1

Métodos de Busca Informada

2

Busca não informada: geração sistemática de estados

- **Busca em profundidade**:
  - boa quando não se deseja encontrar a solução ótima (em especial se existem múltiplas soluções); economia de espaço.
  - ruim se não existir um goal naquela sub-árvore; se compromete muito com um ramo específico da árvore.
- **Busca em largura**:
  - não se compromete com um ramo específico mas consome muita memória e tempo.
- **Não sistematicidade**
  - sistematicidade: nunca visita um nó duas vezes.
  - completeza: visita todos os nós (garantia de se encontrar uma solução, se ela existir).
  - "pular de galho em galho": não completo e não sistemático .

3

Questão

- Como poderiamos modificar os algoritmos gerais de busca para incluir mais conhecimento sobre o problema?

4

Busca informada*: Best-first Search*

- procedimento não uniforme => prefere selecionar nós que "prometem" dirigir a busca mais rapidamente ao goal.
- usa informação específica do domínio que indique qual "parece" ser o melhor caminho para se atingir o goal, ou seja, o próximo nó a ser expandido.
- buscas anteriores só usavam conhecimento do domínio (modelo de ações) na função de expansão.

5

Best-first Search

- Conhecimento específico do problema => função de avaliação *f(n)*:
  - recebe descrições de estados e devolve um real.
  - deve incorporar alguma estimativa do custo do caminho entre o nó corrente e o estado goal.
  - convenção: valores menores de *f(n)* indicam os melhores nós.
- *Best-first search*
  - introduz conhecimento na estratégia geral de busca (QUEUING-FN)
  - expande primeiro o nó "aparentemente melhor" (o nó que possue a melhor avaliação).
  - termina quando o nó a ser expandido for o nó goal.

6

Best-First Search

função *BestFirstSearch*(*problema*, *estratégia*) **devolve** uma solução, ou falha

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

laço faça

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

escolha o primeiro nó da lista de nós terminais

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

senão expanda o nó de acordo com a *estratégia*

fim

Obs: *estratégia* = QUEUING-FN (ordenação da fila de acordo com uma função de avaliação)

7

Questão

- *Best-First Search* expande sempre o melhor nó primeiro?

8

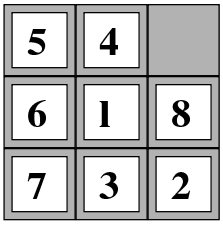
Algorítmos BFS

- Busca de Custo Uniforme
- Greedy Search
- A*
- IDA*

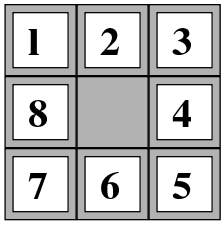
9

Função de avaliação para o 8-puzzle

Quais seriam algumas funções de avaliação para o 8-puzzle ?



Estado Corrente

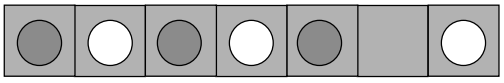


Estado goal

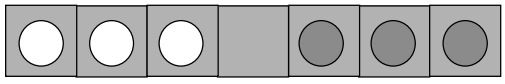
10

Função de avaliação para o régua-puzzle

Quais seriam algumas funções de avaliação para o régua-puzzle ?



Estado corrente



Estado goal

11

Questão

Qual é a diferença entre os métodos:
busca uniforme e *best-first search*?

12

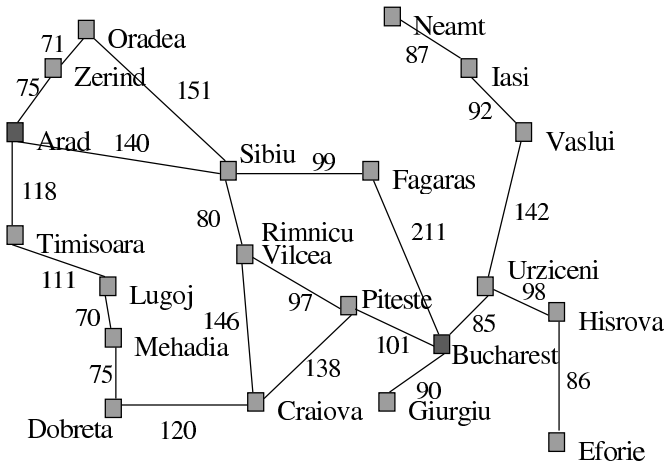
Funções de avaliação - custo uniforme

- Podem incorporar informação sobre o custo da solução
- Exemplo: Busca de Custo Uniforme
  - $f(n) = g(n)$: estimativa do custo do caminho
  - não dirige a busca para a solução

Função heurística - *h(n)*

- *f(n) = h(n)* ⟶
  - estimativa do custo do caminho que leva a solução
  - *h(n) = 0* se *n* é o goal
- constrói-se uma *h(n)* específica para o problema

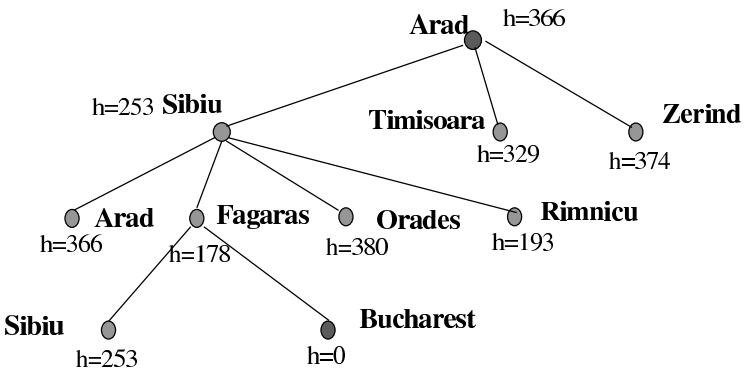
Exemplo: problema de roteamento



Arad	**366**
Bucharest	**0**
Craiova	**160**
Dobreta	**242**
Eforie	**161**
Fagaras	**178**
Giurgiu	**77**
Hirsova	**151**
Iasi	**226**
Lugoj	**244**
Mehadia	**241**
Neamt	**234**
Oradea	**380**
Pitesti	**98**
Rimnicu Vilcea	**193**
Sibiu	**253**
Timisoara	**329**

15

Espaço de busca



h(n): distância ao goal em linha reta

16

Greedy Search

- algoritmo *Best-first Search* que usa *h* para selecionar nós a serem expandidos
- minimiza o custo estimado para se atingir o estado goal (*h*)
- tenta encontrar uma solução rápida mas que nem sempre é a ótima (algoritmo não ótimo)
- como o *DFS:* não completo
- como na *BL* guarda todos os nós gerados
- complexidade de tempo e espaço $O(b^m)$ - pode ser reduzida com uma boa função heurística (depende fortemente da heurística)

17

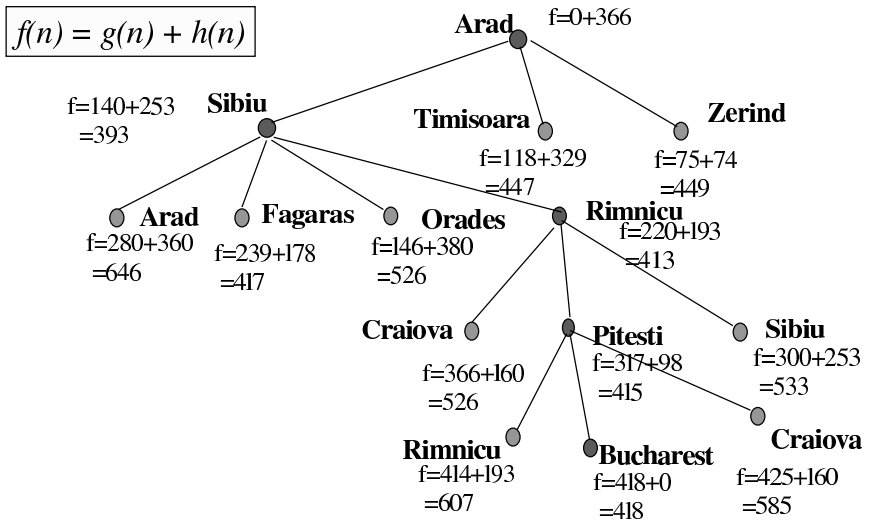
Busca A* - minimiza o custo total

- Combina busca de custo uniforme com busca heurística
- $f(n) = g(n) + h(n)$
  - $g(n)$ = custo gasto até n
  - $h(n)$ = custo estimado para atingir o goal a partir de n
  - $f(n)$ = custo estimado total do caminho que passa por n para atingir o goal
- A* usa uma heurística admissível:
  - h não deve superestimar o custo real para se alcançar a solução: *heurística admissível* (otimista). Ex.: da distância em linha reta.

$h(n) \leq h^*(n)$ ***onde*** $h^*(n)$ *é* o custo verdadeiro a partir de n

18

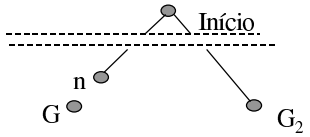
Espaço de busca



Busca A* - generalização

g(n)	h(n)	características
1	0	*BrFS*
custo	0	*Uniform Cost S.*
0	0	*random search*
g(n)	admissível	busca ótima de acordo com g(n)

A* encontra a solução ótima



A* encontra a melhor solução - uma prova por contradição.

• seja G o estado goal ótimo e G_2 o estado goal sub-ótimo

$$f(G) = g(G) + 0 = f^*$$

• Supondo que A* termina no estado goal G_2 (sub-ótimo), então $f(G_2) = g(G_2) + h(G_2) = g(G_2) > f^*$ (vamos mostrar que é impossível)

• Seja n o nó folha no caminho de G. Supondo que n não é escolhido quando comparado com G_2, logo:

(1) sendo h admissível: $f^* \geq f(n)$

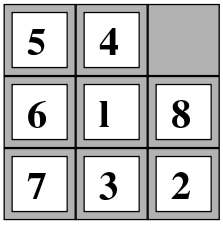
(2) n não é expandido: $f(n) \geq f(G_2)$

de (1) e (2): $f^* \geq f(G_2)$

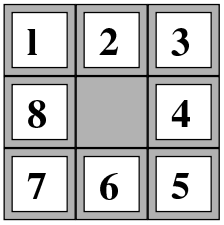
como $f(G_2) = g(G_2)$: $f^* \geq g(G_2)$ *contradição*

Heurísticas para o 8-puzzle

- solução típica possue 20 passos
- fator de ramificação: 3 ou 4 $\Rightarrow \approx 3^{20} \Rightarrow \approx 3.5\,10^9$
- função heurística que nunca superestima a distância ao goal ?



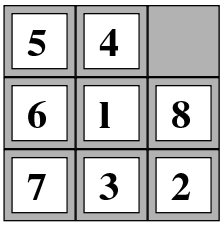
Estado Inicial



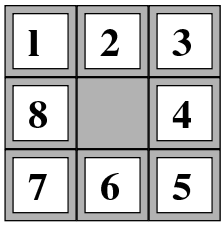
Estado goal

8-puzzle: Heurísticas admissíveis

$h_1(n)$ = número de pastilhas no lugar errado
$h_2(n)$ = total das distâncias de Manhattan



Estado Inicial

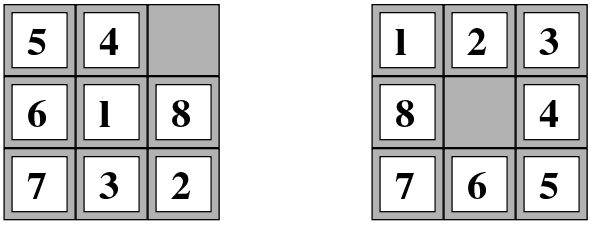


Estado goal

$h_1(I)$ = ?
$h_2(I)$ = ?

8-puzzle: Heurísticas admissíveis

$h_1(n)$ = número de pastilhas no lugar errado (admissível)
$h_2(n)$ = total das distâncias de Manhattan (admissível)



Estado Inicial Estado goal

$h_1(I)$ = 7
$h_2(I)$ = 2 + 3 + 3 + 2 + 4 + 2 + 0 + 2 = 18

Teste de função heurística

Se $h_2(n) \geq h_1(n)$ para todo n

então h_2 domina h_1 e é melhor para a busca (expande nós onde $f(n) < f^*$, *ou seja* $h(n) < f^* - g(n)$)

Custo típico :

d = 14 IDS = 3.473.941 nós
A* (h_1) = 539 nós
A* (h_2) = 113 nós

d = 24 IDS = muitos nós
A* (h_1) = 39.135 nós
A* (h_2) = 1.641 nós

25

Como "inventar" uma boa heurística?

Heurísticas admissíveis podem ser derivadas a partir do custo da solução exata de uma versão relaxada do problema (menos restrita)

- **Versão 1**: as pastilhas podem se mover para qualquer casa $\Rightarrow$ $h_1(n)$ é melhor
- **Versão 2**: as pastilhas podem se mover para qualquer casa adjacente (incluindo casas ocupadas) $\Rightarrow$ $h_2(n)$ é melhor

26

Geração automática de heurísticas

- Problema descrito em linguagem formal
- Exemplo de três problemas relaxados com a remoção de 1 ou mais condições:
  (a) mover de A para B ***se*** A é adjacente a B
  (b) mover de A para B ***se*** B está vazio
  (c) mover de A para B
- ABSOLVER gerou a melhor heurística para o 8-puzzle e a primeira heurística útil para o Cubo Mágico

27

A melhor heurística

- Heurísticas admissíveis:

  $h_1(n), \ldots, h_m(n)$

- Heurística composta

  $h(n) = max(h_1(n), \ldots, h_m(n))$

- Análise estatística da execução do método com diferentes heurísticas para um número grande de casos

28

Custo da função heurística

- Suposição: custo de se calcular *h(n)* = ao custo de se expandir um nó
- nem sempre minimizar o número de nós a serem expandidos com uma boa heurística é uma boa coisa ==> o cálculo de *h(n)* pode ser equivalente a expandir centenas de nós
- uma heurística precisa: heurística que faz busca em largura "*on the sly*"!!
- Uma boa heurística deve ser eficiente e precisa!