【人工智能】— 有信息搜索、最佳优先搜索、贪心搜索、A*搜索

无/有信息的搜索

- Uninformed search无信息的搜索：除了问题中提供的定义之外没有任何关于状态的附加信息。
- Informed search有信息的搜索：在问题本身的定义之外还可利用问题的特定知识。
- 无论任何情况下，**与无信息搜索策略相比，使用好的有信息的启发式搜索可以节省大量的时间和空间。**
- 有信息搜索又叫做启发式搜索，顾名思意，**这种搜索方法和启发式函数`h(n)`有着密切的关系。而`h(n)`就是有信息搜索的关键信息。**这里的**有信息指的是所求解问题之外的，但是与求解问题相关的特定信息或知识。**

Informed Search Algorithms

- **评价函数(evaluation function,`f(n)`)从当前结点出发根据评价函数来选择下一结点。**
- **启发函数(heuristic function,`h(n)`)从结点n到目标结点之间所形成的路径的最小代价值。**
- 启发函数并不唯一，在罗马尼亚问题中，启发函数使用的是两个城市之间的直线距离即欧氏距离。这里作者并未考虑地形因素的影响。
- 在启发函数的选择中，我们还可以以公路里程或者花费的时间作为启发函数。下表即为各个城市距离bucharest的直线距离，即有信息搜索中的关键信息。

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

- 代价一致搜索 Uniform Cost：
- UCS的原理：一致代价搜索总是扩展路径消耗最小的节点`N`。`N`点的路径消耗等于前一节点`N-1`的路径消耗加上`N-1`到`N`节点的路径消耗
  - 优点：UCS是完备的和可以取得最优解的！
  - 缺点：
    - 会扩展所有“方向”

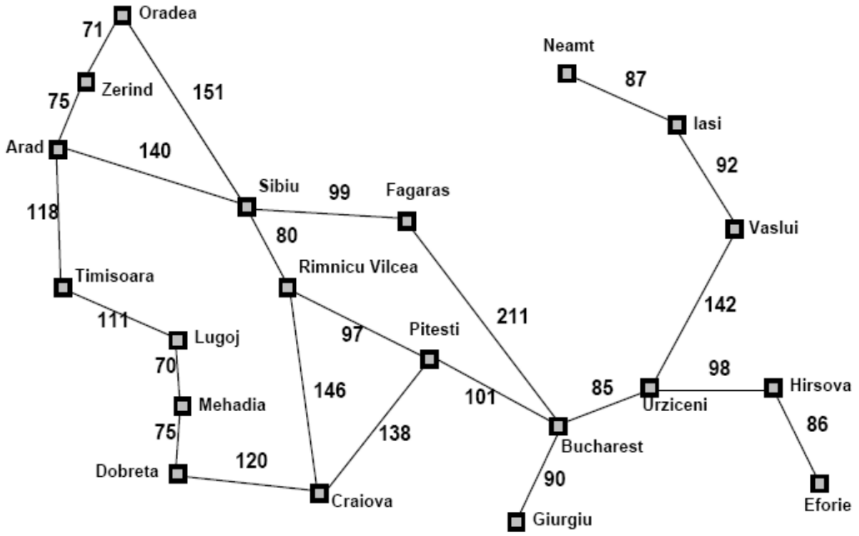
        - 没有关于目标状态的信息
    - 为什么不是有信息搜索：
        - 定义两个函数
            - `g(x)` 为从根节点到x节点的代价总和
            - `h(x)` 为从x节点到目标节点的估计代价总和
        - 代价一致搜索 (Uniform Cost Search or Dijkstra search) `f(x) = g(x)`
        - 贪心搜索 (Greedy Search) `f(x) = h(x)`
        - A星搜索 (A* Search) `f(x) = g(x) + h(x)`
    - 所以，因为`g(x)`是问题的定义获取的信息，不是跟目标状态有关的信息，所以**代价一致搜索不是有信息搜索**

Best-first search(最佳优先搜索)

- 思想：对每个节点使用评价函数`f(n)`
    - 评估"可取性"
    - 扩展最理想的未扩展节点
- 特例：
    - 贪婪搜索
    - A*搜索

Greedy Search

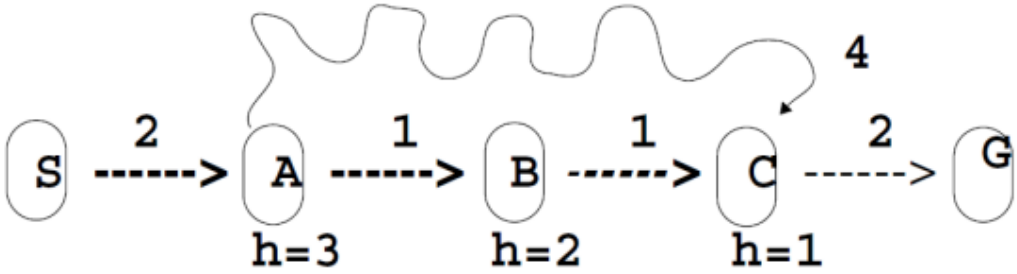
- $h_{SLD}(n)$= 从n到布加勒斯特的直线距离
- Greedy search expands the node that appears to be closest to goal(试图扩展离目标节点最近的点)



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

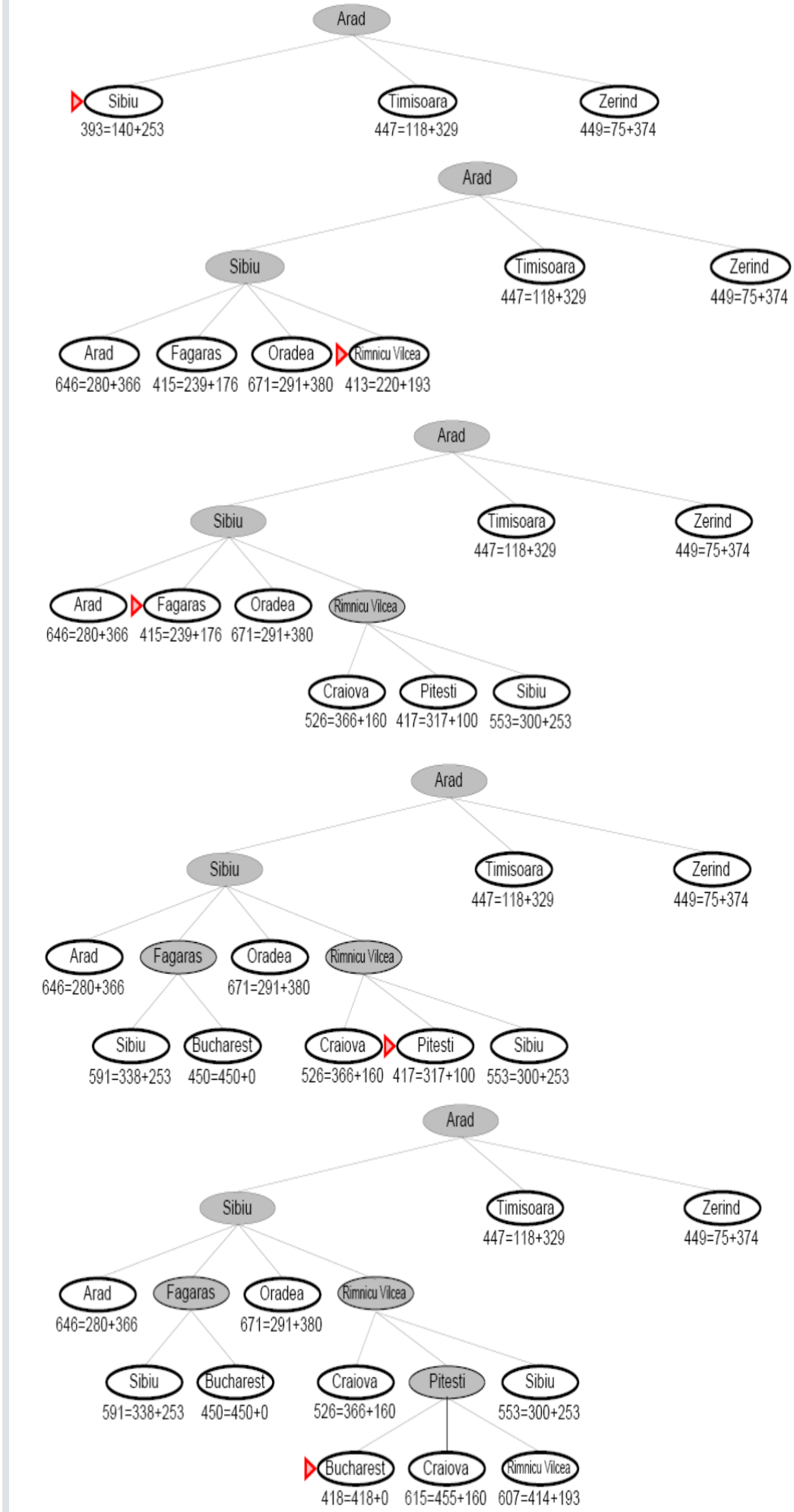
- 完备性：不一定，可能会陷入死循环，例如：**with Oradea as goal，Iasi → Neamt → Iasi → Neamt →...**
    - 在具有重复状态检查的有限空间可以实现完备性
    - b:分支因子、d:解深度、m:最大深度
- 时间复杂度：$O(b^m)$，但一个好的启发式方法可以带来显著的改进
- 空间复杂度：$O(b^m)$，将所有节点保留在内存中

- 最优解：不能保证，例如：



A* Search

- 评估函数：Evaluation function $f(n) = g(n) + h(n)$
- `g(n)` = cost so far to reach n —**到达节点n的耗散**
- `h(n)` = estimated cost to goal from n —**启发函数：从节点n到目标节点的最低耗散路径的耗散估计值**
- `f(n)` = estimated total cost of path through n to goal —**经过节点n的最低耗散的估计函数**
- $h(n) \leq h^*(n)$ where $h^*(n)$ is the true cost from n.
  - (Also require h(n) ≥ 0, so h(G) = 0 for any goal G.)
- E.g., $h_{SLD(n)}$ never overestimates the actual road distance (SLD: Straight-Line Distance)
- 举例：

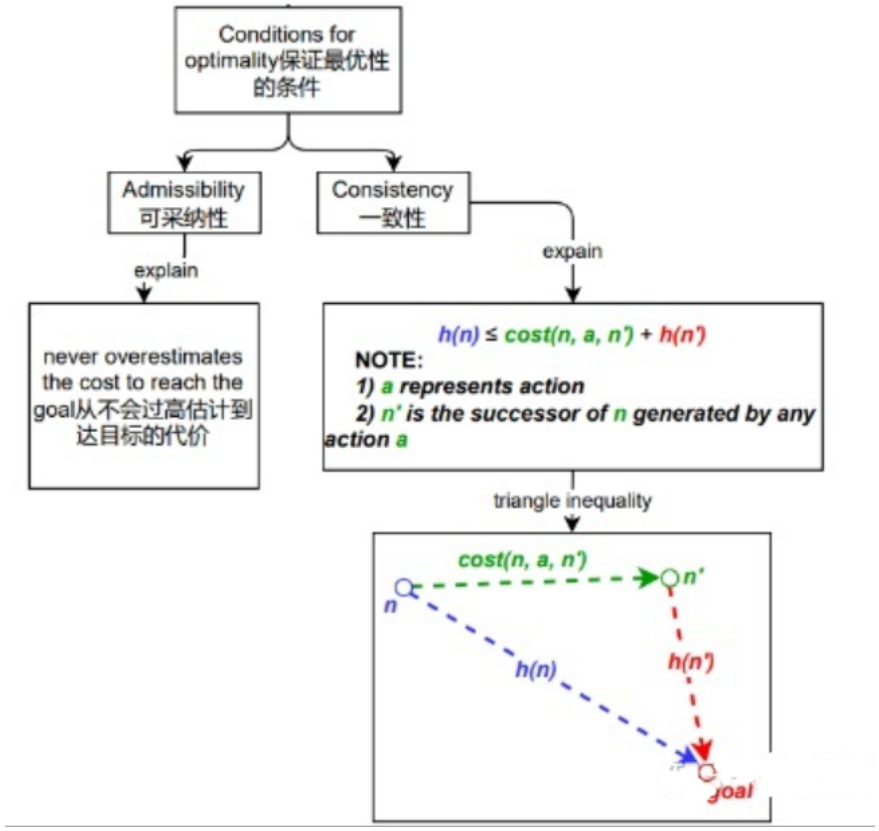
- 
Arad
Sibiu
393=140+253
Timisoara
447=118+329
Zerind
449=75+374
Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vilcea
413=220+193
Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vilcea
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253
Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253
Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
Sibiu
553=300+253
Bucharest
418=418+0
Craiova
615=455+160
Rimnicu Vilcea
607=414+193

- A*启发式 `h(n)` 是**可采纳的**：如果对于每个节点 `n`，$h(n)≤h^(n)，其中 h^*(n)$是从 `n` 达到目标状态的真实成本
- An admissible heuristic never overestimates the cost to reach the goal（**从不会过高估计到达目标的耗散**）, i.e., it is **optimistic（乐观的）**
- 示例：$h_{SLD}(n)$（**永远不会高估实际道路距离**
- 定理：如果 `h(n)` 是**可采纳的**，则使用**TREE-SEARCH的A*是最优的**
  - 如果 `h(n)` 是**一致的**，则使用**GRAPH-SEARCH的A*是最优的**
- 提出**可采纳的启发式方法是在实践中使用A*所涉及的大部分内容**
- 完备性：满足（除非存在无限多个 `f≤f(G)` 的节点）
- 时间复杂度：A*算法对于任何给定的启发函数都是效率最优，但仍然是指数级
- 空间复杂度：要保存所有结点在内存中
- 最优解：可以取得
- $$A^* \text{ expands all nodes with } f(n) < C^*$$
  $$A^* \text{ expands some nodes with } f(n) = C^*$$
  $$A^* \text{ expands no nodes with } f(n) > C^*$$

解释说明A*搜索是代价最优的和完备的

- 首先要先定义出什么是代价最优。这里的最优是指不存在另外一个解法能得到比A*算法所求得解法具有更小开销代价。
- 是否代价最优，取决于启发式函数 `h(n)` 的一些性质，**其中的一个关键性质为可采纳性(admissibility)，一个具备可采纳性的启发式函数永远不会高估到达某个目标的代价。**
- 专门针对启发函数而言，即**启发函数不会过高估计(over-estimate)从节点n到目标结点之间的实际开销代价(即小于等于实际开销)。**如可将两点之间的直线距离作为启发函数，从而保证其可容。
- **另外一个关键性质为一致性(consistency)。**假设节点 `n` 的后续节点是 `n'`，则从 `n` 到目标节点之间的开销代价一定小于从 `n` 到 `n'` 的开销再(单调性)加上从 `n'` 到目标节点之间的开销，满足以下条件：$h(n) \leq cost(n, a, n') + h(n')$
- 这里 `n'` 是 `n` 经过行动 `a` 所抵达的后续节点，`c(n,a,n')` 指 `n'` 和 `n` 之间的开销代价。则启发式函数 `h(n)` 是一致的。

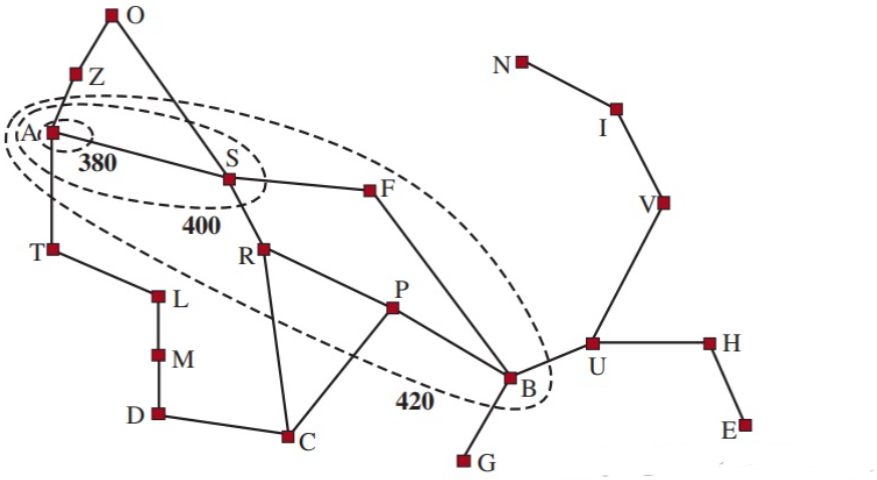
- A*搜索算法保持最优的条件：**启发函数具有可容性(admissible)和一致性 (consistency)**，如下图所示。



- 将直线距离作为启发函数`h(n)`，则启发函数一定是可容的，因为其不会高估开销代价。`g(n)`是从起始节点到节点`n`的实际代价开销，且$f(n)=g(n)+h(n)$，因此$f(n)$不会高估经过节点`n`路径的实际开销。
- $h(n) \leq c(n,a,n')+h(n')$构成了三角不等式。这里节点`n`、节点`n'`和目标结点`Goal`之间组成了一个三角形。
- 如果存在一条经过节点`n'`，从节点`n`到目标结点`Goal`的路径，其代价开销小于`h(n)`，则破坏了`h(n)`是从节点`n`到目标结点`Goal`所形成的具有最小开销代价的路径这一定义。

对搜索等值线如何理解

搜索等值线的概念更接近于地理上的等高线。如下图所示，在标记为400的等值线内，有$f(n)=g(n)+h(n) \leq 400$。由于A*搜索扩展的是`f`最小的边界结点，因此它的等值线是从初始结点以扇形向外扩展的，而一致代价搜索因为只有`g(n)`，因此是以圆形向外扩展的。



- 这里设`C`是`f(n)`的最优解，那么对于满足`f(n)<C`的结点n，称之为必然扩展结点(surely expanded node)，

- A*搜索可能会选出到达目标状态之前，恰好在等值线上面的点，即满足`f(n)=C`的结点。A搜索不会扩展`f(n)>C`的结点，因此可以得出结论：
    - **具有一致启发性函数的A搜索是效率最优的。**

情况	函数	结果
$\hat{h}(n)=0$，即 $\hat{f}(n)=\hat{g}(n)$	A*算法退化为Dijkstra算法	保证能找到最短路径
$\hat{h}(n) \leq$ 实际代价	$\hat{h}(n)$越小，A*扩展的节点越多，运行的越慢	保证能找到一条最短路径，但运算更快了
$\hat{h}(n)=$ 实际代价	仅寻找最佳路径，而不扩展任何别的节点	保证能找到一条最短路径，并且运算非常快
$\hat{h}(n)>$ 实际代价	寻找最佳路径且扩展别的任何节点	不能保证找到一条最短路径，但运算更快了
$\hat{h}(n)>>\hat{g}(n)$	A*算法退化为BFS算法	不能保证找到一条最短路径，但运算非常快