

人工智能与机器学习

叶琦

工业控制研究所

杭州 · 浙江大学 · 2022



第四章 超越经典搜索

Beyond Classical Search

环境：可观察的，确定的，已知的
问题的解：行动序列
关注点：路径代价

局部搜索：

考虑对一个或多个状态进行评价和修改，而不是系统地探索从初始状态开始的路径。

关注解状态而不是路径代价

更接近
于现实
世界



- 无信息搜索
- 有信息搜索
 - 最好优先搜索
 - 贪婪最好优先搜索
 - A*搜索
- 启发式函数
- 局部搜索算法
 - 爬山法
 - 模拟退火算法
 - 遗传算法

局部搜索算法

在很多优化问题中**解与路径不相关**，目标状态本身就是解

状态空间 = “所有的” 状态集合；

- 寻找最优的状态，例如旅行商问题（TSP）、集成电路设计
- 寻找满足约束的状态，例如n皇后问题

在这类情况下可以采用局部搜索算法；

保持一个“当前”状态，然后试图改进它



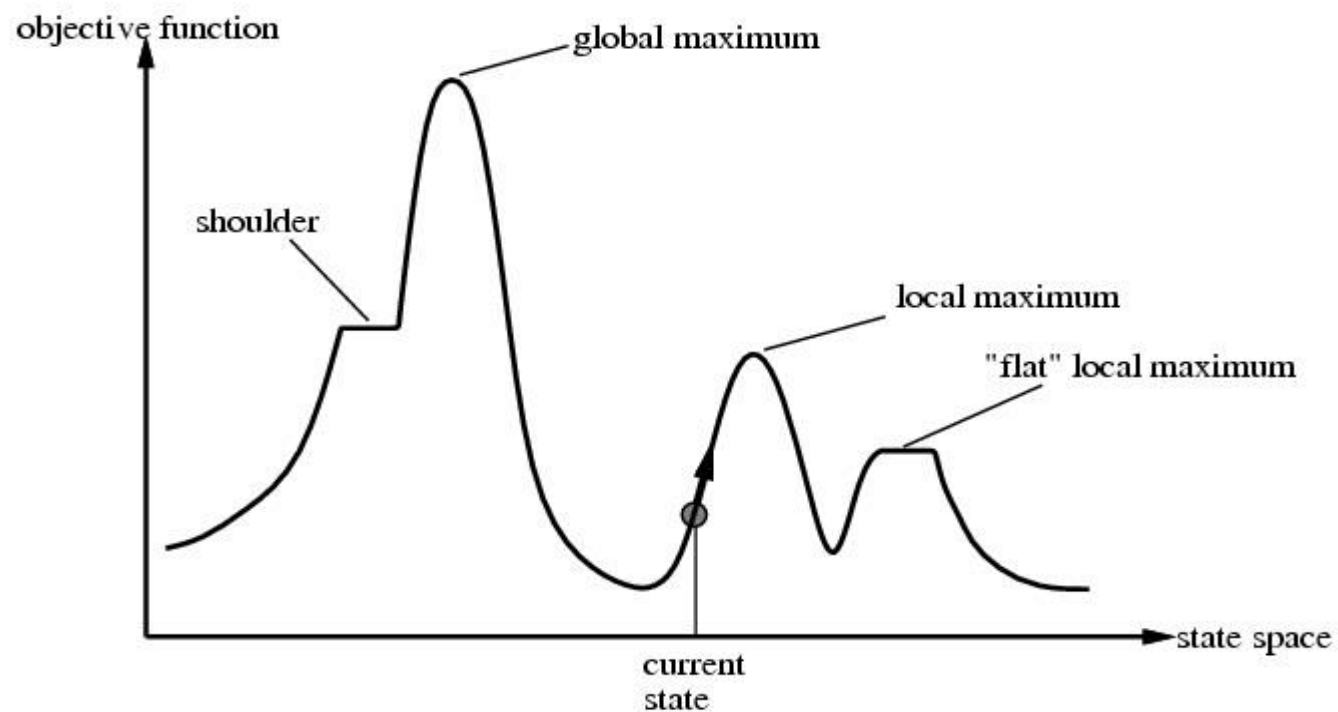
局部搜索算法

- 某些问题不关心到达解的路径，只关心最终状态
- 局部搜索算法：从单独的一个当前状态出发，通常只移动到与之相邻的状态，并且不保留解的路径。
- 优点：
 - 需要很少的内存
 - 经常能在很大或无限的状态空间中找到合理的解



局部搜索算法与最优化问题

- 纯粹的最优化问题(Optimization problems): 根据目标函数 (Objective function) , 找到最佳状态
- 状态空间地形图



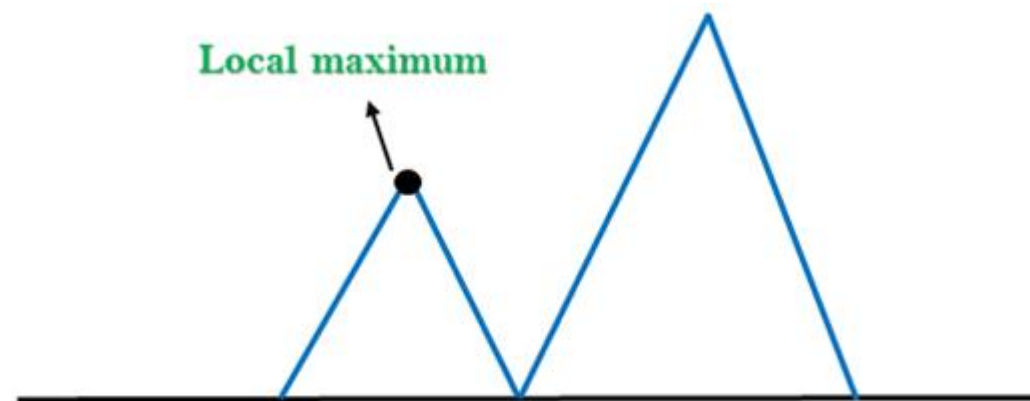
局部搜索算法

- 爬山法搜索
- 模拟退火搜索
- 遗传算法



爬山法

Hill-climbing Search



爬山法搜索

- 向值增加的方向持续移动（最陡上升版本）

Hill-climbing search

“Like climbing Everest in thick fog with amnesia”

```
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
```

简单的循环过程，不断向值增加的方向移动，即“登高”，在到达一个“峰顶”（邻接状态中没有比它更高的）的时候终止。

算法不会考虑与当前状态不相邻的状态，算法不维护搜索树，当前节点的数据结构只记录当前状态和目标函数值。



爬山法 Hill-climbing Search

爬山算法即是模拟爬山的过程，随机选择一个位置爬山，每次朝着更高的方向移动，直到到达山顶，即每次都在临近的空间中选择最优解作为当前解，直到局部最优解。

问题：算法会陷入局部最优解，能否得到全局最优解取决于初始点的位置。初始点若选择在全局最优解附近，则就可能得到全局最优解。

爬山算法是一种局部择优的方法，采用启发式方法，是对深度优先搜索的一种改进，它利用反馈信息帮助生成解的决策。属于人工智能算法的一种。

算法描述

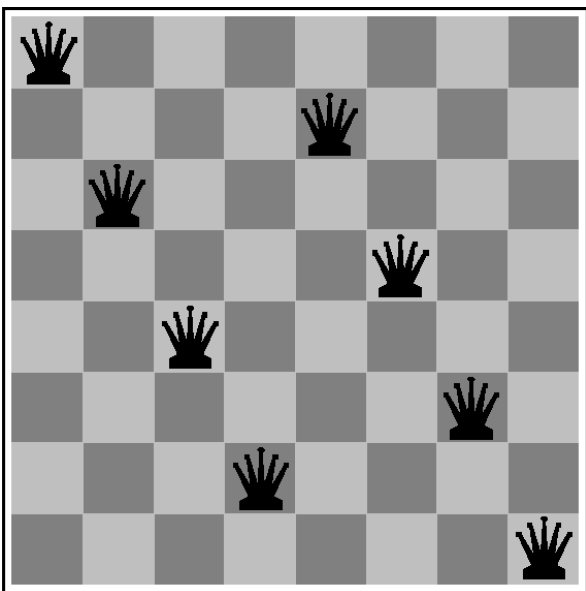
从当前的节点开始，和周围的邻居节点的值进行比较。如果当前节点是最大的，那么返回当前节点，作为最大值(既山峰最高点)；反之就用最高的邻居节点来，替换当前节点，从而实现向山峰的高处攀爬的目的。如此循环直到达到最高点。

在连续空间中选择合适的步长，会缓慢的收敛



例子：8皇后问题

- 目标：任何一个皇后都不会攻击到其他的皇后（皇后可以攻击和它在同一行、同一列或同一对角线上的皇后）
- h取作可以彼此攻击的皇后对的数目（忽略障碍）

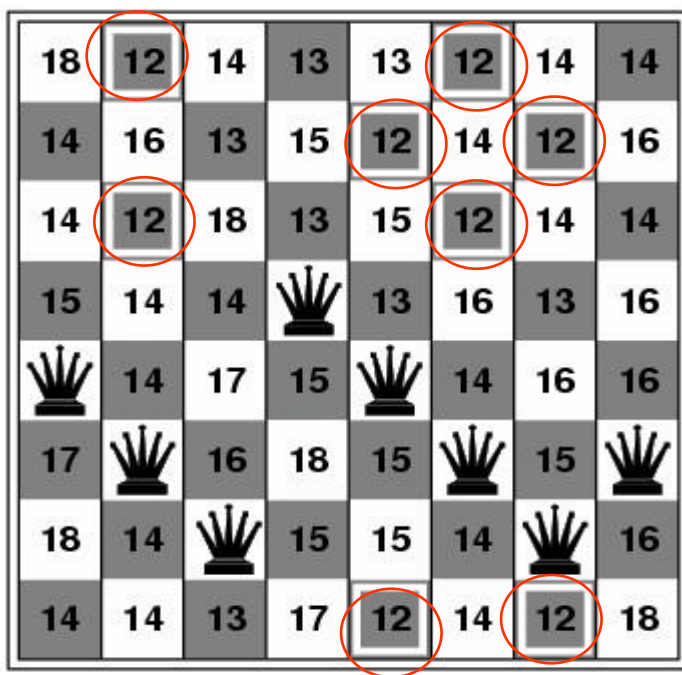


八皇后问题（当前状态指的是一个棋盘放着8个皇后的冲突状态），而且每一次生成新状态（后继节点）是通过：移动某一个（不是多个）皇后到这列的另一个可能方格中，所以每个状态有 $7 \times 8 = 56$ 个后继。



八皇后问题

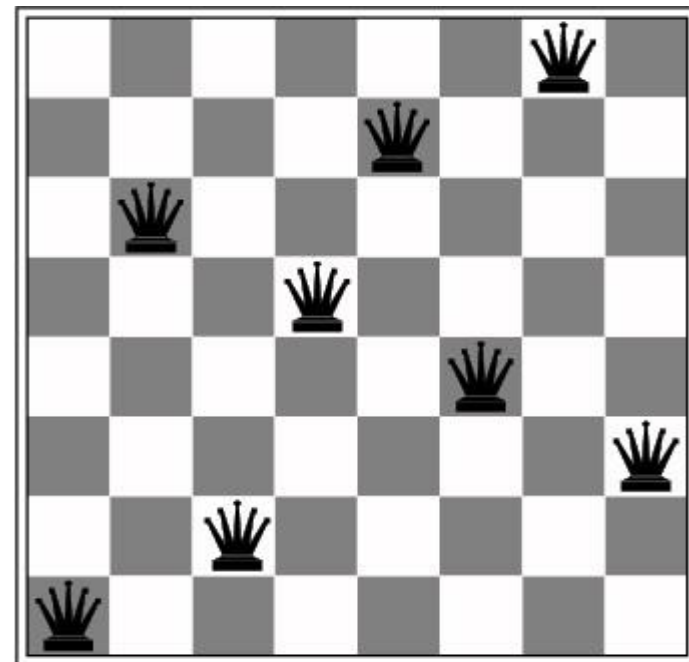
耗散函数 h 是可以彼此攻击的皇后对的数量，
该函数的全局最小值为0。



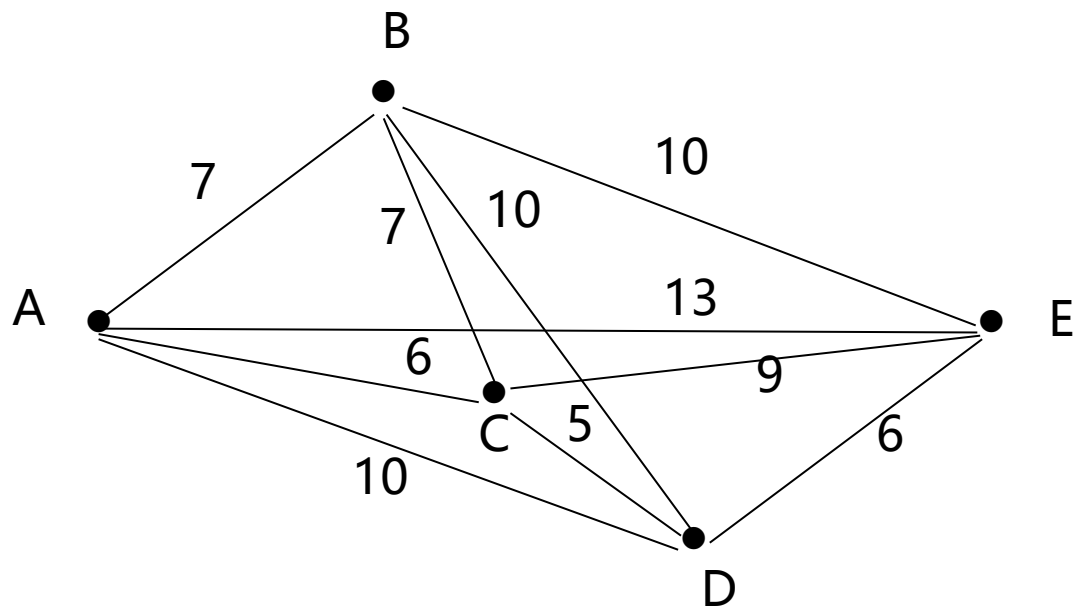
$h = 17$ 的状态

5步

$h = 1$ 的状态
局部极值的一种状态



例：5城市旅行商问题



设初始的可能解: $x_0 = (a, b, c, d, e)$

$$f(x_b) = f(x_0) = 38$$

通过交换两个城市获得邻域

$$P = \{(a, c, b, d, e), (a, d, c, b, e), (a, e, c, d, b), \\ (a, b, d, c, e), (a, b, e, d, c), (a, b, c, e, d)\}$$

设每次随机从P中选择一个邻居。



第一次循环

从P中选择一个元素,

假设 $x_n = (a, c, b, d, e)$,

$f(x_n) = 42$,

$f(x_n) > f(x_b)$,

$P = P - \{x_n\}$

$= \{(a, d, c, b, e), (a, e, c, d, b), (a, b, d, c, e),$
 $(a, b, e, d, c), (a, b, c, e, d)\}$



第二次循环

从P中选择一个元素,

假设 $x_n = (a, d, c, b, e)$,

$f(x_n) = 45$,

$f(x_n) > f(x_b)$,

$P = P - \{x_n\}$

$= \{(a, e, c, d, b), (a, b, d, c, e), (a, b, e, d, c),$
 $(a, b, c, e, d)\}$



第三次循环

从P中选择一个元素,

假设 $x_n = (a, e, c, d, b)$,

$f(x_n) = 44$,

$f(x_n) > f(x_b)$,

$P = P - \{x_n\}$

$= \{(a, b, d, c, e), (a, b, e, d, c), (a, b, c, e, d)\}$



第四次循环

从P中选择一个元素,

假设 $x_n = (a, b, d, c, e)$,

$f(x_n) = 44$,

$f(x_n) > f(x_b)$,

$P = P - \{x_n\} = \{(a, b, e, d, c), (a, b, c, e, d)\}$



第五次循环

从P中选择一个元素,

假设 $x_n = (a, b, e, d, c)$,

$f(x_n) = 34$,

$f(x_n) < f(x_b)$,

$x_b = (a, b, e, d, c)$,

$P = \{(a, e, b, d, c), (a, d, e, b, c), (a, c, e, d, b),$



$(a, b, d, e, c), (a, b, c, d, e), (a, b, e, c, d)\}$

第六次循环

从P中选择一个元素,


假设 $x_n = (a, e, b, d, c)$,

$f(x_n) = 44$,

$f(x_n) > f(x_b)$,

$P = P - \{x_n\}$

$= \{(a, d, e, b, c), (a, c, e, d, b), (a, b, d, e, c),$

 $(a, b, c, d, e), (a, b, e, c, d)\}$

第七次循环

从P中选择一个元素,

假设 $x_n = (a, d, e, b, c)$,

$f(x_n) = 39, f(x_n) > f(x_b)$,

$P = P - \{x_n\}$

$= \{(a, c, e, d, b), (a, b, d, e, c), (a, b, c, d, e),$
 $(a, b, e, c, d)\}$



第八次循环

从P中选择一个元素,

假设 $x_n = (a, c, e, d, b)$,

$f(x_n) = 38$,

$f(x_n) > f(x_b)$,

$P = P - \{x_n\}$

$= \{(a, b, d, e, c), (a, b, c, d, e), (a, b, e, c, d)\}$



第九次循环

从P中选择一个元素,

假设 $x_n = (a, b, d, e, c)$,

$f(x_n) = 38$,

$f(x_n) > f(x_b)$,

$P = P - \{x_n\} = \{(a, b, c, d, e), (a, b, e, c, d)\}$



第十次循环

从P中选择一个元素,

假设 $x_n = (a, b, c, d, e)$,

$f(x_n) = 38$,

$f(x_n) > f(x_b)$,

$P = P - \{x_n\} = \{(a, b, e, c, d)\}$



第十一次循环

从P中选择一个元素，

假设 $x_n = (a, b, e, c, d)$,

$f(x_n) = 41$,

$f(x_n) > f(x_b)$,

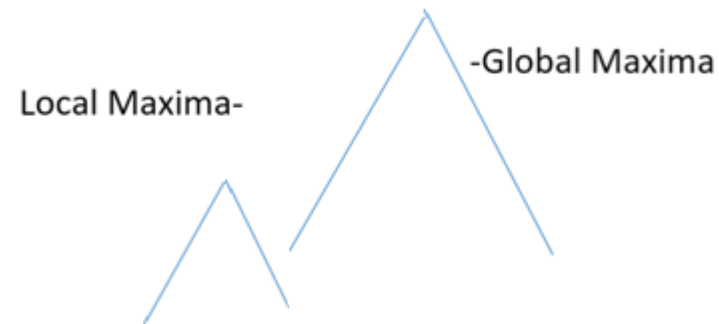
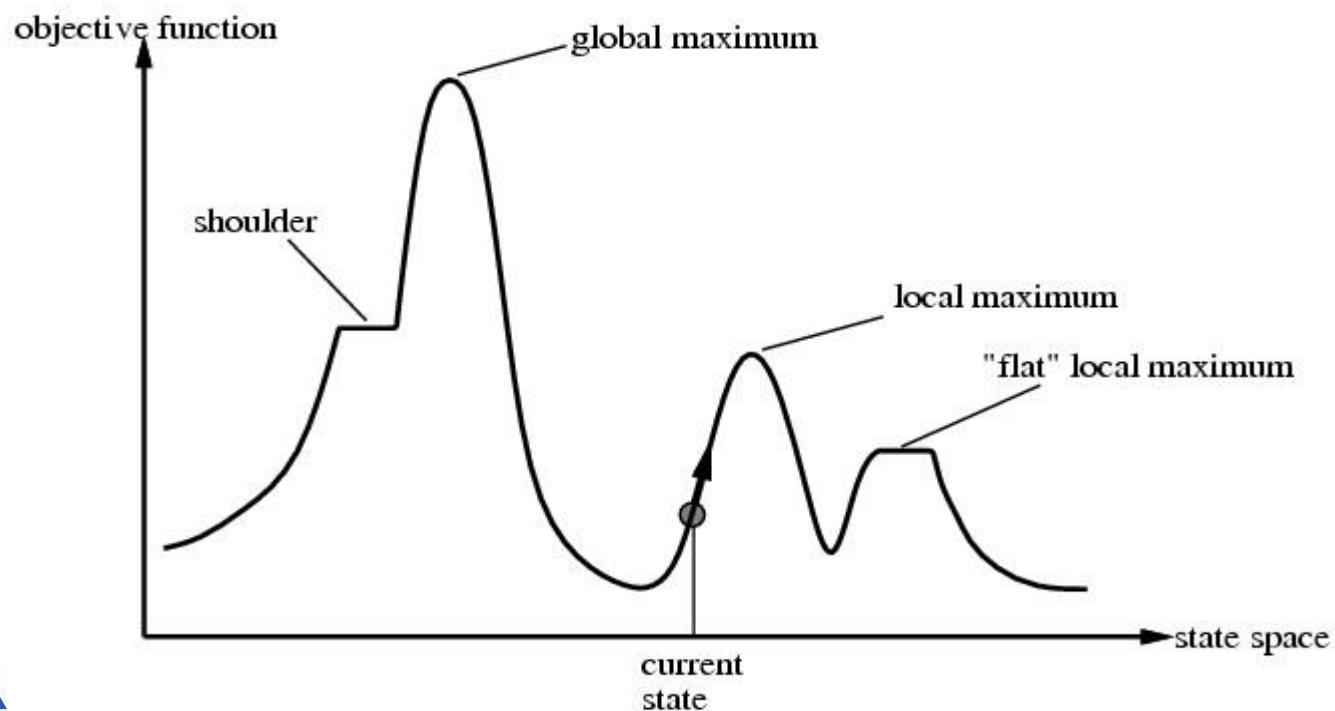
$P = P - \{x_n\} = \{\}$

P等于空，算法结束，

得到结果为 $x_b = (a, b, e, d, c)$, $f(x_b) = 34$ 。

存在的问题

■ 局部最优问题(Local Maxima)



It is a state which is better than all the existing neighboring states but is not the peak point of the hill.



Plateau:



高原：

A plateau is a flat area and therefore no neighboring state in the search space exists so that it is better than the current point because all of them lie on the same plane.

高原是一个平坦的区域，因此搜索空间中不存在任何相邻状态，因此它比当前点更好，因为它们都位于同一平面上。

Ridge:

山脊：

It is an area in the search space which is higher than the surrounding areas but cannot be searched in a simple move.

它是搜索空间中比周围区域高的区域，但无法通过简单的移动进行搜索。



解决方法

- 每次并不一定选择邻域内最优的点，而是依据一定的概率，从邻域内选择一个点，指标函数优的点，被选中的概率比较大，而指标函数差的点，被选中的概率比较小。



选择概率的计算

- 设求最大值：

$$P_{\max}(x_i) = \frac{f(x_i)}{\sum_{x_j \in N(x)} f(x_j)}$$



选择概率的计算

- 当求最小值时：

$$\begin{aligned} P_{\min}(x_i) &= \frac{1 - P_{\max}(x_i)}{\sum_{x_j \in N(x)} (1 - P_{\max}(x_j))} \\ &= \frac{1}{|N(x)| - 1} (1 - P_{\max}(x_i)) \end{aligned}$$



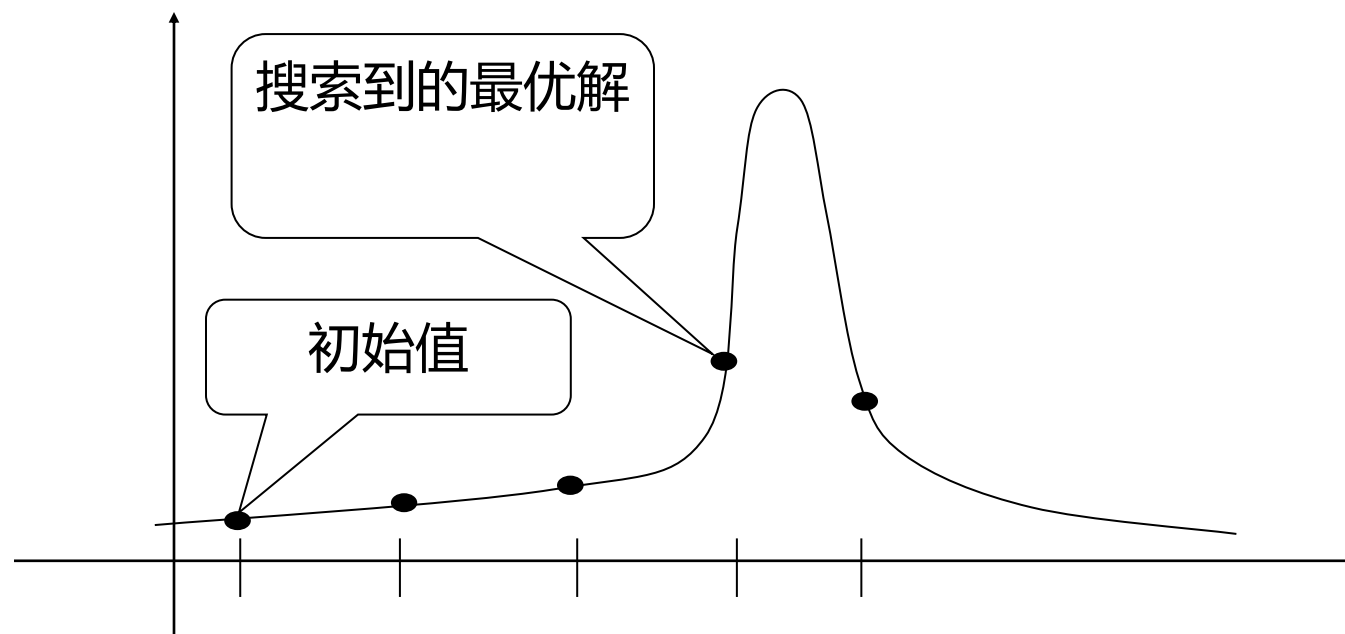
局部搜索算法1 (Local Search 1)

- 1, 随机的选择一个初始的可能解 $x_0 \in D$, $x_b = x_0$, $P = N(x_b)$
- 2, 如果不满足结束条件, 则
- 3, **Begin**
- 4, 对于所有的 $x \in P$ 计算指标函数 $f(x)$, 并计算每一个点 x 的概率
- 5, 依计算的概率值, 从 P 中随机选择一个点
 x_n , $x_b = x_n$, $P = N(x_b)$, 转2
- 6, **End**
- 7, 输出计算结果
- 8, 结束



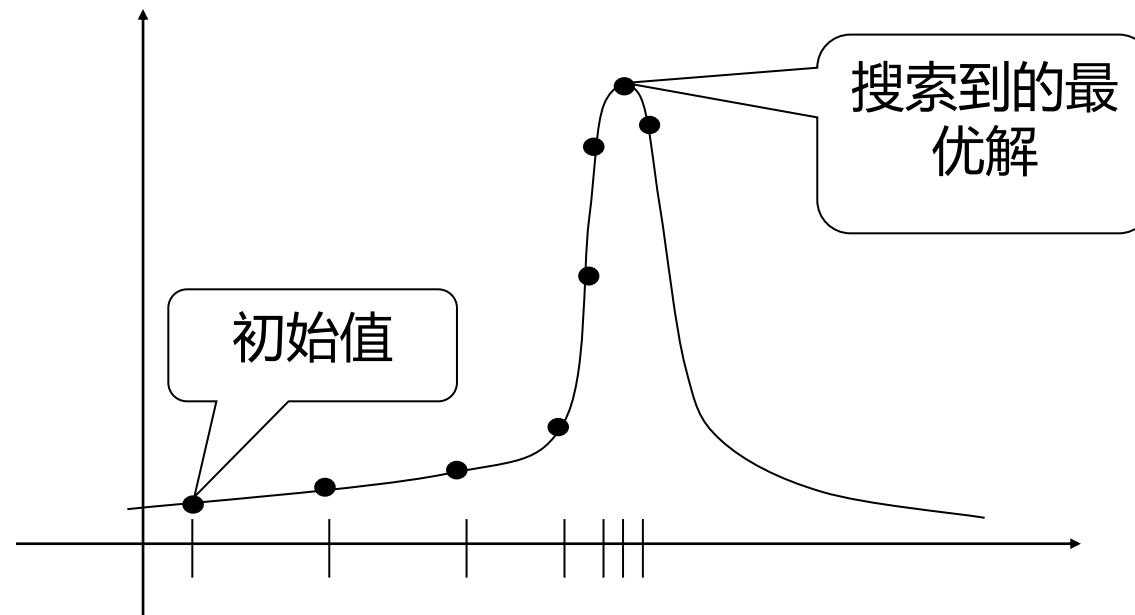
存在的问题

■ 步长问题



解决方法

■ 变步长



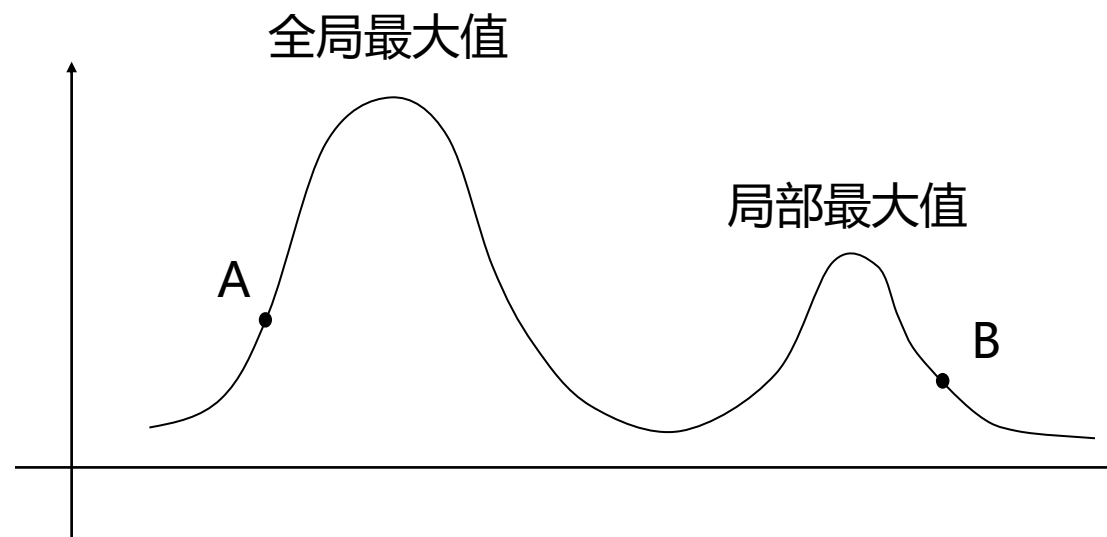
局部搜索算法2 (Local Search 2)

- 1, 随机的选择一个初始的可能解 $x_0 \in D$, $x_b = x_0$, 确定一个初始步长计算 $P = N(x_b)$
- 2, 如果不满足结束条件, 则
- 3, **Begin**
- 4, 选择 P 的一个子集 P' , x_n 为 P' 中的最优解
- 5, 如果 $f(x_n) < f(x_b)$, 则 $x_b = x_n$
- 6, 按照某种策略改变步长, 计算 $P = N(x_b)$,
 转2
- 7, 否则 $P = P - P'$, 转2。
- 8, **End**
- 9, 输出计算结果
- 10, 结束



存在问题

- 起始点问题



解决方法

- 随机的生成一些初始点，从每个初始点出发进行搜索，找到各自的最优解。再从这些最优解中选择一个最好的结果作为最终的结果。



局部搜索算法3 (Local Search 3)

- 1, $k = 0$
- 2, 随机的选择一个初始的可能解 $x_0 \in D$, $x_b = x_0$, $P = N(x_b)$
- 3, 如果不满足结束条件, 则
- 4, Begin
- 5, 选择 P 的一个子集 P' , x_n 为 P' 中的最优解
- 6, 如果 $f(x_n) < f(x_b)$, 则 $x_b = x_n$, $P = N(x_b)$, 转3
- 7, 否则 $P = P - P'$, 转3。
- 8, End
- 9, $k = k + 1$
- 10, 如果 k 达到了指定的次数, 则从 k 个结果中选择一个最好的结果输出, 否则转2
- 11, 输出结果
- 12, 结束



- **随机爬山法**

爬山法的变形。

随机爬山法在上山移动中随机选择下一步；被选中的概率可能随着上山移动的陡峭程度不同而不同。这种算法通常比最陡上升算法的收敛速度慢不少，但是在某些状态空间地形图上它能找到更好的解。

- **首选爬山法**

实现了随机爬山法，随机地生成后继节点直到生成一个优于当前节点的后继。这个算法在后继节点很多的时候（比如上千个）是个好策略。

- **随机重启爬山法**

之前的三个爬山法都是不完备的，经常会在局部极大值卡住。

随机重启爬山法（random restart hill climbing），它通过随机生成初始状态来导引爬山法搜索，直到找到目标。这种算法完备的概率接近1.原因：它最终会生成一个目标状态来作为初始状态。



模拟退火算法

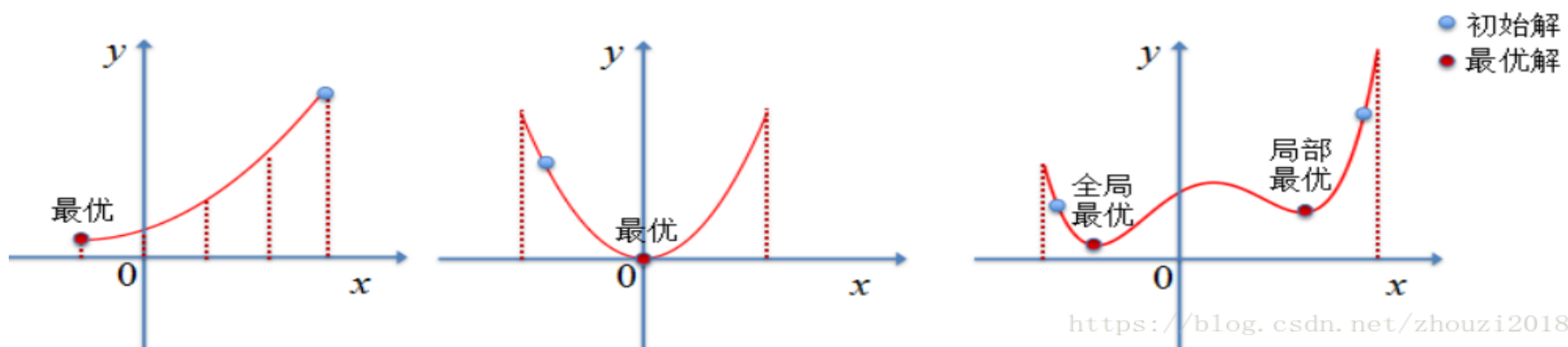
SA, Simulated Annealing



在实际日常中，人们会经常遇到如下问题：在某个给定的定义域X内，求函数f(x)对应的最优值。此处以最小值问题举例（最大值问题可以等价转化成最小值问题），形式化为：

$$\min_{x \in X} f(x).$$

- 1、如果X是离散有限取值，那么可以通过穷取法获得问题的最优解；
- 2、如果X连续，但f(x)是凸的，那可以通过梯度下降等方法获得最优解；
- 3、如果X连续且f(x)非凸，虽说根据已有的近似求解法能够找到问题解，可解是否是最优的还有待考量，很多时候若初始值选择的不好，非常容易陷入局部最优值。



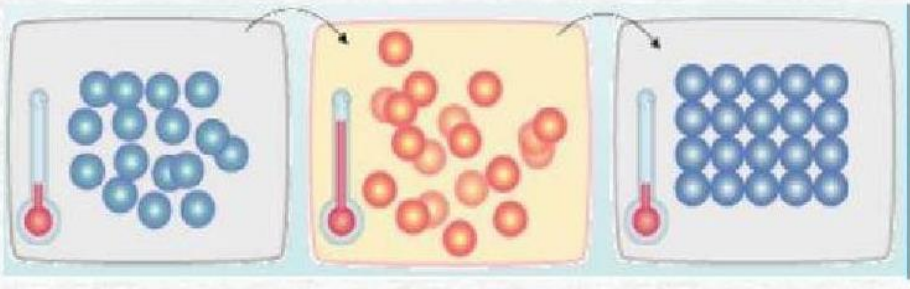
随着日常业务场景的复杂化，第三种问题经常遇见。如何有效地避免局部最优的困扰？



模拟退火也算是启发式算法的一种，具体学习的是冶金学中金属加热-冷却的过程。
由S.Kirkpatrick, C.D.Gelatt和M.P.Vecchi在1983年所发明的，V.Čern在1985年也独立发明此演算法。

模拟退火算法如何模拟**金属退火**的原理？
主要是将**热力学**的理论套用到**统计学**上，将搜寻空间内每一点想像成空气内的分子；分子的能量，就是它本身的动能；而搜寻空间内的每一点，也像空气分子一样带有“能量”，以表示该点对命题的合适程度。先以搜寻空间内一个任意点作起始：每一步先选择一个“邻居”，然后再计算从现有位置到达“邻居”的概率。若概率大于给定的阈值，则跳转到“邻居”；若概率较小，则停留在原位置不动。

SA算法起源于对固体退火过程的模拟。简单而言，在固体退火时，先将固体加热使其温度充分高，再让其徐徐冷却，其物理退火过程由以下三部分组成：加温过程、等温过程、冷却过程。



模拟退火算法与物理退火过程的相似关系

模拟退火	物理退火
解	粒子状态
最优解	能量最低态
设定初温	熔解过程
Metropolis采样过程	等温过程
控制参数的下降	冷却
目标函数	能量



模拟退火算法：结合爬山法和随机行走

把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。

思想：通过允许一些“不好的”移动来避免陷入局部极小点处 但要逐步减小这些移动的幅度和频率

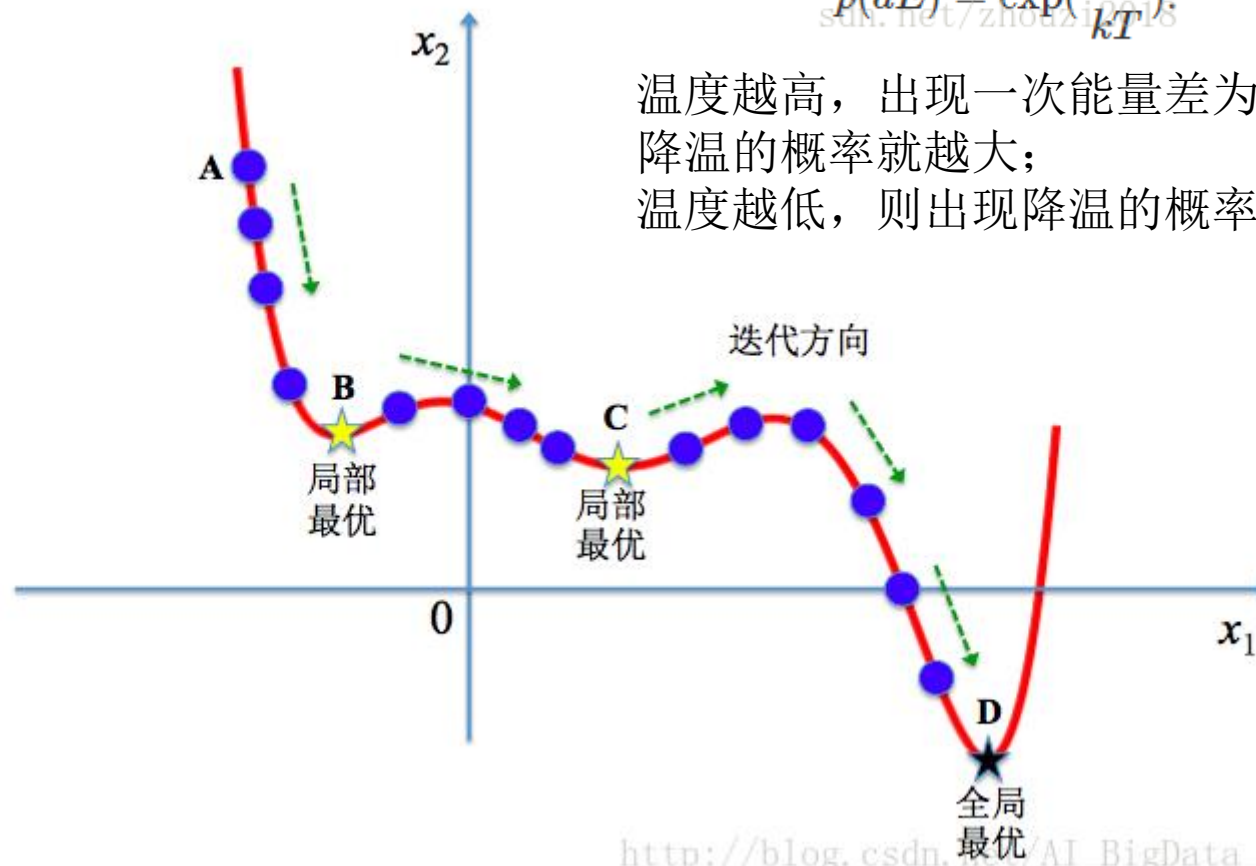
在某“温度” T 下，状态被选中的概率符合Boltzman分布：

$$p(x) = \alpha e^{-\frac{E(x)}{kT}}$$

T 下降充分缓慢 => 达到全局最优状态的概率逼近1

$$p(dE) = \exp\left(-\frac{dE}{kT}\right)$$

温度越高，出现一次能量差为 $p(dE)$ 的降温的概率就越大；
温度越低，则出现降温的概率就越小。



http://blog.csdn.net/AI_BigData_wh



模拟退火算法

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
           schedule, a mapping from time to “temperature”

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

分析概率随 ΔE 和 T 的变化



SA算法的思想为：

- 由初始解 i 和控制参数初值 t 开始，对当前解重复
产生新解

→ 计算目标函数差

→ 接受或舍弃

的迭代，

- 并逐步衰减 t 值，
- 算法终止时的当前解即为所得近似最优解，
- 这是基于蒙特卡罗迭代求解法的一种启发式随机搜索过程。



◆ 基本步骤

给定初温 $t = t_0$, 随机产生初始状态 $s = s_0$, 令 $k = 0$;

Repeat

Repeat

产生新状态 $s_j = \text{Genete}(s)$;

if $\min\{1, \exp[-(C(s_j) - C(s))/t_k]\} \geq \text{randrom}[0, 1]$ $s = s_j$;

Until 抽样稳定准则满足;

退温 $t_{k+1} = \text{update}(t_k)$ 并令 $k = k + 1$;

Until 算法终止准则满足;

输出算法搜索结果。



模拟退火算法的基本步骤

◆ 影响优化结果的主要因素

给定初温 $t=t_0$, 随机产生初始状态 $s=s_0$, 令 $k=0$;

Repeat

Repeat

产生新状态 $s_j = \text{Genete}(s)$;

if $\min\{1, \exp[-(C(s_j) - C(s))/t_k]\} \geq \text{randrom}[0,1]$ $s = s_j$;

Until 抽样稳定准则满足;

退温 $t_{k+1} = \text{update}(t_k)$ 并令 $k = k + 1$;

Until 算法终止准则满足;

输出算法搜索结果。

三函数两准则

初始温度



◆ 模拟退火算法的步骤

Step1 设定初始温度 $t = t_{\max}$, 任选初始解 $r = r_0$

Step2 内循环

Step2.1 从 r 的邻域中随机选一个解 r_t ,

如 r_t 对应目标函数值较小, 则令 $r = r_t$

$\exp(-(E(r_t) - E(r))/t) > \text{random}(0, 1)$, 则令 $r = r_t$.

Step2.2 不满足内循环停止条件时, 重复Step2.1

Step3 外循环

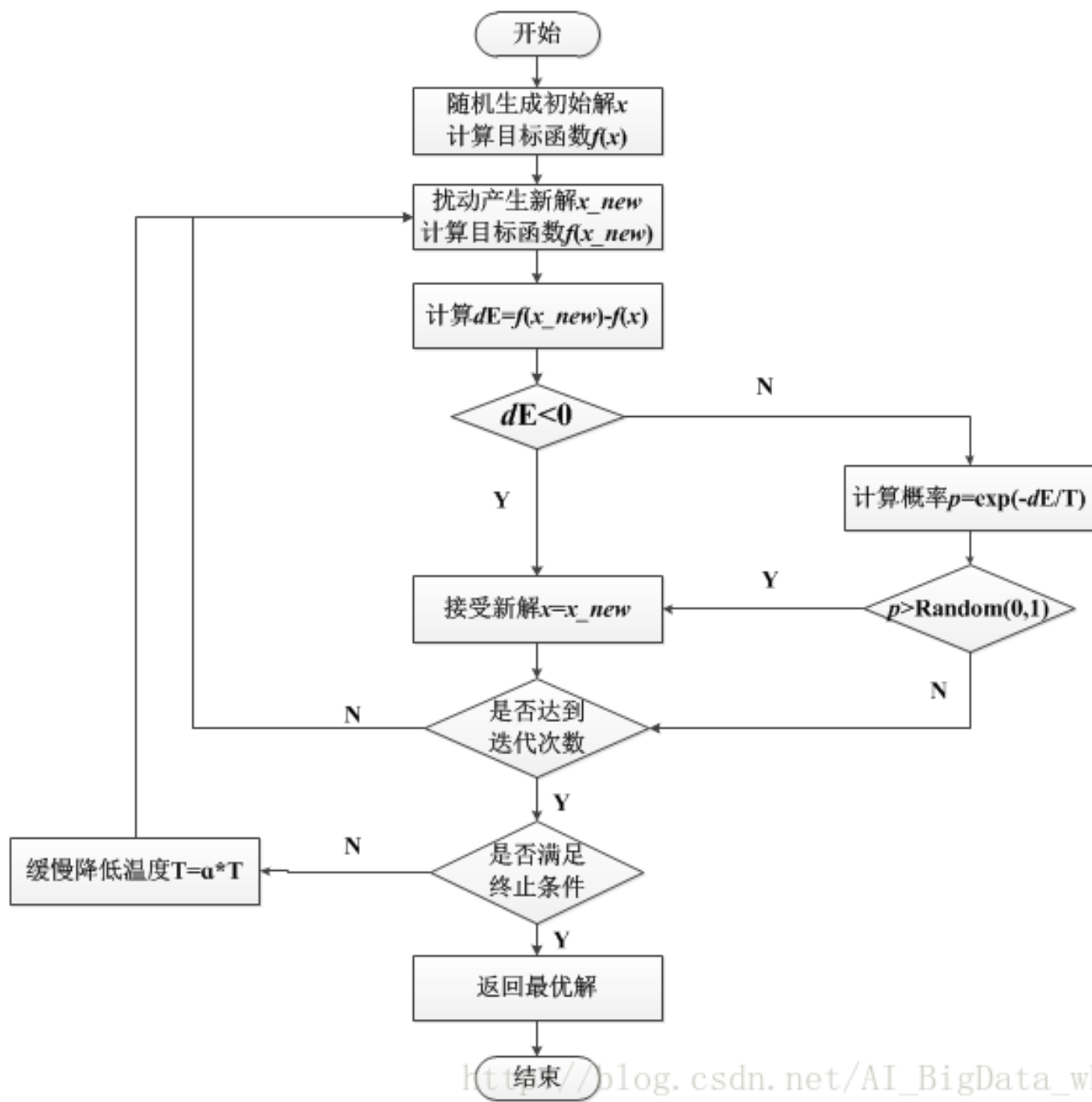
Step3.1 降温 $t = \text{decrease}(t)$

Step3.2 如不满足外循环停止条件, 则转Step2; 否则算法结束

1. 目标函数均值稳定
2. 连续若干步的目标值变化较小
3. 固定的抽样步数

1. 达到终止温度
2. 达到迭代次数
3. 最优值连续若干步保持不变



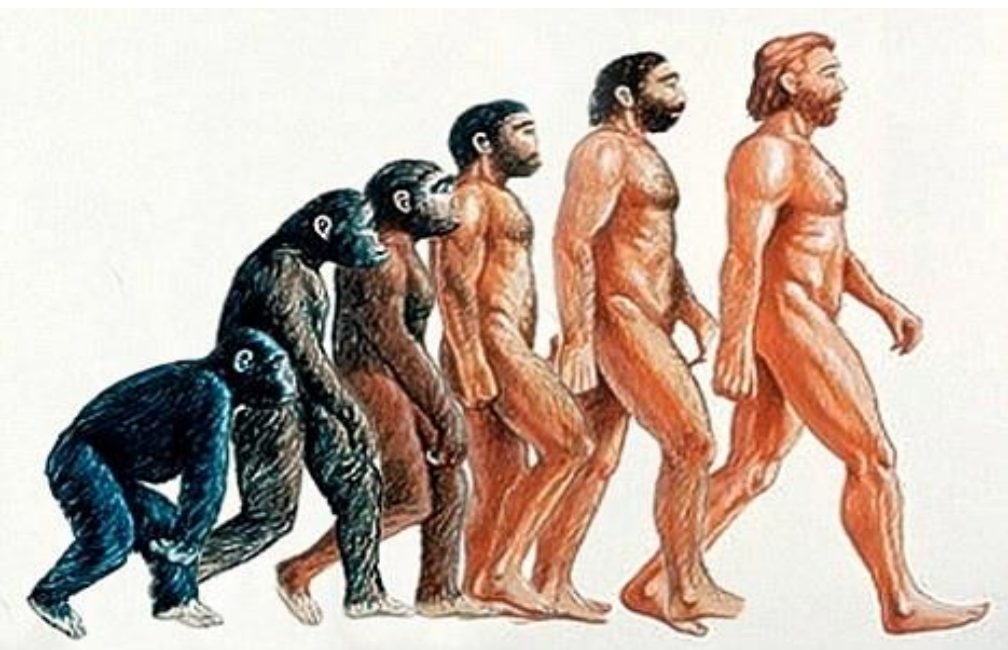


遗传算法

Genetic algorithm, GA



遗传算法 (GA)



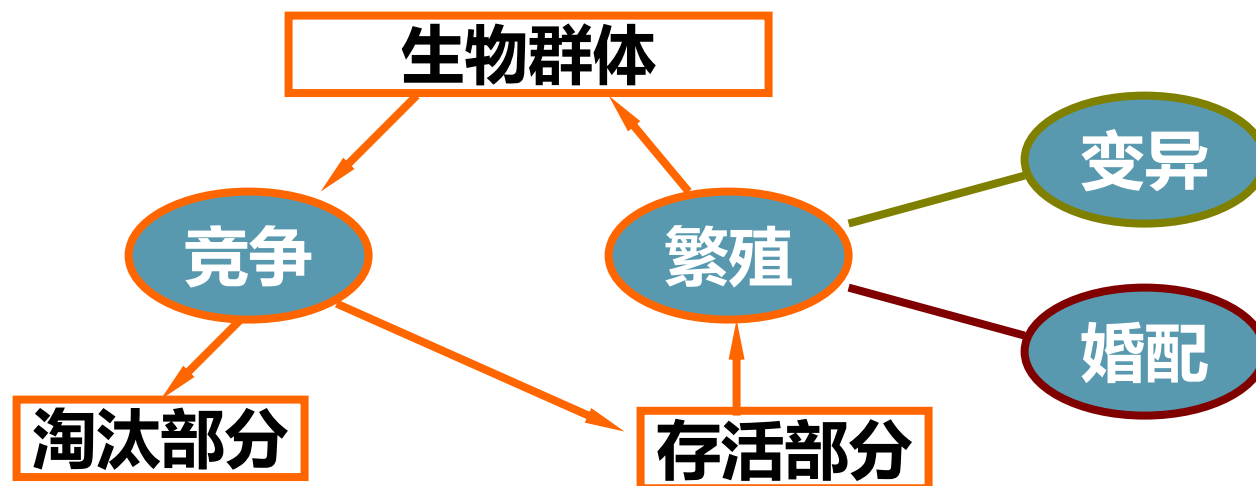
模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，它通过模拟自然进化过程搜索最优解。

- 遗传算法是在生物进化的启示下得到的一种搜索和自适应算法(以字符串表示状态空间)。
- 遗传算法模拟了生物的繁殖、交配和变异现象，从任意一初始种群出发，产生一群新的更适应环境的后代。这样一代一代不断繁殖、进化，最后收敛到一个最适应环境的个体上。
- 遗传算法对于复杂的优化问题无需建模和进行复杂运算，只需要利用遗传算法的算子就能寻找到问题的最优解或满意解。



遗传算法的基本思想

- 1975年Holland
- 基本思想：物竞天择、适者生存



遗传算法

- 通过结合两个父状态来生成后继状态
- 从k个随机生成的状态（**种群**）开始
- 每个状态（**个体**）用一个有限长度的字符串表示（**编码**），通常是0、1串
- 评价函数（**适应度函数**）给出每个状态的评价值（**适应值**），好的状态返回的适应值较高。
- 通过选择、杂交、变异获得下一代



遗传算法与自然进化的比较

自然界	遗传算法
染色体	字符串
基因	字符, 特征
等位基因(allele)	特征值
染色体位置(locus)	字符串位置
基因型(genotype)	结构
表型(phenotype)	参数集, 译码结构

遗传算法本质是一种高效、并行、全局搜索的方法
能在搜索过程中自动获取和积累有关搜索空间的知识
并自适应地控制搜索过程以求得最佳解

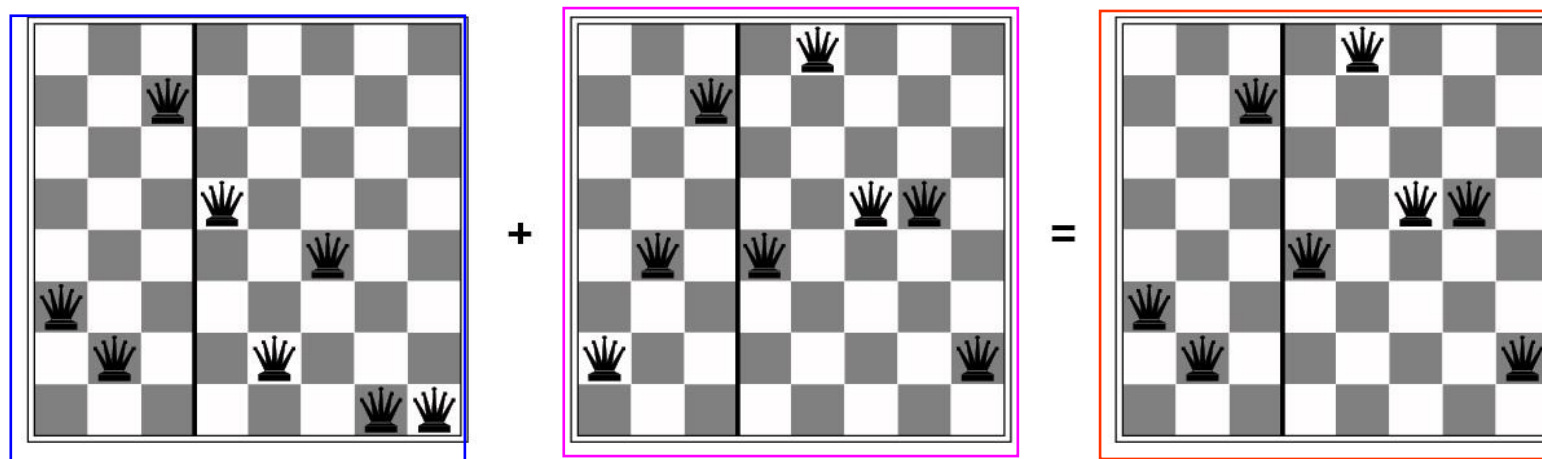
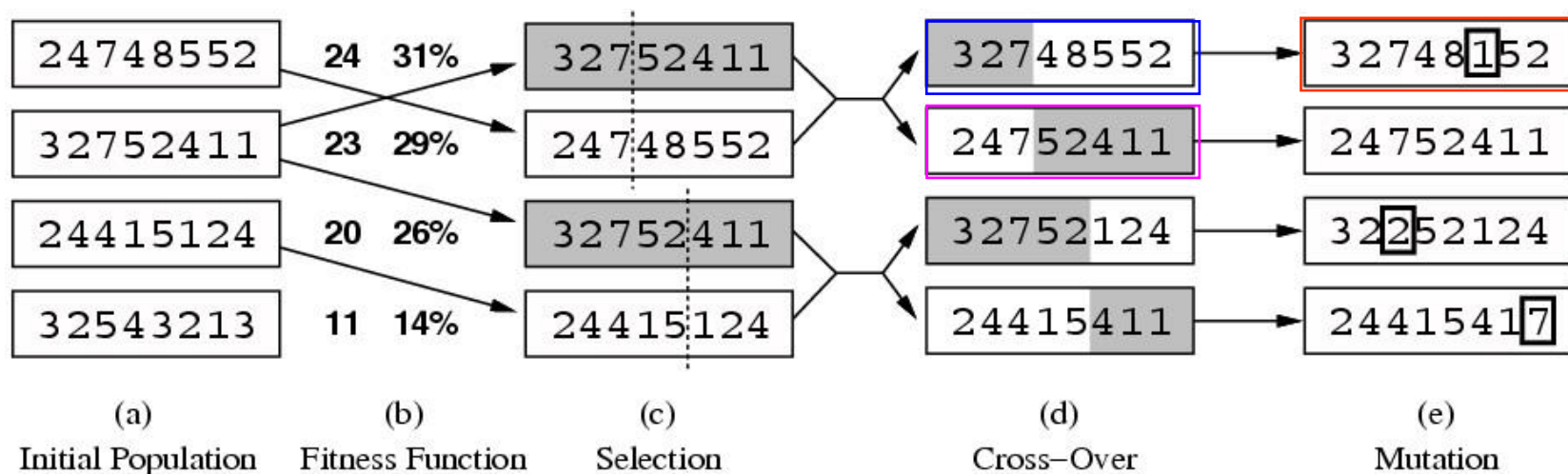


遗传算法

- 对简单遗传算法来说，主要涉及的内容有：
 - 编码和初始群体生成
 - 群体的评价（适应度函数）
 - 个体选择
 - 交换和变异



遗传算法 - 八皇后问题



遗传算法

function GENETIC-ALGORITHM(*population*, FITNESS-FN) **returns** an individual

inputs: *population*, a set of individuals

FITNESS-FN, a function that measures the fitness of an individual

repeat

new_population \leftarrow empty set

for $i = 1$ **to** SIZE(*population*) **do**

$x \leftarrow$ RANDOM-SELECTION(*population*, FITNESS-FN)

$y \leftarrow$ RANDOM-SELECTION(*population*, FITNESS-FN)

child \leftarrow REPRODUCE(x, y)

if (small random probability) **then** *child* \leftarrow MUTATE(*child*)

add *child* to *new_population*

population \leftarrow *new_population*

until some individual is fit enough, or enough time has elapsed

return the best individual in *population*, according to FITNESS-FN

function REPRODUCE(x, y) **returns** an individual

inputs: x, y , parent individuals

$n \leftarrow$ LENGTH(x); $c \leftarrow$ random number from 1 to n

return APPEND(SUBSTRING($x, 1, c$), SUBSTRING($y, c + 1, n$))

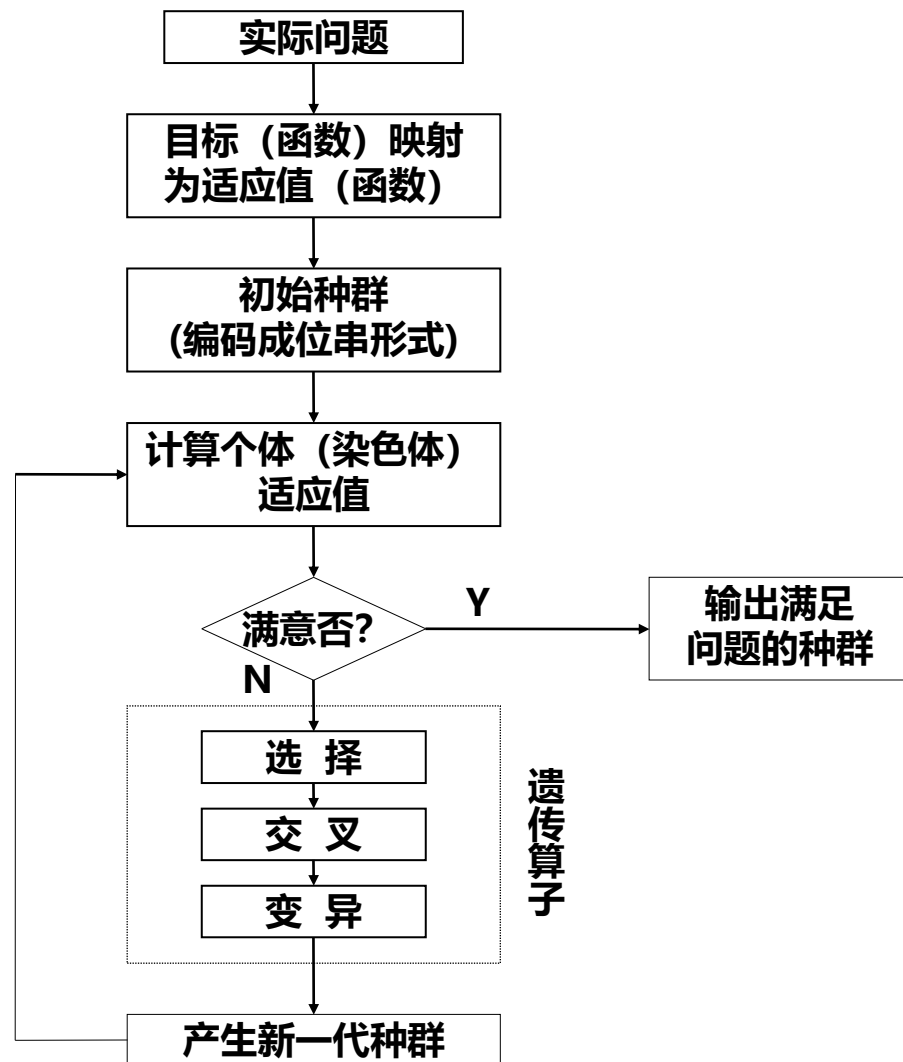


遗传算法

- 与传统的优化算法相比，遗传算法主要有以下几个不同之处
 1. 遗传算法不是直接作用在参变量集上而是利用参变量集的某种编码
 2. 遗传算法不是从单个点，而是从一个点的群体开始搜索；
 3. 遗传算法利用适应值信息，无须导数或其它辅助信息；
 4. 遗传算法利用概率转移规则，而非确定性规则。
- 遗传算法的优越性主要表现在：
 1. 在搜索过程中不容易陷入局部最优，即使所定义的适应函数是不连续的、非规则的或有噪声的情况下，它也能以很大的概率找到整体最优解；
 2. 由于它固有的并行性，遗传算法非常适用于大规模并行计算机。



遗传算法流程

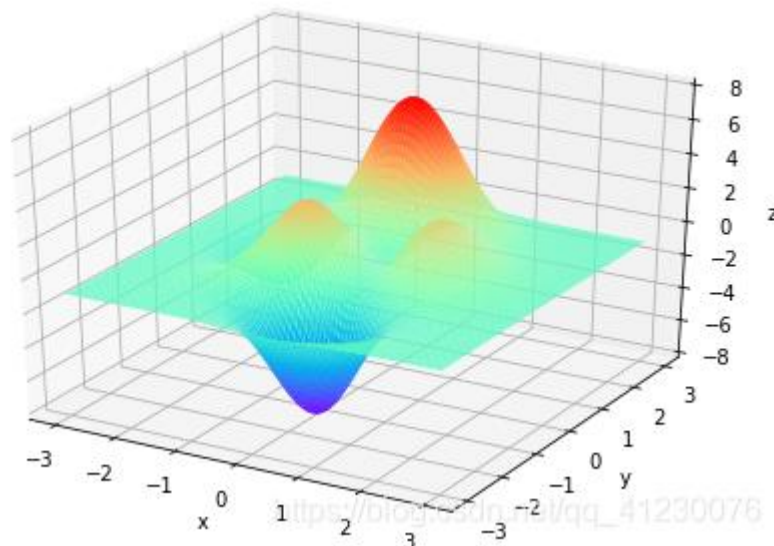


遗传算法应用

遗传算法的有趣应用很多，诸如寻路问题，8数码问题，囚犯困境，动作控制，找圆心问题（在一个不规则的多边形中，寻找一个包含在该多边形内的最大圆圈的圆心），TSP问题，生产调度问题，人工生命模拟等。

遗传算法中每一条染色体，对应着遗传算法的一个解决方案，一般我们用适应性函数（fitness function）来衡量这个解决方案的优劣。所以从一个基因组到其解的适应度形成一个映射。可以把遗传算法的过程看作是一个在多元函数里面求最优解的过程。可以这样想象，这个多维曲面里面有数不清的“山峰”，而这些山峰所对应的就是局部最优解。而其中也会有一个“山峰”的海拔最高的，那么这个就是全局最优解。而遗传算法的任务就是尽量爬到最高峰，而不是陷落在一些小山峰。

（另外，值得注意的是遗传算法不一定要找“最高的山峰”，如果问题的适应度评价越小越好的话，那么全局最优解就是函数的最小值，对应的，遗传算法所要找的就是“最深的谷底”）



相关术语

基因型(genotype): 性状染色体的内部表现。

表现型(phenotype): 染色体决定的性状的外部表现, 或者说, 根据基因型形成的个体的外部表现。

进化(evolution): 种群逐渐适应生存环境, 品质不断得到改良。生物的进化是以种群的形式进行的。

适应度(fitness): 度量某个物种对于生存环境的适应程度。

选择(selection): 以一定的概率从种群中选择若干个个体。一般, 选择过程是一种基于适应度的优胜劣汰的过程。

复制(reproduction): 细胞分裂时, 遗传物质DNA通过复制而转移到新产生的细胞中, 新细胞就继承了旧细胞的基因。

交叉(crossover): 两个染色体的某一相同位置处DNA被切断, 前后两串分别交叉组合形成两个新的染色体。也称基因重组或杂交。

变异(mutation): 复制时可能 (很小的概率) 产生某些复制差错, 变异产生新的染色体, 表现出新的性状。

编码(coding): DNA中遗传信息在一个长链上按一定的模式排列。遗传编码可看作从表现型到基因型的映射。

解码(decoding): 基因型到表现型的映射。

个体 (individual) : 指染色体带有特征的实体。

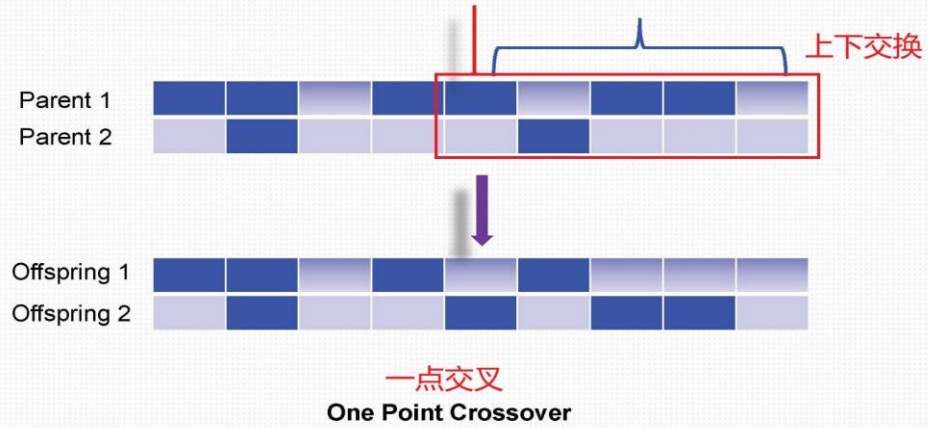
种群 (population) : 个体的集合, 该集合内个体数称为种群的大小。

遗传算法

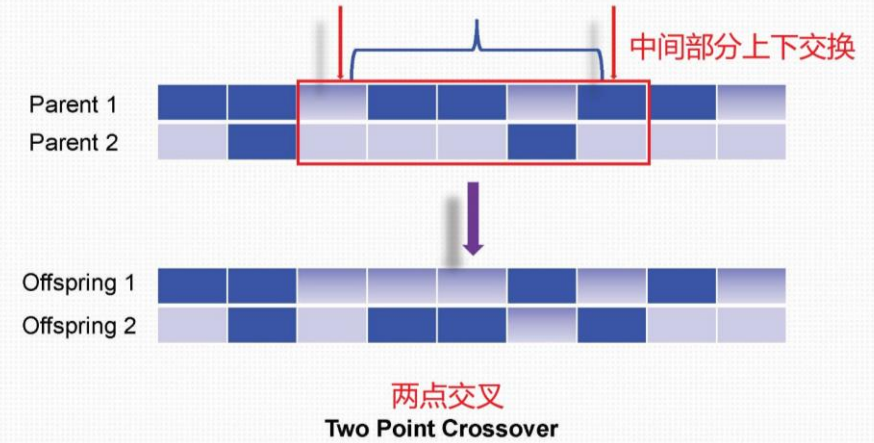
- 遗传算法先将搜索结构编码为字符串形式, 每个字符串结构被称为个体。
- 然后对一组字符串结构(被称为一个群体)进行循环操作。每次循环被称作一代, 包括一个保存字符串中较优结构的过程和一个有结构的、随机的字符串间的信息交换过程。
- 与自然界相似, 遗传算法对求解问题的本身一无所知, 它所需要的仅是对算法所产生的每个染色体进行评价, 并基于适应值来选择染色体, 使适应性好的染色体有更多的繁殖机会。
- 在遗传算法中, 位字符串扮演染色体的作用, 单个位扮演了基因的作用, 随机产生一个个体字符串的初始群体, 每个个体给予一个数值评价, 称为适应度, 取消低适应度的个体, 选择高适应度的个体参加操作。
- 常用的遗传算子有复制、杂交、变异和反转。



Crossover I

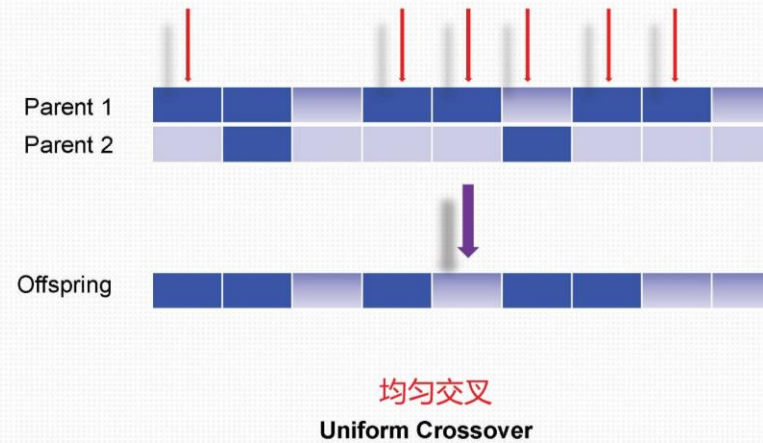


Crossover II

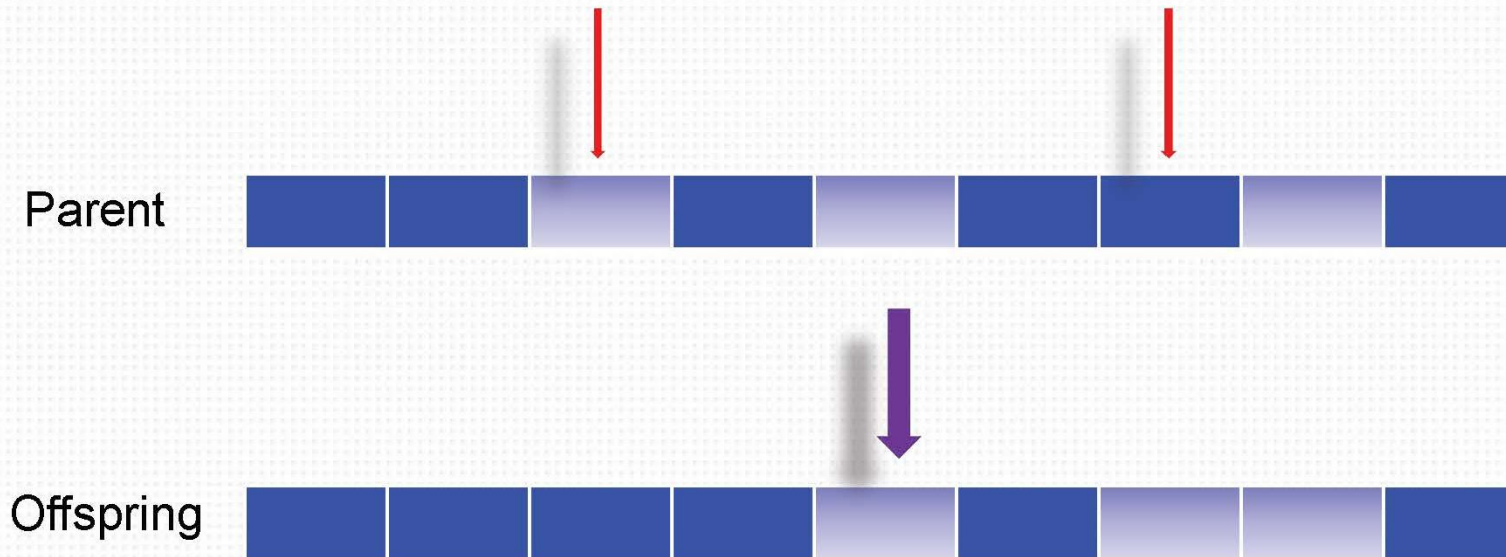


Crossover III

父母同为深或浅色，子代与父母一致；若父母对应位置颜色不一致，子代有50%的几率为深或浅



Mutation

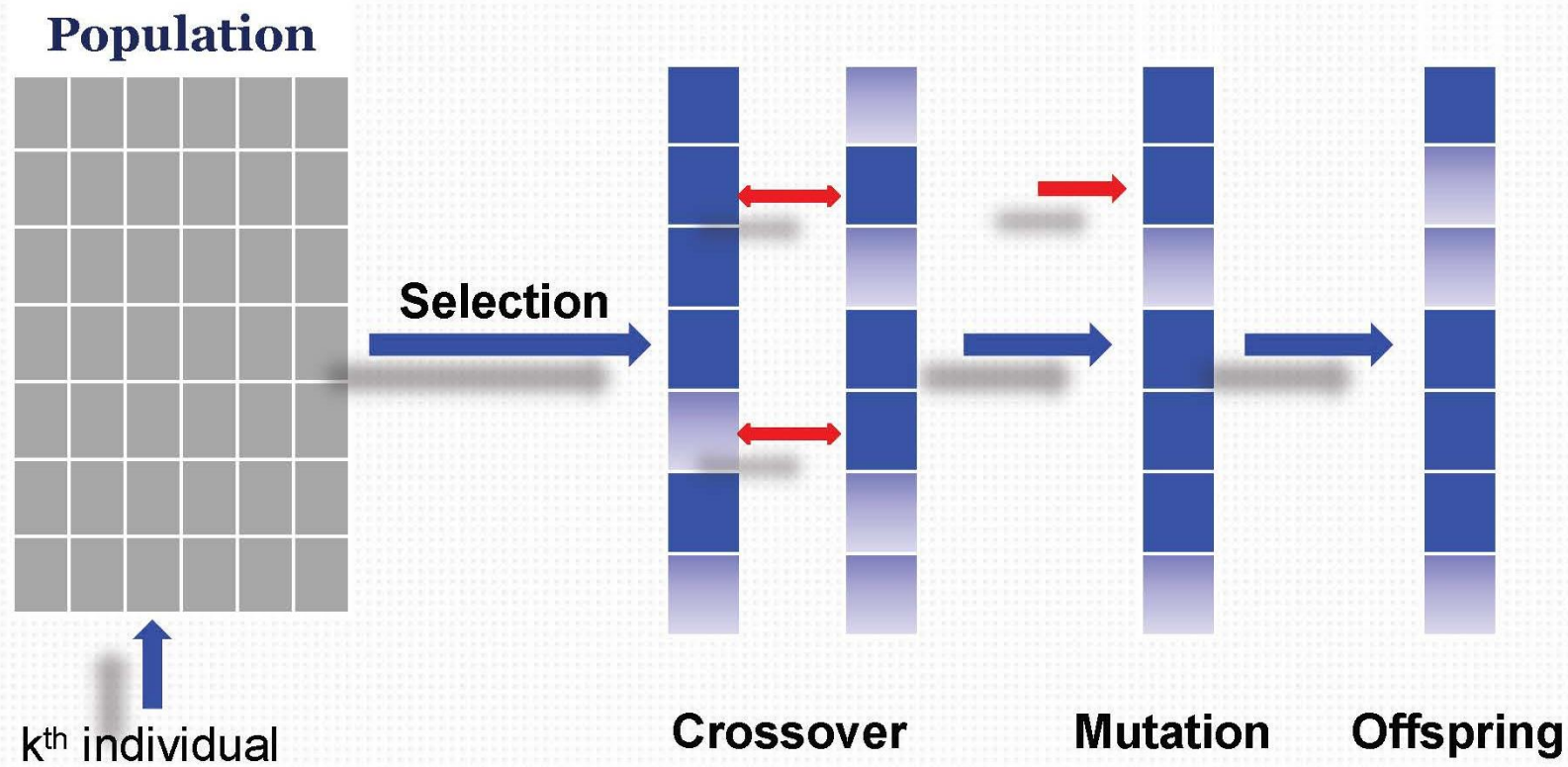


Mutation vs. Crossover

Mutation is mainly used to maintain the genetic diversity.



The Complete Picture



连续空间的搜索问题

- 例子：在罗马尼亚建三个新机场，使地图上的每个城市到它们的距离之和最小。
 - 问题的状态空间 = $\{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$
 - 目标函数 = 距离之和 $X \leftarrow X + \alpha \nabla f(X)$
 - 搜索策略：（利用梯度）来寻找局部极小值
 - 牛顿法
- 约束最优化（constrained optimization）
 - 线性规划(约束是线性不等式，目标函数也是线性的)

在线搜索智能体和未知环境

■ 脱机搜索vs.在线搜索

- 脱机：涉足实际问题之前就计算好完整的方案，然后不需要感知就能够执行行动方案。
- 在线搜索：通过计算和行动的交叉而完成的

■ 在线搜索对探索问题是个必要的思想

- 范例：放在新建大楼里的机器人，要求它探索大楼，绘制出一张从A到B的地图。



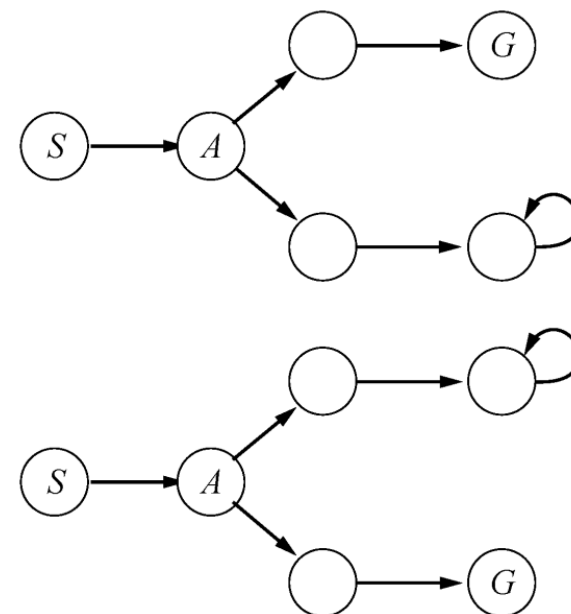
在线搜索问题

- 假设智能体仅知道以下信息：
 - $ACTIONS(s)$: 返回状态 s 下可能行动的列表
 - 单步耗散函数 $c(s, a, s')$
 - $GOAL-TEST(s)$
- 假设智能体总能认出它以前已到达过的状态，以及它的行动是确定的。
- 代价 = 智能体实际旅行经过的路径的总耗散
- 竞争率 = 代价与实际最短路径的比值，理想值为1。
- 但是，在线搜索有可能陷入死路。



死路状态

- 不存在能够在所有可能的状态空间中都避免死路的算法



两个能导致智能体陷入死路的状态空间（敌对参数）



可安全探索的状态空间

- 定义：从每个可到达的状态出发都能到达某些目标状态。
- 例子：具有可逆行动的状态空间，迷宫、八角游戏...



在线搜索智能体

- 在线搜索中智能体只能扩展它实际占据的节点，因此搜索算法最好是局部扩展型的，如深度优先搜索DFS。
- 深度优先联机搜索的难点在于，当智能体尝试完一个状态的所有动作后，不得不实际地回溯。



在线深度优先搜索算法

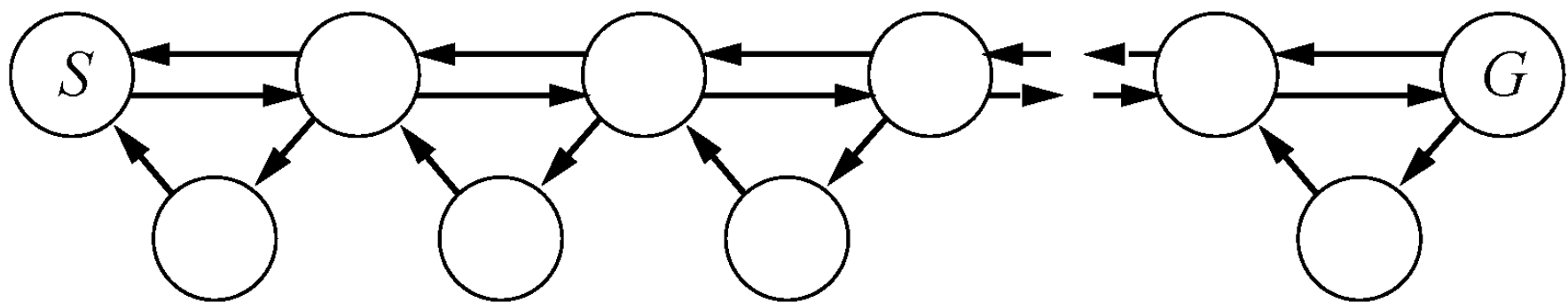
```
function ONLINE_DFS-AGENT( $s'$ ) return an action
  input:  $s'$ , a percept identifying current state
  static: result, a table indexed by action and state, initially empty
         unexplored, a table that lists for each visited state, the action not yet tried
         unbacktracked, a table that lists for each visited state, the backtrack not yet tried
          $s, a$ , the previous state and action, initially null

  if GOAL-TEST( $s'$ ) then return stop
  if  $s'$  is a new state then  $\text{unexplored}[s'] \leftarrow \text{ACTIONS}(s')$ 
  if  $s$  is not null then do
     $\text{result}[a, s] \leftarrow s'$ 
    add  $s$  to the front of  $\text{unbacktracked}[s']$ 
  if  $\text{unexplored}[s']$  is empty then
    if  $\text{unbacktracked}[s']$  is empty then return stop
    else  $a \leftarrow$  an action  $b$  such that  $\text{result}[s', b] = \text{POP}(\text{unbacktracked}[s'])$ 
  else  $a \leftarrow \text{POP}(\text{unexplored}[s'])$ 
   $s \leftarrow s'$ 
  return  $a$ 
```



在线局部搜索

- 在线爬山搜索无法使用随机重新开始。
- 替代方法之一：使用随机行走来探索环境

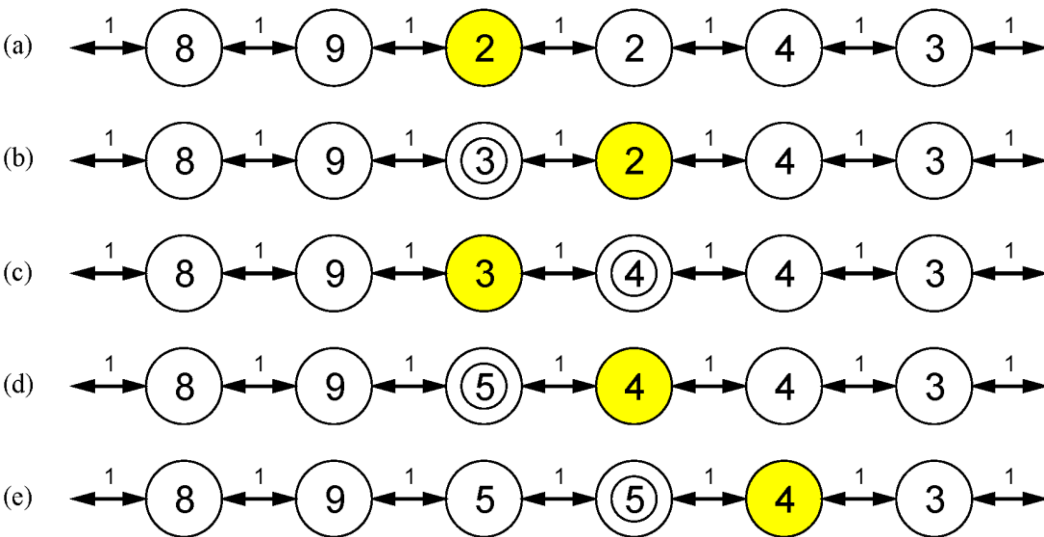


一个使随机行走耗尽指数级步骤才能找到目标的环境



替代方法之二：提高内存利用率

- 基本思想：存储一个从每个访问过的状态到达目标的耗散中的“当前最佳估计耗散” $H(s)$ 。 $H(s)$ 从启发式估计 $h(s)$ 出发，在智能体从状态空间中获得经验时对 $H(s)$ 进行更新。



实时学习A*智能体 (LRTA*)

实时学习A*算法 (LRTA*)

```
function LRTA*-COST(s,a,s' ,H) return an cost estimate
    if s' is undefined the return h(s)
    else return c(s,a,s' ) + H[s' ]
function LRTA*-AGENT(s' ) return an action
    input: s' , a percept identifying current state
    static: result, a table indexed by action and state, initially empty
           H, a table of cost estimates indexed by state, initially empty
           s,a, the previous state and action, initially null

    if GOAL-TEST(s' ) then return stop
    if s' is a new state (not in H) then H[s' ] ← h(s' )
    unless s is null
        result[a,s] ← s'
        H[s] ← MINb ∈ ACTIONS(s) LRTA*-COST(s,b,result[b,s],H)
    a ← an action b in ACTIONS(s' ) that minimizes LRTA*-COST(s' ,b,result[b,s' ],H)
    s ← s'
    return a
```



小结

- 局部搜索算法：爬山法、模拟退火算法，可用来求解连续空间上的问题。
- 遗传算法是一个保留大量状态种群的随机爬山法搜索，杂交和变异是算法的关键操作。
- 在线局部搜索：可以避免局部极小值。



作业

■ 习题4.1, 4.2

- 4.1 跟踪 A*搜索算法用直线距离启发式求解从 Lugoj 到 Bucharest 问题的过程。按顺序列出算法扩展的节点和每个节点的 f , g , h 值。
- 4.2 启发式路径算法是一个最佳优先搜索，它的目标函数是 $f(n) = (2 - w)g(n) + wh(n)$ 。算法中 w 取什么值能保证算法是最优的？当 $w = 0$ 时，这个算法是什么搜索？ $w = 1$ 呢？ $w = 2$ 呢？

