局部搜索/高级搜索(Local Search)

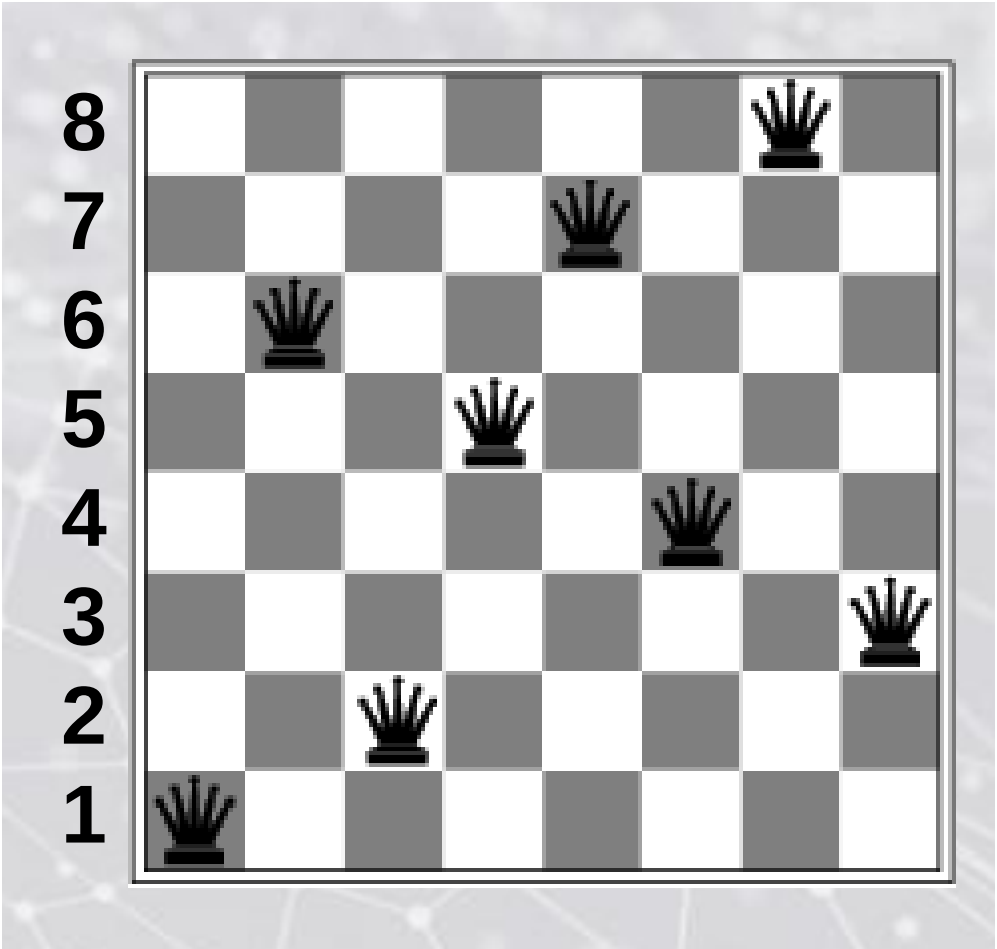
- 爬山法搜索(Hill-climbing search)
- 模拟退火算法(Simulated annealing)
- 遗传算法(Genetic algorithms)

*Slides partly based on those of Jiahai Wang

动因

- 我们目前所见的搜索算法都是为了系统地探索搜索空间。
- 然而，系统的搜索在非常大的问题实例上是低效的。
- 局部搜索：评估和修改一个或多个当前状态。
- 适合那些仅关注解的状态，而不是到达它的路径的问题。

仅关注解的状态: 一个示例



系统的与局部搜索的比较

- 系统搜索：在内存中保存多条路径。
- 当找到一个目标状态时，到达该目标状态的路径就构成了该问题的解。
- 然而，在许多问题中，通往目标的路径是无关紧要的。
- 局部搜索：保存单个当前节点，通常只移动到该节点的邻居。
- 通常，不保存搜索路径。
- 也用于解决纯优化问题：根据目标函数找到最佳状态。

局部搜索：关键优点

- 使用很少的内存——通常是一个常数量
- 通常能够在系统搜索不适用的大的或者无穷的(连续)状态空间中找到合理的解

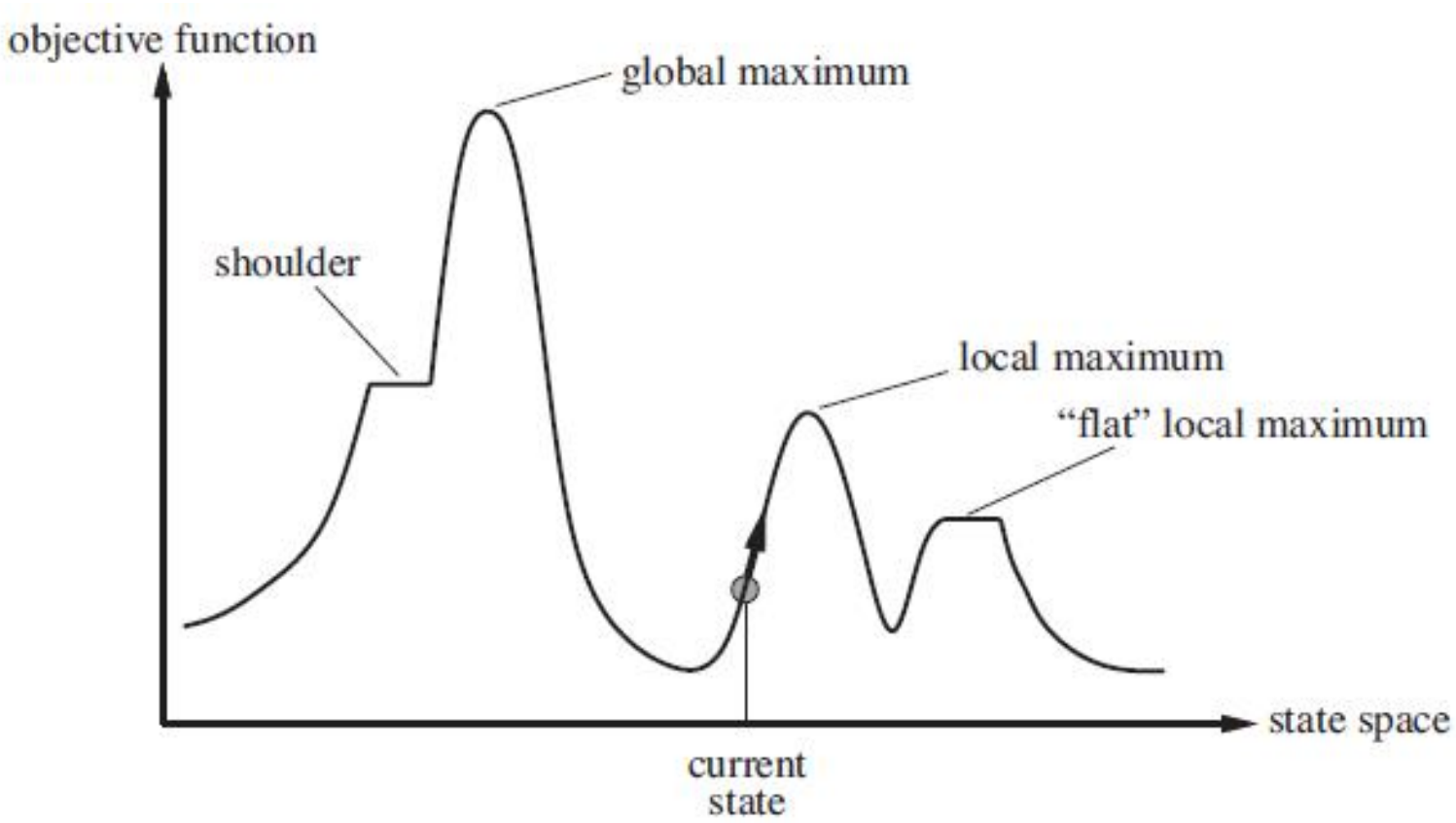
局部搜索的形式化

- 搜索空间S: 状态表示
- 解的集合$S' \subseteq S$
- 邻域关系$N \subseteq S \times S$
- 目标函数$f : S \to \mathbf{R}$或评价函数$g : S \to \mathbf{R}$.

对8-queens的局部搜索

- 每个状态有8个皇后，每列一个。
- 解:没有两个皇后互相攻击的的状态
- 相邻的状态只有一个皇后的位置不同(所以每个状态有8×7=56个邻居).
- 启发式代价函数h: 相互攻击的皇后对的数量；只有对解h为零。

地形图(The state-space landscape)



局部最大:值大于邻近的值的一个状态

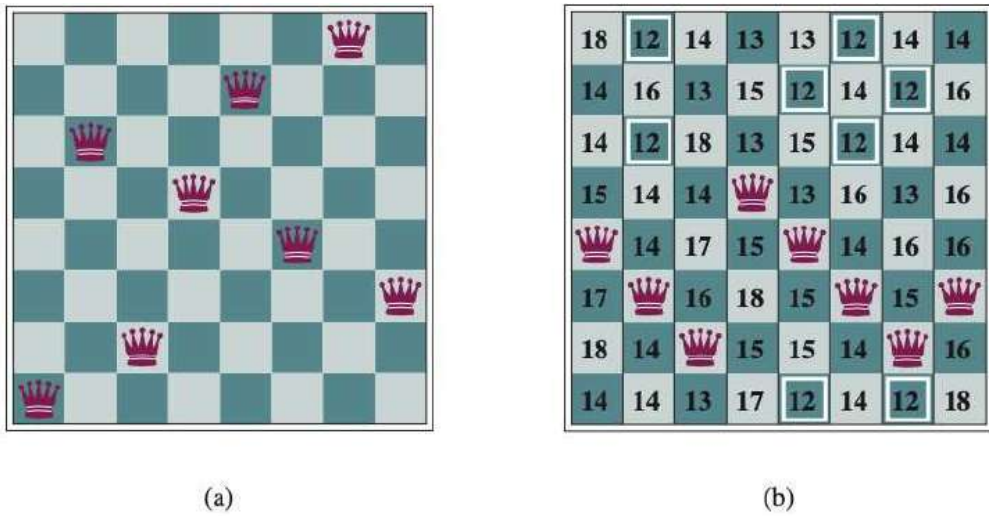
shoulder(山肩)、plateau(平顶区)

爬山(贪心局部搜索)

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    current ← MAKE-NODE(problem.INITIAL-STATE)
    loop do
        neighbor ← a highest-valued successor of current
        if neighbor.VALUE ≤ current.VALUE then return current.STATE
        current ← neighbor
```

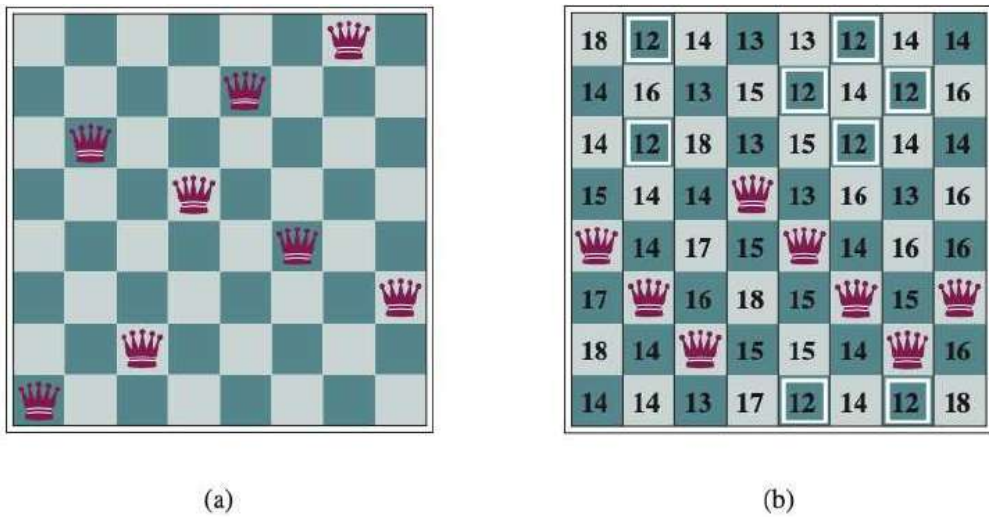
- 向增值方向移动, *i.e.*, 上坡
- 只考虑当前状态的直接邻居
- 当找到局部最大值时终止

一个示例: 8-queens



- 初始状态是随机选择的
- (b)展示了一个$h = 17$的状态.
- 它还显示了它的所有邻居的h值
- 有8个最好的动作,其$h = 12$.
- 爬山算法会从中选择其中一个。

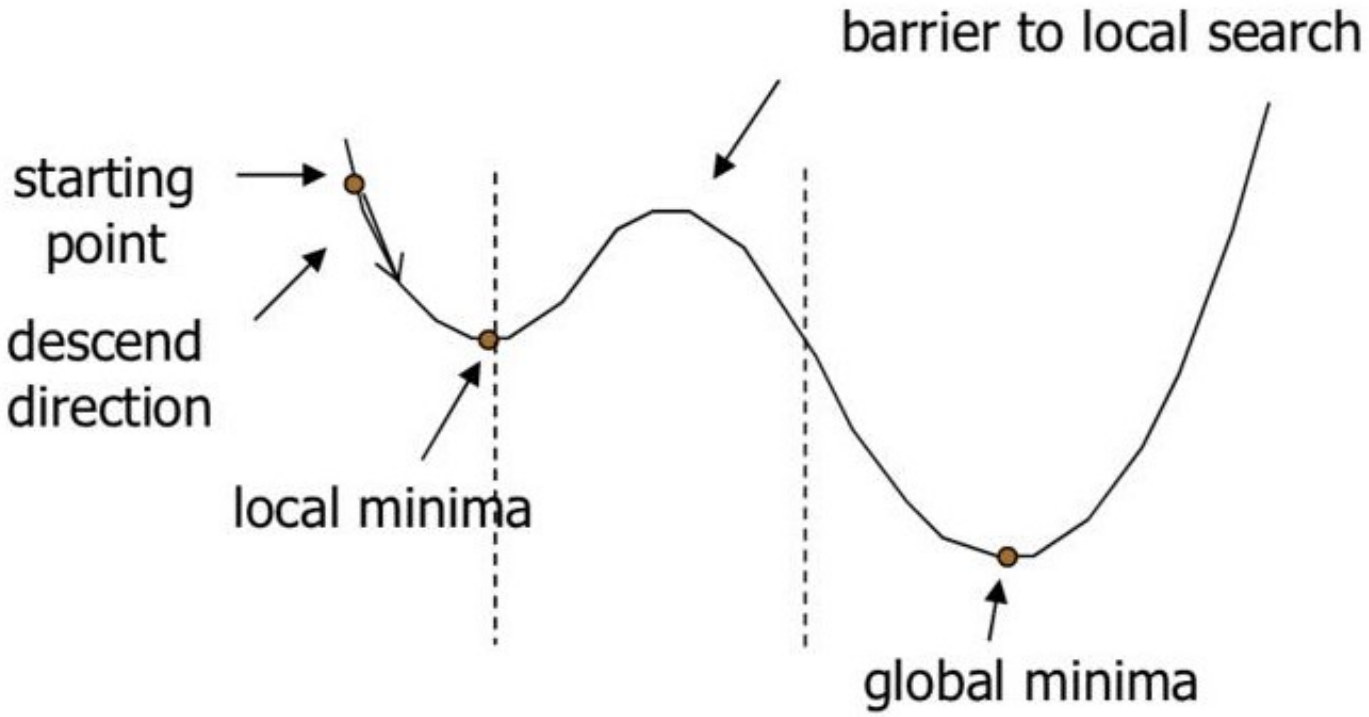
爬山算法的性能



- 也被称为贪心的局部搜索，因为移动到一个最好的邻居
- 可以快速向一个解推进，因为通常很容易改善一个不好的状态。
- *e.g.*, 从(b)中的状态，它只需要五个步骤就可以达到(a)的状态，其$h = 1$，非常接近一个解。

局部搜索的一个典型问题

困在局部最优



也会被困在山肩和平顶区上

8皇后示例

- 从一个随机生成的8皇后状态开始，最佳移动爬山算法86%的时间会卡住，只解决14%的问题实例。
- 另一方面，它工作得很快，成功时平均只走4步，被卡住时平均只走3步—对于一个拥有$8^8 \approx 1700$万个状态的状态空间来说还不错。

解决办法

- 一个答案: 允许平级移动(sideways move)
- 但如果我们实际上处于一个平顶区，那么这种方法将永远在平顶区上徘徊。
- 因而可以限制连续的平级移动的数量，例如，在100次连续平级移动后停止。
- 这将通过爬山法解决的问题实例的比例从14%提高到94%。
- 成功是有代价的：该算法对每个成功的实例平均运行大约21步，对每个失败实例平均64步。

爬山: 随机的(stochastic) 变体(variants)

- 随机的爬山：从上坡动作中随机选择；选择的概率可能会随着上坡动作的陡度而变化。
- 随机重启(Random-restart) 爬山: "如果失败，重新尝试。"
  - 进行一系列的从随机生成的初始状态开始的爬山搜索，直到找到一个目标。
- 随机游走(Random walk) 爬山: 以概率p, 随机选择一个邻居; 以概率$1-p$, 随机选择一个最好的邻居。

随机重启爬山法(Random-restart hill climbing)

- 完备的概率为1，因为最终将生成一个目标状态作为初始状态。
- 如果每个爬山搜索成功的概率为p，那么所需的重启次数的期望值为$1/p$。
- 在不允许平级移动的情况下，对8皇后实例，$p \approx 0.14$, 所以大约需要7次迭代来找到一个目标状态。
- 步骤数的期望值是一次成功迭代的成本加上$(1-p)/p$乘以失败的成本，大约是22。
- 对于8皇后, 随机重启爬山法是非常有效的。
  即使是对于3百万个皇后，也能在几秒钟内找到解。

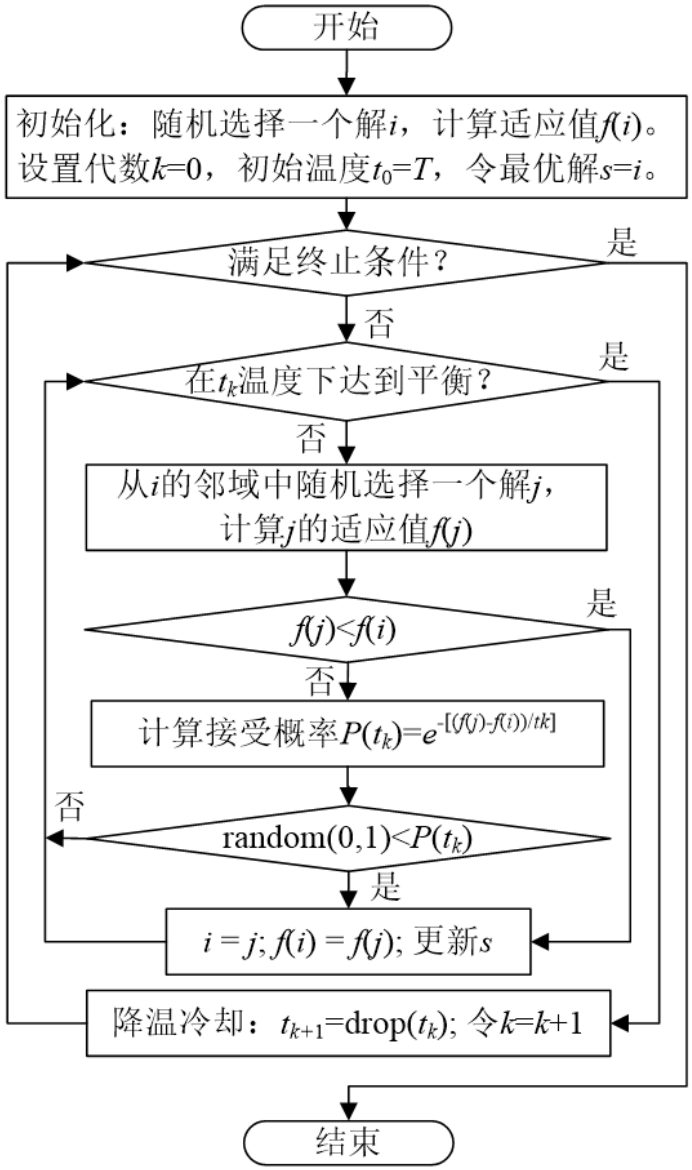
模拟退火: 动因

- 一个从不做下坡动作的爬山算法是不完备的。
- 纯随机游走：移动到均匀随机选择的后继状态，是完备的但效率非常低。
- 模拟退火以一定的方式将爬山和随机游走结合起来，从而兼具效率和完备性。

模拟退火基本思想

- 类似于固体的物理退火的过程,
- 先在高温状态下（相当于随机搜索,大概率接受劣解）
- 然后逐渐降温(接受劣解概率变小直至为0,相当于局部搜索)
- 最终达到物理基态（相当于算法找到最优解）
- 算法的本质是通过温度来控制算法接受劣解的概率

模拟退火基本流程




```
//功能：模拟退火算法伪代码
//说明：本例以求问题最小值为目标
//参数：T为初始温度; L为内层循环次数

procedure SA
  //Initialization
    Randomly generate a solution X0, and calculate its
    fitness value f(X0);
  Xbest=X0; k=0; tk=T;
  while not stop
    //The search loop under the temperature tk
    for i=1 to L  //The loop times
      Generate a new solution Xnew based on the current
      solution Xk, and calculate its fitness value f(Xnew).
      if f(Xnew) < f(Xk)
          Xk = Xnew;
          if f(Xk) < f(Xbest) Xbest=Xk;
          continues;
      end if
      Calculate P(tk)=e^-[(f(Xnew)-f(Xk))/tk];
       if random(0,1) < P
          Xk = Xnew;
      end if
    end for
    //Drop down the temperature
    tk+1=drop(tk); k=k+1;
  end while
  print Xbest
end procedure
```

模拟退火基本要素与设置

功能意义	基本要素	设置方法
影响模拟退火算法全局搜索性能的重要因素之一。 实验表明，初温越大，获得高质量解的几率越大，但花费的计算时间将增加。	初始温度	1、均匀抽样一组状态，以各状态目标值的方差定初温 2、随机产生一组状态，以两两状态间最大差值定初温 3、利用经验公式给出初温
状态空间与状态产生函数。 邻域函数（状态产生函数）应尽可能保证产生的候选解遍布全部解空间。	邻域函数	候选解一般采用按照某一概率密度函数对解空间进行随机采样来获得。 概率分布可以是均匀分布、正态分布、指数分布等等
指从一个状态X_k(一个可行解)向另一个状态X_{new}(另一个可行解)的转移概率，通俗的理解是接受一个新解为当前解的概率	接受概率	一般采用Metropolis准则 $P_{ij}^{T}=\begin{cases}1, & \text{if } E(j)\le E(i)\\ e^{-\left(\frac{E(j)-E(i)}{KT}\right)}=e^{-\left(\frac{\Delta E}{KT}\right)}, & \text{otherwise}\end{cases}$
指从某一较高温状态t_0向较低温状态冷却时的降温管理表，或者说降温方式	冷却控制	1、经典模拟退火算法的降温方式 $t_k=\frac{t_0}{\lg(1+k)}$ 2、快速模拟退火算法的降温方式 $t_k=\frac{t_0}{1+k}$
内层平衡也称Metropolis抽样稳定准则，用于决定在各温度下产生候选解的数目	内层平衡	1、检验目标函数的均值是否稳定 2、连续若干步的目标值变化较小 3、预先设定的抽样数目，内循环代数
算法的终止条件	终止条件	1、设置终止温度的阈值 2、设置外循环迭代次数 3、算法搜索到的最优值连续若干步保持不变 4、检验系统熵是否稳定

局部集束搜索(Local beam search)

- 局部搜索在内存中只保存一个节点。
- 局部集束搜索保存k个状态。
- 它从k个随机生成的状态开始。
- 在每一步中，生成所有k个状态的后继状态。
- 如果有一个是目标，算法终止。
- 否则，选择k个最好的后继状态并重复。

局部集束搜索

- 在随机重启搜索中，每个搜索过程都独立于其他搜索过程。
- 在局部集束搜索中，有用的信息在并行搜索线程之间传递。
- 实际上，产生最好后继的状态会对其他状态说：“Come over here, the grass is greener!”
- 局部集束搜索可能会遇到的问题：在k个状态之间缺乏多样性—它们可能聚集在状态空间的一个小区域内。
- 随机集束搜索：不再选择k个最好的后继状态，而是以值成正比的概率选择后继状态，从而增加多样性。

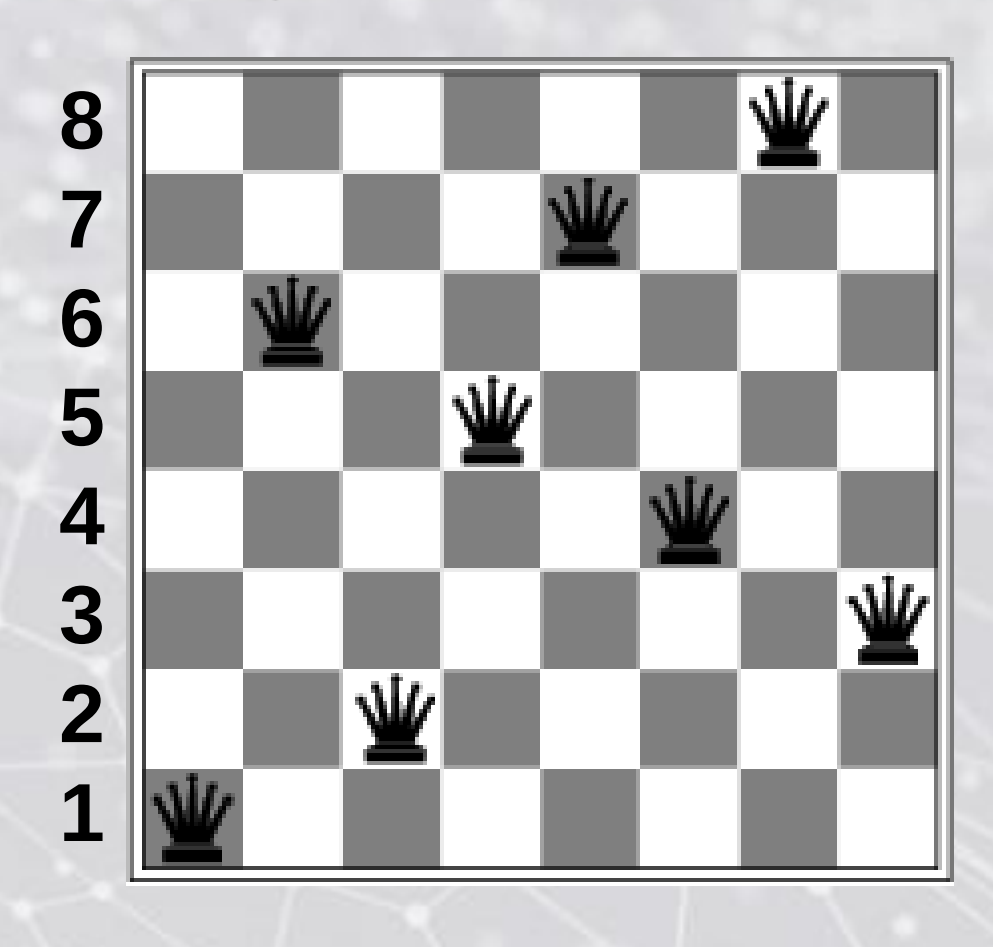
遗传算法(Genetic algorithms): 基本思想

- 随机集束搜索的变体, 受生物学中的自然选择的启发
- 后继状态是通过组合两个父状态而不是通过修改单个状态来生成的。
- 每个状态都被表示为一个有穷的字母表上的一个字符串。
  - *e.g.*, 8-queens: 一个8位数字的字符串, 每个数字从1到8, 表示皇后在每一列上的位置
- 适应函数(fitness function)应该为更好的状态返回更高的值
  - *e.g.*, 8-queens: 非攻击的皇后对的数量,
- 从k个随机生成的状态的集合开始，称为群体(population).
- 通过“模拟进化”产生下一个群体：选择(selection), 交配(crossover), 变异(mutation)

遗传算法(Genetic algorithms, GAs): 基本思想

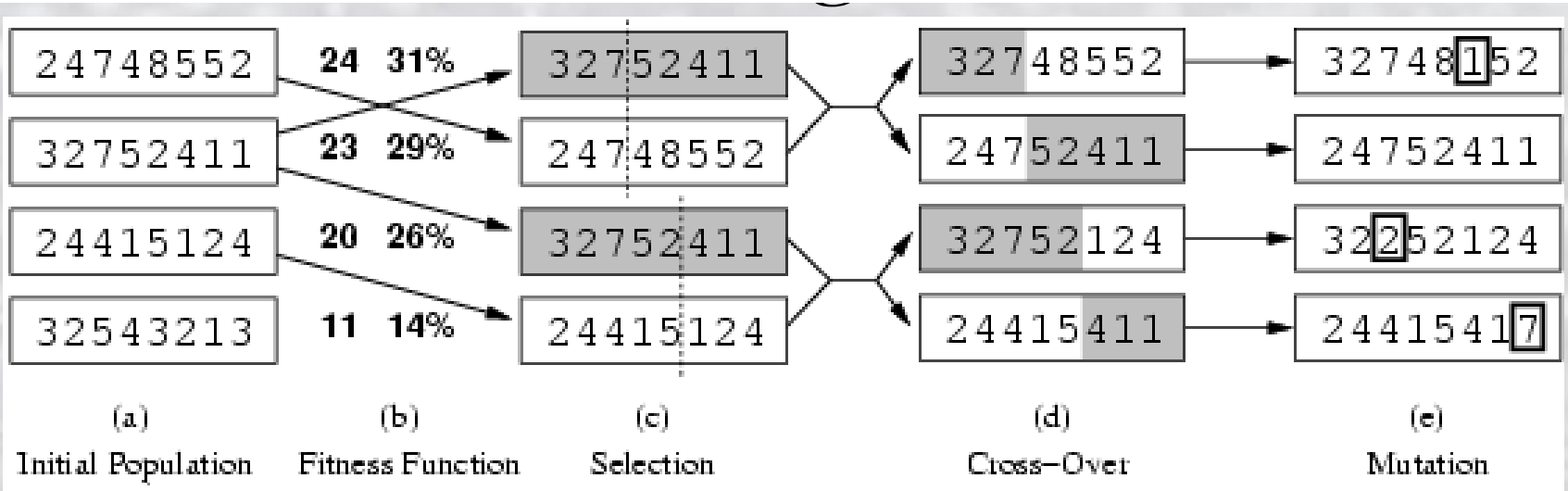
生物遗传概念	遗产算法中的应用
适者生存	目标值比较大的解被选择的可能性大
个体（Individual）	解
染色体（Chromosome）	解的编码（字符串、向量等）
基因（Gene）	解的编码中每一分量
适应性（Fitness）	适应度函数值
群体（Population）	根据适应度值选定的一组解（解的个数为群体的规模）
婚配（Marry）	交叉（Crossover）选择两个染色体进行交叉产生一组新的染色体的过程
变异（Mutation）	编码的某一分量发生变化的过程

8-queens: 状态变量



string representation: 16257483

8-queens: 新群体的生产



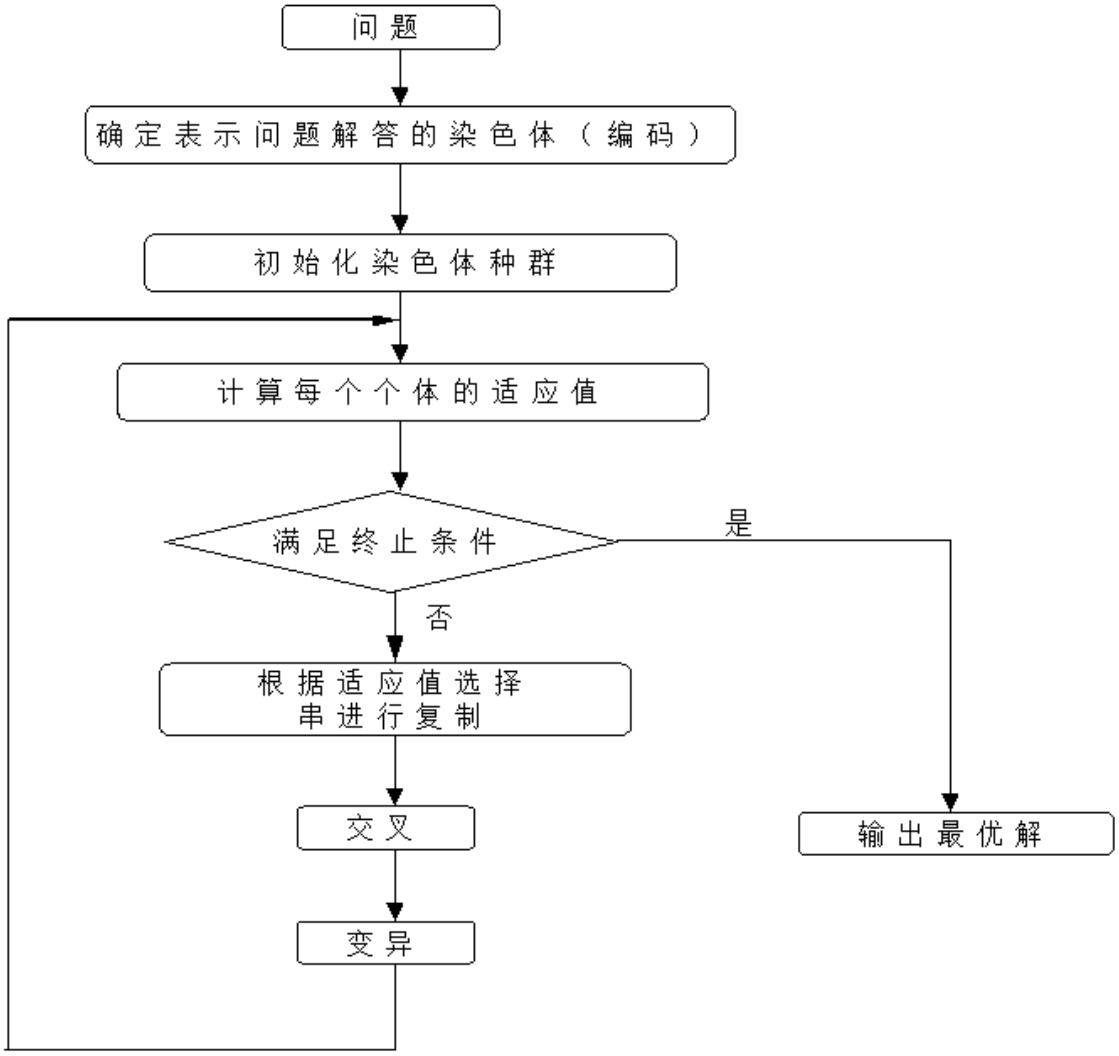
4 states for 8-queens problem

2 pairs of 2 states randomly selected based on fitness. Random crossover points selected

New states after crossover

Random mutation applied

遗传算法的一般步骤


问 题
确 定 表 示 问 题 解 答 的 染 色 体 （ 编 码 ）
初 始 化 染 色 体 种 群
计 算 每 个 个 体 的 适 应 值
满 足 终 止 条 件
是
否
根 据 适 应 值 选 择
串 进 行 复 制
交 叉
变 异
输 出 最 优 解

个体选择概率分配方法

- 适应度比例方法（fitness proportional model）或蒙特卡罗法（Monte Carlo）
  - 各个个体被选择的概率和其适应度值成比例
- 排序方法（rank-based model）
  - 群体成员按适应值大小从好到坏依次排列：$x_1, x_2, \ldots, x_N$
  - 分配选择概率满足条件：$p_1 \geq p_2 \geq \ldots \geq p_N$，且$\Sigma_i p_i = 1$

个体选择方法

- 转盘赌选择：按个体的选择概率产生一个轮盘，产生一个随机数，它落入转盘的哪个区域就选择相应的个体
- 锦标赛选择方法（tournament selection model）：从群体中随机选择n个个体，保存其中适应度最高的个体
- 随机竞争方法（stochastic tournament）：每次按赌轮选择方法选取一对个体，适应度高者被选
- 最佳个体保存方法（elitism）：把群体中适应度最高的个体不进行交叉而直接复制到下一代中

交叉

- 一点交叉：随机设定一个交叉点，将该点前或后的两个个体的部分结构进行互换
- 二点交叉：随机设置两个交叉点，将两个交叉点之间的码串相互交换

变异

- 位点变异：群体中的个体码串，随机挑选一个或多个基因座，并对这些基因座的基因值以变异概率作变动
- 逆转变异：在个体码串中随机选择两点（逆转点），然后将两点之间的基因值以逆向排序插入到原位置中
- 插入变异：在个体码串中随机选择一个码，然后将此码插入随机选择的插入点中间
- 互换变异：随机选取染色体的两个基因进行简单互换
- 移动变异：随机选取一个基因，向左或右移动一个随机位数

一个遗传算法

（1）使用随机方法或者其它方法，产生一个有N个染色 体的初始群体 $pop(1)$, $t := 1$ ；

（2）对群体中的每一个染色体$pop_i(t)$，计算其适应值

$$f_i = fitness(pop_i(t))$$

（3）若满足停止条件，则算法停止；否则，以概率

$$p_i = f_i / \sum_{j=1}^{N} f_j$$

从$pop(t)$中随机选择一些染色体构成一个新种群

$$newpop(t+1) = \{pop_j(t) \mid j = 1,2,..., N\}$$

（4）以概率 p_c 进行交叉产生一些新的染色体，得到一个新的群体

$$crosspop(t+1)$$

（5）以一个较小的概率 p_m 使染色体的一个基因发生变异，形成 $mutpop(t+1)$；$t:=t+1$，成为一个新的群体

$$pop(t)=mutpop(t+1)$$

返回（2）。