

Inteligência Artificial

Busca Informada e Heurísticas

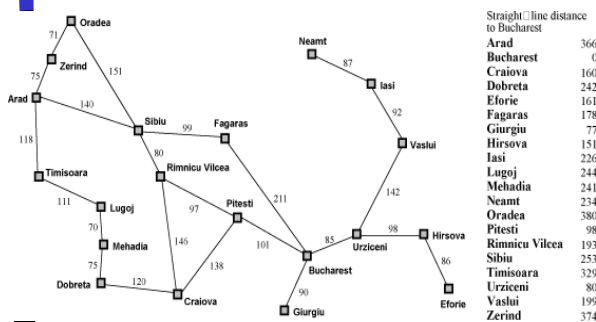
Busca Heurística - Informada

- Estratégias de Busca Heurística
 - Usam *conhecimento específico* do problema na busca da solução
 - Mais eficientes que busca não informada
- Algoritmo geral: **Busca pela Melhor Escolha BME (Best-first search)**
 - Seleciona para expansão o nó que tiver o mínimo custo estimado até a meta, segundo uma *função de avaliação* $f(n)$.
 - Tipicamente $f(n)$ usa uma *função heurística* $h'(n)$ que estima o custo da solução a partir de n . Na meta, $h'(n)=0$.

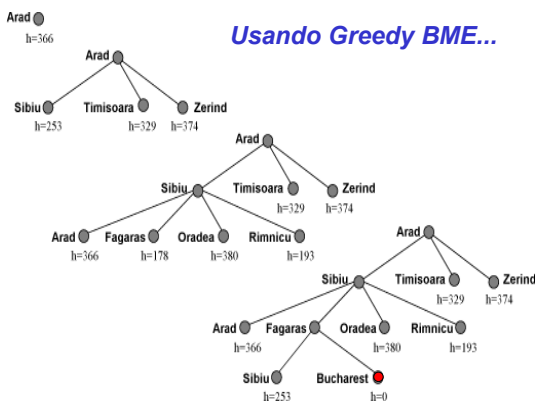
Greedy best-first search Busca gulosa pela melhor escolha

- Avalia nós para expandir com base unicamente na função heurística: $f(n) = h'(n)$
- Semelhante à busca em profundidade com retrocesso (*backtracking*)
- **Exercício:** encontrar a melhor rota (rota mais curta) de uma cidade a outra, num mapa.
 - $h'(n)$ = distância em linha reta entre as cidades e a cidade-meta
 - Note que esta distância não faz parte da descrição do problema (ela precisa ser calculada)

Exemplo: ir de Arad a Bucharest



Usando Greedy BME...



Desempenho da Greedy BME

- **Não é completa**
 - pode entrar em ciclos e não encontrar a solução se não detectar estados repetidos (idem BP)
 - pode se perder em um caminho infinito e nunca retroceder para tentar outras opções (idem BP)
 - **Não é ótima**
 - No ex: encontrou caminho (Arad, Sibiu, Fagaras, Bucharest) que é 32km maior que (Arad, Sibiu, Rimnicu Vilcea, Pitesti, Bucharest)
 - **Complexidade de tempo e espaço no pior caso: $O(b^m)$**
 - m é a máxima profundidade do espaço de busca
- Dependendo do problema e da qualidade da heurística a complexidade pode ser substancialmente reduzida.

BME mais "famoso": A*

■ **Função de avaliação:**

$$f(n) = g(n) + h'(n)$$

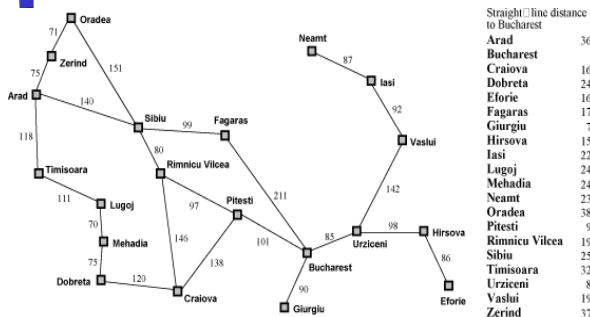
- $g(n)$ = distância (custo) do nó inicial ao nó n
- $h'(n)$ = distância (custo) **estimada** de n ao nó final
- Assim, $f(n)$ **estima** o custo da melhor solução que passa por n .

■ **A* expande o nó de menor valor de f na fronteira do espaço de estados.**

■ **Exercício:** repetir exercício anterior com A*.

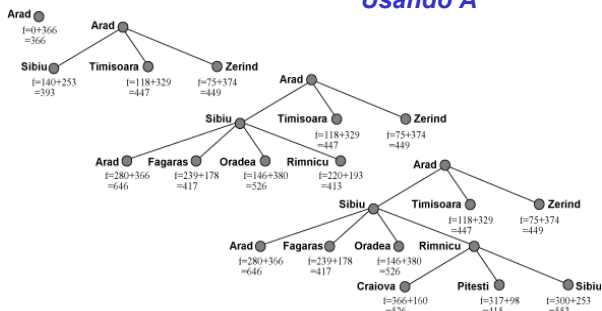
7

Exemplo: ir de Arad a Bucharest



8

Usando A*



Desempenho do A*

■ **A* é completo e ótimo se $h(n)$ for admissível e consistente**

- **h admissível:** nunca superestima o custo de atingir a meta (1)
- **h consistente (ou monotônica) (2)**

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

$$\text{ou } (h(n) - h(n')) \leq c(n, a, n')$$

- n' é sucessor de n , gerado pela ação a ; $c(n, a, n')$ é o custo de sair de n e atingir n' .
- Se h é consistente, os valores de $f(n)$ através de qualquer caminho são crescentes.

(1) TREE-SEARCH

(2) GRAPH-SEARCH (filtra nós repetidos)

- ou deve manter o caminho de menor custo que passa por um nó

10

Busca A* – comentários

- **A* é otimamente eficiente:** nenhum outro algoritmo ótimo garante expandir menos nós que A*.
- Infelizmente há, na maioria das vezes, crescimento exponencial do número de nós com o comprimento da solução (complexidade temporal).
- Mas o maior problema é a complexidade espacial: **A* armazena todos os nós gerados!**
- Assim, A* não é aplicável em muitos problemas de grande escala. Usa-se variantes que encontram soluções **subótimas**.

11

Busca Heurística com Memória Limitada

■ **IDA* (Iterative Deepening A*)**

- igual ao aprofundamento iterativo, porém seu limite é dado pela função de avaliação (f), e não pela profundidade (d).
- necessita de menos memória do que A*

■ **SMA* (Simplified Memory-Bounded A*)**

- O número de nós guardados em memória é fixado previamente

12

Críticas à Busca Heurística

- **Solução de problemas usando técnicas de busca heurística:**
 - dificuldades em definir e usar a *função de avaliação*
 - não consideram conhecimento genérico do mundo (ou "senso comum")
- **Função de avaliação: compromisso (conflito) entre**
 - tempo gasto na seleção de um nó (computar h) e
 - redução do espaço de busca
- **Achar o melhor nó a ser expandido a cada passo pode ser tão difícil quanto o problema da busca em geral.**

13

RESUMO DAS BUSCAS

14

Estratégias de Busca não Informada

- **Busca em Largura:** Fronteira = FIFO (fila)
- **Busca de Custo Uniforme:** expande menor $g(n)$
- **Busca em Profundidade:** Fronteira = LIFO (pilha), com retrocesso
- **Busca em Profundidade Limitada:** impõe limite máximo (l)
- **Busca em Profundidade com Aprofundamento Iterativo:** usa limite máximo variável (l)
- **Busca Bidirecional:** busca simultânea nos dois sentidos
- **Busca com Conhecimento Incompleto**
 - Conformantes: sem sensores
 - Contingenciais: incerteza e observabilidade parcial; caso extremo: problemas de exploração

15

Busca Informada

- **Algoritmo básico: Busca pela Melhor Escolha BME (Best-first search) – usa $f(n)$**
- **Greedy best-first search: $f(n) = h'(n)$**
- **A*: $f(n) = g(n) + h'(n)$**

16

HEURÍSTICAS – COMO DEFINIR?

17

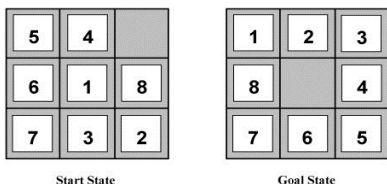
Inventando Funções Heurísticas

- **Como escolher uma boa função heurística h ?**
- **h depende de cada problema particular.**
- **h deve ser *admissível***
 - não superestimar o custo real da solução
- **Exemplo: jogo dos 8 números**
 - um número pode mover-se de A para B se A é adjacente a B e B está vazio
 - busca exaustiva:
 - solução média em 22 passos
 - fator de ramificação médio: 3
 - Assim, $\approx 3^{22}$ estados possíveis

4	5	8
	1	6
7	2	3

18

Heurísticas para jogo 8 números



Algumas heurísticas possíveis:

$h1$ = no. de elementos fora do lugar ($h1=7$)

$h2$ = soma das distâncias de cada número à posição final

($h2=2+3+3+2+4+2+0+2=18$) (distância Manhattan)

Exercício: resolver com A^* e $h1$ e $h2$ e comparar soluções (10 ações).

19

Qualidade da função heurística

■ **Qualidade da função heurística:** medida através do fator de expansão efetivo (b^*).

- b^* é o fator de expansão de uma árvore uniforme com $N+1$ nós e nível de profundidade d

- $N+1 = 1 + b^* + (b^*)^2 + \dots + (b^*)^d$, onde
 N = total de nós gerados pelo A^* para um problema
 d = profundidade da solução;
 Ex: $N = 52, d = 5, b^* = 1.92$

■ **Mede-se empiricamente a qualidade de h a partir do conjunto de valores experimentais de N e d .**

- uma boa função heurística terá o b^* muito próximo de 1.

■ **Se o custo de execução da função heurística for maior do que expandir nós, então ela não deve ser usada.**

- uma boa função heurística deve ser *eficiente* e *econômica*.

20

Experimento (Média de 100 soluções/dado)

d	Número médio de nós expandidos (100 épocas)			Effective Branching Factor b^*		
	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	—	1301	211	—	1.45	1.25
18	—	3056	363	—	1.46	1.26
20	—	7276	676	—	1.47	1.27
22	—	18094	1219	—	1.48	1.28
24	—	39135	1641	—	1.48	1.26

IDS: Iterative-Deepening-Search (Busca de Aprofundamento Iterativo)

Uma boa função heurística terá o b^* muito próximo de 1.

Note que $h2$ é melhor que $h1 \rightarrow$ de fato, para qualquer nó, $h2(n) \geq h1(n)$!

21

Escolhendo Funções Heurísticas

■ **É sempre melhor usar uma função heurística com valores mais altos, contanto que ela seja *admissível* e que o tempo para computá-la não seja muito grande!**

- ex. $h2$ melhor que $h1$.

■ **h_1 domina $h_k \Rightarrow h_1(n) \geq h_k(n) \forall n$ no espaço de estados**

- h_2 domina h_1 no exemplo anterior

■ **Caso existam muitas funções heurísticas para o mesmo problema, e nenhuma delas domine as outras, usa-se uma *heurística composta*:**

- $h(n) = \max(h_1(n), h_2(n), \dots, h_m(n))$
- Assim definida, h é *admissível* e *domina* cada função h_i individualmente

22

Como inventar funções heurísticas admissíveis?

■ **Existem estratégias genéricas para definir h :**

- 1) Relaxar o problema (versão simplificada);
- 2) Usar informação estatística;
- 3) Identificar os atributos relevantes do problema e usar aprendizagem.

23

Relaxando o problema - exemplo

■ **Problema Relaxado:**

- versão simplificada do problema original, onde os operadores são menos restritivos

■ **Descrição original da ação:**

Peça A pode mover do lugar A para o lugar B se A é adjacente (horizontal ou vertical) a B e B é lugar vazio.

■ **Problemas relaxados:**

- Peças podem se mover de A para B mesmo se estes não forem adjacentes e/ou B estiver vazio

24

(1) Relaxando o problema

■ Operadores relaxados:

1. Uma peça pode se mover para lugares adjacentes, mesmo que ocupados
 - h_2 seria o custo da solução “correta” neste jogo
2. Uma peça pode se mover para qualquer lugar vazio, mesmo que não adjacente
 - h ? (Ex. 4.9 do livro)
3. Uma peça pode se mover para qualquer lugar
 - h_1 seria o custo da solução “correta” neste jogo

O custo da solução ótima de um problema relaxado é uma heurística admissível para o problema original!!!

25

(2) Usando informação estatística

■ Funções heurísticas podem ser “melhoradas” com informação estatística:

- executar a busca com um conjunto de treinamento (e.g., 100 configurações diferentes do jogo), e computar os resultados.
 - se, em 90% dos casos, quando $h(n) = 14$, a distância real da solução é 18, então, quando o algoritmo encontrar 14 para o resultado da função, vai substituir esse valor por 18.

■ Informação estatística expande menos nós, porém elimina **admissibilidade**:

- em 10% dos casos do problema acima, a **função de avaliação poderá superestimar o custo da solução**, não sendo de grande auxílio para o algoritmo encontrar a solução menos custosa.

26

(3) Aprendendo heurísticas por experiência

■ Resolve o jogo diversas vezes e computa custo da solução, relacionando a algum atributo relevante do problema.

- Ex1: atributo $x_1(n)$ = número de peças fora do lugar no início do jogo; Para cada valor de atributo, ver custo médio da solução.
 - Ex: para $x_1(n)=5$, resolvo 100 vezes o problema e vejo que custo médio da solução é 14 passos
- Ex2: atributo $x_2(n)$ = número de pares de peças adjacentes que também são adjacentes na configuração de solução

■ → uso $x_1(n)$ ou $x_2(n)$ para estimar $h(n)$ ou a combinação entre elas: $h(n) = c_1 \cdot x_1(n) + c_2 \cdot x_2(n)$, ajustando c_1 e c_2 da melhor forma para os dados de custo da solução.

27