人工智能：高级搜索

饶洋辉
计算机学院，
中山大学
raoyangh@mail.sysu.edu.cn
http://cse.sysu.edu.cn/node/2471

课件来源：中山大学王甲海教授；海军工程大学贲可荣教授等

第4章 高级搜索

4.1 爬山法搜索

4.2 模拟退火搜索

4.3 遗传算法

爬山法搜索

- 搜索算法在内存中保留一条或多条路径并且记录哪些是已经探索过的，哪些是还没有探索过的。当找到目标时，到达目标的路径同时也构成了这个问题的一个解。
- 爬山法搜索：贪婪局部搜索
- 登高：一直向值增加的方向持续移动，将会在到达一个“峰顶”时终止，并且在相邻状态中没有比它更高的值。这个算法不维护搜索树，因此当前节点的数据结构只需要记录当前状态和它的目标函数值。

爬山法搜索

- 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。典型情况下，搜索的路径在局部搜索算法中是不保留的。
- 优点：

（1）它们只用很少的内存

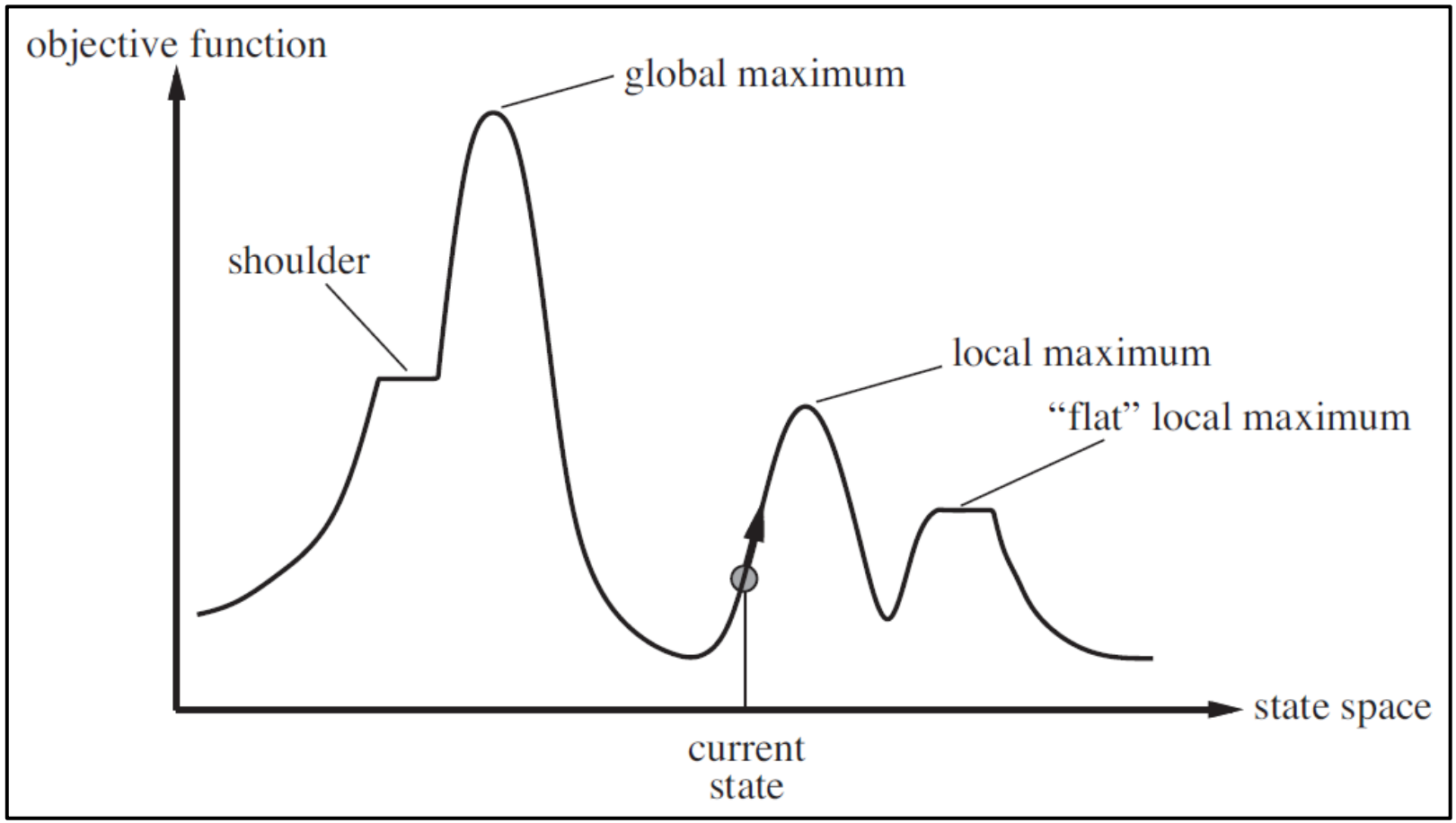
（2）它们通常能在很大状态空间中找到合理的解

除了找到目标，局部搜索算法对于解决纯粹的最优化问题是很有用的，其目标是根据一个目标函数找到最佳状态。

爬山法搜索

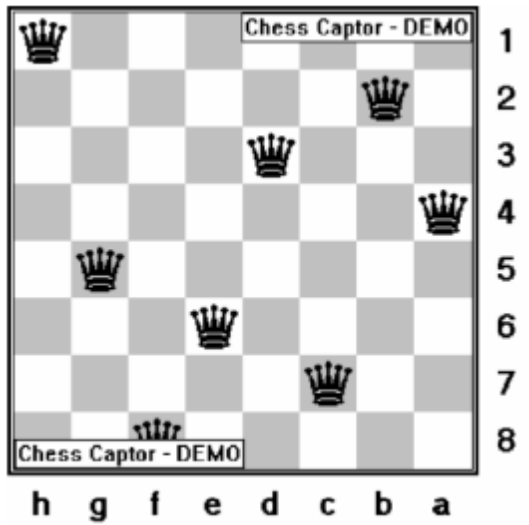
- 地形图既有“位置”（用状态定义），又有“高度”（由启发式耗散函数或目标函数的值定义）。
- 如果高度对应于耗散，那么目标是找到最低谷——即一个全局最小值；
- 如果高度对应于目标函数，那么目标是找到最高峰——即一个全局最大值。
- 局部搜索算法就像对地形图的探索，如果存在解，那么完备的局部搜索算法总能找到解；最优的局部搜索算法总能找到全局最小值／最大值。

爬山法搜索


objective function
global maximum
shoulder
local maximum
"flat" local maximum
state space
current
state

爬山法搜索

- 在集成电路设计、工厂场地布局、作业车间调度、自动程序设计、电信网络优化、车辆寻径等问题中，问题的解与到达目标的路径是无关的。
- 例如，在八皇后问题中，重要的是最终皇后的布局，而不是加入皇后的次序。



八皇后问题：将8个皇后放置在一个8×8的棋盘上，使得不存在任意两个皇后可以彼此攻击（同行、列、对角线）

爬山法搜索

- 局部搜索算法通常使用完全状态形式化，即每个状态都表示为在棋盘上放八个皇后，每列一个。
- 后继函数返回的是移动一个皇后到和它同一列的另一个方格中的所有可能的状态（因此每个状态有$8 \times 7 = 56$个后继）。
- 启发式耗散函数h是可以彼此攻击的皇后对的数量，不管中间是否有障碍。
- 该函数的全局最小值是0，仅在找到完美解时才能得到这个值。下图(a)显示了一个h=17的状态。图中还显示了它的所有后继的值，最好的后继是h=12。爬山法算法通常在最佳后继的集合中随机选择一个进行扩展（如果后继多于一个）。

爬山法搜索

图（a）

18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	Q	13	16	13	16
Q	14	17	15	Q	14	16	16
17	Q	16	18	15	Q	15	Q
18	14	Q	15	15	14	Q	16
14	14	13	17	12	14	12	18

（b）

						Q	
				Q			
	Q						
			Q				
					Q		
							Q
		Q					
Q							

- 爬山法能很快朝着解的方向进展。例如，从图（a）中的状态，它只需要五步就能到达图4-1（b）中的状态，它的h=1，这基本上很接近于解了。可是，爬山法经常会遇到下面的问题：

爬山法搜索

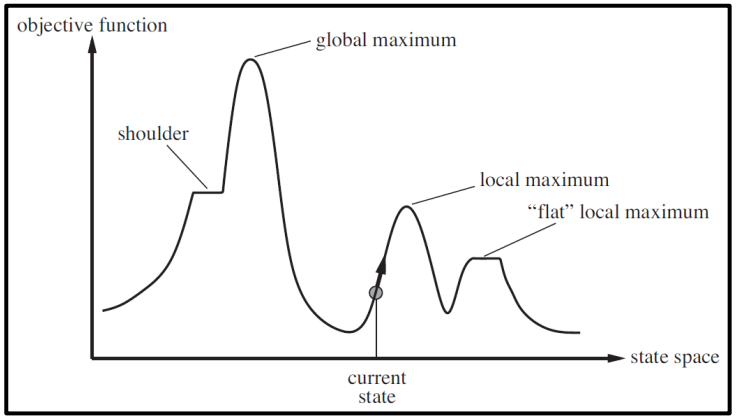
(1) 局部极大值：

➢局部极大值是一个比它的每个邻居状态都高的峰顶，但是比全局最大值要低。
➢爬山法算法到达局部极大值附近就会被拉向峰顶，然后被卡在局部极大值处无处可走。
➢更具体地，图（b）中的状态事实上是一个局部极大值（即耗散h的局部极小值）；不管移动哪个皇后得到的情况都会比原来差。

(2) 山脊：山脊造成的是一系列的局部极大值，贪婪算法处理这种情况是很难的。

(3) 高原：

高原是在状态空间地形图上评价函数值平坦的一块区域。它可能是一块平的局部极大值，不存在上山的出路，或者是一个山肩，从山肩还有可能取得进展。爬山法搜索可能无法找到离开高原的道路。



爬山法搜索

- 在各种情况下，爬山法算法都会达到无法取得进展的状态。
- 从一个随机生成的八皇后问题的状态开始，最陡上升的爬山法86％的情况下会被卡住，只有14%的问题实例能求解。
- 这个算法速度很快，成功找到最优解的平均步数是4步，被卡住的平均步数是3步——对于包含8^8=16, 777, 216个状态的状态空间，这已经是不错的结果了。

爬山法搜索

- 前述算法中，如果到达一个高原，最佳后继的状态值和当前状态值相等时将会停止。
- 如果高原其实是山肩，继续前进——即侧向移动通常是一种好方法。
- 注意，如果在没有上山移动的情况下总是允许侧向移动，那么当到达一个平坦的局部极大值而不是山肩的时候，算法会陷入无限循环。一种常规的解决办法是设置允许连续侧向移动的次数限制。
- 如，在八皇后问题中允许最多连续侧向移动100次。这使问题实例的解决率从14%提高到了94%。
- 代价：算法对于每个成功搜索实例的平均步数为大约21步，每个失败实例的平均步数为大约64步。

模拟退火算法

- 模拟退火算法思想
- 模拟退火基本流程
- 模拟退火基本要素与设置

模拟退火算法思想

- 模拟退火算法概述
  - 模拟退火算法（Simulated Annealing，SA）是一种模拟物理退火的过程而设计的优化算法。它的基本思想最早在1953年就被Metropolis提出，但直到1983年Kirkpatrick等人才设计出真正意义上的模拟退火算法并进行应用。
- 模拟退火算法思想
  - 模拟退火算法采用类似于物理退火的过程，先在一个高温状态下（相当于算法随机搜索，大概率接受劣解），然后逐渐退火，徐徐冷却（接受劣解概率变小直至为零，相当于算法局部搜索），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（劣向转移是脱出局部极小的核心机制）。

模拟退火算法思想

物理退火过程

物体内部的状态

状态的能量

温度

熔解过程

退火冷却过程

状态的转移

能量最低状态

模拟退火算法

问题的解空间

解的质量

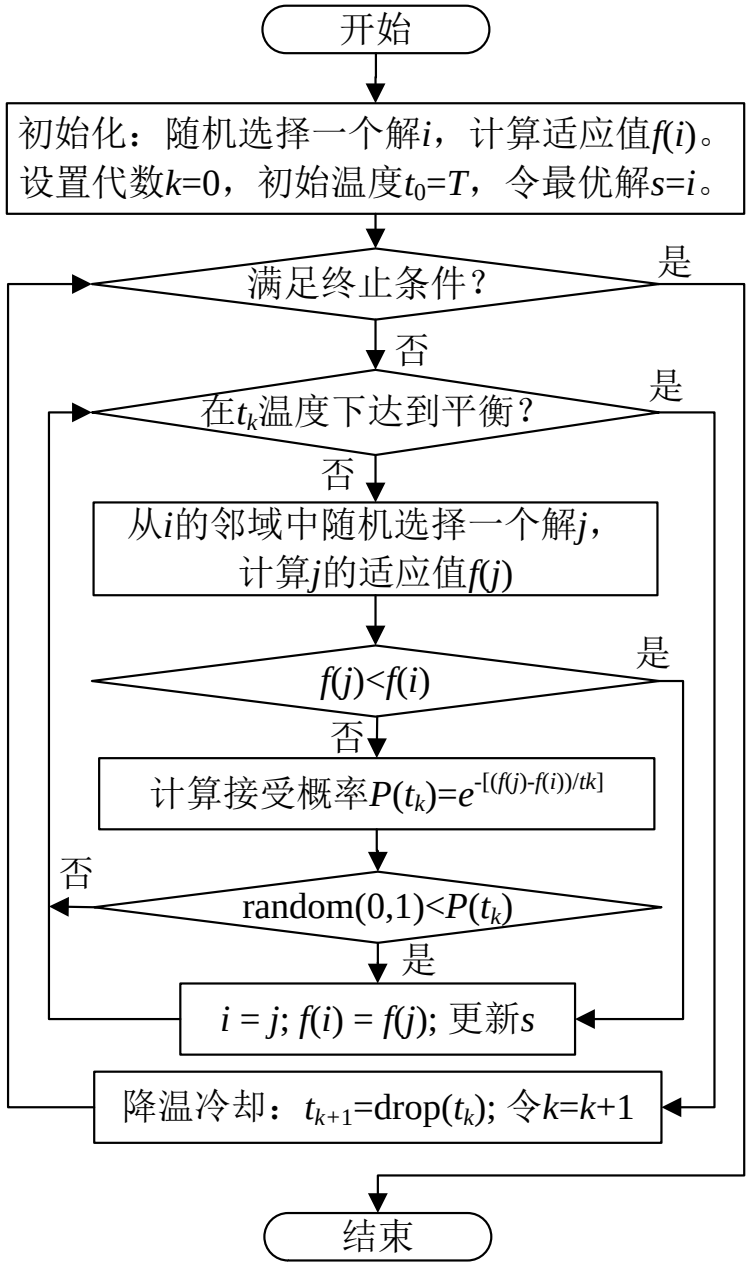
控制参数

设定初始温度

控制参数的修改

解在邻域中的变化

最优解




```
//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

- 从算法流程上看，模拟退火算法包括三函数两准则，即状态产生函数、状态接受函数、温度更新函数、内循环终止准则和外循环终止准则，这些环节的设计将决定SA算法的优化性能。此外，初温的选择对SA算法性能也有很大影响。

- 理论上，SA算法的参数只有满足算法的收敛条件，才能保证实现的算法依概率1收敛到全局最优解。然而，SA算法的某些收敛条件无法严格实现，即使某些收敛条件可以实现，但也常常会因为实际应用的效果不理想而不被采用。因此，至今SA算法的参数选择依然是一个难题，通常只能依据一定的启发式准则或大量的实验加以选取。

模拟退火基本要素与设置

- 状态产生函数
  - 设计状态产生函数（邻域函数）的出发点应该是尽可能保证产生的候选解遍布全部解空间。通常，状态产生函数由两部分组成，即产生候选解的方式和候选解产生的概率分布。
  - 前者决定由当前解产生候选解的方式，后者决定在当前解产生的候选解中选择不同状态的概率。
  - 候选解的产生方式由问题的性质决定，通常在当前状态的邻域结构内以一定概率方式产生，而邻域函数和概率方式可以多样化设计，其中概率分布可以是均匀分布、正态分布、指数分布、柯西分布等。

模拟退火基本要素与设置

- 状态接受函数
  - 状态接受函数一般以概率的方式给出，不同接受函数的差别主要在于接受概率的形式不同。设计状态接受概率，应该遵循以下原则：
    - 在固定温度下，接受使目标函数值下降的候选解的概率要大于使目标函数值上升的候选解的概率；
    - 随温度的下降，接受使目标函数值上升的解的概率要逐渐减小；
    - 当温度趋于零时，只能接受目标函数值下降的解。

模拟退火基本要素与设置

- 状态接受函数
  - 状态接受函数的引入是SA算法实现全局搜索的最关键的因素，但实验表明，状态接受函数的具体形式对算法性能的影响不显著。因此，SA算法中通常采用$\min[1,\exp(-\Delta C/t)]$作为状态接受函数。

  - 在固定温度下，接受使目标函数值下降的候选解的概率要大于使目标函数值上升的候选解的概率；
  - 随温度的下降，接受使目标函数值上升的解的概率要逐渐减小；
  - 当温度趋于零时，只能接受目标函数值下降的解。

模拟退火基本要素与设置

- 初温
  - 初始温度t_0、温度更新函数、内循环终止准则和外循环终止准则通常被称为退火历程（annealing schedule）。
  - 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。因此，初温的确定应折衷考虑优化质量和优化效率。
- 温度更新函数
  - 温度更新函数即温度的下降方式，用于在外循环中修改温度值。
  - 在非时齐SA算法收敛性理论中，更新函数可采用指数函数。

模拟退火基本要素与设置

- 内循环终止准则
  - 内循环终止准则，或称Metropolis抽样稳定准则，用于决定在各温度下产生候选解的数目。在非时齐SA算法理论中，由于在每个温度下只产生一个或少量候选解，所以不存在选择内循环终止准则的问题。而在时齐SA算法理论中，收敛性条件要求在每个温度下产生候选解数目趋于无穷大，以使相应的马氏链达到平稳概率分布，显然在实际应用算法时这是无法实现的。
  - 常用的抽样稳定准则包括:
    - 检验目标函数的均值是否稳定;
    - 连续若干步的目标值变化较小;
    - 按一定的步数抽样。

模拟退火基本要素与设置

- 外循环终止准则
  - 外循环终止准则，即算法终止准则，用于决定算法何时结束。设置温度终值t_e是一种简单的方法。SA算法的收敛性理论中要求t_e趋于零，这显然是不实际的。通常的做法包括：
    - 设置终止温度的阈值；
    - 设置外循环迭代次数（内循环*外循环=算法总循环，给定总循环数，合理分配内外循环，如10*1000=10000）；
    - 算法搜索到的最优值连续若干步保持不变。

遗传算法

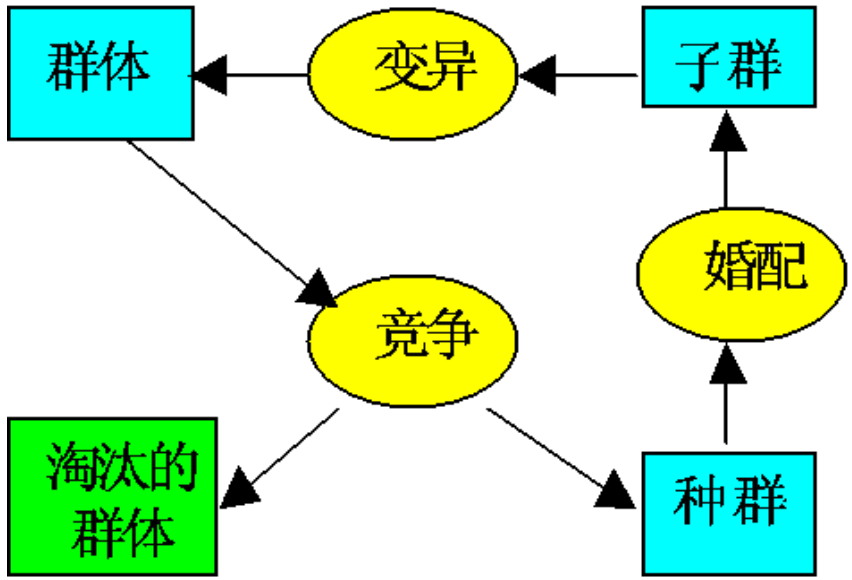
- 进化算法的产生和发展
- 遗传算法的基本思想
- 遗传算法的一般步骤

进化算法的概念

- 进化算法（Evolutionary Algorithm，EA）是基于自然选择和自然遗传等生物进化机制的一种搜索算法。
- 生物进化是通过繁殖、变异、竞争和选择实现的；而进化算法则主要通过选择、交叉（或重组）和变异这三种操作实现优化问题的求解。
- 进化算法是一个“算法簇”，包括遗传算法、遗传规划、进化策略和进化规划等。
- 进化算法的基本框架是遗传算法所描述的框架。
- 进化算法可广泛应用于组合优化、机器学习、自适应控制、规划设计和人工生命等领域。

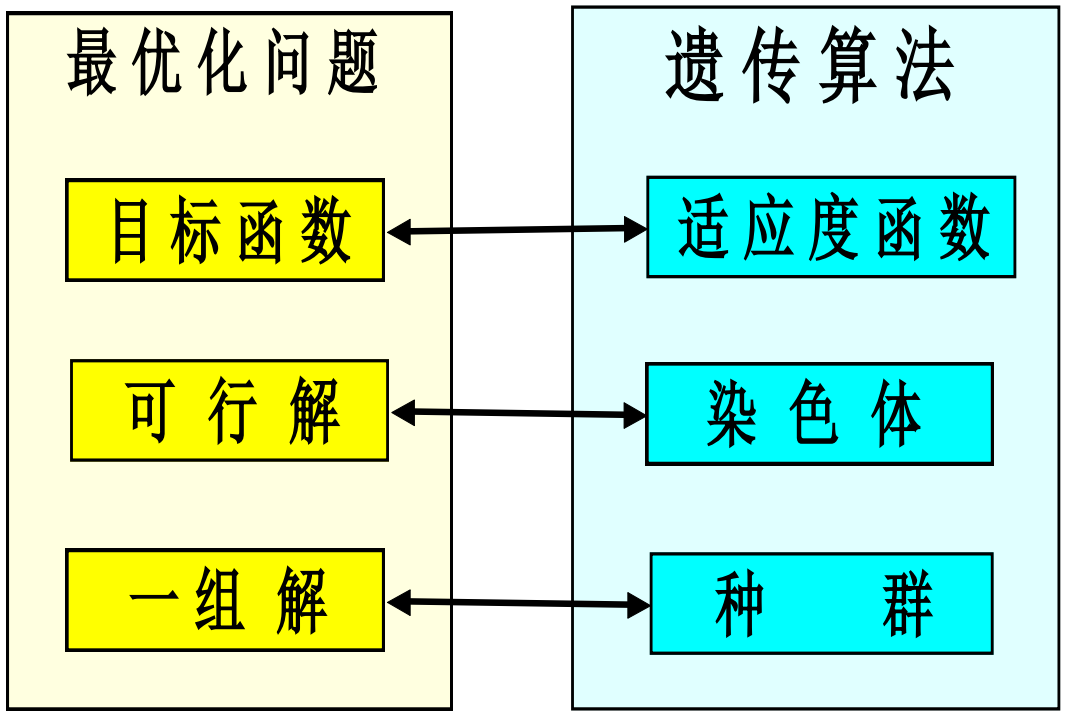
进化算法的生物学背景

- 适者生存：最适合自然环境的群体往往产生了更大的后代群体。
- 生物进化的基本过程：



- 遗传算法（Genetic Algorithm，GA）：一类借鉴生物界自然选择和自然遗传机制的随机搜索算法，非常适用于处理传统搜索方法难以解决的复杂和非线性优化问题。

遗传算法的基本思想



- 遗传算法的基本思想：在求解问题时从多个解开始，然后通过一定的法则进行逐步迭代以产生新的解。

遗传算法的基本思想

- 初代种群产生之后，按照适者生存和优胜劣汰的原理，逐代（generation）演化产生出越来越好的近似解。
- 在每一代，根据问题域中个体的适应度大小挑选个体，并借助于自然遗传学的遗传算子进行组合交叉和变异，产生出代表新的解集的种群。
- 这个过程将导致种群像自然进化一样的后代种群比前代更加适应于环境，末代种群中的最优个体经过解码，可以作为问题近似最优解。
- 遗传算法采纳了自然进化模型，如选择、交叉、变异、迁移、局域与邻域等。

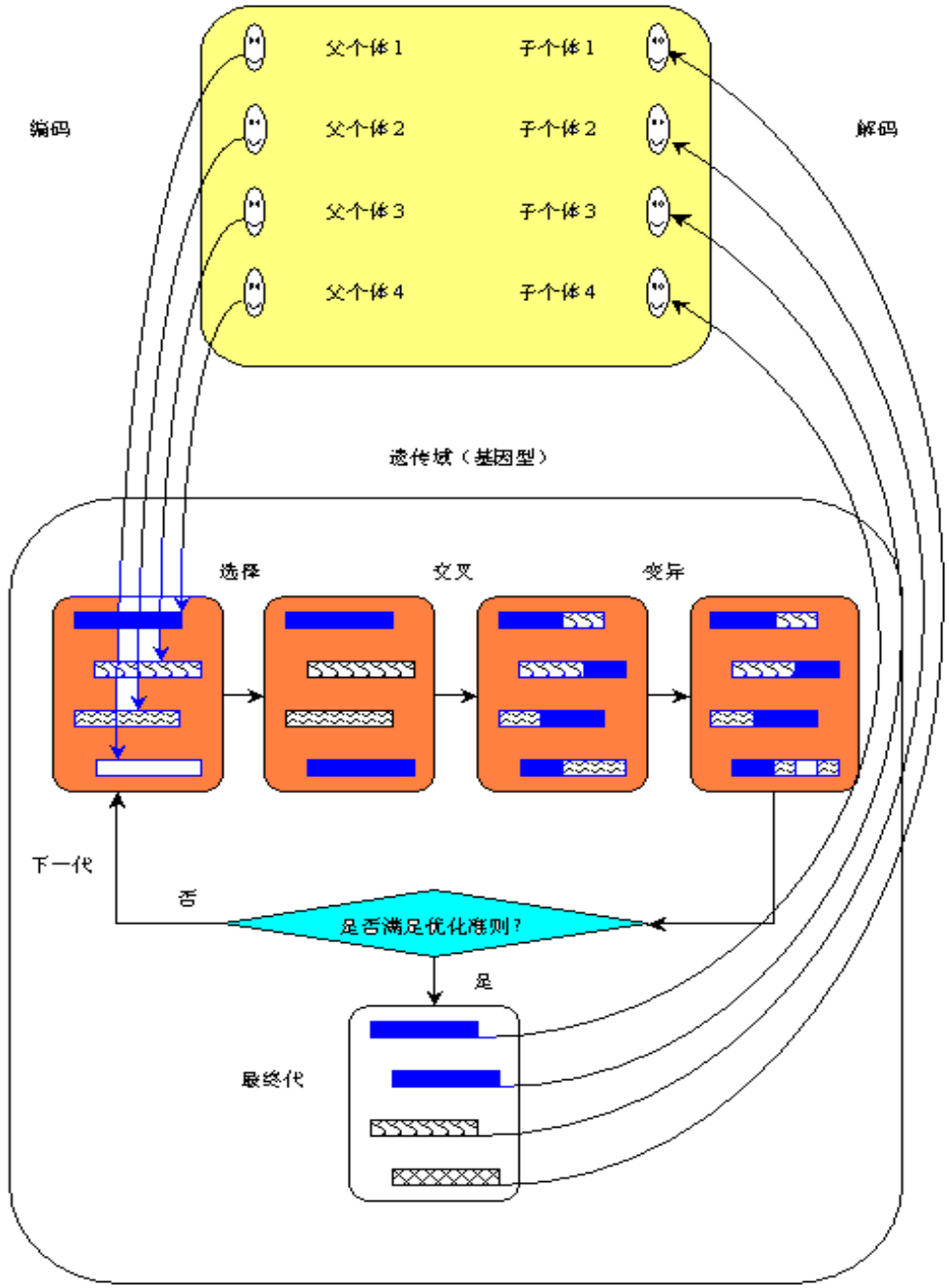
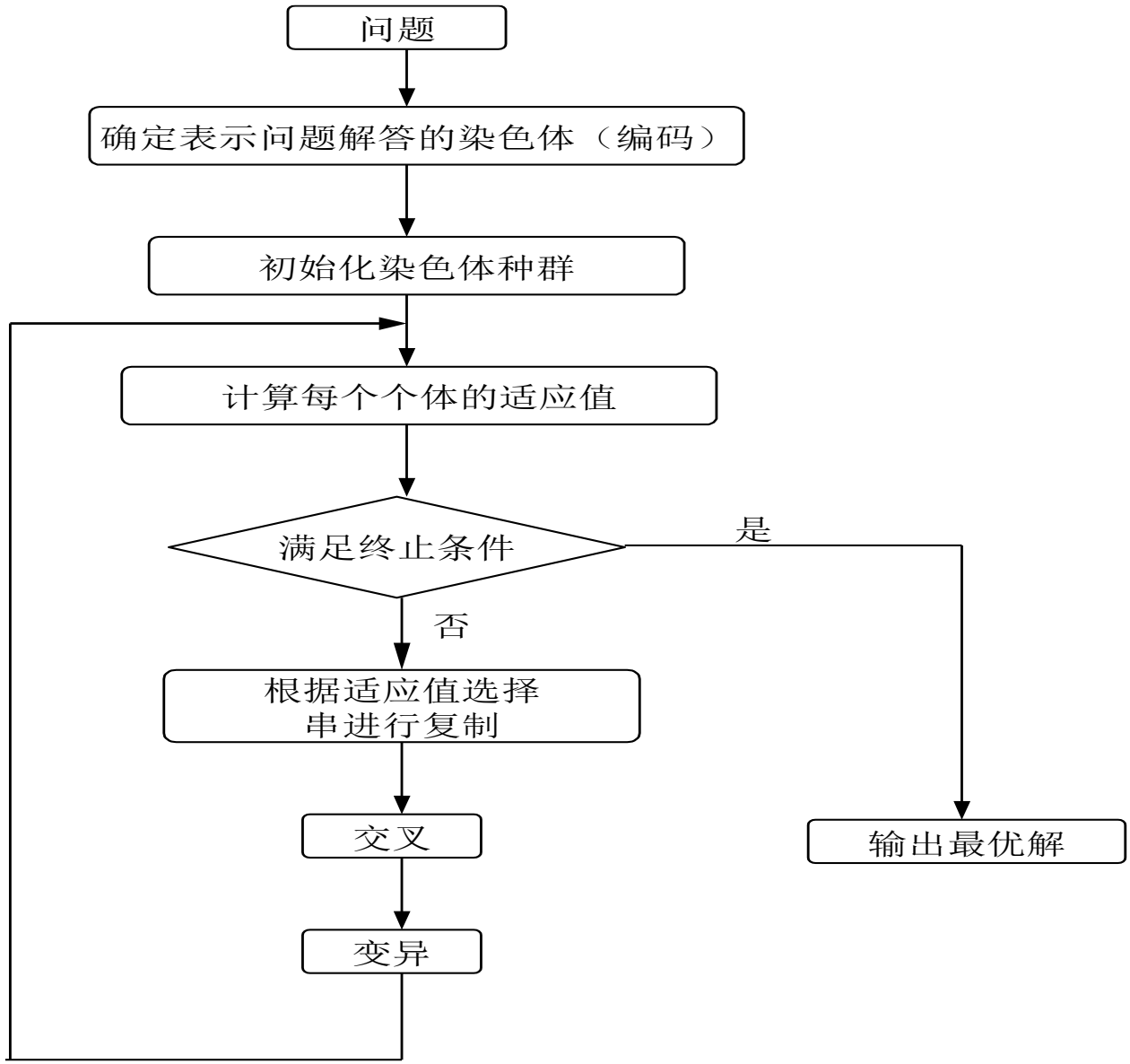


图　遗传算法的过程

- 开始时，种群随机初始化，并计算每个个体的适应度函数，初始代产生了。
- 如果不满足优化准则，开始产生新一代的计算。
- 为了产生下一代，按照适应度选择个体，父代进行基因重组（交叉）而产生子代。
- 所有的子代按一定概率变异。然后子代的适应度又被重新计算，子代被插入到种群中将父代取而代之，构成新的一代。
- 循环，直到满足优化准则。

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

编码

- 位串编码
  - 一维染色体编码方法：将问题空间的参数编码为一维排列的染色体的方法。
  - 例如，用若干二进制数表示一个个体，将原问题的解空间映射到位串空间 B = {0，1}上，然后在位串空间上进行遗传操作。
  - 上述二进制编码的优点：类似于生物染色体的组成，算法易于用生物遗传理论解释，遗传操作如交叉、变异等易实现。
  - 缺点：（1）相邻整数的二进制编码可能具有较大的Hamming距离，降低了遗传算子的搜索效率；（2）要先给出求解的精度；（3）求解高维优化问题的二进制编码串长，搜索效率低。

15：01111　　16：10000

编码

- 位串编码
  - 多参数映射编码的基本思想：把每个参数先进行二进制编码得到子串，再把这些子串连成一个完整的染色体。
  - 多参数映射编码中的每个子串对应各自的编码参数，所以可以有不同的串长度和参数的取值范围。
- 实数编码
  - 采用实数表达法不必进行数制转换。

群体设定

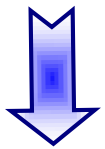
- 种群规模的确定
  - 规模太小，遗传算法的优化性能不太好，易陷入局部最优解。
  - 规模太大，计算复杂。
- 初始种群的产生
  - 随机产生种群规模数目的个体作为初始种群。
  - 随机产生一定数目的个体，从中挑选最好的个体加到初始种群中。不断迭代，直到初始群体中个体数目达到了预先确定的规模。
  - 根据问题固有知识，把握最优解所占空间在整个问题空间中的分布范围，然后，在此分布范围内设定初始种群。

适应度函数

1. 将目标函数映射成适应度函数的方法

若目标函数为最大化问题，则 $Fit(f(x)) = f(x)$

若目标函数为最小化问题，则 $Fit(f(x)) = \dfrac{1}{f(x)}$



将目标函数转换为求最大值的形式，且保证函数值非负！

若目标函数为最大化问题，则

$$Fit(f(x)) = \begin{cases} f(x) - C_{\min} & f(x) > C_{\min} \\ 0 & \text{其他情况} \end{cases}$$

若目标函数为最小化问题，则

$$Fit(f(x)) = \begin{cases} C_{\max} - f(x) & f(x) < C_{\max} \\ 0 & \text{其他情况} \end{cases}$$

适应度函数

2. 适应度函数的尺度变换

- 在遗传算法中，将所有妨碍适应度值高的个体产生，从而影响遗传算法正常工作的问题统称为欺骗问题。
- 过早收敛：缩小这些个体的适应度，以降低这些超级个体的竞争力。
- 停滞现象：改变原始适应值的比例关系，以提高个体之间的竞争力。
- 尺度变换或定标：对适应度函数值域的某种映射变换。

选择

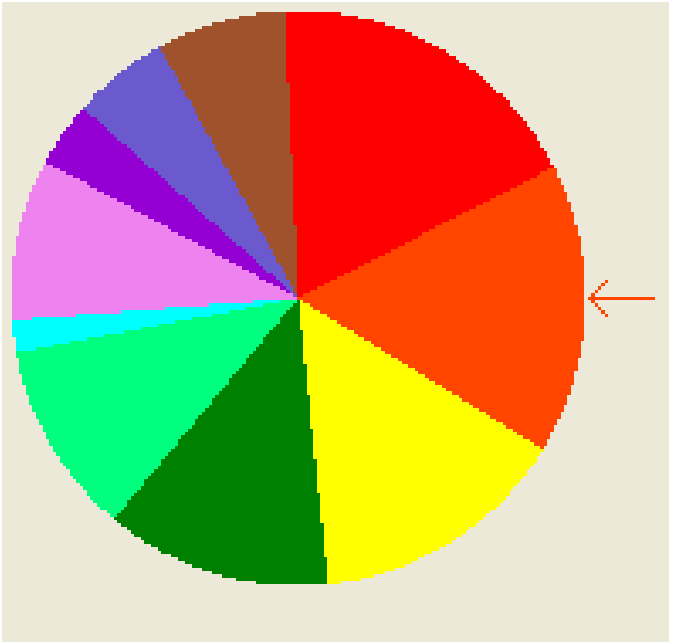
- 选择是用来确定重组或交叉个体，以及被选个体将产生多少个子代个体。首先计算适应度：
  (1) 按比例的适应度计算；
  (2) 基于排序的适应度计算。
- 适应度计算之后是实际的选择，按照适应度进行父代个体的选择。

- 可以挑选以下的算法：
  ① 轮盘赌选择；　② 锦标赛选择；
  ③ 最佳个体保存；　④ 随机遍历抽样。

选择

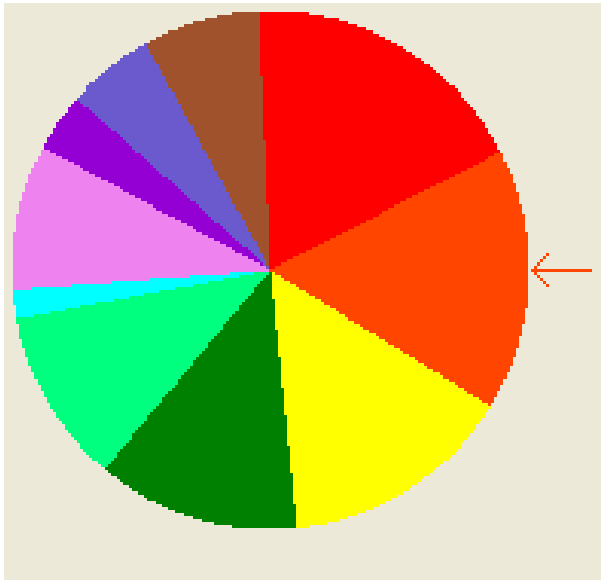
(1) 轮盘赌选择

- 按个体的选择概率产生一个轮盘，轮盘每个区的角度与个体的选择概率成比例。
- 产生一个随机数，它落入轮盘的哪个区域就选择相应的个体交叉。



选择

(1) 轮盘赌选择



个体	1	2	3	4	5	6	7	8	9	10	11
适应度	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.6	0.4	0.2	0.1
选择概率	0.18	0.16	0.15	0.13	0.11	0.09	0.07	0.06	0.03	0.02	0.0
累积概率	0.18	0.34	0.49	0.62	0.73	0.82	0.89	0.95	0.98	1.00	1.00

第1轮产生一个随机数：**0.81**　　第2轮产生一个随机数：**0.32**

选择

(2) 锦标赛选择 (tournament selection model)

- 锦标赛选择方法：从群体中随机选择个个体，将其中适应度最高的个体保存到下一代。这一过程反复执行，直到保存到下一代的个体数达到预先设定的数量为止。

- 随机竞争方法（stochastic tournament）：每次按轮盘赌选择方法选取一对个体，然后让这两个个体进行竞争，适应度高者获胜。如此反复，直到选满为止。

选择

(3) 最佳个体保存

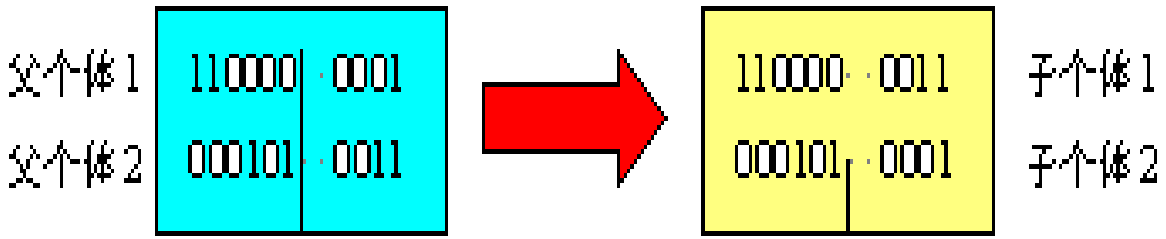
- 最佳个体（elitist model）保存方法：把群体中适应度最高的个体不进行交叉而直接复制到下一代中，保证遗传算法终止时得到的最后结果一定是历代出现过的最高适应度的个体。

交叉

1. 基本的交叉算子

（1）单点交叉（single-point crossover）

- 在个体串中随机设定一个交叉点，实行交叉时，该点前或后的两个个体的部分结构进行互换，并生成两个新的个体。



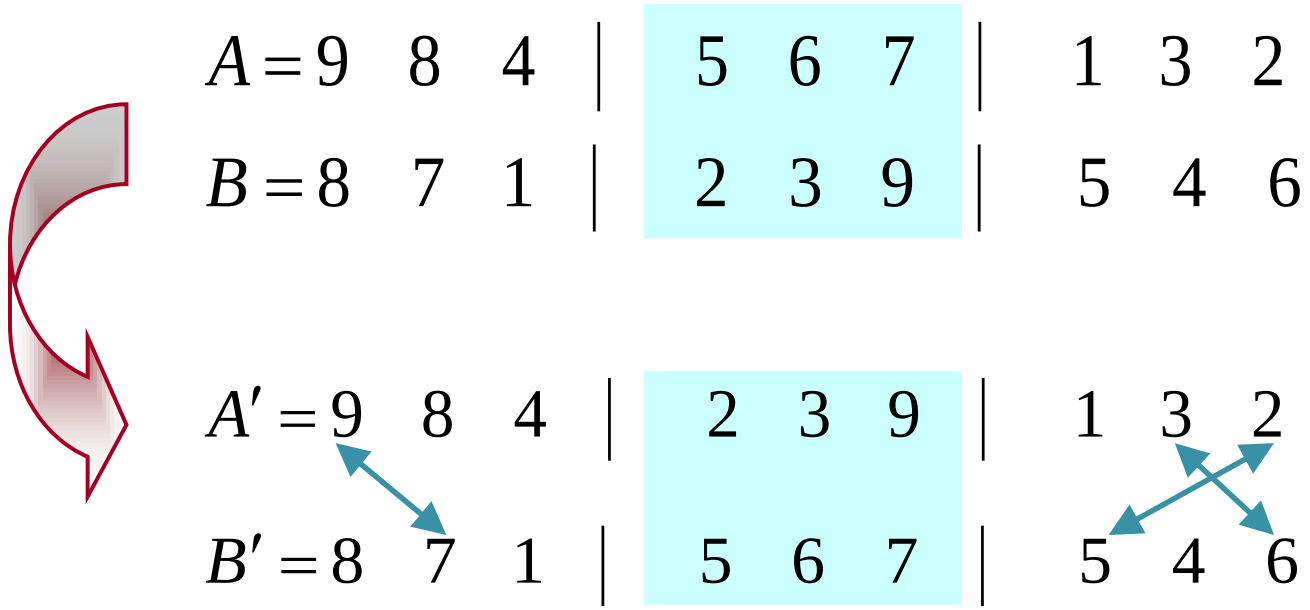
（2）两点交叉 （two-point crossover）

- 随机设置两个交叉点，将两个交叉点之间的码串相互交换。

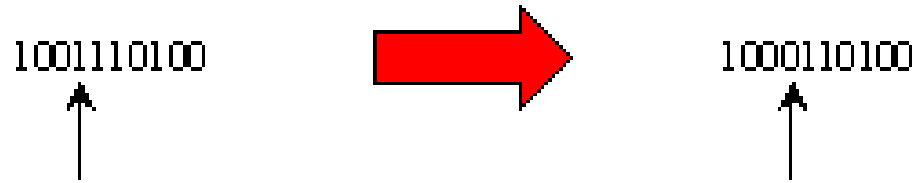
交叉

2. 修正的交叉方法

部分匹配交叉PMX：Goldberg D. E.和R.Lingle (1985)



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，向左或向右移动一个随机位数。

遗传算法的主要特点

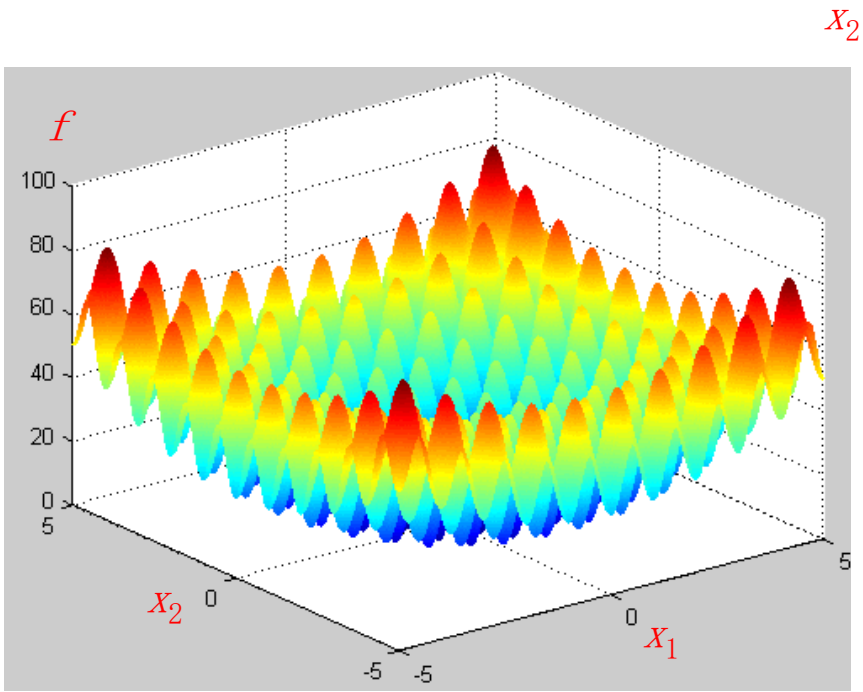
遗传算法是一种全局优化概率算法，主要特点有：

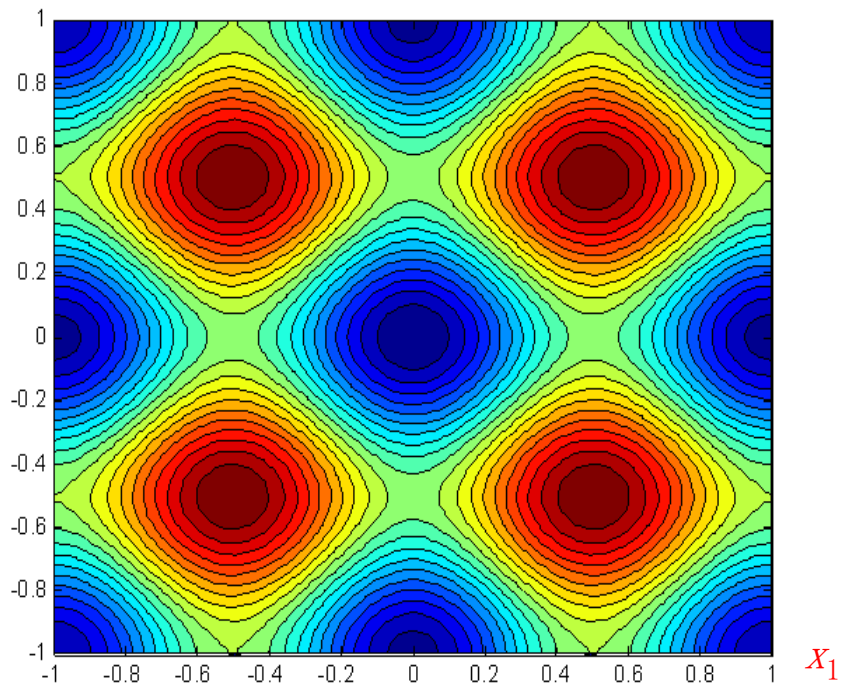
- 遗传算法对所求解的优化问题没有太多的数学要求，由于进化特性，搜素过程中不需要问题的内在性质，可直接对结构对象进行操作。
- 利用随机技术指导对一个被编码的参数空间进行高效率搜索。
- 采用群体搜索策略，易于并行化。
- 仅用适应度函数值来评估个体，并在此基础上进行遗传操作，使种群中个体之间进行信息交换。
- 遗传算法能够非常有效地进行概率意义的全局搜索。

- 例： 用遗传算法求解下面一个Rastrigin函数的最小值。

$$f(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2)$$

$$-5 \le x_i \le 5 \qquad i = 1, 2$$



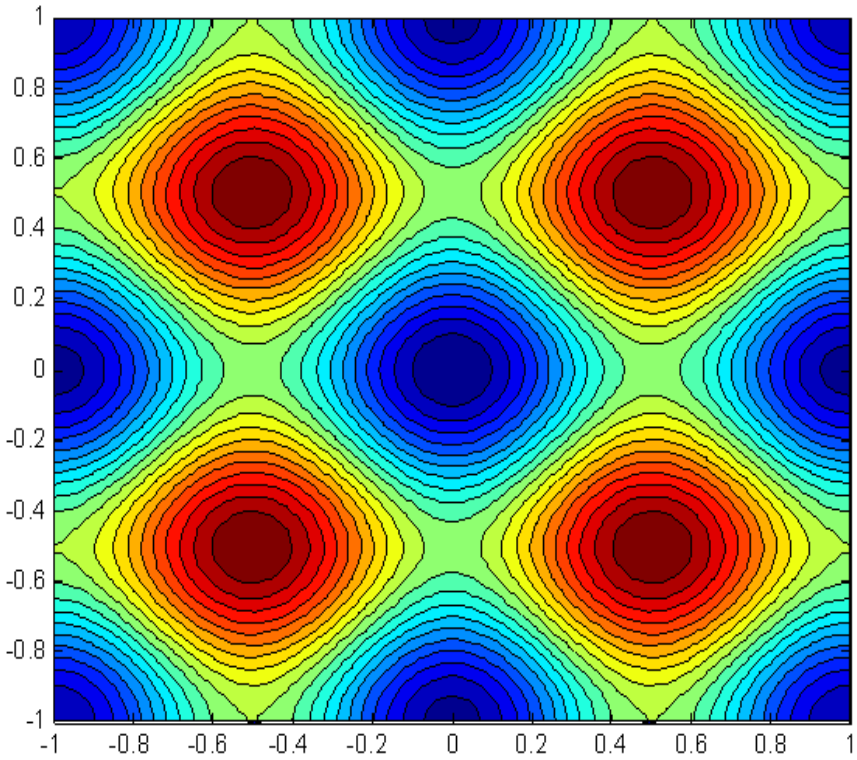


- 例： 用遗传算法求解下面一个Rastrigin函数的最小值。

$$f(x_1, x_2) = 20 + x_1^2 + x_2^{\ 2} - 10(\cos 2\pi x_1 + \cos 2\pi x_2)$$

$$-5 \le x_i \le 5 \qquad i = 1, 2$$

➢ **初始种群：**

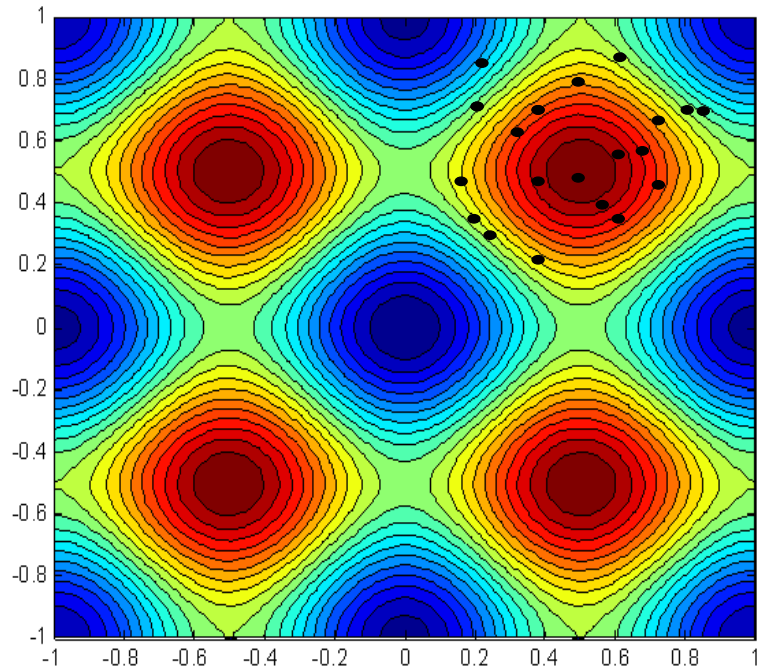


- 例： 用遗传算法求解下面一个Rastrigin函数的最小值。

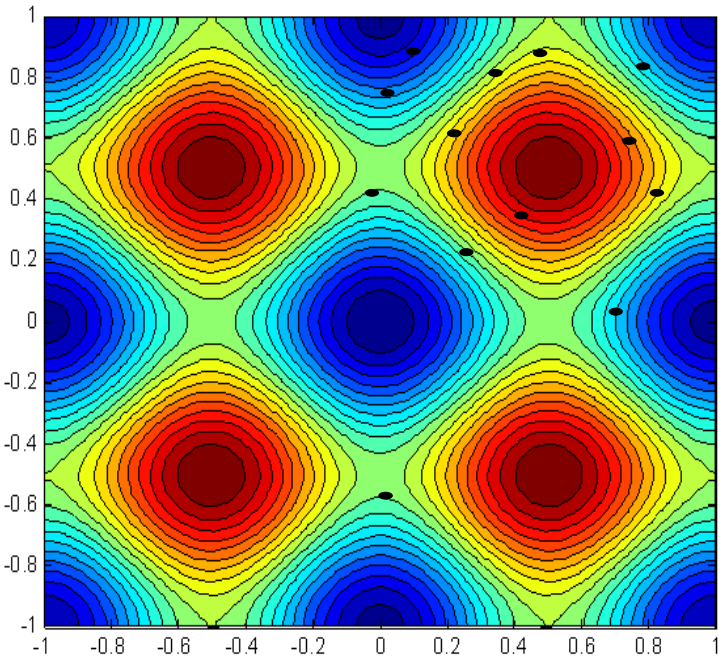
$$f(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2)$$

$$-5 \le x_i \le 5 \qquad i = 1, 2$$

➢ **初始种群**



第二代种群



- 在迭代60、80、95、100次时的种群