Informed Search

吉建民

USTC
jianmin@ustc.edu.cn

2024 年 3 月 11 日

Used Materials

Disclaimer: 本课件采用了 S. Russell and P. Norvig's Artificial Intelligence –A modern approach slides, 徐林莉老师课件和其他网络课程课件，也采用了 GitHub 中开源代码，以及部分网络博客内容

Table of Contents

课程回顾

```
function TREE-SEARCH( problem, fringe) returns a solution, or failure
    fringe ← INSERT(MAKE-NODE(INTIAL-STATE[problem]), fringe)
    loop do
      if fringe is empty then return failure
      node ← REMOVE-FRONT (fringe)
      if GOAL-TEST[problem] applied to STATE(node) succeeds
            return node
      fringe ← INSERTALL(EXPAND(node, problem), fringe)
```

- A strategy is defined by picking the order of node expansion
- Variety of uninformed search strategies
  - Iterative deepening search uses only linear space and not much more time than other uninformed algorithms

Uninformed search strategies

Uninformed search strategies use only the information available in the problem definition

- Breadth-first search (广度优先搜索)
- Uniform-cost search (代价一致搜索)
- Depth-first search (深度优先搜索)
- Depth-limited search (深度有限搜索)
- Iterative deepening search (迭代深入深度优先搜索)
- Bidirectional search (双向搜索)

Summary of algorithms

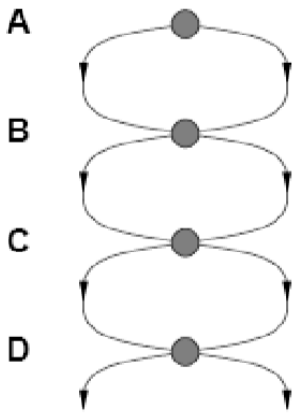
Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

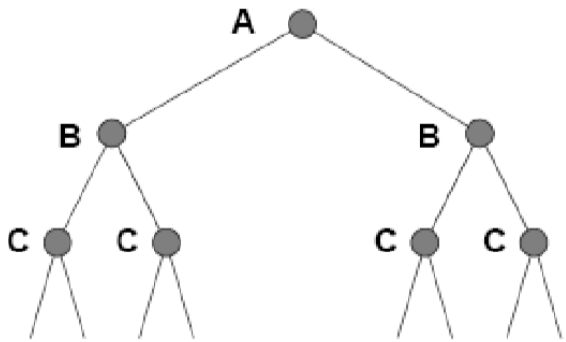
Figure 3.21 Evaluation of tree-search strategies. b is the branching factor; d is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: [a] complete if b is finite; [b] complete if step costs $\geq \epsilon$ for positive ϵ; [c] optimal if step costs are all identical; [d] if both directions use breadth-first search.

- b: Branching factor
- d: Solution Depth
- m: maximum Depth

Repeated states

- Failure to detect repeated states can turn a linear problem into an exponential one!





Graph search

```
function GRAPH-SEARCH( problem, fringe) returns a solution,
    or failure
      closed ← an empty set
      fringe ← INSERT(MAKE-NODE(INITIAL-
    STATE[problem]),fringe)
      loop do
    if fringe is empty then return failure
    node  ← REMOVE-FRONT(fringe)
    if GOAL-TEST(problem , STATE[node]) then return node
    if STATE[node] is not in closed  then
          add STATE[node] to closed
          fringe ← INSERTALL(EXPAND(node, problem), fringe)
      end
```

Informed search

- 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。

Table of Contents

Best-first search

- Idea: use an evaluation function (评价函数) for each node
  — estimate of"desirability"
  $\Longrightarrow$ Expand most desirable unexpanded node

- A heuristic is:
  - A function that estimates how close a state is to a goal
  - Designed for a particular search problem

- Implementation: fringe is a queue sorted in decreasing order of desirability
  — priority queue (优先级队列)

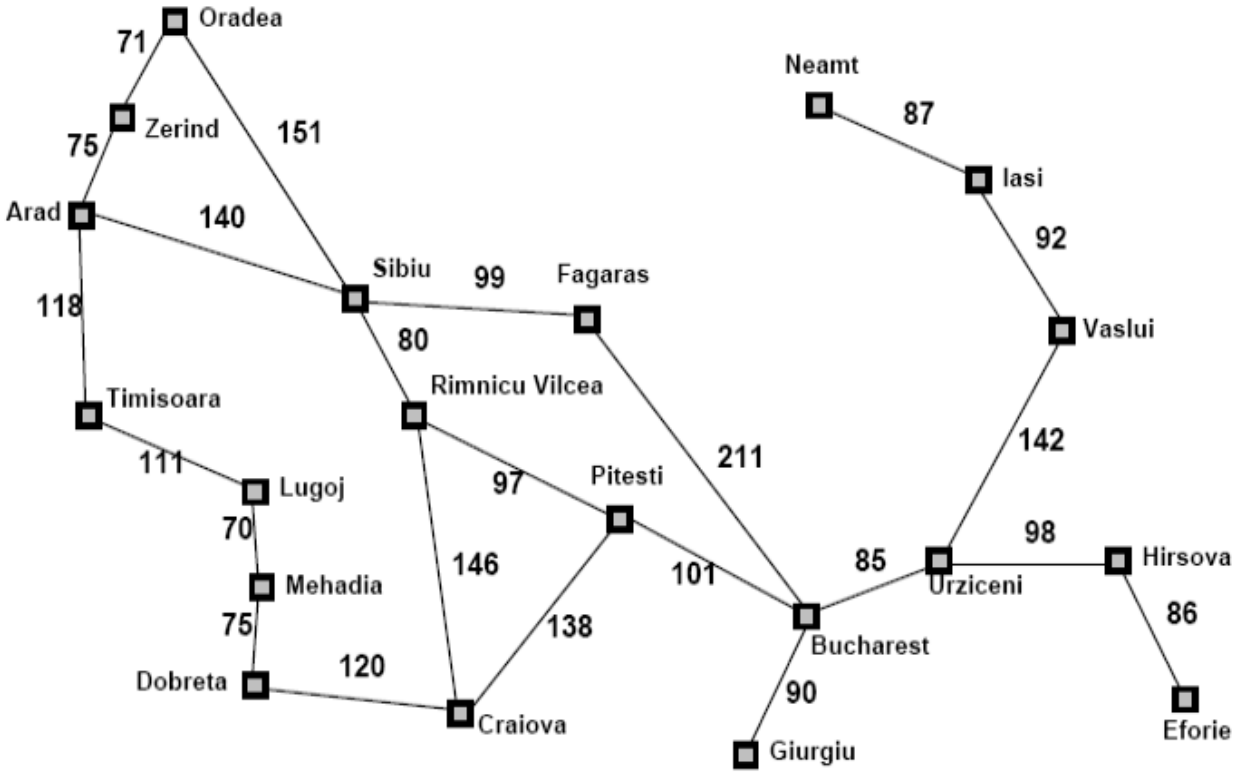
- Special cases: greedy search, A* search

Best-first search

```
Best-first search {

closed list = [ ]
open list = [start node]

    do {
            if open list is empty then{
                    return no solution
            }
            n = heuristic best node
            if n == final node then {
                    return path from start to goal node
            }
            foreach direct available node do{
                    if node not in open and not in closed list do {
                            add node to open list
                            set n as his parent node
            }
            delete n from open list
            add n to closed list
    } while (open list is not empty)
}
```

- Best-first search is an instance of the general TREE-SEARCH or GRAPH-SEARCH algorithm in which a node is selected for expansion based on an evaluation function.

Romania with step costs in km



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 search

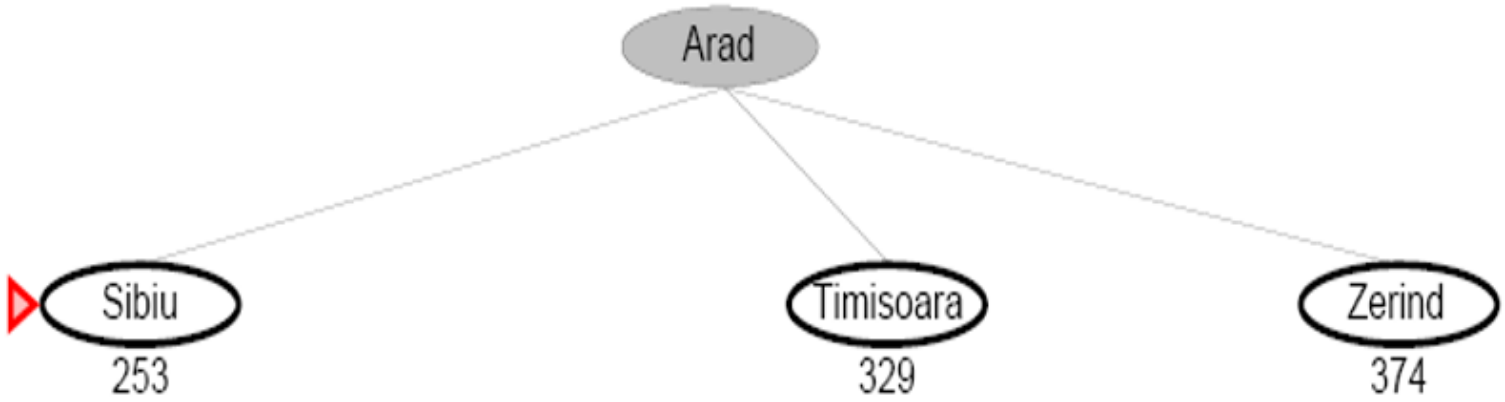
Evaluation function $h(n)$ (heuristic function 启发函数)
= estimate of cost from n to the closest goal
(节点 n 到目标节点的最低耗散路径的耗散估计值)

E.g., $h_{SLD}(n)$= straight-line distance from n to Bucharest

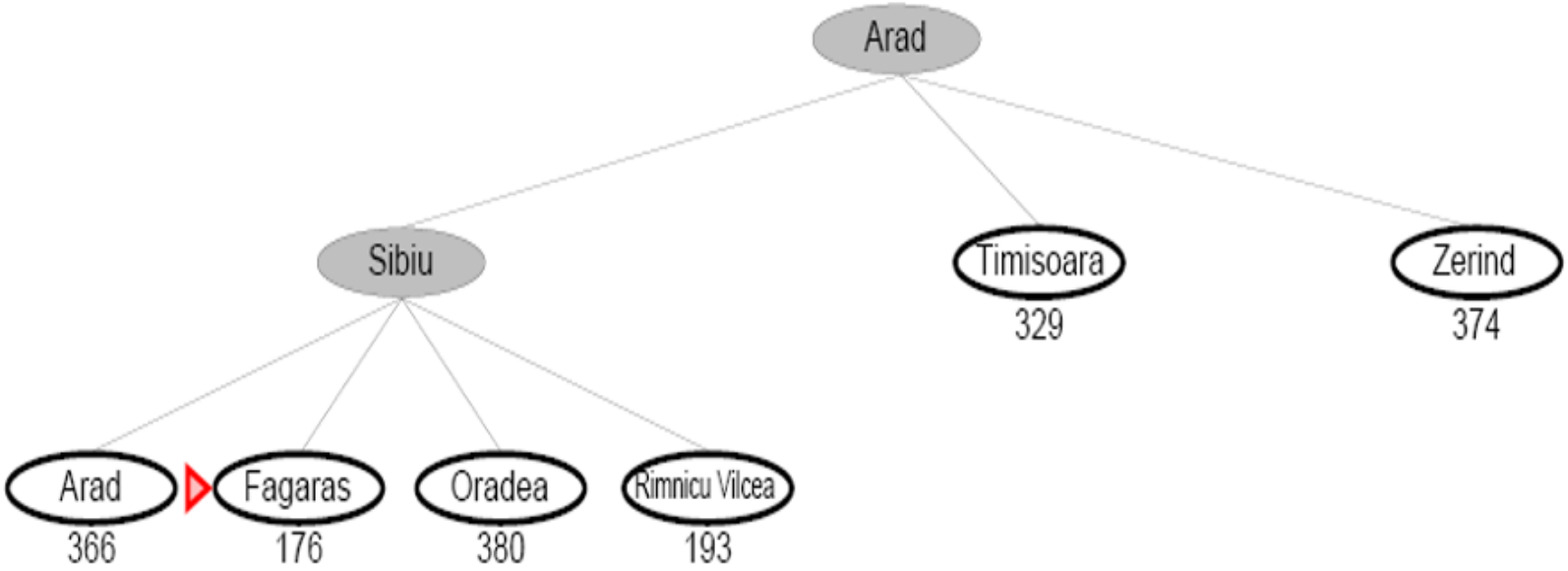
Greedy search expands the node that appears to be closest to goal (试图扩展离目标节点最近的点)

Greedy search example

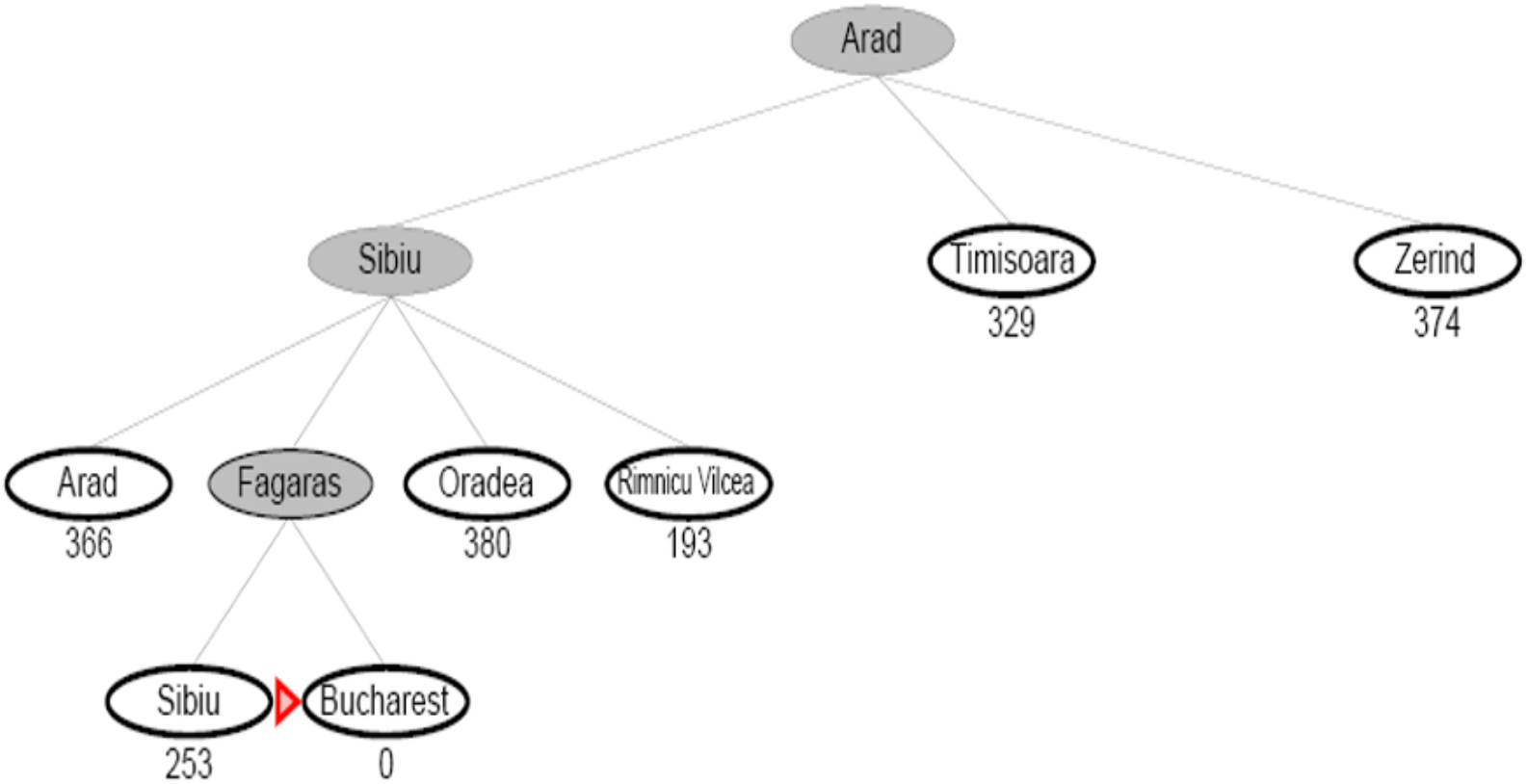
Greedy search example



Greedy search example



Greedy search example



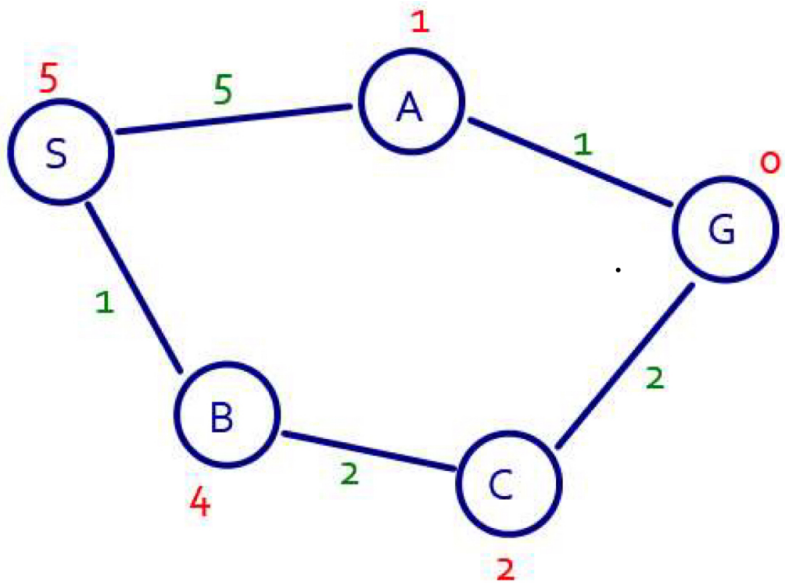
Properties of greedy search

complete? No — can get stuck in loops, e.g. from Iasi to Fagaras,
Iasi→ Neamt→Iasi→ Neamt →
Complete in finite space with repeated-state checking

Time? $O(b^m)$, but a good heuristic can give dramatic improvement

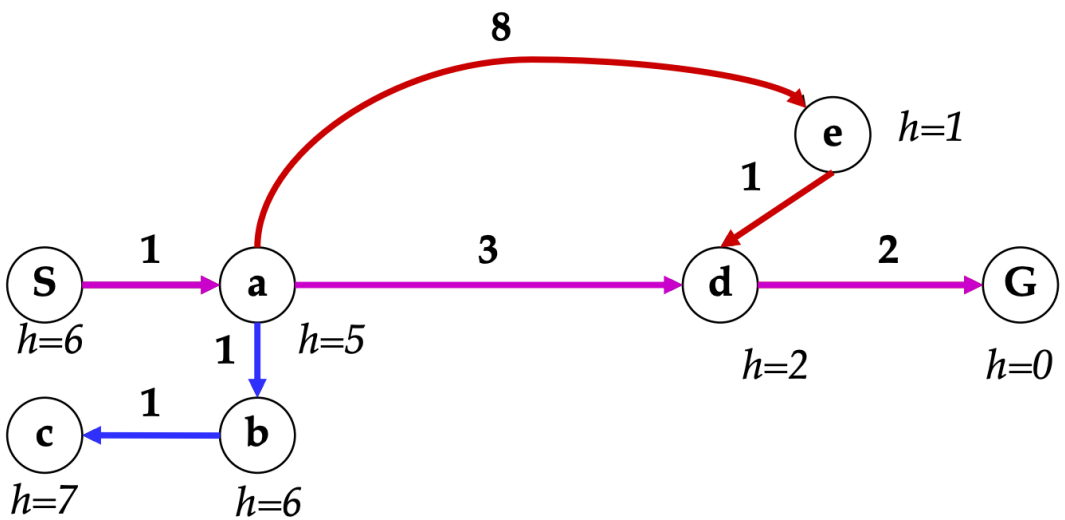
Space? $O(b^m)$ — keeps all nodes in memory
b: Branch factor, d: Solution depth, m: Maximum depth

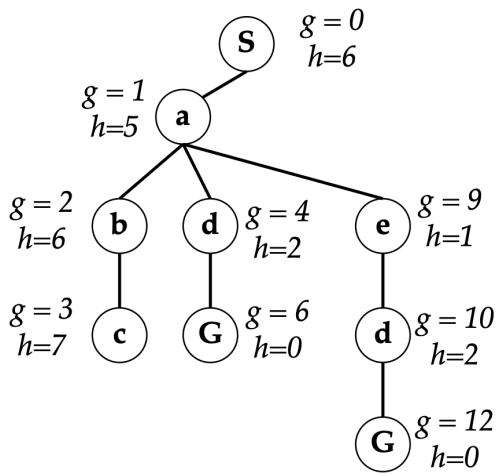
Optimal? No



Combining UCS and Greedy

- Uniform-cost orders by path cost, or backward cost $g(n)$
- Greedy orders by goal proximity, or forward cost $h(n)$
- A* Search orders by the sum: $f(n) = g(n) + h(n)$





A* search

- 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
- A* search uses an admissible heuristic 可采纳启发式
  i.e., $h(n) \leq h^*(n)$ where $h^*(n)$ is true cost from n
  (also require $h(n) \geq 0$, so $h(G) = 0$ for any goal G)
- *e.g.*, $h_{SLD}(n)$ never overestimates the actual road distance
  (SLD: Straight-Line Distance)

定理

A is optimal if $h(n)$ satisfies certain conditions.*

A* search

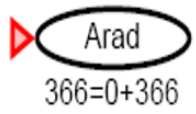
A* 算法的三个版本：

- The tree-search version
- The graph-search version
- The practical version

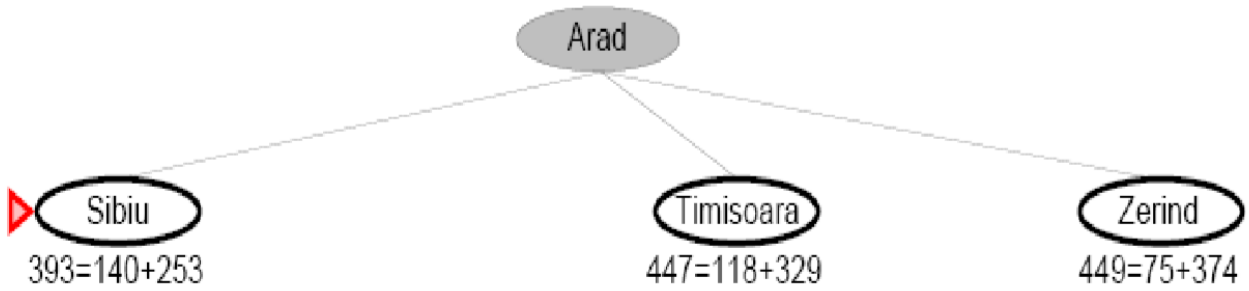
当有一条新路径访问旧节点 n 时，$g(n)$ 是否有可能更小，从而使 n 重新成为待扩展节点

```
A* search {

closed list = [ ]
open list = [start node]

    do {
        if open list is empty then {
            return no solution
        }
        n = heuristic best node
        if n == final node then {
            return path from start to goal node
        }
        foreach direct available node do{
            if current node not in open and not in closed list do {
                add current node to open list and calculate heuristic
                set n as his parent node
            }
            else{
                check if path from star node to current node is
                better;
                if it is better calculate heuristics and transfer
                current node from closed list to open list
                set n as his parrent node
            }
        delete n from open list
        add n to closed list
    } while (open list is not empty)

}
```

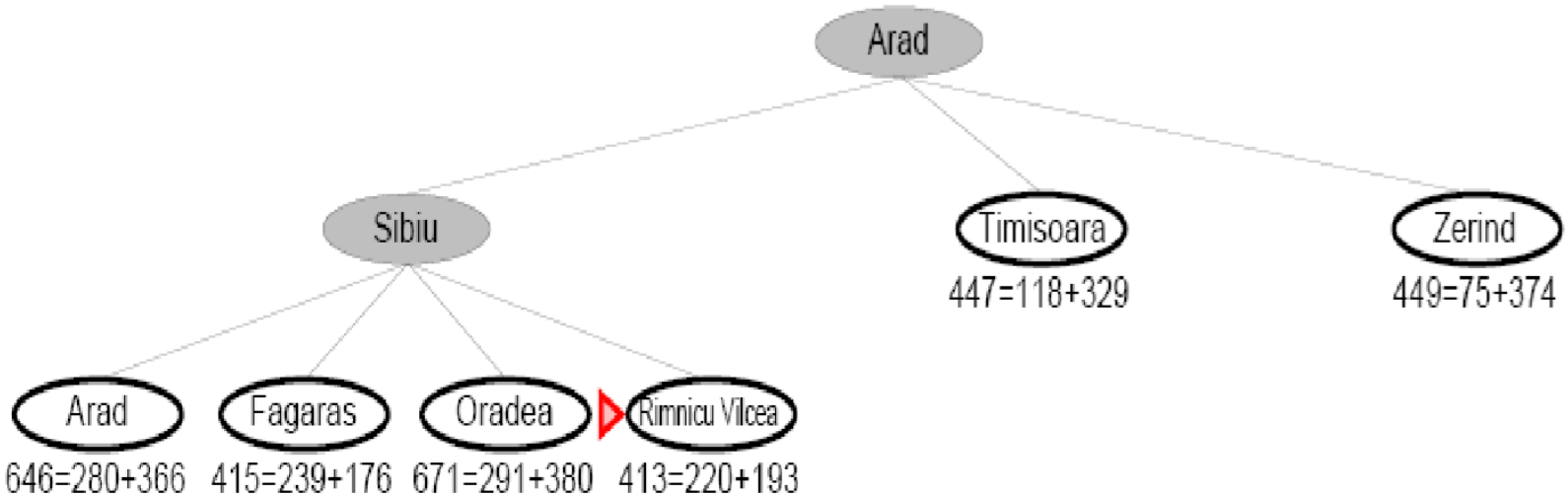
A* search example



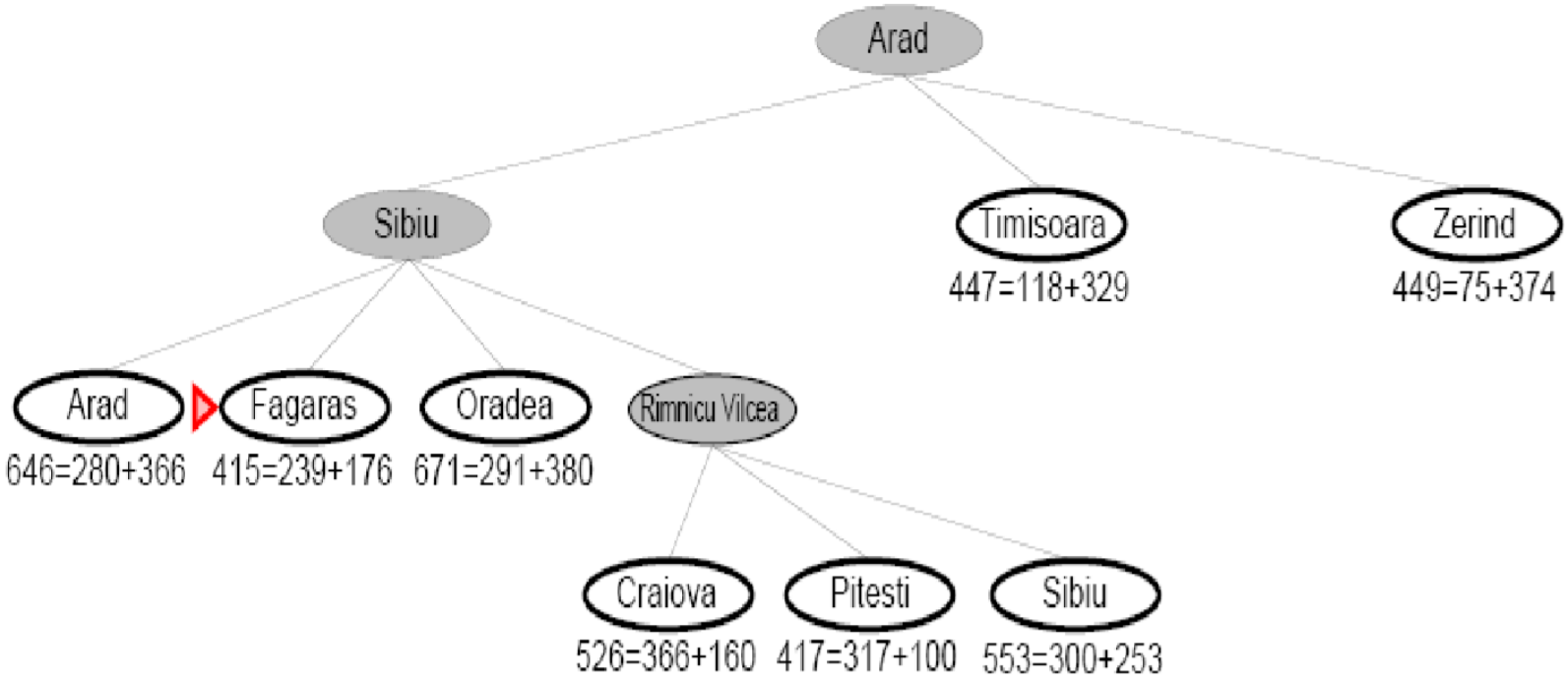
A* search example



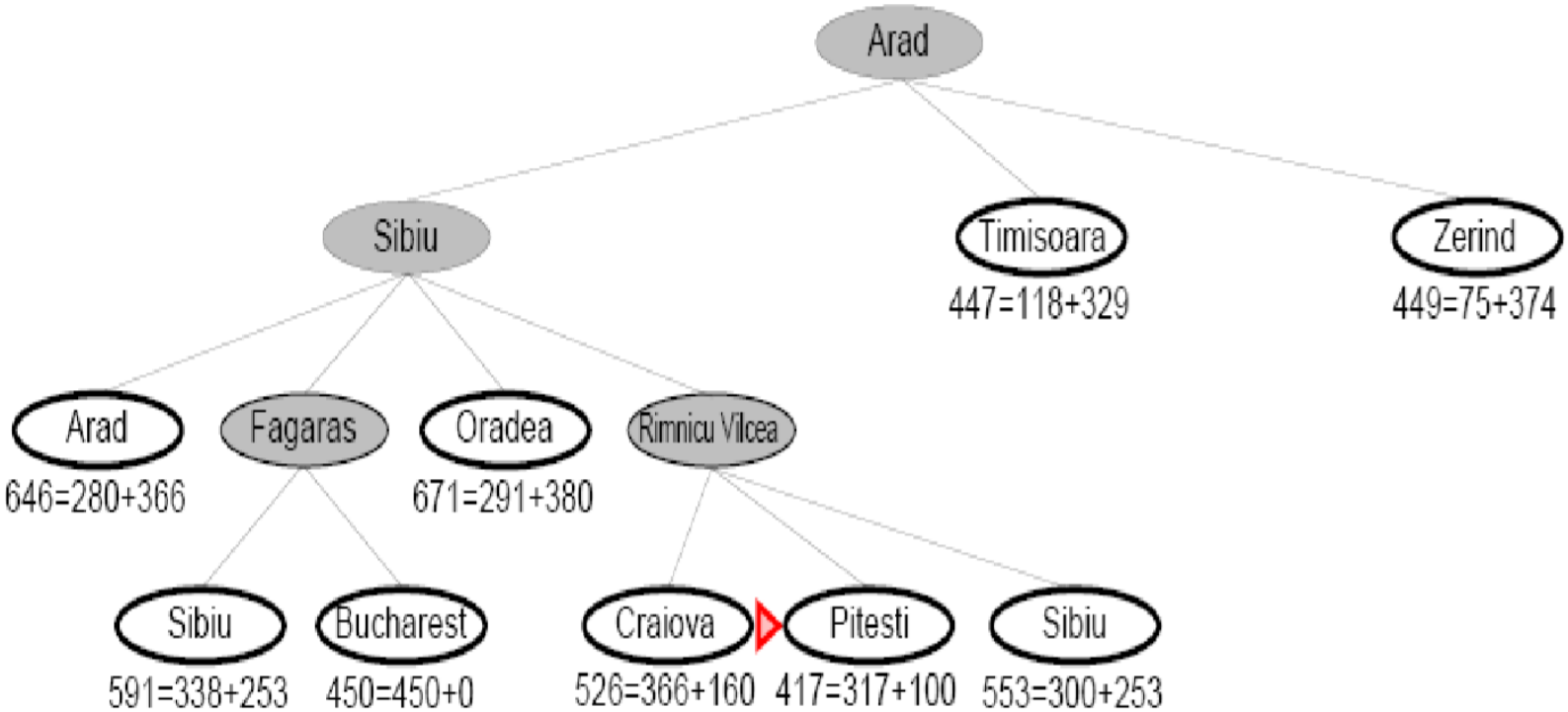
A* search example



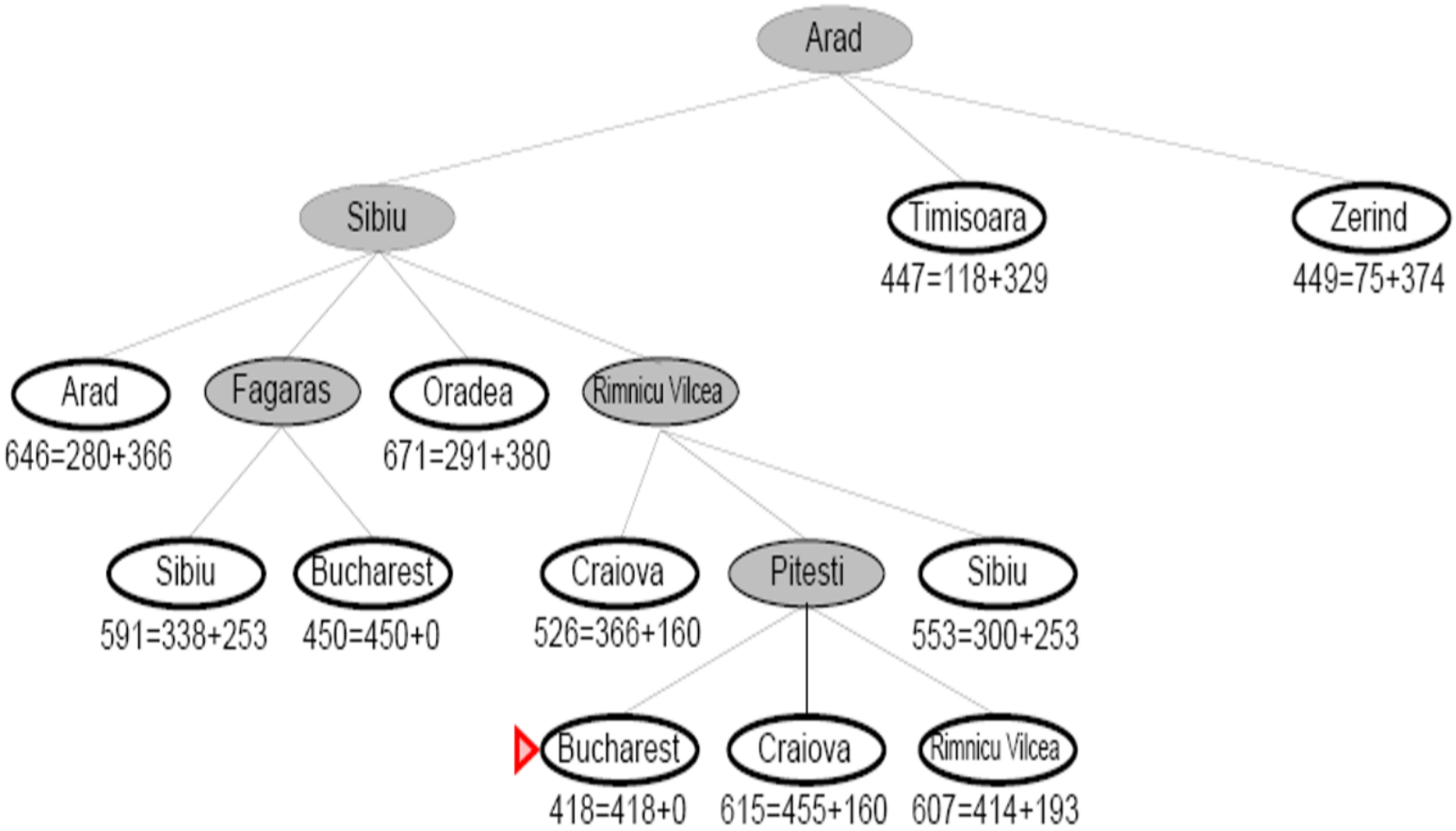
A* search example



A* search example



A* search example



Admissible heuristics

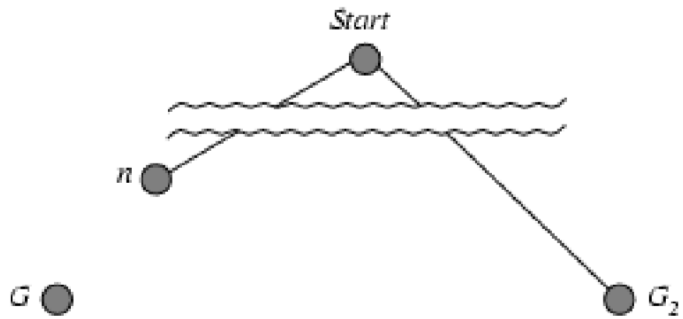
- A* heuristic $h(n)$ is admissible (**可采纳的**) if for every node n, $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from n
- An admissible heuristic never overestimates the cost to reach the goal (**从不会过高估计到达目标的耗散**), *i.e.*, it is optimistic (**乐观的**)
- Example: $h_{SLD}(n)$ (never overestimates the actual road distance)

定理

If $h(n)$ is admissible, A using TREE-SEARCH is optimal*

Optimality of A* (proof)

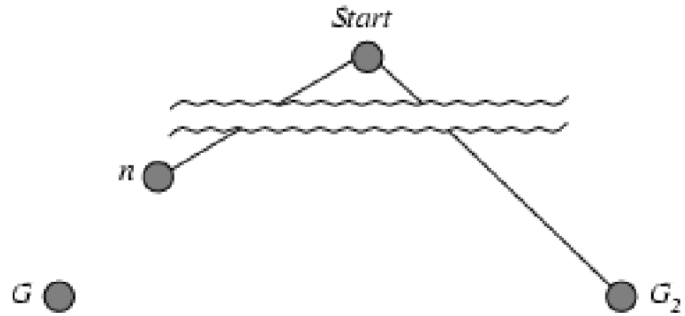
- Suppose some suboptimal (非最优) goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on a shortest path to an optimal goal G .



- $f(G_2) = g(G_2)$ since $h(G_2) = 0$
- $g(G_2) > g(G)$ since G_2 is suboptimal
- $f(G) = g(G)$ since $h(G) = 0$
- $f(G_2) > f(G)$ from above

Optimality of A* (proof)

- Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let n be an unexpanded node in the fringe such that n is on a shortest path to an optimal goal G



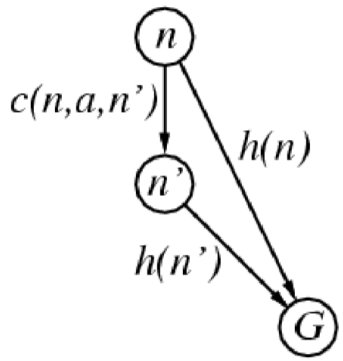
- $f(G_2) > f(G)$ from above
- $h(n) \leq h^*(n)$ since h is admissible
- $g(n) + h(n) \leq g(n) + h^*(n) \leq g(G) = f(G)$
  n is on a shortest path to an optimal goal G
- $f(n) \leq f(G)$

Hence $f(G_2) > f(n)$, and A* will never select G_2 for expansion

Consistent heuristics

- A* heuristic is consistent (一致) if for every node n, every successor n' of n generated by any action a,
  $h(n) \leq c(n, a, n') + h(n')$
- Consistency implies admissibility!

- If h is consistent, we have
  $f(n') = g(n') + h(n')$
  $= g(n) + c(n, a, n') + h(n')$
  $\geq g(n) + h(n)$
  $= f(n)$



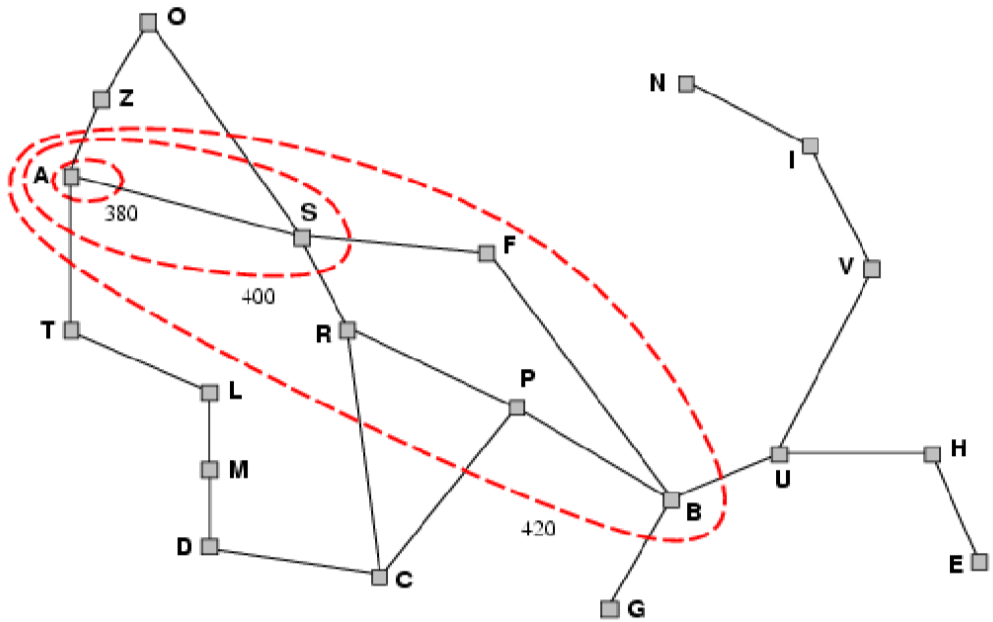
- *i.e.*, f(n) is non-decreasing along any path.

定理

If h(n) is consistent, A using GRAPH-SEARCH is optimal*

Optimality of A* (proof)

- A* expands nodes in order of increasing f value
- Gradually adds "f-contours (等值线)" of nodes
- Contour i has all nodes with $f = f_i$, where $f_i < f_{i+1}$



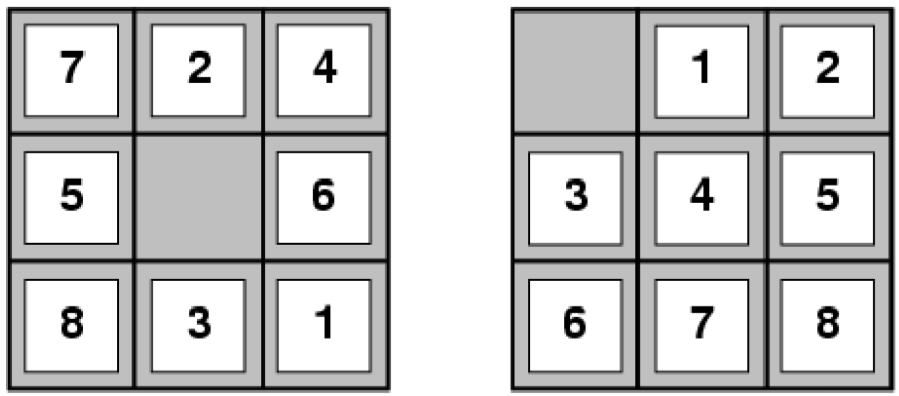
Properties of A*

Complete?	Yes (unless there are infinitely many nodes with $f \leq f(G)$)
Time?	A* 算法对于任何给定的启发函数都是效率最优的 But still exponential
Space?	Keeps all nodes in memory
Optimal?	Yes

- A* expands all nodes with f(n) $<$ C*
- A* expands some nodes with f(n) $=$ C*
- A* expands no nodes with f(n) $>$ C*

8-puzzle revisited

- 8-puzzle–把棋子水平或者竖直地滑动到空格中，直到目标局面:

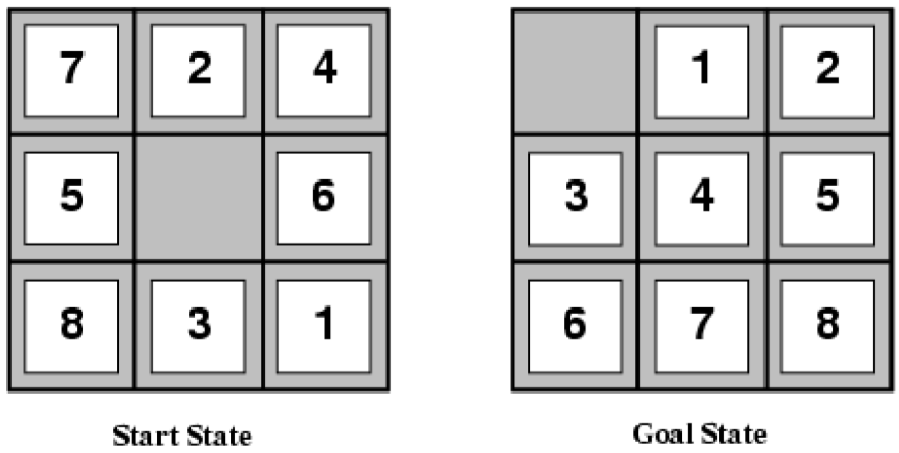


- 平均解步数是 22 步。分支因子约为 3
  - 到达深度为 22 的穷举搜索将考虑约 $3^{22} \approx 3.1x10^{10}$
  - 状态个数 $O((n+1)!)$, NP 完全问题

Admissible heuristics

For the 8-puzzle:

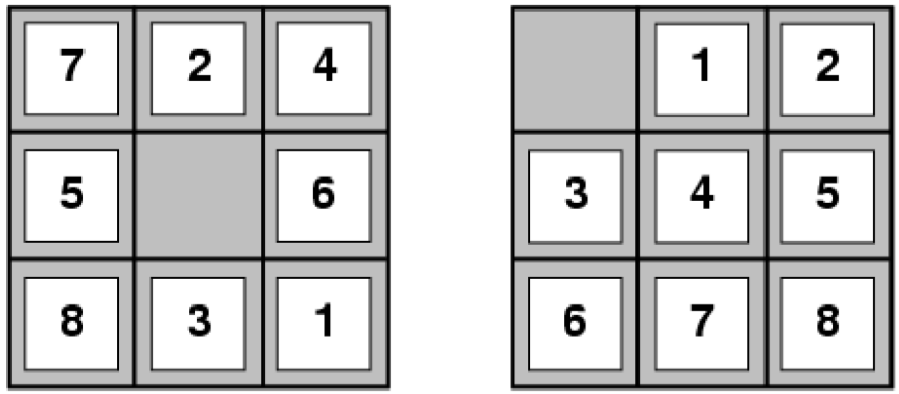
- $h_1(n)$ = number of misplaced tiles (错位的棋子数)
- $h_2(n)$ = total Manhattan distance (所有棋子到其目标位置的水平竖直距离和) (*i.e.*, no. of squares from desired location of each tile)



- $h_1(S)$ =?
- $h_2(S)$ = ?

Admissible heuristics

- $h_1(n)$ = number of misplaced tiles (错位的棋子数)
- $h_2(n)$ = total Manhattan distance (所有棋子到其目标位置的水平竖直距离和) (*i.e.*, no. of squares from desired location of each tile)



Start State　　Goal State

- $h_1(S)$ =8
- $h_2(S)$ =3+1+2+2+2+3+3+2 = 18

Dominance

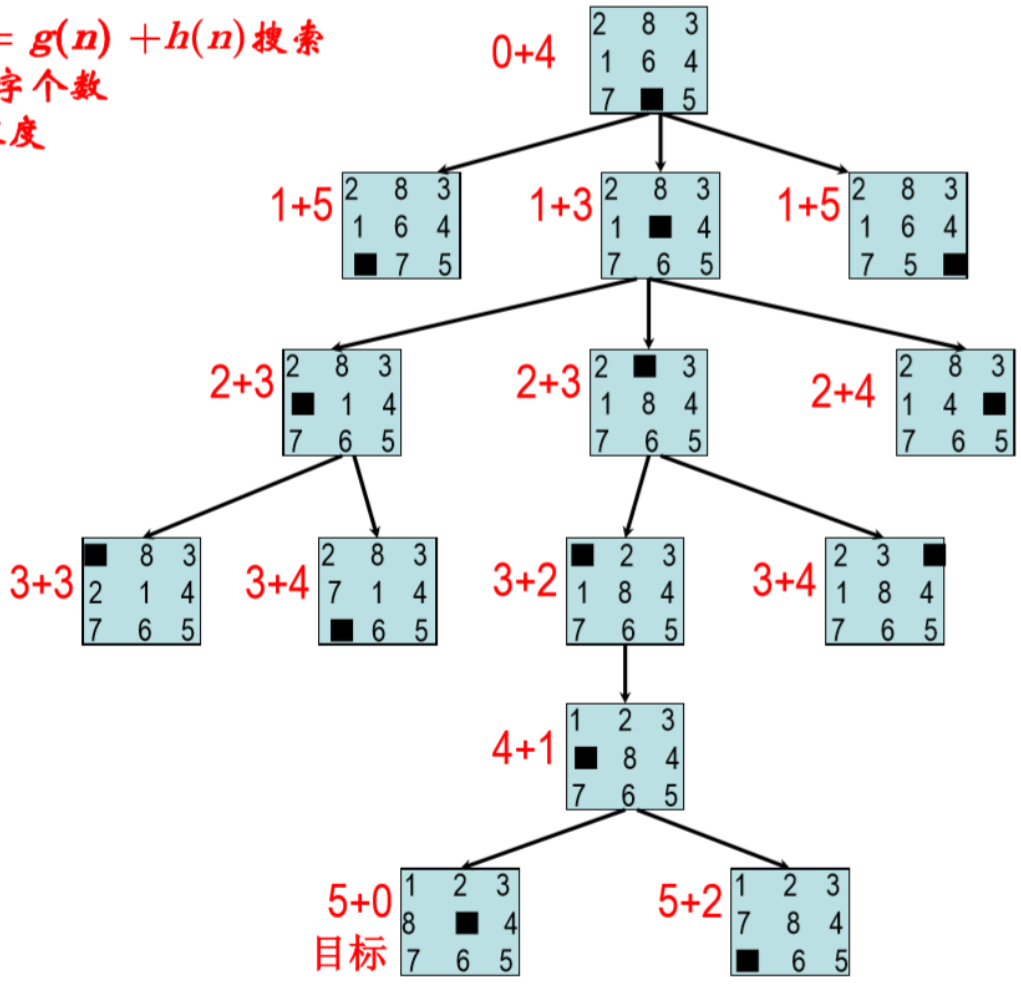
For the 8-puzzle:

- If $h_2(n) \geq h_1(n)$ for all n (both admissible)
- then h_2 dominates h_1 (dominate 统治、占优)
- h_2 is better for search
- Typical search costs (average number of nodes expanded):
- d=12
  - $IDS = 3,644,035$ nodes
  - $A^*(h_1) = 227$ nodes
  - $A^*(h_2) = 73$ nodes
- d=24
  - IDS = too many nodes
  - $A^*(h_1) = 39135$ nodes
  - $A^*(h_2) = 1641$ nodes

Given any admissible heuristics h_a, h_b
$h(n) = max(h_a(n); h_b(n))$ is also admissible and dominates h_a, h_b

8-puzzles

使用 $f(n) = g(n) + h(n)$ 搜索
$h(n)$=错放数字个数
$g(n)$ = 节点深度



Relaxed problems

- A problem with fewer restrictions on the actions is called a relaxed problem (松弛问题)
- The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem
  一个松弛问题的最优解的耗散是原问题的一个可采纳的启发式
- If the rules of the 8-puzzle are relaxed so that a tile can move anywhere, then $h_1(n)$ gives the shortest solution
  如果棋子可以任意移动，则 h_1 给出最短的确切步数
- If the rules are relaxed so that a tile can move to any adjacent square, then $h_2(n)$ gives the shortest solution
  如果棋子可以移动到任意相邻的位置，则 h_2 给出最短的确切步数

Key point: the optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the real problem

Relaxed problems

- 构造松弛问题
  - 原问题：一个棋子可以从方格 A 移动到方格 B，如果 A 与 B 水平或者垂直相邻而且 B 是空的
  - 松弛 1：一个棋子可以从方格 A 移动到方格 B，如果 A 与 B 相邻 — h_2
  - 松弛 2：一个棋子可以从方格 A 移动到方格 B，如果 B 是空的
  - 松弛 3：一个棋子可以从方格 A 移动到方格 B — h_1

- 如果有一个可采纳启发式的集合 $\{h_1, \ldots, h_m\}$ $h(n) = max(h_1(n), \ldots, h_m(n))$ 可采纳并比成员启发式更有优势

Evaluation Function $f(n)$

- $h(n)$ — heuristic, estimate of cost from n to the closest goal (节点 n 到目标节点的最低耗散路径的耗散估计值)
- $g(n)$ —– path cost to n (初始节点到这个节点的路径损耗的总和)

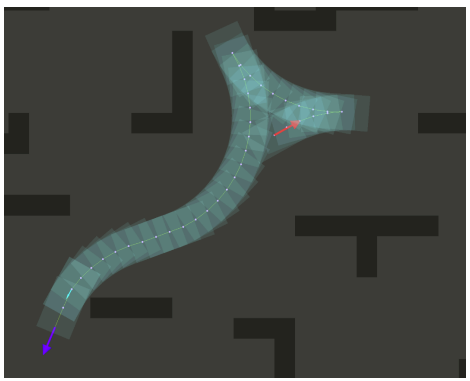
Possible evaluation functions:

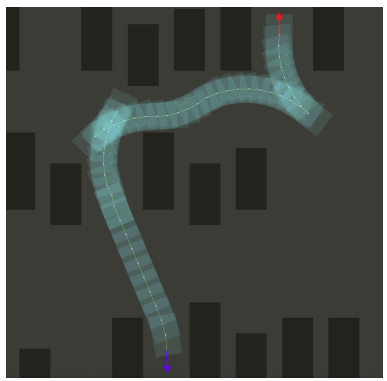
- $f(n) = g(n)$: Uniform Cost
- $f(n) = h(n)$: Greedy
- $f(n) = g(n) + h(n)$: A*

estimates the total cost of a solution path which goes through node n

Hybrid A*

- Hybrid A*: 车辆自主泊车所用的路径规划算法，以较少的换挡数，规划出可行驶轨迹使车达到目标位置
- 搜索状态 (x, y, θ, r), r 表示前进或者后退
- 行动离散化（固定角度转动，固定速度前进）max-left, no-turn, max-right
- 启发式函数
  - non-holonomic-without-obstacles: 忽略障碍物，但要满足可行驶约束（车头朝向连续变化），在格子地图中的最短路径距离
  - holonomic-with-obstacles: 考虑障碍物，但忽略可行驶约束，在格子地图中的最短路径距离
- 最后再做轨迹平滑和优化





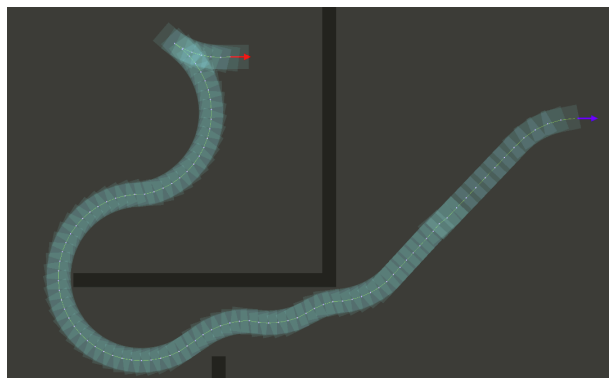


Table of Contents

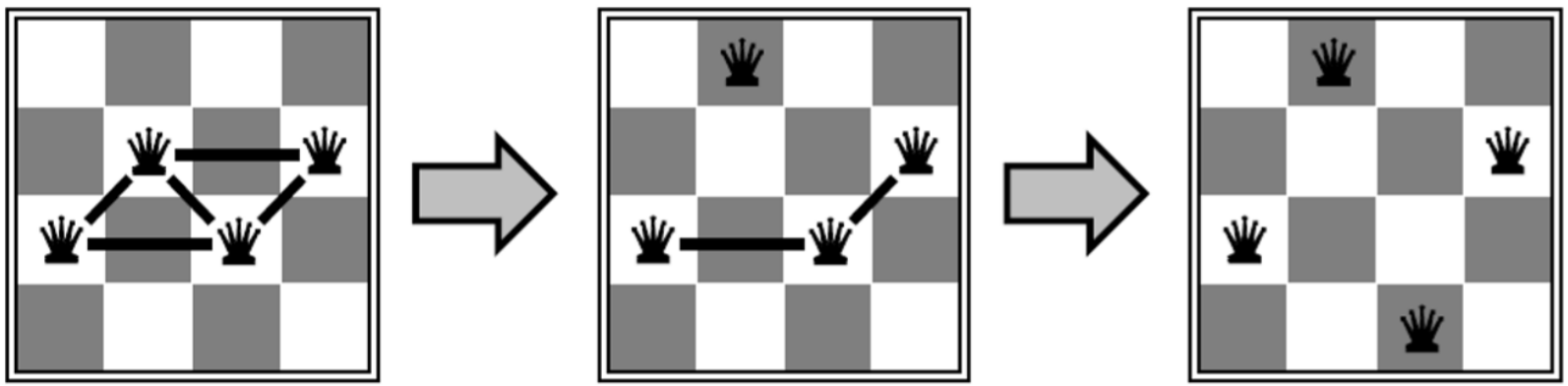
Local search algorithms

- In many optimization problems, the path to the goal is irrelevant; the goal state itself is the solution
- State space = set of "complete" configurations (完全状態)
  — Find configuration satisfying constraints, *e.g.*, n-queens

- In such cases, we can use local search algorithms
- keep a single "current" state, try to improve it
- Constant space, suitable for online as well as offline search

Example: n-queens

- Put n queens on an $n \times n$ board with no two queens on the same row, column, or diagonal



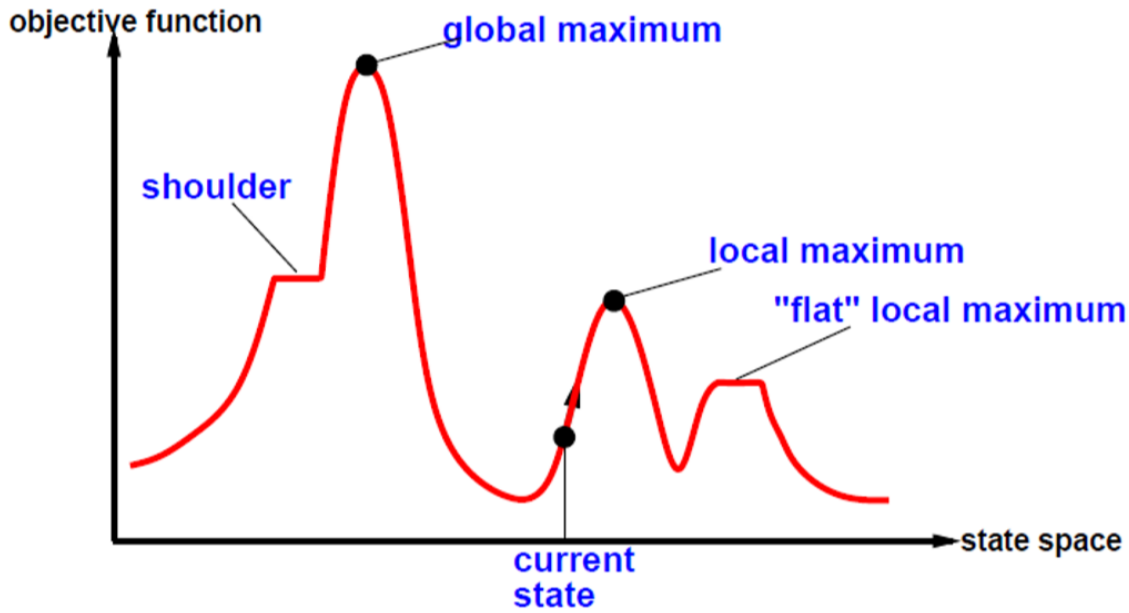
Hill-climbing search

- "Like climbing Everest in thick fog with amnesia (健忘症)"

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    inputs: problem, a problem
    local variables: current, a node
                     neighbor, a node

    current ← MAKE-NODE(INITIAL-STATE[problem])
    loop do
        neighbor ← a highest-valued successor of current
        if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
        current ← neighbor
```

Hill-climbing search

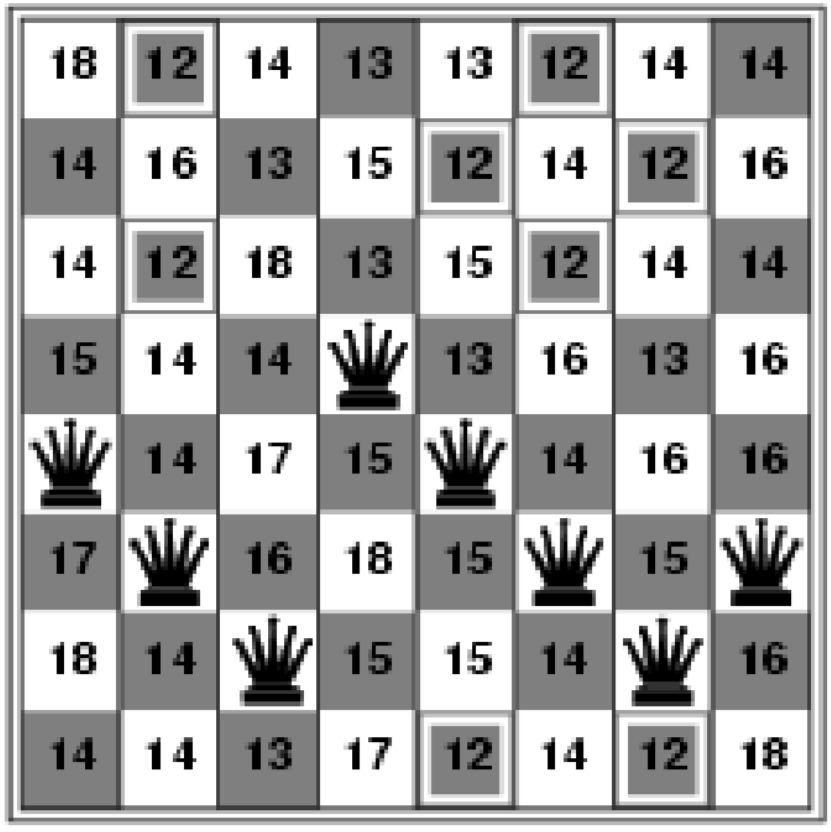
- Problem: depending on initial state, can get stuck in local maxima (局部最大値)



Random-restart hill climbing overcomes local maxima — trivially complete

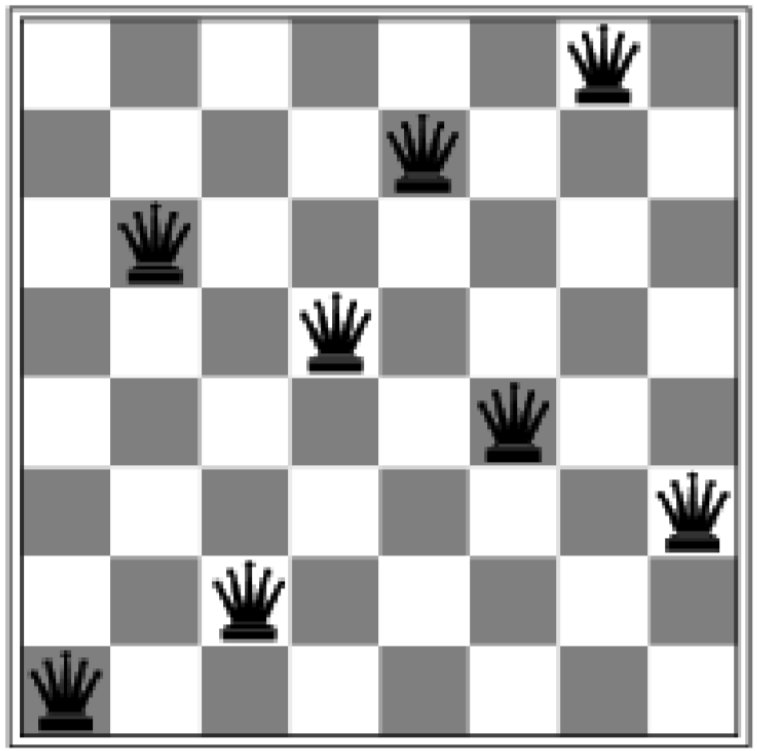
Random sideways moves escape from shoulders loop on at maxima

Hill-climbing search: 8-queens problem



- $h =$ number of pairs of queens that are attacking each other, either directly or indirectly
- $h = 17$ for the above state

Hill-climbing search: 8-queens problem



- A local minimum with $h = 1$

Simulated annealing search (模拟退火搜索)

- Idea: escape local maxima by allowing some "bad" moves but gradually decrease their frequency

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  inputs: *problem*, a problem
    schedule, a mapping from time to "temperature"
  local variables: *current*, a node
    next, a node
    T, a "temperature" controlling prob. of downward steps

  current $\leftarrow$ MAKE-NODE(INITIAL-STATE[*problem*])
  for $t \leftarrow 1$ **to** ∞ **do**
    $T \leftarrow$ *schedule*[t]
    if $T = 0$ **then return** *current*
    next $\leftarrow$ a randomly selected successor of *current*
    $\Delta E \leftarrow$ VALUE[*next*] – VALUE[*current*]
    if $\Delta E > 0$ **then** *current* $\leftarrow$ *next*
    else *current* $\leftarrow$ *next* only with probability $e^{\Delta\ E/T}$

Properties of Simulated Annealing Search

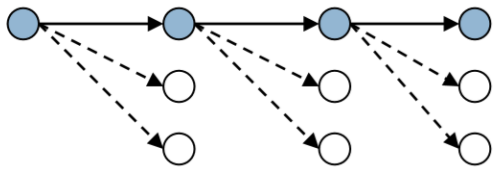
- One can prove: If T decreases slowly enough, then simulated annealing search will find a global optimum with probability approaching 1
- Widely used in VLSI layout (超大规模集成电路布局), airline scheduling, etc

Local beam search (局部剪枝搜索)

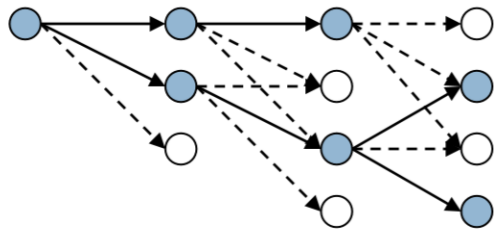
1. Keep track of k states rather than just one
2. Start with k randomly generated states
3. At each iteration, all the successors of all k states are generated
4. If any one is a goal state, stop; else select the k best successors from the complete list and repeat.

Local beam search (局部剪枝搜索)

- Like greedy search, but keep k states at all times:



Greedy Search



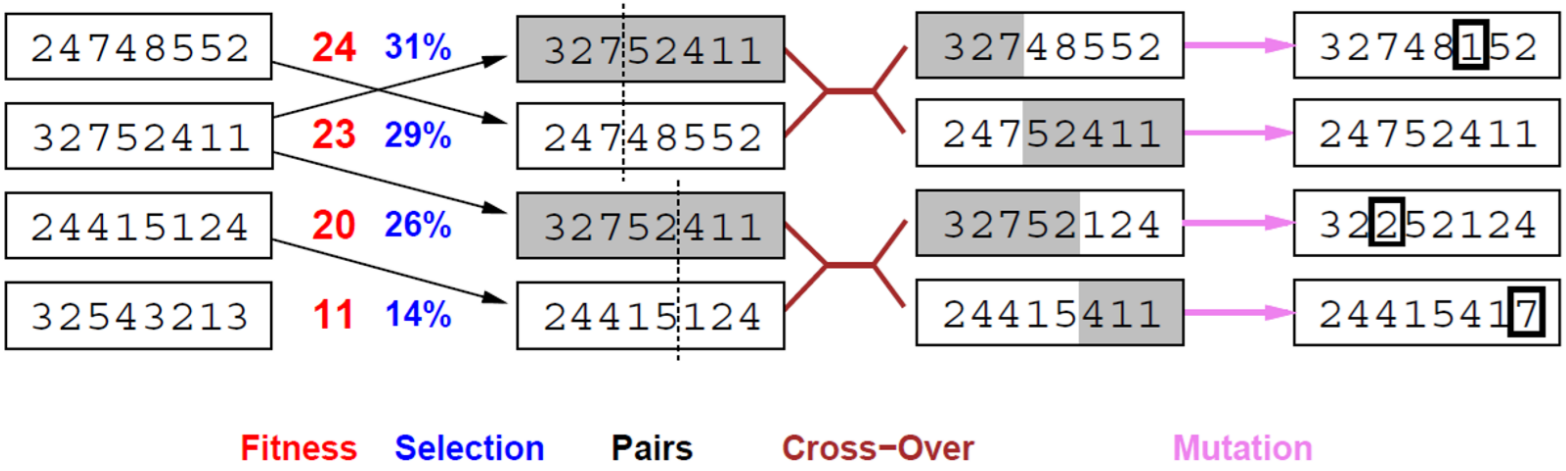
Beam Search

- Not the same as k searches run in parallel!
- Searches that find good states recruit other searches to join them
- Problem: quite often, all k states end up on same local hill
- Idea: choose k successors randomly, biased towards good ones

Genetic algorithms (遗传算法)

- A successor state is generated by combining two parent states
- Start with k randomly generated states (population 种群)
- A state is represented as a string over a finite alphabet (often a string of 0s and 1s)
- Evaluation function (fitness function 适应度函数). Higher values for better states
- Produce the next generation of states by selection, crossover, and mutation (选择, 杂交, 变异)

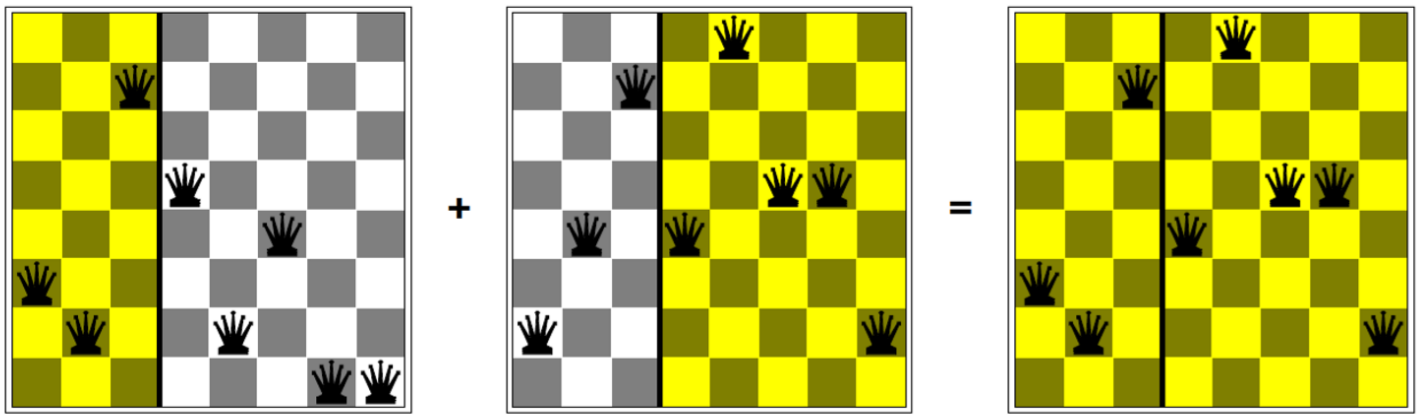
Genetic algorithms



- Fitness function: number of non-attacking pairs of queens 不互相攻击的皇后数目 ($min = 0$, $max = \frac{8\times7}{2} = 28$)

  $24/(24+23+20+11) = 31\%$
  $23/(24+23+20+11) = 29\%$ etc

Genetic algorithms



Summary

- Heuristic functions estimate costs of shortest paths
- Good heuristics can dramatically reduce search cost
- Greedy best-first search expands lowest h
  - incomplete and not always optimal
- A* search expands lowest $g + h$
  - complete and optimal
  - also optimally efficient (up to tie-breaks, for forward search)
- Admissible heuristics can be derived from exact solution of relaxed problems

Summary

- Local search algorithms:
  the path to the goal is irrelevant; the goal state itself is the solution keep a single "current" state, try to improve it
  - Hill-climbing search:
    depending on initial state, can get stuck in local maxima
  - Simulated annealing search:
    escape local maxima by allowing some "bad" moves but gradually decrease their frequency
  - Local beam search:
    keep track of k states rather than just one
  - Genetic algorithms

Uninformed/Informed Search

- 好的启发式搜索能大大提高搜索性能
- 但由于启发式搜索需要抽取与问题本身有关的特征信息，而这种特征信息的抽取有时会比较困难，因此盲目搜索仍不失为一种有用的搜索策略。

Uninformed/Informed Search

- 好的搜索策略应该
  - 引起运动—— 避免原地踏步
  - 系统—— 避免兜圈
  - 运用启发函数—— 缓解组合爆炸
- 搜索树 vs 搜索图
  - 搜索树：结点有重复，但登记过程简单
  - 搜索图：结点无重复，但登记过程复杂 (每次都要查重)

作业

第二版书：

- 4.1, 4.2, 4.6, 4.7