Foundation of Artificial Intelligence

Prof. Fangshi Wang

Beijing Jiaotong University

Email: fshwang@bjtu.edu.cn

3.4 Beyond Classical Search
（超越经典搜索）

3.4.1 Hill-climbing search（爬山搜索）

3.4.2 Simulated annealing search（模拟熄火搜索）

3.4.3 Genetic algorithms（遗传算法）

◆前面介绍过的搜索算法都系统地探索空间。这种系统化通过在内存中保留一条或多条路径和记录路径中的每个结点的选择。

◆当找到目标时，到达此目标的路径就是这个问题的一个解。

◆然而在许多问题中，并不关注到达目标的路径。例如，在八皇后问题中，重要的是最终皇后在棋盘上的布局，而不是皇后加入的先后次序。

◆许多重要的应用都具有这样的性质，例如集成电路设计、工厂场地布局、作业车间调度、自动程序设计，电信网络优化、车辆寻径和文件夹管理。

◆ 若到目标的路径是无关紧要的，我们考虑不关心路径的算法。

◆**局部搜索算法**从单个当前结点（而不是多条路径）出发，通常只移动到它的邻近状态。一般情况下不保留搜索路径。

局部搜索

- **局部搜索的基本思想**：在搜索过程中，始终向着离目标最接近的方向搜索。
- **目标**可以是最大值，也可以是最小值
- 局部搜索算法有如下两个主要**优点**：
  - 使用很少的内存；
  - 在大的或无限（连续）状态空间中，能发现合理的解。
- 局部搜索算法：
  - Hill-climbing search（爬山搜索）
  - Simulated annealing search（模拟熄火搜索）
  - Genetic algorithms（遗传算法）

3.4.1 爬山法

- **爬山法是最基本的局部搜索技术**。爬山法(最陡上升版本)搜索，是简单的循环过程，不断向值增加的方向持续移动—— 即，登高。
- 算法在到达一个“峰顶”时终止，邻接状态中没有比它值更高的。
- 算法不维护搜索树，因此当前结点的数据结构只需要记录当前状态和目标函数值。
- 爬山法不会考虑与当前状态不相邻的状态。这就像健忘的人在大雾中试图登顶珠穆朗玛峰一样。
- 在每一步，当前结点都会被它的最佳邻接结点所代替；
- 这里，最佳邻接结点就是**启发式代价评估函数h值**最低的邻接结点。

（1）Hill-climbing search

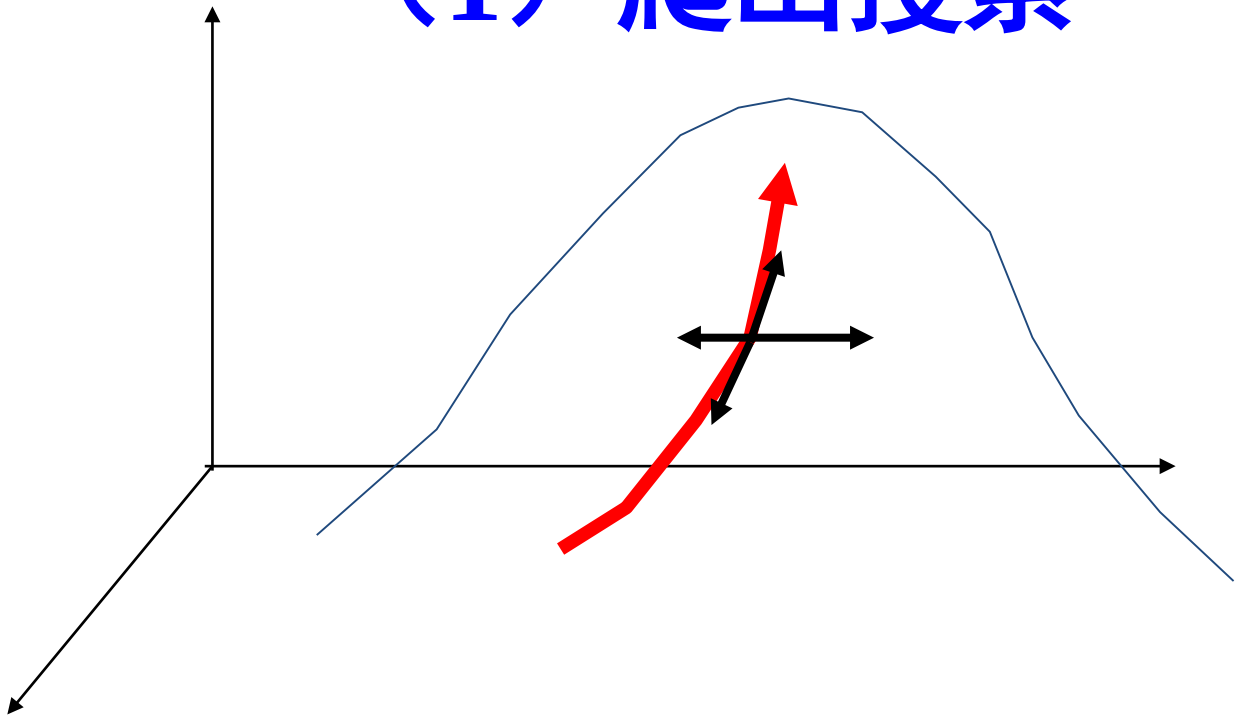
```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    inputs: problem, a problem
    local variables: current, a node
                     neighbor, a node

    current ← MAKE-NODE(INITIAL-STATE[problem]);
    loop do
        neighbor ← a highest-valued successor of current ; // 找到值最大的相邻结点
        if VALUE[neighbor] ≤ VALUE[current] then return STATE[current];
          // 若当前结点的值大于所有的相邻结点，说明已到达最高峰，当前结点即目标，则返回当前结点。
        current ← neighbor; // 否则，说明该相邻结点比当前结点好，用它代替当前结点，进行下一次循环。
```

注：Value 就是启发函数 h

3.4.1 爬山法

- 爬山法是一种**迭代**算法：开始时选择问题的一个任意解，然后递增地修改该解的一个元素，若得到一个更好的解，则将该修改作为新的解；重复上述步骤直到无法找到进一步的改善。
- 爬山法有时被称为**贪婪局部搜索**，因为它总是选择邻居中状态**最好**的一个，而不考虑下一步该如何走。
- 爬山法往往很有效，能很快地朝着解（目标状态）的方向进展，因为它可以很容易地改善一个不良状态。

（1）爬山搜索

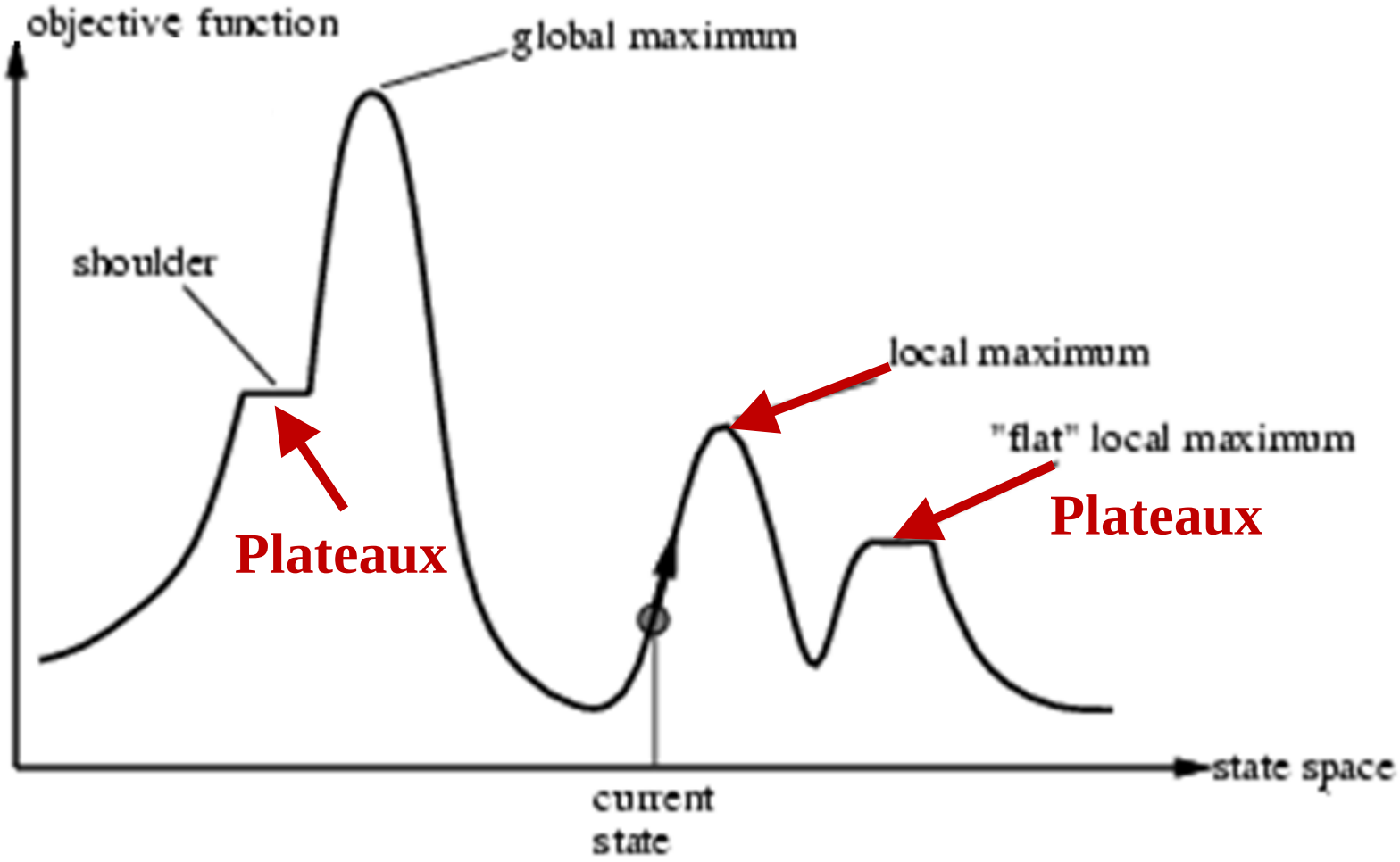


如果把山顶作为目标，h(n)表示当前位置n与山顶之间的高度差，则该算法相当于总是登向山顶。在单峰的条件下，必能到达山顶。

爬山法的弱点（1）

它在如下三种情况下经常被困：

（1）**Local maxima** 局部最大值：是一个比它的每个邻接结点都高的峰顶，但是比全局最大值要小。爬山法到达局部极大值附近，就会被拉向峰顶，然后就卡在局部极大值处无处可走。



（2）**Plateaux** 高原：是一块平原区域，是平的局部极大值，不存在上山的出口。

爬山法的弱点（2）

◆ **Ridges** 山脊：结果是一系列局部最大值，非常难爬行。

◆为什么山脊会使爬山法困难？

图中的状态（黑色圆点）

叠加在从左到右上升的山

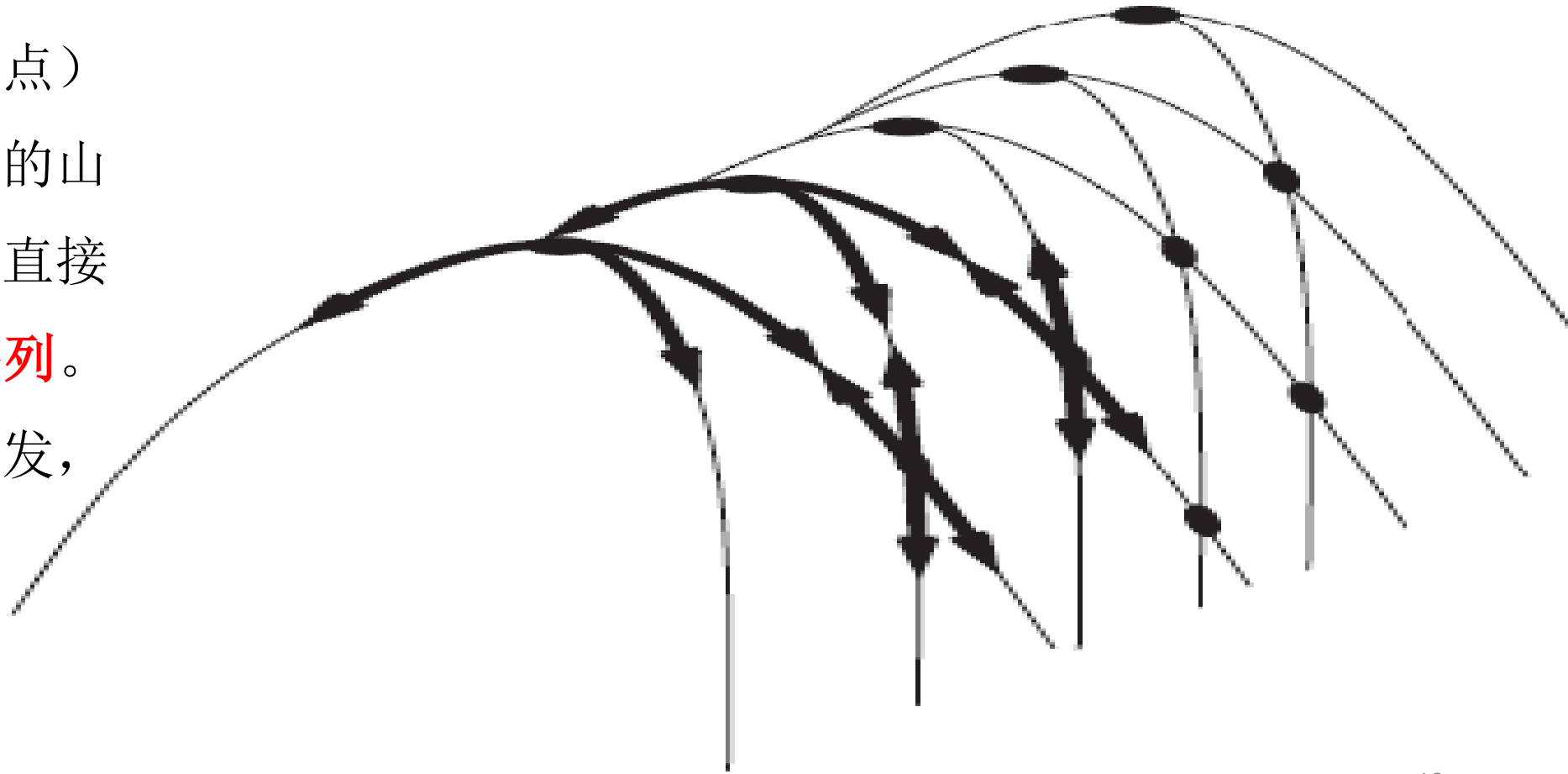
脊上，创造了一个不直接

相连的**局部极大值序列**。

从每个局部极大点出发，

可能的行动

都是指向**下山方向**的



用爬山法解决n皇后问题

◆将 n 个皇后放在n x n的棋盘上。每次移动一个皇后来减少冲突数量，使得没有两个皇后在同一行、同一列、或同一对角线上。

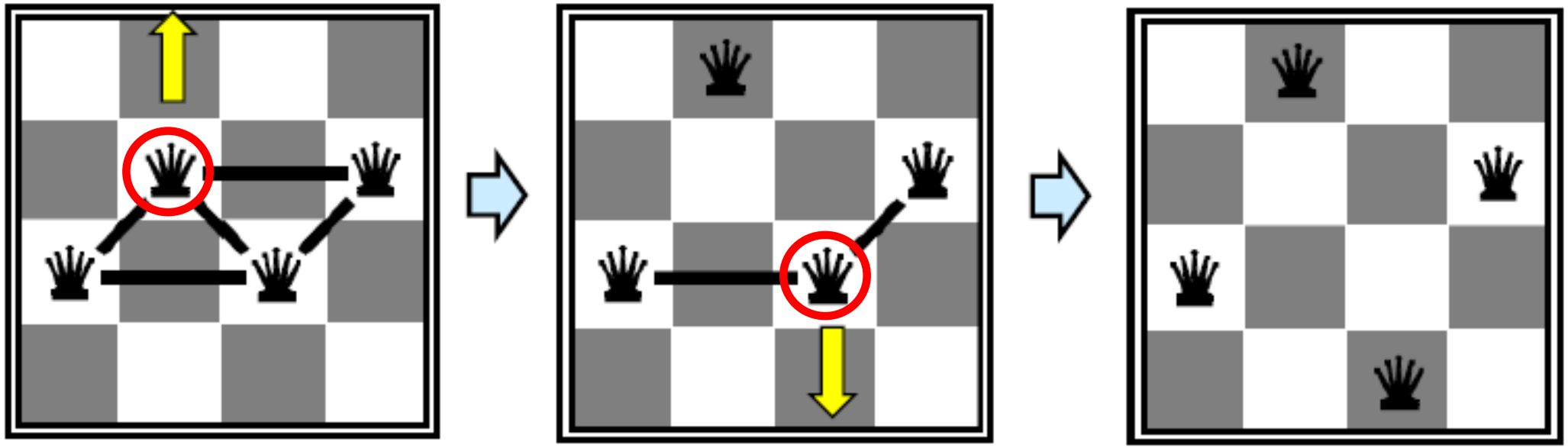
设任意两个皇后的坐标分别是（i，j）和（k，l），则这两个皇后在同一斜线上的充要条件是 |i-k|=|j-l|，即两个皇后的行号之差与列号之差的绝对值相等。

◆每个状态都是在棋盘上放置 *n* 个皇后，**每列有一个皇后**。

◆**启发式评估函数 h 是形成相互攻击的皇后对的数量**；不管是直接还是间接。

◆该函数的**全局最小值是h=0**，**仅在找到解时才会是这个值**。

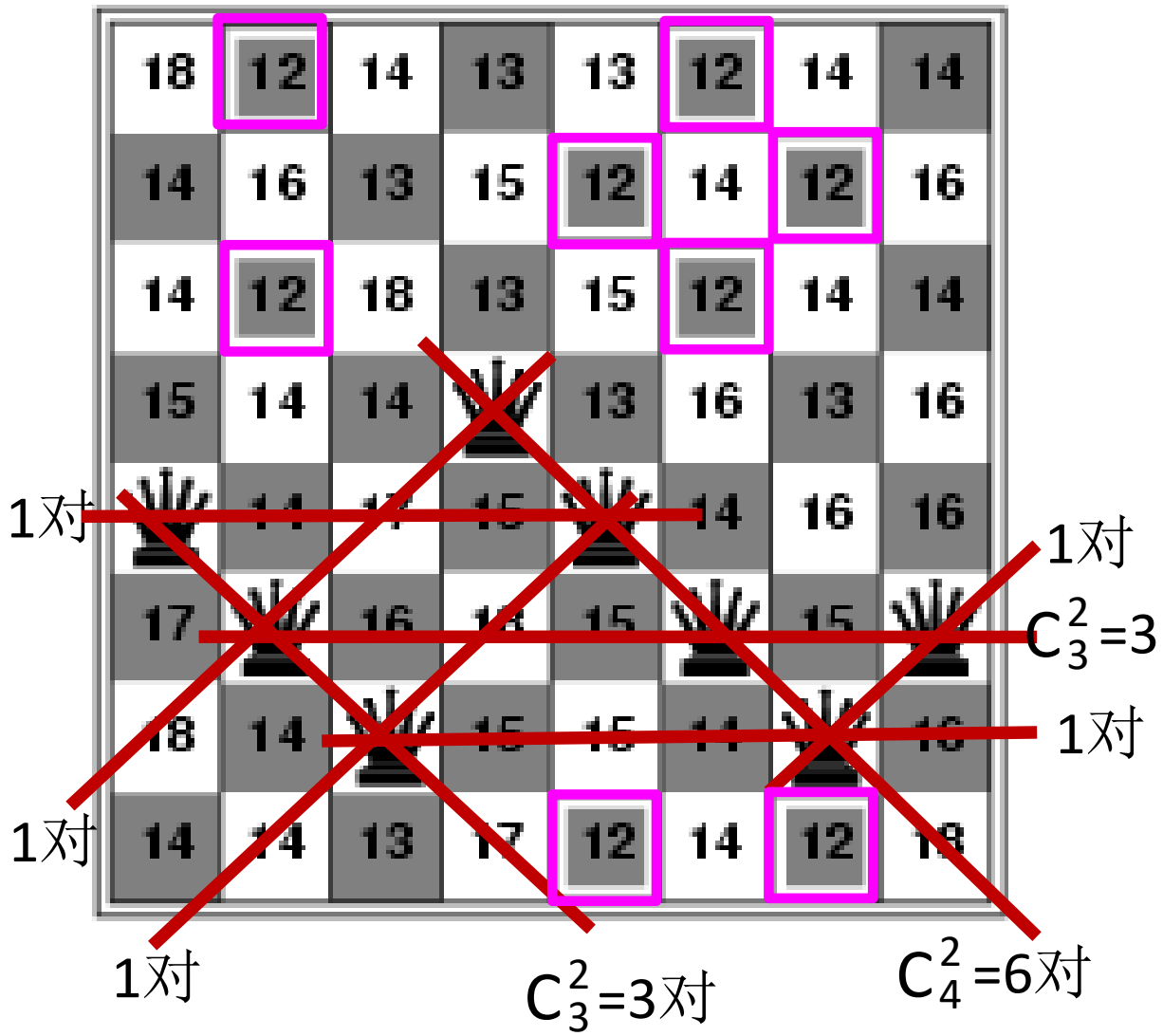
Example: 4-queens problems 4皇后问题



(a) $h = 5$ (b) $h = 2$ (c) $h = 0$ 解

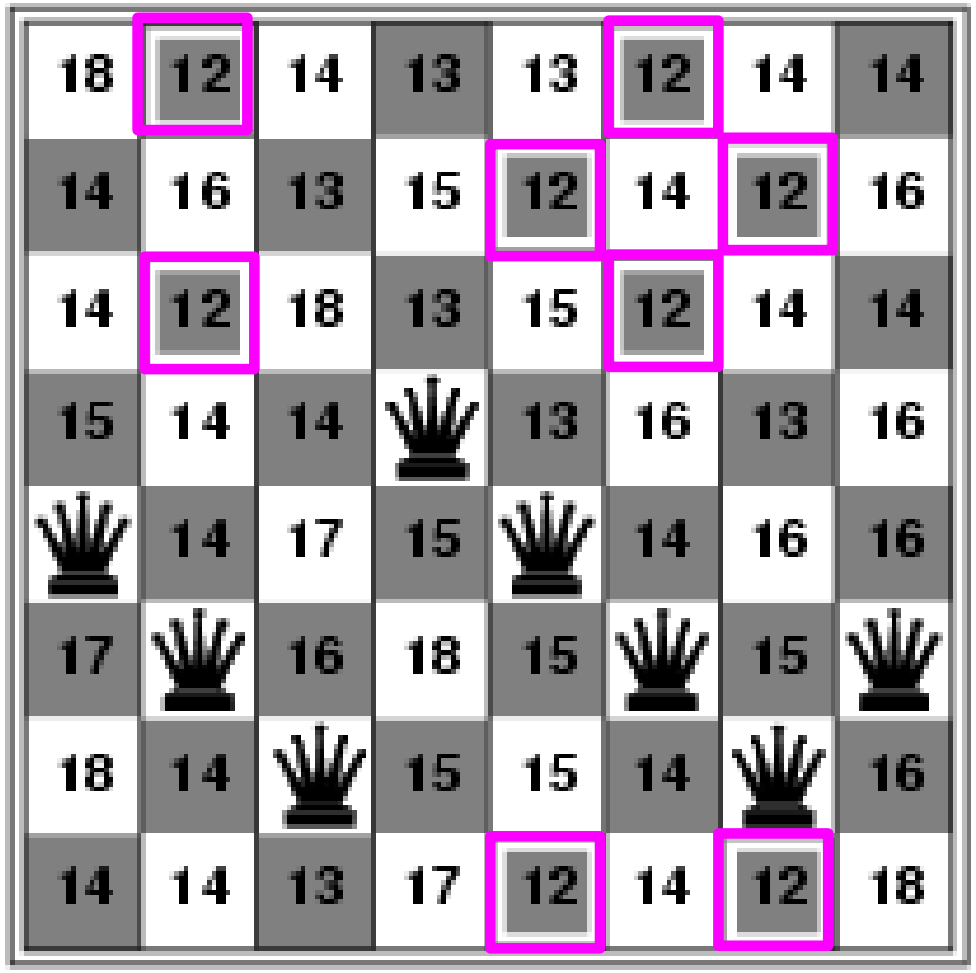
如果有多个最佳后继，爬山算法通常会从一组最佳后继中随机选择一个。

8皇后的状态图，计算启发函数h的值



- 当前状态的启发式代价评估h =形成相互攻击的皇后对的数量，$h = 17$
- 每个方格中显示的数字表示将这一列中的皇后移到该方格而得到的后继的h 值。
- 粉色框格表示最佳移动.
- 如果有多个最佳移动，即多个最小值，爬山法会从中随机选择一个后继进行扩展。

求解8皇后的爬山法思路



(1) 在当前状态中，计算h的值；

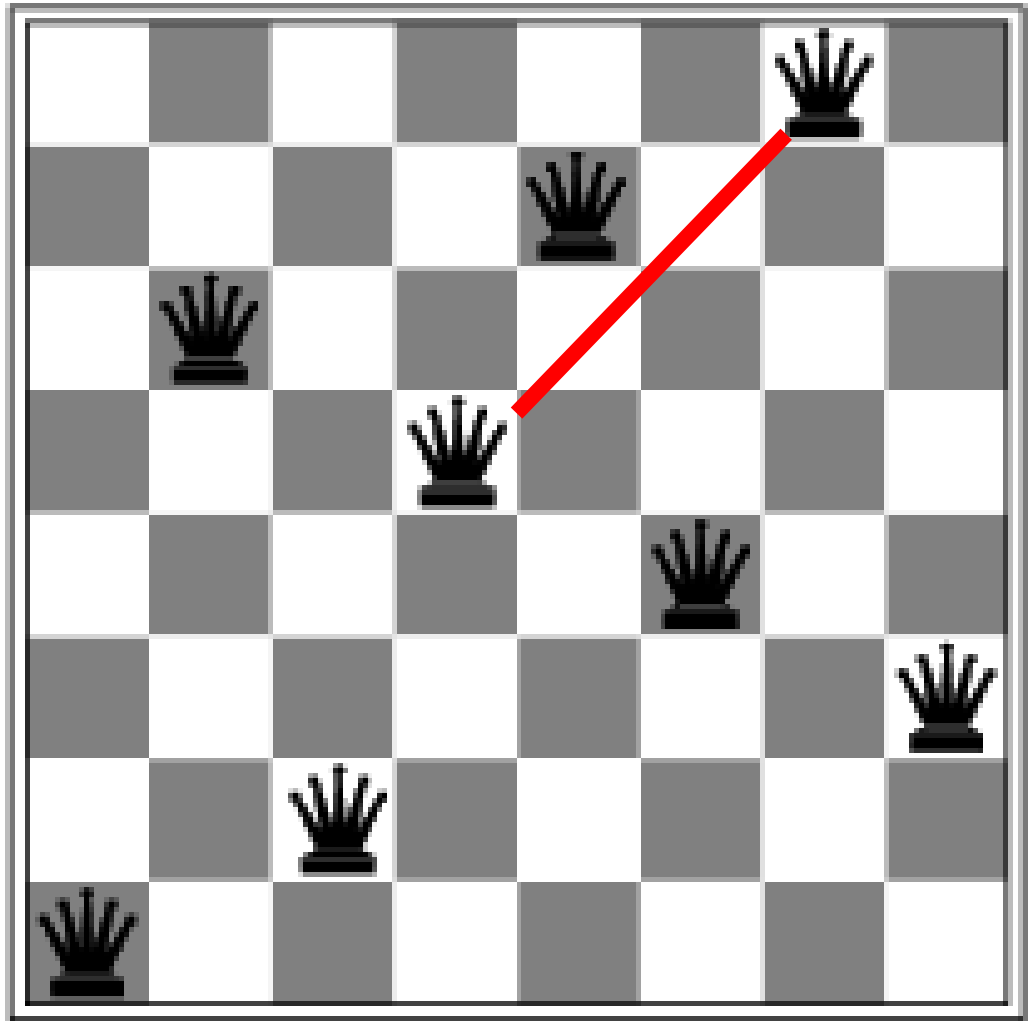
(2) 若h=0，即找到**最优解，终止算法；**

(3) 否则，计算各个方格里的h值；

(4) 若无法找到比当前状态h值更小的相邻状态（**陷入局部极值，找不到解**），终止算法。

(5) 否则从若干个最佳后继（**小于当前的h**）中随机挑选一个，将该列的皇后移到此位置；并 转到步骤(2)。

Hill-climbing search: 8-queens problem



A local minimum with $h = 1$

- 八皇后问题状态空间中的一个**局部极小值**：该状态的 ***h***=l, 但是它的每个后继的 ***h*** 值都比它高.
- 此时，爬山法无法找到全局的最优解（即 h=0），即**爬山法是不完备的**。

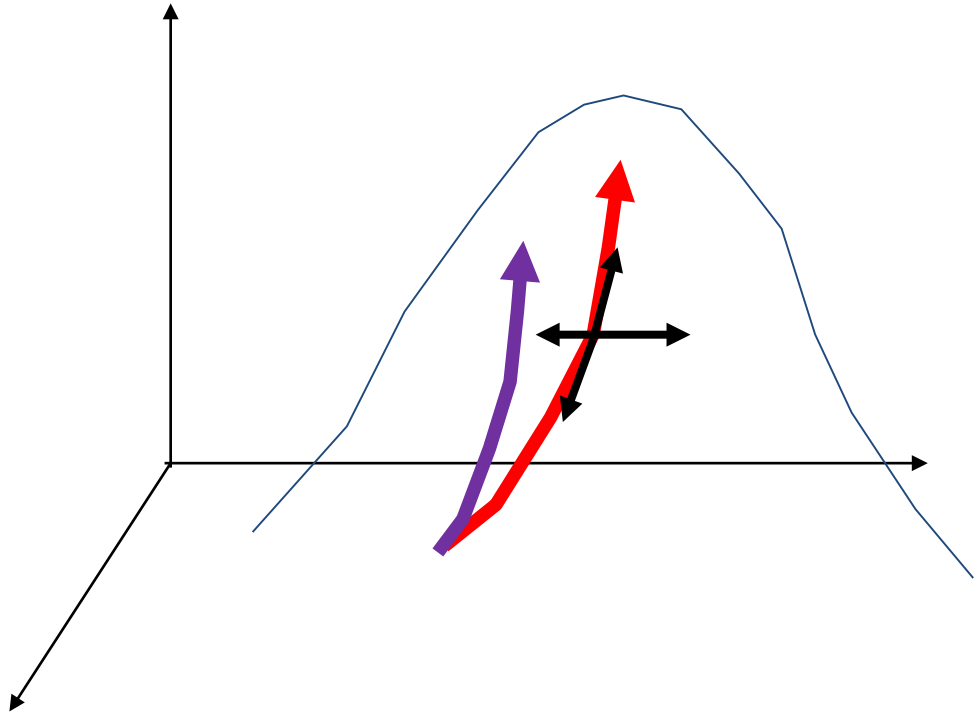
Problem of Hill-Climbing

- 贪婪算法**很难处理陷入局部极大值**的情况。
- 在这种情况下，爬山法均无法再取得进展。
- 从随机生成的八皇后问题开始，采用最陡上升的爬山法，其中**86%的情况**下会**被卡住**，**只有14%的问题实例能求得解**。
- 到现在为止，我们描述的**爬山法是不完备的**—— 它们经常会在**存在目标**的情况下，因为被局部极大值卡住而**找不到目标**。

Variants of Hill-Climbing 爬山法的变型

◆**Stochastic hill-climbing** 随机爬山法

- 在向上移动的过程中，**随机地选择下一步，即不一定选择最陡的路径走**，被选中的概率可能随向上移动的陡峭程度的不同而变化。



- 与最陡上升算法相比，**收敛速度通常较慢**。
- **随机爬山法仍然不完备**，还会被局部极大值卡住。

Variants of Hill-Climbing 爬山法的变型

- **Random-restart hill-climbing 随机重启爬山法**
  - **随机生成一个初始状态**，开始搜索，执行一系列这样的爬山搜索，直到找到目标为止。
  - **随机重启爬山法依然不完备，但以逼近1的概率接近完备**，因为最终它将生成一个目标状态作为初始状态。
  - 如果每次爬山搜索成功的概率为p，则重启需要的期望值是1/p。
  - 对于八皇后问题，随机重启爬山法实际上是有效的。即使**有300 万个皇后**，这个方法**找到解的时间不超过1分钟**。
  - **爬山法成功与否**严重依赖于状态空间地形图的形状：如果在图中几乎没有局部极大值和高原，**随机重启爬山法**会很快找到一个好的解。

（2）模拟退火搜索

- **爬山法**搜索从来不“下山”， 即不会向值比当前结点低的（或代价高的）方向搜索，它肯定**是不完备的**，理由是**可能卡在局部极大值上**。
- 与之相反，**纯粹的随机行走是完备的**，但是效率极低。**随机行走就是从后继集合中完全等概率的随机选取后继**。
- 因此，将**爬山法和随机行走**以某种方式**结合**，同时得到**效率**和**完备性**的想法是合理的。**模拟退火就是这样的算法**。

Annealing 退火

- 在冶金中，**退火**是通过将金属和玻璃加热到高温，然后逐渐冷却，使材料达到低能结晶状态，从而使金属和玻璃回火或硬化的过程。
- 为了更好地理解模拟退火，我们把注意力从爬山法转向梯度下降（即，减小代价），想象在高低不平的平面上有个乒乓球掉到最深的裂缝中。
- 如果只允许乒乓球滚动，那么它会停留在局部极小点。
- 如果晃动平面，我们可以使乒乓球弹出局部极小点。
- 窍门是晃动幅度要足够大让乒乓球能**从局部极小点处弹出来**，但又**不能太大**，以致于把它**从全局最小点处弹出来**。
- **模拟退火的解决方法**就是开始使劲摇晃（也就是先高温加热）然后慢慢降低摇晃的强度（也就是逐渐降温）。

模拟退火搜索的基本思路

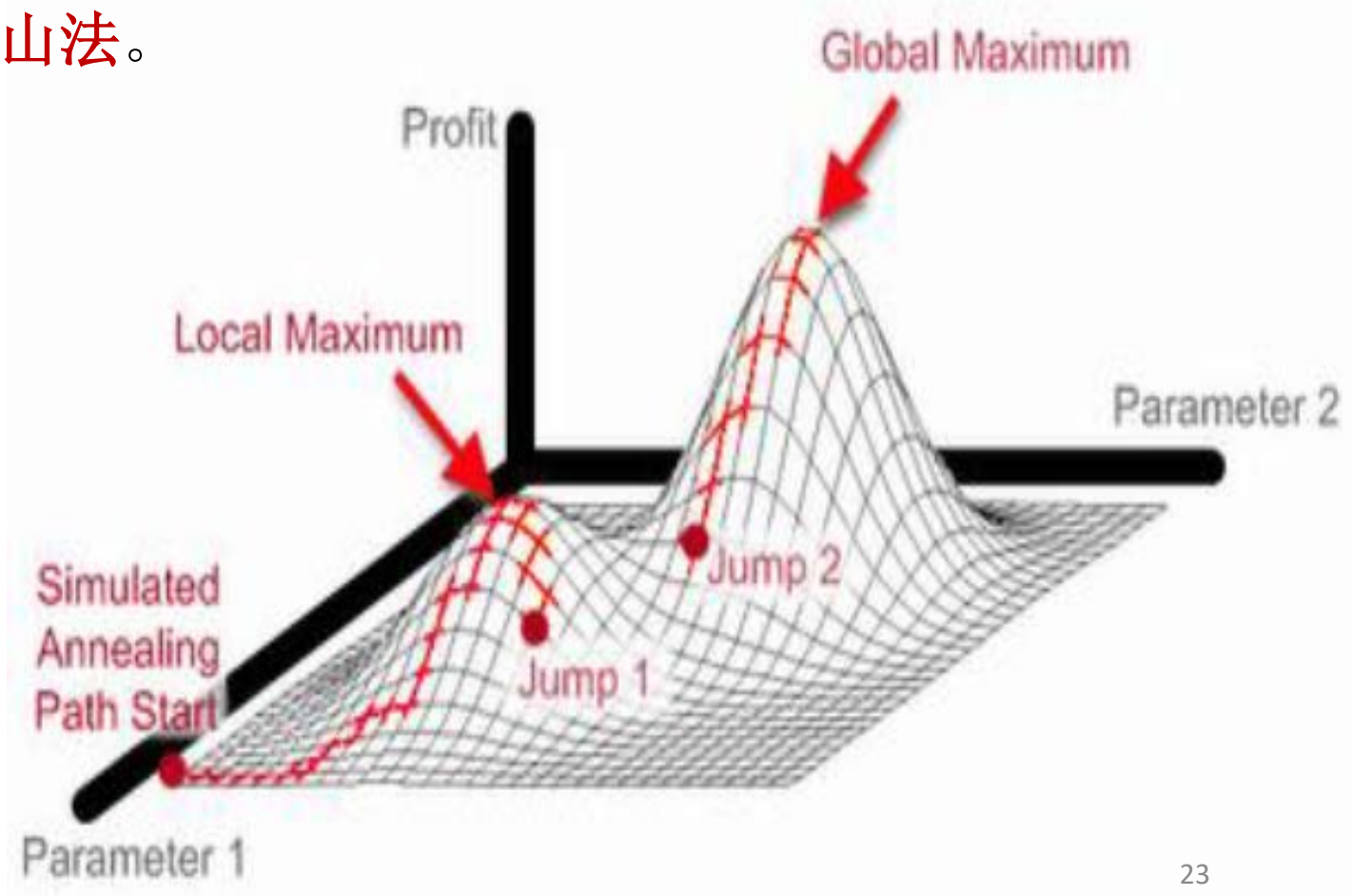
- 避免局部极大值，允许一些“坏”的移动，但逐渐减少他们（“坏”的移动）的频率。
- 以**概率**突破局部最优，走向全局最优。

(2) Simulated Annealing 模拟退火

- 模拟退火算法的内层循环与爬山法类似，只是它**没有选择最佳移动**，选择的是**随机移动**。
- 如果该移动能改善情况，该移动则被接受；否则，算法以某个小于1 的概率接受该（变坏的）移动。
- 随着移动导致状态“变坏”， **接受概率会呈指数级下降**——根据能量差值ΔE 判断。
- 这个**概率也随“温度”T的 降低而下降**：开始T高的时候，可能允许“坏的”移动；当T 降低时，则不可能允许“坏的”移动。
- 如果调度让温度T下降得足够慢，算法找到全局最优解的概率接近于1。
- 模拟退火在20 世纪80 年代早期，被广泛用于求解VLSI (大规模集成电路)布局问题。现在它已经广泛地应用于工厂调度和其他大型最优化任务。

(2) Simulated Annealing 模拟退火

- 模拟退火是一种**逼近全局最优解**的概率方法，发表于1953年。
- 模拟退火算法，是**允许下山的随机爬山法**。
- 在退火初期，下山移动容易被采纳；
- 随时间推移，下山的次数越来越少。



Simulated annealing search

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
    inputs: problem, a problem
            schedule, a mapping from time to "temperature"
    current ← MAKE-NODE(problem.INITIAL-STATE)
    for t = 1 to ∞ do
        T ← schedule(t)
        if T = 0 then return current
        next ← a randomly selected successor of current
        ΔE ← next.VALUE − current.VALUE
        if ΔE > 0 then current ← next
        else current ← next only with probability e^(ΔE/T)
```

// T是温度， t是时间， schedule是将时间 t 映射到温度 T 的一个调度算法。

// T=0 时，说明温度已经最低，退火结束，则返回当前结点。

// 随机选取当前结点的一个后继作为下一个结点。

// 计算当前结点和下一个结点之间的能量差值ΔE。

// 若ΔE>0，说明是上山，状态在变好，则将下一结点作为当前结点，进行下一次循环。

// 若ΔE<0，说明是下山，状态在变坏。此时，可以以此概率将下一结点作为当前结点，进行下一次循环。

（3）Genetic Algorithms 遗传算法

◆遗传算法是约翰·霍兰（ John Holland ）在1970年代早期提出的，尤其他的书《神经元与人工系统的适应性》（1975年），使遗传算法流行起来。

◆遗传算法是一种模仿自然选择过程的启发式搜索算法。

◆在遗传算法中, 后继节点是由两个父辈状态的组合、而不是修改单一状态生成的。其处理过程是有性繁殖，而不是无性繁殖。

◆遗传算法属于**进化算法**这个大分类。

◆遗传算法采用自然进化所派生的技术来生成优化问题的解，例如：遗传、变异、选择、以及杂交。

遗传算法中的基本概念

◆**种群**（Population）：是初始时给定的多个解的集合，其中有一组k个随机生成的状态，称其为**种群**。。

◆**个体**（Individual）：指种群中的单个状态，用于描述其基本遗传结构的数据结构，表示为有限字母表上的一个字符串，通常是0和1的字符串。

◆**染色体**（Chromosome）：指对个体进行编码后所得到的编码串。染色体中的每一位称为基因，染色体上由若干个基因构成的一个有效信息段称为**基因组**。例如：11011为一个染色体，每一位上的0或1表示基因。

遗传算法中的基本概念

◆**适应度**（Fitness）**函数**：一种用来对种群中各个个体的环境适应性进行度量的函数。其函数值是遗传算法实现**优胜劣汰**的主要依据。

◆对于好的状态，适应度函数应返回较高的值，即：**适应度越高，越好**。

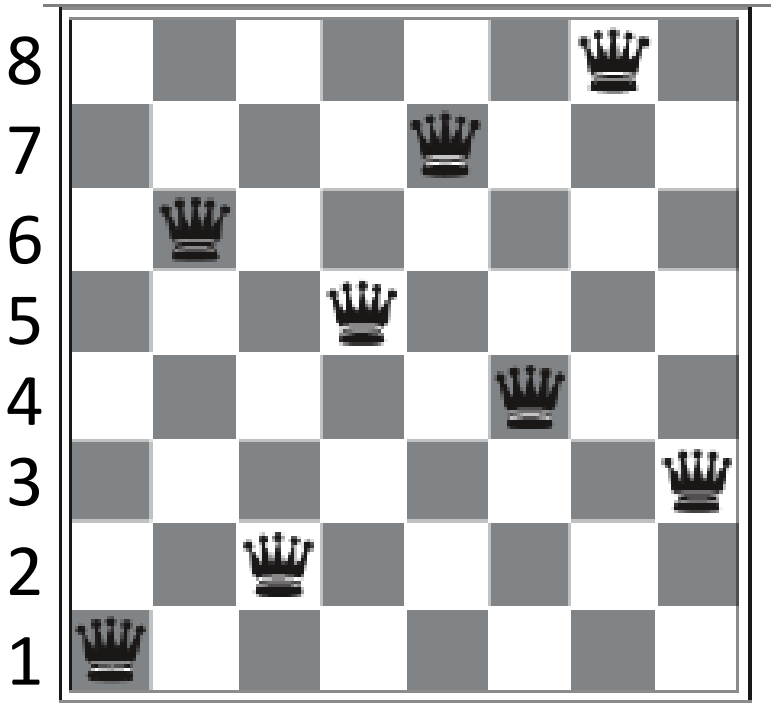
◆**遗传操作**（Genetic Operator）：指作用于种群而产生新的种群的操作。

◆标准的遗传操作包括以下三种基本形式：

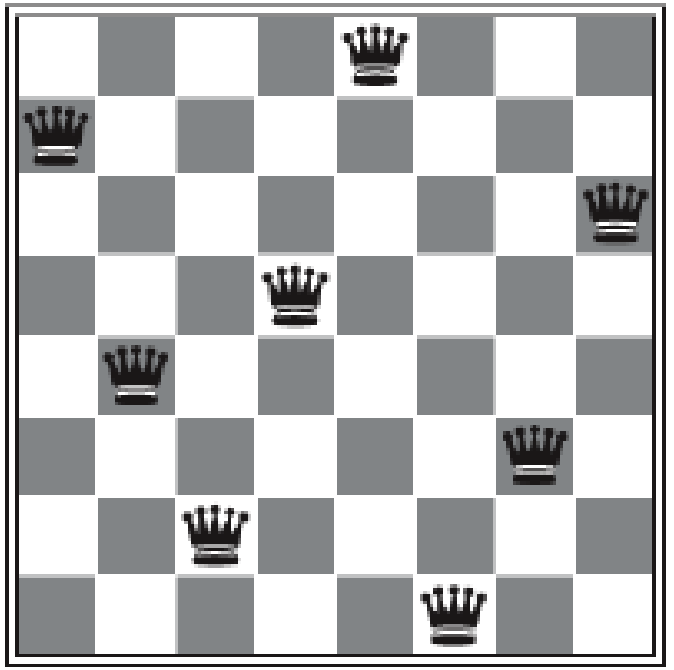
- 选择（Selection）
- 交叉（Crossover）
- 变异（Mutation）

Example: 8-queens problem 8皇后问题

某8皇后状态需要指明8个皇后的位置，每列有一个皇后，其状态可用8个数字表示，每个数字表示该列中皇后所在的行号，其值在1到8之间。



16257483



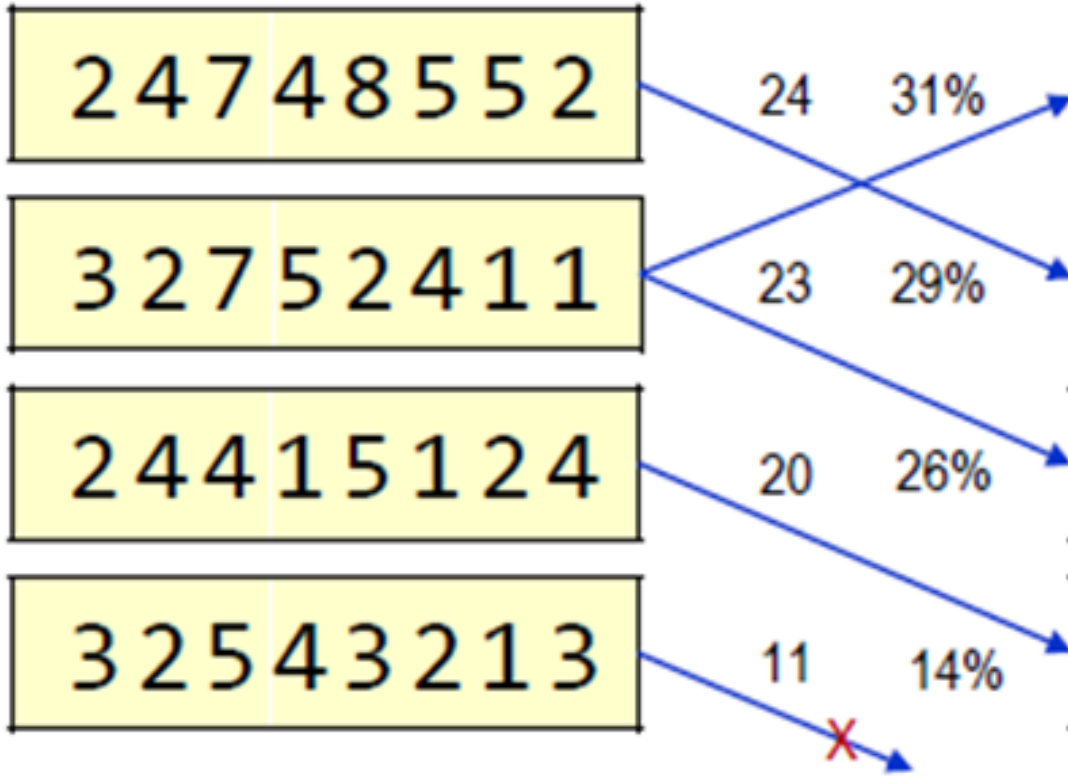
74258136

Example: 8-queens problem 8皇后问题

- 在八皇后问题中，我们用**不相互攻击的皇后对的数目来表示适应度**。最优解的适应度是28。

$$C_8^2=28$$

- 右图中，这四个状态的**适应度**分别是24 (4对攻击)、23 (5对攻击) 、20 和11。



- 在这个特定的遗传算法实现中，**被选择**进行繁殖的**概率**直接**与个体的适应度成正比**，其百分比标在旁边。（即**适应度越高，越好**。）

第一个状态被选择的概率为：24 /（24+23+20+11）=24/78= 30.8%

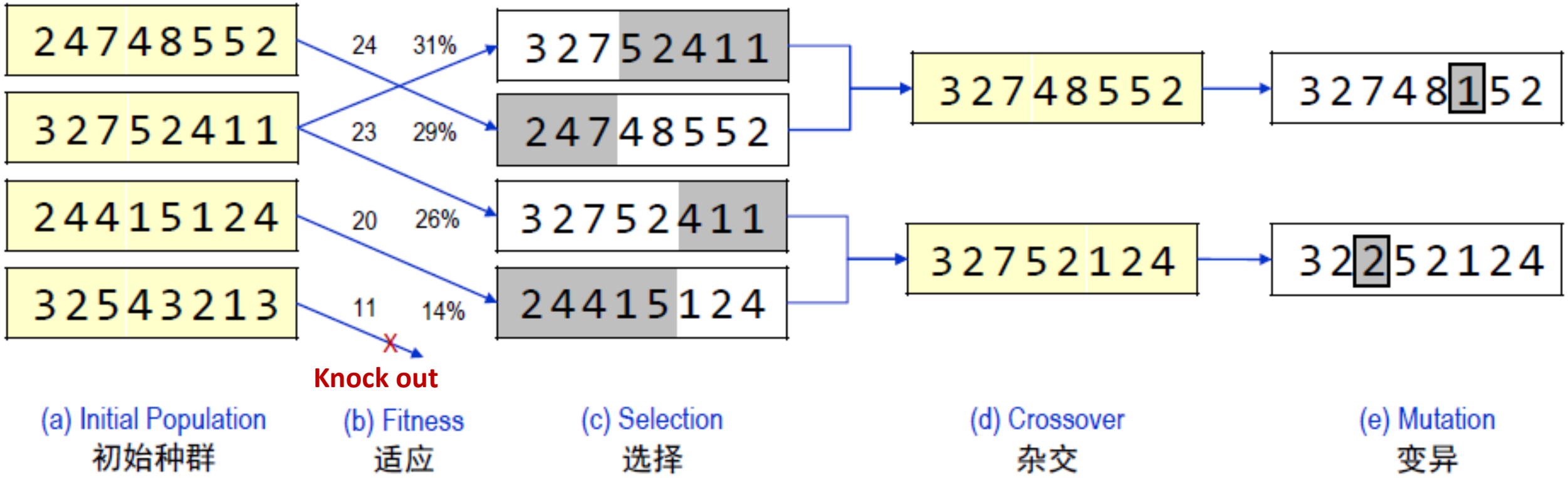
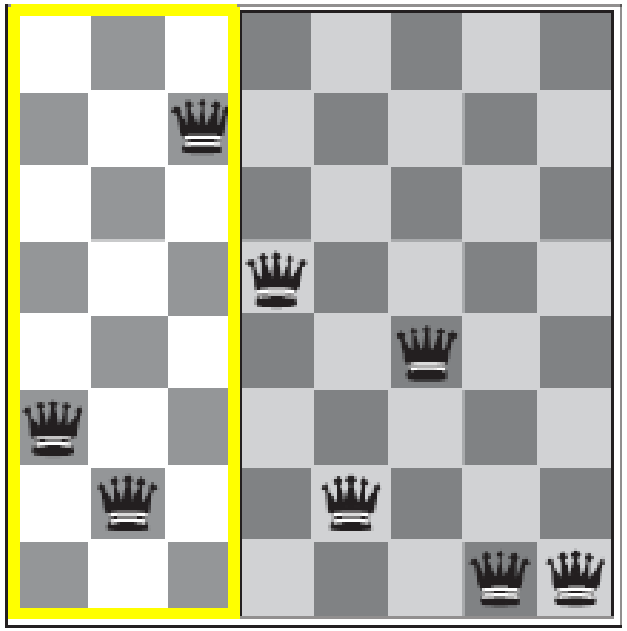


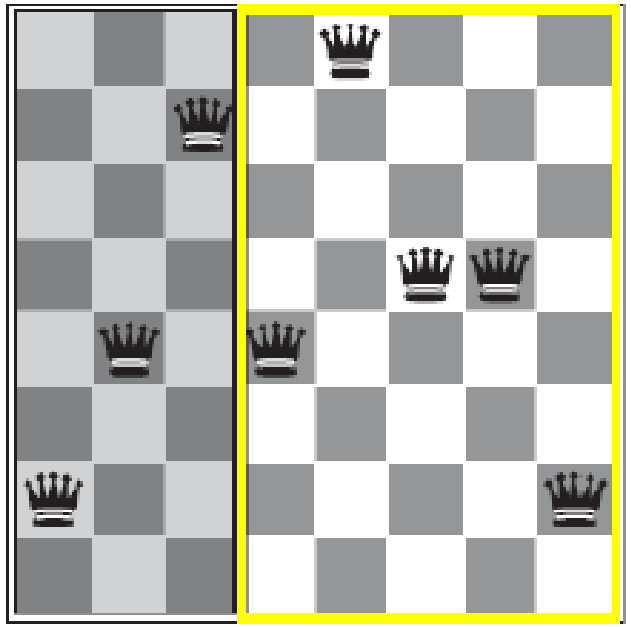
图1. 数字串表示8皇后的状态(a)为初始种群　(b)计算适应度函数　(c) 选择并配对的结果
(d) 杂交产生的后代　(e) 变异的结果

- 按概率随机地选择两对进行繁殖，第2个个体被选中两次, 而第4个个体一次也没被选中。
- 随机选择一个位置作为**杂交点**, 第一对的在第3位数字之后，第二对的在第5位数字之后。
- 图 (e) 中每个位置都会按照某个小的独立概率随机**变异**。
- 在八皇后问题中，这相当于**随机**地**选取一个皇后并把它随机地放到该列的某一个方格里**。

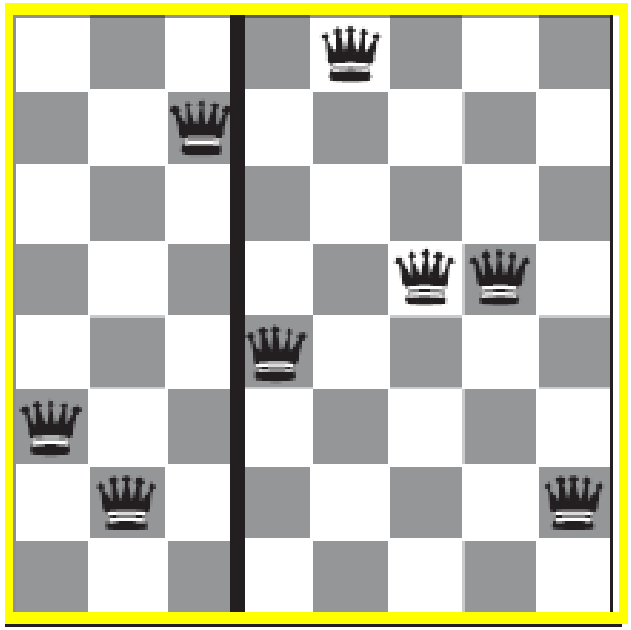
Example: 8-queens problem 8皇后问题



+



=



32752411

24748552

32748552

图2. 这三个8皇后的状态分别对应于“选择”中的两个父辈和“杂交”中它们的后代。阴影的若干列在杂交步骤中被丢掉，而无阴影的若干列则被保留下来。

The Genetic Algorithm

```
function GENETIC-ALGORITHM(population, FITNESS-FN) returns an individual
  inputs: population, a set of individuals   // 输入：种群，是若干个体的集合
          FITNESS-FN, a function that measures the fitness of an individual
                                             // 个体适应度函数
  repeat
    new_population ← empty set               // 初始时，设新种群为空集
    for i = 1 to SIZE(population) do         // 取每一个个体
      x ← RANDOM-SELECTION(population, FITNESS-FN)   // 计算个体的适应度，按选择算子选择2个个体
      y ← RANDOM-SELECTION(population, FITNESS-FN)
      child ← REPRODUCE(x, y)                // 将随机选择的2个个体进行杂交，生成新个体child
      if (small random probability) then child ← MUTATE(child)
      add child to new_population            // 若随机概率小，则将新个体child进行变异。
                                             // 将新个体child加入新种群。
    population ← new_population              // 用新种群替换原来的种群。
  until some individual is fit enough, or enough time has elapsed   // 某些新个体足够健康，或超时
  return the best individual in population, according to FITNESS-FN//根据适应度，返回最佳个体
```

Applications of Genetic Algorithms 遗传算法的应用

bioinformatics	■	生物信息学
computational science	■	计算科学
engineering	■	工程
economics	■	经济学
chemistry	■	化学
manufacturing	■	制造
mathematics	■	数学
physics	■	物理
phylogenetics	■	种系遗传学
pharmacometrics	■	定量药理学

Vivid Interpretation —形象解释

- **爬山算法**：一只袋鼠不断跳向比现在高的地方。它找到了不远处的最高山峰，但不一定真的是最高峰，也许只是次高峰。
- **模拟退火**：袋鼠喝醉了，它**随机**地跳了很长时间，这期间它可能走向高处，也可能走向低处。但是，它渐渐清醒了，并跳向最高山峰。
- **遗传算法**：在山中随机投放N只袋鼠。袋鼠不知道自己的任务是寻找最高山峰。但每过几年，就在低山峰射杀一些袋鼠。于是，低山峰的袋鼠不断死去，而高山峰的袋鼠繁衍生息。许多年后，袋鼠就会不自觉地聚拢到高山峰处。