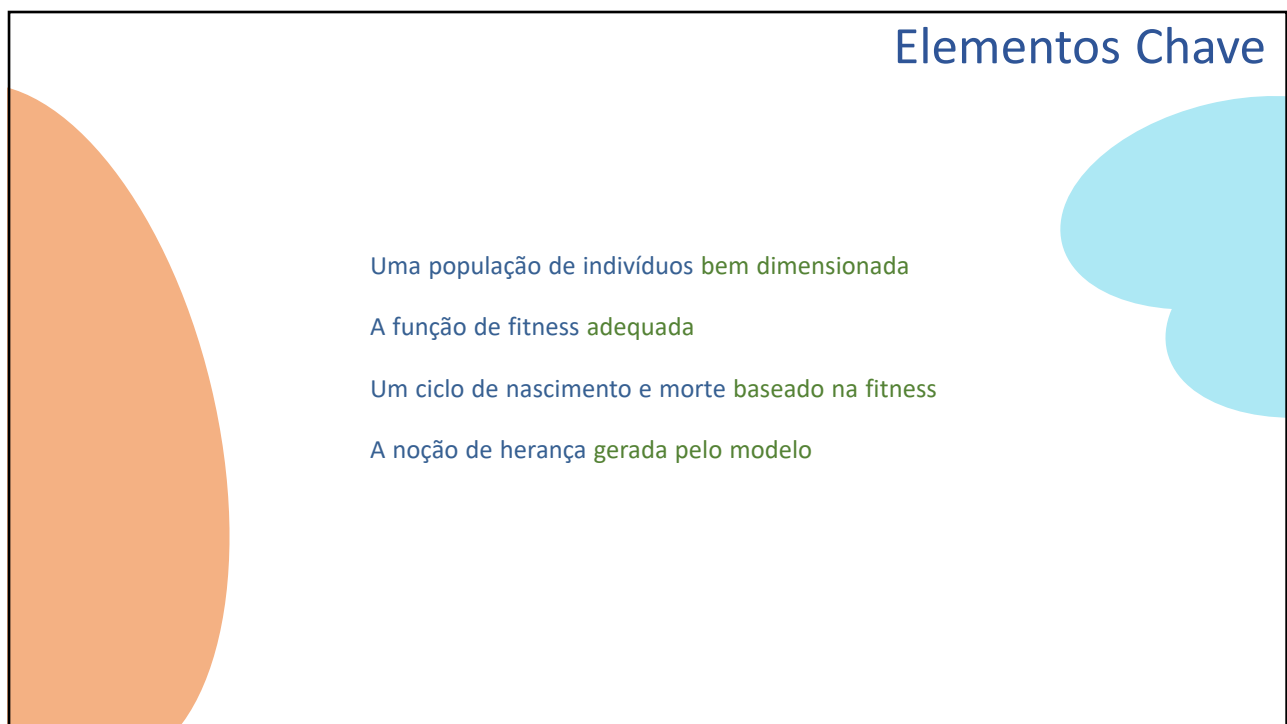
Programação Genética
Evolução de programas

Estimativa de Distribuição
Algoritmos Genéticos competentes

162



163



164

Algoritmos Genéticos

Método computacional com base na genética e na selecção natural.

É uma técnica baseada na evolução natural dos seres vivos.

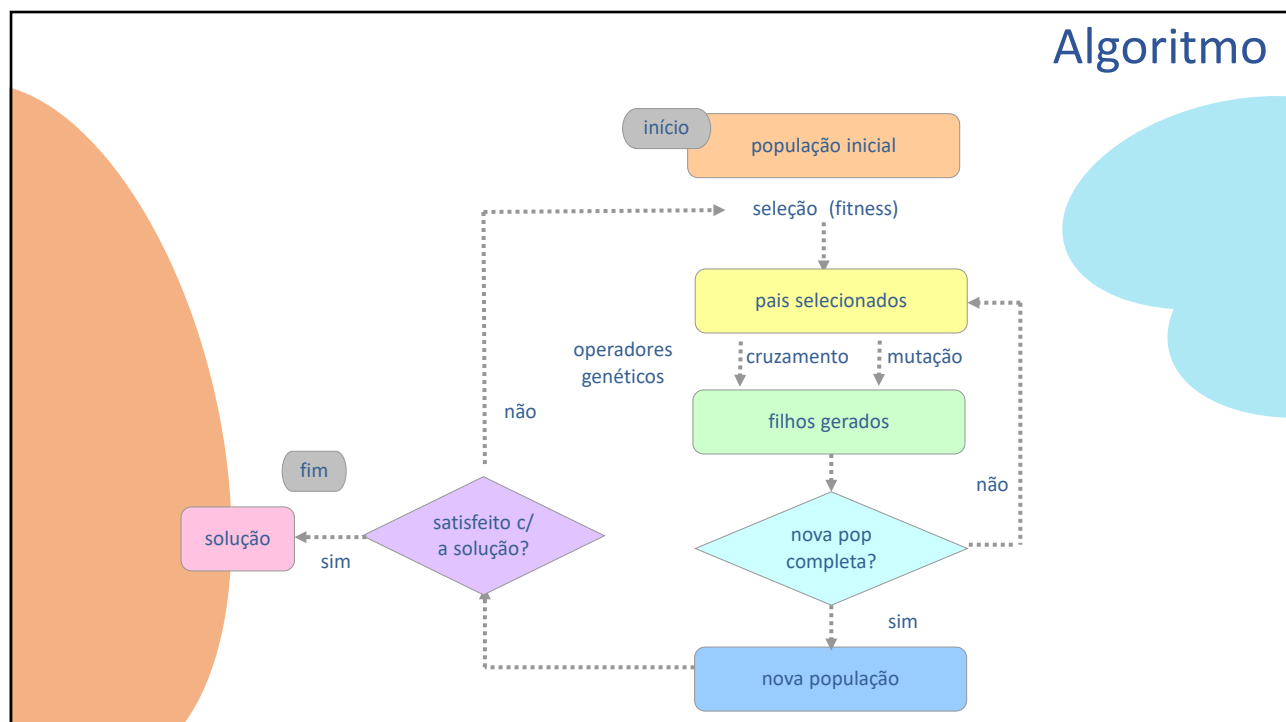
Segundo Charles Darwin, responsável pela teoria da evolução, os indivíduos que conseguem sobreviver as adversidades que encontram no ambiente que habitam, podem gerar indivíduos mais fortes, mais aptos. Apenas tais indivíduos, mais aptos, conseguem sobreviver.

A técnica do AG possui uma estrutura que parte de uma população inicial que pode ser dada de forma aleatória, sendo seleccionados de seguida.

É muito comum utilizar o AG para problemas em que o espaço de pesquisa seja muito grande ou quando é difícil dividir o problema em partes mais pequenas.

165

Algoritmo



166

Terminologia Biológica

Na área dos algoritmos genéticos são utilizados termos biológicos como analogia com a biologia

Cromossoma

Codificação de uma possível solução – indivíduo

Gene

Codifica uma característica particular

Genótipo

Informações do DNA do indivíduo – características de base

Fenótipo

Características visíveis do indivíduo depois da interação com o meio ambiente

167

Indivíduos

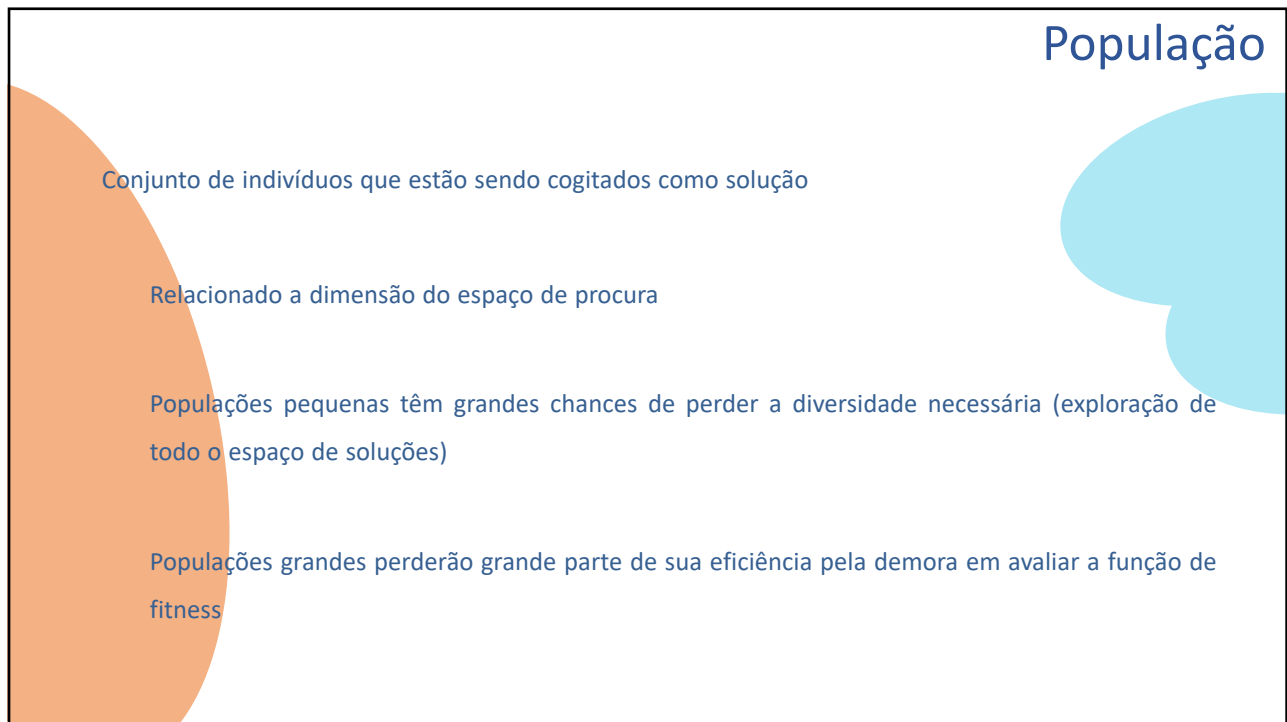
Material genético

Conjunto de atributos da solução

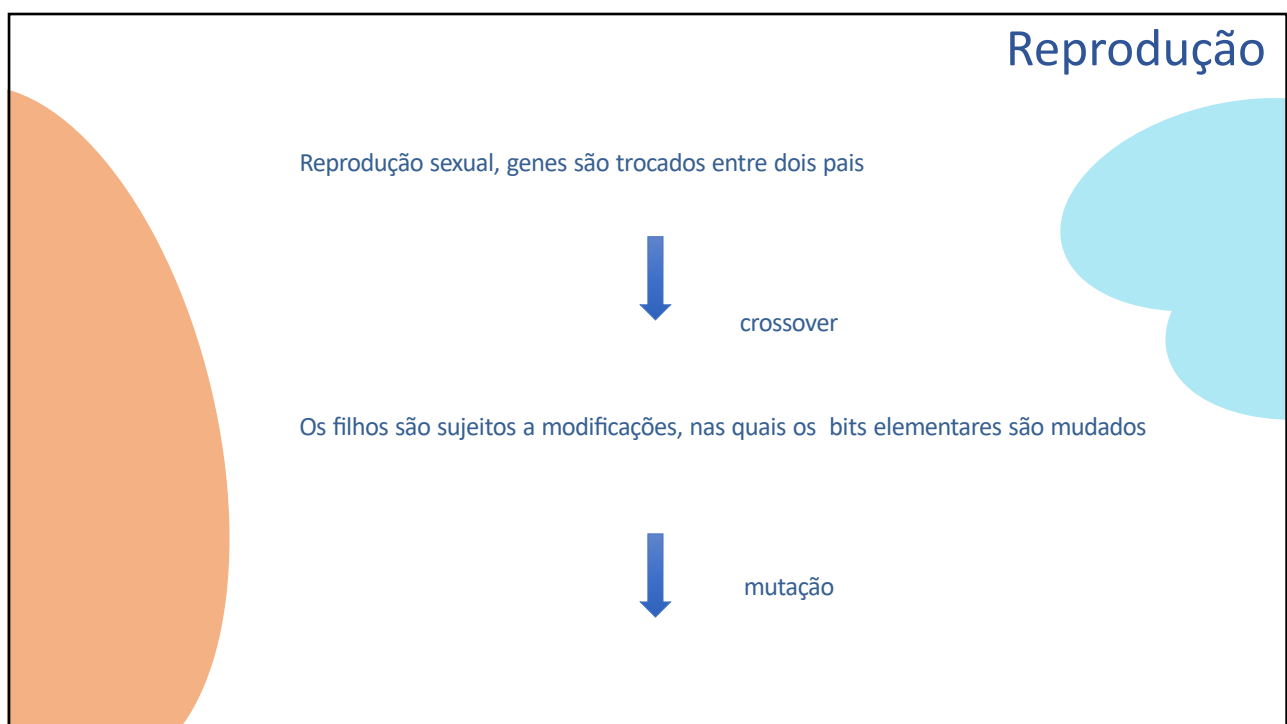
Cada atributo é classificado como uma sequência de bits e o indivíduo como a concatenação das sequências de bits

Codificação binária, códigos

168



169



170

Função de fitness

Mede a adaptação do indivíduo ou quão boa é a solução dada por este indivíduo.

Representativa do problema: diferencia uma solução boa de uma má.

Heurística de busca no espaço de estado

Cuidados com o custo computacional

Seleção

O operador escolhe quais indivíduos participarão na criação da próxima geração

Torneio | Roleta | ...

171

Requisitos para usar AG

Representações das possíveis soluções do problema no formato de um código genético

População inicial que contenha diversidade suficiente para permitir ao algoritmo combinar características e produzir novas soluções

Existência de um método para medir a qualidade de uma solução potencial

Um procedimento de combinação de soluções para gerar novos indivíduos na população

Um critério de escolha das soluções que permanecerão na população ou que serão retirados desta

Um procedimento para introduzir periodicamente alterações em algumas soluções da população.



Manter a diversidade da população e a possibilidade de se produzir soluções inovadoras

172

Métodos de selecção

Torneio

Seleccionar conjuntos de indivíduos (com cardinalidade a definir) e utilizar o melhor deles
A cardinalidade tem impacto na diversidade dos pais seleccionados

Roleta

Seleccionar conjuntos de indivíduos de acordo com uma função de distribuição.
A função de distribuição é baseada na função de fitness

173

Métodos de selecção

Seleccionar a melhor geração da população actual para progenitores

S	F(S)
1	14
2	17
3	17
4	11
5	8
6	23
7	22
8	20

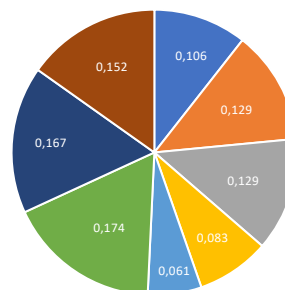
np=3

Método do torneio

- Definir número de participantes (np)
- Selecção aleatória dos participantes

Método da roleta

- Atribuir valor relativo da solução
- Selecção dos participantes de acordo com a distribuição de probabilidades



S	Prb(S)
1	0,106
2	0,129
3	0,129
4	0,083
5	0,061
6	0,174
7	0,167
8	0,152

174

Reprodução e métodos de combinação

Reprodução

Preserva características úteis
Introduz variedade e novidades

Estratégias

Parentes únicos: clonar + mutação
Parentes múltiplos: recombinação + mutação

Métodos de combinação

Cruzamento: cria novos indivíduos misturando características de dois indivíduos pais (crossover)

Cópia de segmentos entre os pais

Crossovers multi-ponto, dois pontos, um ponto, uniforme e inversão

175

Cruzamento e Mutação

Cruzamento

Progenitor 1: 101010101101010111

Progenitor 2: 00001001010101110010

Cruzamento num ponto

10101010110101110010

000010010101010111

Cruzamento uniforme: os filhos são formados a partir dos bits dos pais (sorteado)

Mutação

Esta operação inverte aleatoriamente alguma característica do indivíduo

Cria novas características que não existiam

Mantém diversidade na população

176

Quando AG é bom ?

Funções multimodas

Funções discretas ou contínuas

Funções altamente dimensionais, incluindo combinatórias

Dependência não linear dos parâmetros

Usada frequentemente para obter solução em problemas NP

Não usar quando outro método como hill-climbing, etc., funciona bem

AG's são apropriados para problemas complexos, mas algumas melhorias devem ser feitas

177

Exemplo

Problema da mochila

Exemplo:

$N = 8$

Capacidade (C) = 50

	1	2	3	4	5	6	7	8
Valor	4	3	6	7	2	9	7	6
Peso	12	16	8	21	16	11	6	12

Objectivo

Maximizar $\sum_i O_i \times v_i$

Restrição

$\sum_i O_i \times p_i \leq 50$

?

178

Exemplo

Problema da mochila

	1	2	3	4	5	6	7	8
Valor	4	3	6	7	2	9	7	6
Peso	12	16	8	21	16	11	6	12

Representação das soluções

1	1	0	0	0	0	1	0
---	---	---	---	---	---	---	---

Problema? ←

179

Soluções inválidas

Como resolver?

- Reparar
- Alterar a representação
- Penalizar

Uma função de penalização define em que quantidade a solução X viola a restrição R.

Mede a distância a que a solução está de uma região de aceitação

↓

Transforma um problema com restrições num problema sem restrições

180

Exemplo

Problema da mochila

	1	2	3	4	5	6	7	8
Valor	4	3	6	7	2	9	7	6
Peso	12	16	8	21	16	11	6	12

Avaliação

$$f(s) = \begin{cases} \sum_i o_i \times v_i & \text{se solução válida} \\ 0 & \text{se solução não válida} \end{cases}$$

181

Exemplo

Problema da mochila

	1	2	3	4	5	6	7	8
Valor	4	3	6	7	2	9	7	6
Peso	12	16	8	21	16	11	6	12

Avaliação $f(s) = \sum_i o_i \times v_i - p(s)$

Onde
$$p(s) = \begin{cases} \alpha \left(\sum_i o_i \times v_i \right) - C & \text{se solução não válida} \\ 0 & \text{se solução válida} \end{cases}$$

A penalização de uma solução válida é zero e é proporcional ao grau de violação das restrições

182

Exemplo

Problema da mochila

	1	2	3	4	5	6	7	8
Valor	4	3	6	7	2	9	7	6
Peso	12	16	8	21	16	11	6	12

Criar e avaliar a população inicial

Solução									Valor	Peso	p(s)	f(s)
1	1	0	0	0	0	1	0		14	34	0	14
0	0	0	1	1	0	1	1		22	55	5	17
0	1	0	1	0	0	1	0		17	43	0	17
1	0	0	0	0	0	1	0		11	18	0	11
0	1	1	1	1	0	1	0		25	67	17	8
1	0	0	1	0	0	1	1		24	51	1	23
0	1	0	1	0	1	1	0		26	54	4	22
1	1	0	0	0	0	1	1		20	46	0	20

183

Exemplo

Recombinar soluções

Combinar o material genético de dois progenitores para gerar novas soluções

Objectivo

Combinar características interessantes de duas soluções

Recombinação por ponto de corte

- Alinhar os dois progenitores
- Seleccionar um ponto de corte aleatório
- Combinar secções complementares para obter descendentes



184

Exemplo

Mutação

Alteração do valor de um gene (mutação binária)



Cria variabilidade no conjunto das soluções
Introduz alterações no material genético

Objectivo

Introduzir diversidade na população

185

Exemplo

População
inicial

Solução (progenitores)								Valor	Peso	p(s)	f(s)
1	1	0	0	0	0	1	0	14	34	0	14
0	0	0	1	1	0	1	1	22	55	5	17
0	1	0	1	0	0	1	0	17	43	0	17
1	0	0	0	0	0	1	0	11	18	0	11
0	1	1	1	1	0	1	0	25	67	17	8
1	0	0	1	0	0	1	1	24	51	1	23
0	1	0	1	0	1	1	0	26	54	4	22
1	1	0	0	0	0	1	1	20	46	0	20



Recombinação

Solução (filhos)								Valor	Peso	p(s)	f(s)
0	0	0	1	1	0	1	0	16	43	0	16
1	0	0	0	0	0	1	1	17	30	0	17
0	1	0	1	0	0	1	1	23	55	5	18
1	1	0	0	0	1	1	0	23	45	0	23

186

Exemplo

População
filhos

Solução (filhos)									Valor	Peso	p(s)	f(s)
0	0	0	1	1	0	1	0		16	43	0	16
1	0	0	0	0	0	1	1		17	30	0	17
0	1	0	1	0	0	1	1		23	55	5	18
1	1	0	0	0	1	1	0		23	45	0	23



Muta  o

Solu��o (filhos)									Valor	Peso	p(s)	f(s)
0	0	0	1	1	0	1	0		16	43	0	16
1	0	0	0	0	1	1	1		26	41	0	26
0	1	0	1	0	0	1	1		23	55	5	18
1	1	0	0	0	1	1	0		23	45	0	23

187



188