

第 6 章 进化算法及其应用

教材：

王万良 《人工智能及其应用》（第3版）

高等教育出版社，2016.2

第6章 进化算法及其应用

- 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 基于遗传算法的生产调度方法
- 6.5 差分进化算法及其应用
- 6.6 量子进化算法及其应用

第6章 进化算法及其应用

- 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 基于遗传算法的生产调度方法
- 6.5 差分进化算法及其应用
- 6.6 量子进化算法及其应用

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 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

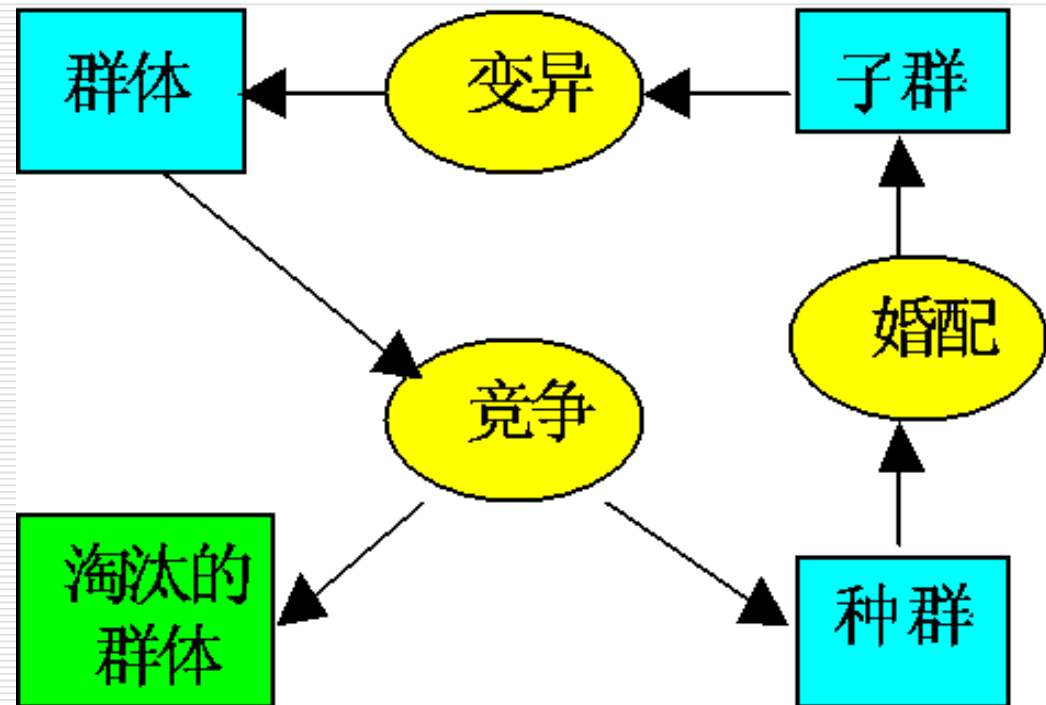
- **适者生存**：最适合自然环境的群体往往产生了更大的后代群体。
- **生物进化的基本过程**：

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。



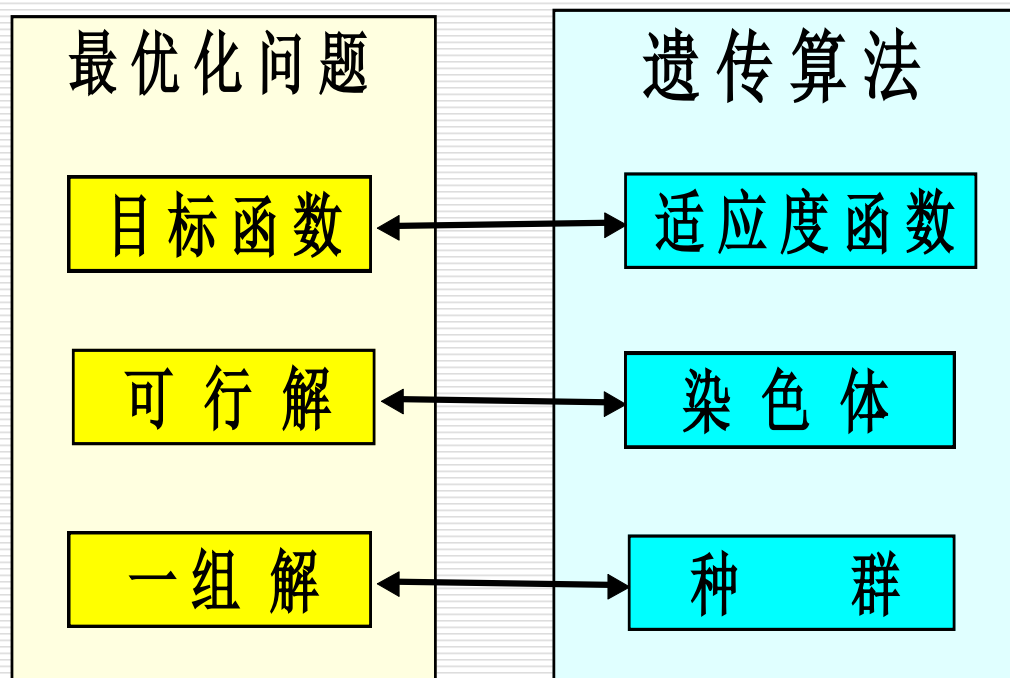
第6章 进化算法及其应用

- 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 基于遗传算法的生产调度方法
- 6.5 差分进化算法及其应用
- 6.6 量子进化算法及其应用

6.2.1 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)	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。

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。

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

6.2.3 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。

③ 求解高维优化问题的二进制编码串长，算法的搜索效率低。

6.2.3 编码

1. 位串编码

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

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

二进制编码 \rightarrow Gray编码

Gray编码 \rightarrow 二进制编码

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

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

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。
- **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

(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。

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




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

6.2.5 适应度函数

1. 将目标函数映射成适应度函数的方法(续)

 存在界限值预选估计困难或者不能精确估计的问题！

- 若目标函数为**最大化**问题，则

$$Fit(f(x)) = \frac{1}{1 + c + f(x)} \quad c \geq 0, \quad c + f(x) \geq 0$$

- 若目标函数为**最小化**问题，则

$$Fit(f(x)) = \frac{1}{1 + c - f(x)} \quad c \geq 0, \quad c - f(x) \geq 0$$

- c : 目标函数界限的保守估计值。

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。
- 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。

6.2.5 适应度函数

2. 适应度函数的尺度变换(续)

(1) 线性变换:

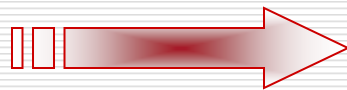
$$f' = af + b$$

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



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

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

6.2.5 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:

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

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, \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

6.2.6 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：

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

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

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

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

6.2.6 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**

6.2.6 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。

6.2.6 选择

2. 选择个体方法

(3) (μ, λ) 和 $\mu + \lambda$ 选择

(μ, λ) 选择

- 从规模为 μ 的群体中随机选取个体通过重组和变异生成 $\lambda (\geq \mu)$ 个后代,
- 再选取 μ 个最优的后代作为新一代种群。

$\mu + \lambda$ 选择

- 从 $\lambda (\geq \mu)$ 个后代与其父体共 $\mu + \lambda$ 中选取 μ 个最优的后代。

6.2.6 选择

2. 选择个体方法

(4) Boltzmann锦标赛选择

- 随机选取两个个体 x_1, x_2 ，若 $|f(x_1) - f(x_2)| \geq \theta$ 则选择适应值好的作为胜者，否则计算概率 $p = \exp[-|f(x_1) - f(x_2)|/T]$ ，若 $p > \text{random}[0,1)$ ，选择差解，否则选择好解。

(5) 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。

6.2.7 交叉

1. 基本的交叉算子

(1) 单点交叉 (single-point crossover)

- 单点交叉：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) 两点交叉 (two-point crossover)

- 两点交叉：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

1. 基本的交叉算子（续）


（3）均匀交叉（uniform crossover）或一致交叉

- 均匀交叉：按照均匀概率抽取一些位，每一位是否被选取都是随机的，并且独立于其他位。然后将两个个体被抽取位互换组成两个新个体。

6.2.7 交叉

2. 修正的交叉方法

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

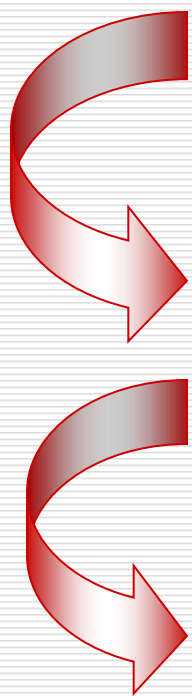


$A =$	9	8	4		5	6	7		1	3	2
$B =$	8	7	1		2	3	9		5	4	6
$A' =$	9	8	4		2	3	9		1	3	2
$B' =$	8	7	1		5	6	7		5	4	6

6.2.7 交叉

2. 修正的交叉方法(续)

(2) 顺序交叉OX: Davis L. (1985)



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

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

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

(3) 循环交叉CX: Smith D. (1985)

6.2.7 交叉

3. 实数编码的交叉方法

(1) 离散交叉 (discrete crossover)

- **部分离散交叉**：在父解向量中选择一部分分量，然后交换这些分量。 —— 二进制的点式交叉
- **整体离散交叉**：以0.5的概率交换父体 s_1 与 s_2 的所有分量。 —— 二进制编码的均匀性交叉

6.2.7 交叉

3. 实数编码的交叉方法（续）

（2）算术交叉（arithmetical crossover）

■ **部分算术**：先在父解向量中选择一部分分量，如第 k 个分量以后的所有分量，然后生成 $n-k$ 个 $[0, 1]$ 区间的随机数，并将两个后代定义为：

$$s_z = (v_1^{(1)}, \dots, v_k^{(1)}, a_{k+1} v_{k+1}^{(1)} + (1 - a_{k+1}) v_{k+1}^{(2)}, \dots, a_n v_n^{(1)} + (1 - a_n) v_n^{(2)})$$

$$s_w = (v_1^{(2)}, \dots, v_k^{(2)}, a_{k+1} v_{k+1}^{(2)} + (1 - a_{k+1}) v_{k+1}^{(1)}, \dots, a_n v_n^{(2)} + (1 - a_n) v_n^{(1)})$$

$$a_{k+1} = \dots = a_n$$

6.2.7 交叉

3. 实数编码的交叉方法（续）

（2）算术交叉（arithmetical crossover）

■ **整体算术交叉**：先生成 n 个区间的随机数，则后代分别定

义为：
$$z_i = a_i v_i^{(1)} + (1 - a_i) v_i^{(2)} = v_i^{(2)} + a_i (v_i^{(1)} - v_i^{(2)})$$

$$w_i = a_i v_i^{(2)} + (1 - a_i) v_i^{(1)} = v_i^{(1)} + a_i (v_i^{(2)} - v_i^{(1)})$$

$$a_1 = a_2 = \dots = a_n$$

6.2.8 变异

1. 整数编码的变异方法

- (1) **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。
- (2) **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。
- (3) **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。

6.2.8 变异

1. 整数编码的变异方法（续）

- （4）**互换变异**：随机选取染色体的两个基因进行简单互换。
- （5）**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）**自适应变异**：类似于位点变异，但变异概率随群体中个体的多样性程度而自适应调整。

6.2.8 变异

2. 实数编码的变异方法

(1) 均匀性变异

$s = (v_1, v_2, \dots, v_k, \dots, v_n)$: 父解

$s' = (v_1, v_2, \dots, v'_k, \dots, v_n)$: 变异产生的后代

- 均匀性变异: 在父解向量中随机地选择一个分量 (第 k 个), 然后在 $[a_k, b_k]$ 中以均匀概率随机选择 v'_k 代替 v_k 以得到 s' , 即

$$s' = \begin{cases} v_i, & i \neq k \\ v'_k, & i = k \end{cases}$$

6.2.8 变异

2. 实数编码的变异方法（续）

(2) 正态性变异 (normal distributed mutation)

解向量 $s = (v_1, v_2, \dots, v_n)$ 摄动向量 $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$

被选个体 (s, σ)  新个体 (s', σ')

$$\sigma'_i = \sigma_i \exp(N_i(0, \Delta\sigma))$$

$$v'_i = v_i + N(0, \sigma'_i) \quad i = 1, 2, \dots, n$$

6.2.8 变异

2. 实数编码的变异方法（续）

（3）非一致性变异

- Z. Michalewicz首先提出将变异算子的结果与演化代数联系起来。
- 在演化初期，变异范围相对较大，而随着演化的推进，变异范围越来越小，起着一种对演化系统的微调作用。

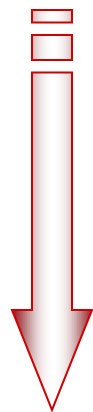
6.2.8 变异

2. 实数编码的变异方法（续）

(3) 非一致性变异

$s = (v_1, v_2, \dots, v_k, \dots, v_n)$: 父解

$[a_k, b_k]$: 区间



$$v'_k = \begin{cases} v_k + \Delta(t, b_k - v_k), & rnd(2) = 0 \\ v_k + \Delta(t, b_k - a_k), & rnd(2) = 1 \end{cases}$$

$$\Delta(t, y) = y \left(1 - r^{(1-t/T)^\lambda} \right)$$

$s' = (v_1, v_2, \dots, v_{k-1}, v'_k, \dots, v_n)$: 变异后的解

6.2.8 变异

2. 实数编码的变异方法（续）

(4) 自适应变异

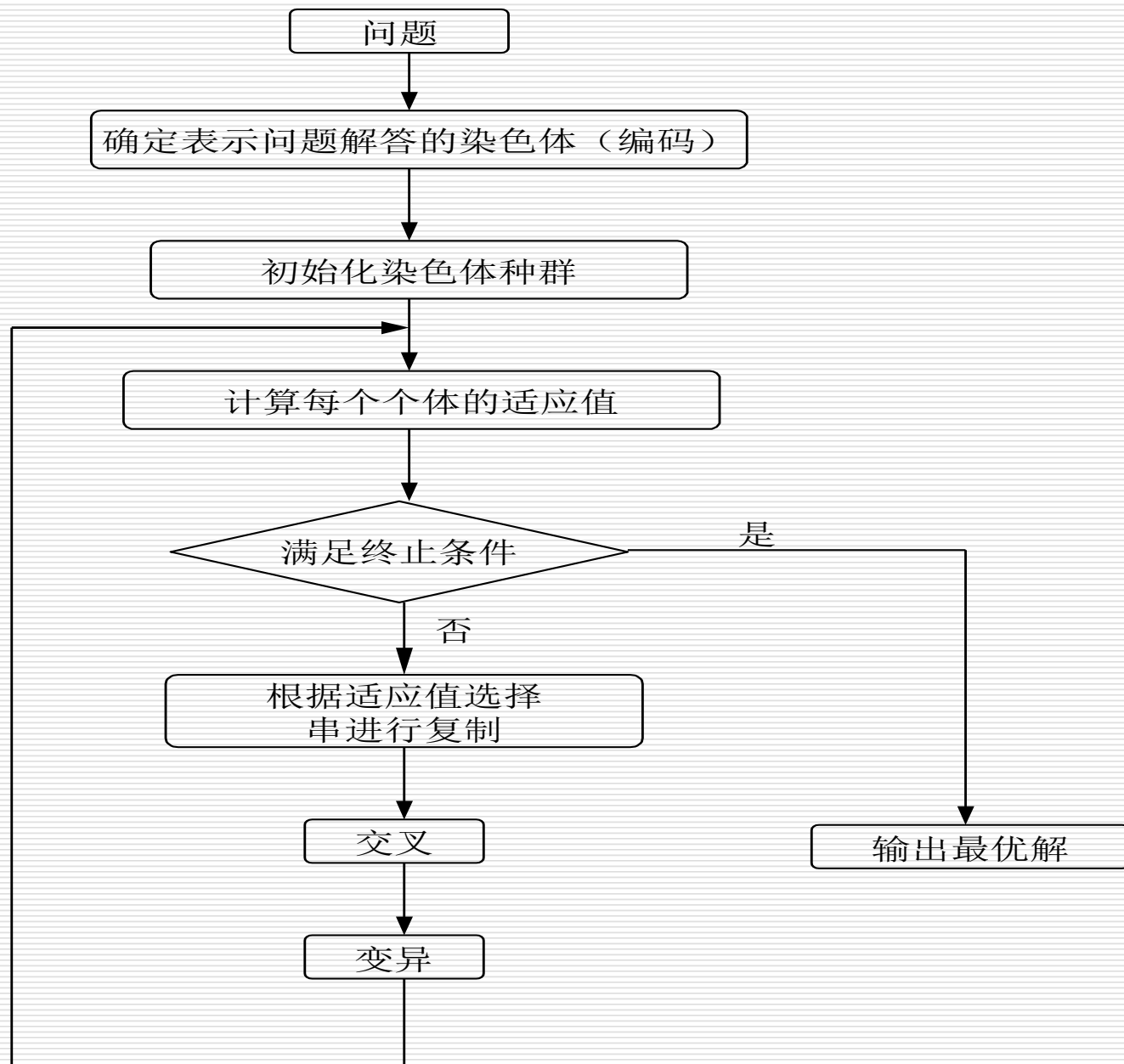
$s = (v_1, v_2, \dots, v_n)$: 解空间的一个向量

变异温度: $T = 1 - \frac{f(s)}{f_{\max}}$

- 自适应变异方式与非一致性变异算子相同，只是将其中的演化代数 t 改为 T ，函数表达式变为：

$$\Delta(T, y) = y(1 - r^{T^\lambda})$$

6.2.10 遗传算法的一般步骤



6.2.10 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

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

6.2.10 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)。

6.2.11 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
- ✓ 6.3 遗传算法的改进算法
- 6.4 基于遗传算法的生产调度方法
- 6.5 差分进化算法及其应用
- 6.6 量子进化算法及其应用

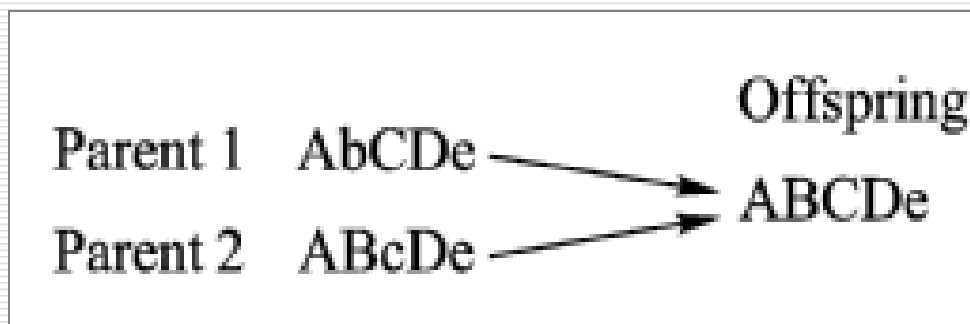
6.3 遗传算法的改进算法

- 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

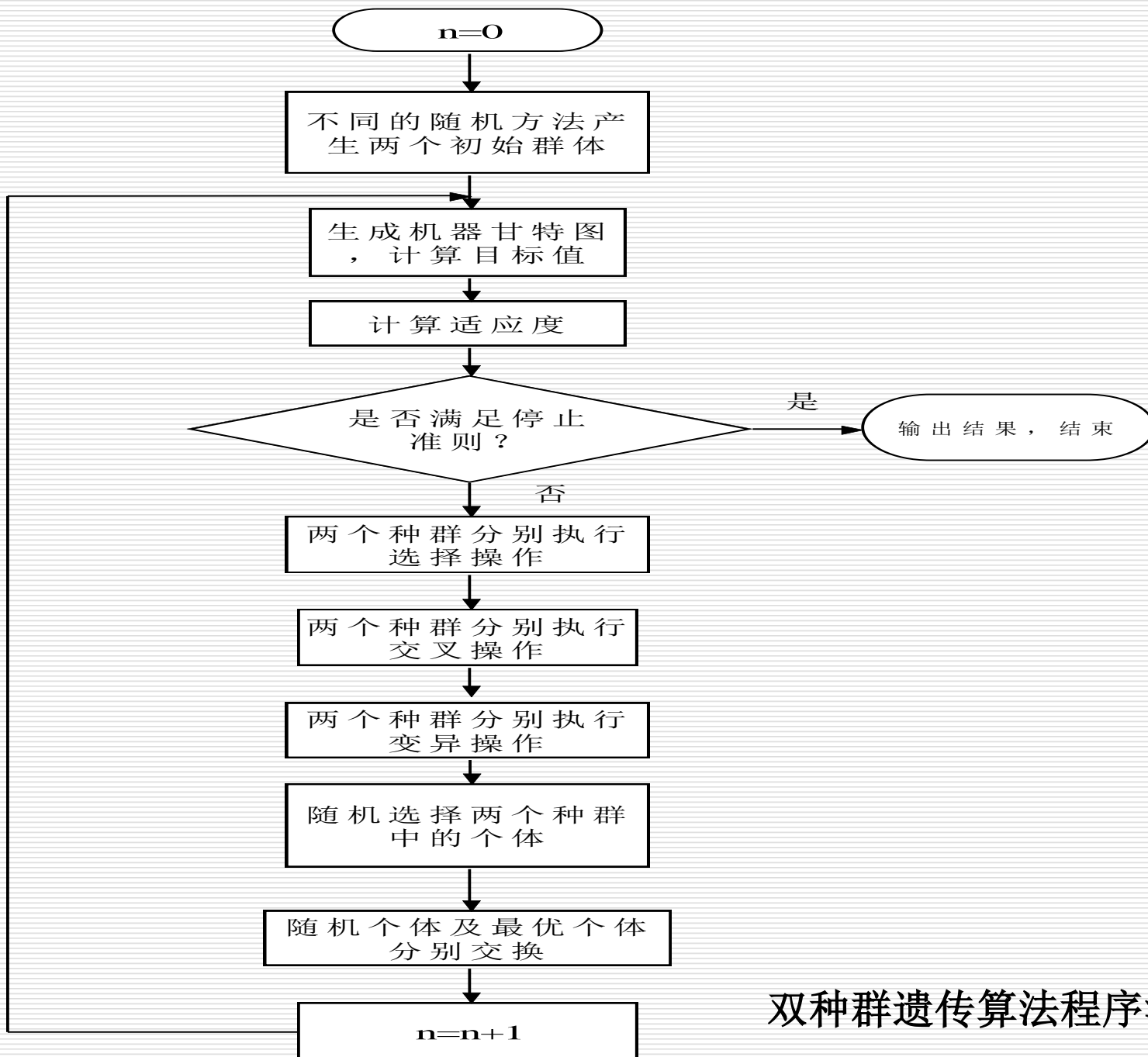
- 在遗传算法中使用多种群同时进化，并交换种群之间优秀个体所携带的遗传信息，以打破种群内的平衡态达到更高的平衡态，有利于算法跳出局部最优。
- **多种群遗传算法**：建立两个遗传算法群体，分别独立地运行复制、交叉、变异操作，同时当每一代运行结束以后，选择两个种群中的随机个体及最优个体分别交换。

6.3.2 双种群遗传算法

2. 双种群遗传算法的设计

- (1) 编码/解码设计
- (2) 交叉算子、变异算子
- (3) 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

■ Srinivas M., Patnaik L. M.等在1994年提出一种自适应遗传算法(adaptive genetic algorithms, AGA): P_c 和 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) 定义适应度函数为 $f = 1/ob$ ，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 $R < P_m$ 则对该染色体进行交叉操作。
- (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

$$P_c = \begin{cases} \frac{k_1(f_{\max} - f')}{f_{\max} - f_{\text{avg}}}, & f' > f_{\text{avg}} \\ k_2, & f' \leq f_{\text{avg}} \end{cases} \quad P_m = \begin{cases} \frac{k_3(f_{\max} - f)}{f_{\max} - f_{\text{avg}}}, & f > f_{\text{avg}} \\ k_4, & f \leq f_{\text{avg}} \end{cases}$$

■ 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。

■ 改进的思想：当前代的最优个体不被破坏，仍然保留（最优保存策略）；但较优个体要对应于更高的交叉概率与变异概率。

6.3.3 自适应遗传算法

3. 自适应的交叉概率与变异概率（续）

■ F—自适应方法：

当 $f' = f_{\max}$, $P_c = P_{c2} > 0$;

当 $f = f_{\max}$, $P_m = P_{m2} > 0$ 。

$$P_c = \begin{cases} P_{c1} - \frac{(P_{c1} - P_{c2})(f' - f_{\text{avg}})}{f_{\max} - f_{\text{avg}}}, & f' > f_{\text{avg}} \\ P_{c1}, & f' \leq f_{\text{avg}} \end{cases}$$

$$P_m = \begin{cases} P_{m1} - \frac{(P_{m1} - P_{m2})(f_{\max} - f)}{f_{\max} - f_{\text{avg}}}, & f > f_{\text{avg}} \\ P_{m1}, & f \leq f_{\text{avg}} \end{cases}$$

$$P_{c1} = 0.9, P_{c2} = 0.6, P_{m1} = 0.1, P_{m2} = 0.001$$

6.3.3 自适应遗传算法

3. 自适应的交叉概率与变异概率（续）

■ S—自适应方法：

$$P_c = \begin{cases} k_1 \sin\left(\frac{\pi}{2} \cdot \frac{f_{\max} - f'}{f_{\max} - f_{\text{avg}}}\right), & f' > f_{\text{avg}} \\ k_2, & f' \leq f_{\text{avg}} \end{cases}$$

$$P_m = \begin{cases} k_3 \sin\left(\frac{\pi}{2} \cdot \frac{f_{\max} - f}{f_{\max} - f_{\text{avg}}}\right), & f > f_{\text{avg}} \\ k_4, & f \leq f_{\text{avg}} \end{cases}$$

$$k_1 = 1.0, k_2 = 1.0, k_3 = 0.5, k_4 = 0.5,$$

6.3.3 自适应遗传算法

3. 自适应的交叉概率与变异概率（续）

■ C — 自适应方法：

$$P_c = \begin{cases} 1 - k_1 \cos\left(\frac{\pi}{2} \cdot \frac{f_{\max} - f'}{f_{\max} - f_{\text{avg}}}\right), & f' > f_{\text{avg}} \\ k_2, & f' \leq f_{\text{avg}} \end{cases}$$

$$P_m = \begin{cases} 1 - k_3 \cos\left(\frac{\pi}{2} \cdot \frac{f_{\max} - f}{f_{\max} - f_{\text{avg}}}\right), & f > f_{\text{avg}} \\ k_4, & f \leq f_{\text{avg}} \end{cases}$$

$$k_1 = 1.0, k_2 = 1.0, k_3 = 0.5, k_4 = 0.5,$$

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
- 6.3 遗传算法的改进算法
- ✓ 6.4 基于遗传算法的生产调度方法
- 6.5 差分进化算法及其应用
- 6.6 量子进化算法及其应用

6.4 基于遗传算法的生产调度方法

- 6.4.1 基于遗传算法的流水车间调度方法
- 6.4.2 基于遗传算法的混合流水车间调度方法

6.4.1 基于遗传算法的流水车间调度方法

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。

6.4.1 基于遗传算法的流水车间调度方法

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。

6.4.1 基于遗传算法的流水车间调度方法

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, $\{j_1, j_2, \Lambda, j_n\}$: 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 $\{j_1, j_2, \Lambda, j_n\}$ 使得 c_{\max} 最小

6.4.1 基于遗传算法的流水车间调度方法

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 。

6.4.1 基于遗传算法的流水车间调度方法

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:

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

6.4.1 基于遗传算法的流水车间调度方法

3. 求解FSP的遗传算法实例

例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

6.4.1 基于遗传算法的流水车间调度方法

用穷举法求得最优解：4-2-5-1-3，加工时间：213；

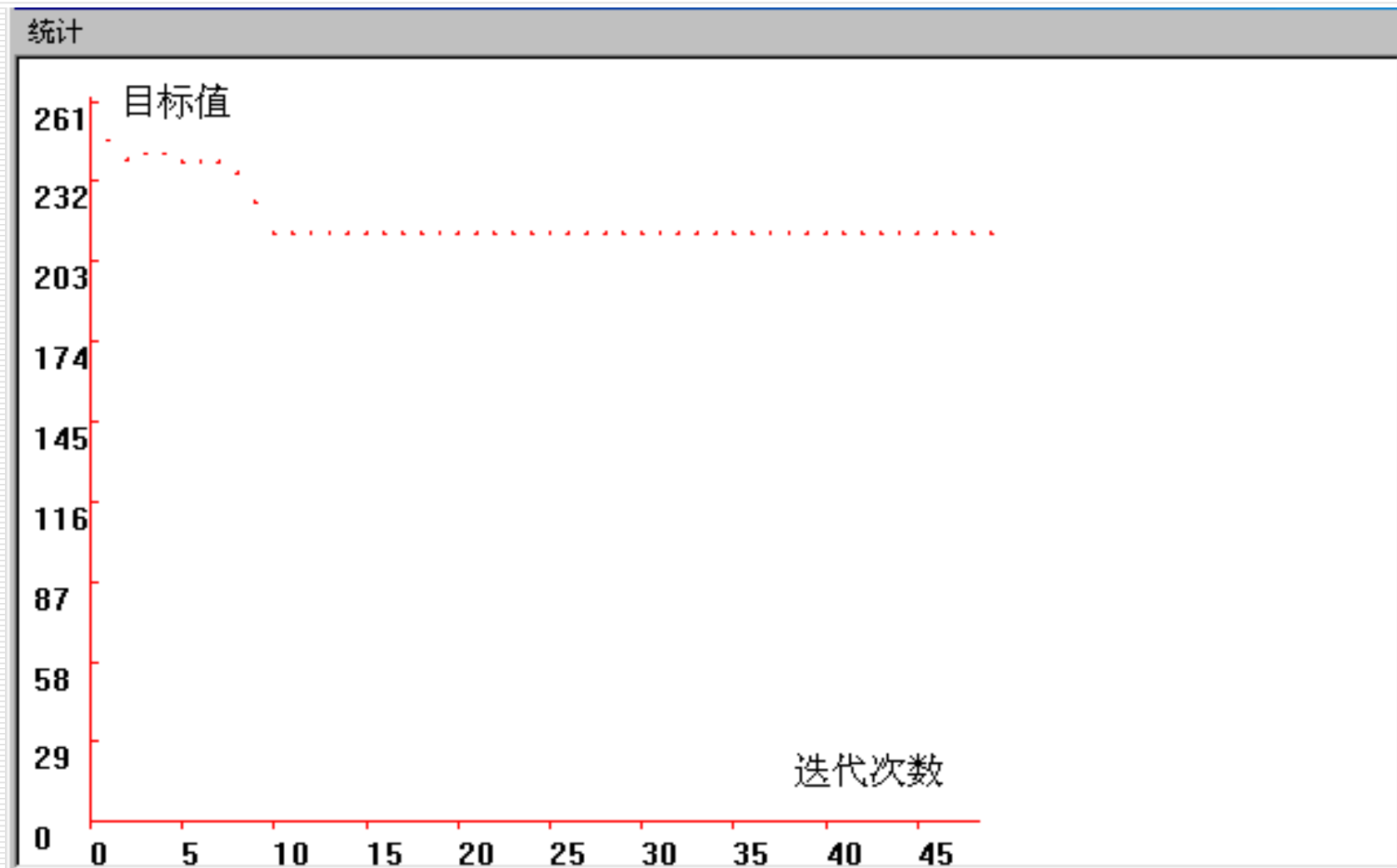
最劣解：1-4-2-3-5，加工时间：294；平均解的加工时间：265。

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

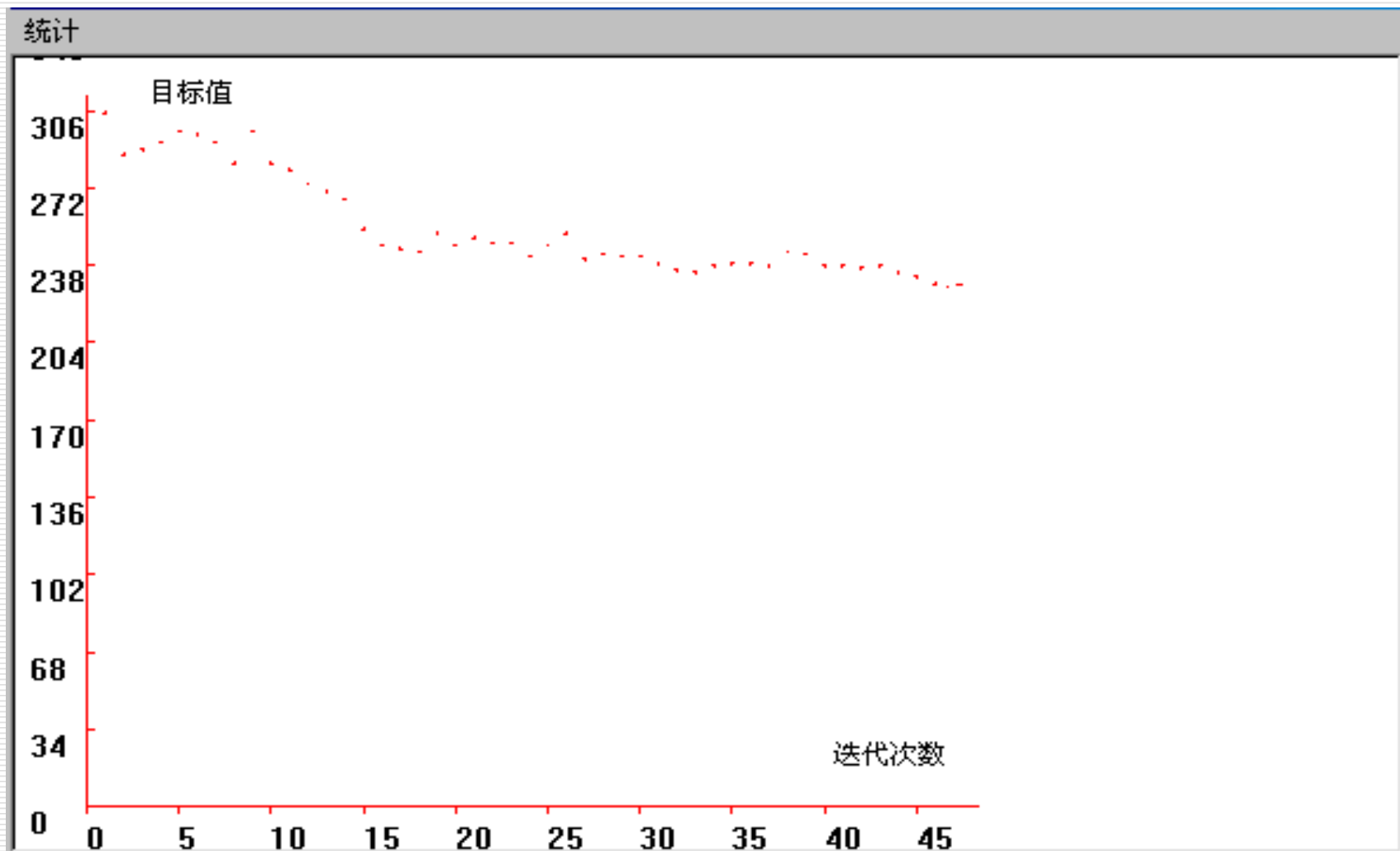
总运行次数	最好解	最坏解	平均	最好解的频率	最好解的平均代数
20	213	221	213.95	0.85	12

6.4.1 基于遗传算法的流水车间调度方法



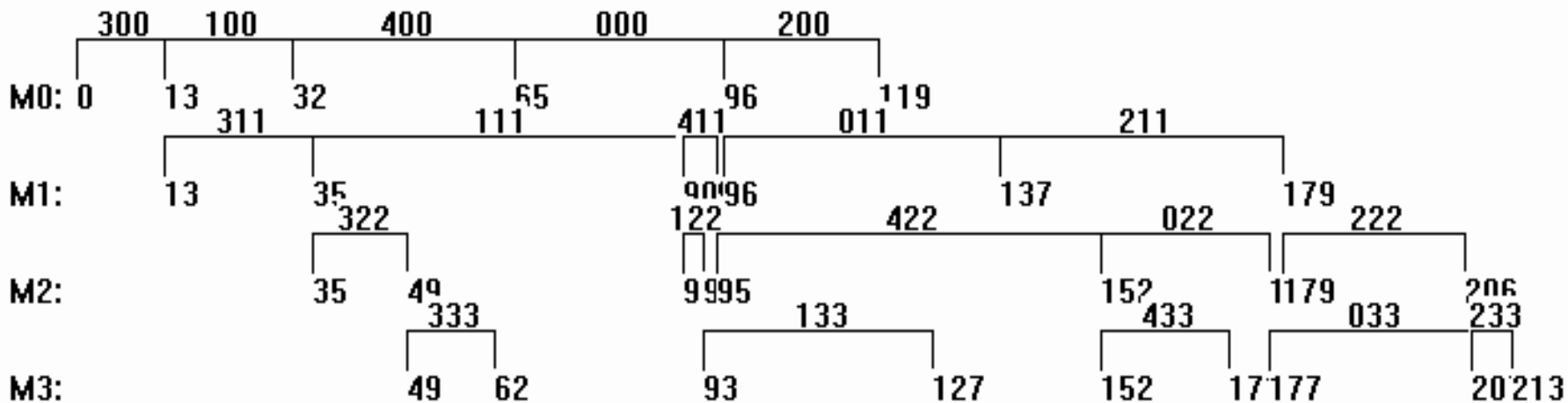
最优解收敛图

6.4.1 基于遗传算法的流水车间调度方法



平均值收敛图

6.4.1 基于遗传算法的流水车间调度方法



机器甘特图

6.4.2 基于遗传算法的混合流水车间调度方法

1. 混合流水车间调度问题

- 问题的特征：在某些工序上存在并行机器。
- **问题的描述**：需要加工多个工件，所有工件的加工路线都相同，都需要依次通过几道工序，在所有工序中至少有一个工序存在着多台并行机器。
- 需要解决的问题：确定并行机器的分配情况以及同一台机器上工件的加工排序。
- **目标**：最小化最大流程时间。

6.4.2 基于遗传算法的混合流水车间调度方法

2. 混合流水车间调度问题的遗传算法编码方法

■ 假设加工 N 个工件，每个工件都要依次经过 S 个加工工序，每个工序的并行机器数为 M_i ，($i=1, \Lambda, S$)。所有工序中至少有一个工序存在并行机，即至少有一个 M_i 大于1。

■ HFSP的编码矩阵

$$A_{S \times N} = \begin{bmatrix} a_{11} & a_{12} & \text{K} & a_{1N} \\ a_{21} & a_{22} & \text{K} & a_{2N} \\ \text{M} & \text{M} & & \text{M} \\ a_{S1} & a_{S2} & \text{K} & a_{SN} \end{bmatrix}$$

a_{ij} : $(1, M_i + 1)$ 上的一个实数，表示 j 工件的第 i 个工序在第 $Int(a_{ij})$ 台并行机上加工。

$Int(a_{ij}) = Int(a_{ik})$, $j \neq k$: 多个工件在同一台机器上加工同一个工序

6.4.2 基于遗传算法的混合流水车间调度方法

- $i = 1$, 按 a_{1j} 的升序来加工工件。
- $i > 1$, 根据每个工件的前一个工序的完成时间来确定其加工顺序, 前一个工序先完成的先加工。
- 假如完成时间相同, 也按 a_{ij} 的升序来加工。

■ 染色体:

$$Ind_k = [a_{11}, a_{12}, \dots, a_{1N}, 0, a_{21}, \dots, a_{2N}, 0, \dots, 0, a_{S1}, a_{S2}, \dots, a_{SN}]$$

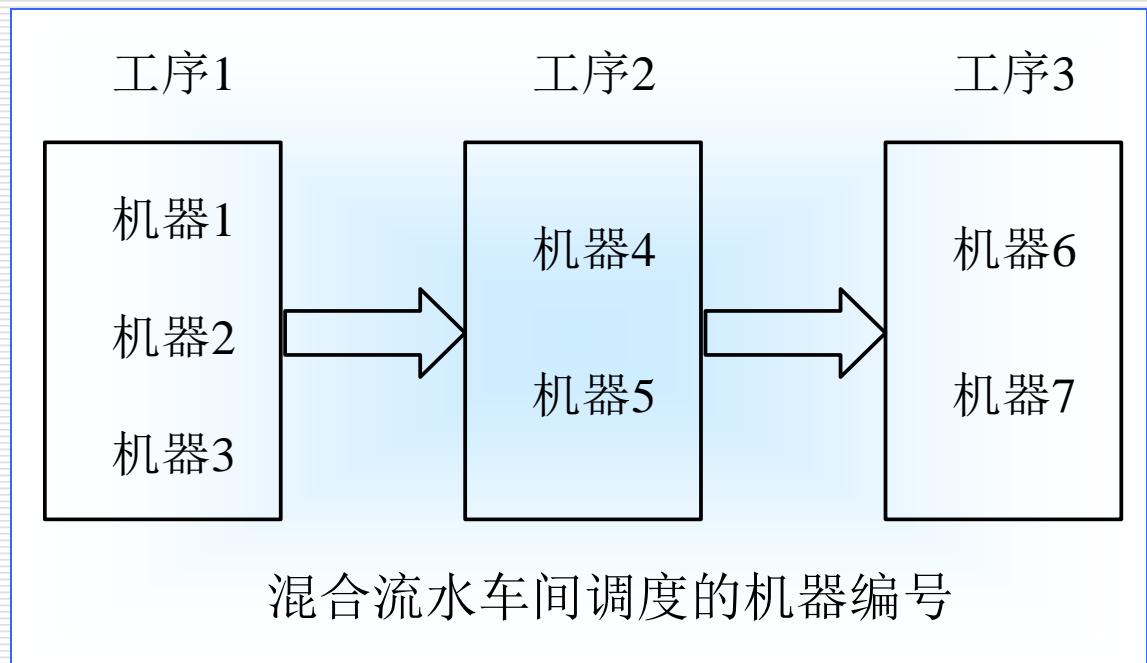
染色体的长度: $S \times N + N - 1$ 。

6.4.2 基于遗传算法的混合流水车间调度方法

- 例如，对于有**3**个工件、**3**道工序、各工序的并行机器数分别为**3**，**2**，**2**的混合流水车间调度问题。

■ 编码矩阵:

$$A = \begin{bmatrix} 2.1 & 2.4 & 1.9 \\ 1.6 & 2.1 & 2.3 \\ 1.1 & 2.4 & 1.2 \end{bmatrix}$$



■ 染色体:

[2.1, 2.4, 1.9, 0, 1.6, 2.1, 2.3, 0, 1.1, 2.4, 1.2]

9.4.2 基于遗传算法的混合流水车间调度方法

3. 基于遗传算法的求解方法

- (1) 初始群体的产生
- (2) 适应度函数的选择：最大流程时间的倒数
- (3) 选择（非线性排名策略）

①种群成员按适应值从好到坏依次排列 $f_1 > f_2 > \dots > f_N$

②按下式分配复制概率：

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

6.4.2 基于遗传算法的混合流水车间调度方法

(4) 交叉操作

分段交叉：在两个父体的各段中随机选取一部分基因，然后交换，得到子代个体。

(5) 变异操作：分段

(a) $d = Rand\{-1,1\}$

(b) 若 $d = 1$ ，则 $r = Rand(0, M_i - a_{ij})$ ，否则 $r = Rand(0, a_{ij})$

(c) $a'_{ij} = a_{ij} + d \times r$

6.4.2 基于遗传算法的混合流水车间调度方法

4. 调度实例

■ 某汽车发动机厂金加工车间要加工12个工件，每个工件都有车、刨、磨3个工序，现有3台车床，2台刨床，4台磨床，每台机床的加工能力不同。

6.4.2 基于遗传算法的混合流水车间调度方法

工件在每个机器上的加工时间

工件	工序1			工序2		工序3			
	机器1	机器2	机器3	机器4	机器5	机器6	机器7	机器8	机器9
1	2	2	3	4	5	2	3	2	3
2	4	5	4	3	4	3	4	5	4
3	6	5	4	4	2	3	4	2	5
4	4	3	4	6	5	3	6	5	8
5	4	5	3	3	1	3	4	6	5
6	6	5	4	2	3	4	3	9	5
7	5	2	4	4	6	3	4	3	5
8	3	5	4	7	5	3	3	6	4
9	2	5	4	1	2	7	8	6	5
10	3	6	4	3	4	4	8	6	7
11	5	2	4	3	5	6	7	6	5
12	6	5	4	5	4	3	4	7	5

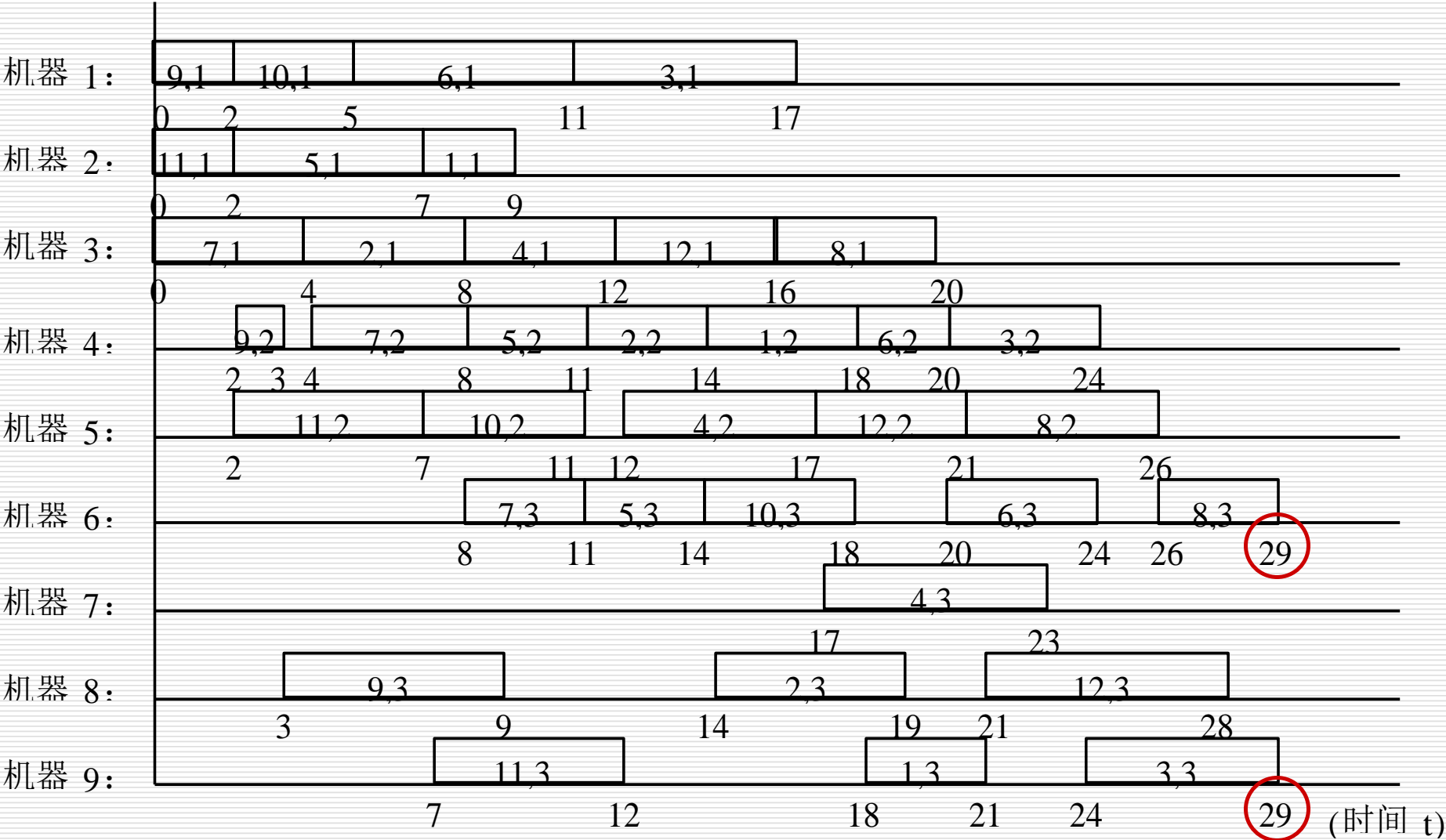
6.4.2 基于遗传算法的混合流水车间调度方法

算法中使用的参数为 $\alpha = 0.07$, $P_c = 0.80$, $P_m = 0.01$, 种群规模为30, 种群经过100代的进化, 目标函数最小值随着种群的进化逐渐地减小, 最后收敛于极值, 目标函数平均值也随着群体的进化逐渐减少, 最后趋近于最优值。

最好的染色体:

[2.77, 3.51, 1.74, 3.52, 2.42, 1.36, 3.28, 3.94, 1.09,
1.22, 2.24, 3.64, 0, 1.60, 1.13, 1.24, 2.97, 1.73, 1.88,
1.08, 2.68, 1.16, 2.69, 2.51, 2.96, 0, 4.99, 3.29, 4.95,
2.35, 1.10, 1.01, 1.73, 1.35, 3.06, 1.20, 4.13, 3.67]

6.4.2 基于遗传算法的混合流水车间调度方法



混合流水车间调度结果甘特图

第6章 进化算法及其应用

- 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 基于遗传算法的生产调度方法
- 6.5 差分进化算法及其应用
- 6.6 量子进化算法及其应用

6.5 差分进化算法及其应用

□ 差分进化算法的概念：

- 差分进化算法（DE）或称为差分演化算法、微分进化算法、微分演化算法、差异演化算法。
- 1995年由Rainer Storn和Kenneth Price提出。
- DE是一种采用实数矢量编码在连续空间中进行随机搜索的优化算法。
- 差分进化算法是一种基于实数编码的具有保优思想的贪心遗传算法。

6.5 差分进化算法及其应用

- 6.5.1 差分进化算法
- 6.5.2 差分进化算法的流程
- 6.5.3 差分进化算法的改进

6.5.1 差分进化算法

□ 差分进化算法的基本要素:

- 参数编码
- 初始群体的产生
- 适应度函数的设计
- 差分操作设计
- 控制参数设置

6.5.1 差分进化算法

参数编码

■ DE采用实数编码方式。

■ DE中的个体 i 表示为： $x_i^t = [x_{i,1}^t, x_{i,2}^t, \text{L}, x_{i,D}^t]$ 。 D 表示问题空间维数， t 表示进化代数。 $x_{i,j}^t$ 为实数。

■ 不必进行数制转换，可直接在解的表现型上进行进化操作。

6.5.1 差分进化算法

初始种群的产生

■ 根据问题固有知识，把握解参数变量 $x_{i,j}^t$ 的取值范围，然后，在此范围内设定初始群体。通常寻找初始种群的一个方法是从给定边界约束内的值中随机选择。

■ 设参数变量的界限为 $x_j^{(L)} < x_j < x_j^{(U)}$ ，则：

$$x_{i,j}^0 = rand[0,1] * (x_j^{(U)} - x_j^{(L)}) + x_j^{(L)}$$

$$i = 1, 2, L, NP \quad j = 1, 2, L, D$$

■ NP 表示种群规模， $rand[0,1]$ 表示在 $[0,1]$ 之间产生的均匀随机数。

6.5.1 差分进化算法

适应度函数的设计

- 在差分进化算法中，差分操作主要通过适应度函数的导向来实现的，它是用来评估个体相对于整个群体的优劣的相对值的大小。
- 通常根据具体问题定义适应度函数，最直观的方法是直接将待求解优化问题的目标函数作为适应度函数。
- 具体设计与遗传算法相似。

6.5.1 差分进化算法

差分操作设计——变异

■ 对每个目标个体 x_i^t , $i=1,2,L, NP$, 它的变异个体 v_i^{t+1} 的产生方式根据差向量的个数以及父代基向量的选取方式的不同分如下几种:

$$\blacksquare \quad v_i^{t+1} = x_{r_1}^t + F * (x_{r_2}^t - x_{r_3}^t) \quad (1)$$

$$\blacksquare \quad v_i^{t+1} = x_{best}^t + F * (x_{r_1}^t - x_{r_2}^t) \quad (2)$$

$$\blacksquare \quad v_i^{t+1} = x_{r_1}^t + F * [(x_{r_2}^t - x_{r_3}^t) + (x_{r_4}^t - x_{r_5}^t)] \quad (3)$$

6.5.1 差分进化算法

差分操作设计——变异（续）

$$\blacksquare \quad v_i^{t+1} = x_{best}^t + F * [(x_{r_1}^t - x_{r_2}^t) + (x_{r_3}^t - x_{r_4}^t)] \quad (4)$$

$$\blacksquare \quad v_i^{t+1} = x_i^t + \lambda * (x_{best}^t - x_i^t) + F * (x_{r_1}^t - x_{r_2}^t) \quad (5)$$

■ 其中， r_1 、 r_2 、 r_3 和 r_4 表示随机产生的 $[1, NP]$ 之间互异且不等于目标个体序号 i 的自然数。 $F \in [0, 2]$ 表示缩放比例因子，是一个实常数因数，控制偏差变量的放大作用。

6.5.1 差分进化算法

差分操作设计——变异（续）

若变异产生的新个体参数值不符合边界约束条件，如何处理？ → 边界条件的处理

- 将不符合边界约束的参数用在可行域中随机产生的参数代替。
- 将不符合边界约束的参数映射到可行域内。

例如：参数 $x \in [0,1]$ ，那么若变异得到的 x 超出该范围，

可通过函数 $x' = \frac{1}{1 + (\frac{1-x}{x})^2}$ 来映射到 $[0,1]$ 范围。

6.5.1 差分进化算法

差分操作设计——交叉

■对目标个体 x_i^t 和变异个体 v_i^{t+1} 实施交叉操作生成试验个体 u_i^{t+1} 的过程称为交叉, $u_i^{t+1} = [u_{i,1}^{t+1}, u_{i,2}^{t+1}, \dots, u_{i,D}^{t+1}]$ 。

■DE交叉操作分二项式 (bin) 交叉和指数 (exp) 交叉两种交叉方式。

■二项式 (bin) 交叉:

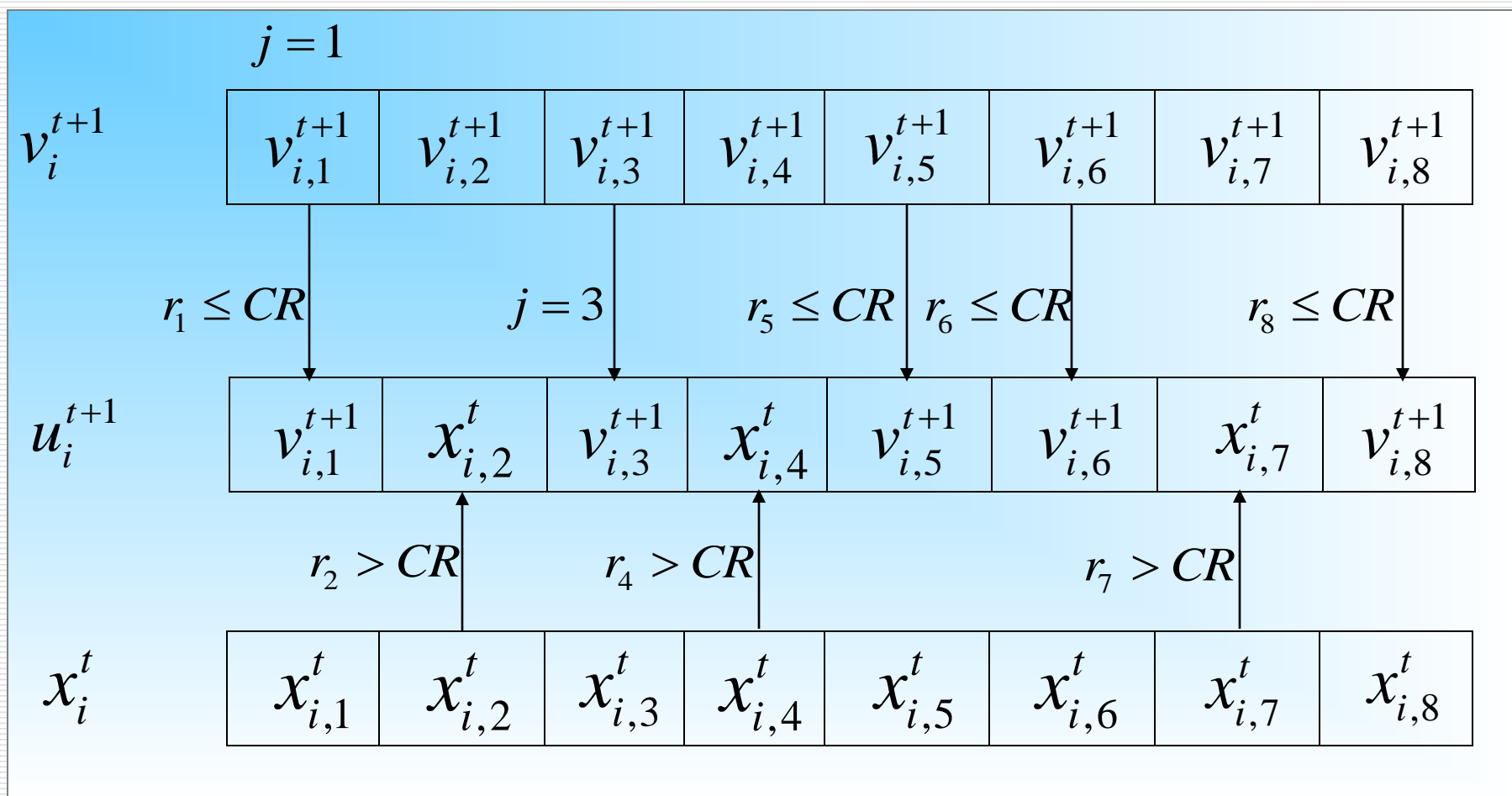
$$u_{i,j}^{t+1} = \begin{cases} v_{i,j}^{t+1}, & \text{if } r_j \leq CR \text{ or } j = rnbr_i \\ x_{i,j}^t, & \text{otherwise} \end{cases}$$

■其中, r_j 为第 j 个 $[0,1]$ 之间的随机数; $rnbr_i$ 为 $[1,D]$ 之间的随机自然数, 它确保了 u_i^{t+1} 至少从 v_i^{t+1} 获得一个参数; CR 为交叉概率, 取值范围为 $[0,1]$ 。

6.5.1 差分进化算法

差分操作设计——二项式交叉（例）

$$D = 8, \quad rnbr_i = 3$$



6.5.1 差分进化算法

差分操作设计——指数交叉

- 指数（exp）交叉。对第 j 位参数：
- 若 $j < rnbr_i$ ：取目标个体上的第 j 位参数作为试验个体上的第 j 位参数；
- 若 $j = rnbr_i$ ：取变异个体上的第 j 位参数作为试验个体上的第 j 位参数；
- 若 $j > rnbr_i$ ：随机产生 $[0,1]$ 之间的随机实数 r_j ，若 $r_j \leq CR$ ，取变异个体上的第 j 位参数作为试验个体上的第 j 位参数，否则，试验个体上剩下的所有参数都从目标个体继承。

6.5.1 差分进化算法

差分操作设计——指数交叉（例）

$$D = 8, \quad rnbr_i = 3$$

$j = 1$

v_i^{t+1}

$v_{i,1}^{t+1}$	$v_{i,2}^{t+1}$	$v_{i,3}^{t+1}$	$v_{i,4}^{t+1}$	$v_{i,5}^{t+1}$	$v_{i,6}^{t+1}$	$v_{i,7}^{t+1}$	$v_{i,8}^{t+1}$
-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------

$j = 3 \quad r_4 \leq CR \quad r_5 \leq CR$

u_i^{t+1}

$x_{i,1}^t$	$x_{i,2}^t$	$v_{i,3}^{t+1}$	$v_{i,4}^{t+1}$	$v_{i,5}^{t+1}$	$x_{i,6}^t$	$x_{i,7}^t$	$x_{i,8}^t$
-------------	-------------	-----------------	-----------------	-----------------	-------------	-------------	-------------

$r_6 > CR$

x_i^t

$x_{i,1}^t$	$x_{i,2}^t$	$x_{i,3}^t$	$x_{i,4}^t$	$x_{i,5}^t$	$x_{i,6}^t$	$x_{i,7}^t$	$x_{i,8}^t$
-------------	-------------	-------------	-------------	-------------	-------------	-------------	-------------

6.5.1 差分进化算法

差分操作设计——选择

■ DE采用贪婪准则在 x_i^t 和 u_i^{t+1} 之间进行选择，产生下一代个体 x_i^{t+1} ：

$$x_i^{t+1} = \begin{cases} u_i^{t+1}, & \text{if } f(u_i^{t+1}) > f(x_i^t) \\ x_i^t, & \text{otherwise} \end{cases}$$

■ $f(\bullet)$ 表示适应度函数。上式是针对最大化问题而言，若是最小化问题，那么选择试验个体作为子代个体的条件是 $f(u_i^{t+1}) < f(x_i^t)$ 。

6.5.1 差分进化算法

控制参数设置

- DE的搜索性能取决于算法全局探索和局部开发能力的平衡，而这在很大程度上依赖于算法的控制参数的选取。
- 种群规模。种群规模 NP 必须满足 $NP \geq 4$ 以确保DE具有足够的不同的变异向量，根据经验， NP 的合理选择在 $5D \sim 10D$ （ D 为问题空间的维数）之间。
- 缩放比例因子。缩放比例因子 $F \in [0, 2]$ 是一个实常数，它决定偏差向量的放大比例。 F 越大，算法更容易逃出局部极小点，更容易收敛到全局最优点。一般取0.7。

6.5.1 差分进化算法

控制参数设置（续）

■交叉概率。交叉概率 CR 是一个范围在 $[0,1]$ 的实数，它控制着一个试验个体参数来自于变异个体的概率。 CR 越大，算法更容易收敛，但易发生早熟现象。 CR 的一个较好的选择是0.3。

■最大迭代代数。一般而言，最大迭代次数越大，最优解越精确，但计算时间越长，要根据具体问题设定。

■终止条件。除最大进化代数可作为DE的终止条件外，有时还需要其它判定准则，一般当目标函数值小于阈值时程序终止，阈值常选为 10^{-6} 。

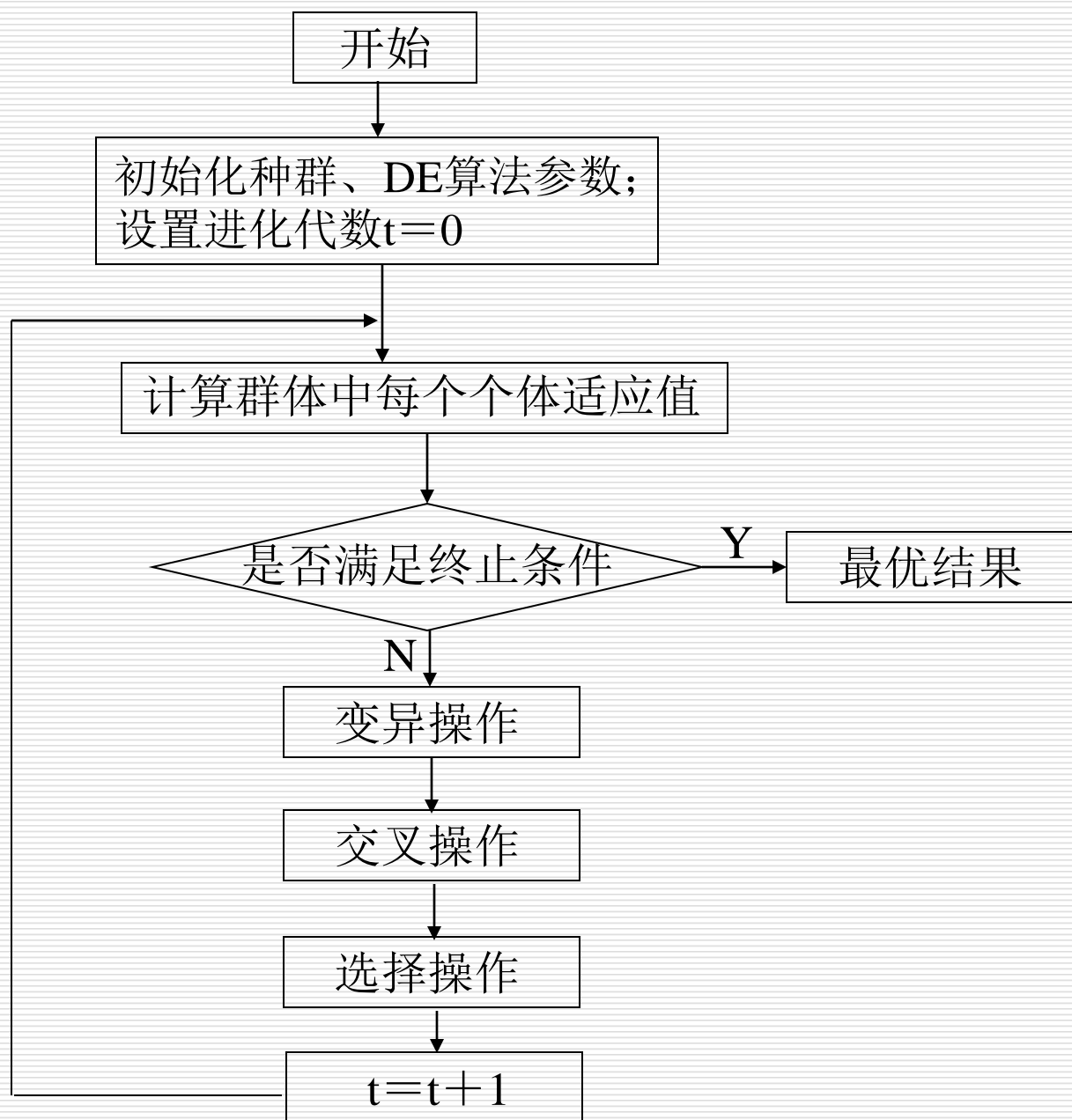
6.5.1 差分进化算法

控制参数设置（续）

■ F , CR 与 NP 一样, 在搜索过程中是常数, 一般 F 和 CR 影响搜索过程的收敛速度和鲁棒性, 它们的优化值不仅依赖于目标函数的特性, 还与 NP 有关。

■ 通常可通过在对不同值做一些试验之后利用试验和结果误差找到 F , CR 和 NP 的合适值。

6.5.2 差分进化算法的流程



差分进化算法的特点

- 算法原理简单，容易实现；算法通用，不依赖于问题信息。
- 分散搜索，能在整个解空间进行全局搜索。
- 保优搜索，具有记忆个体最优解的能力。
- 协同搜索，具有利用个体局部信息和群体全局信息指导算法进一步搜索的能力。
- 直接对结构对象进行操作，不存在对目标函数的限定（如要求函数可导或连续）。

6.5 差分进化算法及其应用

- 6.5.1 差分进化算法
- 6.5.2 差分进化算法的流程
- 6.5.3 差分进化算法的改进

6.5.3 差分进化算法的改进算法

■ DE的收敛速度远胜于一般的进化算法，但基本算法易陷入局部最优，存在早熟收敛现象。许多学者提出了改进型的DE。目前，对DE的改进工作主要包括：

改进进化操作：控制参数的取值、参加变异的父代基向量的选择、差分策略以及差分操作等改进；

嵌入新操作：在DE中嵌入有效的局部搜索策略、引入加速和迁移操作以及搜索空间扩展机制等；

多种群：将DE分成多个子种群，各个子种群独立寻优，在跨种群间实现信息交流，提高摆脱局部极值的能力。

混合算法：与遗传算法、蚁群算法等算法进行混合。

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
- 6.3 遗传算法的改进算法
- 6.4 基于遗传算法的生产调度方法
- 6.5 差分进化算法及其应用
- ✓ 6.6 量子进化算法及其应用

6.6 量子进化算法及其应用

- 6.6.1 量子进化算法的基本概念
- 6.6.2 基本量子进化算法
- 6.6.3 基本量子进化算法的流程
- 6.6.4 基于量子进化算法的生产调度方法

6.6 量子进化算法及其应用

✓ 6.6.1 量子进化算法的基本概念

□ 6.6.2 基本量子进化算法

□ 6.6.3 基本量子进化算法的流程

□ 6.6.4 基于量子进化算法的生产调度方法

量子进化算法的产生

理论上已经证明：进化算法能在概率意义上以随机的方式寻求问题的最优解。但在实际应用中也有着容易产生早熟、收敛速度慢、局部寻优能力差等缺点。

能否将量子理论应用到经典进化算法中去，通过对经典的问题表示作一些调整，使其具有量子理论的特点，从而进行一些类量子的计算来实现更为有效的算法呢？

量子进化算法的产生

- 量子计算（Quantum Computing）是20世纪80年代初提出的一种新兴计算理论，是物理学中的量子力学和计算机科学相结合的产物。
- 量子计算利用量子叠加、纠缠和干涉等量子态所具有的特性，通过并行计算来求解问题。

量子进化算法的产生

将量子物理学引入到计算机科学，利用量子理论中量子态的叠加纠缠等特性，通过量子并行计算解决问题

量子计算理论

量子进化算法

传统进化算法

一种基于量子位、量子叠加态等量子机制的进化算法，建立在量子的态矢量表达基础上，将量子比特的概率幅表示应用于染色体的编码。

模拟自然界物种进化机制的启发式算法，基于适者生存的思想，将问题的求解表达成“染色体”的适者生存的过程，通过染色体群的不断进化最终收敛到问题的满意解

量子进化算法的产生

■国际上最早将量子计算和进化算法相结合的是在1996年，Narayanan将量子理论中的“宇宙”概念类比为遗传算法中的种群，指出量子态的干涉作用可以通过遗传算法的交叉操作实现，设计了量子衍生遗传算法的框架，解决了小规模TSP问题。

■Han在2000到2004年间，提出了一系列量子进化算法的新想法，包括量子遗传算法、并行量子遗传算法、基于移民操作的量子进化算法、基于一种新的终止判断条件的两段式量子进化算法，同时对算法的参数设计问题进行了探讨，并成功应用于背包问题，取得了优于传统遗传算法的效果。

6.6.1 量子进化算法的基本概念

□ 量子位（量子比特*Q-bit*）

在量子进化算法中，染色体不是用确定的值表示，而是采用量子位表示，或者概率幅表示。一个量子位的状态可以取值 $|0\rangle$ 或 $|1\rangle$ ，还可以是两种状态的线性叠加，因此其状态可以表示为：

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad |\alpha|^2 + |\beta|^2 = 1$$

其中， α 和 β 为复数，分别表示状态 $|0\rangle$ 和 $|1\rangle$ 的概率幅， $|\alpha|^2$ 和 $|\beta|^2$ 表示该量子位处于状态 $|0\rangle$ 和 $|1\rangle$ 的概率大小。

6.6.1 量子进化算法的基本概念

□ 量子染色体

进化算法EA 常用编码方式有二进制编码、十进制编码和符号编码。

量子进化算法采用了一种新颖的编码方式——量子比特编码，具体形式可以描述为：

$$q = \left[\begin{array}{c|c|c} \alpha_1 & \alpha_2 & \alpha_m \\ \beta_1 & \beta_2 & \beta_m \end{array} \right] \mathbf{L}$$

这种表示方法可以表征任意的线性叠加态。基于概率，长度为 m 的染色体可以表示 2^m 种状态。

6.6.1 量子进化算法的基本概念

例如，一个具有如下概率幅的3量子比特组成的量子染色体：

$$q = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 1/2 \\ 1/\sqrt{2} & -1/\sqrt{2} & \sqrt{3}/2 \end{bmatrix}$$

系统状态可以表示为：

$$\frac{1}{4}|000\rangle + \frac{\sqrt{3}}{4}|001\rangle - \frac{1}{4}|010\rangle - \frac{\sqrt{3}}{4}|011\rangle + \frac{1}{4}|100\rangle + \frac{\sqrt{3}}{4}|101\rangle - \frac{1}{4}|110\rangle - \frac{\sqrt{3}}{4}|111\rangle$$

而传统的编码方式只能表示一个具体的状态，所以QEA比其他传统进化算法更容易保持种群的多样性。

6.6 量子进化算法及其应用

□ 6.6.1 量子进化算法的基本概念

✓ 6.6.2 基本量子进化算法

□ 6.6.3 基本量子进化算法的流程

□ 6.6.4 基于量子进化算法的生产调度方法

6.6.2 基本量子进化算法

□ 量子进化算法基本要素:

- 染色体编码
- 初始化群体
- 量子观测
- 量子评价
- 量子更新
- 量子进化

6.6.2 基本量子进化算法

□ 染色体编码

■ 二进制编码

$$\text{令 } t=0, P(t) = \{p_1^t, L, p_n^t\} \quad p_j^t = \left[\begin{array}{c|c} \alpha_1^t & \alpha_m^t \\ \beta_1^t & \beta_m^t \end{array} \right] L$$

■ 多进制编码

根据二进制编码的特点，可以用两个量子位表示4个状态，以此类推，用n个量子位表示 2^n 个状态。

多进制编码是将二进制编码扩展到n维，每一维都代表一个量子比特，通常用一个量子矩阵表示。

矩阵的每一行即一个二进制编码表示，矩阵由n列这样的二进制编码组成，采用多个量子位来表示一个多状态基因，对实际问题的编码设计通用性好且实现简单。

6.6.2 基本量子进化算法

□ 初始化群体

■ **初始化方法一**：初始种群中量子染色体的所有量子位的初始值均可设为 $[1/\sqrt{2}, 1/\sqrt{2}]$ 。这种初始化方法十分简单，但没有考虑到具体的应用问题，因此优化效果有一定的局限性。

■ **初始化方法二**：在第一阶段中随机初始化量子位，经过若干次量子进化搜索后，将得到的优良结果用于第二阶段量子染色体的初始化，进行进一步的解空间的全局搜索。所以称为**两阶段初始化方法**。

6.6.2 基本量子进化算法

□ **量子观测**：量子种群 $Q(t)$ \rightarrow 观测种群 $R(t)$

■ **量子观测的一般方法**：根据概率幅的取值情况构造长度为 m 的二进制串。产生 $[0, 1]$ 上的一个随机数 s ，若 $s > |\alpha_i|^2$ ，则对应取值为1，否为0。由此得到二进制观测种群 $R(t)$ 。

■ **量子观测的结果和作用**就是将只有在量子计算机中才能观测的信息转换成在二进制计算机中能表示的信息。这一过程将导致具有 2^m 个状态的量子染色体退变成一个确定的状态，在量子理论中被称为**量子坍塌**（collapse）。

6.6.2 基本量子进化算法

□ 量子评价

- 量子评价即根据实际的应用问题，对 $R(t)$ 各个体进行评价并保留最优个体。
- 该过程与遗传算法等进化算法中的评价适应度函数过程相同。

6.6.2 基本量子进化算法

□ 量子更新

■ 量子更新可以采用很多种方法，例如量子异或门、量子Hadamard变换门等各种量子门。

■ 应用最多的是采用量子旋转门来更新量子位概率幅：

$$\begin{bmatrix} \alpha'_i \\ \beta'_i \end{bmatrix} = U(\theta_i) \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix}, \quad \theta_i = s(\alpha_i, \beta_i) \Delta\theta_i$$

r_i	b_i	$f(r) < f(b)$	$\Delta\theta_i$	$s(\alpha_i, \beta_i)$			
				$\alpha_i\beta_i > 0$	$\alpha_i\beta_i < 0$	$\alpha_i = 0$	$\beta_i = 0$
0	0	false	0	0	0	0	0
0	0	true	0	0	0	0	0
0	1	false	0	0	0	0	0
0	1	true	0.05π	-1	+1	± 1	0
1	0	false	0.01π	-1	+1	± 1	0
1	0	true	0.025π	+1	-1	0	± 1
1	1	false	0.005π	+1	-1	0	± 1
1	1	true	0.025π	+1	-1	0	± 1

r_i 和 b_i 分别表示解 r 与当前最优个体 b 的第 i 个量子位对应的二进制位。

$\Delta\theta_i$ 和 $s(\alpha_i, \beta_i)$ 分别为旋转角的幅度和旋转方向。

6.6.2 基本量子进化算法

□ 量子进化

■基本量子进化算法中，由于量子旋转门更新实际上就相当于对量子染色体进行了相应的进化操作，因此一般认为这些操作可以采用也可以省略。

■基本的量子进化操作主要是量子交叉和量子变异。量子交叉和量子变异的基本思想和进化算法中的交叉和变异相同，所不同的只是操作的对象是量子染色体。

6.6.2 基本量子进化算法

□ 量子进化

■还可以采用**灾变操作**来确保算法全局优化性能的提高。灾变是指如果进化在某个特定代数之后没有产生任何优化，则算法将灾变到某个特定的状态并进行重新优化。

■也可以采用**全干扰算子**：指同时引入多个个体进行操作，即将多个个体的染色体拆开，依照预先拟定的准则重组以构造新的个体，即使在多个个体完全相同的极端情况下也可以产生新个体等。

量子进化算法的改进

■ 目前对量子进化算法的改进主要可归纳为如下几方面：

对量子染色体的编码、种群初始化方法、种群规模的确定等方面的改进；

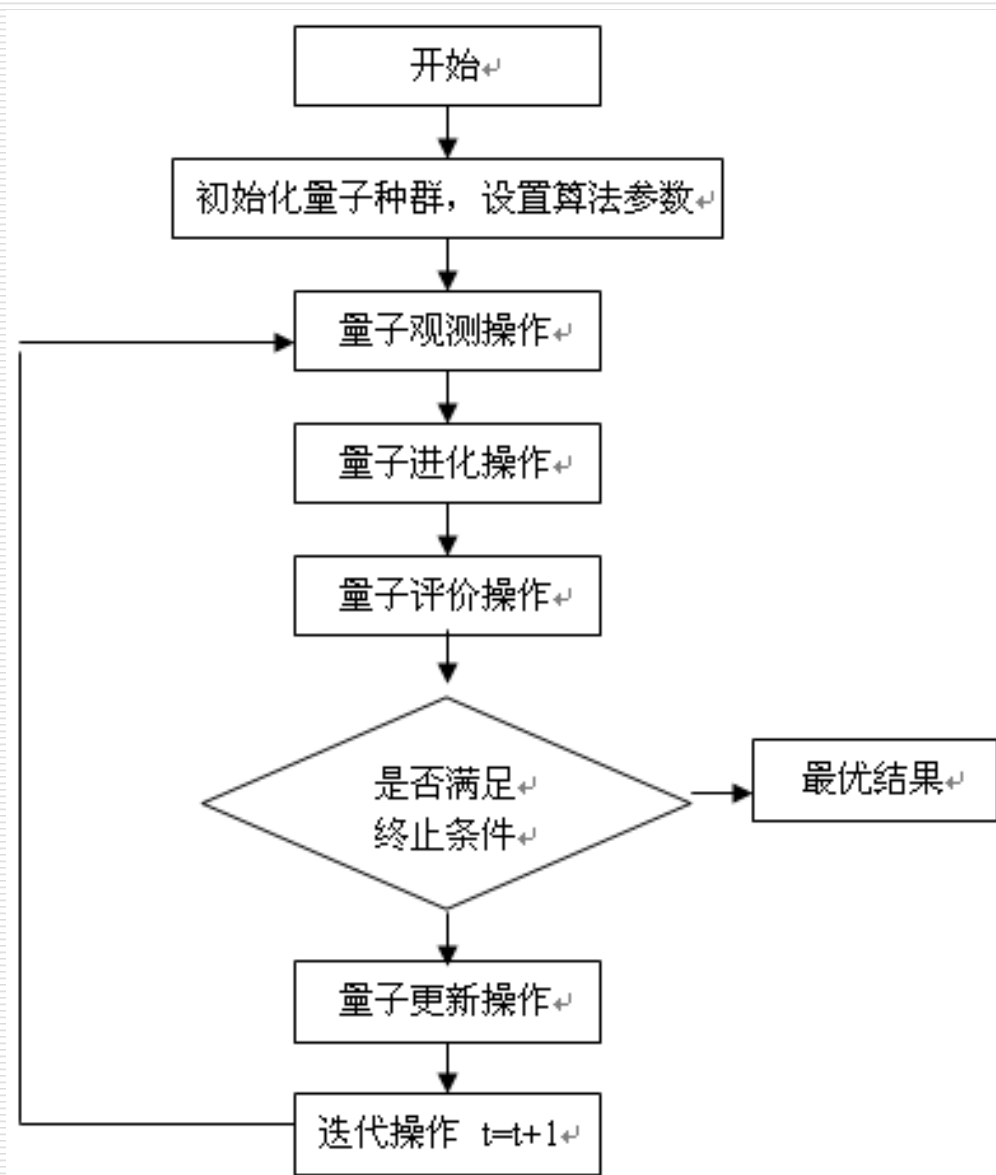
对量子更新（动态量子旋转角、非量子旋转门更新）、量子进化等操作的改进；

算法终止的判断条件方面的改进；

与其他算法的混合；

扩展为并行算法。

6.6.3 基本量子进化算法的流程



量子进化算法的特点

- 算法原理简单，容易实现；
- 算法通用，不依赖于问题信息。
- 种群分散性好，小种群个体可以对应多个个体编码。
- 群体搜索，具有极强的全局搜索能力。
- 协同搜索，具有利用当前最优个体信息算法进一步搜索的能力。
- 收敛速度快，易于与其他算法混合。

6.6 量子进化算法及其应用

- 6.6.1 量子进化算法的概念
- 6.6.2 基本量子进化算法
- 6.6.3 量子进化算法的流程
- ✓ 6.6.4 基于量子进化算法的生产调度方法

6.6.4 基于量子进化算法的生产调度方法

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。

6.6.4 基于量子进化算法的生产调度方法

■ 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) 工序的准备时间与顺序无关，且包含在加工时间中。
- (5) 工件在每台机器上的加工顺序相同，且是确定的。

6.6.4 基于量子进化算法的生产调度方法

■ Flow Shop生产调度问题的数学模型:

$c(j_i, k)$: 工序 j_i 在机器 k 上的加工完工时间, $\{j_1, j_2, \dots, j_n\}$
 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: $c_{\max} = c(j_n, m)$

$$\{j_1, j_2, \dots, j_n\} \quad c_{\max}$$

6.6.4 基于量子进化算法的生产调度方法

求解Flow Shop调度问题的QEA算法设计

(1) 量子编码

■ 随机产生 n （工件数）个量子比特构成量子染色体。

(2) 量子选择

■ 计算适应度函数 $\min Z = C_{\max}$ 将所有染色体从好到差排序。

(3) 量子交叉

■ 随机产生一个自然数 r （ $r < n$ ），交换两条父代量子染色体位于位置 r （包括位置 r ）之后的量子比特位。

6.6.4 基于量子进化算法的生产调度方法

(4) 量子变异

■ 随机产生一个数 r ($r < n$)，交换量子染色体上 α_r 和 β_r 的位置。

(5) 量子更新

■ 用量子旋转门来更新量子位概率幅。

6.6.4 基于量子进化算法的生产调度方法

求解Flow Shop调度问题的QEA算法设计

(6) 量子观测

- 将量子染色体转变成一般的实数染色体。

(7) 量子评价

- 把 $|\alpha_i|^2$ 认为是工件 i 排在前面的概率，即 $|\alpha_i|^2$ 越大，工件 i 排在前面的可能性越大。

6.6.4 基于量子进化算法的生产调度方法

求解FSP的QEA算法实例

采用由Carlier设计的8个不同规模的Benchmark问题Car1, Car2, ..., Car8进行实验。仿真时的参数设置为：种群规模40，最大代数（停止条件）为 $J \times M$ ，染色体长度为J，量子交叉概率和遗传交叉概率为1，量子变异概率和遗传变异概率都为0.05。对于每个问题，算法执行20次。

6.6.4 基于量子进化算法的生产调度方法

问题	工件数, 机器数	C*	NEH	QEA	
			RE	BRE	ARE
Car1	11, 5	7038	0	0	0
Car2	13, 4	7166	2.93	0	1.90
Car3	12, 5	7312	1.19	1.19	1.65
Car4	14, 4	8003	0	0	0.06
Car5	10, 6	7720	1.49	0	0.11
Car6	8, 9	8505	3.15	0	0.19
Car7	7, 7	6590	0	0	0
Car8	8, 8	8366	2.37	0	0.03

6.7 小结

- 差分进化算法是一种基于实数编码的具有保优思想的贪婪遗传算法，也包括选择、交叉和变异等操作，但在产生子代的方式上有所不同，**DE**在父代个体间的差向量基础上生成变异个体，然后按一定的概率对父代个体与变异个体进行交叉操作，最后采用“贪婪”选择策略产生子代个体。
- **DE**的收敛速度远胜于一般的进化算法，但基本算法易陷入局部最优，存在早熟收敛现象。

6.7 小结

- ❑ 量子进化算法是一种基于量子位、量子叠加态等量子机制的进化算法。主要包含了六个基本要素：量子编码，初始化种群，量子观测，进化操作，量子评价，量子更新等。
- ❑ 基于量子比特表示与传统表示方法相比提供了一种更好的多样性，因为它能表示量子态的叠加。
- ❑ 量子门操作的存在使得量子遗传算法有着很强的全局搜索能力。



THE END

