Introdução a Inteligência Artificial

Aula 5

Prof. Lucas Cambuim

Busca Heurística

Capítulo 3 – Russell & Norvig

Estratégias de Busca Exaustiva (Cega)

- Encontram soluções para problemas pela geração *sistemática* de novos estados, que são comparados ao objetivo;
- São *ineficientes* na maioria dos casos:
  - utilizam apenas o *custo de caminho* do nó atual ao nó inicial (função *g*) para decidir qual o próximo nó da fronteira a ser expandido.
  - essa medida nem sempre conduz a busca na direção do objetivo.
- Como encontrar um barco perdido?
  - não podemos procurar no oceano inteiro...
  - observamos as correntes marítimas, o vento, etc...

Estratégias de busca heurística (Informada)

- Utilizam conhecimento específico do problema na escolha do próximo nó a ser expandido
  - barco perdido: correntes maritimas, vento, etc
- Aplicação de uma função de avaliação a cada nó na fronteira do espaço de estados
  - essa função ESTIMA o custo de caminho do nó atual até o objetivo mais próximo utilizando uma FUNÇÃO HEURÍSTICA.

Busca Heurística

- Classes de algoritmos para busca heurística:
  1. Busca pela melhor escolha (*Best-First search*)
  2. Busca com limite de memória
  3. Busca com melhora iterativa

Busca pela Melhor Escolha

Best-First Search

- Busca pela Melhor Escolha - BME
  - Busca genérica onde o nó de menor custo "aparente" na fronteira do espaço de estados é expandido primeiro
- Duas abordagens básicas:
  - 1. Busca Gulosa (Greedy search)
  - 2. Algoritmo A*

Busca pela Melhor Escolha

Algoritmo geral

- Função-Insere
  - insere novos nós na fronteira ordenados com base na Função-Avaliação
    - Que está baseada na função heurística

função Busca-Melhor-Escolha **(***problema*, *Função-Avaliação***)**

Busca-Genérica (*problema, Função-Insere*)

retorna **uma solução**

BME: Busca Gulosa

- Semelhante à busca em profundidade com *backtracking*
- Tenta expandir o nó mais próximo do nó final com base na estimativa feita pela função heurística *h*
- *Função-Avaliação*
  - função heurística *h*

Funções Heurísticas

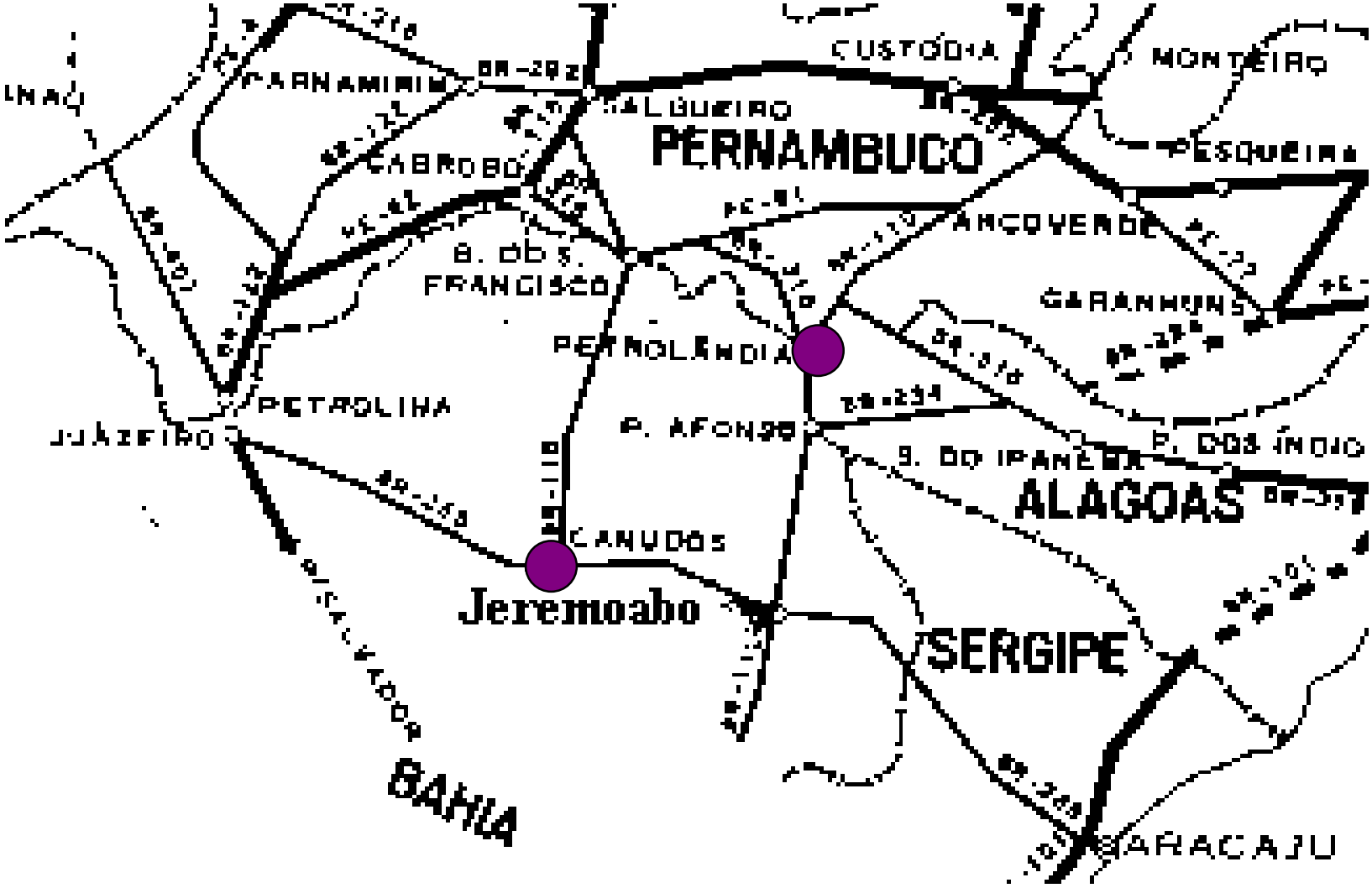
- Função heurística - *h*
  - estima o custo do caminho mais barato do estado atual até o estado final mais próximo.
- Funções heurísticas são específicas para cada problema
- Exemplo:
  - encontrar a rota mais barata de Canudos a Petrolândia
  - $h_{dd}(n)$ = distância direta entre o nó *n* e o nó final.

Funções Heurísticas

- Como escolher uma boa função heurística?
  - ela deve ser admissível
  - i.e., nunca *superestimar* o custo real da solução
- Distância direta (h_{dd}) é *admissível* porque o caminho mais curto entre dois pontos é sempre uma linha reta

Exemplo: encontrar a rota mais barata de Canudos a Petrolândia

hdd(n) = distância direta entre o nó *n* e o nó final



Busca Gulosa

- Custo de busca mínimo!
  - não expande nós fora do caminho
- Porém *não* é ótima:
  - escolhe o caminho que é mais econômico à primeira vista
    - Belém do S. Francisco, Petrolândia = 4,4 unidades
  - porém, existe um caminho mais curto de Canudos a Petrolândia
    - Jeremoabo, P. Afonso, Petrolândia = 4 unidades
- A solução via Belém do S. Francisco foi escolhida por este algoritmo porque
  - $h_{dd}(BSF) = 1,5$ *u.,* enquanto $h_{dd}(Jer) = 2,1$ u.

Busca Gulosa

- Não é completa:
  - pode entrar em *looping* se não detectar a expansão de estados repetidos
  - pode tentar desenvolver um caminho infinito

- Custo de tempo e memória: $O(b^d)$
  - guarda todos os nós expandidos na memória

BME: Algoritmo A*

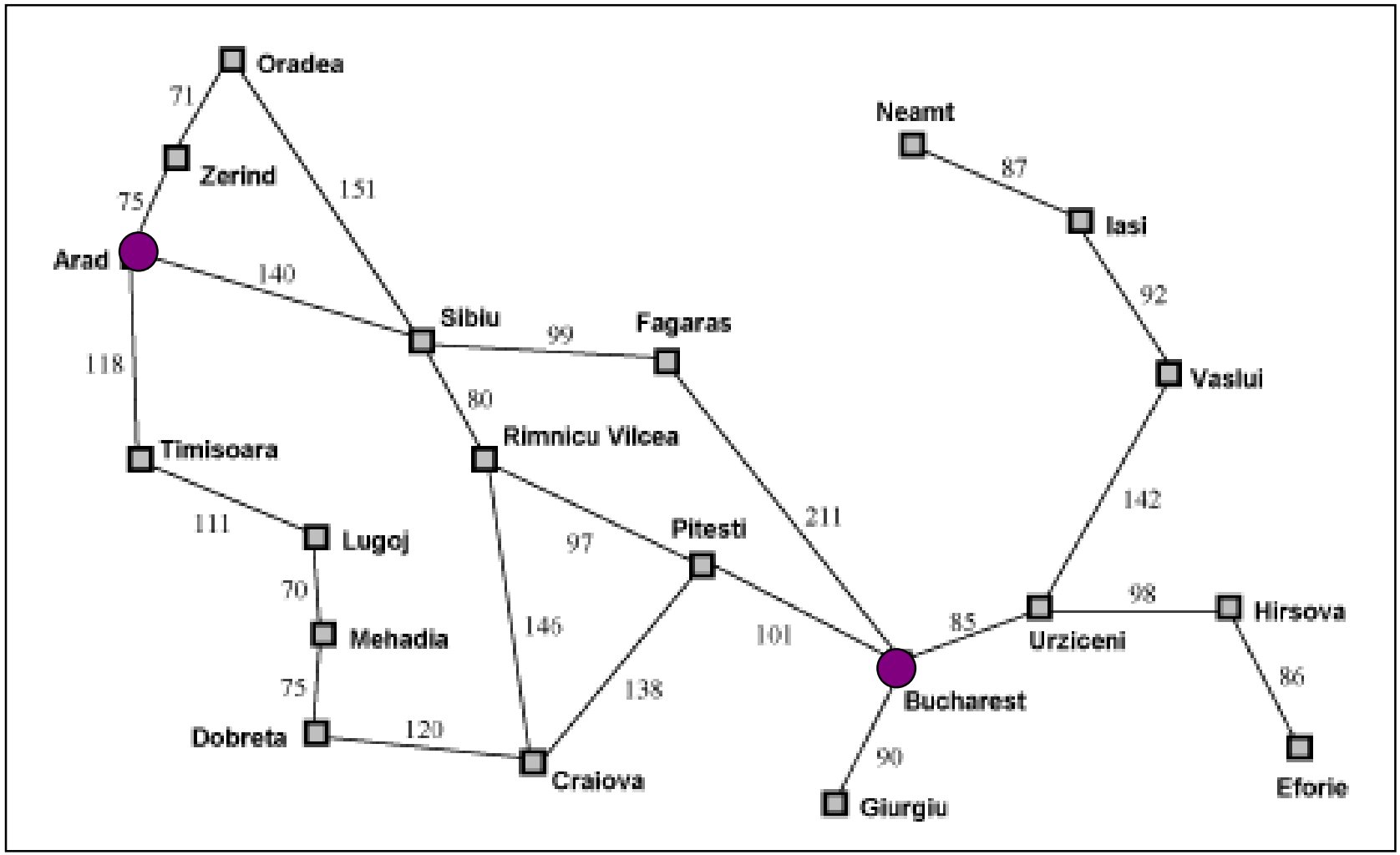
- A* expande o nó de menor valor de *f* na fronteira do espaço de estados
- Tenta minimizar o custo total da solução combinando:
  - Busca Gulosa (h)
    - econômica, porém não é completa nem ótima
  - Busca de Custo Uniforme (g)
    - ineficiente, porém completa e ótima
- *f* - Função de avaliação do A*:
  - *f (n) = g (n) + h (n)*
  - *g (n)* = distância de *n* ao nó inicial
  - *h (n)* = distância estimada de *n* ao nó final

Algoritmo A*

- Se *h* é *admissível*, então *f (n)* é admissível também
  - i.e., *f* nunca irá superestimar o custo real da melhor solução através de *n*
  - pois *g* guarda o valor exato do caminho já percorrido.
- Com A*, a rota escolhida entre *Canudos* e *Petrolândia* é de fato a mais curta, uma vez que:
  - *f (BSF) = 2,5 u + 1,5 u = 4 u*
  - *f (Jeremoabo) = 1,5 u + 2,1 u = 3,6 u*

Algoritmo A*: outro exemplo

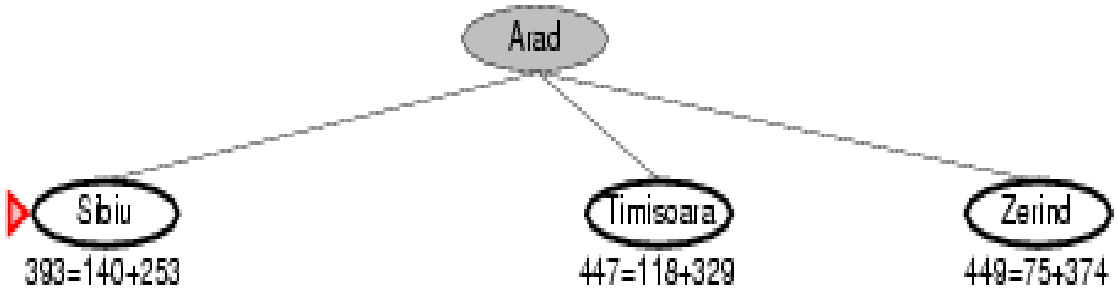
Viajar de Arad a Bucharest



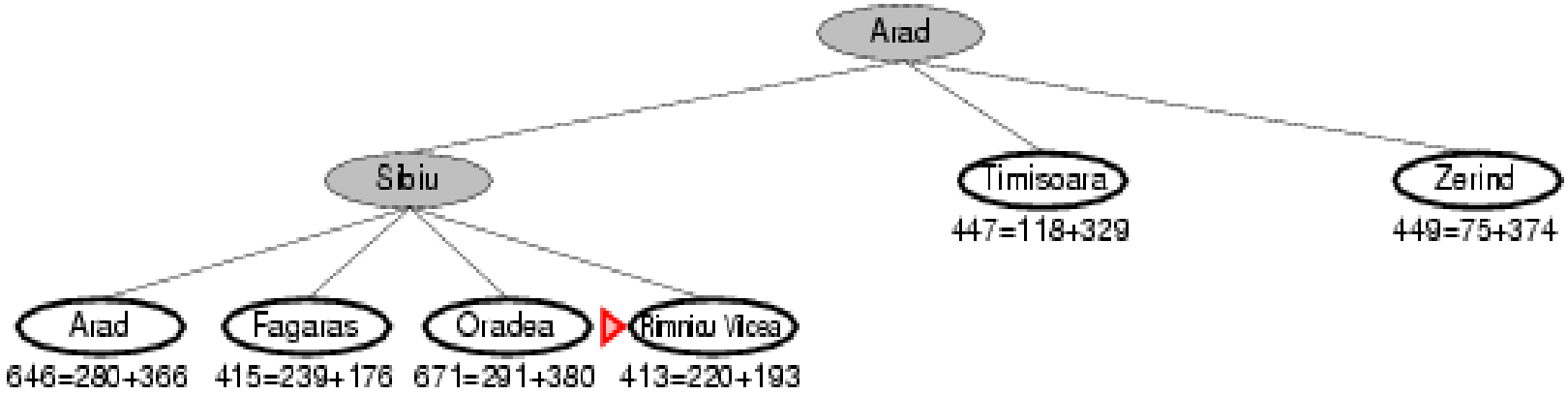
Straight line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Busca A*: exemplo

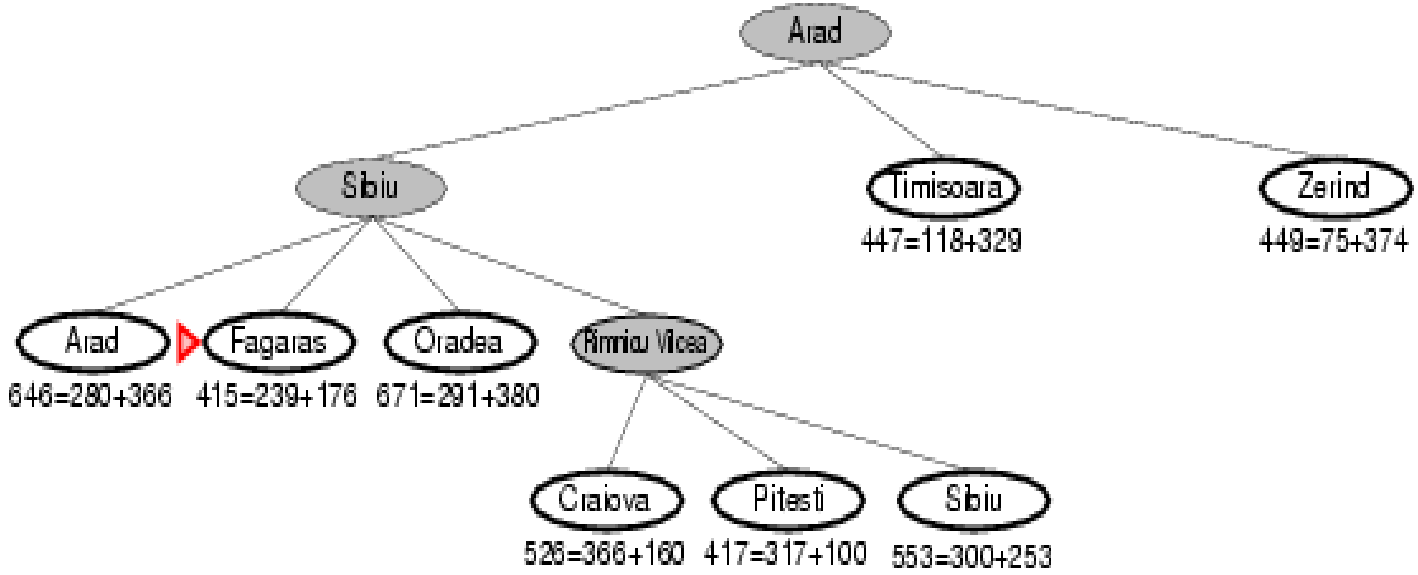
Busca A*: exemplo



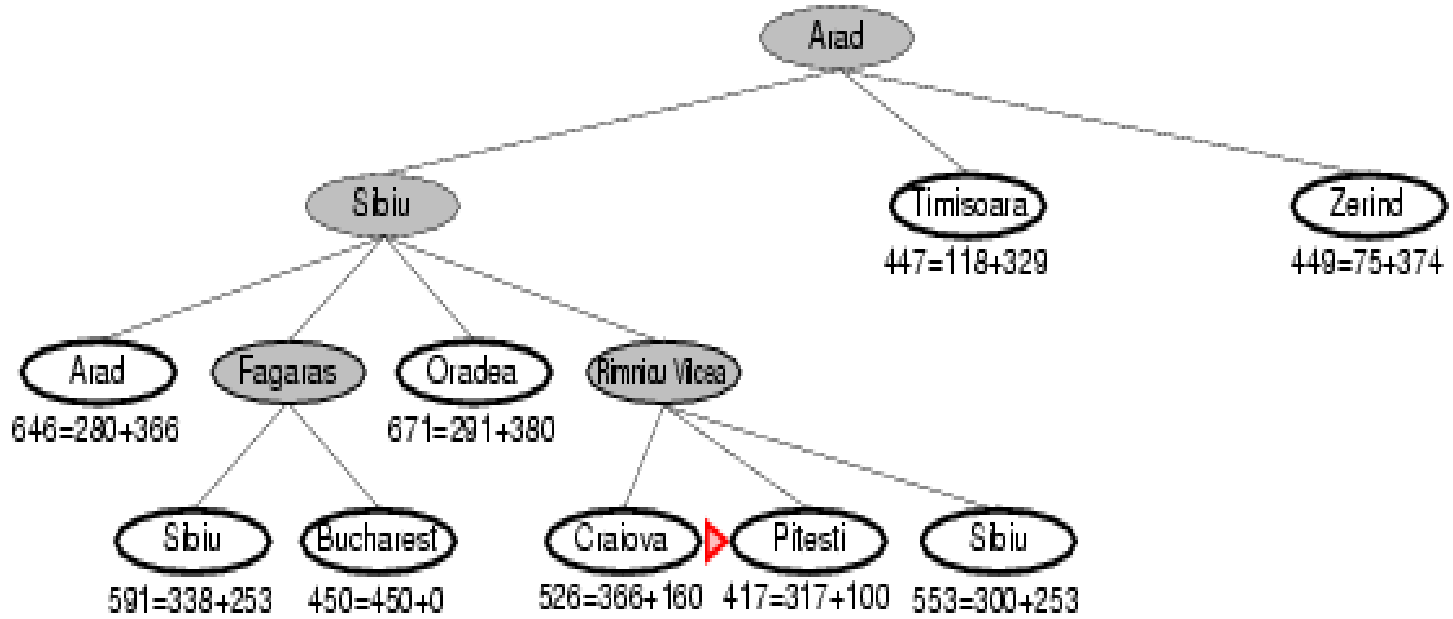
Busca A*: exemplo



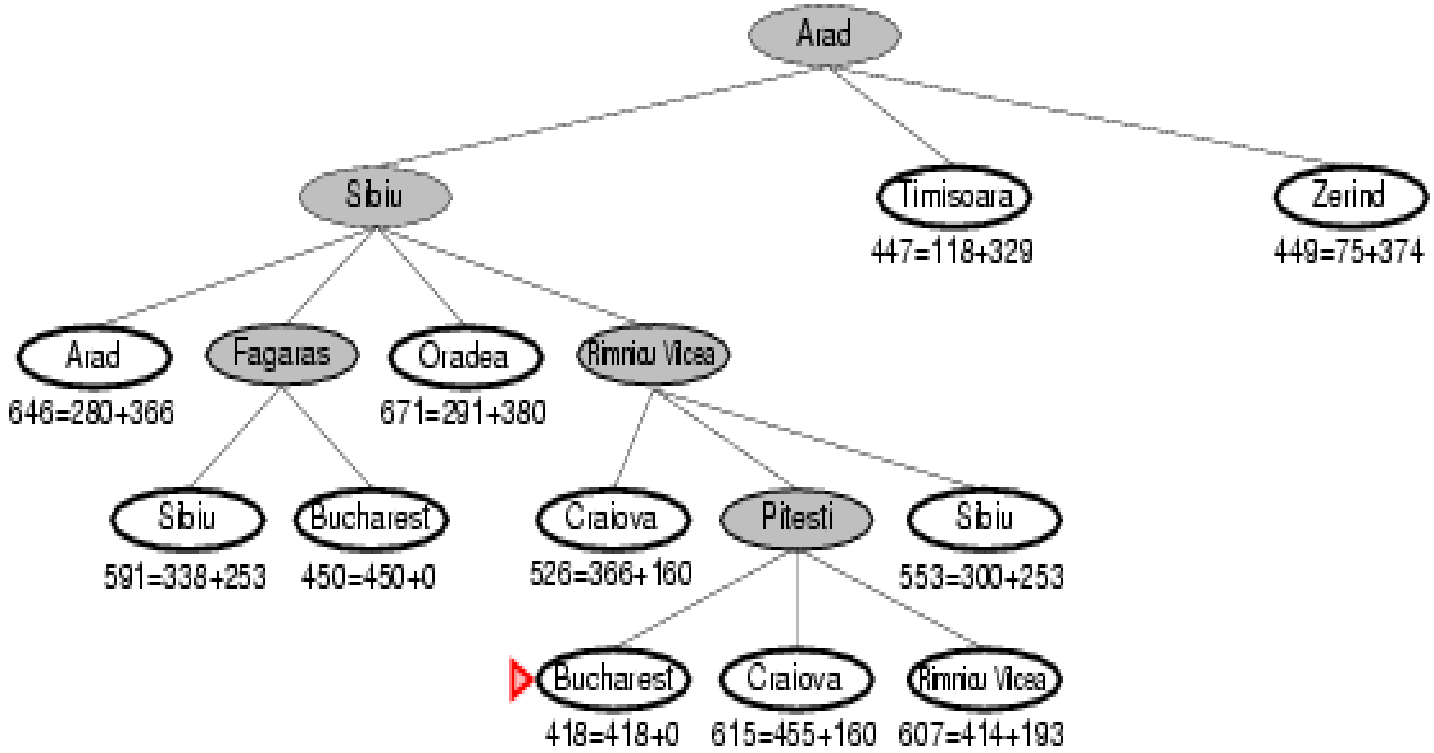
Busca A*: exemplo



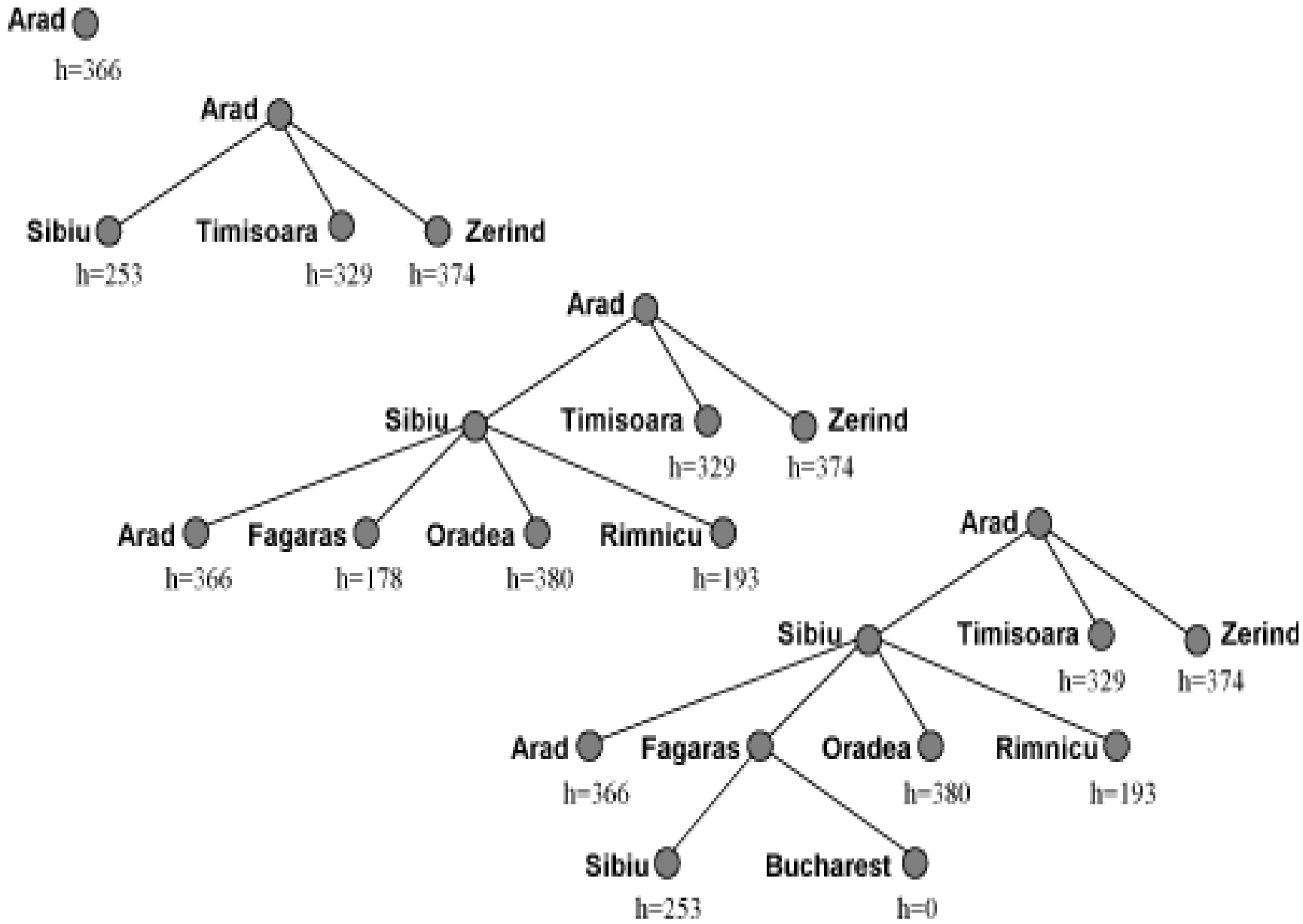
Busca A*: exemplo



Busca A*: exemplo



Se fosse pela Busca Gulosa...



Perceberam que A* tomou um rumo diverso do algoritmo guloso??
A* tomou o caminho ótimo..

Algoritmo A*:
Análise do comportamento

- A estratégia é completa e ótima
- Custo de tempo:
  - exponencial com o comprimento da solução, porém boas funções heurísticas diminuem significativamente esse custo
    - o fator de expansão fica próximo de 1
- Custo de memória: $O(b^d)$
  - guarda todos os nós expandidos na memória, para possibilitar o *backtracking*

Algoritmo A*

Análise do comportamento

- A estratégia apresenta eficiência ótima
  - nenhum outro algoritmo ótimo garante expandir menos nós
- A* só expande nós com $f(n) \leq C^*$, onde C^* é o custo do caminho ótimo
- Para se garantir otimalidade do A*, o valor de f em um caminho particular deve ser não decrescente!!!
  - $f(\text{sucessor}(n)) \geq f(n)$
  - i.e., o custo de cada nó gerado no mesmo caminho nunca é menor do que o custo de seus antecessores

Algoritmo A*

Análise do comportamento

- $f = g + h$ deve ser não decrescente
  - g é não decrescente (para operadores não negativos)
    - custo real do caminho já percorrido
  - h deve ser não-crescente (consistente, monotônica)
    - $h(n) \geq h(\text{sucessor}(n))$
    - i.e., quanto mais próximo do nó final, menor o valor de h
    - isso vale para a maioria das funções heurísticas
- Quando h não é consistente, para se garantir otimalidade do A*, temos:
  - quando $f(\text{suc}(n)) < f(n)$
  - usa-se $f(\text{suc}(n)) = \max(f(n), g(\text{suc}(n)) + h(\text{suc}(n)))$

A* define Contornos

$f(n) \leq C^*$

fator de expansão

próximo de 1