

第 6 章 遗传算法及其应 用



第 6 章 遗传算法及其应用

✓ 6.1 遗传算法的产生与发展

□ 6.2 遗传算法的基本算法

□ 6.4 遗传算法的应用

6.1 遗传算法的产生与发展

- **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

6.1 遗传算法的产生与发展

- 6.1.1 遗传算法的生物背景
- 6.1.2 遗传算法的基本思想
- 6.1.3 遗传算法的发展历史
- 6.1.4 设计遗传算法的基本原则与内容

6.1.1 遗传算法的生物学背景

□ **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。

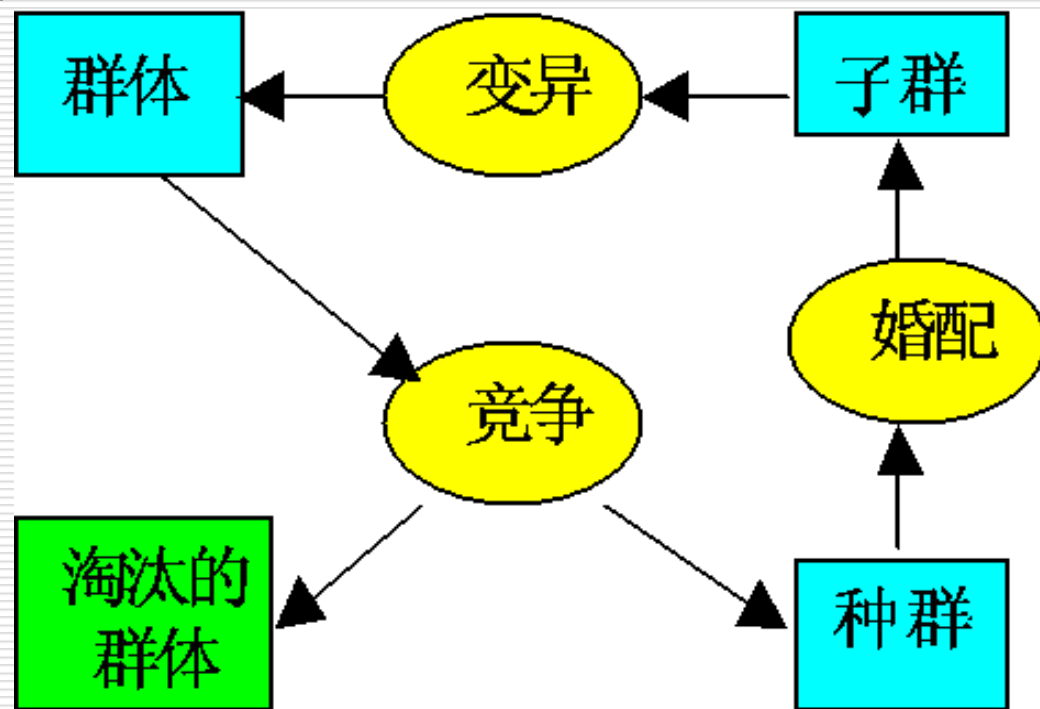
□ **生物进化的基本过程**：

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。

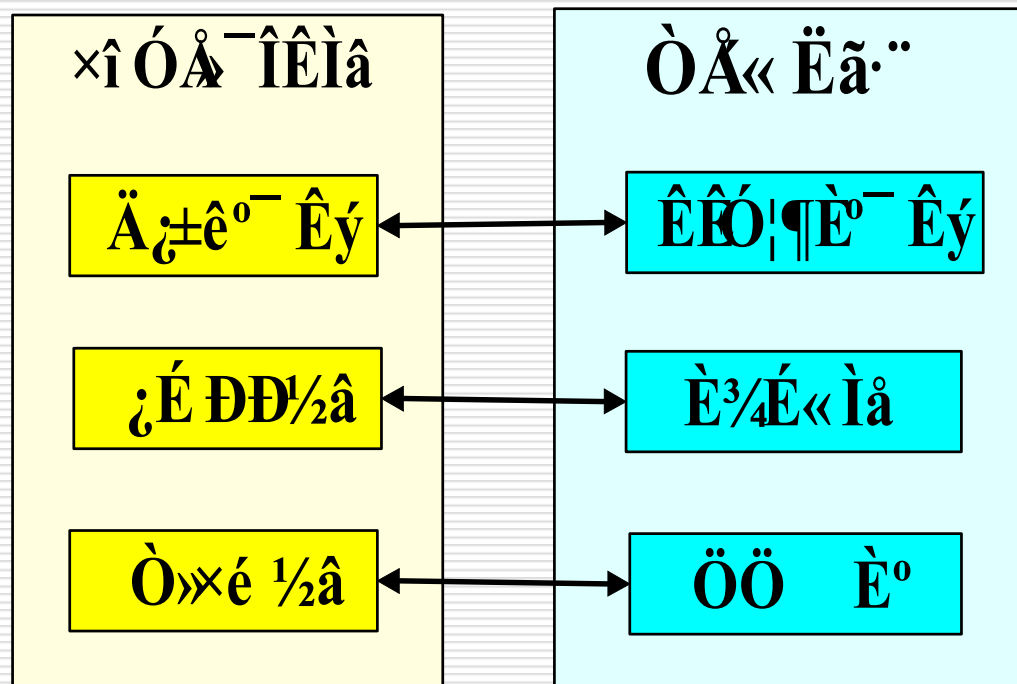
等位基因 (alleles)：基因所取的值。



6.1.2 遗传算法的基本思想

生物遗传概念	遗传算法中的应用
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 ）	选择染色体中的某些基因进行变异

6.1.2 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

6.1.4 遗传算法设计的基本内容

编码方案：怎样把优化问题的解进行编码。

适应度函数：怎样根据目标函数构建适应度函数。

选择策略：优胜劣汰。

控制参数：种群的规模、算法执行的最大代数、执行不同遗传操作的概率等。

遗传算子：选择、交叉、变异。

算法终止准则：规定一个最大的演化代数，或算法在连续多少代以后解的适应值没有改进。

第 6 章 遗传算法及其应用

□ 6.1 遗传算法的产生与发展

✓ 6.2 遗传算法的基本算法

□ 6.4 遗传算法的应用

6.2 遗传算法的基本算法

□ 遗传算法的五个基本要素：

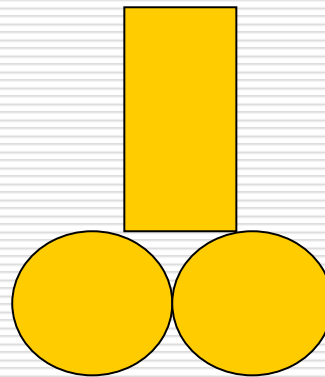
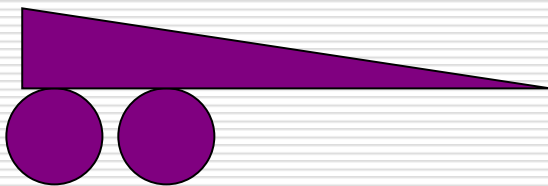
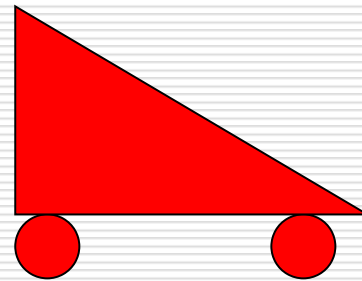
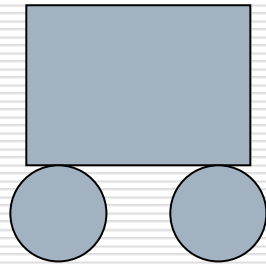
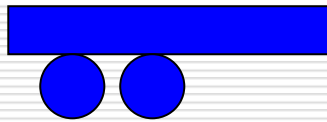
- 参数编码
- 初始群体的设定
- 适应度函数的设计
- 遗传操作设计
- 控制参数设定

6.2 遗传算法的基本算法

- 6.2.1 编码
- 6.2.2 群体设定
- 6.2.3 适应度函数
- 6.2.4 选择
- 6.2.5 交叉
- 6.2.6 变异
- 6.2.7 遗传算法的一般步骤

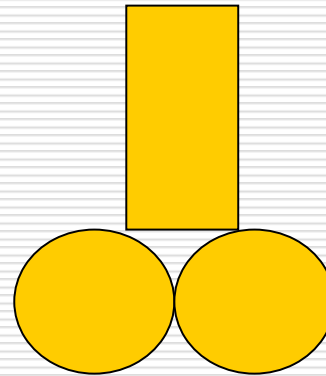
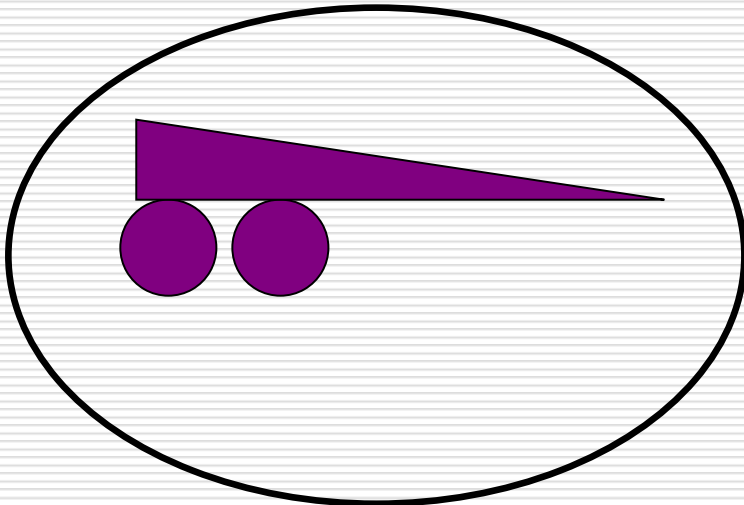
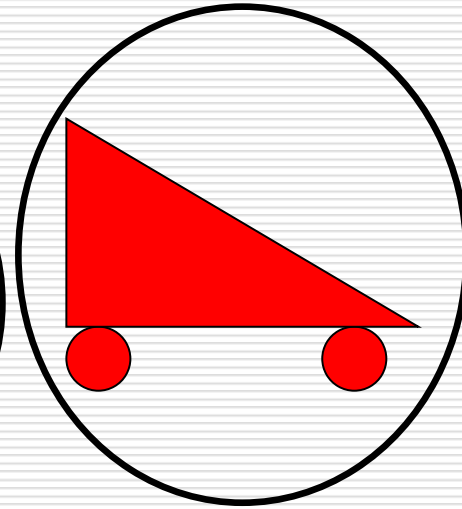
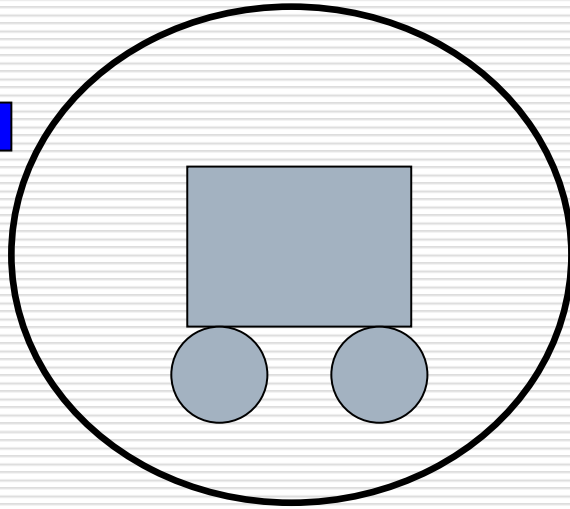
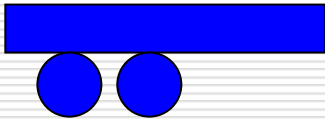
6.2 遗传算法的基本算法—辅助理解

Initial population



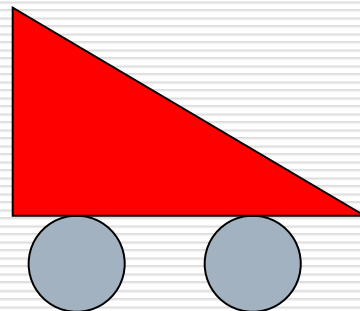
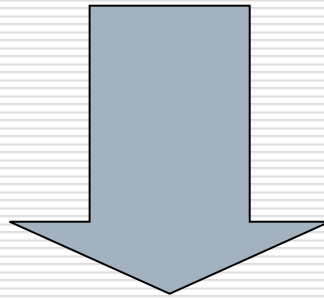
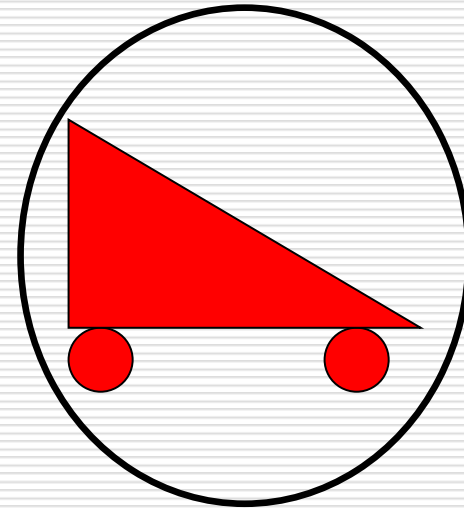
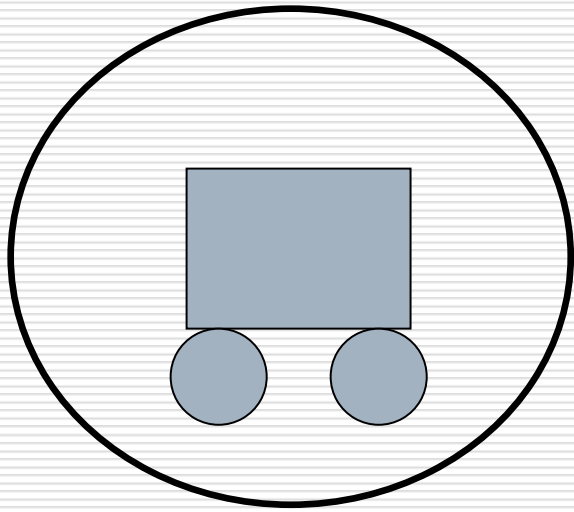
6.2 遗传算法的基本算法—辅助理解

Select



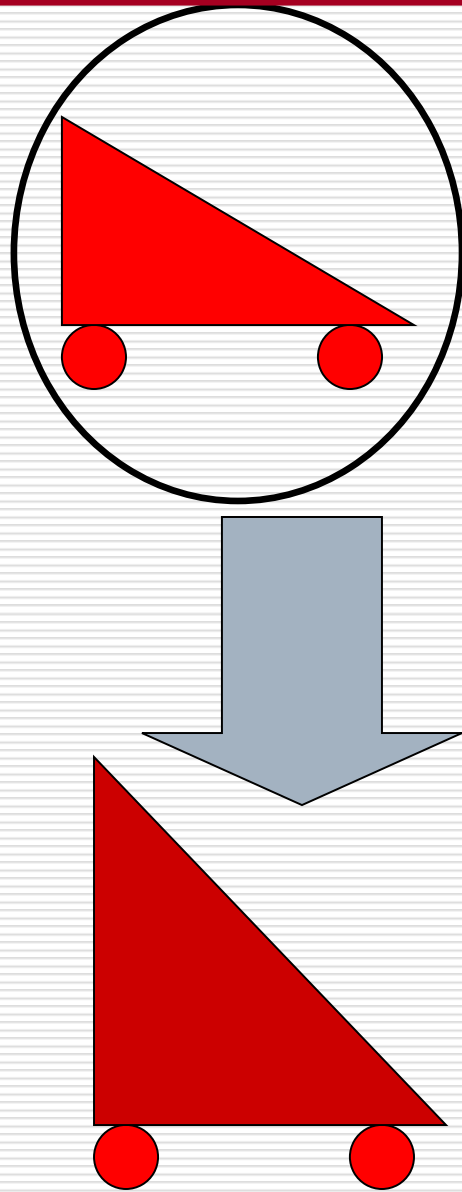
6.2 遗传算法的基本算法—辅助理解

Crossover



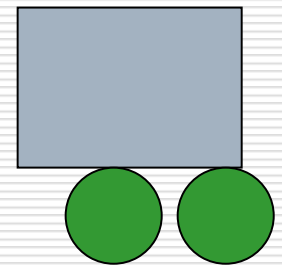
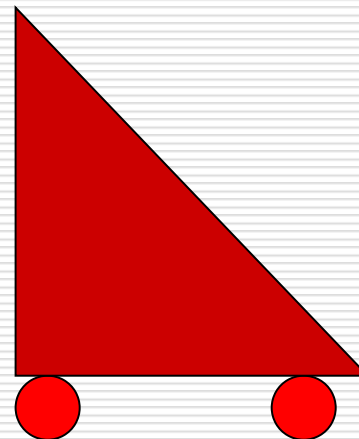
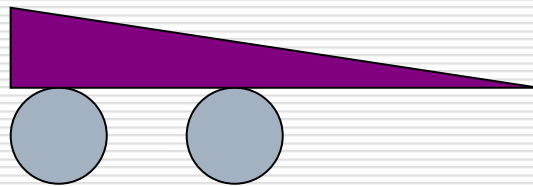
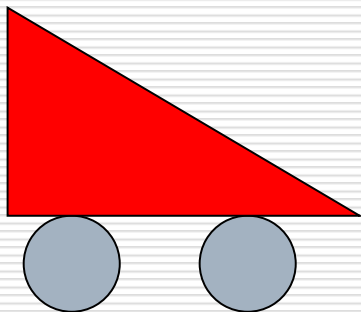
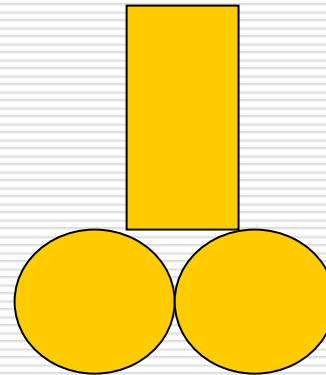
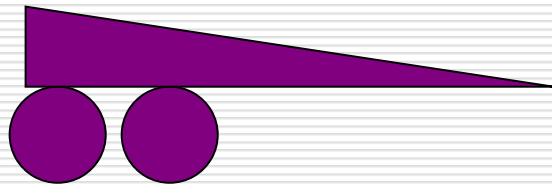
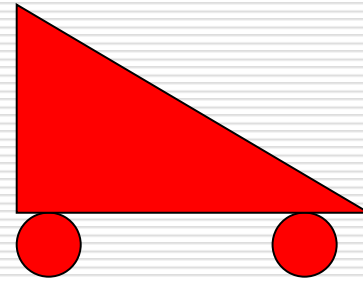
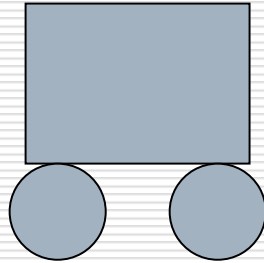
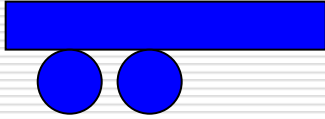
6.2 遗传算法的基本算法—辅助理解

mutation



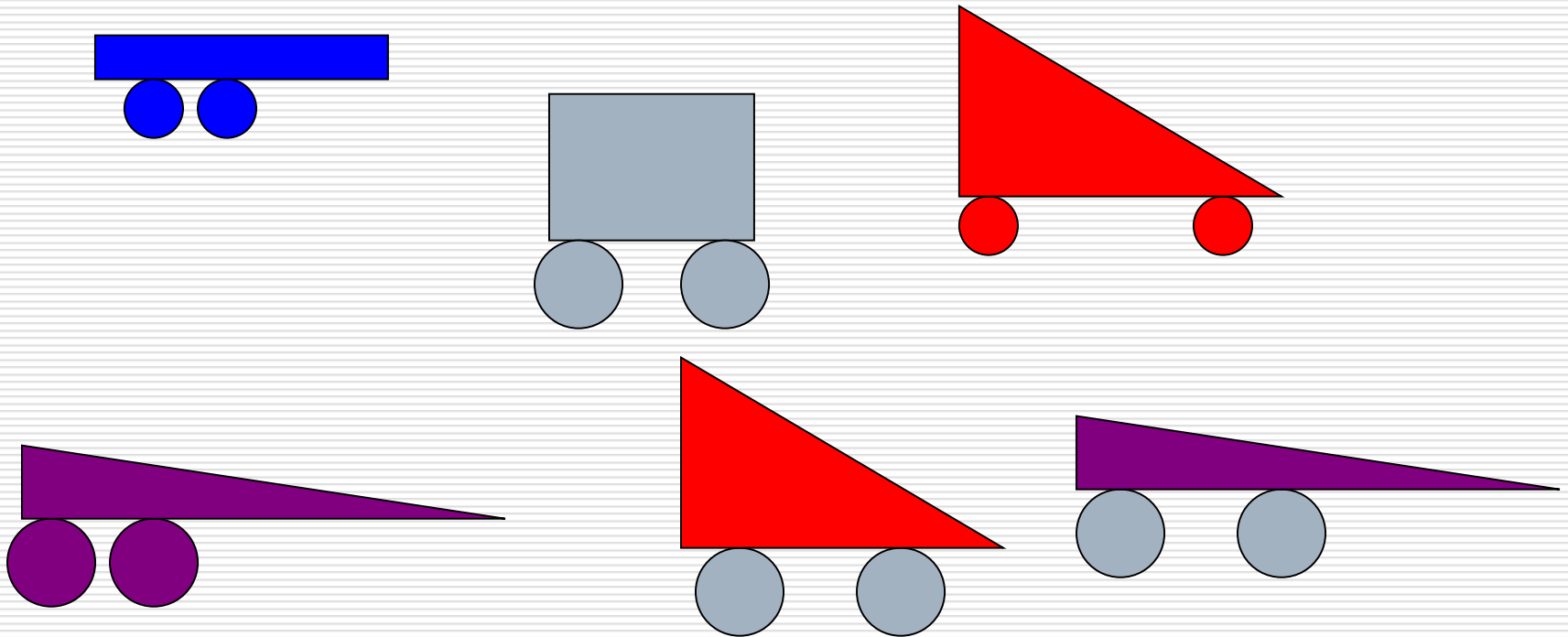
6.2 遗传算法的基本算法—辅助理解

Old population + children



6.2 遗传算法的基本算法—辅助理解

New Population: Generation 2



6.2.1 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

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

6.2.1 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

①要先给出求解的精度

②求解高维优化问题的二进制编码串长，算法的搜索效率低

6.2.1 编码

1. 位串编码

(2) 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 $\langle \beta_1 \beta_2 \dots \beta_n \rangle$

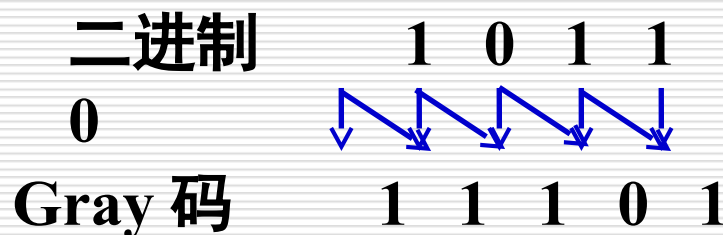
Gray $\langle \gamma_1 \gamma_2 \dots \gamma_n \rangle$

二进制编码 \rightarrow Gray 编码

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

Gray 编码 \rightarrow 二进制编码

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



6.2.1 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
- **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

6.2.2 群体设定

1. 初始种群的产生

- ① 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。
- ② **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。

6.2.2 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

6.2.3 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 $Fit(f(x)) = \frac{1}{f(x)}$



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

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

6.2.3 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。
- 适应度函数的**尺度变换**（ fitness scaling ）或者**定标**：对适应度函数值域的某种映射变换。

6.2.3 适应度函数

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



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

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

6.2.3 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:

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

6.2.4 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**个体的**适应度值**：个体适应度越高，其被选择的机会就越多

6.2.4 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：

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

6.2.4 选择

1. 个体选择概率分配方法

② 排序方法 (rank-based model)

□ 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, \dots, x_N

◆ 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

6.2.4 选择

1. 个体选择概率分配方法

② 排序方法 (rank-based model)

□ 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：

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

6.2.4 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 $P = \{x_1, x_2, \dots, x_M\}$ 且 $f(x_1) \geq f(x_2) \geq \dots \geq f(x_M)$ ，则 p_i 满足

$$p_1 \geq p_2 \geq \dots \geq p_M$$

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

6.2.4 选择

2. 选择个体方法

① 转盘赌选择 (roulette wheel selection)

- 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

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

6.2.4 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 k 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）：每次按赌轮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。

6.2.4 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。

6.2.5 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。

6.2.5 交叉

2. 修正的交叉方法

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

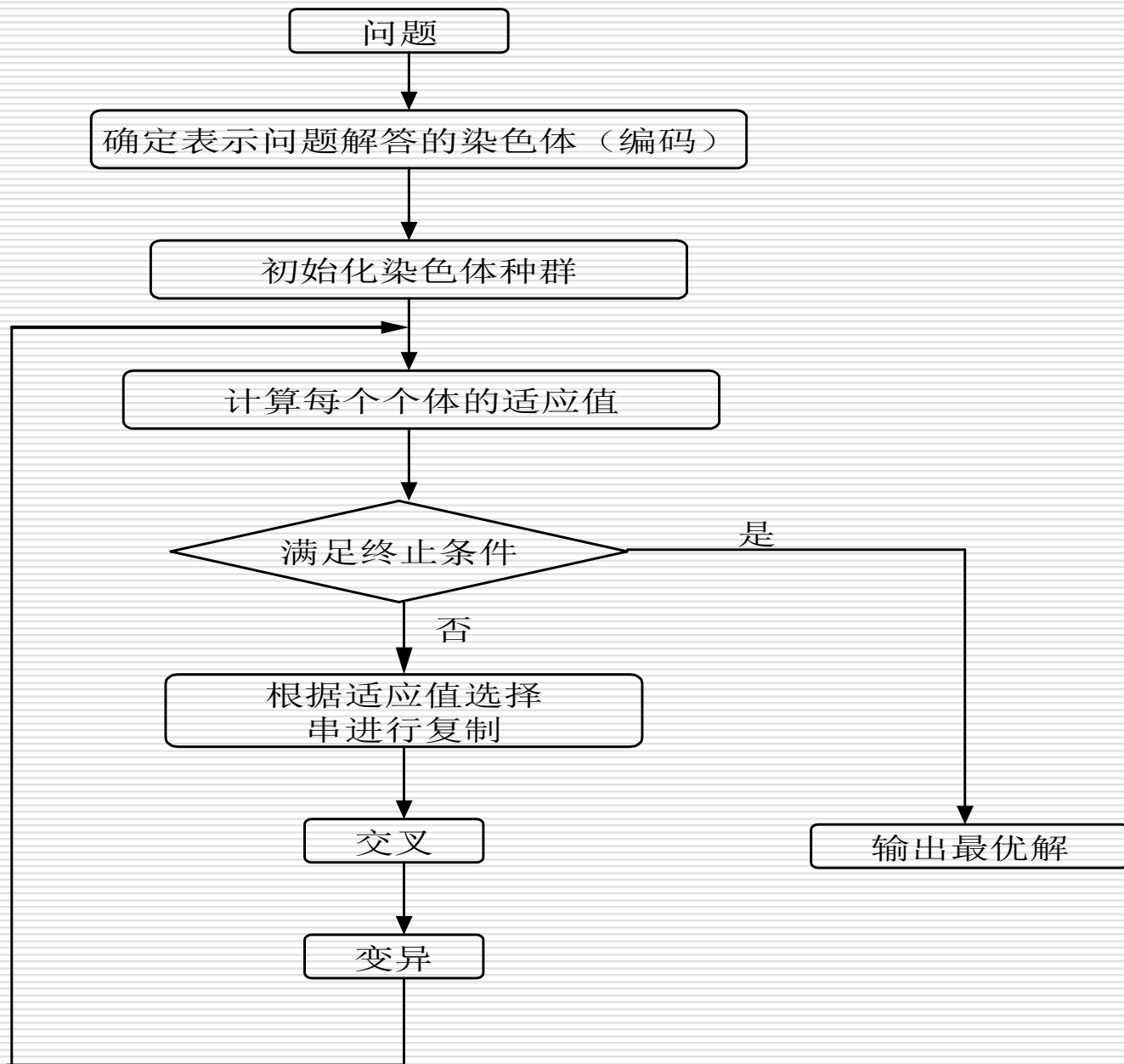


$A =$	9	8	4		5	6	7		1	3	2
$B =$	8	7	1		2	3	9		5	4	6
$A' =$	9	8	4		2	3	9		1	3	2
$B' =$	8	7	1		5	6	7		5	4	6
$A'' =$	7	8	4		2	3	9		1	6	5
$B'' =$	8	9	1		5	6	7		2	4	3

6.2.6 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

6.2.7 遗传算法的一般步骤



6.2.7 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

从 $pop(t)$ 中随机选择一些染色体构成一个新种群

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

6.2.7 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)

6.2.8 遗传算法的特点

- 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。

第 6 章 遗传算法及其应用

□ 6.1 遗传算法的产生与发展

□ 6.2 遗传算法的基本算法

□ 6.4 遗传算法的应用

第 6 章 遗传算法及其应用

□ 6.1 遗传算法的产生与发展

□ 6.2 遗传算法的基本算法

✓ 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

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

- 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。

6.4 遗传算法的应用

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, $\{j_1, j_2, \dots, j_n\}$: 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:

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

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

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

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

最大流程时间: $c_{\max} = c(j_n, m)$

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

6.4 遗传算法的应用

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 。

6.4 遗传算法的应用

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 : k 个染色体 v_k 的最大流程时间,

FSP 的适应度函数:

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

6.4 遗传算法的应用

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

果

用穷举法求得最优解： 4-2-5-1-3 ， 加工时间： 213 ；

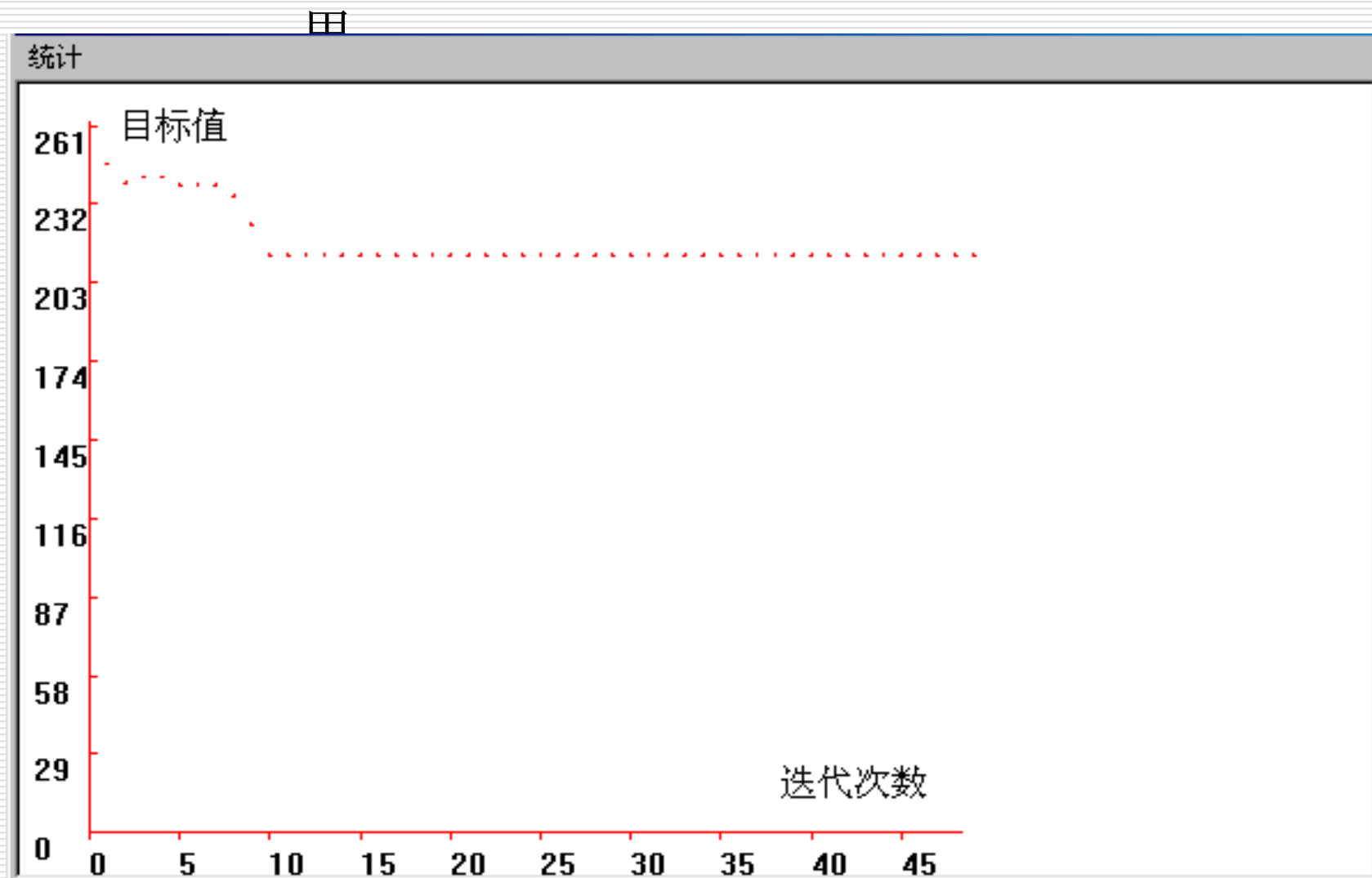
最劣解： 1-4-2-3-5 ， 加工时间： 294 ； 平均解的加工时间： 265 。

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

遗传算法运行的结果

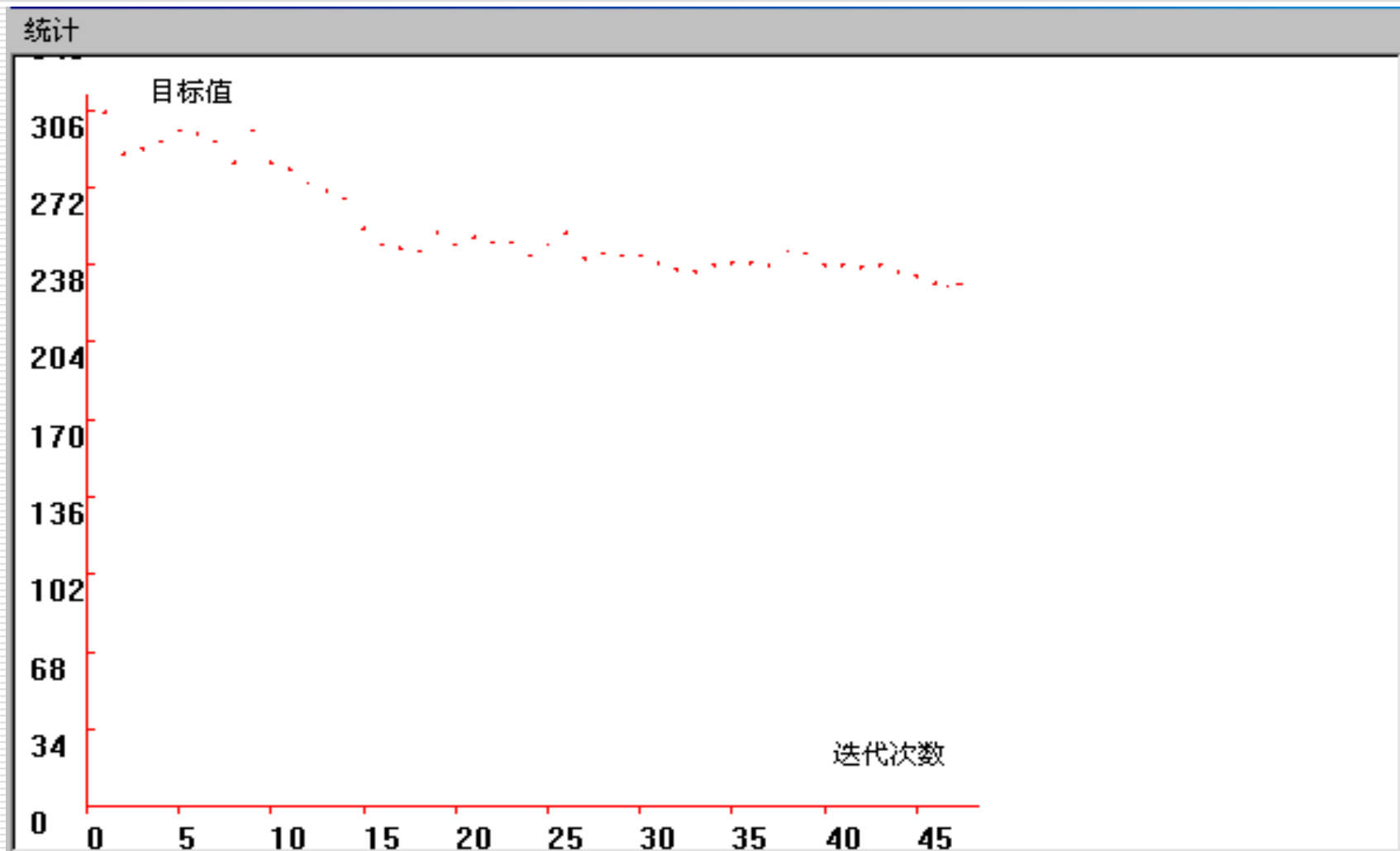
总运行 次数	最好解	最坏解	平均	最好解 的频率	最好解的 平均代数
20	213	221	213.95	0.85	12

6.4 遗传算法的应用



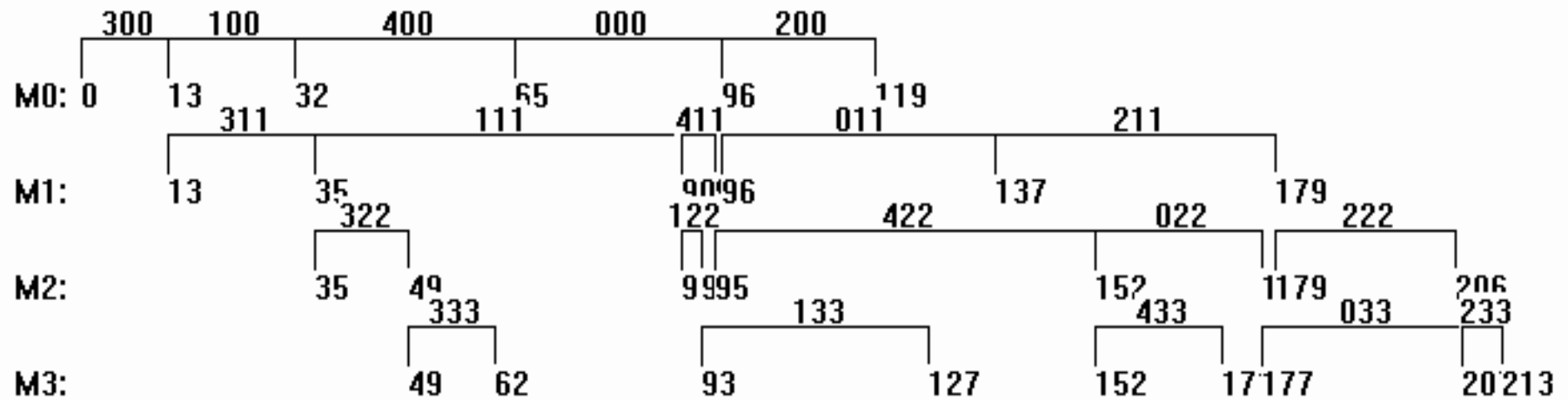
最优解收敛图

6.4 遗传算法的应用



平均值收敛图

6.4 遗传算法的应用



机器甘特图

遗传算法

□ 示例

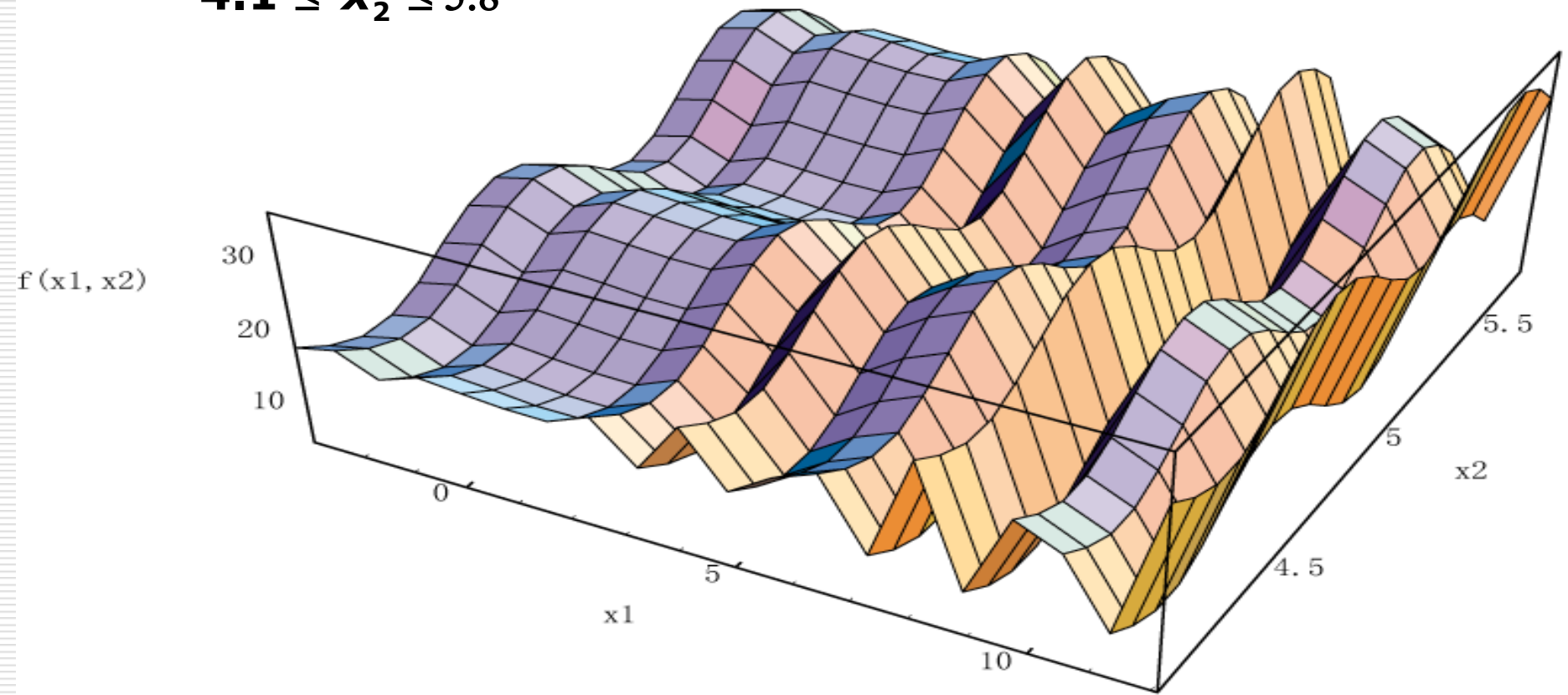
$$\max \quad f(x_1, x_2) = 21.5 + x_1 \cdot \sin(4\pi x_1) + x_2 \cdot \sin(20\pi x_2)$$

$$\text{s. t.} \quad -3.0 \leq x_1 \leq 12.1$$

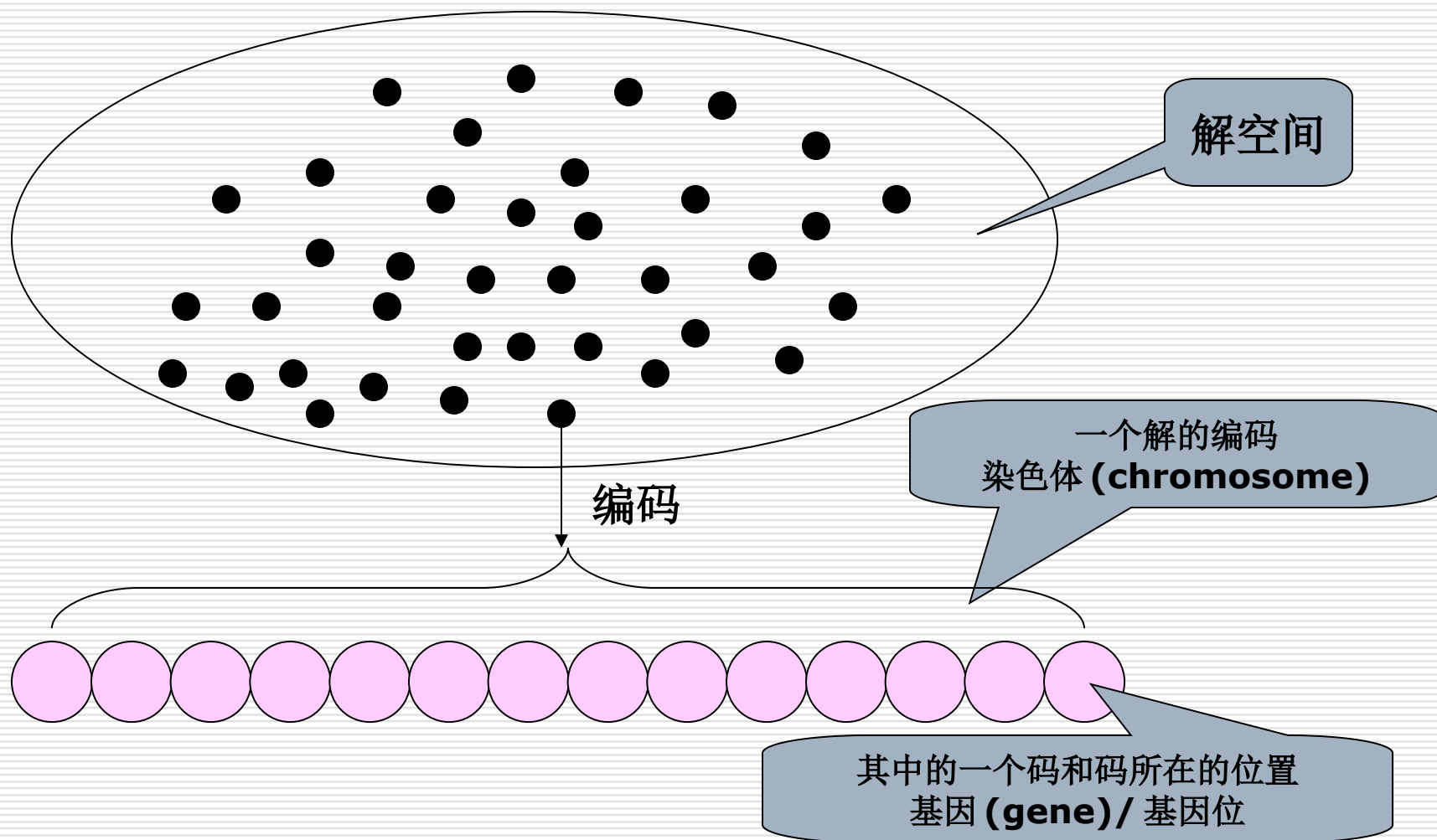
$$4.1 \leq x_2 \leq 5.8$$

遗传算法

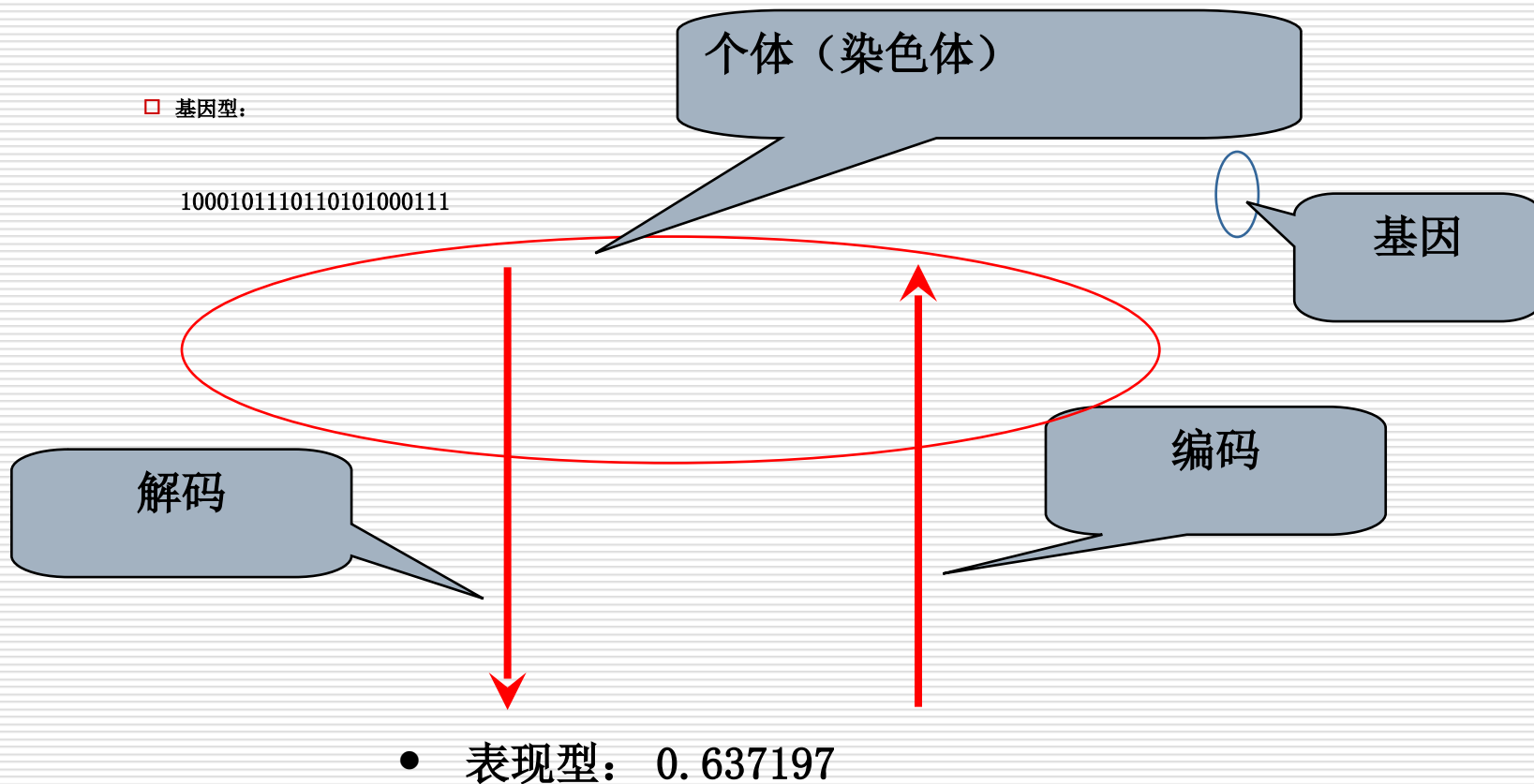
$$\begin{aligned} \max \quad & f(x_1, x_2) = 21.5 + x_1 \cdot \sin(4\pi x_1) + x_2 \cdot \sin(20\pi x_2) \\ \text{s. t.} \quad & -3.0 \leq x_1 \leq 12.1 \\ & 4.1 \leq x_2 \leq 5.8 \end{aligned}$$



遗传算法



遗传算法



遗传算法

二进制串表示

➤ 精度保留到小数点后 **5** 位 . $\mathbf{x}_j \in [\mathbf{a}_j, \mathbf{b}_j]$ （变量 \mathbf{x} 的第 \mathbf{j} 个分量）

➤ 精度对应的整数范围为 $(\mathbf{b}_j - \mathbf{a}_j) \times 10^5$

➤ 所需字节个数 \mathbf{m}_j 计算如下：

$$2^{m_j-1} < (b_j - a_j) \times 10^5 \leq 2^{m_j} - 1$$

➤ 从二进制串转换为实值

$$x_j = a_j + \text{decimal}(\text{substring}_j) \times \frac{b_j - a_j}{2^{m_j} - 1}$$

遗传算法

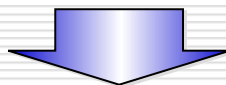
示例

$$x_1 : (12.1 - (-3.0)) \times 10,000 = 151,000$$

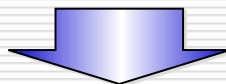
$$2^{17} < 151,000 \leq 2^{18}, \quad m_1 = 18 \text{ bits}$$

$$x_2 : (5.8 - 4.1) \times 10,000 = 17,000$$

$$2^{14} < 17,000 \leq 2^{15}, \quad m_2 = 15 \text{ bits}$$



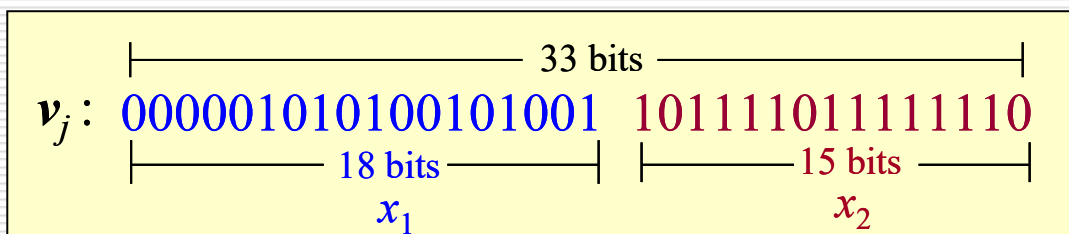
$$m = m_1 + m_2 = 18 + 15 = 33 \text{ bits}$$



$$v_j : \begin{array}{|c|c|} \hline \text{33 bits} & \\ \hline 000001010100101001 & 101111011111110 \\ \hline \text{18 bits} & \text{15 bits} \\ x_1 & x_2 \\ \hline \end{array}$$

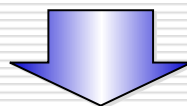
遗传算法

示例



	Binary Number	Decimal Number
x_1	000001010100101001	5417
x_2	101111011111110	24318

$$x_j = a_j + \text{decimal}(\text{substring}_j) \times \frac{b_j - a_j}{2^{m_j} - 1}$$



$$\begin{aligned} x_1 &= -3.0 + 5417 \times \frac{12.1 - (-3.0)}{2^{18} - 1} \\ &= -2.687069 \end{aligned}$$

$$\begin{aligned} x_2 &= 4.1 + 24318 \times \frac{5.8 - 4.1}{2^{15} - 1} \\ &= 5.361653 \end{aligned}$$

遗传算法—初始化

$$v_1 = [\textcolor{red}{000001010100101001}\textcolor{blue}{101111011111110}] = [x_1 \ x_2] = [-2.687969 \ 5.361653]$$

$$v_2 = [001110101110011000000010101001000] = [x_1 \ x_2] = [0.474101 \ 4.170144]$$

$$v_3 = [111000111000001000010101001000110] = [x_1 \ x_2] = [10.419457 \ 4.661461]$$

$$v_4 = [100110110100101101000000010111001] = [x_1 \ x_2] = [6.159951 \ 4.109598]$$

$$v_5 = [000010111101100010001110001101000] = [x_1 \ x_2] = [-2.301286 \ 4.477282]$$

$$v_6 = [1111101010110110000000010110011001] = [x_1 \ x_2] = [11.788084 \ 4.174346]$$

$$v_7 = [110100010011111000100110011101101] = [x_1 \ x_2] = [9.342067 \ 5.121702]$$

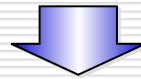
$$v_8 = [001011010100001100010110011001100] = [x_1 \ x_2] = [-0.330256 \ 4.694977]$$

$$v_9 = [111110001011101100011101000111101] = [x_1 \ x_2] = [11.671267 \ 4.873501]$$

遗传算法一个体评价

$$\begin{aligned} eval(v_k) &= f(x_i) & (k = 1, 2, \dots, popSize) \\ & & (i = 1, 2, \dots, n) \end{aligned}$$

$$f(x_1, x_2) = 21.5 + x_1 \cdot \sin(4\pi x_1) + x_2 \cdot \sin(20\pi x_2)$$



Example: $(x_1 = -2.687969, x_2 = 5.361653)$

$$\begin{aligned} eval(v_1) &= f(-2.687969, 5.361653) \\ &= 19.805119 \end{aligned}$$

遗传算法一个体评价

$$eval(v_1) = f(-2.687969, 5.361653) = 19.805119$$

$$eval(v_2) = f(0.474101, 4.170144) = 17.370896$$

$$eval(v_3) = f(10.419457, 4.661461) = 9.590546$$

$$eval(v_4) = f(6.159951, 4.109598) = 29.406122$$

$$eval(v_5) = f(-2.301286, 4.477282) = 15.686091$$

$$eval(v_6) = f(11.788084, 4.174346) = 11.900541$$

$$eval(v_7) = f(9.342067, 5.121702) = 17.958717$$

$$eval(v_8) = f(-0.330256, 4.694977) = 19.763190$$

$$eval(v_9) = f(11.671267, 4.873501) = 26.401669$$

遗传算法一个体选择

input: population $P(t-1)$, $C(t-1)$

output: population $P(t)$, $C(t)$

step 1: 计算群体总适应度 $F = \sum_{k=1}^{popSize} eval(v_k)$

step 2: 计算个体适应度比率 p_k

$$p_k = \frac{eval(v_k)}{F}, \quad k = 1, 2, \dots, popSize$$

step 3: 计算个体选择概率 q_k

$$q_k = \sum_{j=1}^k p_j, \quad k = 1, 2, \dots, popSize$$

step 4: 生成随机数 r in $[0, 1]$

step 5: If $r \leq q_1$, 则选择第一个染色体 v_1 ;

Else 选择第 k 个染色体 v_k ($2 \leq k \leq popSize$) S.T.

$$q_{k-1} < r \leq q_k.$$

遗传算法一个体选择

step 1: 计算群体总适应度

$$F = \sum_{k=1}^{10} eval(v_k) = 178.135372$$

step 2: 计算个体适应度比率 p_k

$$\begin{aligned} p_1 &= 0.111180, p_2 = 0.097515, p_3 = 0.053839, p_4 = 0.165077, \\ p_5 &= 0.088057, p_6 = 0.066806, p_7 = 0.100815, p_8 = 0.110945, \\ p_9 &= 0.148211, p_{10} = 0.057554 \end{aligned}$$

step 3: 计算个体选择概率 q_k

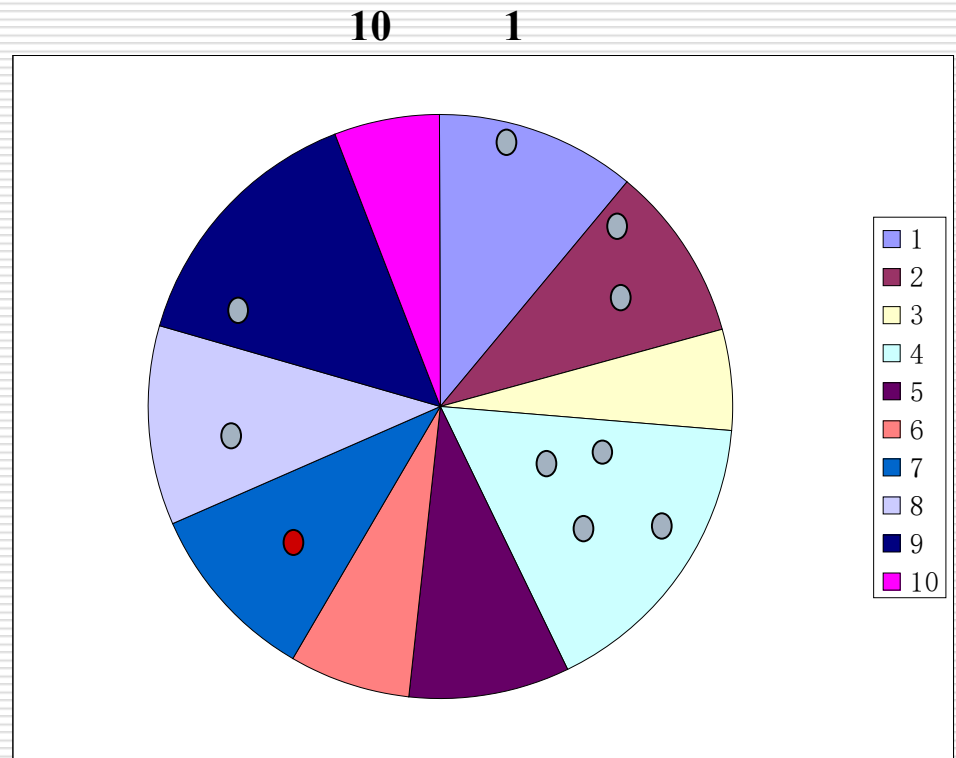
$$\begin{aligned} q_1 &= 0.111180, q_2 = 0.208695, q_3 = 0.262534, q_4 = 0.427611, \\ q_5 &= 0.515668, q_6 = 0.582475, q_7 = 0.683290, q_8 = 0.794234, \\ q_9 &= 0.942446, q_{10} = 1.000000 \end{aligned}$$

step 4: 生成随机数 r in $[0, 1]$

0.301431. 0.322062. 0.766503. 0.881893. 0.350871,
0.583392, 0.177618, 0.343242, 0.032685, 0.197577

遗传算法一个体选择

$q_1 = 0.111180$, $q_2 = 0.208695$, $q_3 = 0.262534$, $q_4 = 0.427611$,
 $q_5 = 0.515668$, $q_6 = 0.582475$, $q_7 = 0.683290$, $q_8 = 0.794234$,
 $q_9 = 0.942446$, $q_{10} = 1.000000$



0.301431, 0.322062, 0.766503, 0.881893, 0.350871,
0.583392, 0.177618, 0.343242, 0.032685, 0.197577

遗传算法一个体选择

step 5: $q_3 < r_1 = 0.301432 \leq q_4$, 选择染色体 v_4

$q_3 < r_2 = 0.322062 \leq q_4$, 选择 v_4

$v_1' = [1\ddot{0}\ddot{0}\ddot{1}\ddot{1}0110100101101000000010111001]$ (v_4)

$v_2' = [100110110100101101000000010111001]$ (v_4)

$v_3' = [001011010100001100010110011001100]$ (v_8)

$v_4' = [111110001011101100011101000111101]$ (v_9)

$v_5' = [100110110100101101000000010111001]$ (v_4)

$v_6' = [110100010011111000100110011101101]$ (v_7)

$v_7' = [0011101011100110000000010101001000]$ (v_2)

$v_8' = [100110110100101101000000010111001]$ (v_4)

$v_9' = [000001010100101001101111011111110]$ (v_1)

遗传算法—交叉

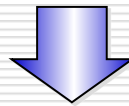
□ Crossover (One-cut point Crossover)

■ 示例: cut point = 17

crossing point at 17th gene

$v_1 = [100110110100101101000000010111001]$

$v_2 = [001110101110011000000010101001000]$



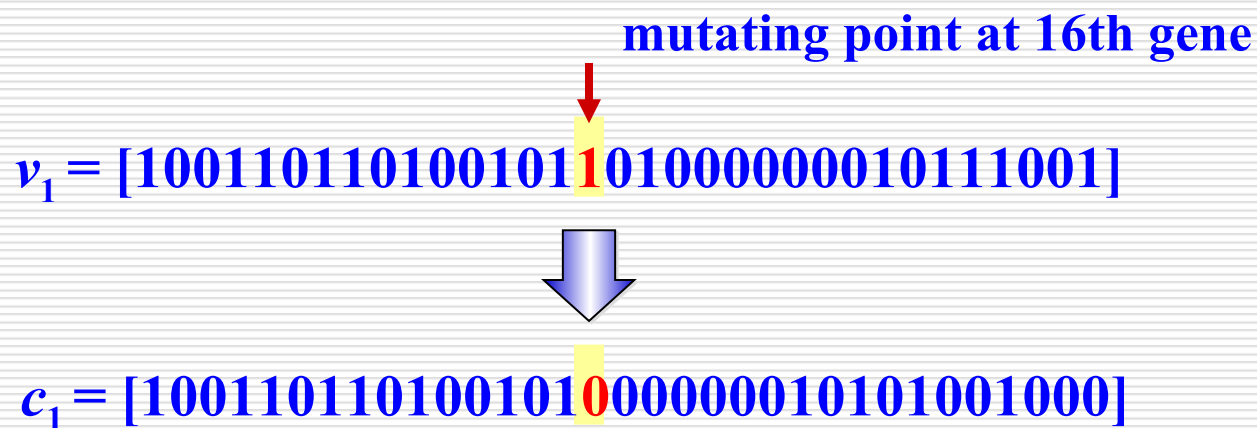
$c_1 = [100110110100101100000010101001000]$

$c_2 = [00111010111001100100000010111001]$

遗传算法—变异

□ Mutation

■ 示例: 染色体中第16位被选中



遗传算法

□ Next Generation

$$v_1' = [100110110100101101000000010111001], \quad f(6.159951, 4.109598) = 29.406122$$

$$v_2' = [100110110100101101000000010111001], \quad f(6.159951, 4.109598) = 29.406122$$

$$v_3' = [001011010100001100010110011001100], \quad f(-0.330256, 4.694977) = 19.763190$$

$$v_4' = [111110001011101100011101000111101], \quad f(11.907206, 4.873501) = 5.702781$$

$$v_5' = [100110110100101101000000010111001], \quad f(8.024130, 4.170248) = 19.91025$$

$$v_6' = [110100010011111000100110011101101], \quad f(9.34067, 5.121702) = 17.958717$$

$$v_7' = [100110110100101101000000010111001], \quad f(6.159951, 4.109598) = 29.406122$$

$$v_8' = [100110110100101101000000010111001], \quad f(6.159951, 4.109598) = 29.406122$$

$$v_9' = [000001010100101001101111011111110], \quad f(-2.687969, 5.361653) = 19.805199$$

遗传算法

$$\max f(x_1, x_2) = 21.5 + x_1 \cdot \sin(4\pi x_1) + x_2 \cdot \sin(20\pi x_2)$$

$$\text{s. t. } -3.0 \leq x_1 \leq 12.1$$

$$4.1 \leq x_2 \leq 5.8$$

$$\begin{aligned} eval(v^*) &= f(11.622766, 5.624329) \\ &= 38.737524 \end{aligned}$$

$$x_1^* = 11.622766$$

$$x_2^* = 5.624329$$

$$f(x_1^*, x_2^*) = 38.737524$$

□ Evolutional Process

$maxGen: 1000$ $p_C: 0.25$ $p_M: 0.01$

