第 7 章　群智能算法及其应用

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

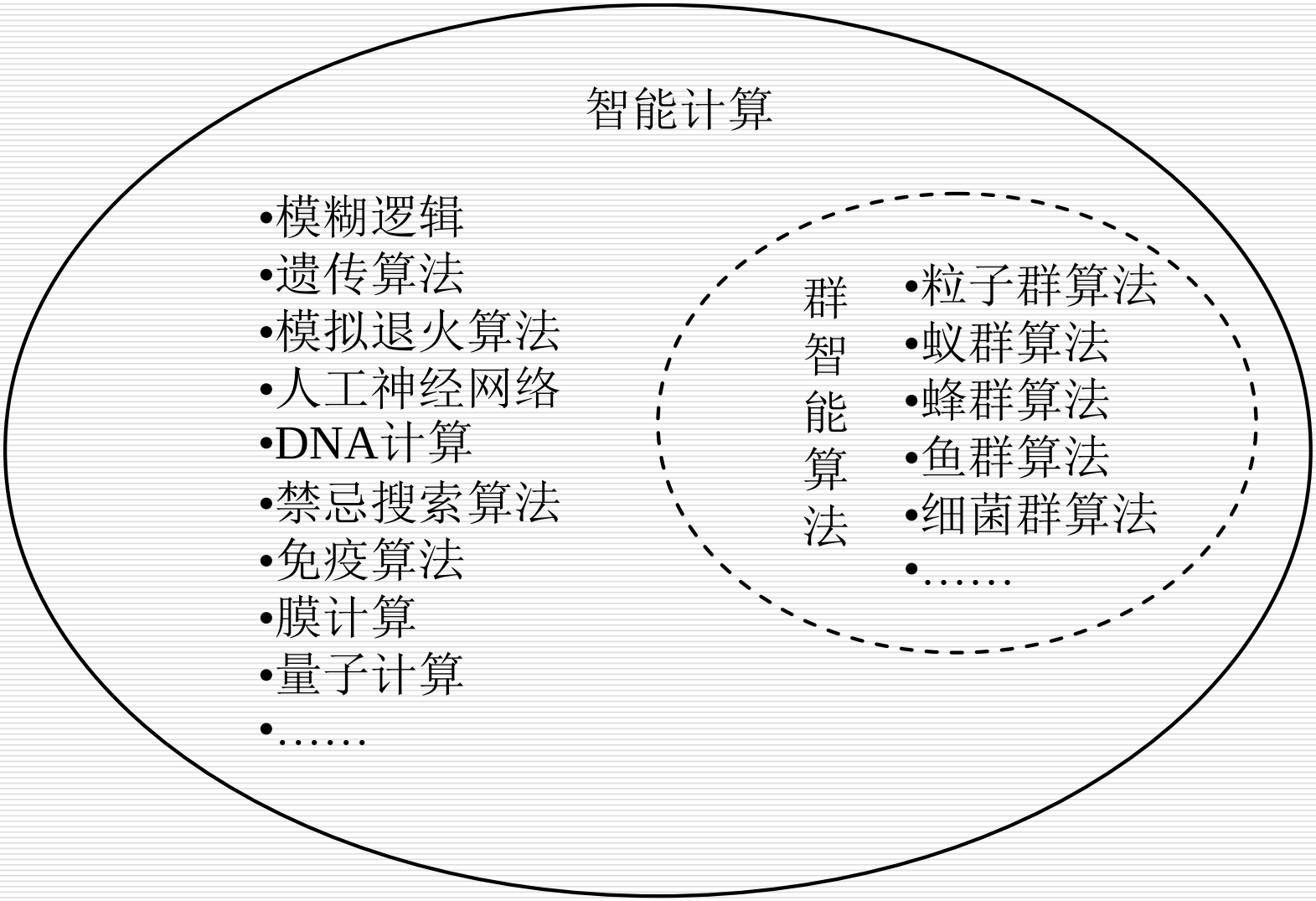
第7章 群智能算法及其应用

- ✓ 7.1 群智能算法产生的背景
- ☐ 7.2 粒子群优化算法
- ☐ 7.3 量子粒子群优化算法
- ☐ 7.4 粒子群优化算法的应用
- ☐ 7.5 基本蚁群算法
- ☐ 7.6 改进蚁群算法
- ☐ 7.7 蚁群算法的应用

7.1 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、……

7.1 群智能算法产生的背景



第7章 群智能算法及其应用

- 7.1 群智能算法产生的背景
- ✓ 7.2 粒子群优化算法
- 7.3 量子粒子群优化算法
- 7.4 粒子群优化算法的应用
- 7.5 基本蚁群算法
- 7.6 改进蚁群算法
- 7.7 蚁群算法的应用

7.2 粒子群优化算法

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。

7.2 粒子群优化算法

- **7.2.1 粒子群优化算法的基本原理**
- **7.2.2 粒子群优化算法的参数分析**

7.2.1 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。另个一是整个种群目前找到的最优解，这个解称为全局极值。

7.2.1 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, …, m)个粒子进行定义：

- $x^i(k)=\left[x_1^i \quad x_2^i \quad \mathrm{L} \quad x_n^i\right]^T$：表示搜索空间中粒子的当前位置。

- $p^i(k)=\left[p_1^i \quad p_2^i \quad \mathrm{L} \quad p_n^i\right]^T$：表示该粒子至今所获得的具有最优适应度 $f_p^i(k)$ 的位置。

- $v^i(k)=\left[v_1^i \quad v_2^i \quad \mathrm{L} \quad v_n^i\right]^T$：表示该粒子的搜索方向。

7.2.1 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 $p^i(k)=\left[\begin{array}{cccc} p_1^i & p_2^i & \mathrm{L} & p_n^i \end{array}\right]^T$，群体经历过的最优位置(gbest)记为 $p^g(k)=\left[\begin{array}{cccc} p_1^g & p_2^g & \mathrm{L} & p_n^g \end{array}\right]^T$，则基本的PSO算法为：

$$v_j^i(k+1)=\omega(k)v_j^i(k)+\varphi_1 rand(0,\mathrm{a}_1)\left(p_j^i(k)-x_j^i(k)\right)+\varphi_2 rand(0,\mathrm{a}_2)\left(p_j^g(k)-x_j^i(k)\right)$$

——（7.1a）

$$x_j^i(k+1)=x_j^i(k)+v_j^i(k+1)$$

——（7.1b）

$i=1,2,\mathrm{L}\ ,m;\qquad j=1,2,\mathrm{L}\ ,n$

其中，ω 是惯性权重因子。φ_1 ，φ_2 是加速度常数，均为非负值。$rand(0,\mathrm{a}_1)$和$rand(0,\mathrm{a}_2)$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。

7.2.1 粒子群优化算法的基本原理

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

——（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。
- φ_1 ，φ_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。
- 随机系数增加搜索方向的随机性和算法多样性。

7.2.1 粒子群优化算法的基本原理

基于学习率φ_1,φ_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。

7.2.1 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P_i。

7.2.1 粒子群优化算法的基本原理

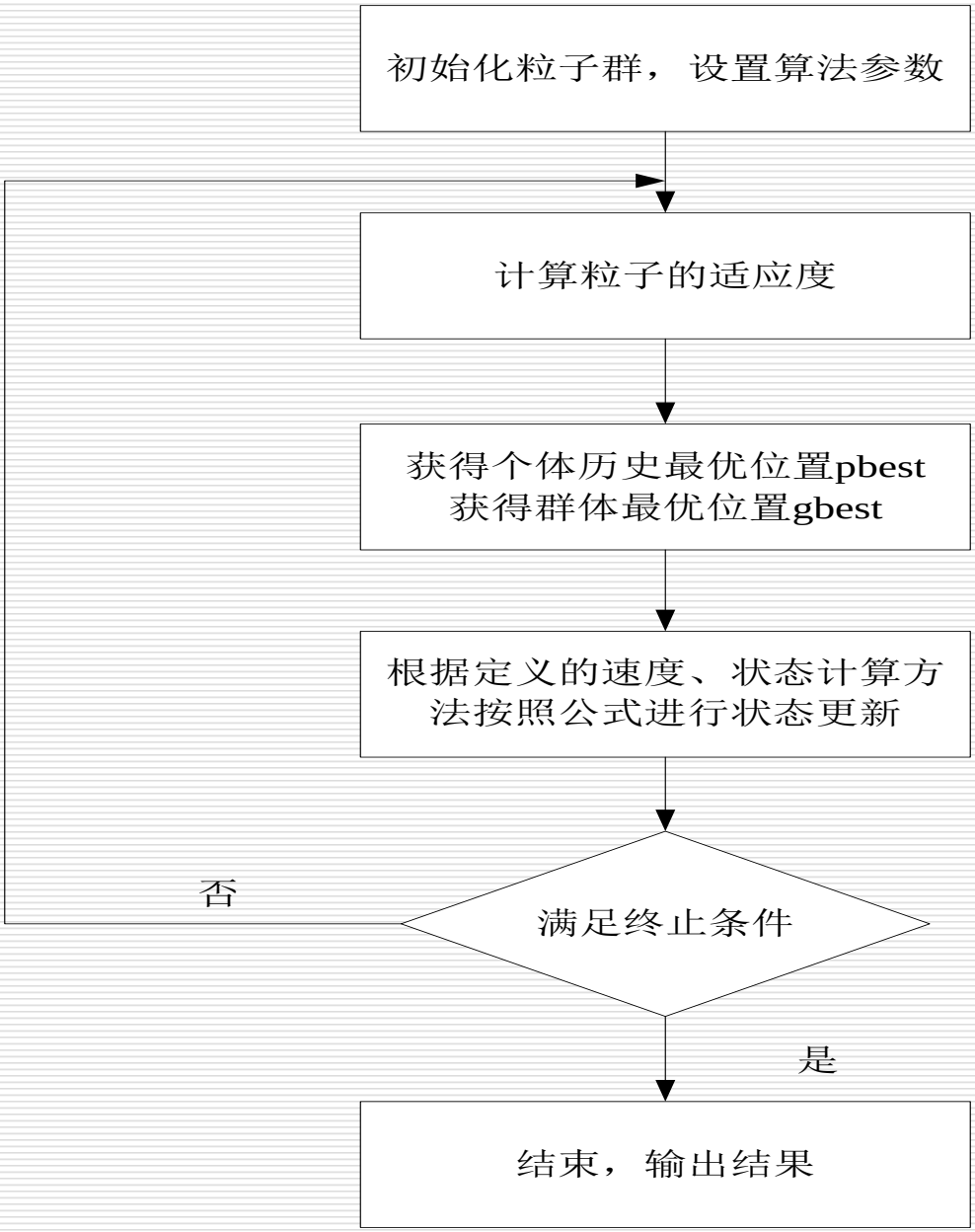
粒子群优化算法的流程：

（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。

7.2.1 粒子群优化算法流程图



7.2.2 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ω，加速度φ_1，φ_2，最大速度Vmax， 最大代数Gmax。

（1）最大速度Vmax

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

（2）权重因子

3个权重因子：惯性权重ω，加速度φ_1，φ_2 。

7.2.2 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

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

（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。

7.2.2 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

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

（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。

7.2.2 粒子群优化算法的参数分析

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

（3）没有第2部分，即 φ_1=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。

7.2.2 粒子群优化算法的参数分析

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

（4）没有第3部分，即 φ_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。

7.2.2 粒子群优化算法的参数分析

3. 参数设置

早期的实验：ω 固定为1.0，φ_1和φ_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，φ_1和φ_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 ω。

第7章 群智能算法及其应用

- 7.1 群智能算法产生的背景
- 7.2 粒子群优化算法
- ✓ 7.3 量子粒子群优化算法
- 7.4 粒子群优化算法的应用
- 7.5 基本蚁群算法
- 7.6 改进蚁群算法
- 7.7 蚁群算法的应用

7.3 量子粒子群优化算法

产生背景

在量子力学中是没有确定的轨迹的，因为根据不确定性原理，位置向量x_i和速度向量v_i是不可能同时确定的。

J. Sun受到量子物理学的启发，于2004年提出了一种能够保证全局收敛的具有量子行为的量子粒子群优化(Quantum-behaved particle swarm optimization，QPSO)算法，并对算法的收敛性进行了分析。

7.3 量子粒子群优化算法

- **7.3.1 基本量子粒子群优化算法**
- **7.3.2 改进量子粒子群优化算法**

7.3.1 基本量子粒子群优化算法

1. 粒子不再被描述为位置向量 x_i和速度向量v_i，而是采用波函数来表示。

- 粒子的波函数为

$$\psi(x)=\frac{1}{\sqrt{L}}\exp(-\frac{\|p-x\|}{L}) \qquad (7.2)$$

- 其概率密度函数为

$$Q(x)=|\psi(x)|^2=\frac{1}{L}\exp(-2\frac{\|p-x\|}{L}) \qquad (7.3)$$

其中，p为每个粒子历史的最好位置。

- 参数L的取值定义为

$$L(t+1)=2\alpha\times|p-x(t)| \qquad (7.4)$$

L指出了微粒的搜索空间范围。

7.3.1 基本量子粒子群优化算法

2. 引入了mbest为所有微粒的中心

$$mbest = \sum_{i=1}^{M} \frac{p_i}{M} = (\sum_{i=1}^{M} \frac{p_{i1}}{M}, \sum_{i=1}^{M} \frac{p_{i2}}{M}, \ldots \sum_{i=1}^{M} \frac{p_{id}}{M}) \qquad (7.5)$$

其中，M是种群数目，p_i是第 i 个粒子的最好位置。

通过将所有粒子的中心mbest取代每个粒子的最好位置p，可以有效提高算法的全局搜索能力。

7.3.1 基本量子粒子群优化算法

- 参数L表示为

$$L(t+1)=2\beta\times|mbest-x(t)| \tag{7.6}$$

其中，β 为收敛系数，不同的β 影响算法的收敛速度，一般取β 的值为

$$\beta=\frac{(1.0-0.5)\times(MAXITER-T)}{MAXITER}+0.5 \tag{7.7}$$

其中，$MAXITER$为最大迭代次数。

- 由概率密度函数通过Monte Carlo算法计算得到

$$x(t)=p\pm\frac{L}{2\ln(\frac{1}{u})} \tag{7.8}$$

- 代入参数L，QPSO算法的进化方程为

$$x(t+1)=p\pm\beta|mbest-x(t)|\ln u \tag{7.9}$$

7.3.1 基本量子粒子群优化算法

量子粒子群优化算法的基本步骤：

Step1： 确定种群规模和粒子维数，初始化粒子群体；

Step2： 计算个体历史最优值(pbest)：计算每一个微粒的适应度值，通过和个体的历史最优值比较，如果当前值优于个体历史最优值，则把当前值替换为个体最优值(pbest)，否则不替换；

Step3： 计算所有微粒的适应值，与当前全局最优值(gbest)比较，若当前值优于全局最优值，则把当前值替换为全局最优值；

Step4： 计算所有粒子的重心（mbest）；根据公式(7.5)来更新所有粒子的重心（mbest）；

Step5: 根据进化方程(7.9)更新每个粒子的位置，产生新种群；

Step6： 粒子适应度满足收敛条件或者是达到最大迭代次数，则算法结束，否则跳转到步骤2继续迭代执行。

7.3.1 基本量子粒子群优化算法

优点：

相对于粒子群优化算法具有更好的收敛性和全局搜索能力。

缺点：

求解约束优化问题的时

① 会产生大量的不可行解

② 破坏种群的多样性

③ 导致算法陷入局部极值

7.3.2 改进量子粒子群优化算法

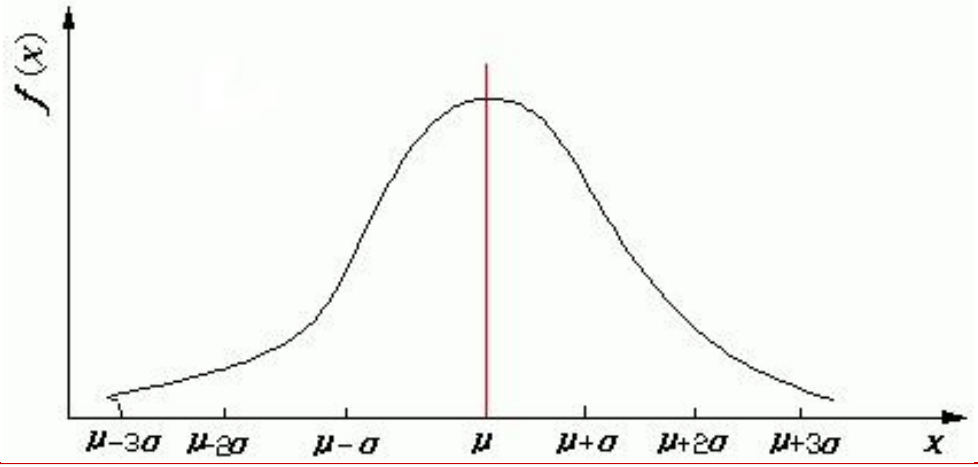
1. 三种概率分布函数

（1）正态分布

正态分布是具有两个参数 μ 和 σ^2 的连续型随机变量的分布，正态分布记作$N(\mu,\sigma^2)$。正态分布的概率密度函数为

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad (7.10)$$

其中，μ 是服从正态分布的随机变量的均值，σ^2是该随机变量的方差。



7.3.2 改进量子粒子群优化算法

1. 三种概率分布函数

（2）χ^2 分布

设随机变量$X_1, X_2, ...X_k$相互独立，并且都服从标准正态分布$N(0,1)$，则随机变量 $\chi^2 = X_1^2 + X_2^2 + ... + X_k^2$ 的概率密度为

$$f_{\chi^2}(x) = \begin{cases} \dfrac{1}{2^{\frac{k}{2}}\Gamma(\frac{k}{2})} x^{\frac{k}{2}-1} e^{-\frac{x}{2}}, & x \geq 0; \\ 0, & x < 0; \end{cases} \qquad (7.11)$$

7.3.2 改进量子粒子群优化算法

1. 三种概率分布函数

（3）t 分布

设随机变量X与Y独立，并且X服从标准正态分布 $N(0,1)$，Y服从自由度为 k 的 χ^2 分布，则随机变量 $t = X\sqrt{\frac{k}{Y}}$ 的概率密度为

$$f_t(z) = \frac{\Gamma(\frac{k+1}{2})}{\sqrt{k\pi}\Gamma(\frac{k}{2})}(1+\frac{z^2}{k})^{-\frac{k+1}{2}} \qquad (7.12)$$

7.3.2 改进量子粒子群优化算法

2. 变异操作

（1）生成符合正态分布的随机数

产生$U(0,1)$均匀分布的随机数30个，记为 $u_1, u_2, ..., u_{30}$ ；

由于 $E(u_i)=\frac{1}{2}$，$D(u_i)=\frac{1}{12}(i=1,2,...,30)$ 。根据中心极限定理，可以认为近似服从均值为$\frac{1}{12}*30=15$，方差为2.5的正态分布。

计算 $v=\frac{1}{\sqrt{2.5}}(u_1+u_2+\Lambda\ u_{30}-15)$，由中心极限定理，它可以看作是来自标准正态分布 的一个随机数。

7.3.2 改进量子粒子群优化算法

2. 变异操作

（2）对个体的每个维度产生在可行域区间内符合概率分布的可行解

① 正态分布

生成1个符合正态分布的随机数v，变换$x=\mu+\sigma v$由，正态分布的性质可知，它可以看作是来自正态分布$N(\mu, \sigma^2)$的一个随机数。取 $\mu=\frac{X_{\max it}-X_{\min it}}{2}$。$\sigma$为可变参数，用于控制解在可行域范围内的分布情况。可行解 $X'_{it}=u+\sigma v$。

② χ^2 分布

生成k个满足标准正态分布$N(0,1)$的随机数 (Y_1, Y_2, Λ, Y_k)，取$\chi^2=Y_1^2+Y_2^2+\ldots+Y_k^2$。$k$为可变参数，用于控制解在可行域范围内的分布情况。可行解$X'_{it}=\chi^2+\frac{X_{\max it}-X_{\min it}}{2}$。

③ t 分布

生成2个满足标准正态分布$N(0,1)$的随机数(Y_1, Y_2)，取 $\chi^2=Y_1^2+Y_2^2$，生成一个符合正态分布的随机数 X，取 $t=X\sqrt{\frac{2}{Y}}$。可行解 $X'_{it}=t+\frac{X_{\max it}-X_{\min it}}{2}$。

7.3.2 改进量子粒子群优化算法

2. 变异操作

（3）计算适应度

由前面所生成的个体 $X'=(x'_1,x'_2,x'_3,...x'_n)$ 在可行域区间内，符合概率分布。根据适应度公式计算个体的适应度$f'(x)$。

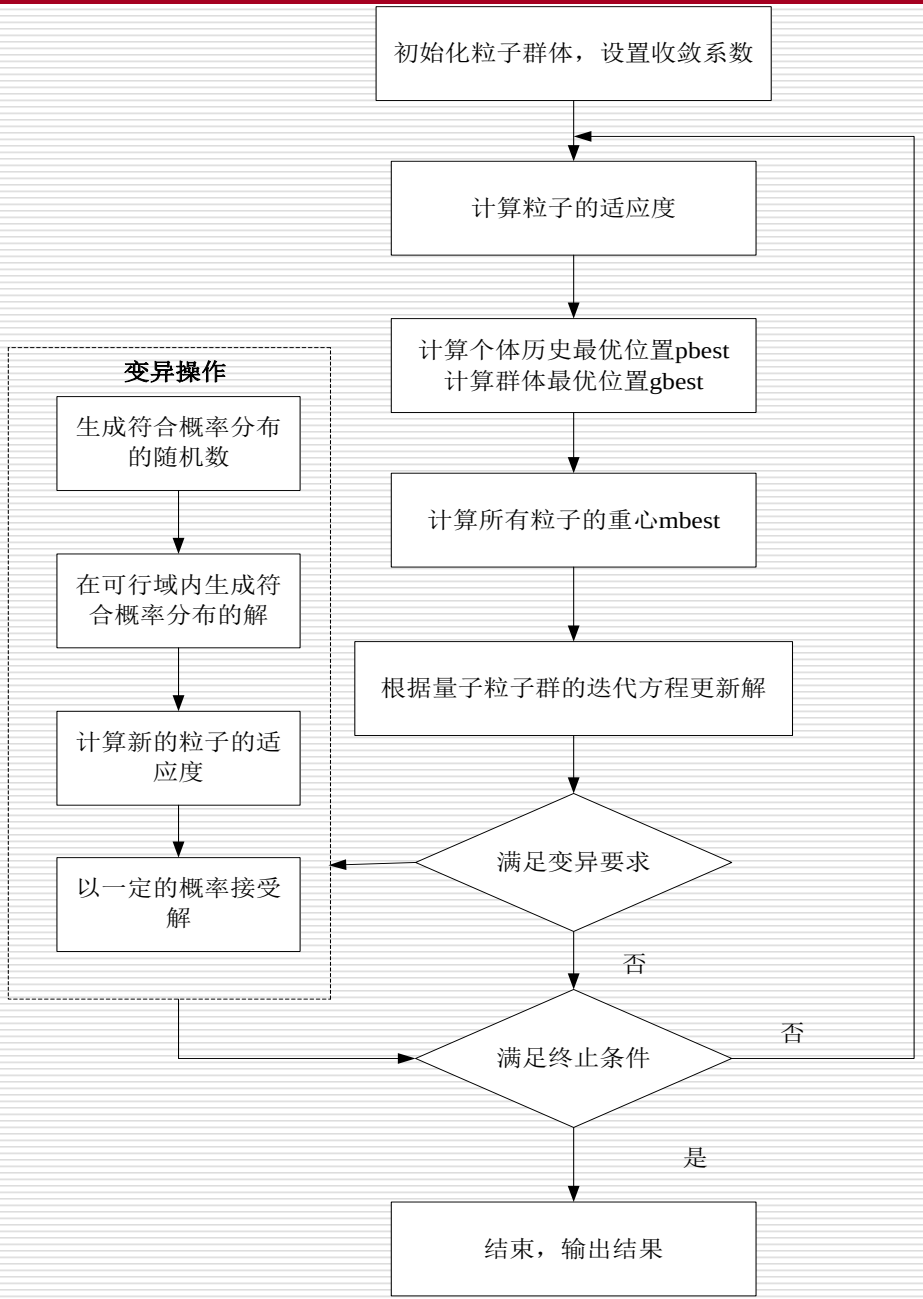
（4）以一定的概率接受解

计算动态变异率

$$p_m=\frac{f'(x)-f(x)}{f(x)}+p_{\min} \tag{7.13}$$

其中，$f(x)$为原个体的适应度。$f'(x)$ 为变异操作后个体的适应度。依照概率 $\min\{1,p_m\}>random[0,1]$接受解，即将原个体$X$替换为变异后的解$X$’。上式表明若$p_m$>1则表示$X$’是个极好解，这个解必定被接受。

基于概率分布的量子粒子群优化算法流程图



第7章 群智能算法及其应用

- 7.1 群智能算法产生的背景
- 7.2 粒子群优化算法
- 7.3 量子粒子群优化算法
- ✓ 7.4 粒子群优化算法的应用
- 7.5 基本蚁群算法
- 7.6 改进蚁群算法
- 7.7 蚁群算法的应用

7.4 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**

7.4.1 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：

（1）神经网络训练

（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

（7）经济领域

（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

……

7.4.2 粒子群优化算法在PID参数整定中的应用

典型的PID控制系统的控制量u与偏差$e=(R-y)$之间满足以下差分方程

$$u(n)=K_p\left[e(n)+\frac{1}{T_i}\sum_{k=0}^{n}e(k)T+T_d\frac{e(n)-e(n-1)}{T}\right] \qquad (7.14)$$

PID控制器就是通过调整K_p，T_i，T_d这3个参数来使系统的控制性能达到给定的要求。

从最优控制的角度，就是在K_p，T_i，T_d这3个变量的参数空间中，寻找最优的值使系统的控制性能达到最优。

为优化PID参数，可以选取如下函数作为评价控制性能的指标

$$Q=\int_0^{\infty}t\,|e(t)|\,dt \qquad (7.15)$$

7.4.2 粒子群优化算法在PID参数整定中的应用

1. 编码与初始种群

- 实数编码

- 在初始群体的生成上，首先根据经验估计出PID三个参数的取值范围，在此范围内采用随机生成的方式，使粒子群优化算法在整个可行解空间中进行搜索。

2. 适应度函数

- 由于PID参数优化是求目标函数Q的极小值问题，因而需要将极小值问题转换为极大值问题，适应度函数可以取为

$$F = \frac{1}{\int_0^{\infty} t\,|e(t)|\,dt} \tag{7.16}$$

7.4.2 粒子群优化算法在PID参数整定中的应用

举例

采用PID控制器对被控对象进行控制，假定控制对象具有二阶惯性加延迟的模型，其传递函数为: $H(s)=\dfrac{e^{-0.4s}}{(0.3s+1)^2}$ 。

假定采样周期选择为0.1s，根据经验K_p参数范围为（0，4），T_i参数范围为（0，1），T_d参数范围为（0，1）。

取粒子群种群规模为20，迭代次数为50，c_1的取值根据迭代的次数线性减小，初始值为1.5，最终值0.4。 $\varphi_1=\varphi_2=2$。

7.4.2 粒子群优化算法在PID参数整定中的应用

举例

PID参数粒子群优化算法寻优结果见表7.1所示。

为了说明粒子群优化算法的有效性，表中同时也给出了用单纯形法的寻优结果。

表7.1 优化结果及比较

算法	K_p	T_i	T_d	Q
粒子群优化算法	0.62932	0.59349	0.23715	4.84232
单纯形法	0.63057	0.59481	0.23703	4.86818

7.4.3 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$K(k=1,2\mathrm{L}\ K)$辆车对$L(i=1,2\mathrm{L}\ L)$个客户进行运输配送，$i=0$表示仓库。每个车辆载重为$b_k(k=1,2,\mathrm{L}\ K)$，每个客户的需求为$d_i(i=1,2,\mathrm{L}\ L)$，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：

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

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

7.4.3 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：

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

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

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

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

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

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

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

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

7.4.2 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$\mathbf{X}=(x_1,x_2,\Lambda\ ,x_L)$。然后分别用节约法或者最近插入法构造初始解。

7.4.2 粒子群优化算法在PID参数整定中的应用

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p=50，迭代次数N=1000，c_1的初始值为1，随着迭代的进行，线性减小到0，c_2=c_3=1.4，$|\mathbf{V}_{max}|$≤100。

表7.2 优化结果及其比较

实例	PSO		GA	
	best	dev(%)	best	dev(%)
A-n32-k5	829	5.73	818	4.34
A-n33-k5	705	6.65	674	1.97
A-n34-k5	832	6.94	821	5.52
A-n39-k6	872	6.08	866	5.35
A-n44-k6	1016	8.49	991	5.76
A-n46-k7	977	6.89	957	4.7
A-n54-k7	1205	3.26	1203	3.08
A-n60-k9	1476	9.01	1410	4.13
A-n69-k9	1275	10	1243	7.24
A-n80-k10	1992	12.98	1871	6.12

第7章 群智能算法及其应用

- 7.1 群智能算法产生的背景
- 7.2 粒子群优化算法
- 7.3 量子粒子群优化算法
- 7.4 粒子群优化算法的应用
- ✓ 7.5 基本蚁群算法
- 7.6 改进蚁群算法
- 7.7 蚁群算法的应用

7.5 基本蚁群算法

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。

7.5 基本蚁群算法

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。

7.5 基本蚁群算法

（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。

7.5 基本蚁群算法

- 7.5.1 基本蚁群算法模型
- 7.5.2 蚁群算法的参数选择

7.5.1 基本蚁群算法模型

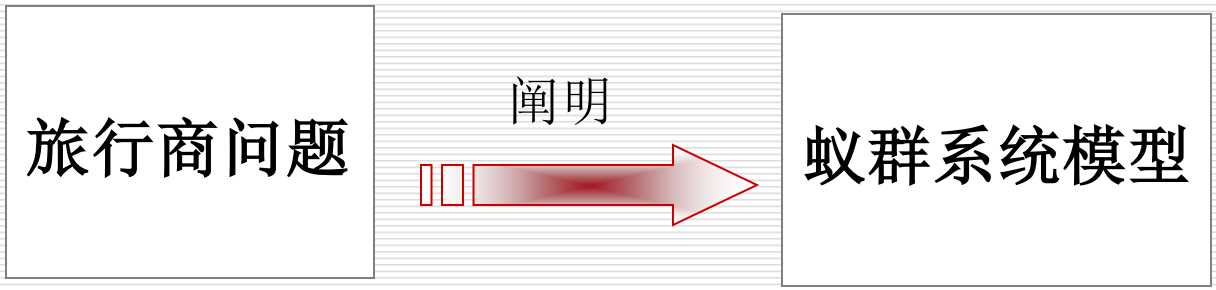
蚁群优化算法的第一个应用是著名的旅行商问题。

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。



7.5.1 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

$d_{xy}(x, y = 1, ..., n)$ 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 $\eta_{xy}(t) = \frac{1}{d_{xy}}$

$b_x(t)$ 表示t时刻位于城市x的蚂蚁的个数，$m = \sum_{x=1}^{n} b_x(t)$

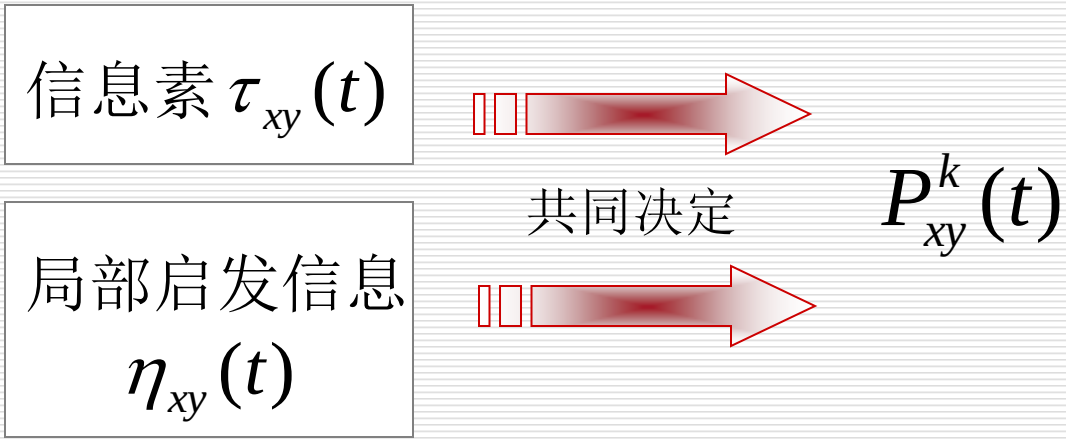
$\tau_{xy}(t)$ 表示t时刻在xy连线上残留的信息素，初始时刻，各条路径上的信息素相等即 $\tau_{xy}(0) = C(const)$

蚂蚁k在运动过程中，根据各条路径上的信息素决定转移方向。

7.5.1 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。



7.5.1 基本蚁群算法模型

$P_{xy}^{k}(t)$ 表示如下：

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

其中：

$allowed_k(x)=\{0,1,\ldots,n-1\}-tabu_k(x)$ 表示蚂蚁k下一步允许选择的城市

$tabu_k(x)(k=1,2,\ldots,m)$ 记录蚂蚁k当前所走过的城市

α 是信息素启发式因子，表示轨迹的相对重要性

7.5.1 基本蚁群算法模型

$P_{xy}^{k}(t)$ 表示如下：

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

其中：

α 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 $\beta = 0$时	算法就成为纯粹的正反馈的启发式算法。

7.5.1 基本蚁群算法模型

用参数1-ρ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：

$$\tau_{xy}(t) = \rho\tau_{xy}(t) + \Delta\tau_{xy}(t) \qquad (7.19)$$

蚁群的信息素浓度更新规则为：

$$\Delta\tau_{xy}(t) = \sum_{k=1}^{m} \Delta\tau_{xy}^{k}(t) \qquad (7.20)$$

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

7.5.1 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：

$$\Delta\tau_{xy}^{k}(t)=\begin{cases}\dfrac{Q}{L_k} & \text{若第}k\text{只蚂蚁在本次循环中从}x\text{到}y\\ 0 & \text{否则}\end{cases} \tag{7.21}$$

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

7.5.1 基本蚁群算法模型

2. 蚂蚁数量系统（Ant-quantity System）

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

3. 蚂蚁密度系统（Ant-density System）

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

7.5.1 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。

7.5.1 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$\Delta\tau_{xy}^{k}(t) >> \eta_{xy}(t)$ 使期望值的作用无法体现；
- 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。

7.5.2 蚁群算法的参数选择

信息素启发因子α

- 反映了蚁群在路径搜索中随机性因素作用的强度；
- α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α过大时会使蚁群的搜索过早陷于局部最优。

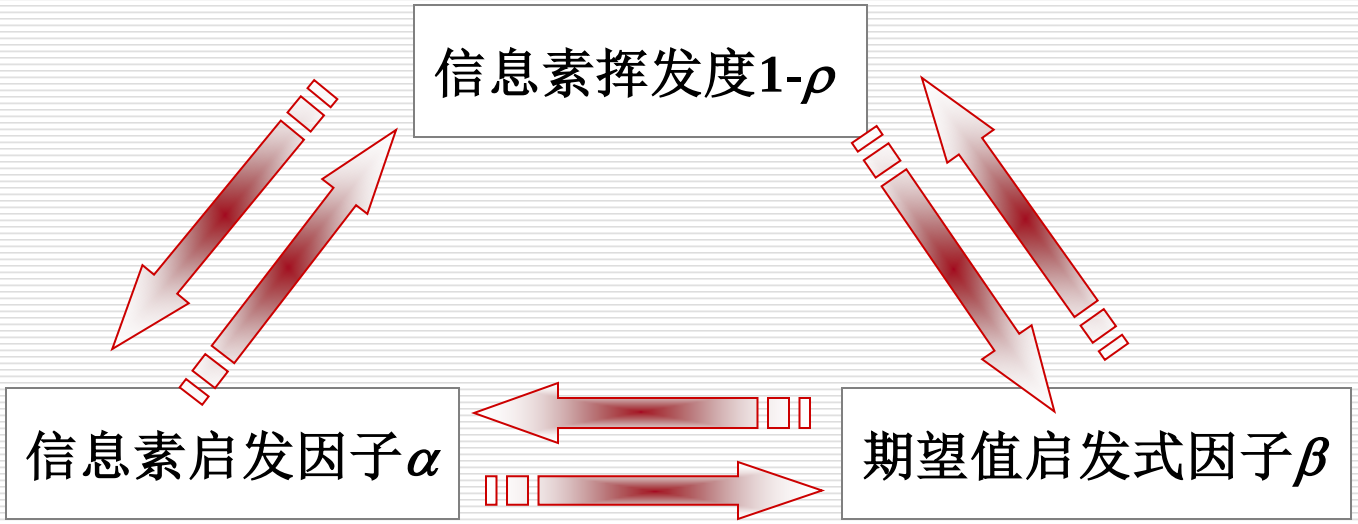
期望值启发式因子β

- 反映了蚁群在路径搜索中先验性、确定性因素作用的强度；
- β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。

7.5.2 蚁群算法的参数选择

信息素挥发度1-ρ

- 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-ρ 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-ρ 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。



第7章 群智能算法及其应用

- 7.1 群智能算法产生的背景
- 7.2 粒子群优化算法
- 7.3 量子粒子群优化算法
- 7.4 粒子群优化算法的应用
- 7.5 基本蚁群算法
- ✓ 7.6 改进蚁群算法
- 7.7 蚁群算法的应用

7.6 改进蚁群算法

- **7.6.1 蚂蚁-Q系统**
- **7.6.2 蚁群系统**
- **7.6.2 最大-最小蚂蚁系统**
- **7.6.2 自适应蚁群算法**

7.6.1 蚂蚁-Q系统

- 1995年，意大利学者M.Luca，M.Gambardella，M.Dorigo提出了蚂蚁–Q 系统（Ant-Q System）。

在ACA算法的基础上，进行了以下创新：

- 在解构造过程中提出了伪随机比例状态迁移规则；
- 信息素局部更新规则引入强化学习中的Q学习机制；
- 在信息素的全局更新中采用了精英策略。

7.6.1 蚂蚁-Q系统

$$y=\begin{cases}\arg\max\limits_{y\in allowed_k(x)}\left\{[HE(x,u)]^{\alpha}\cdot[AQ(x,u)]^{\beta}\right\} & if\ q\le q_0\\ Y & otherwise\end{cases} \quad (7.24)$$

根据式（7.25）计算概率分布：

$$P_k(x,y)=\begin{cases}\dfrac{HE(x,y)^{\alpha}\cdot AQ(x,y)^{\beta}}{\sum\limits_{y\in allowed_k(x)}\left\{HE(x,u)^{\alpha}\cdot AQ(x,u)^{\beta}\right\}} & if\ y\in allowed_k(x)\\ 0 & otherwise\end{cases} \quad (7.25)$$

AQ值按照如下规则进行更新：

$$AQ(x,y)\leftarrow(1-\alpha)AQ(x,y)+\alpha\left(\Delta AQ(x,y)+\gamma\cdot\max_{y\in allwoed_k(x)}AQ(x,u)\right) \quad (7.26)$$

$$\Delta AQ(x,y)=\begin{cases}\dfrac{w}{L_k} & if\ ant\ k\ goes\ from\ x\ to\ y\\ 0 & otherwise\end{cases} \quad (7.27)$$

7.6.2 蚁群系统

- 1996年，Gambardella和Dorigo又在Ant-Q算法的基础上，提出一种修正的蚁群算法，称之为蚁群系统(ant colony system, ACS)，该算法可以看成是Ant-Q算法的特例。

与ACA算法的不同之处：

- 蚂蚁选择城市时遵循的规则不同，这里使用的是所谓的状态转移规则:

$$S_k = \begin{cases} \arg\max\limits_{s \in j_k(r)} \left\{ [\tau(r,s)]^{\alpha} [\eta(r,s)]^{\beta} \right\} & q \le q_0 \\ S & \text{其他} \end{cases} \qquad (7.28)$$

其中：S_k是序号为k的蚂蚁所选中的下一个节点；q表示一个随机变量，q_0是一个适当选定的阈值。

7.6.2 蚁群系统

相比ACA算法，进行了以下创新：

- 主要不同在于蚂蚁选择下一城市之前，多进行了一次随机试验，将选择情况分成“利用已知信息”和“探索”两类。
- 还提出了所谓的精英策略(elitist strategy) 。
- 结果发现，对精英蚂蚁数而言有一个最优的范围：低于此范围，增加精英蚂蚁数可较早地发现更好的路径，高于此范围，精英蚂蚁会在搜索早期迫使寻优过程始终在次优解附近，导致性能变差。

7.6.3 最大-最小蚂蚁系统

- 最大-最小蚂蚁系统(Max-Min Ant System，MMAS)是德国学者Thomas Stutzle等在1997年提出的。

MMAS算法思想：

- 该算法在启动时将所有支路上的信息素浓度初始化为最大值τ_{max}；为了更好地利用历史信息，每次迭代后按挥发系数ρ降低信息素浓度，只有最佳路径上的支路才允许增加其信息素浓度并保持在高水平上，也就是用当前找到的最好解更新信息素来指引蚂蚁向更高质量的解空间搜索的贪婪策略。

7.6.3 最大-最小蚂蚁系统

$$\tau(x,y)=\rho\cdot\tau(x,y)+\Delta\tau(x,y)^{best} \qquad (7.29)$$

为了避免算法过早收敛于局部最优解，将各条路径可能的信息素浓度限制于$[\tau_{\min},\ \tau_{\max}]$。

为了在较长的运行时间里消除停滞现象

采用了让轨迹上信息素浓度的增加正比于$\tau_{\max}$和当前浓度$\tau(x,y)$之差的平滑机制，如式（7.30）所示，其中$0<\delta<1$。

$$\tau'(x,y)=\tau(x,y)+\delta(\tau_{\max}(x,y)-\tau(x,y)) \qquad (7.30)$$

7.6.4 自适应蚁群算法

自适应蚁群算法能根据判断搜索结果是否陷入局部收敛从而采用一种新的信息素更新策略，**自适应动态调整**陷入局部收敛的蚂蚁所经过路径上的信息素ρ和信息素强度Q，使得算法能更快的跳出局部收敛，防止“早熟”，同时对所有路径上的信息素**取值限定范围**，有利于算法的全局搜索。

7.6.4 自适应蚁群算法

1. 状态转移规则

第k只蚂蚁由节点r转移到节点s的概率按式（7.31）计算，所得的概率记为 P_{rs}^{k}：

$$P_{rs}^{k}=\begin{cases}\dfrac{\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta}}{\sum_{i\in p}\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta}} & if\,(p_{ij}<p_{i\max})\\ 0 & otherwise\end{cases} \tag{7.31}$$

其中：

τ_{ij} 表示节点r到节点 s（其中 表示第i个工件的第j道工序）的信息素

η_{ij} 表示节点r到节点s的可见度

p_{ij} 表示第i个工件的第j道工序

p_{imax} 表示第i个工件的最大工序

α、β 分别表示信息素和可见度的偏重系数

7.6.4 自适应蚁群算法

1. 状态转移规则

可见度 η_{ij} 由公式（7.32）来计算。

$$\eta_{ij} = \frac{1}{t_{wait} + c} \tag{7.32}$$

其中，t_{wait} 为在加工 p_{ij} 前的等待时间。

经过（7.31）式计算后，再用轮盘赌方法从工件集中选择一个节点,并记下这个节点的起止时间，以便计算等待时间和最后完成所有工件的加工时间。

7.6.4 自适应蚁群算法

2. 判断是否发生局部收敛

各代所有蚂蚁爬行完毕后对所搜索到的最优解进行判断，看是否陷入局部收敛，判断方法如下：

- 当连续几代最优蚂蚁搜索得到的路径相同时，算法即陷入了局部收敛
- 当连续几代的最优蚂蚁爬行路径总长度相同时算法陷入了局部最优

7.6.4 自适应蚁群算法

3. 自适应信息素挥发系数ρ

当算法陷入局部收敛时，ρ不再为常数，而是随着连续最优解相同的代数的增大而增大，表达式如下：

$$\rho=\begin{cases}\rho_0 & n\le n_0+1\\ 1-\dfrac{1-\rho_0}{n-n_0} & n>n_0+1\end{cases} \qquad (7.33)$$

其中ρ_0为初始挥发度，n为各代最优解连续相等的次数，n_0为大于1的整数，当$n>n_0+1$时ρ开始减小n越大ρ越小。算法具体实现时，ρ_0、n_0可以根据需要进行调节。

7.6.4 自适应蚁群算法

4.自适应信息素强度$Q(n)$

当算法陷入局部收敛时采用时变函数$Q(n)$来代替基本蚁群算法中调整信息素 $\Delta\tau_{ij}^{k}=Q/L_k$中为常数项的信息素强度$Q$，即选择 $\Delta\tau_{ij}^{k}=Q/L_k$ ，随着人工蚂蚁搜索过程动态的调整，$Q(n)$如下所示：

$$Q(n)=\begin{cases}Q_0 & n\le n_0\\ -Q_0*(n-n_0) & n>n_0\end{cases} \qquad (7.34)$$

其中，Q_0为初始信息素强度，可以根据需要调整。

7.6.4 自适应蚁群算法

5. 改进的信息素更新策略

信息素更新策略存在多种方式：

- 走过的全部路径上的信息素进行更新，导致算法获得的结果振荡, 不易收敛
- 更新搜索到最优边上的信息素，则进一步加强了蚁群算法的正反馈作用，导致搜索过程迅速陷入局部最优解

7.6.4 自适应蚁群算法

5. 改进的信息素更新策略

记l为每代最优解对应的蚂蚁，蚂蚁总数为m。

（1）当算法未陷入局部最优时，采用全局更新和局部更新结合的策略，其中ρ和Q均为初始值：

Step1：全局更新，计算所有蚂蚁经过路径上的信息素增量：

$$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}\qquad \Delta\tau_{ij}^{k}=Q(n)/L_k\qquad Q(n)=Q_0\qquad k=1,\ldots,m \tag{7.35}$$

Step2：局部更新，如果该代最优解为历代最优解，则调整蚂蚁l经过路径上的信息素增量：

$$\Delta\tau_{ij(new)}^{(l)}=\Delta\tau_{ij(old)}^{(l)}+\Delta\tau_{ij}^{l}\qquad \Delta\tau_{ij}^{l}=Q(n)/L_l\qquad Q(n)=Q_0 \tag{7.36}$$

Step3：更新所有蚂蚁经过路径上的信息素：

$$\tau_{ij(new)}=(1-\rho)\tau_{ij(old)}+\Delta\tau_{ij}\qquad \rho=\rho_0 \tag{7.37}$$

7.6.4 自适应蚁群算法

5. 改进的信息素更新策略

记l为每代最优解对应的蚂蚁，蚂蚁总数为m。

（2）当算法陷入局部最优时,仅采用全局更新策略：

Step1：计算除最优蚂蚁l外所有其他蚂蚁经过路径上的信息素增量：

$$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}\quad \Delta\tau_{ij}^{k}=Q(n)/L_k\quad Q(n)=Q_0\quad k=1,\ldots,l-1,l+1,\ldots,m \tag{7.38}$$

Step2：计算蚂蚁l经过的路径上的信息素增量：

$$\Delta\tau_{ij}^{(l)}=\Delta\tau_{ij}^{l}\qquad \Delta\tau_{ij}^{l}=Q(n)/L_l\qquad Q(n)=-Q_0*(n-n_0) \tag{7.39}$$

Step3：更新除蚂蚁l外所有其他蚂蚁经过路径上的信息素：

$$\tau_{ij}=(1-\rho)\tau_{ij}+\Delta\tau_{ij}\qquad \rho=\rho_0 \tag{7.40}$$

Step4：更新蚂蚁l经过路径上的信息素：

$$\tau_{ij}^{(l)}=(1-\rho)\tau_{ij}^{(l)}+\Delta\tau_{ij}^{(l)}\qquad \rho=1-\frac{1-\rho_0}{n-n_0} \tag{7.41}$$

7.6.4 自适应蚁群算法

6. 限定信息素的范围

通过缩小各路径信息素的差距，可以使算法有更好的全局收敛性。对各路径上的信息素进行限定，以防止某些路径上的信息素过大或过小而影响算法的全局收敛性。

7.6.4 自适应蚁群算法

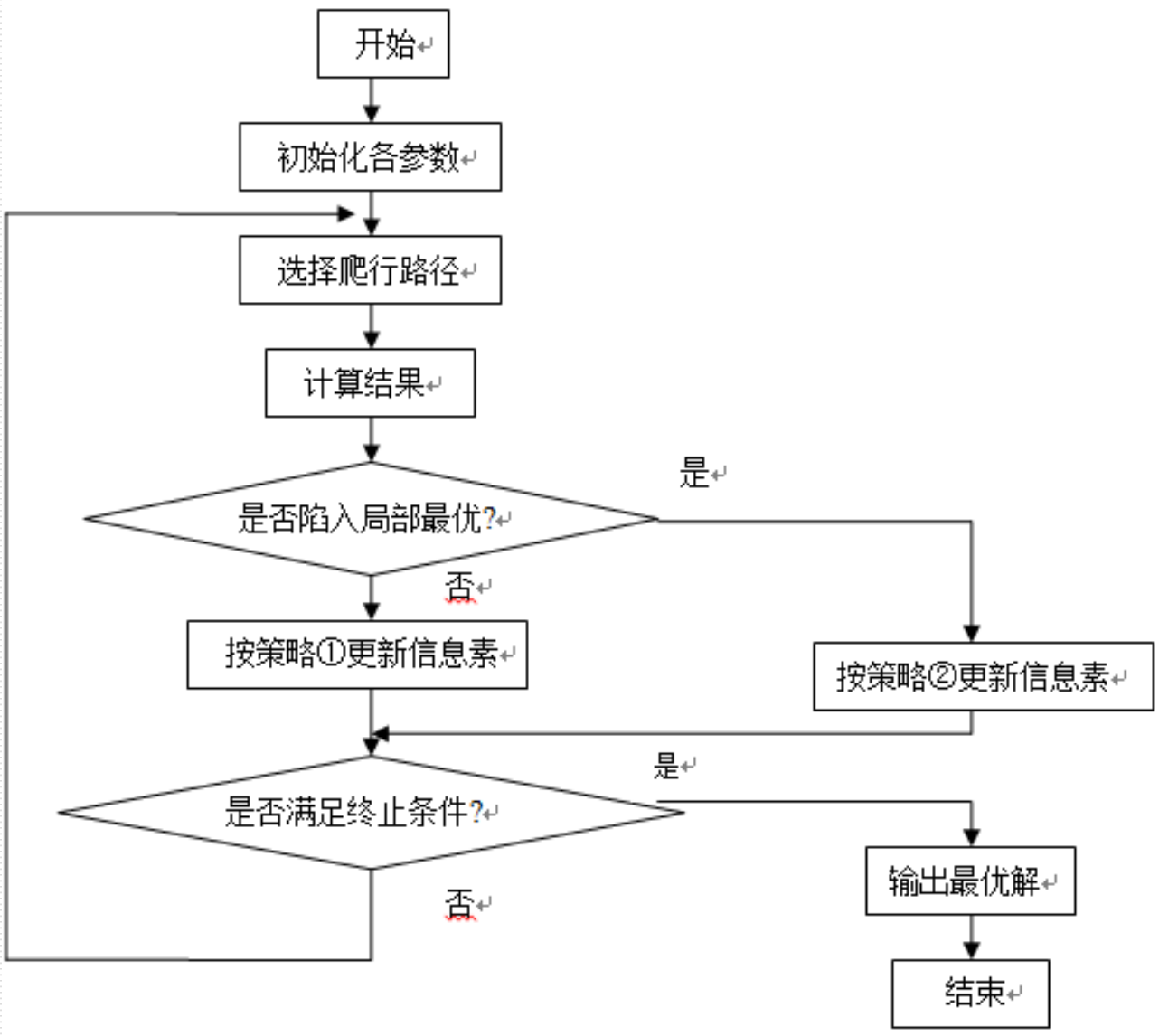


图7.5 自适应蚁群算法流程图

第7章 群智能算法及其应用

- 7.1 群智能算法产生的背景
- 7.2 粒子群优化算法
- 7.3 量子粒子群优化算法
- 7.4 粒子群优化算法的应用
- 7.5 基本蚁群算法
- 7.6 改进蚁群算法
- ✓ 7.7 蚁群算法的应用

7.7 蚁群算法的应用

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。

7.7 蚁群算法的应用

表7.3 柔性作业车间调度事例

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

7.7 蚁群算法的应用

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。

7.7 蚁群算法的应用

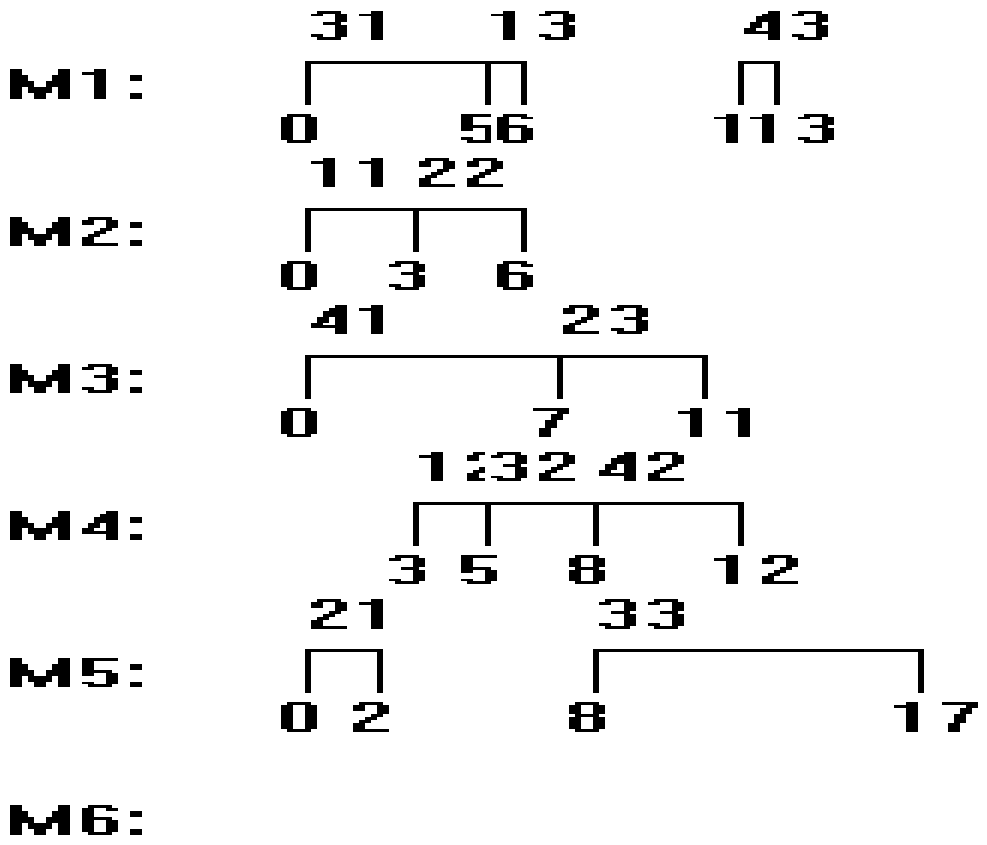
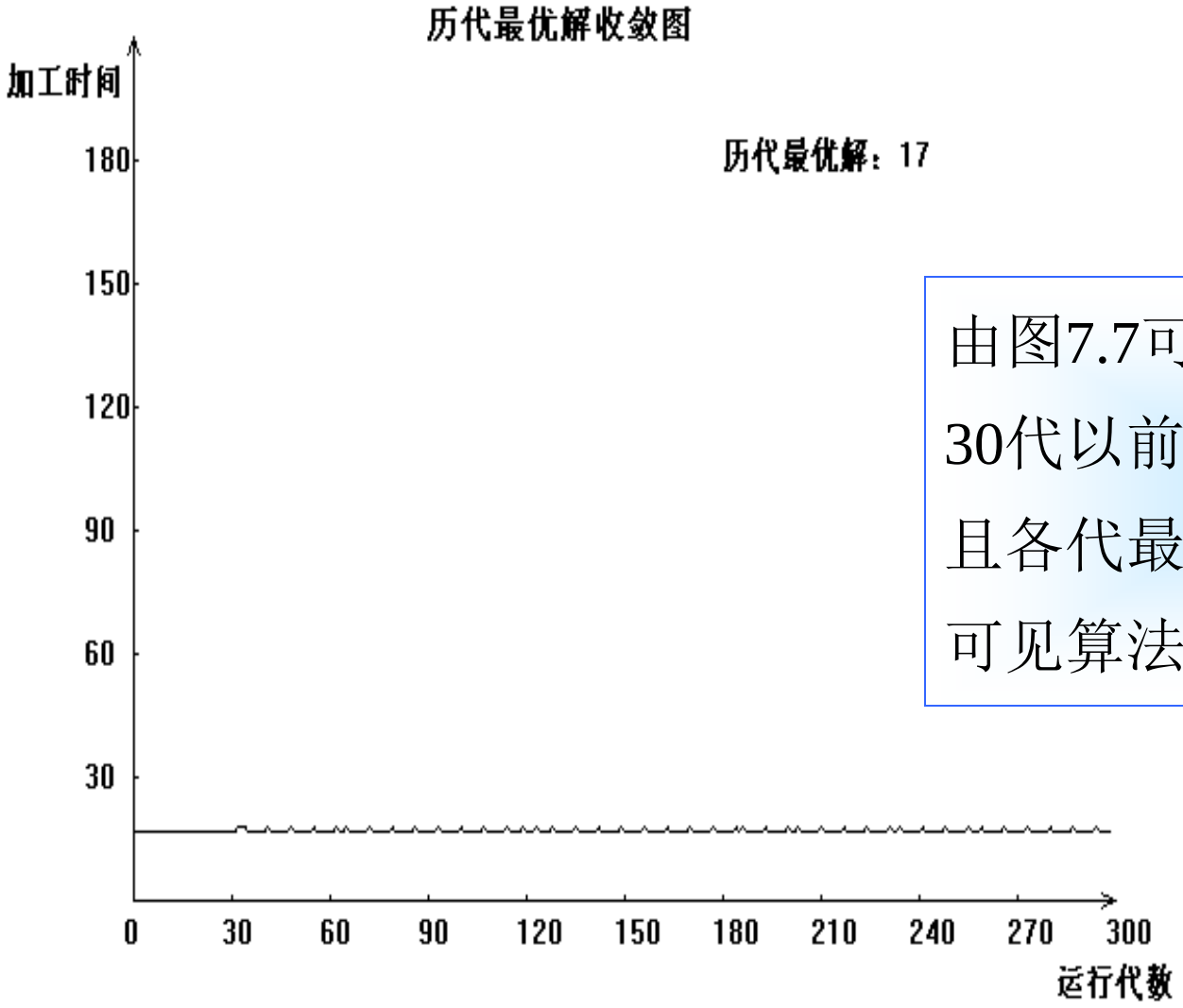


图7.6 最优解甘特图

7.7 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

7.8 小结

1. 粒子群优化算法

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P_i 。

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。

7.8 小结

2. 量子粒子群优化算法

① 确定种群规模和粒子维数，初始化粒子群体。

② 计算个体历史最优值(pbest)：根据适应度函数计算每一个微粒的适应度值，通过和个体的历史最优值比较，如果当前值优于个体历史最优值，则把当前值替换为个体最优值(pbest)，否则不替换。

③ 计算群体的历史最优值(gbest)：计算所有微粒的适应值，并与当前的全局最优值(gbest)比较，若当前值优于全局最优值，则把当前值替换为全局最优值(gbest) 。

④ 计算所有粒子的重心(mbest)；根据公式(7.5)来更新所有粒子的重心(mbest) 。

⑤ 根据量子粒子群进化方程(7.9)更新每个粒子的位置，产生新的种群。

⑥ 粒子适应度满足收敛条件或者是达到最大迭代次数，则算法结束，否则跳转到步骤2继续迭代执行。

3. 基本蚁群算法

- 蚂蚁在运动过程中，根据各条路径上的信息素决定转移方向。
- 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：

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

- 各路径上信息素浓度消散规则为：

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

- 蚁群的信息素浓度更新规则为：

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

THE END