Informed Search Strategies

School of Electronic and Computer Engineering
Peking University

Wang Wenmin

What is Informed Search 什么是有信息搜索

- Also known as Heuristic Search.

  亦被称为启发式搜索。

- The strategies use problem-specific knowledge beyond the definition of the problem itself, so that can find solutions more efficiently than can an uninformed strategy.

  这类策略采用超出问题本身定义的、问题特有的知识，因此能够找到比无信息搜索更有效的解。

- The general approaches use one or both of following functions:

  一般方法使用如下函数中的一个或两者：

  - An evaluation function, denoted $f(n)$, used to select a node for expansion.
    评价函数，记作 $f(n)$，用于选择一个节点进行扩展。
  - A heuristic function, denoted $h(n)$, as a component of f.
    启发式函数，记作 $h(n)$，作为 f 的一个组成部分。

Contents

Best-first Search 最佳优先搜索

Search Strategy 搜索策略

- A node is selected for expansion based on an evaluation function, $f(n)$.

  搜索策略：一个节点被选择进行扩展是基于一个评价函数，$f(n)$。

- Most best-first algorithms also include a heuristic function, $h(n)$.

  大多数的最佳优先算法还包含一个启发式函数，$h(n)$。

Implementation 实现方法

- Identical to that for uniform-cost search.

  实现方法：与一致代价搜索相同。

- However best-first search uses of $f(n)$ instead of $g(n)$ to order the priority queue.

  然而，最佳优先搜索使用$f(n)$代替$g(n)$来整理优先队列。

Best-first Search 最佳优先搜索

Heuristic function 启发式函数

$h(n)$ = estimated cost of the cheapest path from the state at node n to a goal state.

$h(n)$ = 从节点n到目标状态的最低路径估计代价。

Special cases 特例

- Greedy Search

  贪婪搜索

- A* search

  A*搜索

Greedy Search 贪婪搜索

Search Strategy 搜索策略

- Try to expand the node that is closest to the goal.

  试图扩展最接近目标的节点。

Evaluation function 评价函数

$$f(n) = h(n)$$

- It evaluates nodes by using just the heuristic function.

  它仅使用启发式函数对节点进行评价。

- $h(n)$ -- estimated cost from n to the closest goal.

  $h(n)$ -- 从n到最接近目标的估计代价。

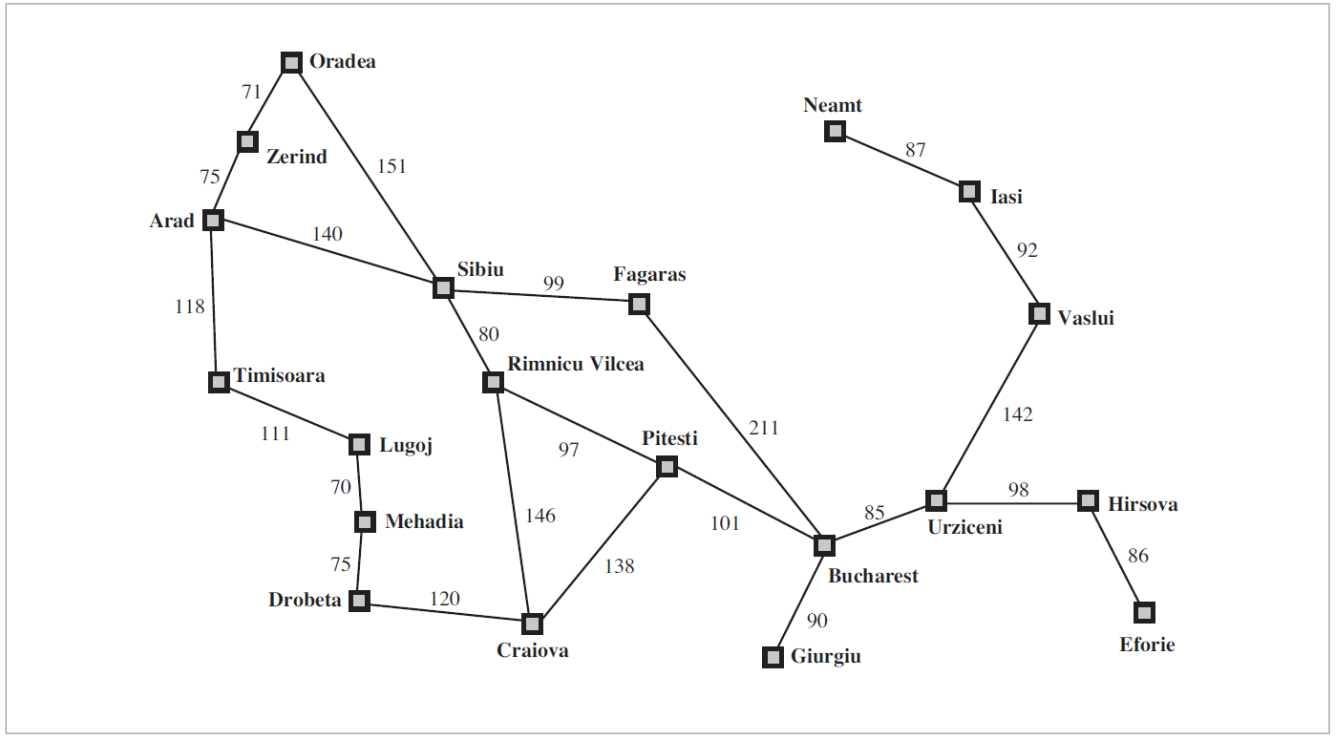
Why call "greedy" 为什么称为"贪婪"

- at each step it tries to get as close to the goal as it can.

  每一步它都试图得到能够最接近目标的节点。

Example: from Arad to Bucharest 举例：从Arad到Bucharest

- h_{SLD} : straight-line distance 直线距离



h_{SLD} Values

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

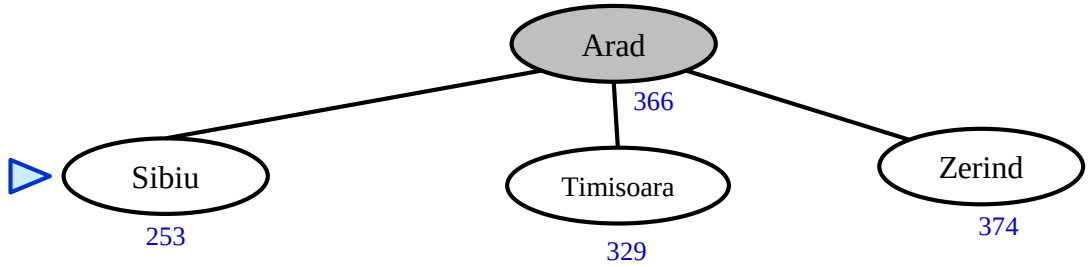
Notice: the values of h_{SLD} cannot be computed from the problem description itself. Moreover, it takes a certain amount of experience to know that h_{SLD} is correlated with actual road distances and therefore is a useful heuristic.

注意：h_{SLD}的值无法从问题描述本身来计算。此外，它要积累一定的经验才能知道，h_{SLD}与实际道路的距离相关，因此是一个有用的启发。

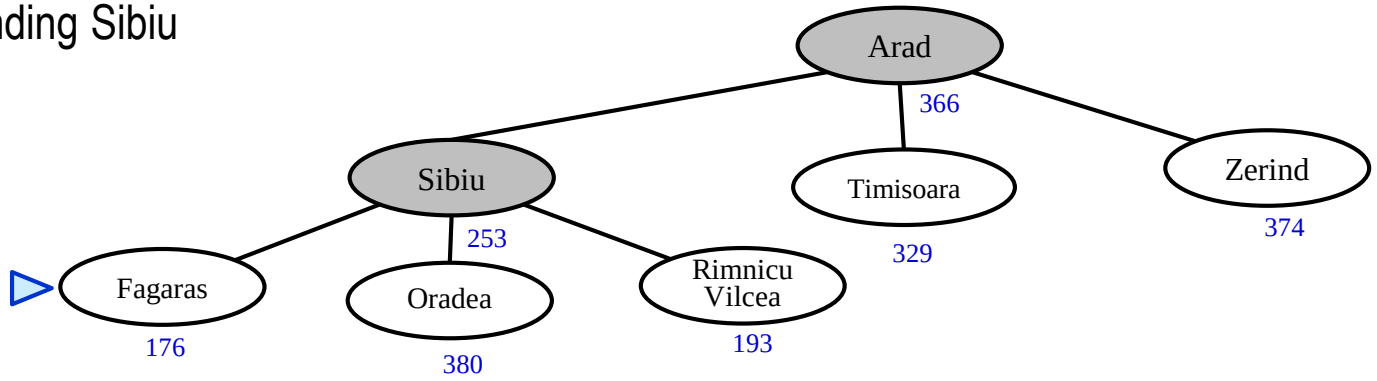
Example: from Arad to Bucharest 举例：从Arad到Bucharest

(a) The initial state

(b) After expanding Arad



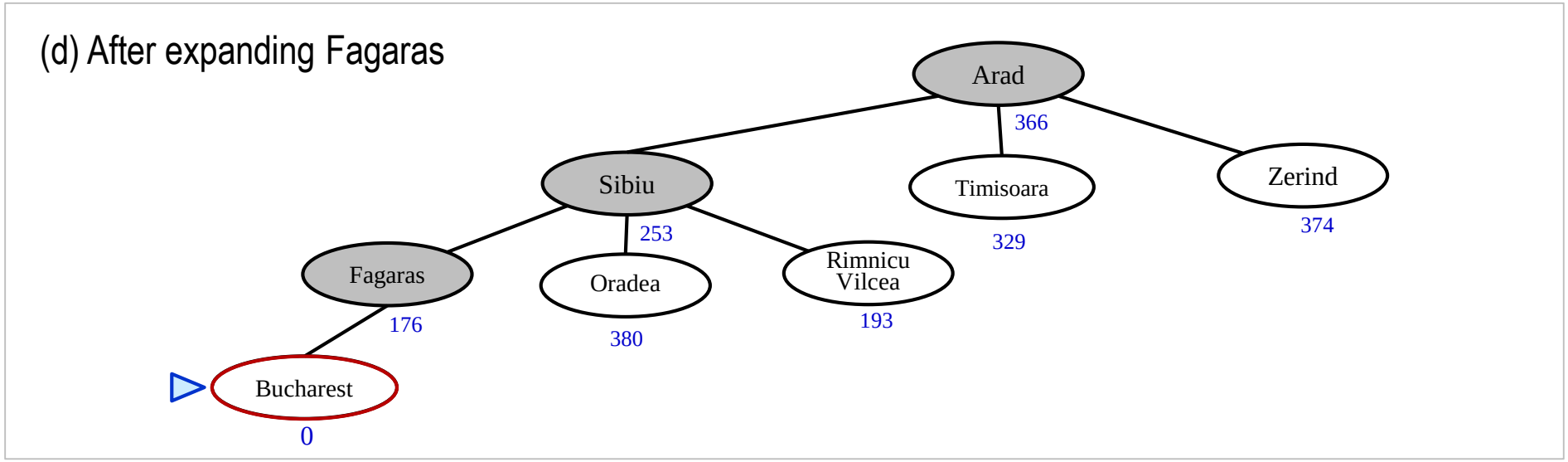
(c) After expanding Sibiu



h_{SLD} Values

Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

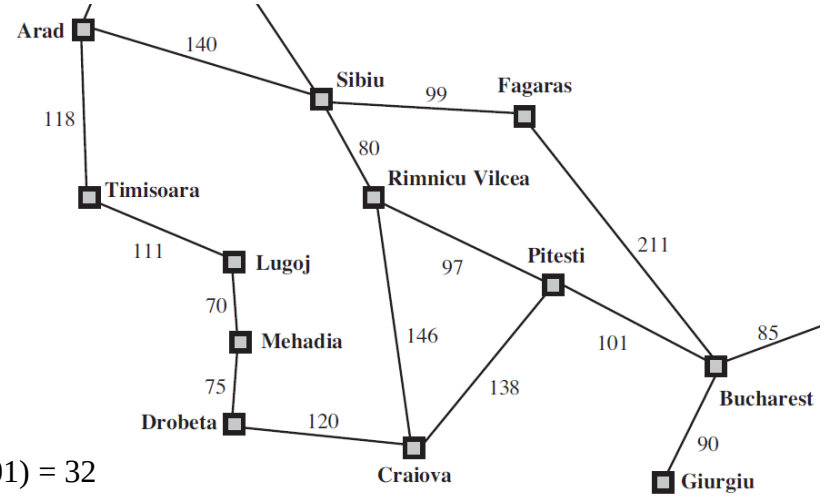
Example: from Arad to Bucharest 举例：从Arad到Bucharest



h_{SLD} Values

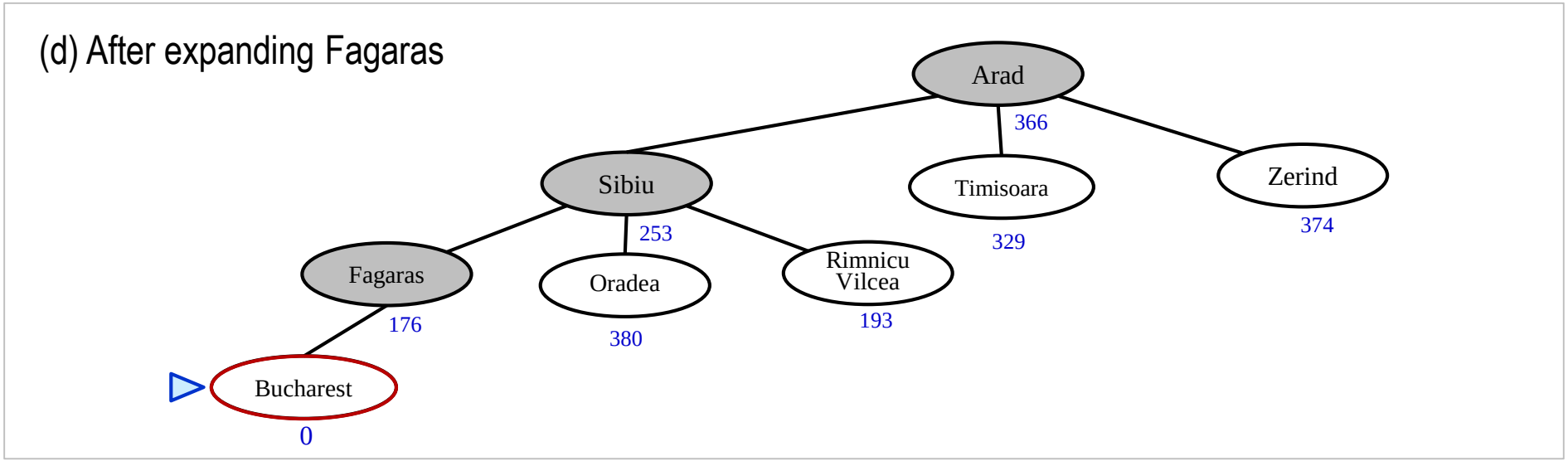
Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Notice: For this particular problem, it uses h_{SLD} to find a solution, hence its search cost is minimal. However it is not optimal: the path via Sibiu and Fagaras to Bucharest is 32 kilometers longer than the path through Rimnicu Vilcea and Pitesti.



(140+99+211) – (140+80+97+101) = 32

Example: from Arad to Bucharest 举例：从Arad到Bucharest



Notice: For this particular problem, it uses h_{SLD} to find a solution, hence its search cost is minimal. However it is not optimal: the path via Sibiu and Fagaras to Bucharest is 32 kilometers longer than the path through Rimnicu Vilcea and Pitesti.

注意：对这个具体问题，它采用h_{SLD}找到解，因此搜索代价是最小的。然而它不是最优的：如果计算路径代价的话，这条经由Sibiu和Fagaras到Bucharest的路径比经过Rimnicu Vilcea 和Pitesti远32公里。

(140+99+211) – (140+80+97+101) = 32

h_{SLD} Values

Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Properties of Greedy Tree Search 贪婪树搜索的特性

- Worst-case time: $O(b^m)$

  最差情况下的时间

- Space complexity: $O(b^m)$

  空间复杂性

where

- b -- the branching factor

  分支因子

- m -- the maximum depth of the search space

  搜索空间的最大深度

Thank you for your attention!