目录 CONTENTS

中国石油大学(华东)
CHINA UNIVERSITY OF PETROLEUM
计算机科学与技术学院

第六章 群智能算法

人工智能课程组

智能科学系

目录 CONTENTS

群智能算法

- 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，已经广泛应用于组合优化、智能控制、模式识别、规划设计、网络安全等领域。
- 本章首先简要介绍群智能算法产生的背景，然后介绍遗传算法、粒子群优化算法、蚁群算法及其应用。

6.1 群智能算法产生的背景

- **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?

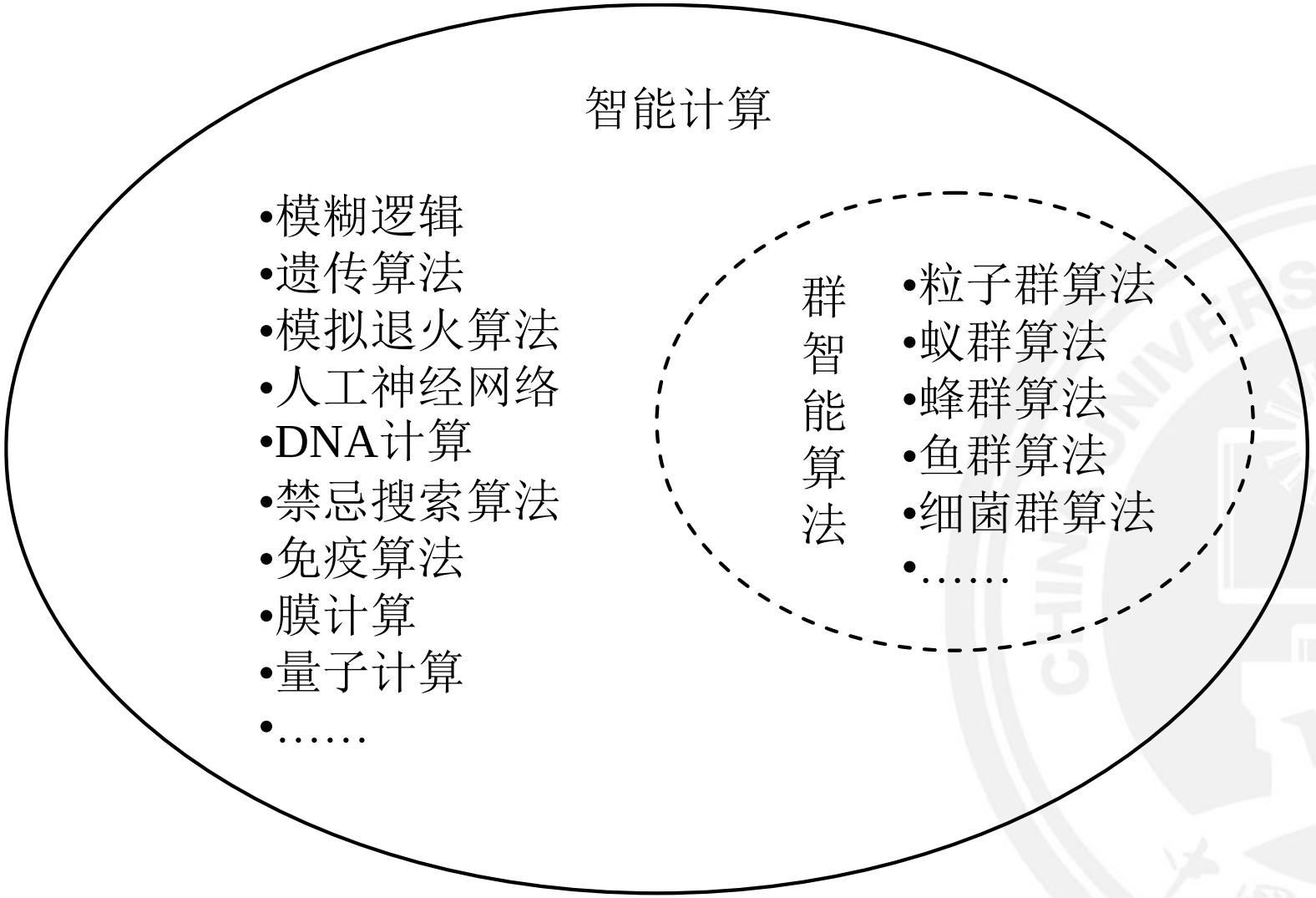
最简单有效的就是搜寻目前离食物最近的鸟的周围区域。

要素：群体行为的同步性（个体努力维持自身与邻居之间的距离为最优），因此，每个个体必须知道自身的位置和邻居的信息。

6.1 群智能算法产生的背景

- 群智能：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**（swarm algorithms，SI）：受动物群体智能启发的算法。
- 群智能算法包括：粒子群优化算法、蚁群算法、遗传算法、蜂群算法……

6.1 群智能算法产生的背景



6.2 遗传算法

- 1967年，美国密歇根大学J Holland教授的学生Bagley在他的博士论文中，首次提出了**遗传算法**这一术语。
- 1975年， J Holland提出了模式理论，出版了专著《自然系统和人工系统的适配》，在书中系统阐述了遗传算法的基本理论和方法。
- 20世纪80年代后，遗传算法进入兴盛发展时期，被广泛应用于自动控制、生产计划、图像处理、机器人等应用领域。

6.2 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。

- 遗传算法非常适用于处理传统方法难以解决的复杂和非线性优化问题。

  - 线性优化问题

$$\begin{aligned} &\text{maximize} && \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ &\text{subject to} && A\mathbf{x} \leq \mathbf{b} \\ &\text{and} && \mathbf{x} \geq \mathbf{0} \end{aligned}$$

- 遗传算法可广泛应用于组合优化、自适应控制、规划设计和网络安全等领域。

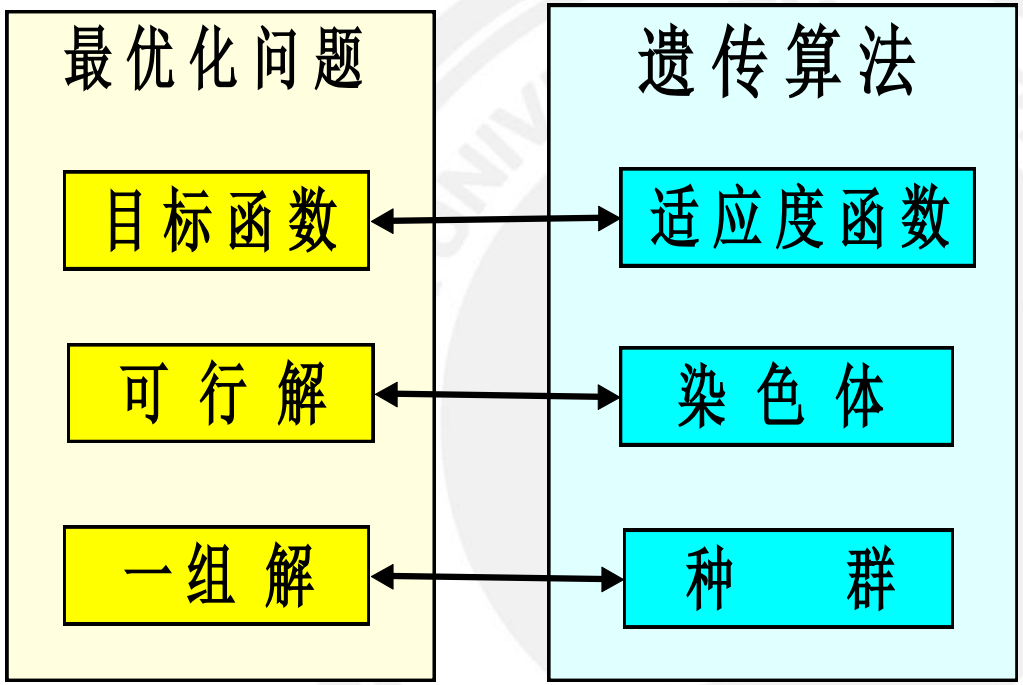
6.2.1 遗传算法的基本思想

- 遗传算法的基本思想：

对于自然界中生物遗传与进化机理的模仿。进化算法类似于生物进化，需要经过长时间的成长演化。

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。

$$\begin{aligned} &\text{maximize} && \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ &\text{subject to} && A\mathbf{x} \leq \mathbf{b} \\ &\text{and} && \mathbf{x} \geq \mathbf{0} \end{aligned}$$



6.2.1 遗传算法的基本思想

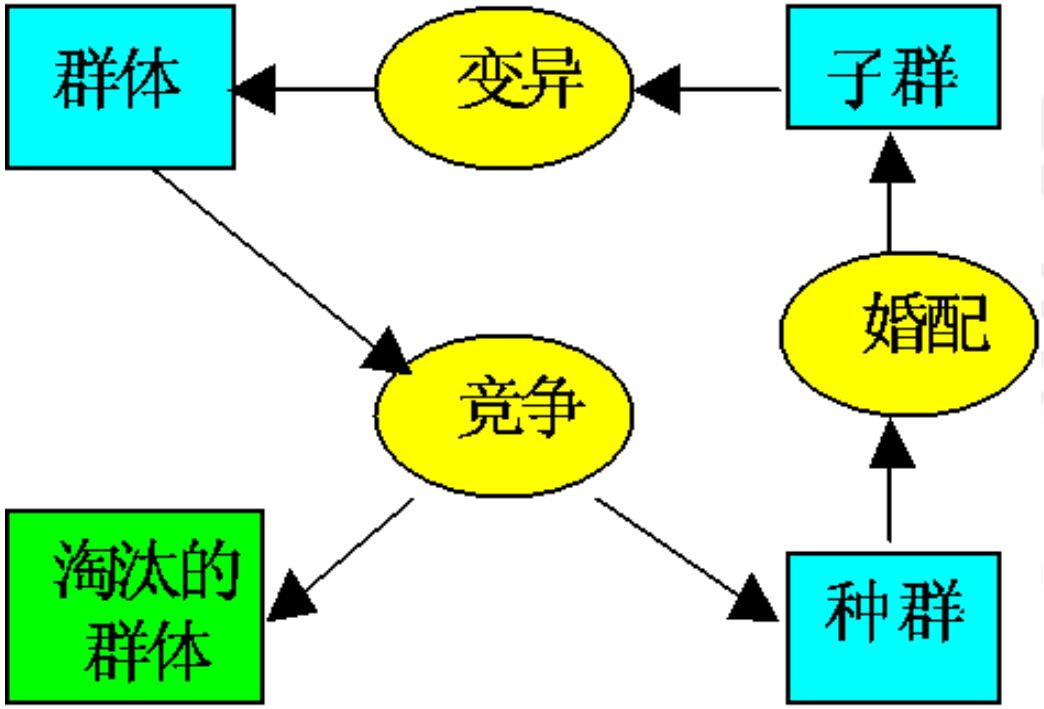
$$\begin{aligned} \text{maximize} \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ \text{and} \quad & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

- **生物遗传的概念：**

概念	遗产算法中的应用
适者生存	适应度函数目标值比较大的解被选择的可能性大
个体	解
染色体	解的编码（字符串、向量等）
基因	解的编码中每一分量
适应性	适应度函数值
群体	根据适应度值选定的一组解（解的个数为群体的规模）
婚配	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	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

- **生物进化的基本过程：**

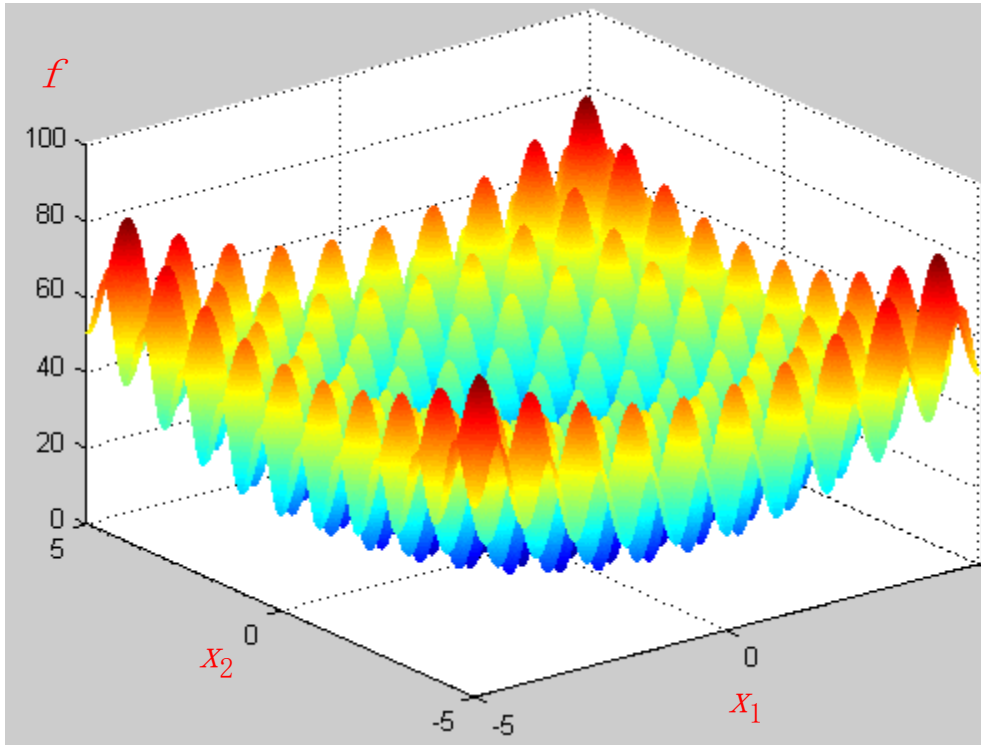


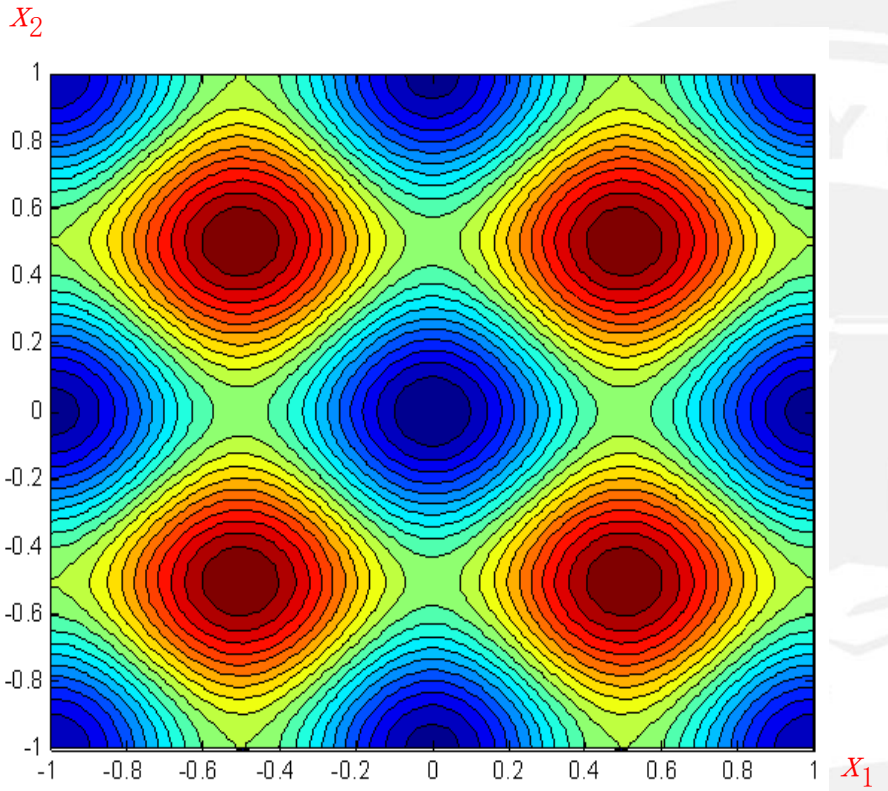
6.2.1 遗传算法的基本思想

例： 用遗传算法求解下面一个Rastrigin函数的最小值。

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

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





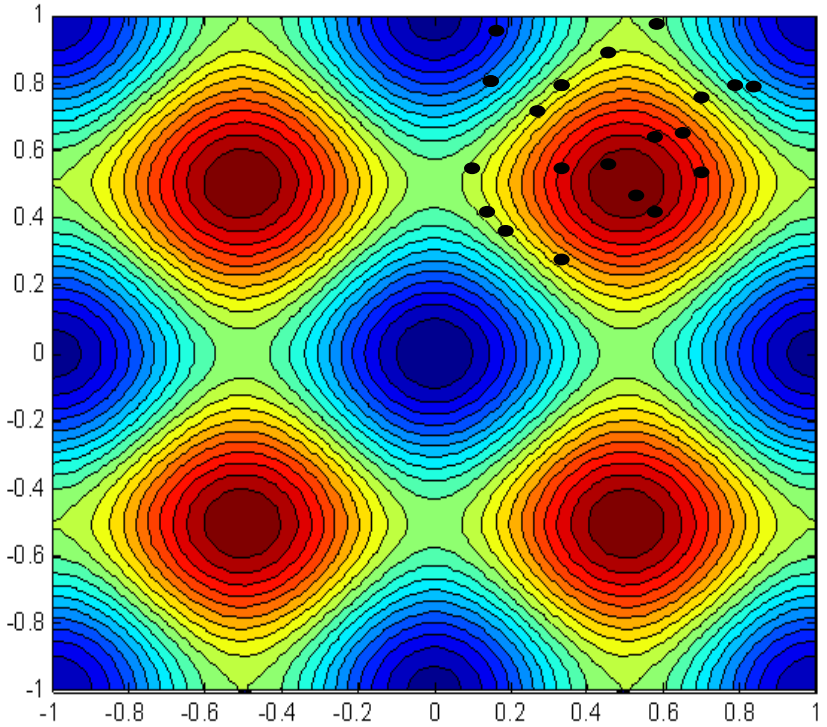
6.2.1 遗传算法的基本思想

例： 用遗传算法求解下面一个Rastrigin函数的最小值。

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

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

- 初始种群：

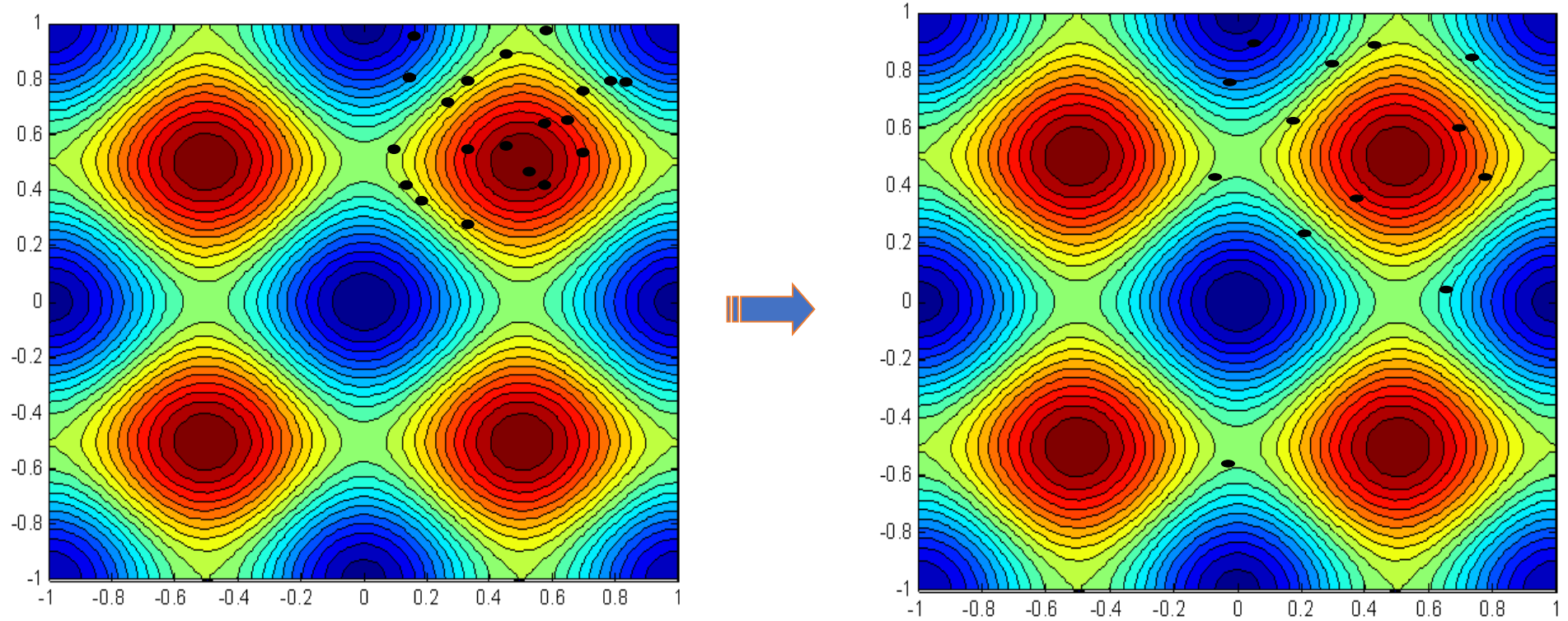


例： 用遗传算法求解下面一个Rastrigin函数的最小值。

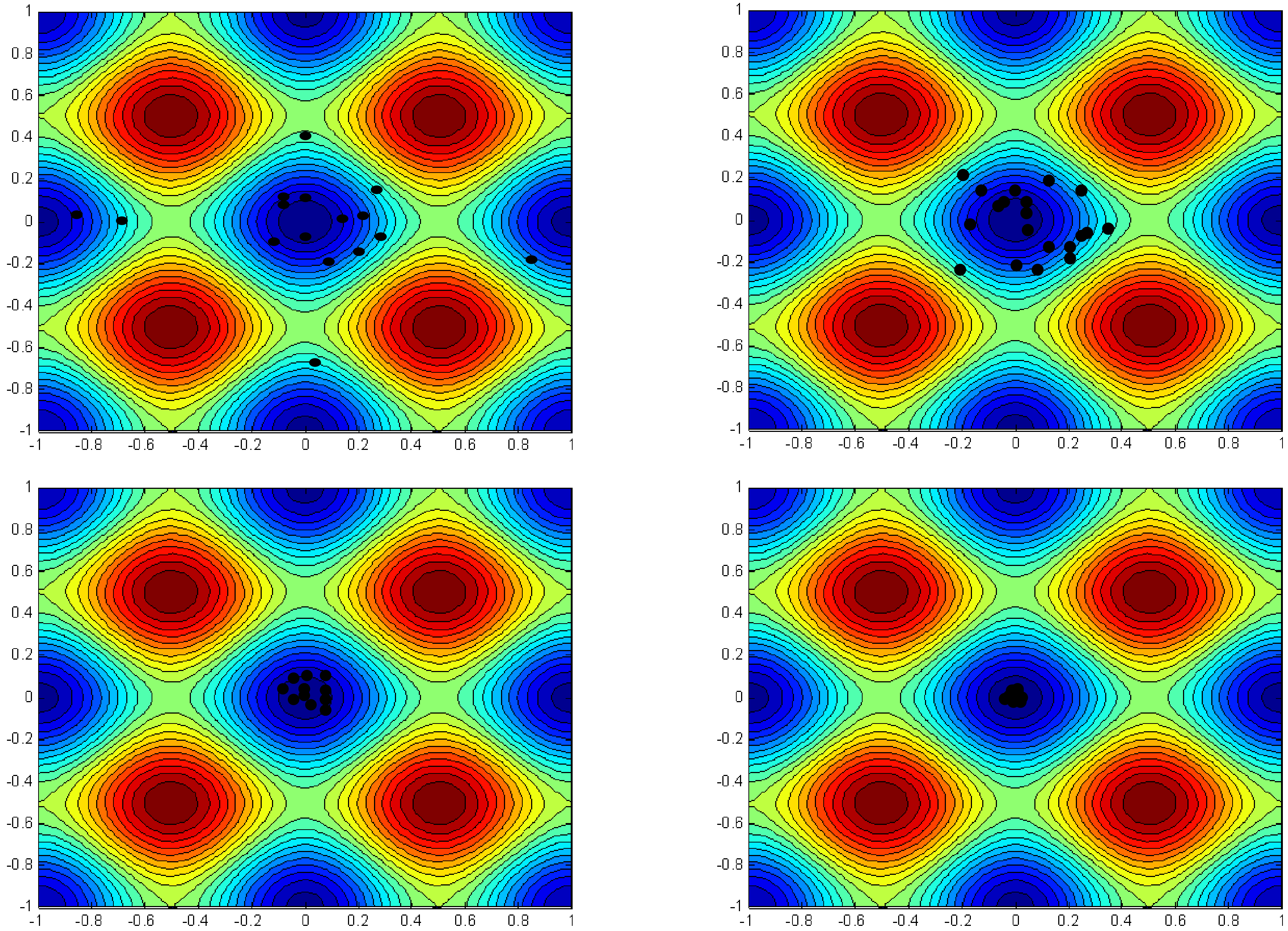
$$f(x_1, x_2) = 20 + x_1^2 + x_2^{\ 2} - 10(\cos 2\pi x_1 + \cos 2\pi x_2)$$

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

- 初始种群　　第二代种群



- 在迭代60、80、95、100次时的种群



遗传算法

长期以来，人们针对不同问题设计了许多不同的编码方法来表示问题的可行解，但是这些遗传算法都具有共同的特点，即通过对生物遗传和进化过程中选择、交叉、变异机理的模仿来完成对于问题最优解的自适应搜索过程。

- **遗传算法包含五个基本原则要素：具体参数编码，初始群体的设定，适应度函数的设计，遗传操作设计和控制参数设定。**

6.2.2 编码

$$f(x_1,x_2)=20+x_1^2+x_2^{\ 2}-10(\cos 2\pi x_1+\cos 2\pi x_2)$$
$$-5\le x_i\le 5 \qquad i=1,2$$

1.二进制编码

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。

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。

●**优点**：类似生物染色体的组成，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。

●**缺点**：求解高维优化问题的二进制编码串长，搜索效率低。

6.2.2 编码

2. 实数编码

$$f(x_1, x_2) = 20 + x_1^2 + x_2^{\ 2} - 10(\cos 2\pi x_1 + \cos 2\pi x_2)$$
$$-5 \le x_i \le 5 \qquad i = 1, 2$$

实数编码：用若干实数表示一个个体，然后在实数空间上进行遗传操作。

- 针对问题变量是实向量的情形，可以直接采用实数编码。
- 实数编码**不必进行数制转换**，可直接在解的表现型上进行遗传操作，从而可以引入与问题领域相关的启发式信息来增加算法的搜索能力。
- 遗传算法在求解高维或复杂优化问题时一般使用实数编码。

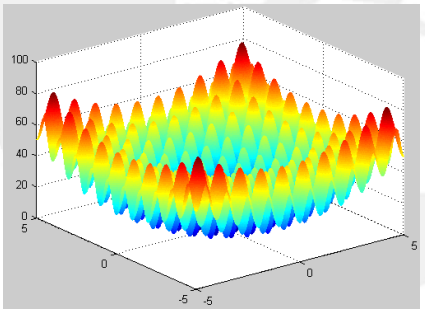
6.2.3 群体设定

1. 初始种群的产生

- 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。

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。

6.2.4 适应度函数

将目标函数映射成适应度函数的方法

- 若目标函数为**最大化**问题，则 $Fit(f(x)) = f(x)$
- 若目标函数为**最小化**问题，则 $Fit(f(x)) = \dfrac{1}{f(x)}$

•例子：求最小值

$$f(x_1, x_2) = 20 + x_1^2 + x_2^{\ 2} - 10(\cos 2\pi x_1 + \cos 2\pi x_2)$$
$$-5 \le x_i \le 5 \qquad i = 1, 2$$

6.2.5 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。

6.2.5 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：

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

6.2.5 选择

1. 个体选择概率分配方法

（2） 排序方法 （rank-based model）

① 线性排序：J. E. Baker

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

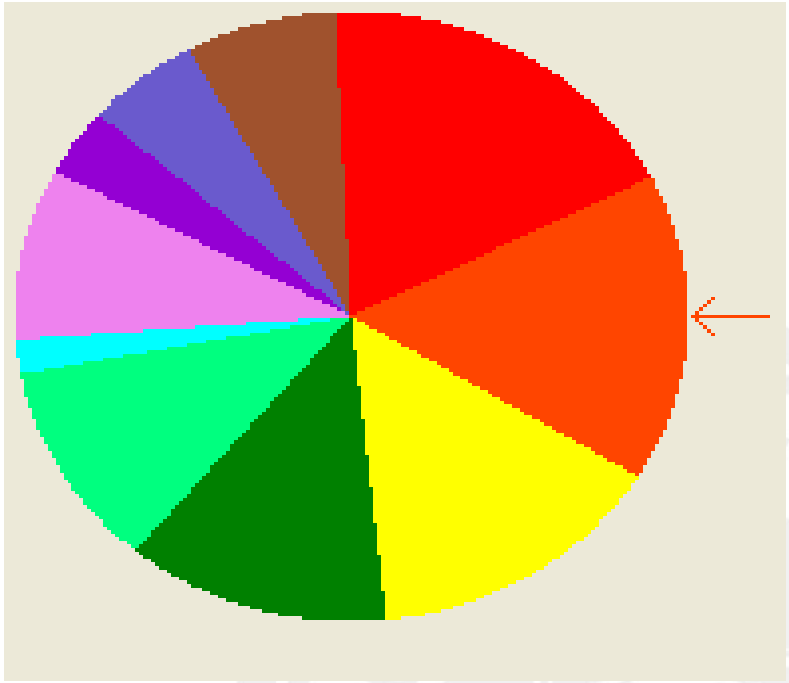
$$p_i = \frac{a - bi}{M(M+1)}$$

6.2.5 选择

2. 选择个体方法

2.1 转盘赌选择

- 按个体的选择概率产生一个轮盘，轮盘每个区的角度与个体的选择概率成比例。
- 产生一个随机数，它落入转盘的哪个区域就选择相应的个体交叉。



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**

6.2.5 选择

2. 选择个体方法

2.2 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。

6.2.6 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。

6.2.6 交叉

2. 修正

- 交叉有可能产生不满足约束条件的非法染色体，可以采取对交叉变异等遗传操作进行适当的修正。

3.交叉概率：确定两个染色体进行局部互换以产生两个新的子代的概率。**0.7**

- 较大的交叉概率可以增强遗传算法，开辟新的搜索区域的能力。

  但高性能模式遭到破坏的可能性也会随之增加。

- 太低的交叉概率会使搜索陷入迟钝状态。

6.2.7 变异

1. 主要变异方法

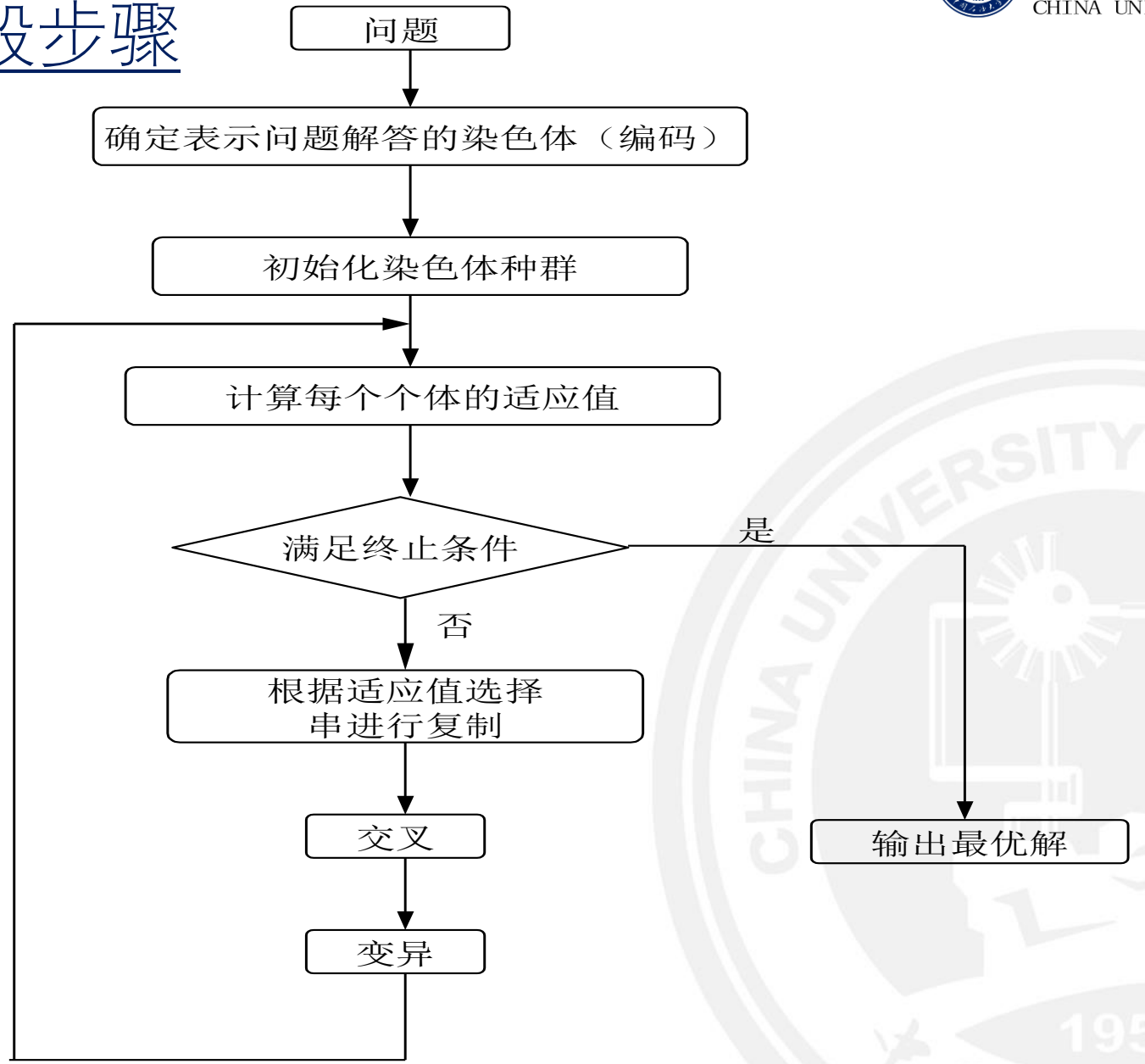
- **位点变异**：群体中的个体码串，随机挑选一个或多个基因座，并对这些基因座的基因值以变异概率作变动。

- **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。

2. 变异概率：变异概率一般不能大，以防止群体中重要单一的基因被丢失。**通常变异概率为0.001左右**

6.2.8 遗传算法的一般步骤



6.2.9 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，搜素过程中不需要问题的内在性质，可直接对结构对象进行操作。
- 利用随机技术指导对一个被编码的参数空间进行高效率搜索
- 采用群体搜索策略，易于并行化。
- 仅用适应度函数值来评估个体，并在此基础上进行遗传操作，使种群中个体之间进行信息交换。
- 遗传算法的适应度函数不受连续可微的约束，其定义域也可以任意设定。非常适用于传统优化算法难以解决的复杂优化问题，

6.2.10 遗传算法的应用

1. 流水车间调度问题

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

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

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

6.2.10 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。

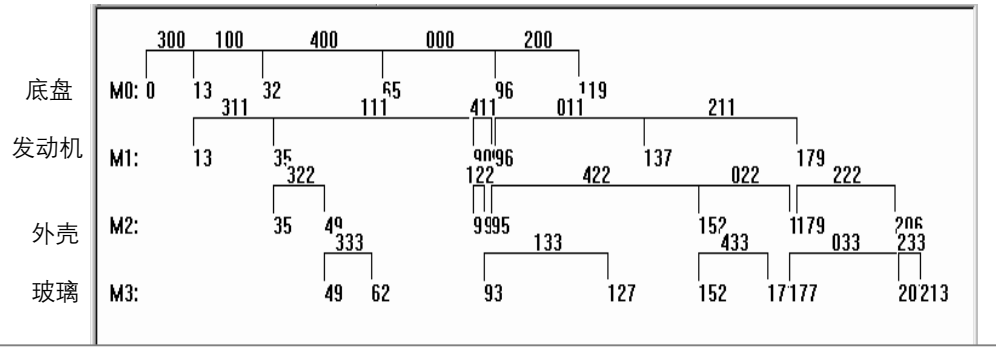
(5) 工序的准备时间与顺序无关，且包含在加工时间中。

(6) 工件在每台机器上的加工顺序相同，且是确定的。

6.2.10 遗传算法的应用

1. 流水车间调度问题

- **问题的数学模型：**



已知t_{jk}，

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

n个工件、m台机器的流水车间调度问题的完工时间：

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

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

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

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

最大流程时间：$c_{\max} = c(j_n, m)$

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

6.2.10 遗传算法的应用

1. 流水车间调度问题

- **问题的数学模型：**

加工时间表

工件 j	底盘 t_{j1}	发动机 t_{j2}	外壳 t_{j3}	玻璃 t_{j4}
1-劳斯莱斯	31	41	25	30
2-丰田	19	55	3	34
3-红旗	23	42	27	6
4-奥拓	13	22	14	13
5-玛莎拉蒂	33	5	57	19

已知t_{jk}，

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

n个工件、m台机器的流水车间调度问题的完工时间：

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

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

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

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

最大流程时间：$c_{\max} = c(j_n, m)$

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

6.2.10 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（Fixed Job Scheduling Problem固定的调度问题）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。

对于有五个工件的FSP，第 k 个染色体 $v_k = [1,2,3,4,5]$ ，表示工件的加工顺序为： j_1, j_2, j_3, j_4, j_5 。

6.2.10 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：

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

6.2.10 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

6.2.10 遗传算法的应用

表9.3 遗传算法运行的结果

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 $N = 50$。

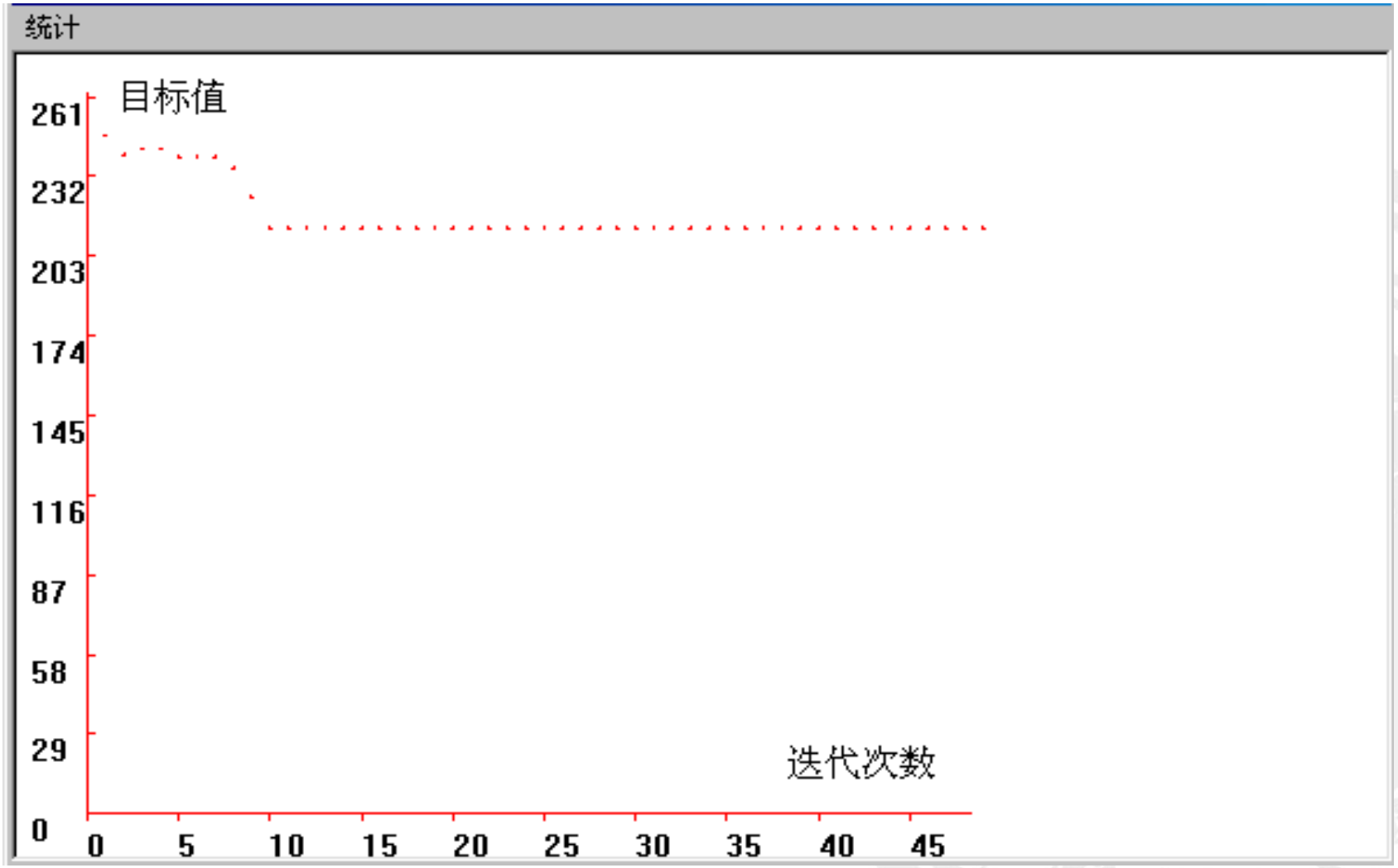
遗传算法运行的结果

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

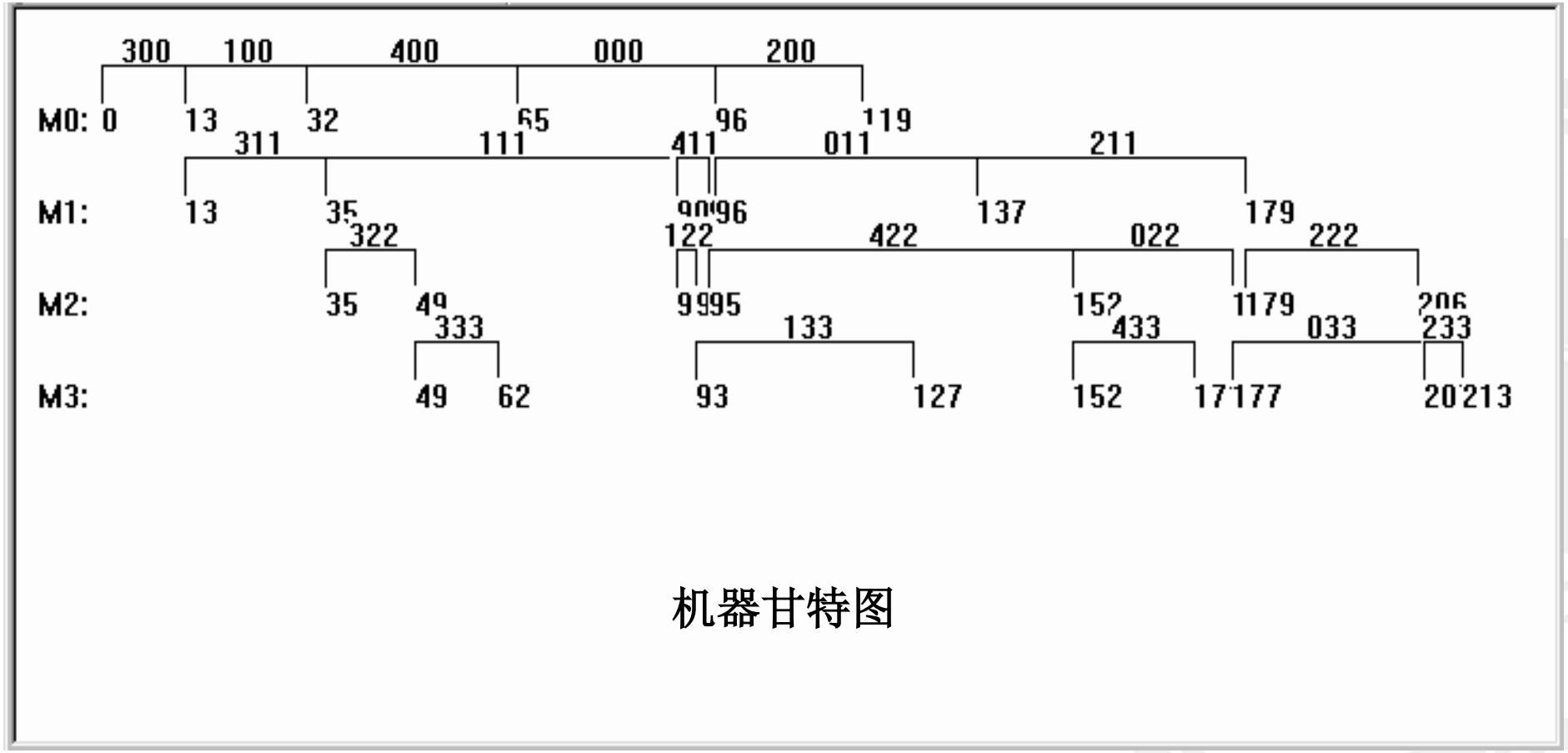
6.2.10 遗传算法的应用

表9.3 遗传算法运行的结果

最优解收敛图



6.2.10 遗传算法的应用



机器甘特图

6.3 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。

6.3.1 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。

6.3.1 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：

- $x^i(k)=\begin{bmatrix} x_1^i & x_2^i & \cdots & x_n^i \end{bmatrix}^T$ ：表示搜索空间中粒子i的当前位置。

- $p^i(k)=\begin{bmatrix} p_1^i & p_2^i & \cdots & p_n^i \end{bmatrix}^T$ ：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。

- $v^i(k)=\begin{bmatrix} v_1^i & v_2^i & \cdots & v_n^i \end{bmatrix}^T$：表示该粒子的搜索方向。

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

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

——（6.8a）

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

——（6.8b）

$$i=1,2,\cdots,m;\qquad j=1,2,\cdots,n$$

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。

6.3.1 粒子群优化算法的基本原理

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

——（6.8a）

- 式(6.8a)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。

6.3.1 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。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进行比较，如果优于P_i，则将其作为该粒子的最好位置 。

6.3.1 粒子群优化算法的基本原理

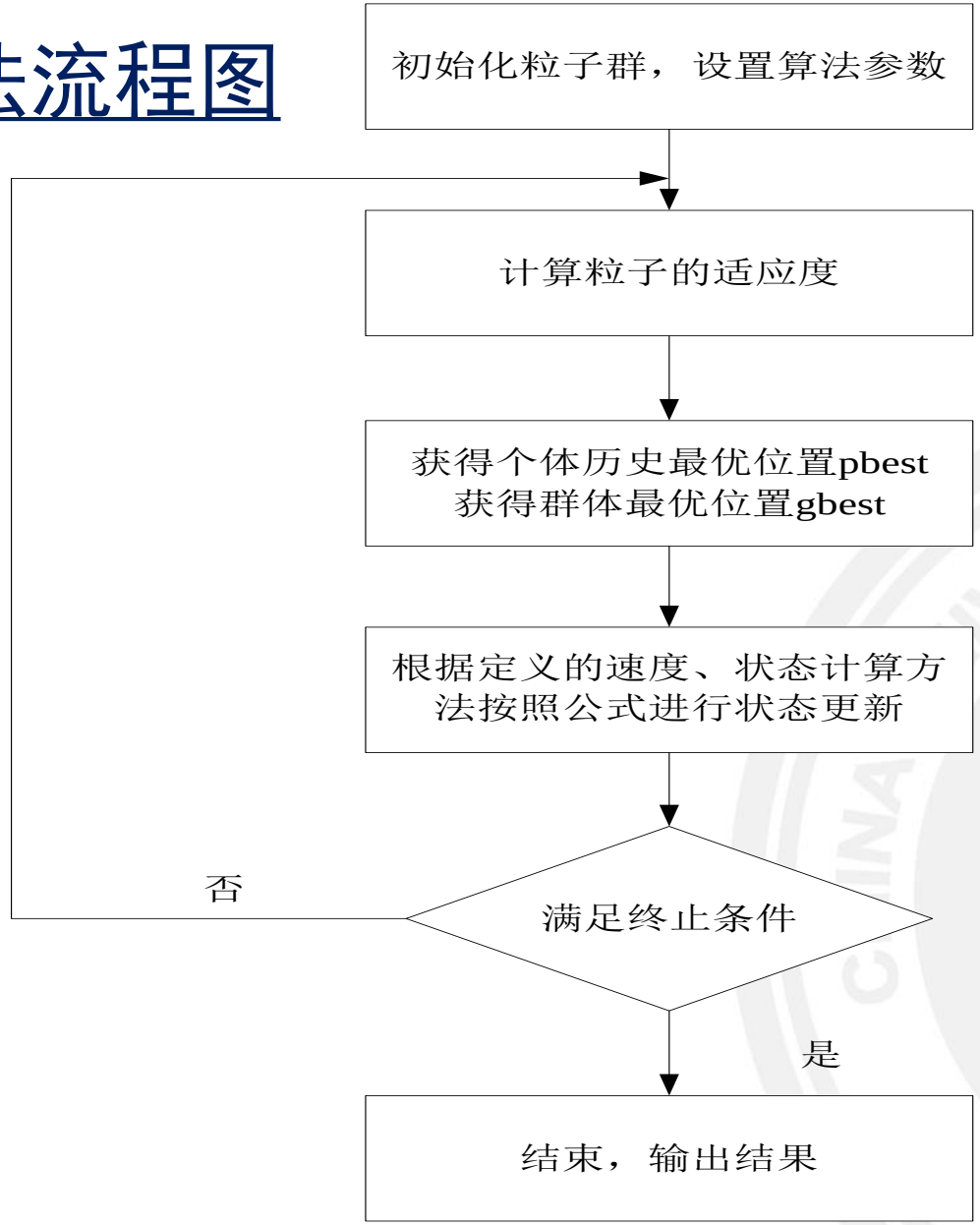
粒子群优化算法的流程：

（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）更新粒子的速度和位置。根据式（6.8）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。

6.3.1 粒子群优化算法流程图



6.4 蚁群算法

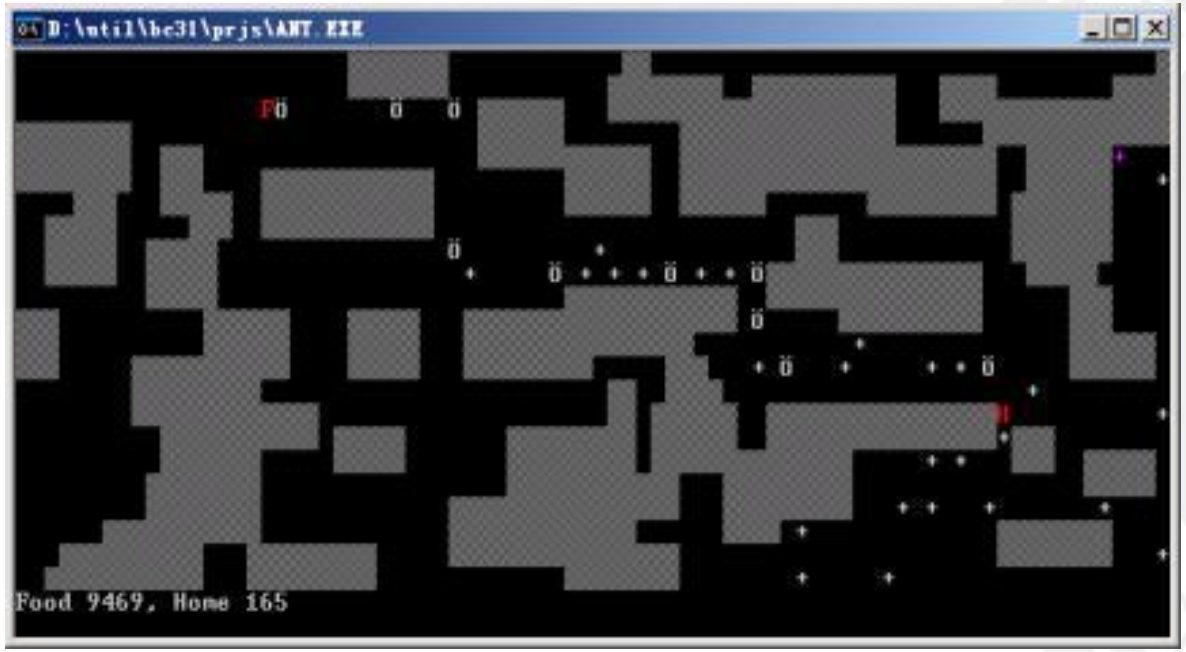
产生背景

- 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。

6.4.1 蚁群觅食习性

- **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。

- **信息素跟踪**：按照一定的**概率**沿着信息素较强的路径觅食。



6.4.1 蚁群觅食习性

（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。

6.4.2 基本蚁群算法

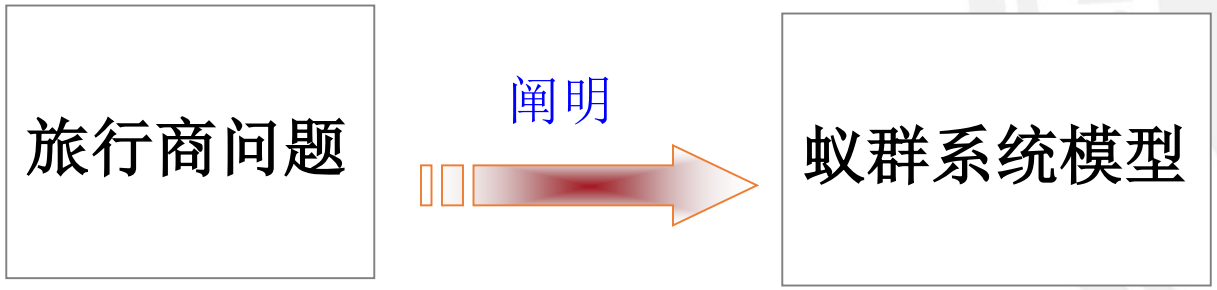
蚁群优化算法的第一个应用是著名的旅行商问题。

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。



6.4.2 基本蚁群算法

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。

信息素 $\tau_{xy}(t)$

局部启发信息 $\eta_{xy}(t)$

共同决定

$P_{xy}^{k}(t)$

6.4.2 基本蚁群算法模型

$P_{xy}^{k}(t)$ 表示如下：

$$P_{xy}^{k}(t)=\begin{cases}\dfrac{\left|\tau_{xy}(t)\right|^{\alpha}\left|\eta_{xy}(t)\right|^{\beta}}{\sum_{y\in allowed_k(x)}\left|\tau_{xy}(t)\right|^{\alpha}\left|\eta_{xy}(t)\right|^{\beta}} & if \quad y\in allowed_k(x)\\ 0 & 其他\end{cases} \tag{6.9}$$

其中：

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

$tabu_k(x)(k=1,2,...,m)$ 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

6.4.2 基本蚁群算法

随着时间的推移，以前留下的信息速度逐渐消失。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：

$$\tau_{xy}(t+1)=\rho\tau_{xy}(t)+\Delta\tau_{xy}(t) \tag{6.10}$$

蚁群的信息素浓度更新规则为：

$$\Delta\tau_{xy}(t)=\sum_{k=1}^{m}\Delta\tau_{xy}^{k}(t) \tag{6.11}$$

6.4.2 基本蚁群算法

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：

$$\Delta\tau_{xy}^{k}(t)=\begin{cases}\dfrac{Q}{L_k} & \text{若第}k\text{只蚂蚁在本次循环中从}x\text{到}y\\ 0 & \text{否则}\end{cases} \tag{6.12}$$

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

6.4.2 基本蚁群算法

全局信息更新方法的优点：

- 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；
- 提高了系统搜索收敛的速度。

本章小结

- 群智能算法产生的背景
- 遗传算法
- 粒子群优化算法
- 蚁群算法

THANKS