

# 第 6 章 智能计算及其应用

---



# 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。介绍了群智能算法产生的背景和粒子群优化算法。介绍了蚁群算法及其应用。

# 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
- 6.5 群智能算法产生的背景
- 6.6 粒子群优化算法及其应用
- 6.7 蚁群算法及其应用

# 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

✗ 6.5 群智能算法产生的背景

✗ 6.6 粒子群优化算法及其应用

✗ 6.7 蚁群算法及其应用

# 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

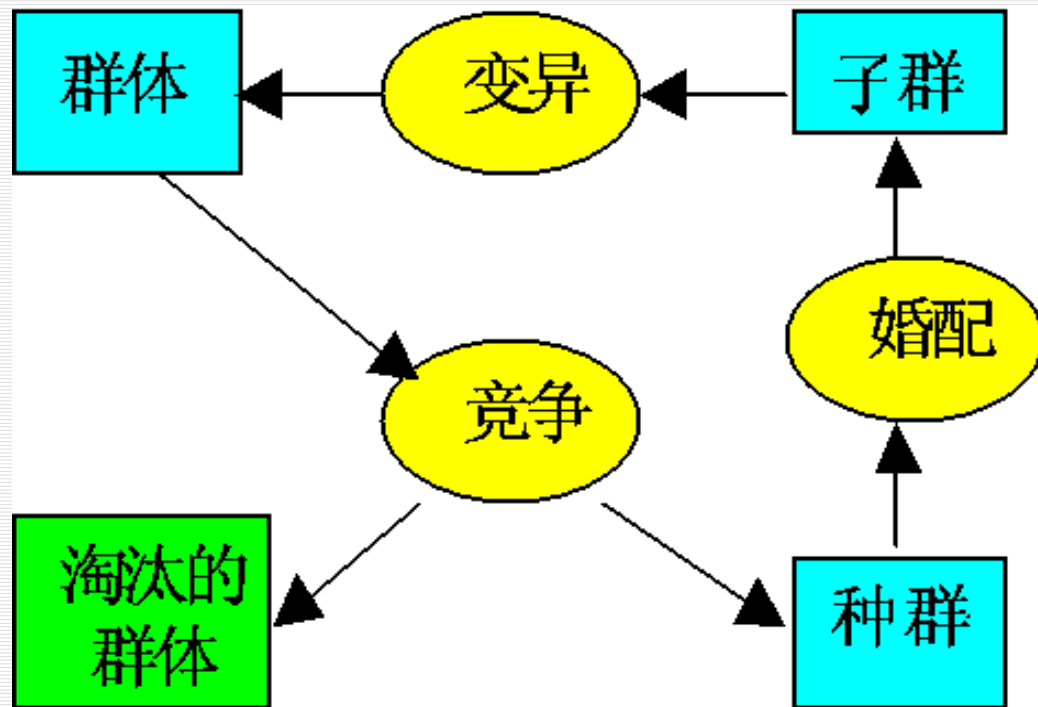
- **适者生存**：最适合自然环境的群体往往产生了更大的后代群体。
- **生物进化的基本过程**：

**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：一个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：在生物界被认为是有效的方法及操作可以通过类比的方法引入到算法中，有时会

# 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
- 6.5 群智能算法产生的背景
- 6.6 粒子群优化算法及其应用
- 6.7 蚁群算法及其应用

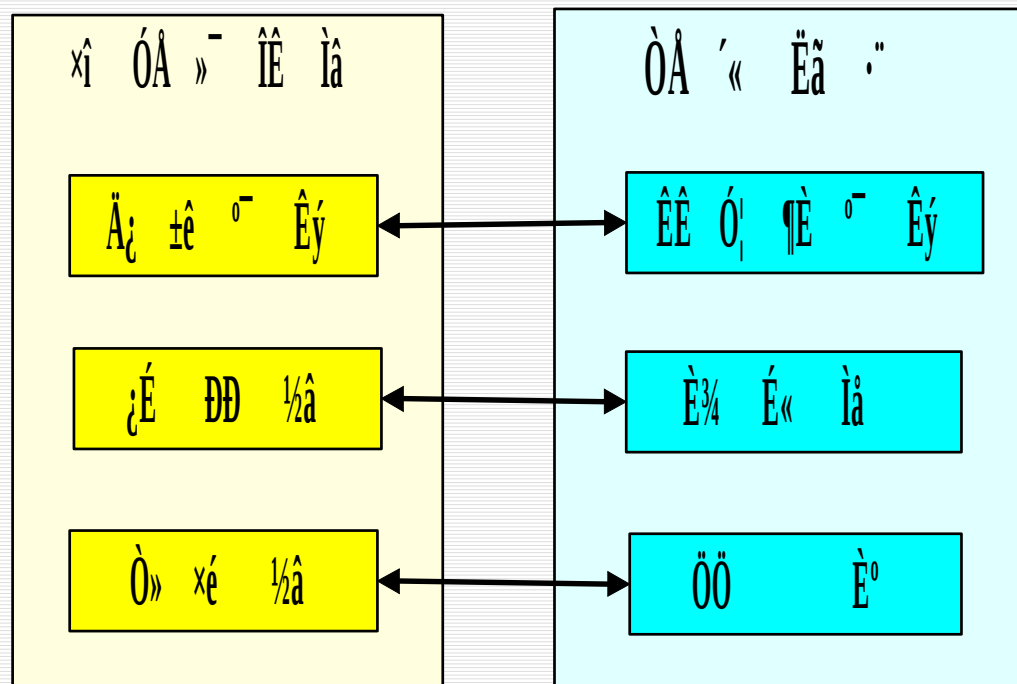
## 6.2 基本遗传算法

- **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（ <b>Individual</b> ）	解
染色体 （ <b>Chromosome</b> ）	解的编码（字符串、向量等）
基因（ <b>Gene</b> ）	解的编码中每一分量
适应性（ <b>Fitness</b> ）	适应度函数值
群体（ <b>Population</b> ）	根据适应度值选定的一组解（解的个数为群体的规模）
婚配（ <b>Marry</b> ）	交叉（ <b>Crossover</b> ）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（ <b>Mutation</b> ）	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。

## 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  $\approx$  Gray 编码

Gray 编码  $\approx$  二进制编码

$$\gamma_k = \begin{cases} \beta_1 & k=1 \\ \beta_{k-1} \oplus \beta_k & k \geq 2 \end{cases}$$

$$\beta_k = \sum_{i=1}^k \gamma_i \otimes m_i$$

## 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

$$Fit(f(x)) = \frac{f(x) - \min}{\max - \min}$$

- 若目标函数为最小化问题, 则

$$Fit(f(x)) = \frac{\max - f(x)}{\max - \min}$$

## 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 mode**  
1) 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 mode**

- 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) 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) 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)

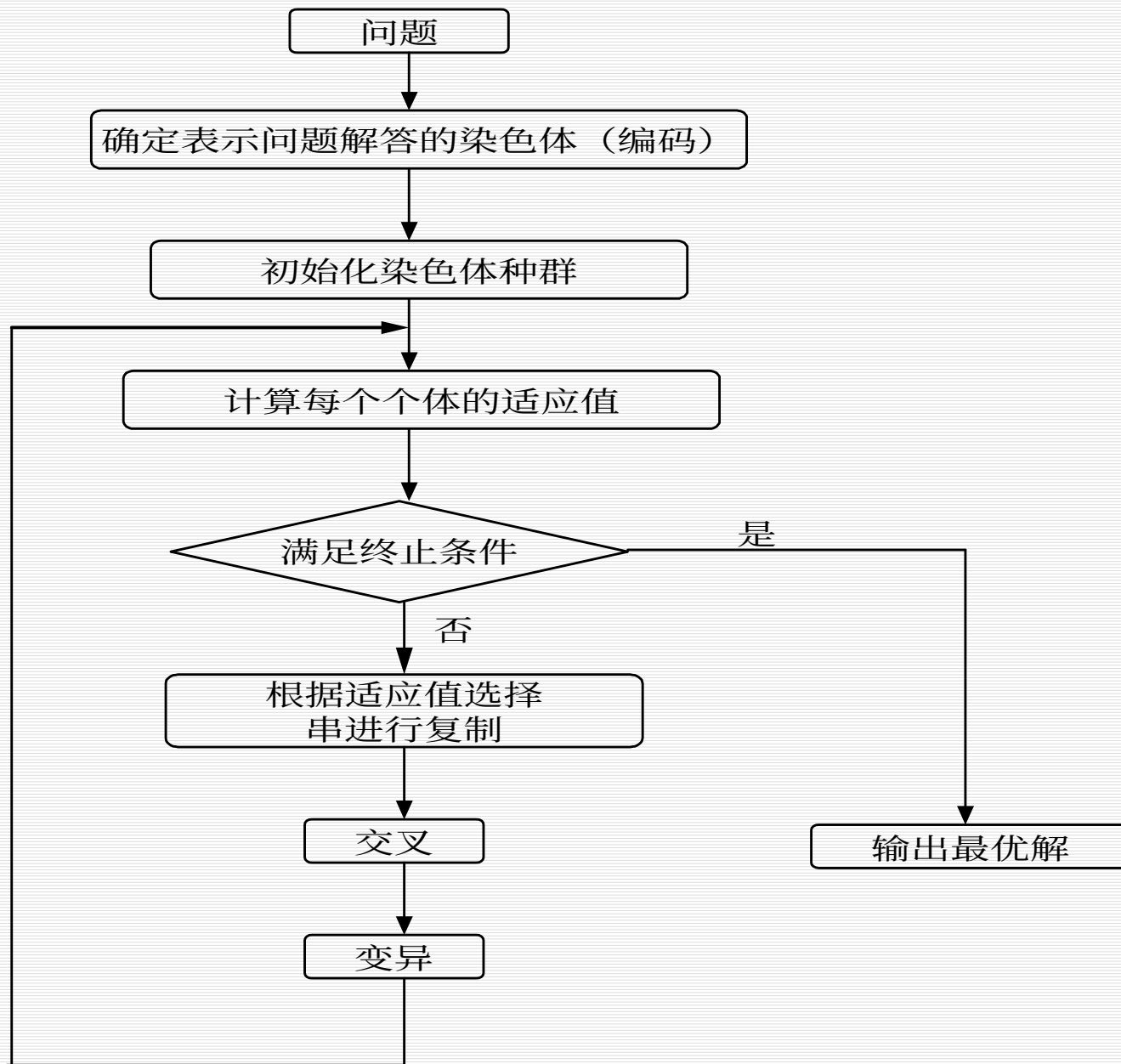


$A =$	9	8	4		5	6	7		1	3	2
$B =$	8	7	1		2	3	9		5	4	6
$A' =$	9	8	4		2	3	9		1	3	2
$B' =$	8	7	1		5	6	7		5	4	6

## 6.2.8 变异

- (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。
- (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。

## 6.2.9 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  $pop(1) := 1$  ;

(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

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

## 6.2.9 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

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

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

返回 (2)。

## 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，无论是线性的还是非线性的，离散的还是连续的都可处理，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。
- **2.** 进化算子的各态历经性使得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
- 6.5 群智能算法产生的背景
- 6.6 粒子群优化算法及其应用
- 6.7 蚁群算法及其应用



## 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

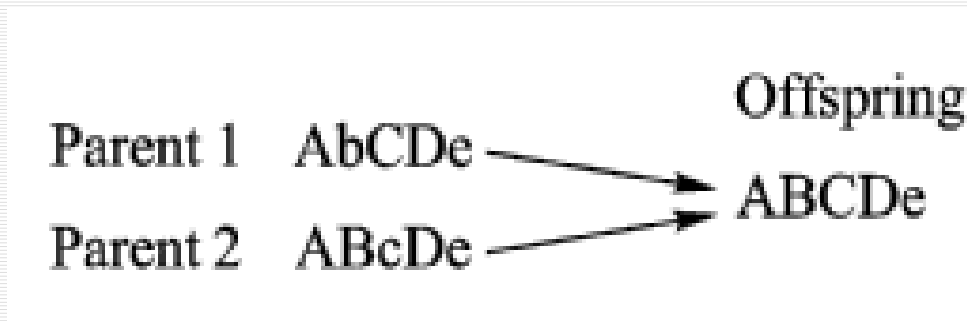
 **6.3.3** 自适应遗传算法

## 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。

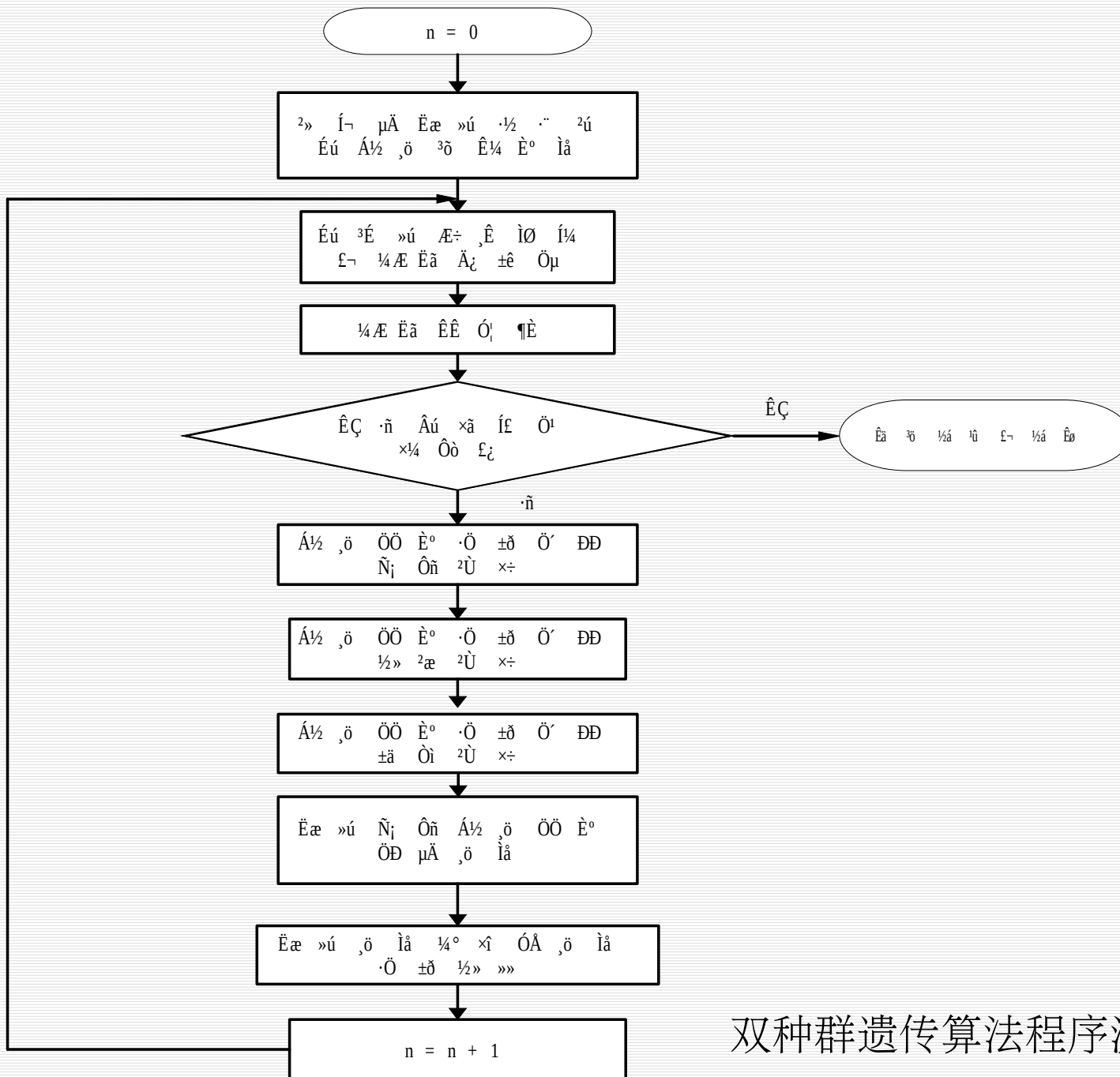


■ 双倍体遗传延长了有用基因块的寿命，提高了算法的收敛能力，在变异概率低的情况下能保持一定水平的多样性。

## 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。



双种群遗传算法程序流程图

## 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。

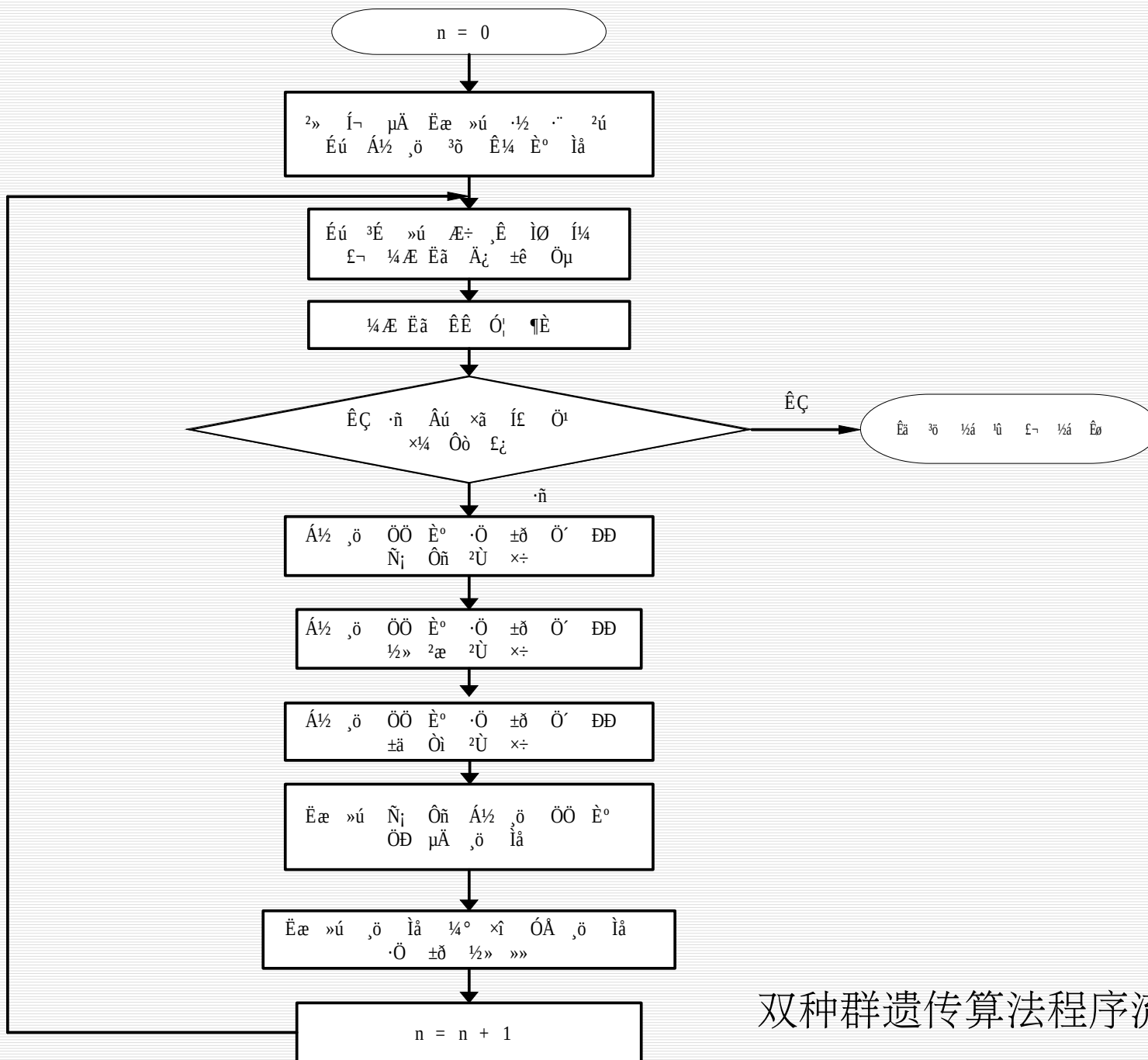
■ **多种群遗传算法**：建立两个遗传算法群体，分别独立地运行复制、交叉、变异操作，同时当每一代运行结束以后，选择两个种群中的随机个体及最优个体分别交换。

## 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) 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  $P_c$  和  $P_m$  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  $P_m$

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。

# 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
- 6.5 群智能算法产生的背景
- 6.6 粒子群优化算法及其应用
- 6.7 蚁群算法及其应用

## 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

$C_{\max}^k$ : 染色体  $k$  的最大流程时间,

FSP 的适应度函数:

$$ev(q_k) = \frac{1}{C_{\max}^k}$$

## 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

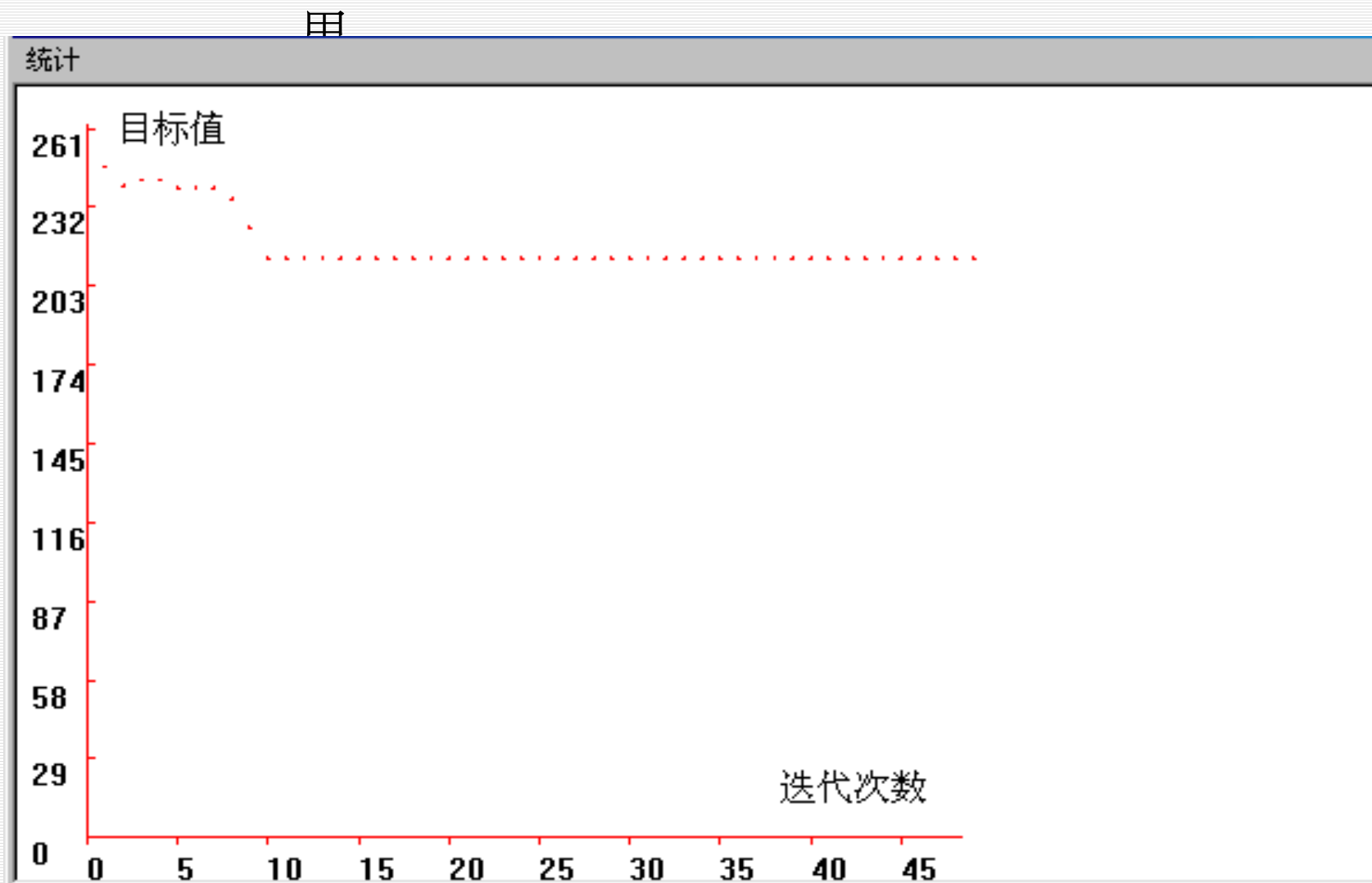
用穷举法求得最优解： 4-2-5-1-3 ， 加工时间： 213 ；  
最劣解： 1-4-2-3-5 ， 加工时间： 294 ； 平均解的加工时间：  
265 。

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

遗传算法运行的结果

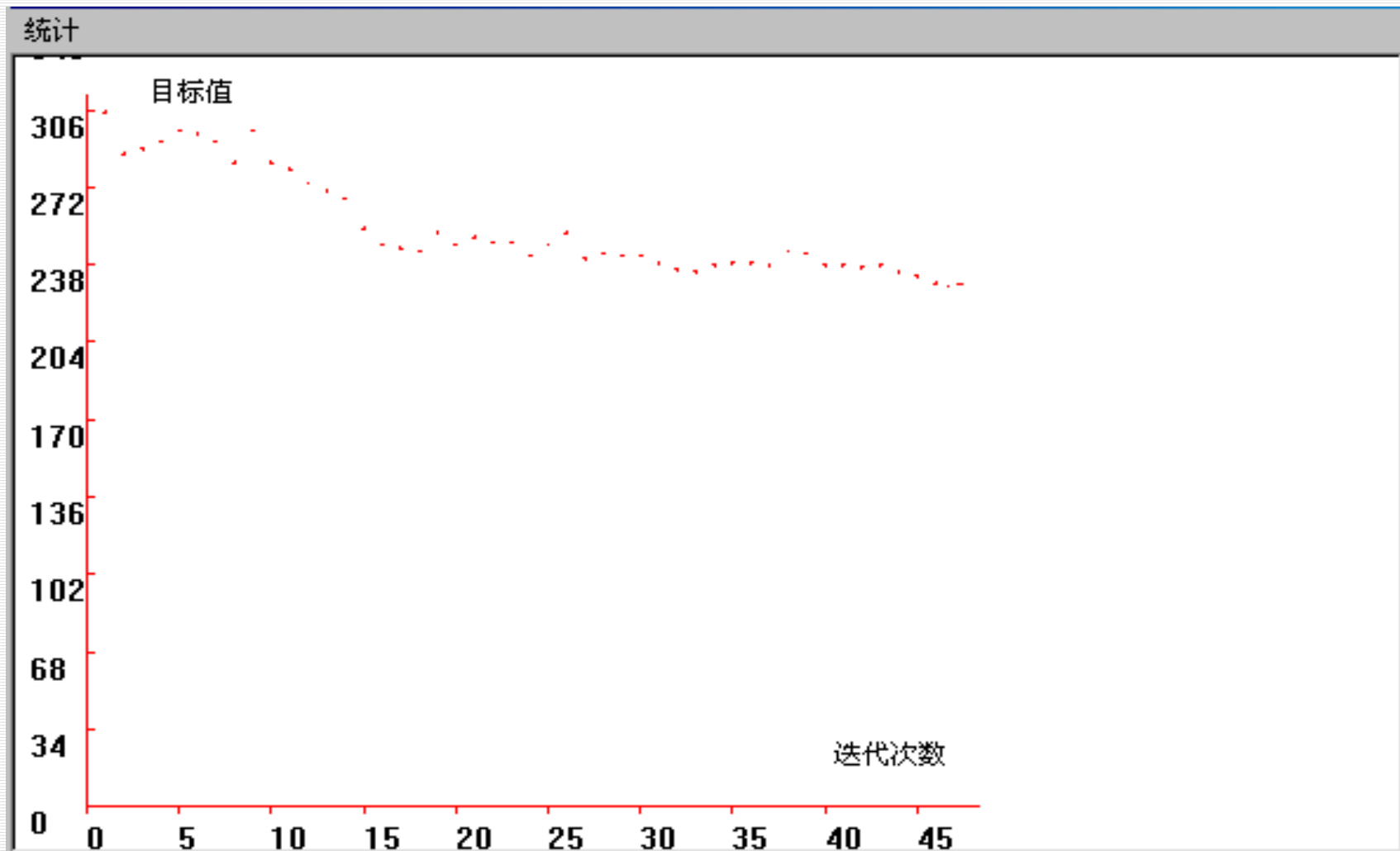
总运行 次数	最好解	最坏解	平均	最好解 的频率	最好解的 平均代数
20	213	221	213.95	0.85	12

## 6.4 遗传算法的应用



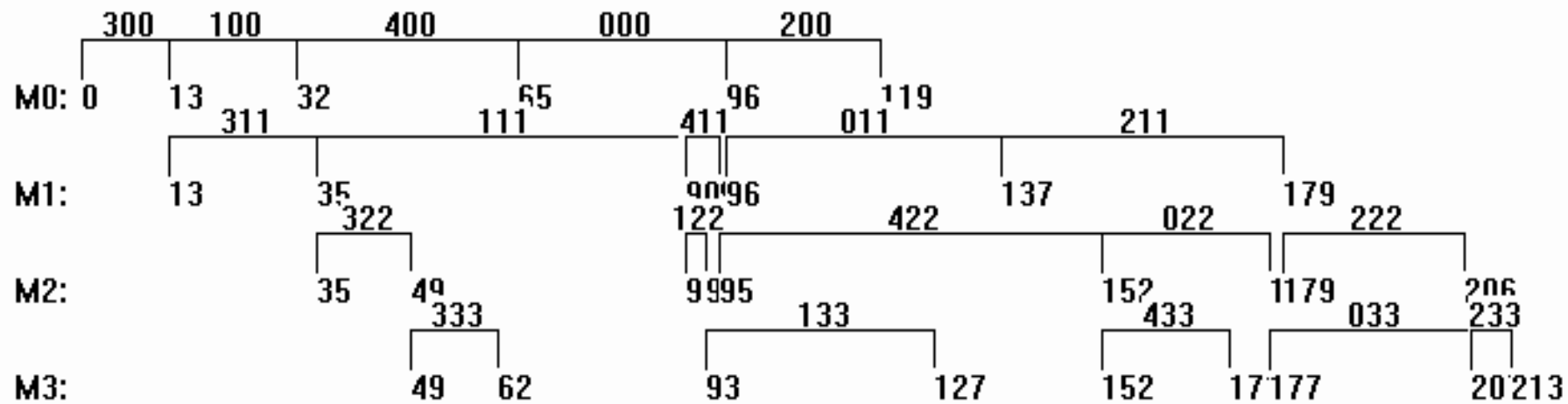
最优解收敛图

## 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

# 第 6 章 智能计算及其应用

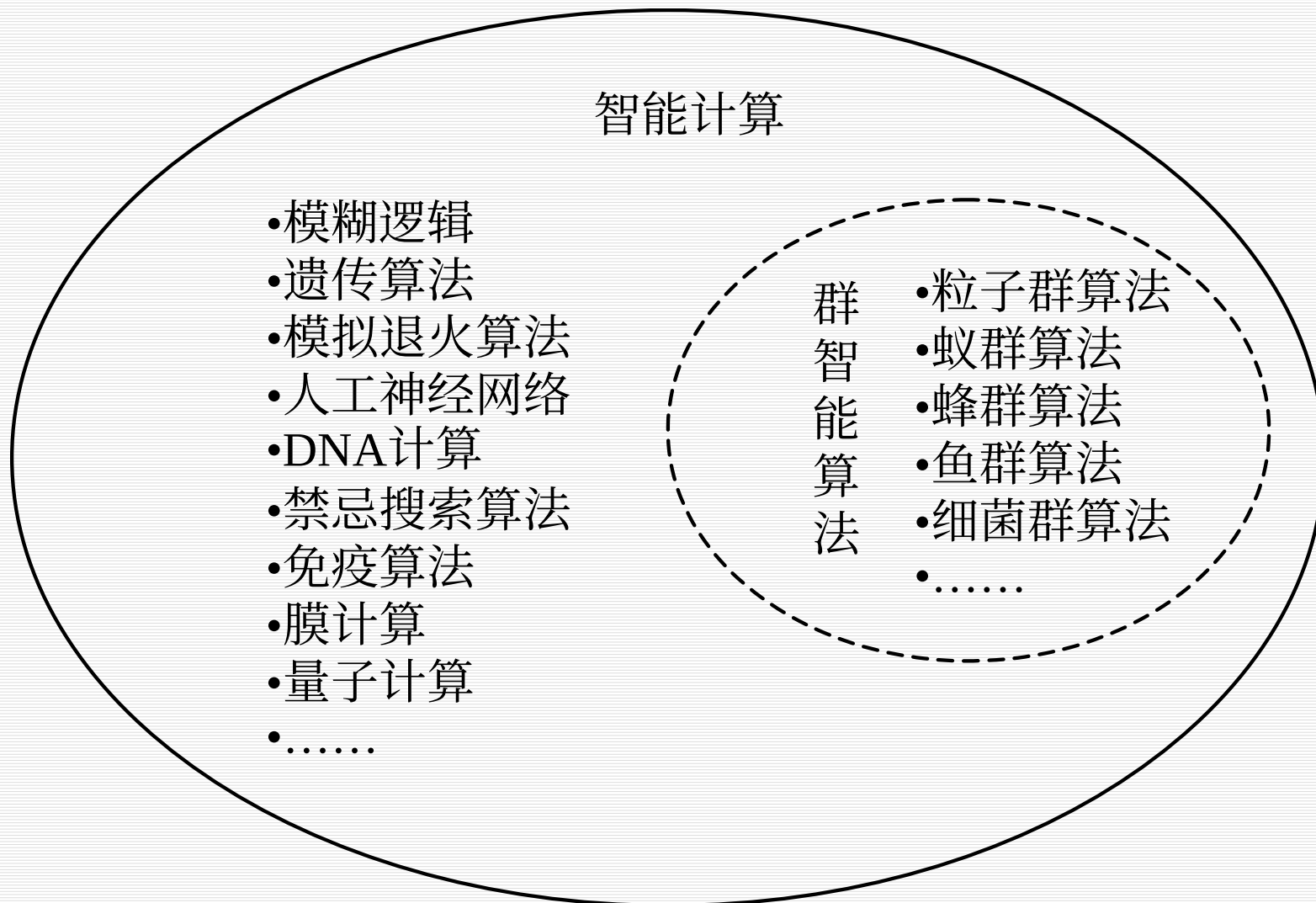
- 6.1 进化算法的产生与发展
- 6.2 基本遗传算法
- 6.3 遗传算法的改进算法
- 6.4 遗传算法的应用
- 6.5 群智能算法产生的背景
- 6.6 粒子群优化算法及其应用
- 6.7 蚁群算法及其应用

## 6.5 群智能算法产生的背景

- **群智能算法**（swarm algorithms , SI）：受动物群体智能启发的算法。
- **群体智能**：由简单个体组成的群落与环境以及个体之间的互动行为。
- **群智能算法**包括：粒子群优化算法、蚁群算法、蜂群算法、.....



## 6.5 群智能算法产生的背景



# 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
- 6.5 群智能算法产生的背景
- 6.6 粒子群优化算法及其应用
- 6.7 蚁群算法及其应用

## 6.6 粒子群优化算法及其应用

### □ 产生背景

粒子群优化（Particle Swarm Optimization, PSO）算法是由美国普渡大学的 Kennedy 和 Eberhart 于 1995 年提出，它的基本概念源于对鸟群觅食行为的研究。

### □ 设想这样一个场景：

一群鸟在随机搜寻食物，在这个区域里只有一块食物，所有的鸟都不知道食物在哪里，但是它们知道当前的位置离食物还有多远。那么找到食物的最优策略是什么呢？

最简单有效的就是搜寻目前离食物最近的鸟的周围区域。

## 6.6 粒子群优化算法及其应用

 **6.6.1** 粒子群优化算法的基本原理

 **6.6.2** 粒子群优化算法的参数分析

## 6.6.1 粒子群优化算法的基本原理

### □ 基本思想

将每个个体看作  $n$  维搜索空间中一个没有体积质量的粒子，在搜索空间中以一定的速度飞行，该速度决定粒子飞行的方向和距离。所有粒子还有一个由被优化的函数决定的适应值。

### □ 基本原理

PSO 初始化为一群随机粒子，然后通过迭代找到最优解。在每一次迭代中，粒子通过跟踪两个“极值”来更新自己。第一个就是粒子本身所找到的最优解，这个解称为个体极值。另一个是整个种群目前找到的最优解，这个解称为全局极值。

## 6.6.1 粒子群优化算法的基本原理

### □ 算法定义

在  $n$  维连续搜索空间中，对粒子群中的第  $i$  ( $i=1, 2, \dots, m$ ) 个粒子进行定义：

$$x^i(k) = [x_1^i \quad x_2^i \quad \cdots \quad x_n^i]^T$$

■  $x^i(k)$  : 表示搜索空间中粒子的当前位置。

■  $p^i(k) = [p_1^i \quad p_2^i \quad \cdots \quad p_n^i]^T$  表示该粒子至今所获得的具有最优适应度的位置。

■  $v^i(k) = [v_1^i \quad v_2^i \quad \cdots \quad v_n^i]^T$  表示该粒子的搜索方向。

## 6.6.1 粒子群优化算法的基本原理

每个粒子经历过的最优位置 ( pbest ) 记为  $p^i(k) = [p_1^i \ p_2^i \ \cdots \ p_n^i]^T$  ; 群体经历过的最优位置 (gbest) 记为  $p^g(k) = [p_1^g \ p_2^g \ \cdots \ p_n^g]^T$  , 则基本的 PSO 算法为:

$$v_j^i(k+1) = \omega(k)v_j^i(k) + \varphi_1 rand(0, a_1)(p_j^i(k) - x_j^i(k)) + \varphi_2 rand(0, a_2)(p_j^g(k) - x_j^i(k)) \quad \text{--- (7.1a)}$$

$$x_j^i(k+1) = x_j^i(k) + v_j^i(k+1) \quad \text{--- (7.1b)}$$

$$i = 1, 2, \dots, m; \quad j = 1, 2, \dots, n$$

其中,  $\omega$  是惯性权重因子。 $\varphi_1$ ,  $\varphi_2$  是加速度常数, 均为非负值。 $rand(0, a_1)$  和  $rand(0, a_2)$  为  $[0, a_1]$ 、 $[0, a_2]$  范围内的具有均匀分布的随机数,  $a_1$  与  $a_2$  为相应的控制参数。

## 6.6.1 粒子群优化算法的基本原理

$$v_j^i(k+1) = \omega(k)v_j^i(k) + \varphi_1 \text{rand}(0, a_1)(p_j^i(k) - x_j^i(k)) + \varphi_2 \text{rand}(0, a_2)(p_j^g(k) - x_j^i(k)) \quad \text{--- (7.1a)}$$

- 式 (7.1a) 右边的第 1 部分是粒子在前一时刻的速度；
- 第 2 部分为个体“认知”分量，表示粒子本身的思考，将现有的位置和曾经经历过的最优位置相比。
- 第 3 部分是群体“社会 (social)”分量，表示粒子间的信息共享与相互合作。
- $\varphi_1$ ， $\varphi_2$  分别控制个体认知分量和群体社会分量相对贡献的学习率。



## 6.6.1 粒子群优化算法的基本原理

基于学习率  $\varphi_1$   $\varphi_2$  ,

**Kennedy** 给出以下 4 种类型的 **PSO** 模型:

■ 若  $\varphi_1 > 0$  ,  $\varphi_2 > 0$  , 则称该算法为 PSO 全模型。

■ 若  $\varphi_1 > 0$  ,  $\varphi_2 = 0$  , 则称该算法为 PSO 认知模型。

■ 若  $\varphi_1 = 0$  ,  $\varphi_2 > 0$  , 则称该算法为 PSO 社会模型。

■ 若  $\varphi_1 = 0$  ,  $\varphi_2 > 0$  且  $g \neq i$  , 则称该算法为 PSO 无私模型。

## 6.6.1 粒子群优化算法的基本原理

粒子群优化算法的流程：

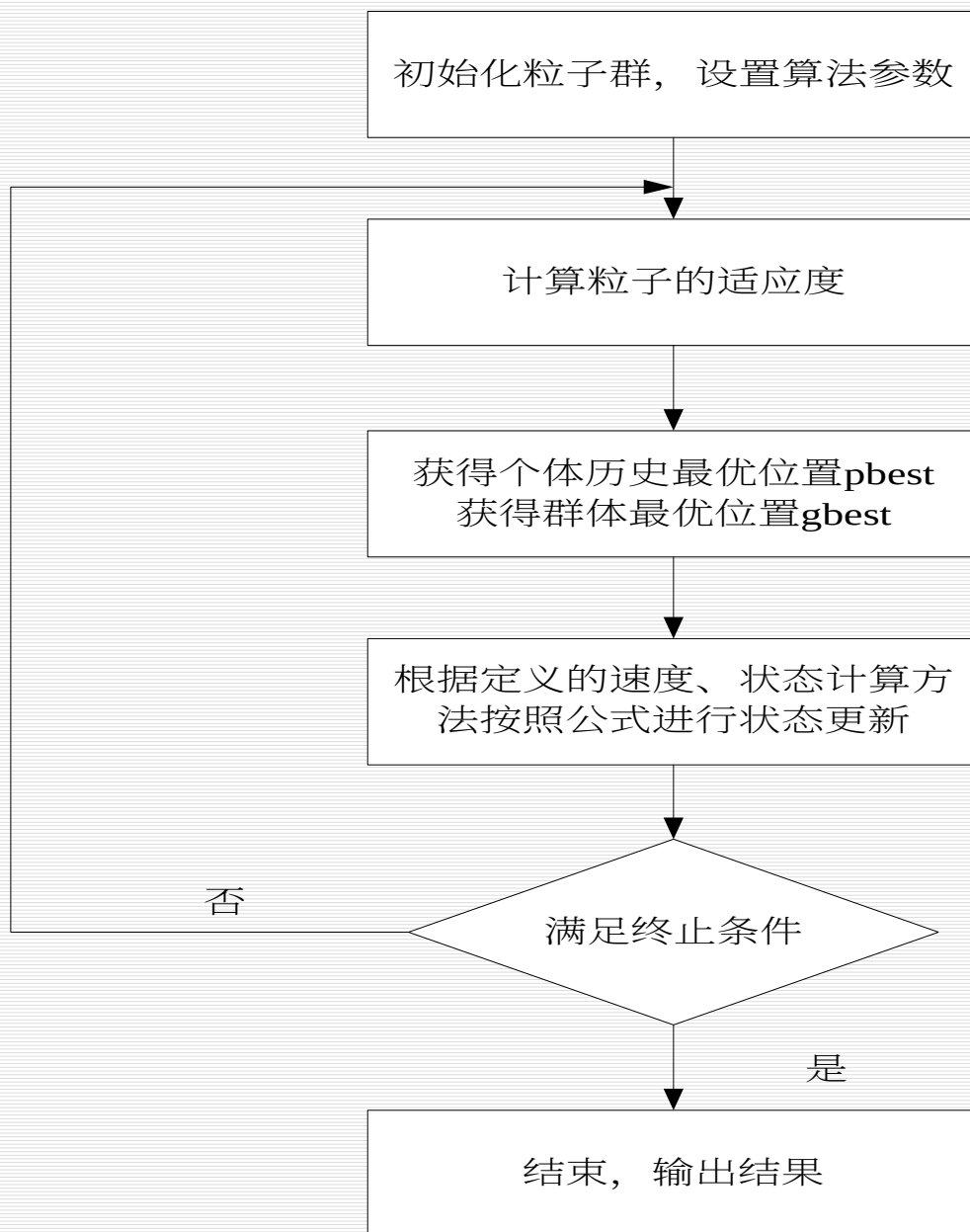
- (1) 初始化每个粒子，即在允许范围内随机设置每个粒子的初始位置和速度。
- (2) 评价每个粒子的适应度，计算每个粒子的目标函数。
- (3) 设置每个粒子的  $P_i$ 。对每个粒子，将其适应度与其经历过的最好位置  $P_i$  进行比较，如果优于  $P_i$ ，则将其作为该粒子的最好位置。

## 6.6.1 粒子群优化算法的基本原理

粒子群优化算法的流程：

- （4）设置全局最优值 $P_g$ 。对每个粒子，将其适应度与群体经历过的最好位置 $P_g$  进行比较，如果优于 $P_g$ ，则将其作为当前群体的最好位置 $P_g$ 。
- （5）根据式（7.1）更新粒子的速度和位置。
- （6）检查终止条件。如果未达到设定条件（预设误差或者迭代的次数），则返回第（2）步。

## 6.6.1 粒子群优化算法流程图



## 6.6.2 粒子群优化算法的参数分析

### 1. PSO 算法的参数

包括：群体规模  $m$ ，惯性权重  $\omega$ ，加速度  $a_1$ ， $a_2$ ，最大速度  $V_{max}$ ，最大代数  $G_{max}$ 。

#### （1）最大速度 $V_{max}$

对速度  $v_i$ ，算法中有最大速度  $V_{max}$  作为限制，如果当前粒子的某维速度大于最大速度  $V_{max}$ ，则该维的速度就被限制为最大速度  $V_{max}$ 。

#### （2）权重因子

3 个权重因子：惯性权重  $\omega$ ，加速度  $a_1$ ， $a_2$ 。

## 6.6.2 粒子群优化算法的参数分析

### 2. 位置更新方程中各部分的影响

$$v_j^i(k+1) = \omega(k)v_j^i(k) + \varphi_1 rand(0, a_1)(p_j^i(k) - x_j^i(k)) + \varphi_2 rand(0, a_2)(p_j^g(k) - x_j^i(k))$$

( 1 ) 只有第 1 部分 , 即

$$\varphi_1 = \varphi_2 = 0$$

粒子将一直以当前的速度飞行, 直到达边界。

由于它只能搜索有限的区域, 所以很难找到好解。

## 6.6.2 粒子群优化算法的参数分析

### 2. 位置更新方程中各部分的影响

$$v_j^i(k+1) = \omega(k)v_j^i(k) + \varphi_1 \text{rand}(0, a_1)(p_j^i(k) - x_j^i(k)) + \varphi_2 \text{rand}(0, a_2)(p_j^g(k) - x_j^i(k))$$

(2) 没有第1部分, 即  $\omega = 0$

速度只取决于粒子当前位置和其历史最好位置  $P_i$  和  $P_g$ , 速度本身没有记忆性。

## 6.6.2 粒子群优化算法的参数分析

$$v_j^i(k+1) = \omega(k)v_j^i(k) + \varphi_1 \text{rand}(0, a_1)(p_j^i(k) - x_j^i(k)) + \varphi_2 \text{rand}(0, a_2)(p_j^g(k) - x_j^i(k))$$

(3) 没有第2部分, 即  $\varphi_2 = 0$

粒子没有认知能力, 也就是“只有社会模型”。

在粒子的相互作用下, 有能力达到新的搜索空间。但对复杂问题, 容易陷入局部最优点。



## 6.6.2 粒子群优化算法的参数分析

$$v_j^i(k+1) = \omega(k)v_j^i(k) + \varphi_1 \text{rand}(0, a_1)(p_j^i(k) - x_j^i(k)) + \varphi_2 \text{rand}(0, a_2)(p_j^g(k) - x_j^i(k))$$

(4) 没有第3部分, 即  $\varphi_2 = 0$

粒子间没有社会共享信息, 也就是“只有认知”模型。

因为个体间没有交互, 一个规模为  $M$  的群体等价于  $M$  个单个粒子的运行, 因而得到最优解的机率非常小。

## 6.6.2 粒子群优化算法的参数分析

### 3. 参数设置

早期的实验：。固定为 1.0， $\omega_1$  和  $\omega_2$  固定为 2.0，因此 Vmax 成为唯一需要调节的参数，通常设为每维变化范围 10 % ~20%。Suganthan 的实验表明， $\omega_1$  和  $\omega_2$  为常数时可以得到较好的解，但不一定必须为 2。

这些参数也可以通过模糊系统进行调节。Shi 和 Eberhart 提出一个模糊系统来调节。

## 6.6.3 粒子群优化算法的应用

-  7.4.1 粒子群优化算法应用领域
-  7.4.2 粒子群优化算法在 **PID** 参数整定中的应用
-  7.4.3 粒子群优化算法在车辆路径问题中的应用

## 6.6.3 粒子群优化算法应用领域

粒子群优化算法已在诸多领域得到应用，归纳如下：

（ 2 ） 化工系统领域

（ 3 ） 电力系统领域

（ 4 ） 机械设计领域

（ 5 ） 通讯领域

（ 6 ） 机器人领域

（ 8 ） 图像处理领域

（ 9 ） 生物信息领域

（ 10 ） 医学领域

（ 11 ） 运筹学领域

.....

## 6.6.4 粒子群优化算法在车辆路径问题中的应用

### 1. 车辆路径问题（VRP）的模型

车辆路径问题：假定配送中心最多可以用  $K(k=1,2,\dots,K)$  辆  
车对  $L(i=1,2,\dots,L)$  客户进行运输配送， $d_i$  ( $i$  表示仓库) 每个  
车辆载重为  $b_k$  ( $k=1,2,\dots,K$ )，每个客户的需求为  $d_i$  ( $i$  表示仓库)

，客户  $i$  到客户  $j$  的运输成本为  $c_{ij}$ （可以是距离，时间，费用等）。定义如下变量：

客户  $i$  由车辆  $k$  配送

$$y_{ik} = \begin{cases} 1 & \text{客户 } i \text{ 由车辆 } k \text{ 配送} \\ 0 & \text{其他} \end{cases}$$

$$x_{ijk} = \begin{cases} 1 & \text{车辆 } k \text{ 从 } i \text{ 访问 } j \\ 0 & \text{其他} \end{cases}$$

## 6.6.4 粒子群优化算法在车辆路径问题中的应用

### 1. 车辆路径问题（VRP）的模型

则车辆路径问题的数学模型如下表示：

$$\min \sum_{k=1}^K \sum_{i=0}^L \sum_{j=0}^L c_{ij} x_{ijk} \quad (7.17a)$$

$$\sum_{i=1}^L d_i y_{ik} \leq b_k \quad \forall k \quad (7.17b) \quad \text{每辆车的能力约束}$$

$$\sum_{k=1}^K y_{ik} = 1 \quad \forall i \quad (7.17c) \quad \text{保证每个客户都被服务}$$

$$\sum_{i=1}^L x_{ijk} = y_{jk} \quad \forall j, k \quad (7.17d) \quad \text{保证客户是仅被一辆车访问}$$

$$\sum_{j=1}^L x_{ijk} = y_{ik} \quad \forall i, k \quad (7.17e) \quad \text{保证客户是仅被一辆车访问}$$

$$\sum_{i,j \in S \times S} x_{ijk} \leq |S| - 1 \quad S \in \{1, 2, \dots, L\} \quad \forall k \quad (7.17f) \quad \text{消除子回路}$$

$$x_{ijk} = 0 \text{ 或 } 1 \quad \forall i, j, k \quad (7.17g) \quad \text{表示变量的取值范围}$$

$$y_{ik} = 0 \text{ 或 } 1 \quad \forall i, k \quad (7.17h) \quad \text{表示变量的取值范围}$$

## 6.6.4 粒子群优化算法在车辆路径问题中的应用

### 2. 编码与初始种群

- 对这类组合优化问题，编码方式、初始解的设置对问题的求解都有很大的影响。
- 采用常用的自然数编码方式。
- 对于  $K$  辆车和  $L$  个客户的问题，用从 1 到  $L$  的自然数随机排列来产生一组解  $(x_1, x_2, \dots, x_L)$ 。然后分别用节约法或者最近插入法构造初始解。

## 6.6.4 粒子群优化算法在车辆路径问题中的应用

- ❑ (3) 实验结果
- ❑ 粒子群优化算法的各个参数设置如下：
- ❑ 种群规模： 50
- ❑ 迭代次数： 1000
- ❑  $c_1$  的初始值为 1，随迭代的进行，线性减小到 0
- ❑  $C_2=c_3=1.4$
- ❑  $V_{\max}<1000$
- ❑ 优化结果及其与遗传算法的比较如表 6.4 所示。



# 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
- 6.5 群智能算法产生的背景
- 6.6 粒子群优化算法及其应用
- 6.7 蚁群算法及其应用

## 6.7 蚁群算法及其应用

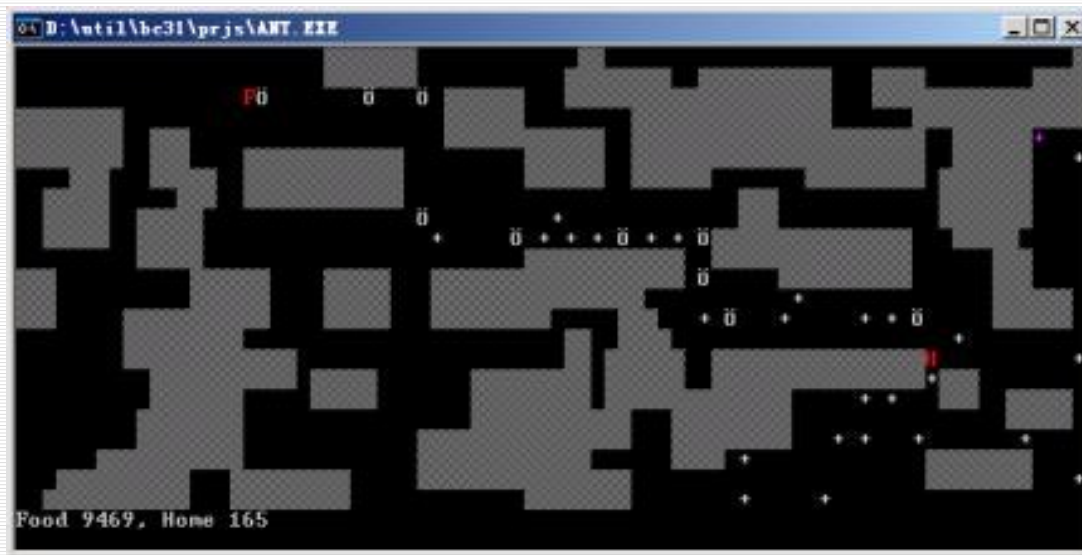
### 产生背景

- 20 世纪 90 年代初，意大利科学家 Marco Dorigo 等受蚂蚁觅食行为的启发，提出蚁群算法 (Ant Colony Optimization, ACO)。
- 一种应用于组合优化问题的启发式搜索算法。
- 在解决离散组合优化方面具有良好的性能。

## 6.7 蚁群算法及其应用

### 基本思想

- 信息素跟踪：按照一定的概率沿着信息素较强的路径觅食。
- 信息素遗留：会在走过的路上会释放信息素，使得在一定的范围内的其他蚂蚁能够觉察到并由此影响它们的行为。



## 6.7 蚁群算法及其应用

- (1) 环境：有障碍物、有其他蚂蚁、有信息素。
- (2) 觅食规则：范围内寻找是否有食物，否则看是否有信息素，每只蚂蚁都会以小概率犯错。
- (3) 移动规则：都朝信息素最多的方向移动，无信息素则继续朝原方向移动，且有随机的小的扰动，有记忆性。
- (4) 避障规则：移动的方向如有障碍物挡住，蚂蚁会随机选择另一个方向。
- (5) 信息素规则：越靠近食物播撒的信息素越多，越离开食物播撒的信息素越少。

## 6.7 蚁群算法及其应用



### 6.7.1 基本蚁群算法模型



### 6.7.2 蚁群算法的参数选择

## 6.7.1 基本蚁群算法模型

蚁群优化算法的第一个应用是著名的旅行商问题。

旅行商问题（ **Traveling Salesman Problem** , **TSP** ）：

在寻求单一旅行者由起点出发，通过所有给定的需求点之后，最后再回到原点的最小路径成本。

蚂蚁搜索食物的过程：

通过个体之间的信息交流与相互协作最终找到从蚁穴到食物源的最短路径。

旅行商问题

阐明

蚁群系统模型

## 6.7.1 基本蚁群算法模型

### 蚁群系统的模型

$m$  是蚁群中蚂蚁的数量

$d_{xy}(x, y = 1, \dots, n)$  表示元素 (城市) 和元素 (城市) 之间

的距离

$\eta_{xy}(t)$

表示能见度, 称为启发信息函数, 等于距离的倒数, 即

$b_x(t)$

的倒数, 即

$$m = \sum_{x=1}^n b_x(t)$$

$\tau_{xy}(t)$

表示  $t$  时刻位于城市  $x$  的蚂蚁的个数,

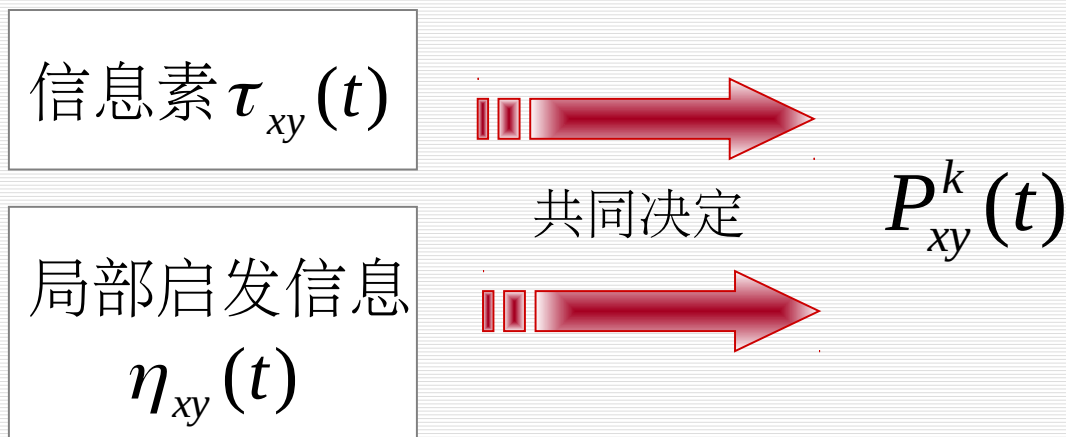
表示  $t$  时刻在  $xy$  连线上残留的信息素, 初始时  $\tau_{xy}(0) = C(const)$

始时

## 6.7.1 基本蚁群算法模型

### 蚁群系统的模型

$P_{xy}^k(t)$  表示在  $t$  时刻蚂蚁  $k$  选择从元素 (城市)  $x$  转移到元素 (城市)  $y$  的概率, 也称为随机比例规则。





## 6.7.1 基本蚁群算法模型

$P_{xy}^k(t)$  表示如下:

$$P_{xy}^k(t) = \begin{cases} \frac{|\tau_{xy}(t)|^\alpha |\eta_{xy}(t)|^\beta}{\sum_{y \in allowed_k(x)} |\tau_{xy}(t)|^\alpha |\eta_{xy}(t)|^\beta} & \text{if } y \in allowed_k(x) \\ 0 & \text{其他} \end{cases} \quad (7.18)$$

其中:  
 $allowed_k(x) = \{0, 1, \dots, n-1\} - tabu_k(x)$

$tabu_k(x) (k = 1, 2, \dots, m)$   
的城市

表示蚂蚁  $k$  下一步允许选择

记录蚂蚁  $k$  当前所走过的城

市

## 6.7.1 基本蚁群算法模型

$P_{xy}^k(t)$  表示如下:

$$P_{xy}^k(t) = \begin{cases} \frac{|\tau_{xy}(t)|^\alpha |\eta_{xy}(t)|^\beta}{\sum_{y \in allowed_k(x)} |\tau_{xy}(t)|^\alpha |\eta_{xy}(t)|^\beta} & \text{if } y \in allowed_k(x) \\ 0 & \text{其他} \end{cases} \quad (7.18)$$

$\alpha$ 值越大	该蚂蚁越倾向于选择其它蚂蚁经过的路径，该状态转移概率越接近于贪婪规则。
当 $\alpha = 0$ 时	不再考虑信息素水平，算法就成为有多重起点的随机贪婪算法。
当 $\rho = 0$ 时	算法就成为纯粹的正反馈的启发式算法。

## 6.7.1 基本蚁群算法模型

用参数  $1-\rho$  表示信息素消逝程度，蚂蚁完成一次循环，各路径上信息素浓度消散规则为：

$$\tau_{xy}(t) = \rho\tau_{xy}(t) + \Delta\tau_{xy}(t)$$

( 7.19 )

蚁群的信息素浓度更新规则为：

$$\Delta\tau_{xy}(t) = \sum_{k=1}^m \Delta\tau_{xy}^k(t)$$

( 7.20 )

M. Dorigo 给出  $\Delta\tau_{xy}^k(t)$  的三种不同模型

# 6.7.1 基本蚁群算法模型

## 1. 蚂蚁圈系统 ( Ant-cycle System )

单只蚂蚁所访问路径上的信息素浓度更新规则为：

$$\Delta\tau_{xy}^k(t) = \begin{cases} \frac{Q}{L_k} & \text{若第}k\text{只蚂蚁在本次循环中从}x\text{到}y \\ 0 & \text{否则} \end{cases}$$

( 7.21 )

其中： $\tau_{xy}(t)$

为当前路径上的信息素

$\Delta\tau_{xy}(t)$

为路径 (  $x, y$  ) 上信息素的增量

$\Delta\tau_{xy}^k(t)$

第  $k$  只蚂蚁留在路径 (  $x, y$  ) 上的信息素的增量

$Q$

为常数

$L_k$

为优化问题的目标函数值，表示第  $k$  只蚂蚁在本次循环中所走路径的长度

## 6.7.1 基本蚁群算法模型

### 2. 蚂蚁数量系统 ( Ant-quantity System )

$$\Delta\tau_{xy}^k(t) = \begin{cases} \frac{Q}{d_k} & \text{若第}k\text{只蚂蚁在本次循环中从}x\text{到}y \\ 0 & \text{否则} \end{cases} \quad (7.22)$$

### 3. 蚂蚁密度系统 ( Ant-density System )

$$\Delta\tau_{xy}^k(t) = \begin{cases} Q & \text{若第}k\text{只蚂蚁在本次循环中从}x\text{到}y \\ 0 & \text{否则} \end{cases} \quad (7.23)$$

# 6.7.1 基本蚁群算法模型

## 三种模型比较

效果最好，通常作为蚁群优化算法的基本模型。

蚂蚁圈系统

利用的是全局信息  $Q/L_k$ ，即蚂蚁完成一个循环后，更新所有路径上的信息。

蚂蚁数量系统

利用的是局部信息  $Q/d_{xy}$ ，即蚂蚁每走一步都要更新残留信息素的浓度。

蚂蚁密度系统

利用的是局部信息  $Q$ ，即蚂蚁每走一步都要更新残留信息素的浓度。

## 6.7.1 基本蚁群算法模型

### 全局信息更新方法

优点:

- 保证了残留信息素不至于无限累积;
- 如果路径没有被选中, 那么上面的残留信息素会随时间的推移而逐渐减弱, 这使算法能“忘记”不好的路径;
- 即使路径经常被访问也不至于因为  $\Delta\tau_{xy}^k(t)$  的累积, 而产生使期望值的作用无法体现;
- $\Delta\tau_{xy}^k(t) \gg \eta_{xy}(t)$  充分体现了算法中全局范围内较短路径 (较好解) 的生存能力;
- 加强了信息正反馈性能;
- 提高了系统搜索收敛的速度。

## 6.7.2 蚁群算法的参数选择

### 信息素启发因子 $\tau$

- 反映了蚁群在路径搜索中随机性因素作用的强度；
- $\tau$  值越大，蚂蚁选择以前走过的路径的可能性越大，搜索的随机性减弱；
- 当  $\tau$  过大时会使蚁群的搜索过早陷于局部最优。

### 期望值启发式因子 $\eta$

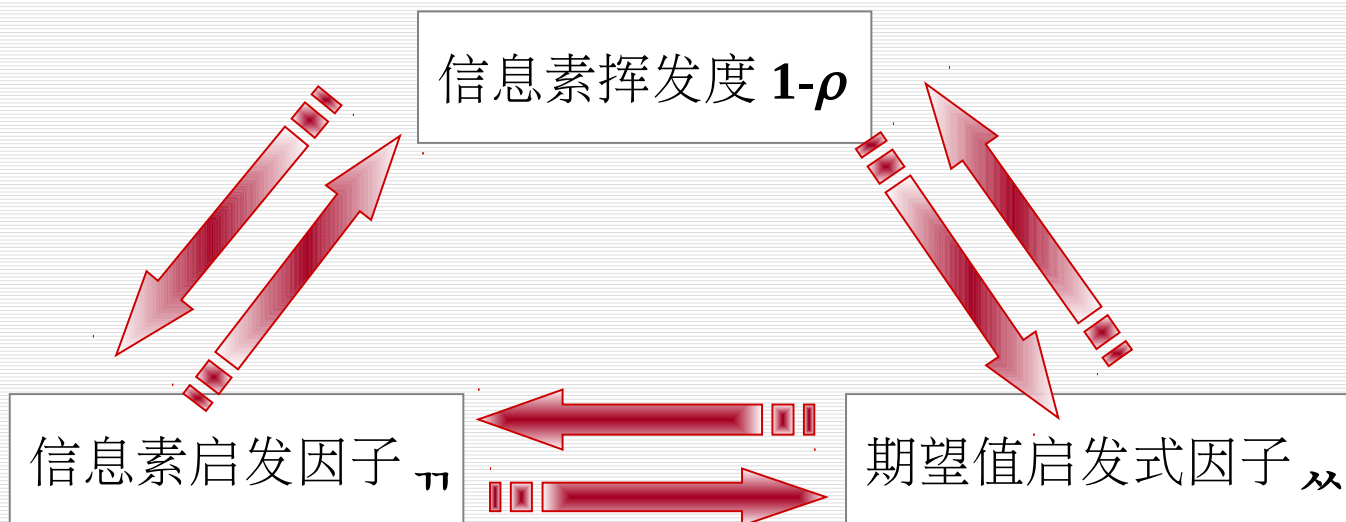
- 反映了蚁群在路径搜索中先验性、确定性因素作用的强度；
- $\eta$  值越大，蚂蚁在某个局部点上选择局部最短路径的可能性越大；
- 虽然搜索的收敛速度得以加快，但蚁群在最优路径的搜索过程中随机性减弱，易于陷入局部最优。



## 6.7.2 蚁群算法的参数选择

### 信息素挥发度 $1-\rho$

- 当要处理的问题规模比较大时，会使那些从来未被搜索到的路径（可行解）上的信息量减小到接近于 0，因而降低了算法的全局搜索能力；
- 而且当  $1-\rho$  过大时，以前搜索过的路径被再次选择的可能性过大，也会影响到算法的随机性能和全局搜索能力；
- 反之，通过减小信息素挥发度  $1-\rho$  虽然可以提高算法的随机性能和全局搜索能力，但又会使算法的收敛速度降低。



## 6.7.3 蚁群算法的应用

### 果

柔性作业车间调度问题：某加工系统有 6 台机床，要加工 4 个工件，每个工件有 3 道工序，如表 7.3 所示。

## 6.7.3 蚁群算法的应用

表 7.3 柔性作业车间调度事例

工序选择 <sup>↗</sup>		加工机床及加工时间 <sup>↗</sup>					
		1 <sup>↗</sup>	2 <sup>↗</sup>	3 <sup>↗</sup>	4 <sup>↗</sup>	5 <sup>↗</sup>	6 <sup>↗</sup>
$J_1$ <sup>↗</sup>	$p_{11}$ <sup>↗</sup>	2 <sup>↗</sup>	3 <sup>↗</sup>	4 <sup>↗</sup>	<sup>↗</sup>	<sup>↗</sup>	<sup>↗</sup>
	$p_{12}$ <sup>↗</sup>	<sup>↗</sup>	3 <sup>↗</sup>	<sup>↗</sup>	2 <sup>↗</sup>	4 <sup>↗</sup>	<sup>↗</sup>
	$p_{13}$ <sup>↗</sup>	1 <sup>↗</sup>	4 <sup>↗</sup>	5 <sup>↗</sup>	<sup>↗</sup>	<sup>↗</sup>	<sup>↗</sup>
$J_2$ <sup>↗</sup>	$p_{21}$ <sup>↗</sup>	3 <sup>↗</sup>	<sup>↗</sup>	5 <sup>↗</sup>	<sup>↗</sup>	2 <sup>↗</sup>	<sup>↗</sup>
	$p_{22}$ <sup>↗</sup>	4 <sup>↗</sup>	3 <sup>↗</sup>	<sup>↗</sup>	6 <sup>↗</sup>	<sup>↗</sup>	<sup>↗</sup>
	$p_{23}$ <sup>↗</sup>	<sup>↗</sup>	<sup>↗</sup>	4 <sup>↗</sup>	<sup>↗</sup>	7 <sup>↗</sup>	11 <sup>↗</sup>
$J_3$ <sup>↗</sup>	$p_{31}$ <sup>↗</sup>	5 <sup>↗</sup>	6 <sup>↗</sup>	<sup>↗</sup>	<sup>↗</sup>	<sup>↗</sup>	<sup>↗</sup>
	$p_{32}$ <sup>↗</sup>	<sup>↗</sup>	4 <sup>↗</sup>	<sup>↗</sup>	3 <sup>↗</sup>	5 <sup>↗</sup>	<sup>↗</sup>
	$p_{33}$ <sup>↗</sup>	<sup>↗</sup>	<sup>↗</sup>	13 <sup>↗</sup>	<sup>↗</sup>	9 <sup>↗</sup>	12 <sup>↗</sup>
$J_4$ <sup>↗</sup>	$p_{41}$ <sup>↗</sup>	9 <sup>↗</sup>	<sup>↗</sup>	7 <sup>↗</sup>	9 <sup>↗</sup>	<sup>↗</sup>	<sup>↗</sup>
	$p_{42}$ <sup>↗</sup>	<sup>↗</sup>	6 <sup>↗</sup>	<sup>↗</sup>	4 <sup>↗</sup>	<sup>↗</sup>	5 <sup>↗</sup>
	$p_{43}$ <sup>↗</sup>	1 <sup>↗</sup>	<sup>↗</sup>	3 <sup>↗</sup>	<sup>↗</sup>	<sup>↗</sup>	3 <sup>↗</sup>

## 6.7.3 蚁群算法的应用

### 果

由图 7.6 可以看出机器 6 并没有加工任何工件。分析其原因因为它虽然可以加工工序  $p_{23}$  ,  $p_{33}$  ,  $p_{42}$  ,  $p_{43}$  但从表 7.3 可知机器 6 的加工时间大于其他可加工机器, 特别是  $p_{23}$  ,  $p_{33}$  的加工时间, 因此机器 6 并未分到任何加工任务。

## 6.7.3 蚁群算法的应用

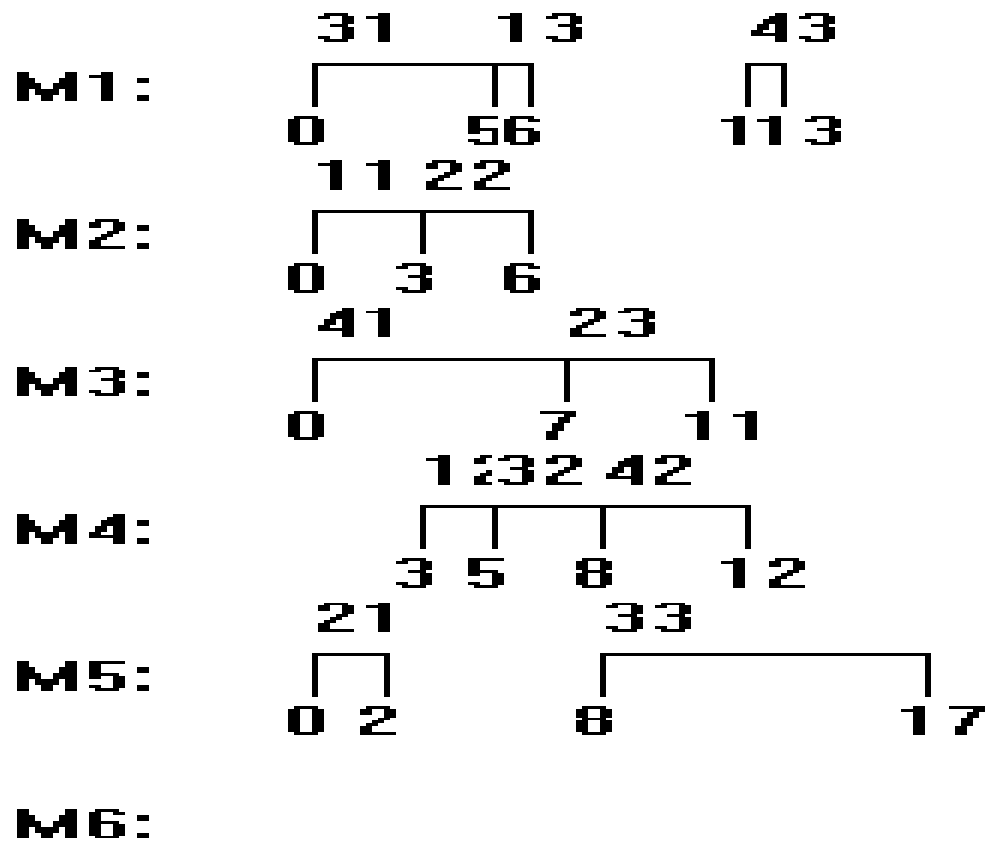
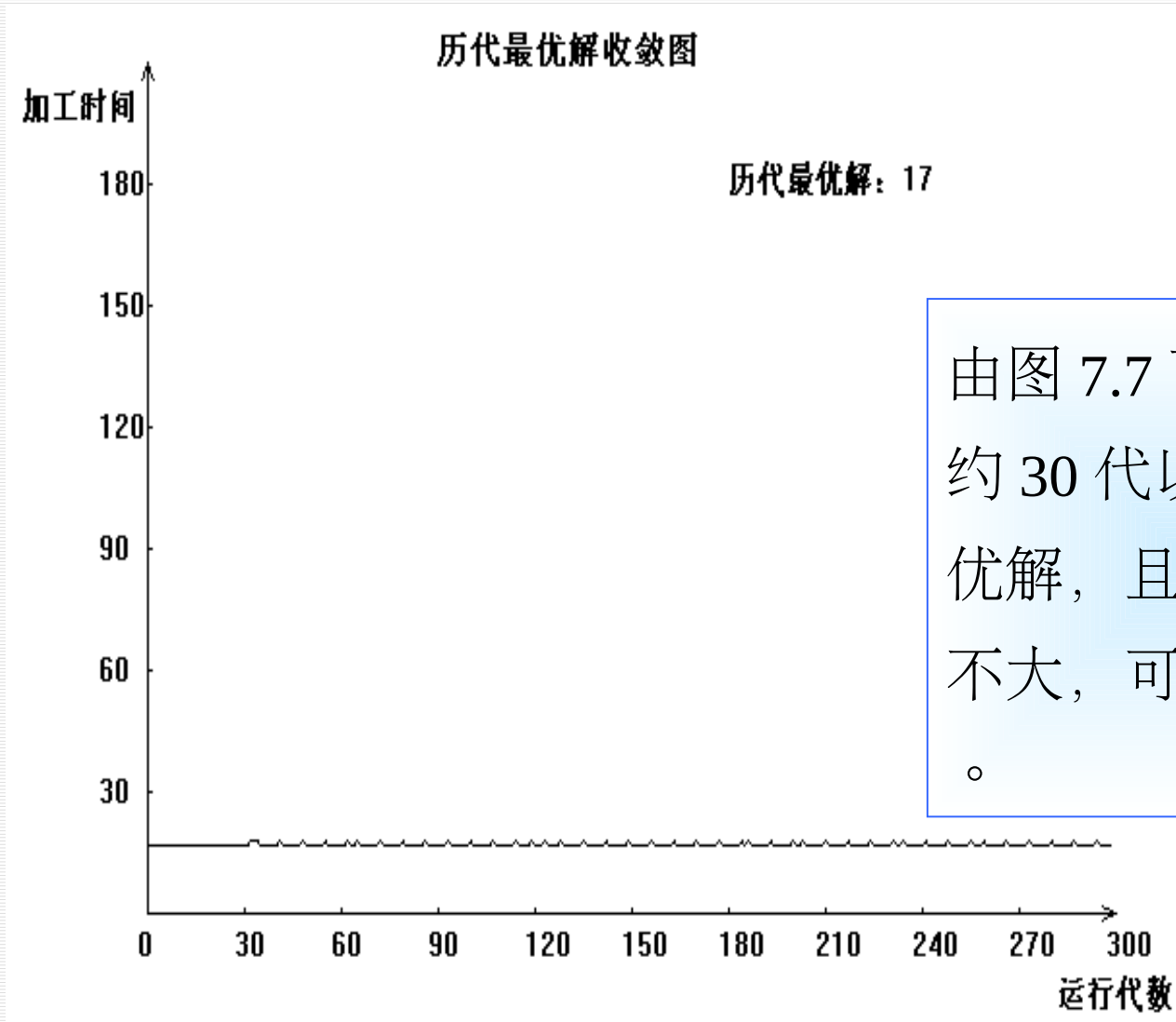


图 7.6 最优解甘特图

## 6.7.3 蚁群算法的应用



由图 7.7 可知，算法在大约 30 代以前就收敛到最优解，且各代最优解相差不大，可见算法较为稳定。

图 7.7 历代最优解收敛图



**THE END**

