第 6 章　智能计算及其应用

教材：

王万良《人工智能导论》（第5版）

高等教育出版社，2020

第6章 智能计算及其应用

- 受自然界和生物界规律的启迪，人们根据其原理模仿设计了许多求解问题的算法，包括人工神经网络、模糊逻辑、遗传算法、DNA计算、模拟退火算法、禁忌搜索算法、免疫算法、膜计算、量子计算、粒子群优化算法、蚁群算法、人工蜂群算法、人工鱼群算法以及细菌群体优化算法等，这些算法称为智能计算也称为计算智能(computational intelligence, CI)。

第6章 智能计算及其应用

- 智能优化方法通常包括进化计算和群智能等两大类方法，是一种典型的元启发式随机优化方法，已经广泛应用于组合优化、机器学习、智能控制、模式识别、规划设计、网络安全等领域，是21世纪有关智能计算中的重要技术之一。
- 本章首先简要介绍进化算法的概念，详细介绍基本遗传算法，这是进化算法的基本框架。然后介绍双倍体、双种群、自适应等比较典型的改进遗传算法及其应用。

第6章 智能计算及其应用

- 6.1 进化算法的产生与发展
- 6.2 基本遗传算法
- 6.3 遗传算法的改进算法
- 6.4 遗传算法的应用

第6章 智能计算及其应用

- ✓ **6.1 进化算法的产生与发展**
- □ **6.2 基本遗传算法**
- □ **6.3 遗传算法的改进算法**
- □ **6.4 遗传算法的应用**

6.1 进化算法的产生与发展

- **6.1.1 进化算法的概念**
- **6.1.2 进化算法的生物学背景**
- **6.1.3 进化算法的设计原则**

6.1.1 进化算法的概念

- **进化算法**(evolutionary algorithms，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、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(GA)、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。

6.1.2 进化算法的生物学背景

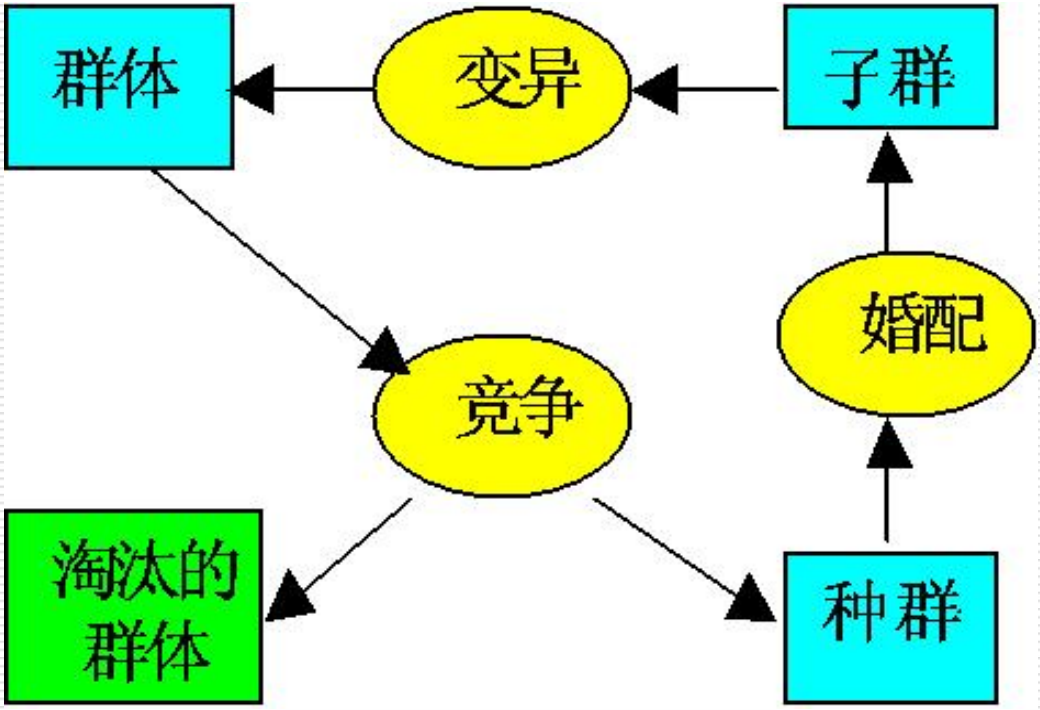
- 适者生存：最适合自然环境的群体往往产生了更大的后代群体。
- 生物进化的基本过程：

染色体(chromosome)：生物的遗传物质的主要载体。

基因(gene)：扩展生物性状的遗传物质的功能单元和结构单位。

基因座（locus）：染色体中基因的位置。

等位基因（alleles）：基因所取的值。



6.1.3 进化算法的设计原则

- （1）适用性原则：一个算法的适用性是指该算法所能适用的问题种类，它取决于算法所需的限制与假定。
- （2）可靠性原则：算法的可靠性是指算法对于所设计的问题，以适当的精度求解其中大多数问题的能力。
- （3）收敛性原则： 指算法能否收敛到全局最优。在收敛的前提下，希望算法具有较快的收敛速度。
- （4）稳定性原则： 指算法对其控制参数及问题的数据的敏感度。
- （5）生物类比原则：生物界有效方法及操作可以通过类比的方法引入到算法中，有时会带来较好的结果。

- 6.1 进化算法的产生与发展
- 6.2 基本遗传算法
- 6.3 遗传算法的改进算法
- 6.4 遗传算法的应用

6.2 基本遗传算法

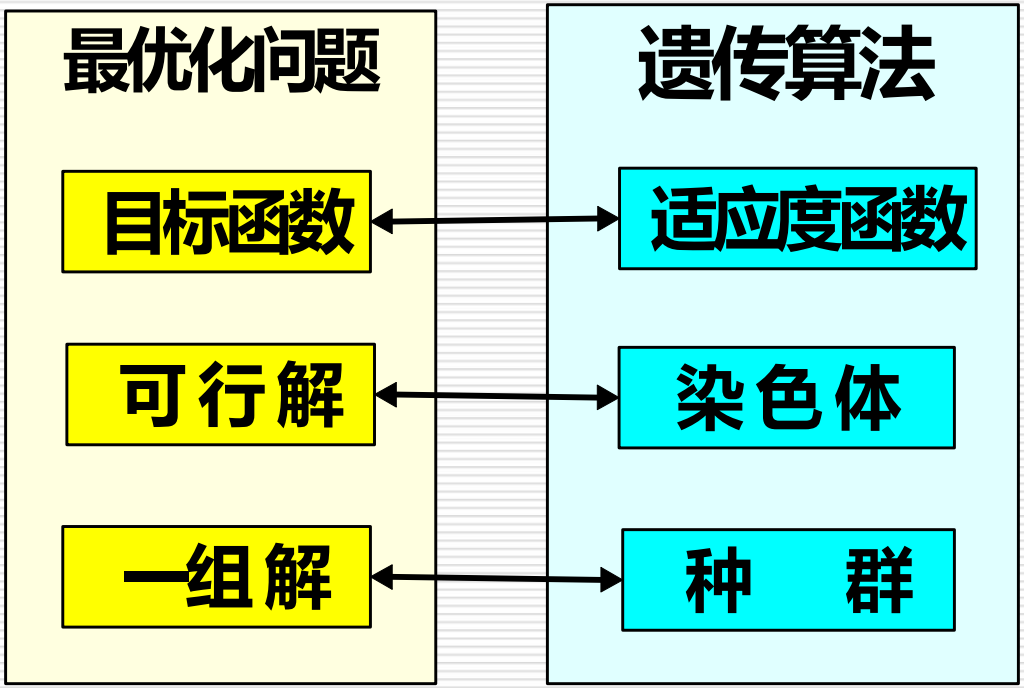
- **遗传算法**（genetic algorithms，GA）：一类借鉴生物界自然选择和自然遗传机制的随机搜索算法,非常适用于处理传统搜索方法难以解决的复杂和非线性优化问题。

- 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。

6.2.1 遗传算法的基本思想

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

6.2.1 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。

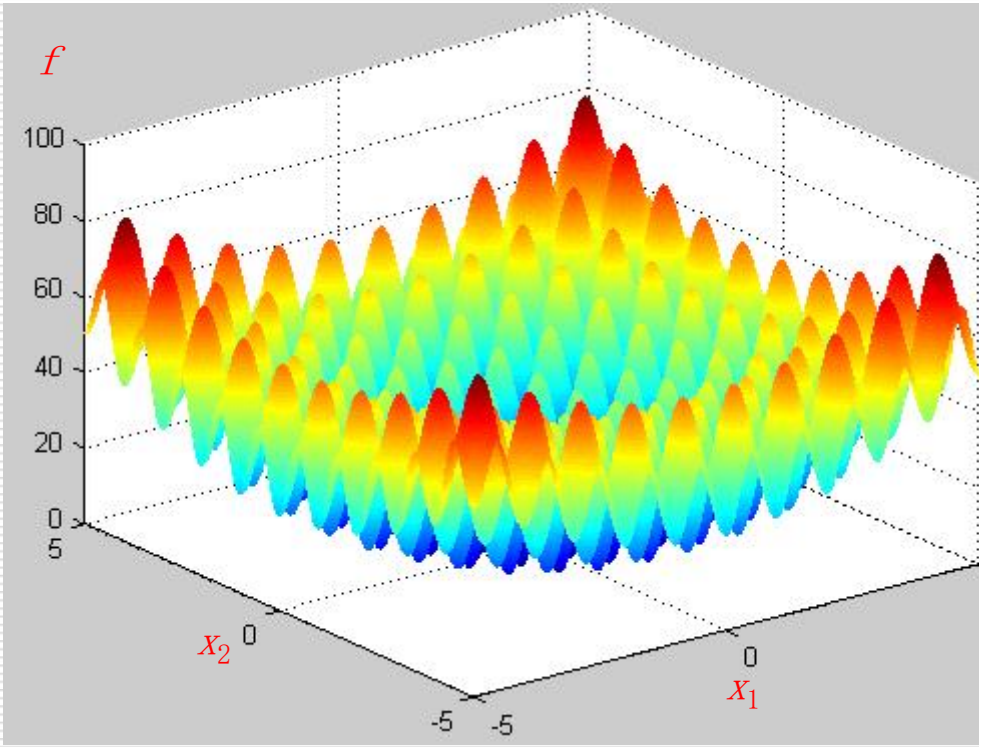
6.2.2 遗传算法的发展历史

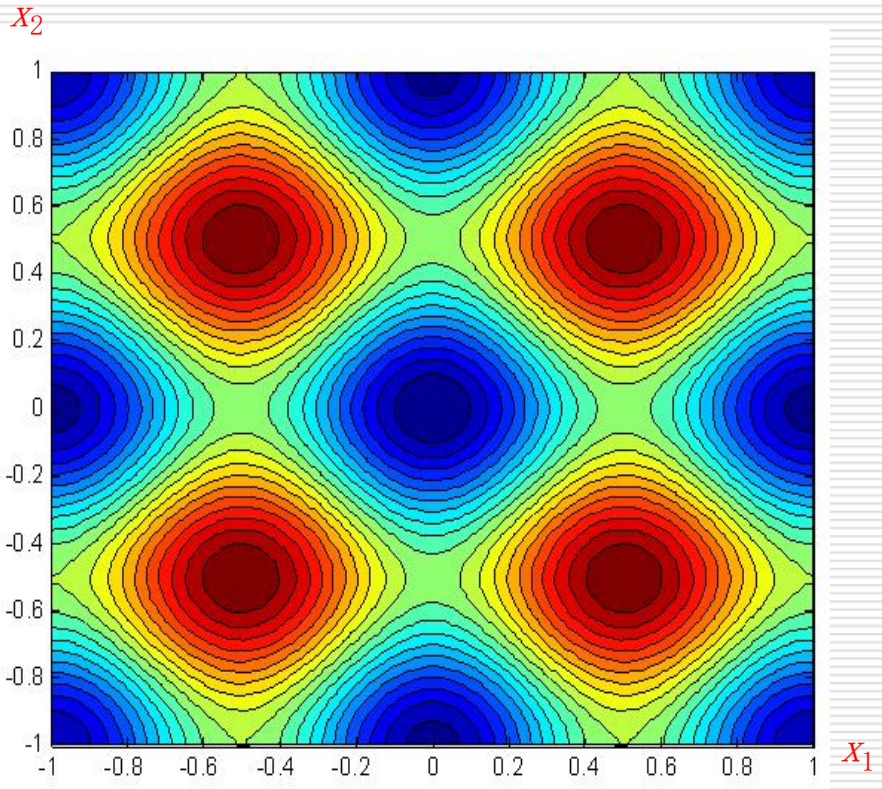
- 1962年，Fraser提出了自然遗传算法。
- 1965年，Holland首次提出了人工遗传操作的重要性。
- 1967年，Bagley首次提出了遗传算法这一术语。
- 1970年，Cavicchio把遗传算法应用于模式识别中。
- 1971年，Hollstien在论文《计算机控制系统中人工遗传自适应方法》中阐述了遗传算法用于数字反馈控制的方法。
- 1975年，美国J. Holland出版了《自然系统和人工系统的适配》；DeJong完成了重要论文《遗传自适应系统的行为分析》。
- 20世纪80年代以后，遗传算法进入兴盛发展时期。

- 例： 用遗传算法求解下面一个Rastrigin函数的最小值。

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

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



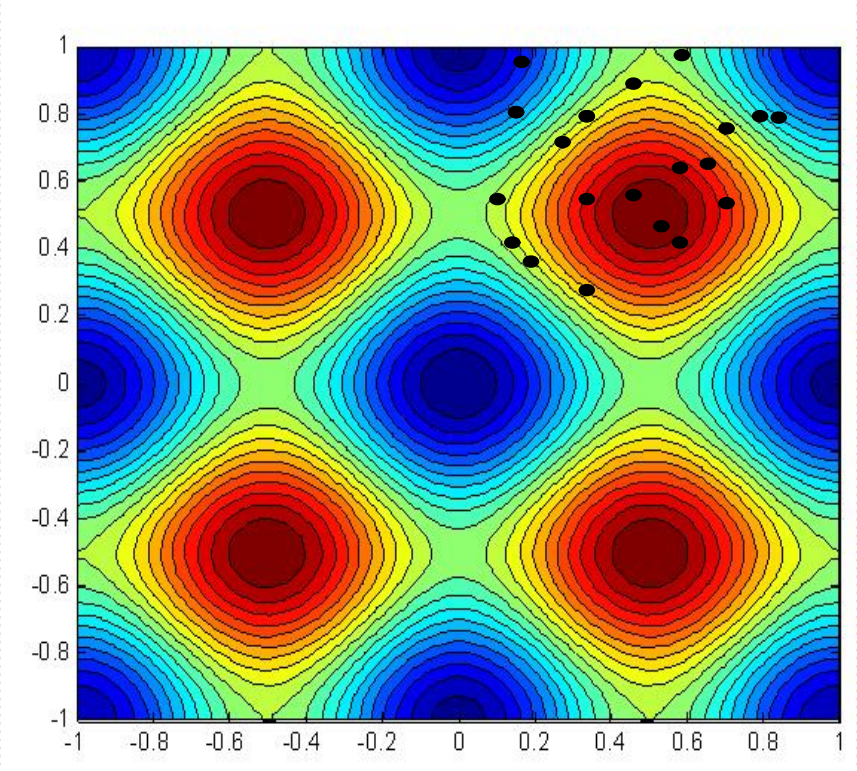


- 例： 用遗传算法求解下面一个Rastrigin函数的最小值。

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

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

- 初始种群：

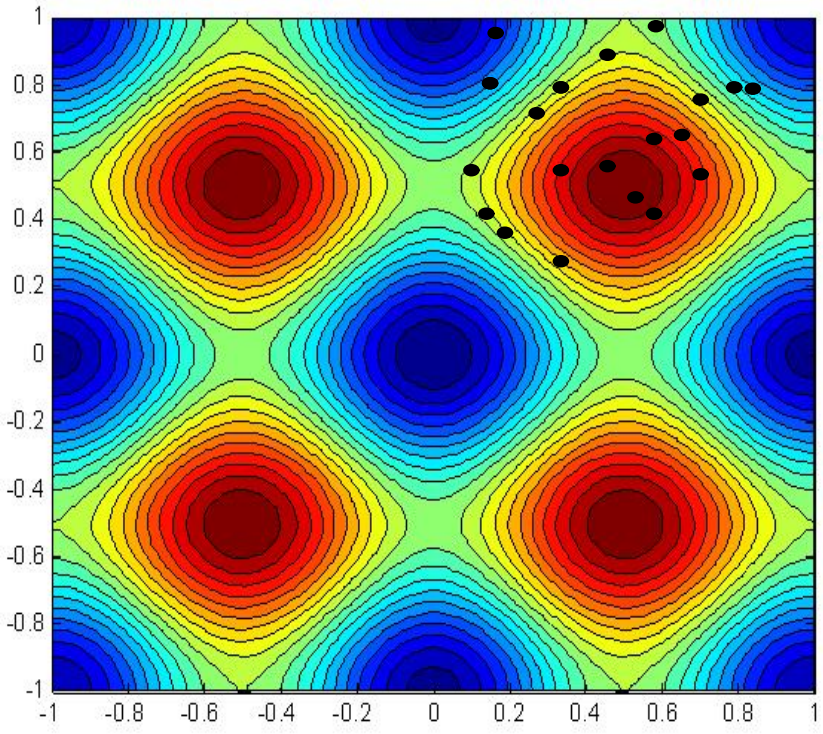


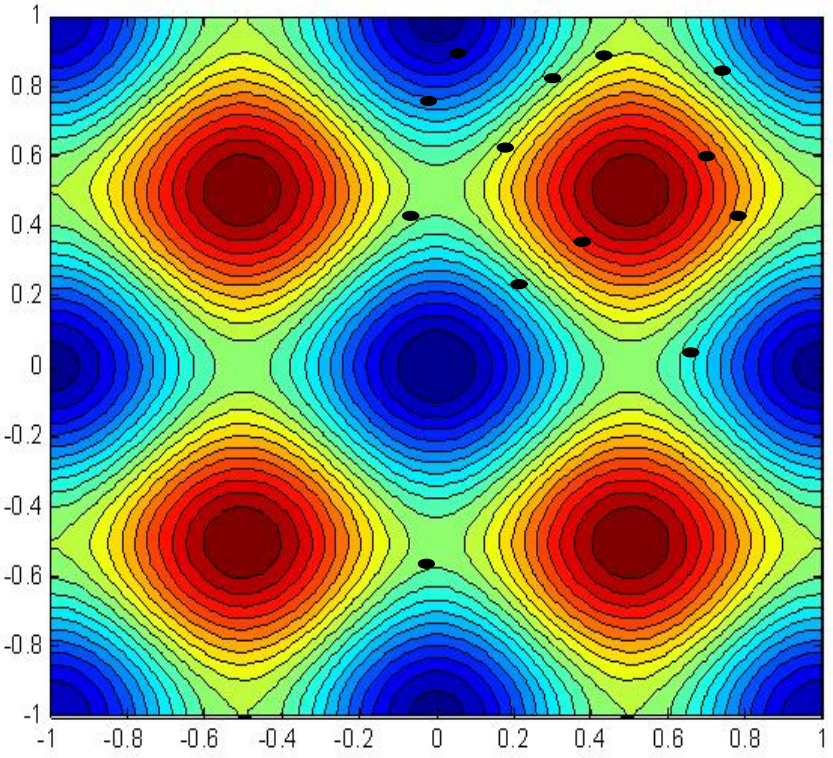
- 例： 用遗传算法求解下面一个Rastrigin函数的最小值。

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

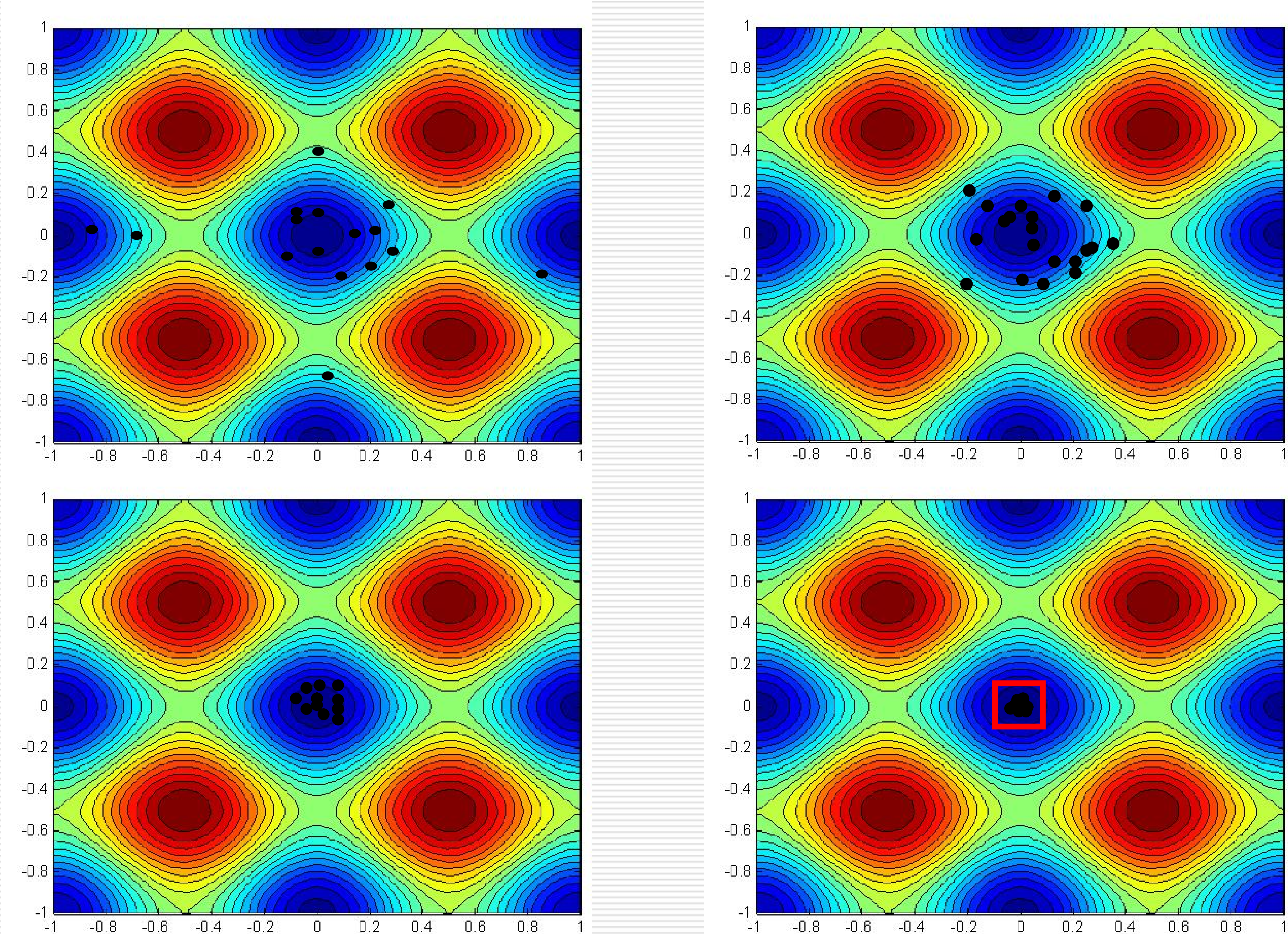
$$-5 \le x_i \le 5 \qquad i = 1, 2$$

- 初始种群　　第二代种群





在迭代60、80、95、100次时的种群



6.2.3 编码

1. 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。

（1） 二进制编码

二进制编码：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。

6.2.3 编码

(1) 二进制编码（续）

●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；算法处理的模式数最多。

●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。

15：01111　　16：10000

② 要先给出求解的精度，即编码长度。优化中难以调整。

③ 求解高维问题时，二进制编码串很长，算法的搜索效率低。

（2） Gray 编码

Gray编码：将二进制编码通过一个变换进行转换得到的编码。

二进制串 $< \beta_1 \beta_2 \cdots \beta_n >$　　Gray $< \gamma_1 \gamma_2 \cdots \gamma_n >$

二进制编码 → Gray编码　　Gray编码 → 二进制编码

$$\gamma_k = \begin{cases} \beta_1 & k = 1 \\ \beta_{k-1} \oplus \beta_k & k > 1 \end{cases}$$

对2取模的加法

$$\beta_k = \sum_{i=1}^{k} \gamma_i \pmod 2$$

6.2.3 编码

2.实数编码

- 采用实数表达法**不必进行数制转换**，可直接在解的表现型上进行遗传操作。

3.多参数级联编码

- **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**。

6.2.4 群体设定

1. 初始种群的产生

- 随机产生群体规模数目的个体作为初始群体。

- 随机产生一定数目的个体，从中挑选最好的个体加到初始群体中。这种过程不断迭代，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群体。

6.2.4 群体设定

2. 种群规模的确定

- 群体规模太小，遗传算法的优化性能不太好，易陷入局部最优解。

- 群体规模太大，计算复杂。

- **模式定理**表明：若群体规模为M，则遗传操作可从这M个个体中生成和检测 M^3 个模式，并在此基础上能够不断形成和优化积木块，直到找到最优解。

6.2.5 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

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

6.2.5 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**（deceptive problem）。

- **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（fitness scaling）**或**定标**：对适应度函数值域的某种映射变换。

2. 适应度函数的尺度变换(续)

（1）线性变换：

$$f' = af + b$$

满足 $f'_{avg} = f_{avg}$，$f'_{\max} = C_{mult} \cdot f_{avg}$

$$a = \frac{(C_{mult} - 1) f_{avg}}{f_{\max} - f_{avg}}$$

$$b = \frac{(f_{\max} - C_{mult} f_{avg}) f_{avg}}{f_{\max} - f_{avg}}$$

满足最小适应度值非负

$$a = \frac{f_{avg}}{f_{avg} - f_{\min}}$$

$$b = \frac{-f_{\min} f_{avg}}{f_{avg} - f_{\min}}$$

6.2.5 适应度函数

2. 适应度函数的尺度变换(续)

（2）幂函数变换法：

$$f' = f^K$$

（3）指数变换法：

$$f' = \mathrm{e}^{-af}$$

6.2.6 选择

1. 个体选择概率分配方法

- 选择操作也称为复制（reproduction）操作：从当前群体中按照一定概率选出优良的个体，使它们有机会作为父代繁殖下一代子孙。

- 判断个体优良与否的准则是各个个体的适应度值：个体适应度越高，其被选择的机会就越多。

6.2.6 选择

1. 个体选择概率分配方法

（1）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。

- 个体 i 被选择的概率为：

$$p_{si} = \frac{f_i}{\sum_{i=1}^{M} f_i}$$

6.2.6 选择

1. 个体选择概率分配方法

（2） 排序方法 （rank-based model）

① 线性排序：J. E. Baker

- 群体成员按适应值大小从好到坏依次排列：$x_1, x_2, \cdots, x_M$
- 个体 x_i分配选择概率 p_i

$$p_i = \frac{a - bi}{M(M+1)}$$

- 按转盘式选择的方式选择父体

6.2.6 选择

1. 个体选择概率分配方法

（2） 排序方法 （rank-based model）

② 非线性排序： Z. Michalewicz

● 将群体成员按适应值从好到坏依次排列，并按下式分配选择概率：

$$p_i = \begin{cases} q(1-q)^{i-1} & i = 1,2,\cdots,M-1 \\ (1-q)^{M-1} & i = M \end{cases}$$

6.2.6 选择

1. 个体选择概率分配方法

（2） 排序方法 （rank-based model）

- 可用其他非线性函数来分配选择概率，只要满足以下条件：

(1) 若 $P=\{x_1, x_2 \cdots, x_M\}$ 且 $f(x_1) \geq f(x_2) \geq \ldots \geq f(x_M)$, 则$p_i$满足

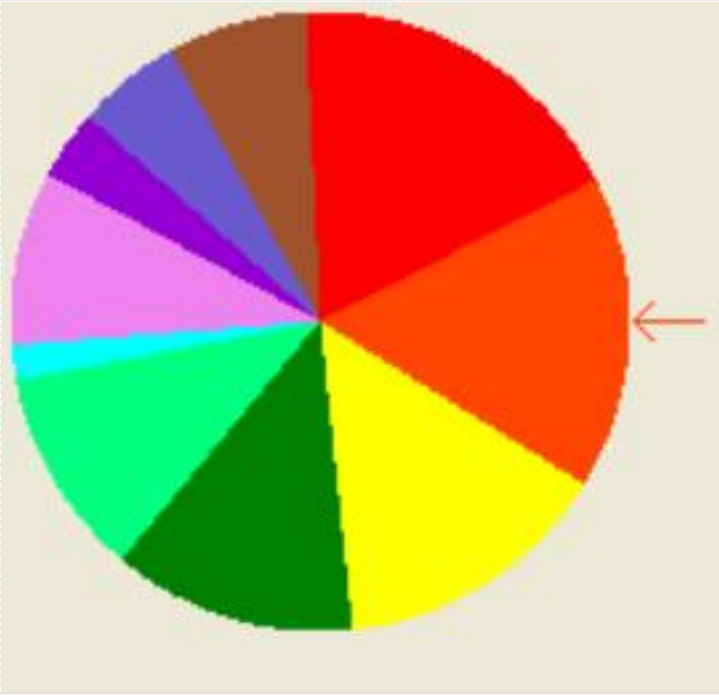
$$p_1 \geq p_2 \geq \cdots \geq p_M$$

(2) $\sum_{i=1}^{M} p_i = 1$

2. 选择个体方法

（1）转盘赌选择

- 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，它落入转盘的哪个区域就选择相应的个体交叉。



2. 选择个体方法

（1）转盘赌选择

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**

6.2.6 选择

2. 选择个体方法

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

- **锦标赛选择方法：**从群体中随机选择一定数量个体（有放回抽样），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。

6.2.6 选择

2. 选择个体方法

（3）最佳个体保存方法

- **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。

具体步骤：（1）将上一代群体中的最佳个体X抽取出来不做交叉变异，其他所有个体进行交叉变异。（2）选择本轮的最优的部分个体进入下一代（其中必须包含上一代的最佳个体X）。

这种方式的目的：让每轮迭代的最优解一定是单调变化的。其他的方法并不保护本轮最优解，可能会出现震荡现象。

6.2.7 交叉

1. 基本的交叉算子

（1）一点交叉（single-point crossover）

- 一点交叉：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）二点交叉 （two-point crossover）

- 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。

6.2.7 交叉

2. 修正的交叉方法

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

$$A = 9\ \ 8\ \ 4 \mid 5\ \ 6\ \ 7 \mid 1\ \ 3\ \ 2$$

$$B = 8\ \ 7\ \ 1 \mid 2\ \ 3\ \ 9 \mid 5\ \ 4\ \ 6$$

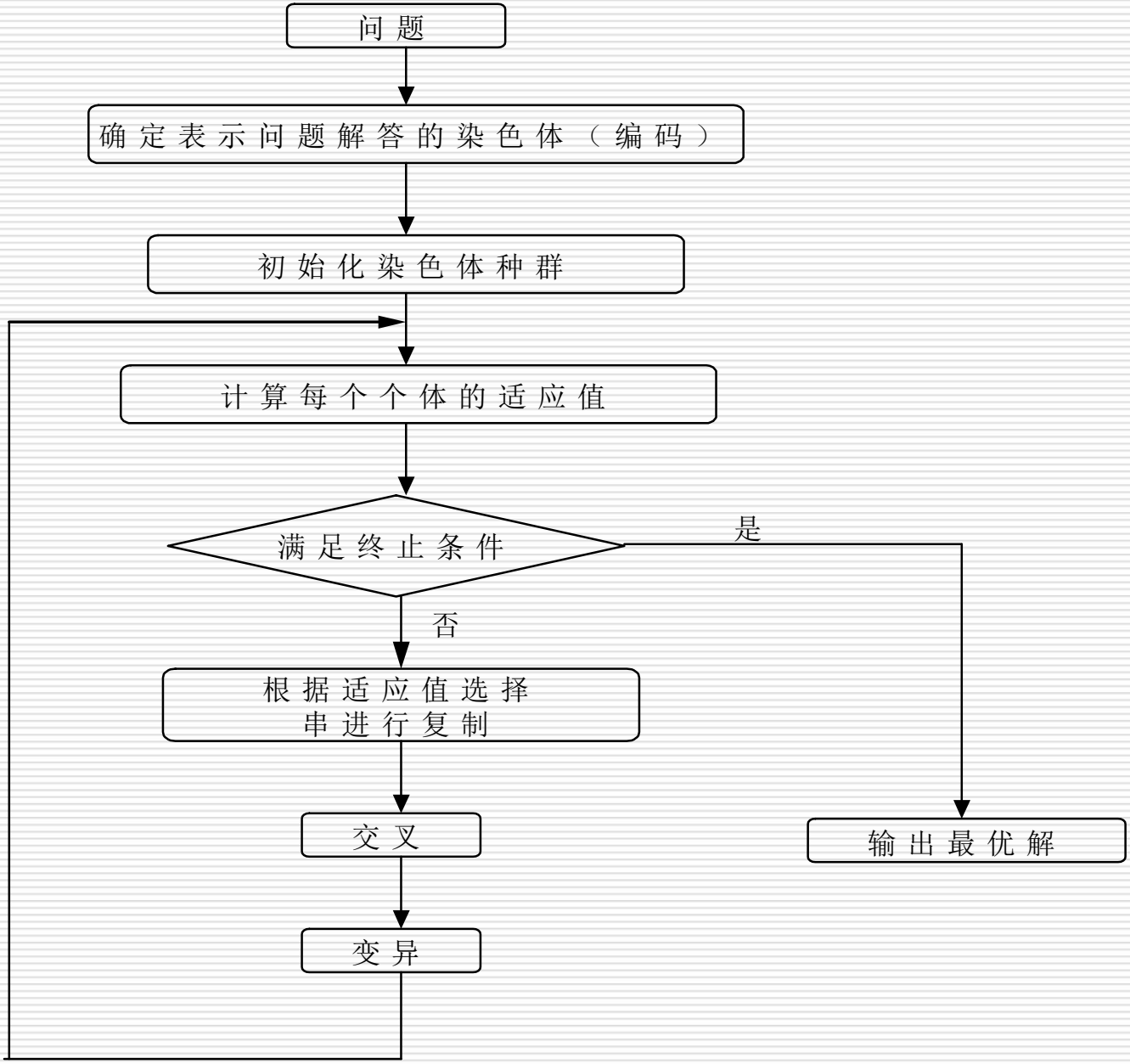
$$A' = 9\ \ 8\ \ 4 \mid 2\ \ 3\ \ 9 \mid 1\ \ 3\ \ 2$$

$$B' = 8\ \ 7\ \ 1 \mid 5\ \ 6\ \ 7 \mid 5\ \ 4\ \ 6$$

6.2.8 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，向左或者向右移动一个随机位数。

6.2.9 遗传算法的一般步骤



6.2.9 遗传算法的一般步骤

（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

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

6.2.9 遗传算法的一般步骤

（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

$$crosspop(t+1)$$

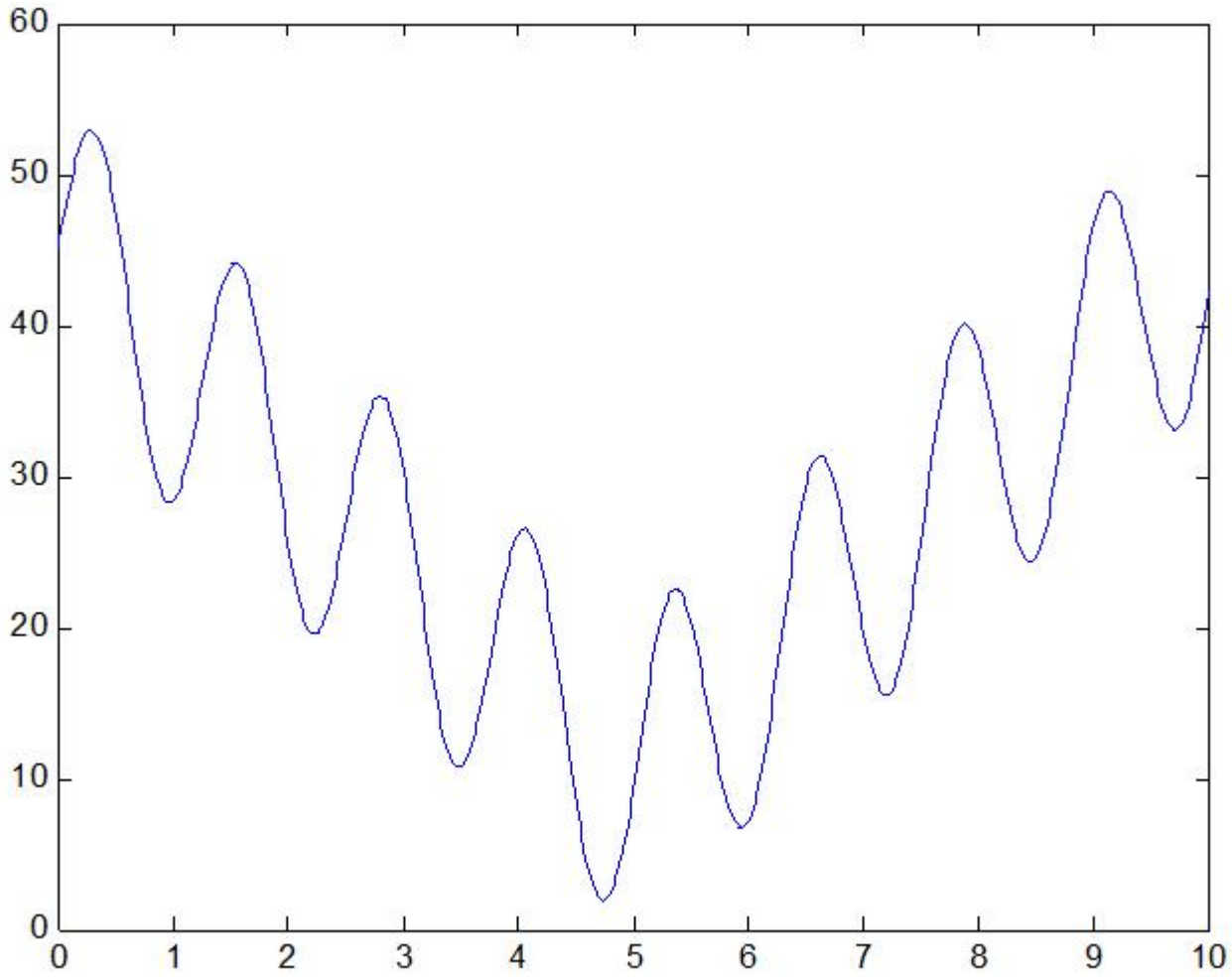
（5）以一个较小的概率 p_m 使染色体的一个基因发生变异，形成 $mutpop(t+1)$；　$t:=t+1$，成为一个新的群体

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

返回（2）。

6.2.9 遗传算法的一般步骤（补充案例）

求函数y=10*sin(5*x)+7*abs(x-5)+10的极大值



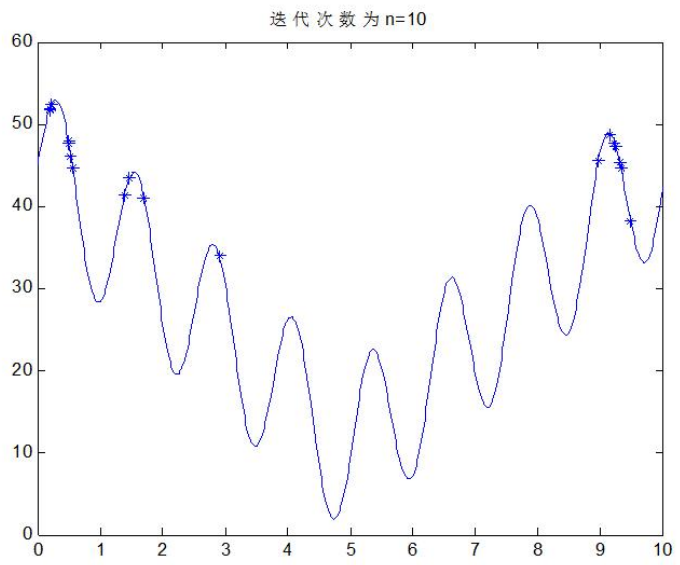
6.2.9 遗传算法的一般步骤（补充案例）

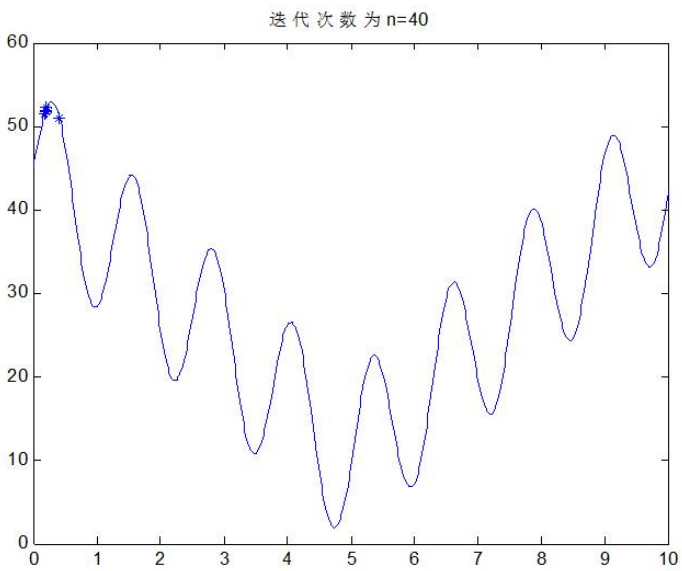
```
function main()
clear;
clc;
%种群大小
popsize=100;
%二进制编码长度
chromlength=10;
%交叉概率
pc = 0.6;
%变异概率
pm = 0.001;
%初始种群
pop = initpop(popsize,chromlength);
```

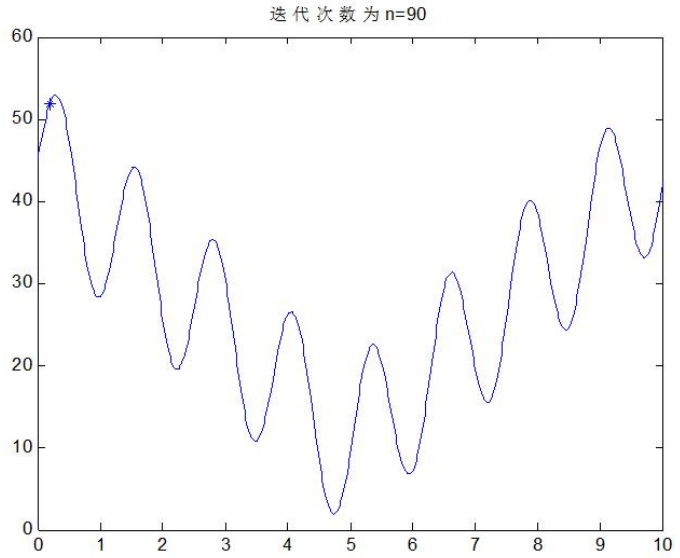
6.2.9 遗传算法的一般步骤（补充案例）

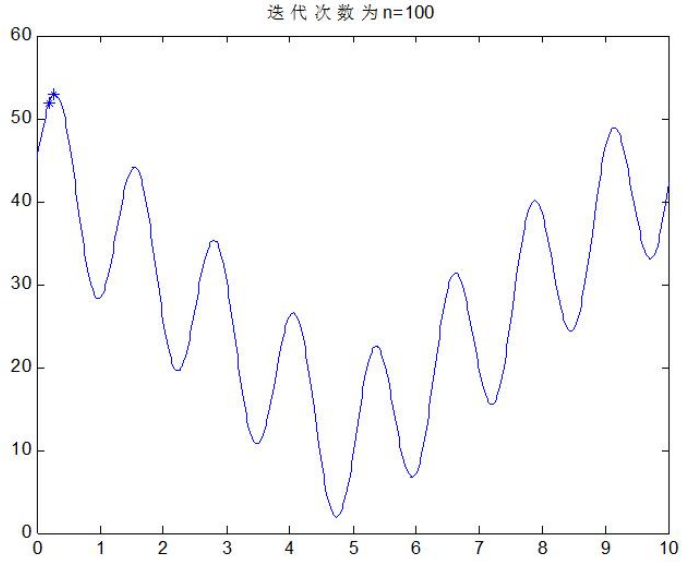
```
for i = 1:100        %100为迭代次数，用户根据需要设定
    %计算适应度值（函数值）
    objvalue = cal_objvalue(pop);
    fitvalue = objvalue;
    %可以增加判定是否满足要求，进行分支
    %选择操作
    newpop = selection(pop,fitvalue);
    %交叉操作
    newpop = crossover(newpop,pc);
    %变异操作
    newpop = mutation(newpop,pm);
    %更新种群
    pop = newpop;
    %寻找最优解
    [bestindividual,bestfit] = best(pop,fitvalue);
end
```

6.2.9 遗传算法的一般步骤（补充案例）









6.2.10 遗传算法的特点

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，搜索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。

第6章 智能计算及其应用

- 6.1 进化算法的产生与发展
- 6.2 基本遗传算法
- 6.3 遗传算法的改进算法
- 6.4 遗传算法的应用

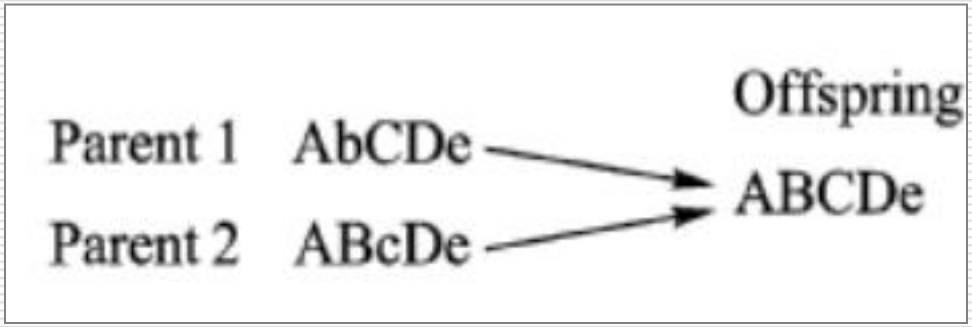
6.3 遗传算法的改进算法

- **6.3.1 双倍体遗传算法**
- **6.3.2 双种群遗传算法**
- **6.3.3 自适应遗传算法**

6.3.1 双倍体遗传算法

1. 基本思想

- 双倍体遗传算法采用**显性**和**隐性**两个染色体同时进行进化，提供了一种记忆以前有用的基因块的功能。
- 双倍体遗传算法采用**显性遗传**。



- 双倍体遗传延长了有用基因块的寿命，提高了算法的收敛能力，在变异概率低的情况下能保持一定水平的多样性。

6.3.1 双倍体遗传算法

2. 双倍体遗传算法的设计

（1）编码/解码：两个染色体（显性、隐性）

（2）复制算子：计算显性染色体的适应度，按照显性染色体的复制概率将个体复制到下一代群体中。

（3）交叉算子：两个个体的显性染色体交叉、隐性染色体也同时交叉。

（4）变异算子：个体的显性染色体按正常的变异概率变异；隐性染色体按较大的变异概率变异。

（5）双倍体遗传算法显/隐性重排算子：个体中适应值较大的染色体设为显性染色体，适应值较小的染色体设为隐性染色体。

6.3.2 双种群遗传算法

1. 基本思想

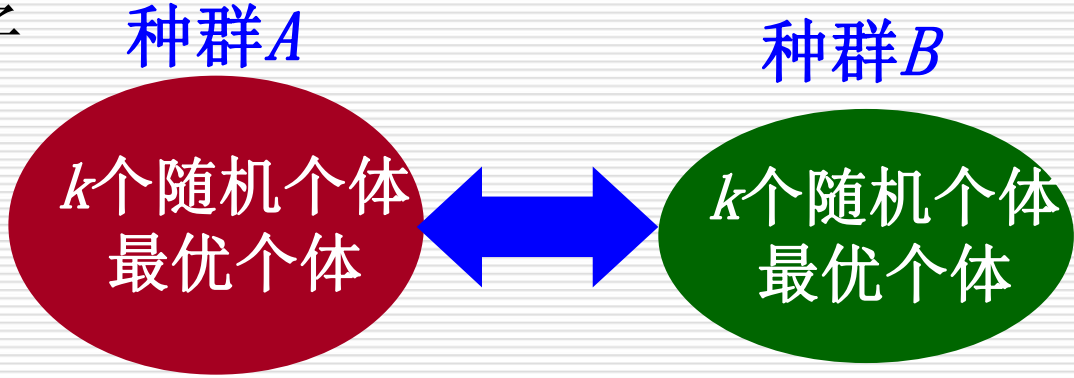
- 在遗传算法中使用多种群同时进化，并交换种群之间优秀个体所携带的遗传信息，以打破种群内的平衡态达到更高的平衡态，有利于算法跳出局部最优。

- **双种群遗传算法**：建立两个遗传算法群体，分别独立地运行复制、交叉、变异操作，同时当每一代运行结束以后，选择两个种群中的随机个体及最优个体分别交换。

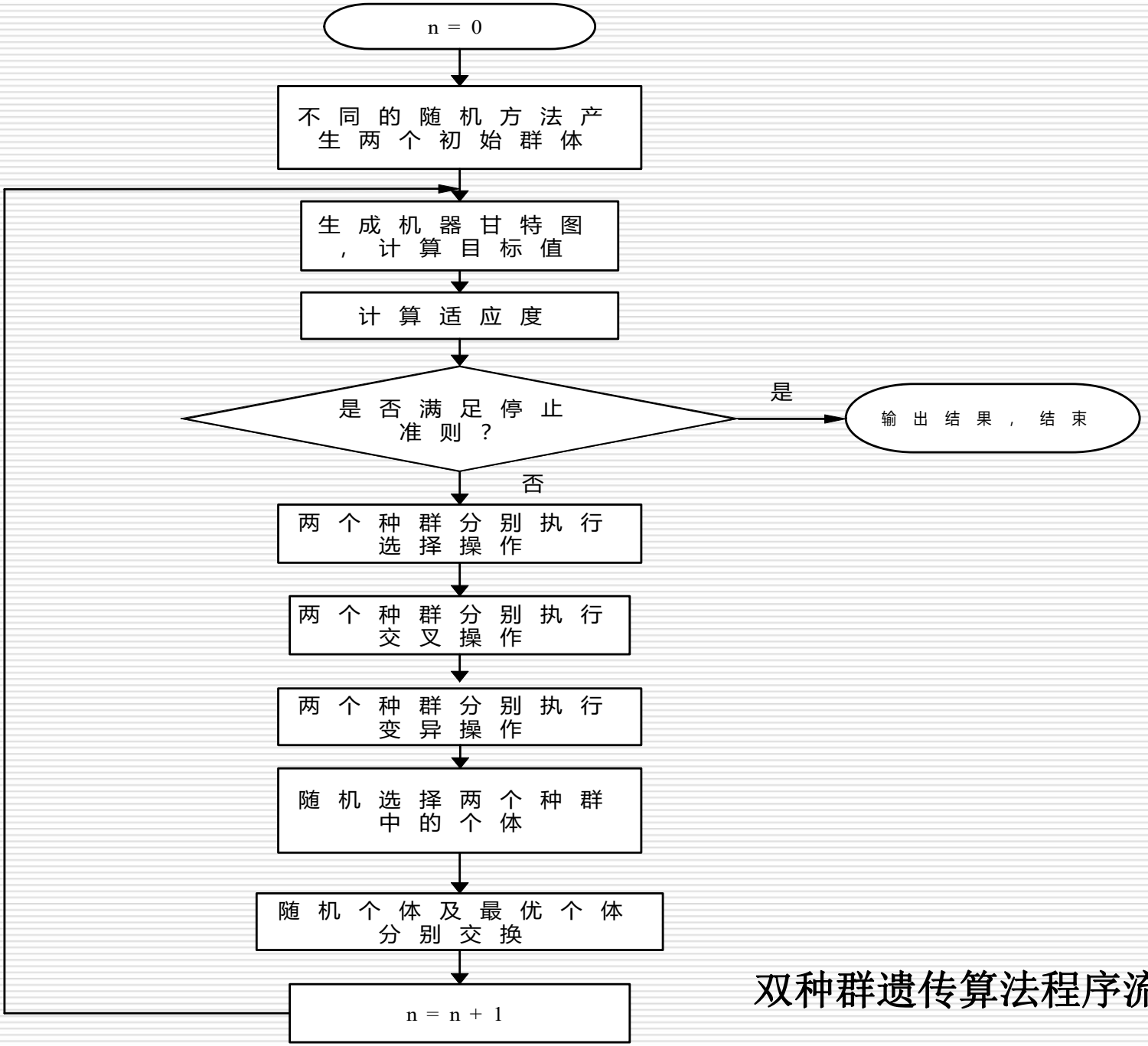
6.3.2 双种群遗传算法

2. 双种群遗传算法的设计

- 编码/解码设计
- 交叉算子、变异算子
- 杂交算子



设种群A与种群B，当A与B种群都完成了选择、交叉、变异算子后，产生一个随机数num，随机选择A中num个个体与A中最优个体，随机选择B中num个个体与B中最优个体，交换两者，以打破平衡态。



双种群遗传算法程序流程图

6.3.3 自适应遗传算法

1. 基本思想

- Srinvivas M.，Patnaik L. M.等在1994年提出一种自适应遗传算法(adaptive genetic algorithms，AGA)：交叉概率P_c和变异概率 P_m能随适应度自动改变。

- AGA：当种群各个体适应度趋于一致或者趋于局部最优时，使P_c和P_m增加，以跳出局部最优；而当群体适应度比较分散时，使P_c和P_m减少，以利于优良个体的生存。

- 同时，对于适应度高于群体平均适应值的个体，选择较低的P_c和P_m，使得该解得以保护进入下一代；对低于平均适应值的个体，选择较高的P_c和P_m值，使该解被淘汰。

6.3.3 自适应遗传算法

2. 自适应遗传算法的步骤

（1） 编码/解码设计。

（2） 初始种群产生：N（N 是偶数）个候选解，组成初始解集。

（3） 定义适应度函数，计算适应度 f_i。

（4） 按轮盘赌规则选择N 个个体，计算 f_{avg}和$f_{\max}$。

（5）将群体中的各个个体随机搭配成对，共组成$N/2$对，对每一对个体，按照自适应公式计算自适应交叉概率P_c，随机产生$R(0,1)$，如果$R< P_c$则对该对染色体进行交叉操作。

6.3.3 自适应遗传算法

2. 自适应遗传算法的步骤（续）

（6）对于群体中的所有个体，共N个，按照自适应变异公式计算自适应变异概率 P_m，随机产生 $R(0,1)$，如果 $R < P_m$ 则对该染色体进行变异操作。

（7）计算由交叉和变异生成新个体的适应度，新个体与父代一起构成新群体。

（8）判断是否达到预定的迭代次数，是则结束；否则转（4）。

6.3.3 自适应遗传算法

3. 适应的交叉概率与变异概率

$$P_c=\begin{cases}\dfrac{k_1(f_{\max}-f')}{f_{\max}-f_{\text{avg}}}, & f'>f_{\text{avg}}\\ k_{2,} & f'\le f_{\text{avg}}\end{cases}$$

$$P_m=\begin{cases}\dfrac{k_3(f_{\max}-f\ )}{f_{\max}-f_{\text{avg}}}, & f>f_{\text{avg}}\\ k_4, & f\le f_{\text{avg}}\end{cases}$$

- 普通自适应算法中，当个体适应度值越接近最大适应度值时，交叉概率与变异概率就越小；当等于最大适应度值时，交叉概率和变异概率为零。

- 6.1 进化算法的产生与发展
- 6.2 基本遗传算法
- 6.3 遗传算法的改进算法
- 6.4 遗传算法的应用

6.4 遗传算法的应用

1. 流水车间调度问题

- 问题描述：n 个工件要在 m 台机器上加工，每个工件需要经过 m 道工序，每道工序要求不同的机器，n 个工件在 m 台机器上的加工顺序相同。工件在机器上的加工时间是给定的，设为

$$t_{ij}(i=1,\cdots,n;\ j=1,\cdots,m)$$

- 问题的目标：确定 n 个工件在每台机器上的最优加工顺序，使最大流程时间达到最小。

6.4 遗传算法的应用

1. 流水车间调度问题

- 假设：

(1) 每个工件在机器上的加工顺序是给定的。

(2) 每台机器同时只能加工一个工件。

(3) 一个工件不能同时在不同的机器上加工。

(4) 工序不能预定。

(5) 工序的准备时间与顺序无关，且包含在加工时间中。

(6) 工件在每台机器上的加工顺序相同，且是确定的。

1. 流水车间调度问题

- 问题的数学模型：

$c(j_i,k)$：工序j_i在机器k上的加工完工时间，$\{j_1,j_2,\cdots,j_n\}$：工件的调度

n个工件、m台机器的流水车间调度问题的完工时间：

$$c(j_1,1)=t_{j_1 1}$$

$$c(j_1,k)=c(j_1,k-1)+t_{j_1 k},\quad k=2,\ldots,m$$

$$c(j_i,1)=c(j_{i-1},1)+t_{j_i 1},\quad i=2,\ldots,n$$

$$c(j_i,k)=\max\{c(j_{i-1},k),c(j_i,k-1)\}+t_{j_i k},\qquad i=2,\ldots,n;k=2,\ldots,m$$

最大流程时间：$c_{\max}=c(j_n,m)$

调度目标：确定$\{j_1,j_2,\cdots,j_n\}$使得$c_{\max}$最小

2. 求解流水车间调度问题的遗传算法设计

（1） FSP的编码方法

对于FSP，最自然的编码方式是用染色体表示工件的顺序。

对于有四个工件的FSP，第 k 个染色体 $v_k=[1,2,3,4]$，表示工件的加工顺序为： j_1, j_2, j_3, j_4 。

2. 求解流水车间调度问题的遗传算法设计

（2）FSP的适应度函数

$c_{\max}^{k}$：第k个染色体 v_k 的最大流程时间

FSP的适应度函数：

$$eval(v_k) = \frac{1}{c_{\max}^{k}}$$

3. 求解FSP的遗传算法实例

例6.1　Ho 和 Chang(1991) 给出的5个工件、4台机器问题。

加工时间表

工件 j	t_{j1}	t_{j2}	t_{j3}	t_{j4}
1	31	41	25	30
2	19	55	3	34
3	23	42	27	6
4	13	22	14	13
5	33	5	57	19

6.4 遗传算法的应用

用穷举法求得最优解：4-2-5-1-3，加工时间：213；

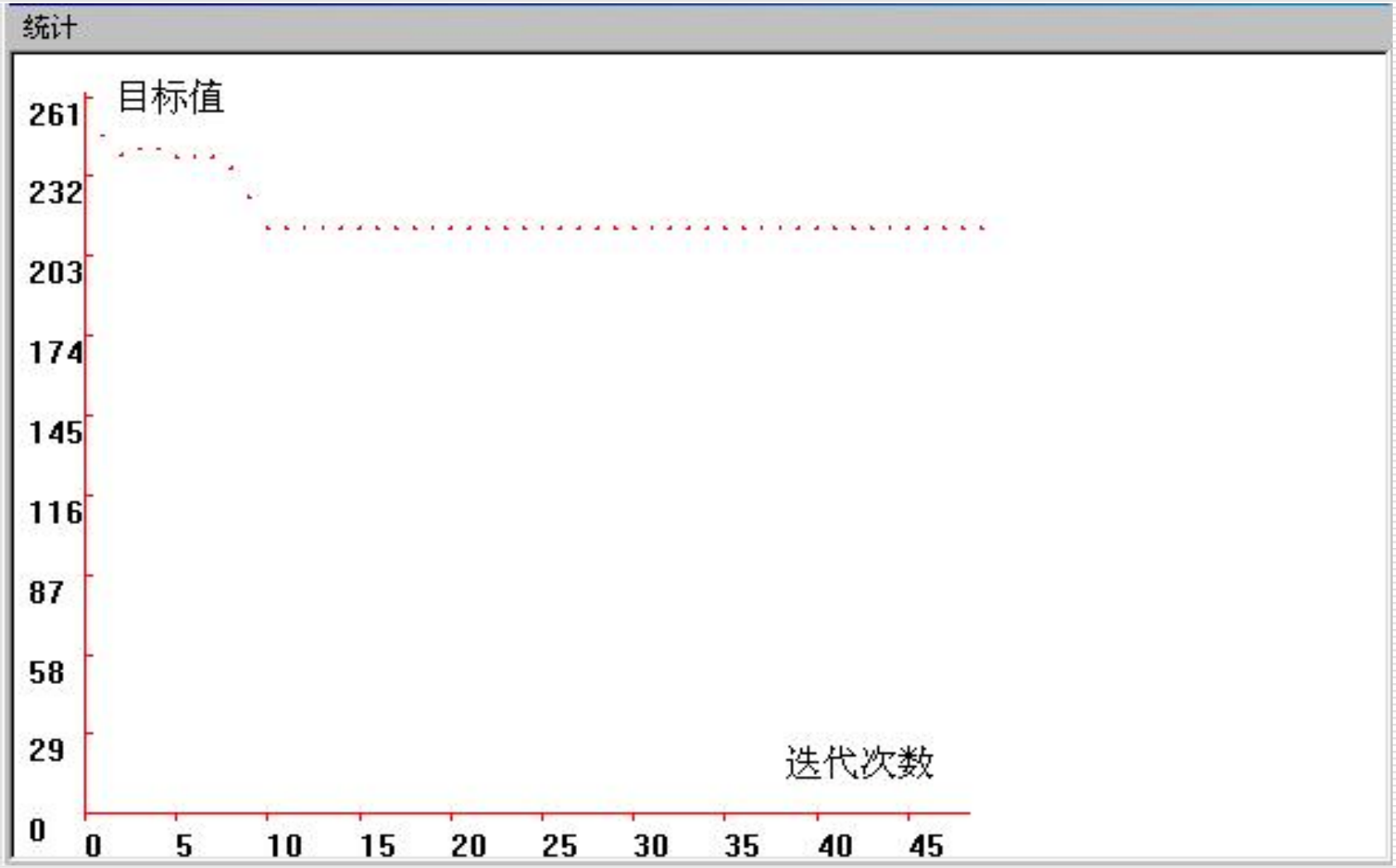
最劣解：1-4-2-3-5，加工时间：294；平均解的加工时间：265。

用遗传算法求解。选择交叉概率 $p_c = 0.6$，变异概率 $p_m = 0.1$，种群规模为20，迭代次数 $N = 50$。

遗传算法运行的结果

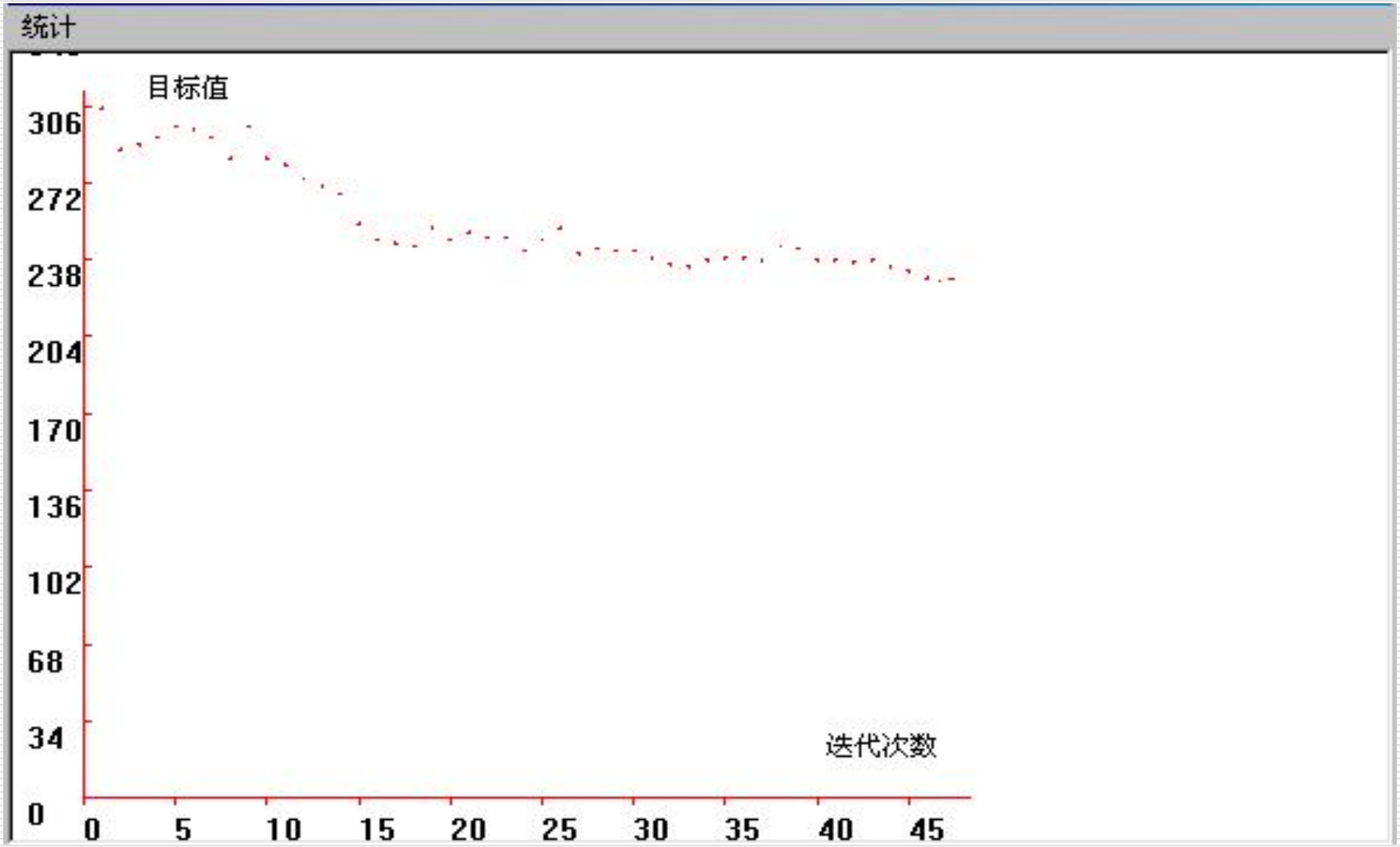
总运行次数	最好解	最坏解	平均	最好解的频率	最好解的平均代数
20	213	221	213.95	0.85	12

6.4 遗传算法的应用



最优解收敛图

6.4 遗传算法的应用



平均值收敛图

THE END