

# Heuristic search

- $A^*$  search
- $A^*$  properties
- Constructing heuristics
- Running  $A^*$  on examples

\*Slides based on those of Sheila McIlraith

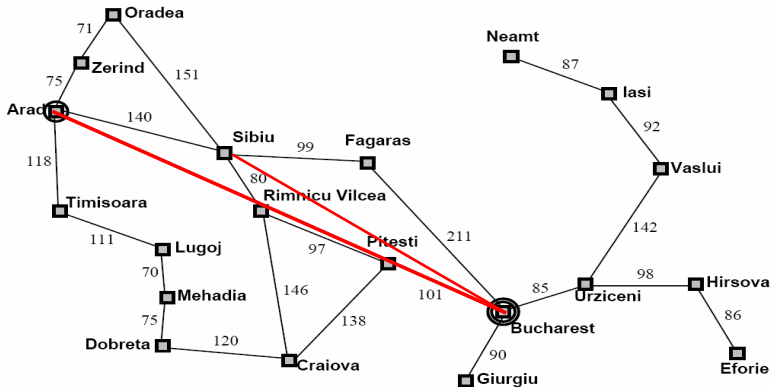
# Motivation

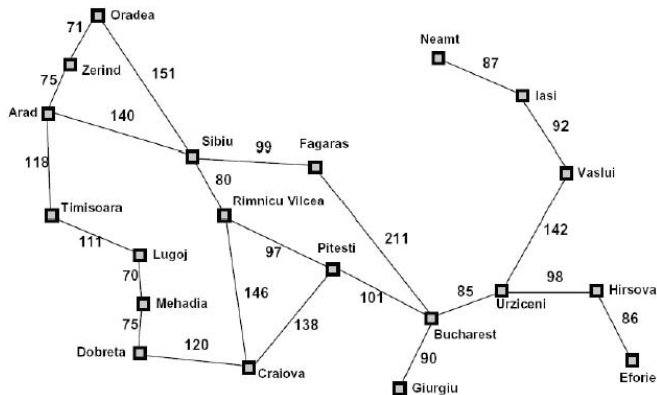
- In uninformed search, we don't try to evaluate which of the nodes on the frontier are most promising.
  - e.g., in uniform cost search we always expand the cheapest path. We don't consider the cost of getting to the goal from the end of the current path.
- However, often we have some other knowledge about the merit of nodes.
  - 优点 e.g., how costly it is to get to the goal from that node.

# Heuristic search

- The idea is to develop a domain specific heuristic function  $h(n)$ , guessing the cost of getting to the goal from node  $n$
- We require that  $h(n) = 0$  for every node  $n$  whose state satisfies the goal
- There are different ways of guessing this cost in different domains. *i.e.*, heuristics are domain specific.

# Example: straight line distance, *i.e.*, Euclidean distance



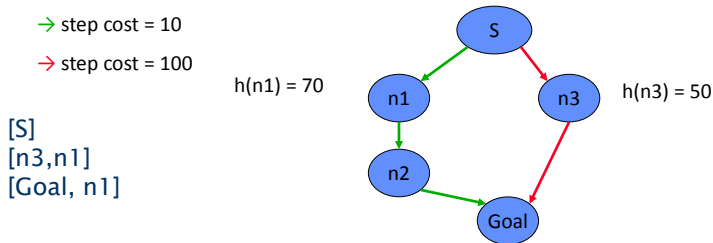


Straight-line distance  
to Bucharest

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

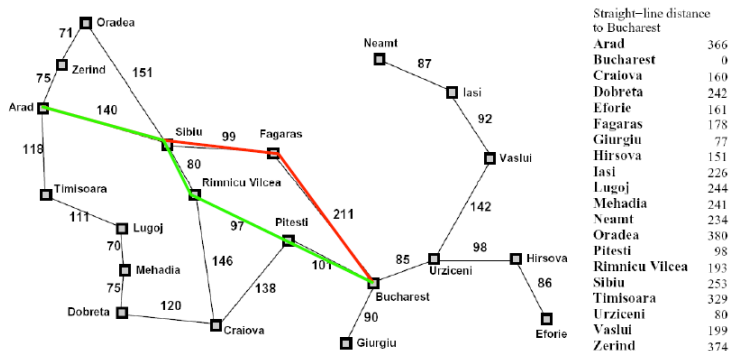
# Greedy best-first search (Greedy BFS)

- We use  $h(n)$  to order the nodes on the frontier.
- We are greedily trying to achieve a low cost solution.
- However, this method ignores the cost of getting to  $n$ , so it can be lead astray exploring nodes that cost a lot to get to but seem to be close to the goal



- Thus Greedy BFS is incomplete, not optimal

# An example



Arad-Sibiu-RV-Pitesli-Bucharest:

$$140 + 80 + 97 + 101 = 140 + 278 = 418$$

$$\text{Arad-Sibiu-Fagaras-Bucharest: } 140 + 99 + 211 = 140 + 310 = 450$$

实际成本  $g(n)$  : 从起始节点到  $n$  的实际成本

$h(n)$  : 启发式估计节点  $n$  到目标节点的成本

$h^*(n)$  : 节点  $n$  到目标节点的实际最小成本

$h(n)=0$  :  $n$  为目标节点

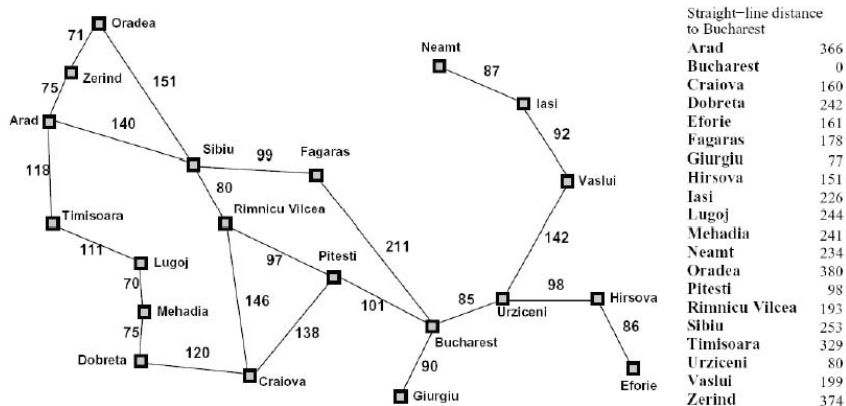
- Define an evaluation function  $f(n) = g(n) + h(n)$ 
  - $g(n)$  is the cost of the path to node  $n$
  - $h(n)$  is the heuristic estimate of the cost of getting to a goal node from  $n$
- So  $f(n)$  is an estimate of the cost of getting to the goal via node  $n$ .
- We use  $f(n)$  to order the nodes on the frontier.

可采纳 :  $h(n) \leq h^*(n)$  如果  $h(n) > h^*(n)$  将找不到最优路径

单调性 :  $h(n_1) - h(n_2) \leq c(n_1, n_2)$



# An example



Arad-Sibiu-RV-Pitesli-Bucharest: 418

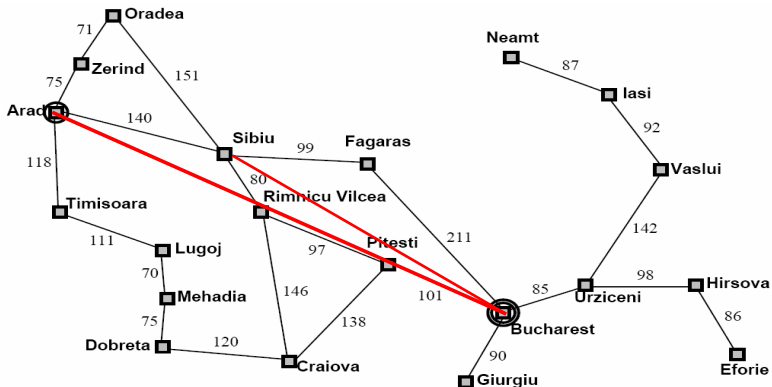
# Conditions on $h(n)$ : Admissible

- We always assume that  $c(n_1 \rightarrow n_2) \geq \epsilon > 0$ . The cost of any transition is greater than zero and can't be arbitrarily small. 任意
- Let  $h^*(n)$  be the cost of an optimal path from  $n$  to a goal node ( $\infty$  if there is no path).
- $h(n)$  is admissible if for all nodes  $n$ ,  $h(n) \leq h^*(n)$
- So an admissible heuristic underestimates the true cost to reach the goal from the current node
- Hence  $h(g) = 0$  for any goal node  $g$ , and  $h(n) = \infty$  if there is no path from  $n$  to the goal

# Consistency (aka monotonicity)

- $h(n)$  is consistent/<sup>单调性</sup>monotone if for any nodes  $n_1$  and  $n_2$ ,  
$$h(n_1) \leq c(n_1 \rightarrow n_2) + h(n_2)$$
- Note that <sup>单调 可采纳</sup>consistency implies admissibility (proof)
  - Case 1: no path from  $n$  to the goal
  - Case 2: Let  $n = n_1 \rightarrow n_2 \rightarrow \dots \rightarrow n_k$  be an optimal path from  $n$  to a goal node. We prove by induction <sup>归纳</sup>on  $i$  that for all  $i$ ,  $h(n_i) \leq h^*(n_i)$ .
- Most admissible heuristics are also monotone.  
<sup>可采纳</sup>

# An example: straight line distance



# An example: admissible but nonmonotonic 可采纳但不单调

带环检测则将找不到最佳路径

The following  $h$  is **not consistent** (i.e., not monotone) since  $h(n2) > c(n2 \rightarrow n4) + h(n4)$ . But it is **admissible**.

→ step cost = 200

→ step cost = 100

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

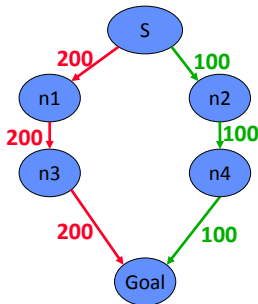
$\{S\} \rightarrow \{n1 [200+50=250], n2 [200+100=300]\}$   
→  $\{n2 [100+200=300], n3 [400+50=450]\}$   
→  $\{n4 [200+50=250], n3 [400+50=450]\}$   
→  $\{goal [300+0=300], n3 [400+50=450]\}$

$h(n1) = 50$

$h(n3) = 50$

$h(n2) = 200$

$h(n4) = 50$



We **do find** the optimal path as the heuristic is still admissible. **But** we are misled into ignoring  $n2$  until after we expand  $n1$ .

# Time and space complexities

- When  $h(n) = 0$ , for all  $n$ ,  $h$  is monotone.  $A^*$  becomes uniform-cost
- Hence the same bounds as uniform-cost apply. (These are worst case bounds). Still exponential unless we have a very good  $h!$

与一致代价搜索的复杂度相同

指数

# Admissibility implies optimality

- Suppose that an optimal solution has cost  $C^*$
- Any optimal solution will be expanded before any path with cost  $> C^*$  (to be proved later)
- There are finitely many paths with cost  $\leq C^*$
- Eventually we must examine an optimal solution

Any optimal path will be expanded before any path of cost  $> C^*$

Proof:

- Let  $p^*$  be an optimal solution
- Assume that  $p$  is a path s.t.  $c(p) > c(p^*)$  and  $p$  is expanded before  $p^*$  **Assume that  $x$  is the final node at  $p$**
- Then there must be a node  $n$  on  $p^*$  which is still in the frontier  
 $f(x) = g(x) + h(x)$
- So  $c(p) \leq \boxed{f(x)} \leq f(n) = g(n) + h(n)$   
 $\leq g(n) + h^*(n) = c(p^*)$ , contradicting  $c(p) > c(p^*)$   
**矛盾**



# What about cycle checking?

- We will show that monotonicity guarantees we have found an optimal path to a node the first time we visit it  
在单调性下，环检测保持最优性
- Thus with monotonicity, cycle checking preserves optimality  
在可采纳下，环检测可能不能保持最优性
- However, with only admissibility, cycle checking might not preserve optimality.
- To fix this: for previously visited nodes, must remember cost of previous path. If new path is cheaper must explore again.

# Consequences of consistency

$$f(n_i) \leq f(n_{i+1})$$

**Proposition 1.** The  $f$ -values of nodes along a path must be non-decreasing

# Consequences of consistency

**Proposition 2.** If  $n_2$  is expanded after  $n_1$ , then  $f(n_1) \leq f(n_2)$

Proof. There are two cases:

- ① When  $n_1$  is expanded,  $n_2$  is on the frontier. 显然
- ② When  $n_1$  is expanded, some ancestor  $n$  of  $n_2$  is on the frontier.  $n_2$ 的父节点 $n_2$ 在边界上则 $f(n_1) \leq f(n_i) \leq f(n_2)$

# Consequences of consistency

**Proposition 3.** When  $n$  is expanded every path with lower  $f$ -value has already been expanded.

Proof.

- Suppose that  $p = n_1 \rightarrow n_2 \dots \rightarrow n_k$  has not expanded but  $f(n_k) < f(n)$
- Suppose that  $n_i$  is the last expanded node on  $p$
- Then  $n_{i+1}$  must be on the frontier when  $n$  is expanded
- So  $f(n) \leq f(n_{i+1})$
- By Proposition 1,  $f(n_{i+1}) \leq f(n_k)$ ,
- Hence  $f(n) \leq f(n_k)$ , a contradiction

# Consequences of consistency

**Proposition 4.** The first time  $A^*$  expands a state, it has found the minimum cost path to that state.

Proof.

- Let  $p = n_1 \rightarrow n_2 \dots \rightarrow n_k = n$  be the first path to  $n$  found.
- Let  $p' = m_1 \rightarrow m_2 \dots \rightarrow m_j = n$  be a path to  $n$  found later
- By Proposition 2,  $g(p) + h(n) \leq g(p') + h(n)$
- Hence  $g(p) \leq g(p')$ .

- A\* has the same potential space problems as BFS or UCS
- IDA\* - Iterative Deepening A\* is similar to Iterative Deepening Search and similarly addresses space issues. 解决
- Like iterative deepening, but now the cutoff is the f-value 截断 (g+h) rather than the depth 按f值迭代（先最小，再次小）
- At each iteration, the cutoff value is the smallest f-value of any node that exceeded the cutoff on the previous iteration  
在每次迭代中，截止值是超过上一次迭代截止值的节点中的最小f值

# A\* search: Summary

- Define an evaluation function  $f(n) = g(n) + h(n)$
- We use  $f(n)$  to order the nodes on the frontier.
- $h(n)$  is admissible if for all nodes  $n$ ,  $h(n) \leq h^*(n)$
- $h(n)$  is consistent/monotone if for any nodes  $n_1$  and  $n_2$ ,  
 $h(n_1) \leq c(n_1 \rightarrow n_2) + h(n_2)$
- Consistency implies admissibility
- Admissibility implies optimality
- Exponential time and space complexity

# Consequences of consistency: Summary

- ① The  $f$ -values of nodes along a path must be non-decreasing
- ② If  $n_2$  is expanded after  $n_1$ , then  $f(n_1) \leq f(n_2)$
- ③ 当 $n$ 被拓展时，所有 $f$ 值较低的路径都已经被拓展  
When  $n$  is expanded every path with lower  $f$ -value has already been expanded.
- ④ The first time  $A^*$  expands a state, it has found the minimum cost path to that state.

Thus with monotonicity, cycle checking preserves optimality



# Building heuristics: Relaxed problem

- One useful technique is to consider an easier problem, and let  $h(n)$  be the cost of reaching the goal in the easier problem.
- 8-puzzle moves: can move a tile from square A to B if
  - A is adjacent (left, right, above, below) to B
  - and B is blank
- Can relax some of these conditions
  - 1 can move from A to B if A is <sup>邻</sup>adjacent to B (ignore whether B is blank)
  - 2 can move from A to B if B is blank (ignore adjacency)
  - 3 can move from A to B (ignore both conditions)

# Building heuristics: Relaxed problem

- #3 leads to the **misplaced tiles** heuristic 单调的
  - $h(n)$  = number of misplaced tiles
  - admissible: for each misplaced tile, we need at least one action to move it to its goal position; such actions are different for any two different misplaced tiles
  - monotone: any action can remove at most one misplaced tile, hence for any nodes  $n_1$  and  $n_2$ ,  $h(n_1) - h(n_2) \leq c(n_1 \rightarrow n_2)$
- #1 leads to the **Manhattan distance** heuristic 单调的
  - $h(n)$  = sum of the Manhattan distances
  - admissible: for each misplaced tile, we need at least  $d$  actions to move it to its goal position, where  $d$  is the Manhattan distance between the initial and the goal positions; the sets of such actions are **不相交** disjoint for any two different misplaced tiles
  - monotone: any action can decrease  $h(n)$  by at most 1

# Building heuristics: Relaxed problem

松弛问题中节点的最优代价是原问题的一种可采纳启发式算法

**Theorem.** The optimal cost to nodes in the relaxed problem is an admissible heuristic for the original problem!

Proof:

- Let  $P$  be the original problem, and  $P'$  the relaxed problem
- Then  $Sol(P) \subseteq Sol(P')$
- So  $mincost(Sol(P')) \leq mincost(Sol(P))$
- Thus  $h(n) \leq h^*(n)$

# Comparing two heuristics

$h_2$ 比 $h_1$ 更具优势

**Definition.** We say that  $h_2$  dominates  $h_1$  (or is more informed than  $h_1$ ), if both are admissible and for every node  $n$  other than the goal nodes, we have  $h_1(n) \leq h_2(n)$ .

**Theorem.** If  $h_2$  dominates  $h_1$ , then every node that is expanded by A\* using  $h_2$  is also expanded by A\* using  $h_1$ .

Depth	IDS $h=0$	A*(Misplaced) $h_1$	A*(Manhattan) $h_2$
10	47,127	93	39
14	3,473,941	539	113
24	---	39,135	1,641

$h_2$ 探索的节点较少更有优势

# Running A\* with cycle checking on the 8-puzzle problem

采用Manhattan启发式函数，用带环检测的A\*搜索初始状态和目标状态如下图所示的8数码问题，画出搜索图，图中标明所有节点的 $f, g, h$ 值。

初始:

2	8	3
1	6	4
7		5

目标:

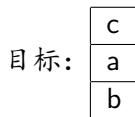
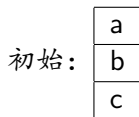
1	2	3
8		4
7	6	5

# 积木世界规划

现有积木若干，积木可以放在桌子上，也可以放在另一块积木上面。有两种操作：

- ①  $move(x, y)$ ：把积木 $x$ 放到积木 $y$ 上面。前提是积木 $x$ 和 $y$ 上面都没有其他积木。
- ②  $moveToTable(x)$ ：把积木 $x$ 放到桌子上，前提是积木 $x$ 上面无其他积木，且积木 $x$ 不在桌子上。

设计本问题的一个启发式函数 $h(n)$ ，满足 $h(n) \leq h^*(n)$ ，然后用A\*搜索初始状态和目标状态如下图所示的规划问题：



$h(n)$ 越接近 $h^*(n)$ 越好

# 积木世界规划

- 我们说一块积木在其目标位置如果以这块积木为顶的子塔出现在目标状态。
- 令 $h(n)$ 为不在目标位置的积木个数。
- admissible: for each misplaced block, we need at least one action to move it to its goal position; such actions are different for any two different misplaced blocks
- monotone: any action can remove at most one misplaced block

Can you design a better admissible heuristic function?

- We say  $x$  is a good tower if  $x$  is in its goal position
- If it is at all possible to create a good tower, do so;
- else, move a block to the table, but be sure it doesn't come from a good tower.



# 滑动积木块游戏

- 一个盒子中有七个格子，里面放了黑色，白色两种木块；
- 三个黑色在左边，三个白色在右边，最右边一个格子空着；
- 一个木块移入相邻空格，耗散值为1；
- 一个木块相邻一个或两个其他木块跳入空格，耗散值为跳过的木块数；
- 游戏中将所有白色木块跳到黑色木块左边为成功；

# 滑动积木块游戏

- 令 $h(n)$ 为每个白色木块前的黑色木块数目和
- $EX \Rightarrow XE$ ,  $EXY \Rightarrow YXE$ ,  $EXYZ \Rightarrow ZXYE$
- 每个代价为1的动作使 $h(n)$ 至多下降1
- 每个代价为2的动作使 $h(n)$ 至多下降2
- 因此 $h(n)$ 是单调的

# The missionaries and cannibals problem

- $N$  missionaries and  $N$  cannibals are at the left bank of a river
- There is a boat that can hold  $K$  people
- Find a way to get everyone to the right bank
- So that at any time, at any place (on either bank, or in the boat),  $\# \text{missionaries} \geq \# \text{cannibals}$  or  $\# \text{missionaries} = 0$

# Formulation of the MC problem

- States  $(M, C, B)$  where  $M$  – #missionaries,  $C$  – #cannibals at the left bank,  $B = 1$  indicates the boat is at the left bank
- Actions  $(m, c)$  where  $m$  – #missionaries,  $c$  – #cannibals on the boat
- Precondition: #missionaries and #cannibals satisfy the constraint
- Effects:  $(M, C, 1) \xrightarrow{(m,c)} (M - m, C - c, 0)$  and  $(M, C, 0) \xrightarrow{(m,c)} (M + m, C + c, 1)$

# Heuristics for $K \leq 3$

- Is  $h_1(n) = M + C$  admissible? No, in case of  $(1, 1, 1)$ ,  $h_1(n) = 2$ , but  $h^*(n) = 1$
- Let  $h(n) = M + C - 2B$
- monotone:
  - $(M, C, 1) \xrightarrow{(m,c)} (M - m, C - c, 0)$ :  
 $h(n_1) - h(n_2) = m + c - 2 \leq K - 2 \leq 1$
  - $(M, C, 0) \xrightarrow{(m,c)} (M + m, C + c, 1)$ :  
 $h(n_1) - h(n_2) = 2 - (m + c) \leq 1$ , since  $m + c \geq 1$

# Directly proving admissibility

When  $B = 1$ , in the best situation

- In the last step, we can get 3 people to the right bank
- Preceding that, we can get 3 people to the right bank, and then a person gets the boat back to the left bank
- Thus we need  $\geq 2 \cdot \lceil \frac{M+C-3}{2} \rceil + 1 \geq M + C - 2$  actions

When  $B = 0$ ,

- We need a person to get the boat back to the left bank
- From above, now to get  $M + C + 1$  people to the right bank, we need  $\geq M + C + 1 - 2$  actions
- Thus in total, we need  $\geq M + C$  actions

# Exercise

Running breadth-first with cycle-checking for  $M = 3$  and  $K = 2$

# Exercise

Running  $A^*$  with cycle checking for  $M = 5$  and  $K = 3$