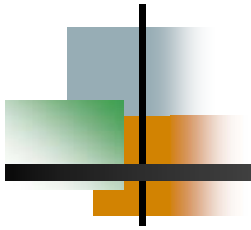
遗传算法



Contents

Contents of Section 1

1 遗传算法概述

1 遗传算法概述

1.1 遗传算法（Genetic Algorithm, GA）

- 一种仿生全局优化算法
- 模仿生物的遗传进化原理（Darwin's theory of evolution & Mendel's law of inheritance)，通过选择（Selection）、交叉（Crossover）与变异（Mutation）等操作机制，使种群中个体的适应性（Fitness）不断提高
- 核心思想：物竞天择，适者生存
（“天”——适应度函数，Fitness Function)

1 遗传算法概述

1.2 遗传算法的特点

四大优点：

- ✓ 良好的并行性（操作对象是一组可行解；搜索轨道有多条）
- ✓ 强大的通用性（只需利用目标的取值信息，无需梯度等高价值信息）
- ✓ 良好的全局优化性和鲁棒性
- ✓ 良好的可操作性

两个缺点：

- × 未成熟收敛问题
- × 收敛速度较慢，算法实时性欠佳

1 遗传算法概述

1.3 遗传算法的发展历史

表1.1 遗传算法理论的经典研究成果

年份	贡献者	内容
1962	Holland	程序漫游元胞计算机自适应系统框架
1968	Holland	模[illegible]理的建[illegible]
1971	Hollstien	[illegible]函数优化
1972	[illegible] Foo, Zei	[illegible]传算法的基因操作
1972	Frantz	[illegible]
1973	Holland	遗[illegible]法中试[illegible]的最优配置和双臂强盗问题
1973	Martin	类似遗传算法的概率算法理论
1975	De Jong	用于5个测试函数的研究基本遗传短发基准参数
1975	Holland	出版开创性著作《Adaptation in Natural and Artificial Systems》

第一阶段：
20世纪60年代至70年代中期
（萌芽期）

1 遗传算法概述

续表1.1

年份	贡献者	内容
1981	Bethke	应用Walsh函数分析模式
1981	Brindle	研究遗传算法中的选择和支配问题
1983	Pettit, Swigger	遗传算法应用于非稳定问题的粗略研究
1983	Wetzel	用遗传算法解决旅行商问题（TSP)
1984	Mauldin	基本遗传算法中用启发知识维持遗传多样性
1985	Baker	试验基于排序的选择方法
1985	Booker	建议采用部分分配计分、分享操作和交配限制法
1985	Goldberg, Lingle	TSP问题中采用部分匹配交叉
1985	Grefenstette, Fitzpattrick	对含噪声的函数进行测试
1985	Schaffer	多种群遗传算法解决多目标优化问题

1 遗传算法概述

续表1.1

年份	贡献者	内容
1986	Goldberg	最优种群大小估计
1986	Grefenstette	元级遗传算法控制的遗传算法
1987	Baker	选择中随机误差的较小[illegible]法
1987		[illegible]题（MDP）
1987		[illegible]物种归纳法
1987	Go[illegible]	[illegible]夫链
1987	Goldberg, Smith	双倍染色体遗传算法应用于非稳定函数优化
1987	Oliver, Smith, Holland	排列重组算子的模拟和分析
1987	Schaffer, Morishima	串编码自适应交叉试验
1987	Whitley	子孙测试应用于遗传算法的选择操作

第二阶段：
20世纪80年代（蓬勃发展期）

1 遗传算法概述

1.4 遗传算法的应用领域

（1）函数优化（经典应用）

（2）组合优化（旅行商问题——已成为衡量算法优劣的标准、背包问题、装箱问题等）

（3）生产调度问题

（4）自动控制（如航空控制系统的优化设计、模糊控制器优化设计和在线修改隶属度函数、人工神经网络结构优化设计和调整人工神经网络的连接权等优化问题）

（5）机器人智能控制（如移动机器人路径规划、关节机器人运动轨迹规划、机器人逆运动学求解等）

Hotspot

（6）图像处理和模式识别（如图像恢复、图像边缘特征提取、几何形状识别等）

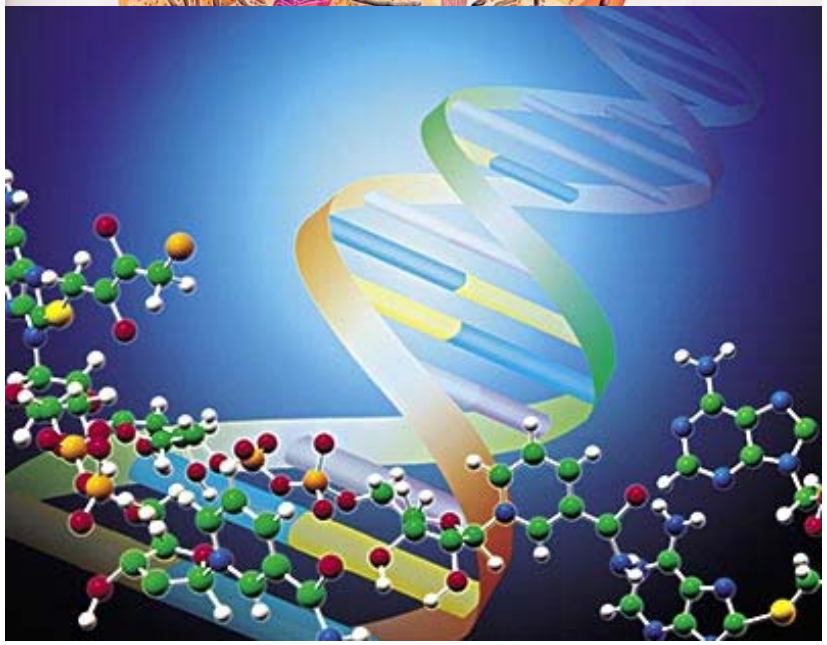
（7）机器学习（将GA用于知识获取，构建基于GA的机器学习系统）

此外，遗传算法在人工生命、遗传程序设计、社会和经济领域等方面的应用尽管不是很成熟，但还是取得了一定的成功。在日后，必定有更深入的发展。

Contents of Section 2

2 标准遗传算法

2.1 遗传算法的生物学基础



细胞（Cell）

染色体（Chromosome）

脱氧核糖核酸（Deoxyribonucleic Acid，DNA）

基因（Gene）、等位基因（Allele）

- 复制（Reproduction）
- 交叉（Crossover）
- 变异（Mutation）

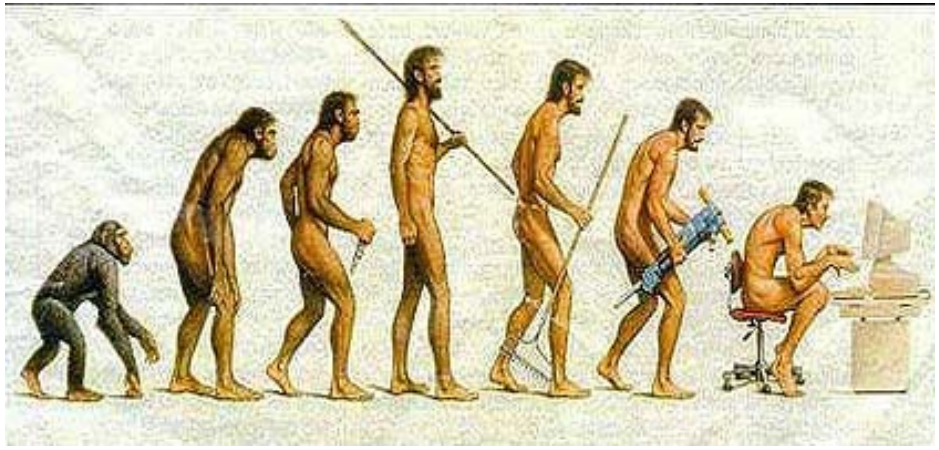
基因型（Genotype）

表现型（Phenotype）→ 对环境的适应性

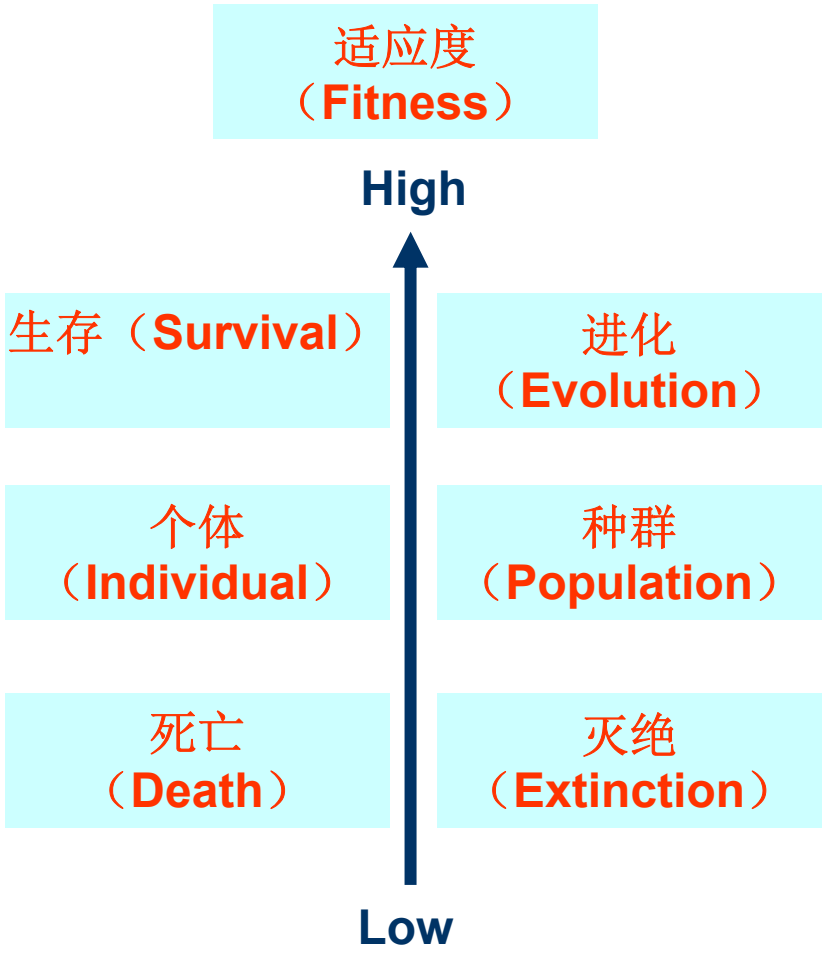
“种瓜得瓜，种豆得豆”

2 标准遗传算法

2.1 遗传算法的生物学基础

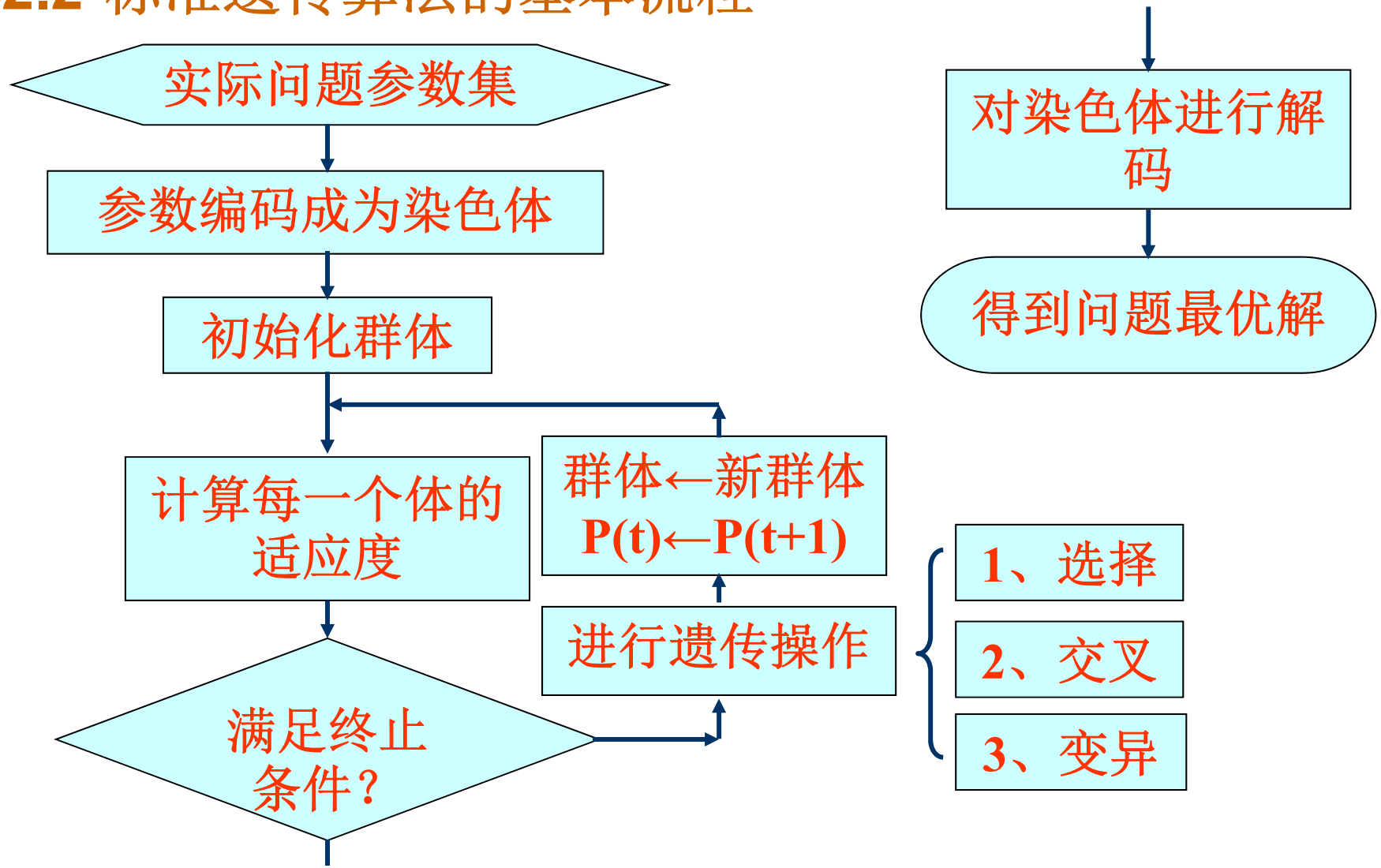


“物竞天择，适者生存”



2 标准遗传算法

2.2 标准遗传算法的基本流程



2 标准遗传算法

2.3 遗传算法的若干概念

- 个体(Individual)　称$S=\{0,1\}^l$为个体空间，个体空间的元素$a=a_0a_1\cdots a_{l-1}\in S$称为个体，它是染色体带有特征的实体。分量$a_j\in\{0,1\}$称为基因，正整数$l$称为个体的基因长度。

- 种群(Population)　称个体空间S中N个个体组成的一个子集（个体允许重复）称为一个种群，记为：

$$A=(A_1,A_2,\cdots,A_N)$$

其中$A_j(j=1,2,\cdots,N)\in S$，N称为种群规模。

2 标准遗传算法

2.3 遗传算法的若干概念

- 适应度（Fitness）　在研究自然界中生物的遗传和进化现象时，生物学家使用适应度这个术语来度量某个物种对于生存环境的适应程度。对生存环境适应程度较高的物种将获得更多的繁殖机会，而对生存环境适应程度较低的物种，其繁殖机会就会相对较少，甚至逐渐灭绝。在遗传算法中，一般通过适应度函数（Fitness function）来衡量某一个体的适应度高低。

2 标准遗传算法

2.3 遗传算法的若干概念

- 编码（Coding） 将一个待求解的问题的实际可行解从其解空间转换到遗传算法所能处理的搜索空间（即个体空间）的过程，就称为编码。

- 解码（Decoding） 解码是将遗传算法所搜索到的最优个体的染色体转换成待求解问题的实际最优解的过程，即编码的逆过程。

2 标准遗传算法

2.3 遗传算法的若干概念

- 选择操作（Selection） 根据各个个体的适应度，按照一定的规则，从第t代群体P(t)中选择出一些优良的个体遗传到下一代群体P(t+1)中。一般地，选择操作通过选择算子（Selection Operator）进行。
- 交叉操作（Crossover） 将群体P(t)内的各个个体随机搭配成对，对每一对个体，以某个概率（称为交叉概率，Crossover Rate）遵循某一种规则交换它们之间的部分染色体。
- 变异操作（Mutation） 对群体P(t)中的每一个个体，以某一概率（称为变异概率，Mutation Rate）改变某一个或某一些基因座上的基因值为其他的等位基因。

2 标准遗传算法

2.4 遗传算法的应用步骤

（1）确定决策变量及各种约束条件，即确定出个体的表现型X和问题的解空间。

（2）建立优化模型，确定出目标函数的类型及其数学描述形式或量化方法。

（3）确定表示可行解的染色体编码方法，即确定出个体的基因型X*，及遗传算法的搜索空间。

编码是遗传算法解决问题的先决条件和关键步骤：

①不仅决定个体基因的排列形式（从而决定选择与繁殖等操作的作用方式），而且也决定从搜索空间的基因型到解空间的表现型的解码方式（从而决定对GA所获解的翻译与理解）；

②决定GA搜索的困难度与复杂性；

③决定对问题的求解精度。

常用的遗传算法编码方法主要有：二进制编码、浮点数编码等。可以证明，二进制编码比浮点数编码搜索能力强，但浮点数编码比二进制编码在变异操作上能够保持更好的种群多样性。

2 标准遗传算法

2.4 遗传算法的应用步骤

标准遗传算法多采用二进制编码方法，将决策变量用二进制字符串表示，二进制编码串的长度由所求精度决定。然后将各决策变量的二进制编码串连接在一起，构成一个染色体。

例如：变量x的定义域为[-2，3]，要求其精度为10^{-5}，则需要将[-2，3]分成至少500 000个等长小区域，而每个小区域用一个二进制串表示。于是有，$2^L=500\ 000$，即

$$\log_2 500000 \approx 18.93 \qquad \delta = \frac{X_r - X_l}{2^L - 1}$$

向上取整，可得到$L=19$。即可用19位二进制串$a_{18}a_{17}\ldots a_0$来表示。

（4）确定解码方法，即确定出由个体基因型X*，到个体表现型X的对应关系和转换方法。

例如：对于二进制编码，其解码过程如下：若X*的取值范围为$[X_l,X_r]$，参数的二进制编码码长为L，码串对应的十进制整数为k，则解码公式为:

$$k = \sum_{i=0}^{L-1} a_i \cdot 2^i \qquad X = \frac{(X_r - X_l)\cdot k}{(2^L - 1)} + X_l$$

式中，$[X_l,X_r]$——参数最小、最大值；

L——参数编码长度；

k——二进制串对应的实数值。

2 标准遗传算法

2.4 遗传算法的应用步骤

（5）确定个体适应度的量化评价方法，就是确定出由目标函数值到个体适应度的转换规则。标准遗传算法的适应度函数常用一下三种：

I 直接以待求解的目标函数为适应度函数

若目标函数为最大值问题，则$Fit(f(X))=f(X)$

若目标函数为最小值问题，则$Fit(f(X))=-f(X)$

优点：简单直观；

缺点：其一，可能不满足非负的要求；其二，某些代求解的函数值分布相差很大，由此得到的平均适应度可能不利于体现种群的平均性能。

II 界限构造法

若目标函数为最大值问题，则

$$Fit(f(X))=\begin{cases} c_{\max}+f(x), & f(x)<c_{\max} \\ 0, & \text{其它} \end{cases}$$

C_{max}为$f(x)$的最大值估计。

2 标准遗传算法

2.4 遗传算法的应用步骤

若目标函数为最小值问题，则

$$Fit(f(X)) = \begin{cases} f(x) - c_{\min}, & f(x) > c_{\min} \\ 0, & \text{其它} \end{cases}$$

C_{min}为$f(x)$的最小值估计。

该方法是第一种方法的改进，但有时存在界限值预先估计困难或估计不精确等问题。

Ⅲ 倒数法

若目标函数为最小值问题，则

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

若目标函数为最大值问题，则

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

C为目标函数界限的保守估计值。

2 标准遗传算法

2.4 遗传算法的应用步骤

（6）确定各遗传具体操作方法。

①选择算子和选择操作

个体选择概率的常用分配方法有以下两种：

A 按比例的适应度分配（**Proportional Fitness Assignment**） 亦可称为选择的蒙特卡罗方法，是利用比例于各个个体适应度的概率决定其子孙的遗留可能性。若某个个体i，其适应度为f_i，则其被选取的概率表示为，

$$P_i = f_i \Big/ \sum_{i=1}^{M} f_i$$

显然选择概率大的个体，能多次被选中，它的遗传因子就会在种群中扩大。

B 基于排序的适应度分配（**Rank-based Fitness Assignment**） 在基于排序的适应度分配中，种群按目标值进行排序。适应度仅仅取决于个体在种群中的序位，而不是实际的目标值。

排序方法比比例方法表现出更好的鲁棒性，它能在一定程度上克服了比例适应度计算的尺度问题和过早收敛问题。

2 标准遗传算法

2.5 遗传算法的应用步骤

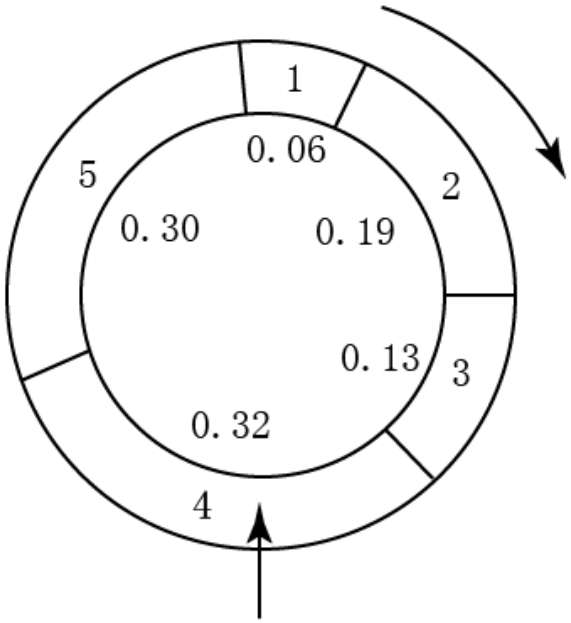
个体选择概率确定后，可以选用的常用选择算法有轮盘赌选择法（**Roulette Wheel Selection**）、随机遍历抽样法（**Stochastic Universal Sampling**）、局部选择法（**Local Selection**）、截断选择法（**Truncation Selection**）和锦标赛选择法（**Tournament Selection**）等。标准遗传算法常用的轮盘赌选择法的原理如右下图所示。

$$S=\sum_{i=1}^{n} f(X_i) \longrightarrow p_i=\frac{f(X_i)}{S} \longrightarrow q_k=\sum_{j=1}^{k} p_j \longrightarrow \text{pointer}$$

另外，还可采用以下的几种提高遗传算法性能的选择方法

Ⅰ稳态繁殖（Steady State Reproduction） 在迭代过程中用部分优质新子个体来更新群体中部分父个体，作为下一代种群。

Ⅱ没有重串的稳态繁殖（Steady State Reproduction without Duplicates） 在稳态繁殖的基础上，形成下一代新种群时，使其中的个体不重复。



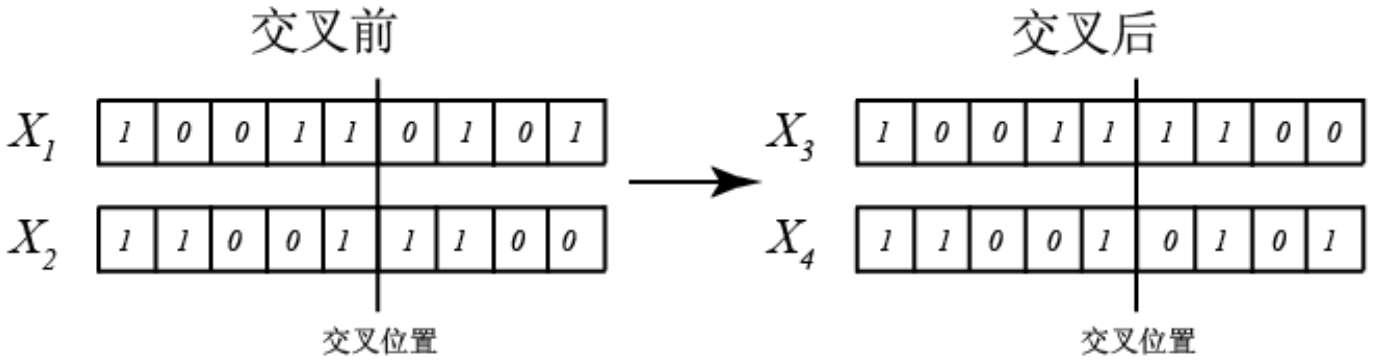
2 标准遗传算法

② 交叉率及交叉操作

交叉，也可以称为基因重组（**Recombination**），是遗传算法获取新的优良个体的最重要的手段，决定了遗传算法的全局搜索能力。

一般地，当随机产生的概率大于交叉率，遗传算法就会按一定规则选择两个个体，执行交叉操作。交叉率的选择决定了交叉的频率，较大的交叉率使各代充分交叉，但群体中的优良模式遭到破坏的可能性增大，以致产生较大的代沟，从而使搜索走向随机化；交叉率越低，产生的代沟越小，就会使得更多的个体直接复制到下一代，遗传搜索可能陷入停滞状态，一般建议取值范围**0.4~0.9**。

对于二进制编码，常用的交叉方法有：单点交叉、多点交叉和均匀交叉等。一个单点交叉的例子如下图所示。



2 标准遗传算法

此外，还有部分匹配交叉（**Partially Matched Crossover**）、顺序交叉（**Ordered Crossover**）、洗牌交叉（**Shuffle Crossover**）等等。

③变异率及变异操作

变异本身是一种局部随机搜索，使遗传算法具有局部的随机搜索能力；同时使得遗传算法保持种群的多样性，以防止出现非成熟收敛。

一般地，随机产生的概率大于变异率就会触发变异操作。变异率一般可取**0.001~0.1**。变异率不能取得太大，如果大于**0.5**，遗传算法就退化为随机搜索，而遗传算法的一些重要的数学特性和搜索能力也不复存在了。

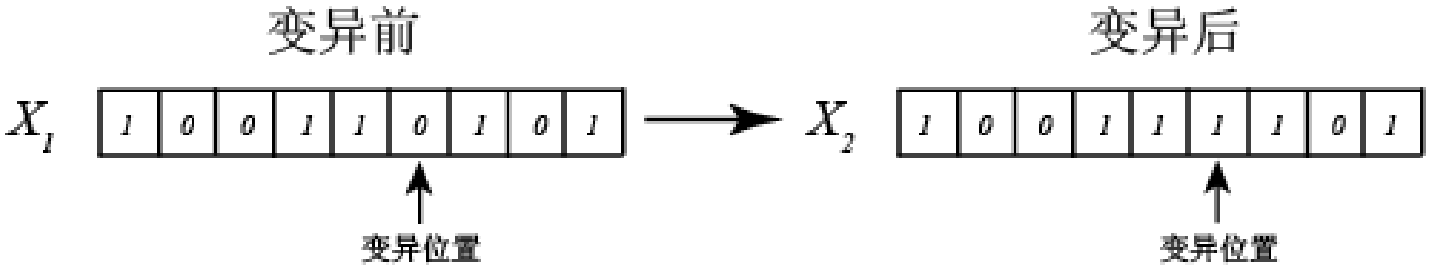
常用的变异操作方法有：实值变异法和二进制变异法等。

实值变异法

$$X' = X \pm 0.5L\Delta \qquad \Delta = \sum_{i=0}^{m=1} \frac{a(i)}{2}$$

a(i)*以概率*1/m*取值*1*，以概率*1-1/m*取值*0*，通常*m=20

二进制变异法



2 标准遗传算法

（7）确定遗传算法的有关运行参数，包括群体规模(Population Size)、迭代次数（一般取为100~500）、选择算子、交叉率、变异率等等。

popsize	取值较小时	提高运算和收敛速度	却降低了群体多样性，可能引起早熟现象	一般取为20~100
	取值较大时	含有较多模式，可提高GA搜索质量	但计算量增大，收敛速度降低	

（8）初始化群体。

- 初始群体一般随机产生
- 初始值最好能在解空间中均匀采样（收敛速度比较快 ）
- 对于非二进制编码，还要考虑所生成的染色体是否在可行区域内。

（9）计算群体中个体解码后的适应值。

（10）按照遗传策略，运用所选定的选择、交叉和变异算子作用于群体，生成下一代群体。

（11）判断群体性能是否满足某一指标或是否完成预定迭代次数，不满足则返回（9）。

2 标准遗传算法

2.6 未成熟收敛问题（Premature Convergence）

（1）未成熟收敛现象

未成熟收敛现象是遗传算法中的特有现象，且十分常见。它是指，当遗传算法还没找到全局最优解或满意解时，群体中不能再产生性能超过父代的后代，群体中的各个个体非常相似。未成熟收敛的重要特征是群体中个体结构的多样性急剧减少。

（2）未成熟收敛产生的原因

①理论上考虑的选择、交叉、变异操作都是绝对精确的，他们相互协调，能搜索到整个解空间，但实际不然；

②存在随机误差（主要包括取样误差和选择误差）；

③所求解的问题是遗传算法欺骗问题。

（3）未成熟收敛的防止

- 重新启动法
- 匹配策略（Mating Strategies）
- 重组策略（Recombination Strategies ）
- 替代策略（Replacement Strategies ）

} 提高多样性

2 标准遗传算法

❖遗传算法具体步骤

① 选择编码策略，把参数集合（可行解集合）转换染色体结构空间；

② 定义适应度函数，便于计算适应值；

③ 确定遗传策略，包括选择群体大小，选择、交叉、变异方法以及确定交叉概率、变异概率等遗传参数；

④ 随机产生初始化群体；

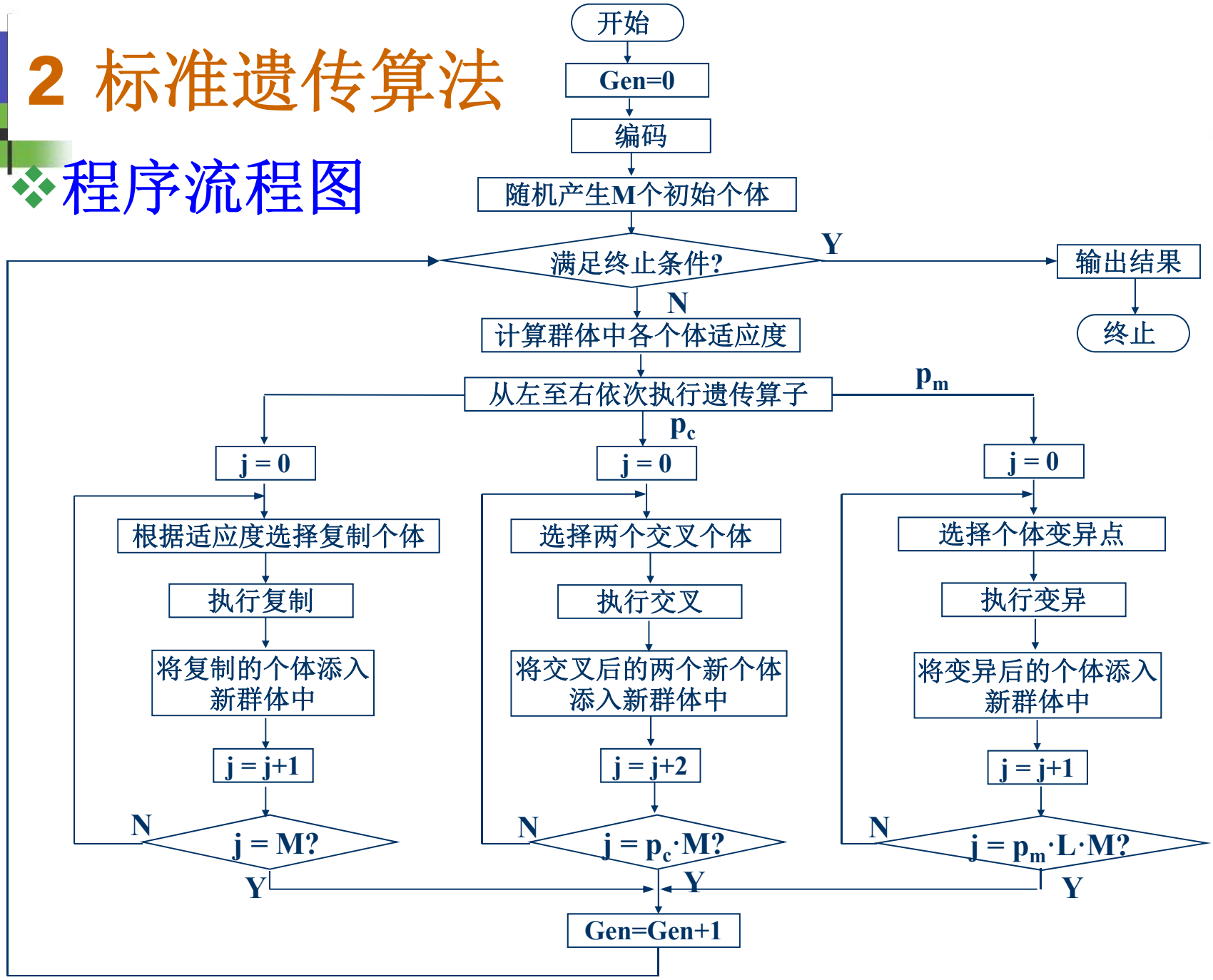
⑤ 计算群体中的个体或染色体解码后的适应值；

⑥ 按照遗传策略，运用选择、交叉和变异算子作用于群体，形成下一代群体；

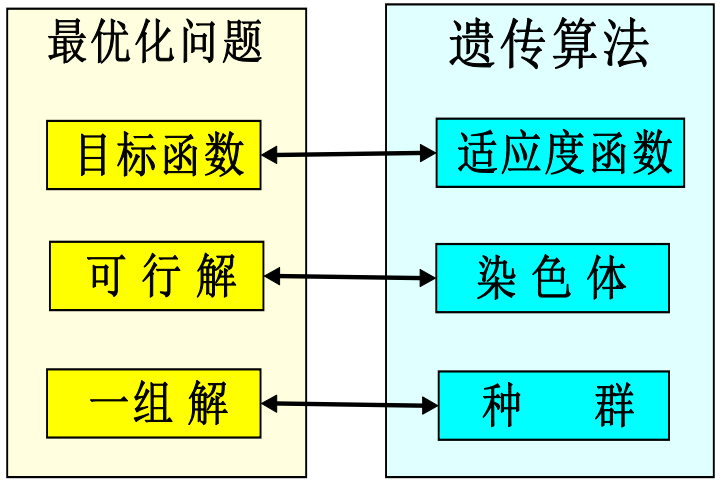
⑦ 判断群体性能是否满足某一指标，或者已完成预定的迭代次数，不满足则返回第五步，或者修改遗传策略再返回第六步.

2 标准遗传算法

❖程序流程图



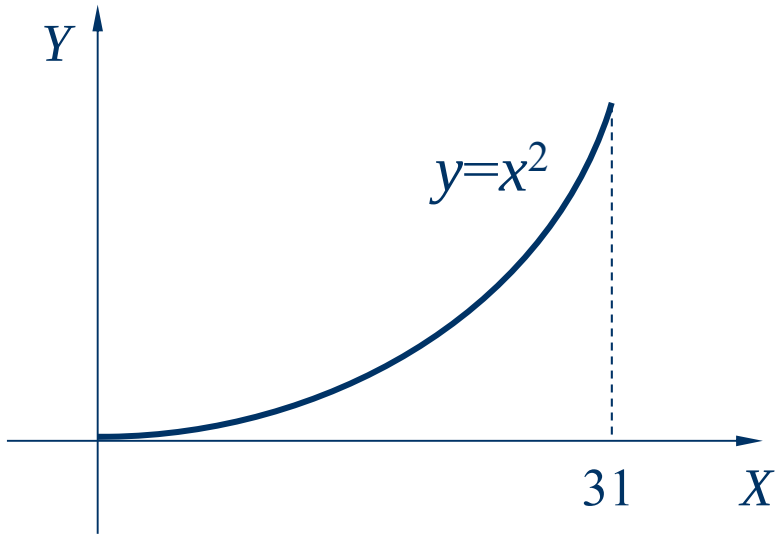
❖遗传算法与最优化问题：



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。

3、遗传算法简单举例：函数极值

例1 利用遗传算法求解区间［0,31］上的二次函数$y=x^2$的最大值，精度要求达到个位。



3、遗传算法简单举例：函数极值

分析

原问题可转化为在区间［0, 31］中搜索能使y取最大值的点a的问题。那么，［0, 31］ 中的点x就是个体, 函数值f(x)恰好就可以作为x的适应度，区间［0, 31］就是一个(解)空间 。这样, 只要能给出个体x的适当染色体编码, 该问题就可以用遗传算法来解决。

3、遗传算法简单举例：函数极值

(1) 设定种群规模,编码染色体，产生初始种群。

将种群规模设定为4；用5位二进制数编码染色体；取下列个体组成初始种群S1:

s_1= 13 (01101), s_2= 24 (11000)

s_3= 8 (01000), s_4= 19 (10011)

(2) 定义适应度函数,

取适应度函数：$f(x)=x^2$

3、遗传算法简单举例：函数极值

(3) 计算各代种群中的各个体的适应度, 并对其染色体进行遗传操作,直到适应度最高的个体(即31（11111）)出现为止。

首先计算种群S_1中各个体

$$s_1 = 13(01101), \quad s_2 = 24(11000)$$

$$s_3 = 8(01000), \quad s_4 = 19(10011)$$

的适应度$f(s_i)$。

容易求得：

$$f(s_1) = f(13) = 13^2 = 169$$

$$f(s_2) = f(24) = 24^2 = 576$$

$$f(s_3) = f(8) = 8^2 = 64$$

$$f(s_4) = f(19) = 19^2 = 361$$

3、遗传算法简单举例：函数极值

再计算种群S1中各个体的选择概率。

选择概率的计算公式为

$$P(x_i)=\frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)}$$

由此可求得

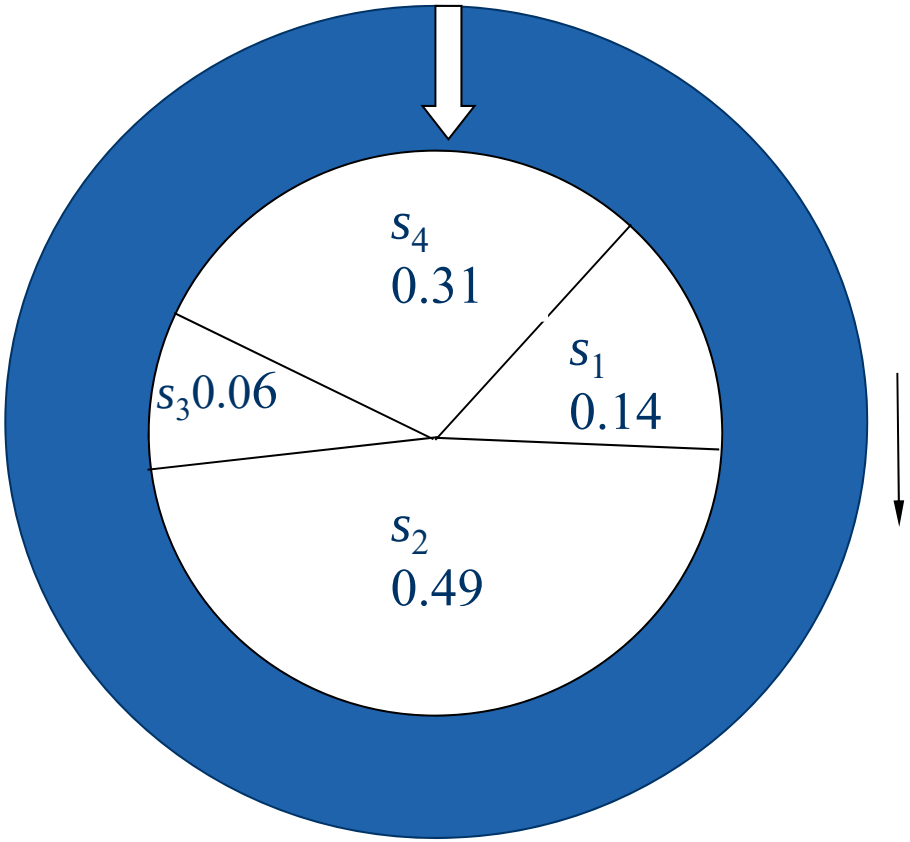
$$P(s_1)=P(13)=0.14$$

$$P(s_2)=P(24)=0.49$$

$$P(s_3)=P(8)=0.06$$

$$P(s_4)=P(19)=0.31$$

3、遗传算法简单举例：函数极值



轮盘赌选择示意图

3、遗传算法简单举例：函数极值

选择-复制

染色体	适应度	选择概率	选中次数
s_1=01101	169	0.14	1
s_2=11000	576	0.49	2
s_3=01000	64	0.06	0
s_4=10011	361	0.31	1

于是，经选择复制得群体：

s_1'=11000（24），s_2'=01101（13）

s_3'=11000（24），s_4'=10011（19）

3、遗传算法简单举例：函数极值

交叉

设交叉率p_c=100%，即S_1中的全体染色体都参加交叉运算。

设s_1'与s_2'配对，s_3'与s_4'配对。分别交换后两位基因，得新染色体：

s_1''=11001（25），s_2''=01100（12）

s_3''=11011（27），s_4''=10000（16）

3、遗传算法简单举例：函数极值

变异

设变异率p_m=0.001。这样，群体S_1中共有

$$5\times4\times0.001=0.02$$

位基因可以变异。0.02位显然不足1位，所以本轮遗传操作不做变异。

于是，得到第二代种群$\boldsymbol{S}_2$：

$\boldsymbol{s}_1$=11001（25）, $\boldsymbol{s}_2$=01100（12）

$\boldsymbol{s}_3$=11011（27）, $\boldsymbol{s}_4$=10000（16）

3、遗传算法简单举例：函数极值

第二代种群S_2中各染色体的情况

染色体	适应度	选择概率	估计的选中次数
$s_1=11001$	625	0.36	1
$s_2=01100$	144	0.08	0
$s_3=11011$	729	0.41	2
$s_4=10000$	256	0.15	1

3、遗传算法简单举例：函数极值

假设这一轮选择-复制操作中，种群S_2中的**4个染色体都被选中**，则得到群体：

s_1'=11001（25），s_2'= 11011（27）

s_3'=11011（27），s_4'= 10000（16）

做交叉运算，让s_1’与s_2’，s_3’与s_4’分别交换后三位基因，得

s_1”=11100（28）, s_2”= 01001（9）

s_3”=11000（24）, s_4”= 10011（19）

这一轮仍然不会发生变异。

于是，得第三代种群S3：

s_1=11100（28）, s_2=01001（9）

s_3=11000（24）, s_4=10011（19）

3、遗传算法简单举例：函数极值

第三代种群S_3中各染色体的情况

染色体	适应度	选择概率	估计的选中次数
$s_1=11100$	784	0.44	2
$s_2=01001$	81	0.04	0
$s_3=11000$	576	0.32	1
$s_4=10011$	361	0.20	1

3、遗传算法简单举例：函数极值

设这一轮的选择-复制结果为：

s_1'=11100（28），s_2'=11100（28）

s_3'=11000（24），s_4'=10011（19）

做交叉运算，让s_1’与s_4’，s_2’与s_3’分别交换后两位基因，得

s_1''=11111（31），s_2''=11100（28）

s_3''=11000（24），s_4''=10000（16）

这一轮仍然不会发生变异。

于是，得第四代种群S_4：

s_1=11111（31），s_2=11100（28）

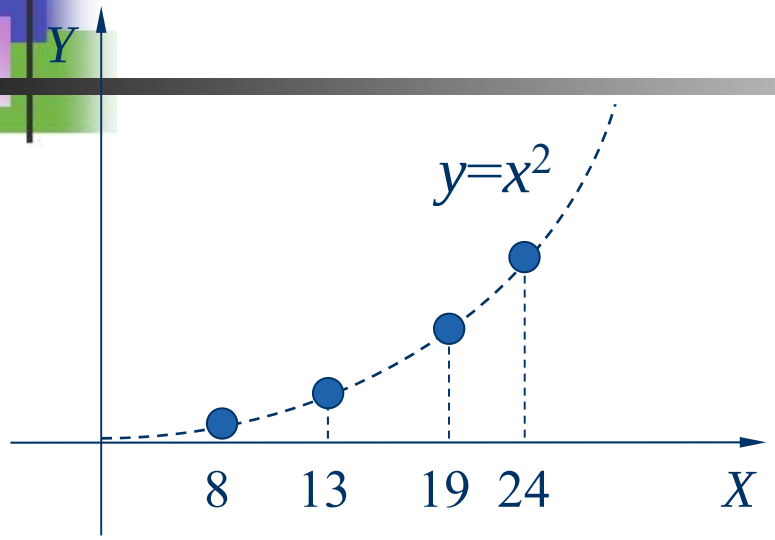
s_3=11000（24），s_4=10000（16）

3、遗传算法简单举例：函数极值

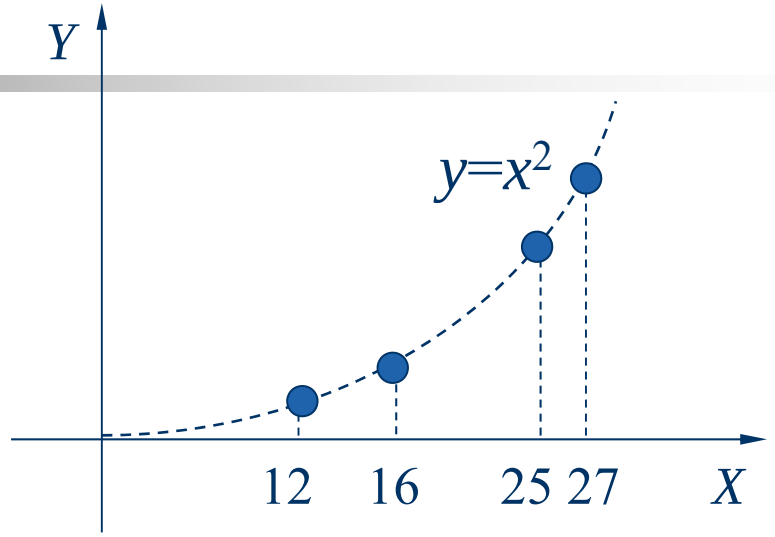
显然，在这一代种群中已经出现了适应度最高的染色体s_1=11111。于是，遗传操作终止，将染色体“11111”作为最终结果输出。

然后，将染色体“11111”解码为表现型，即得所求的最优解：31。

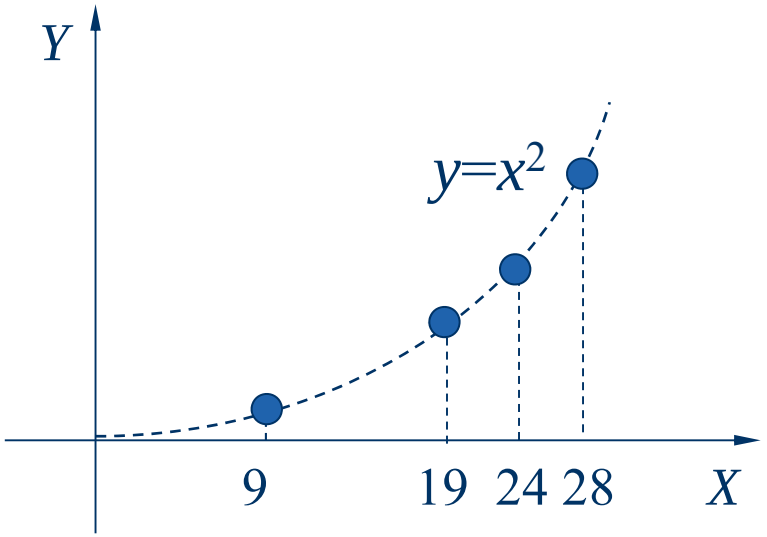
将31代入函数$y=x^2$中，即得原问题的解，即函数$y=x^2$的最大值为961。



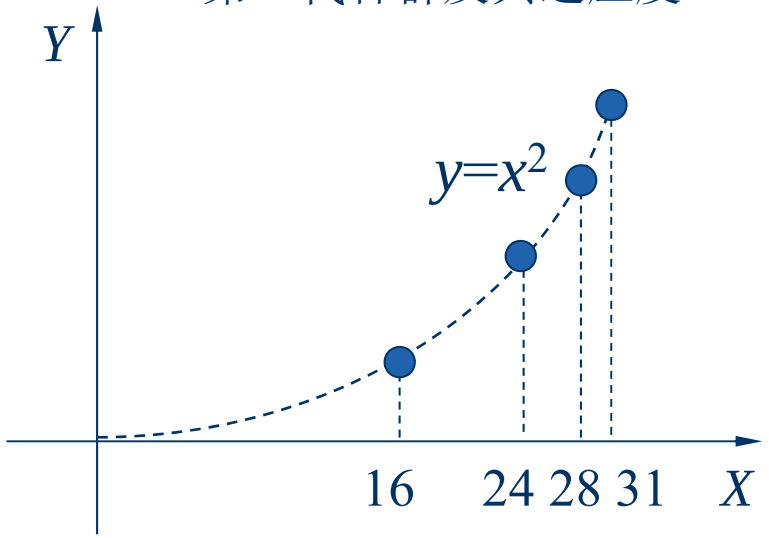
第一代种群及其适应度



第二代种群及其适应度



第三代种群及其适应度

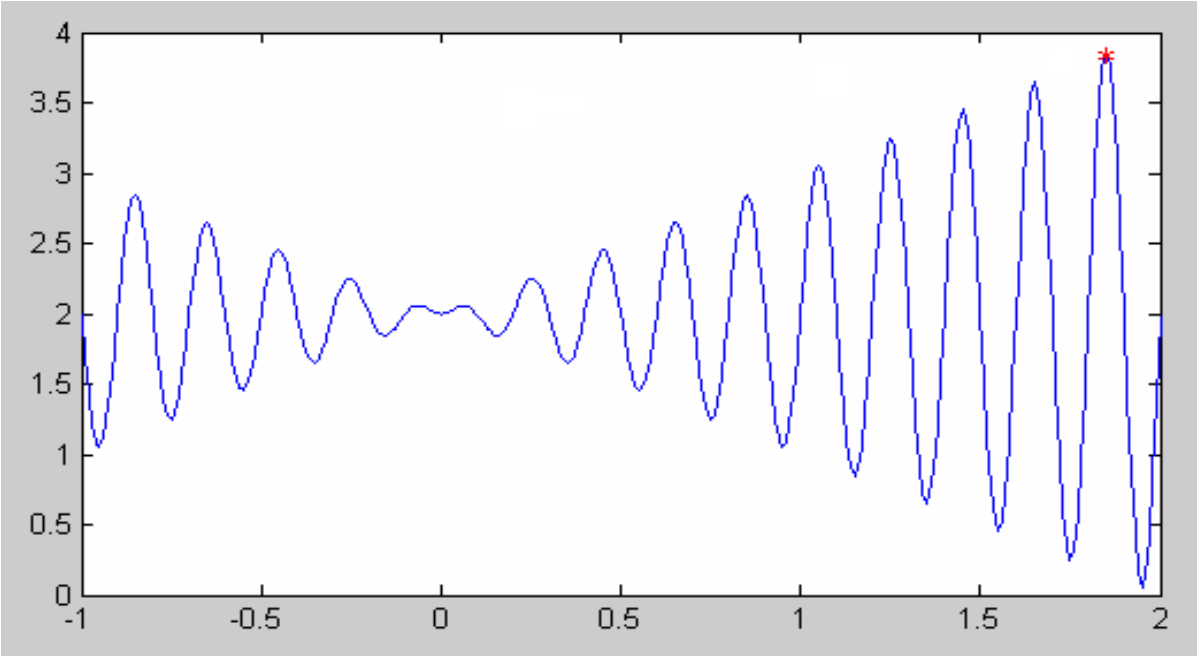


第四代种群及其适应度

❖ 求一元函数f(x)的最大值:

$$f(x) = x\sin(10\pi * x) + 2.0 \qquad x \in [-1,2]$$

要求求解精度到6位小数



- 编码

表现型：x

基因型：二进制编码（串长取决于求解精度）

按编码原理：假设要求求解精度到6位小数，区间长度为2-(-1)＝3，即需将区间分为$3/0.000001=3\times10^6$等份。

$$2097152=2^{21}<3000000<2^{22}=4194304$$

所以编码的二进制串长应为22位。

- 产生初始种群

  产生的方式：随机

  产生的结果：长度为22的二进制串

  产生的数量：种群的大小（规模），如30，50

  1111010011100001011000

  1100110011101010101110

  1010100011110010000100

  1011110010011100111001

  0001100101001100000011

  0000011010010000000000

  ……

- 计算适应度

  直接用目标函数作为适应度函数

  ①解码：将个体s转化为[-1,2]区间的实数：

  $s=<100010111011010100011 1>$ → $x=0.637197$

  ②计算x的函数值（适应度）：

  $f(x)=x\sin(10\pi x)+2.0=2.586345$

- 遗传操作

  选择：比例选择法；

  交叉：单点交叉；

  变异：小概率变异

- 模拟结果

  设置的参数：

  种群大小50；交叉概率0.75；变异概率0.05；最大代数200。

  得到的最佳个体：

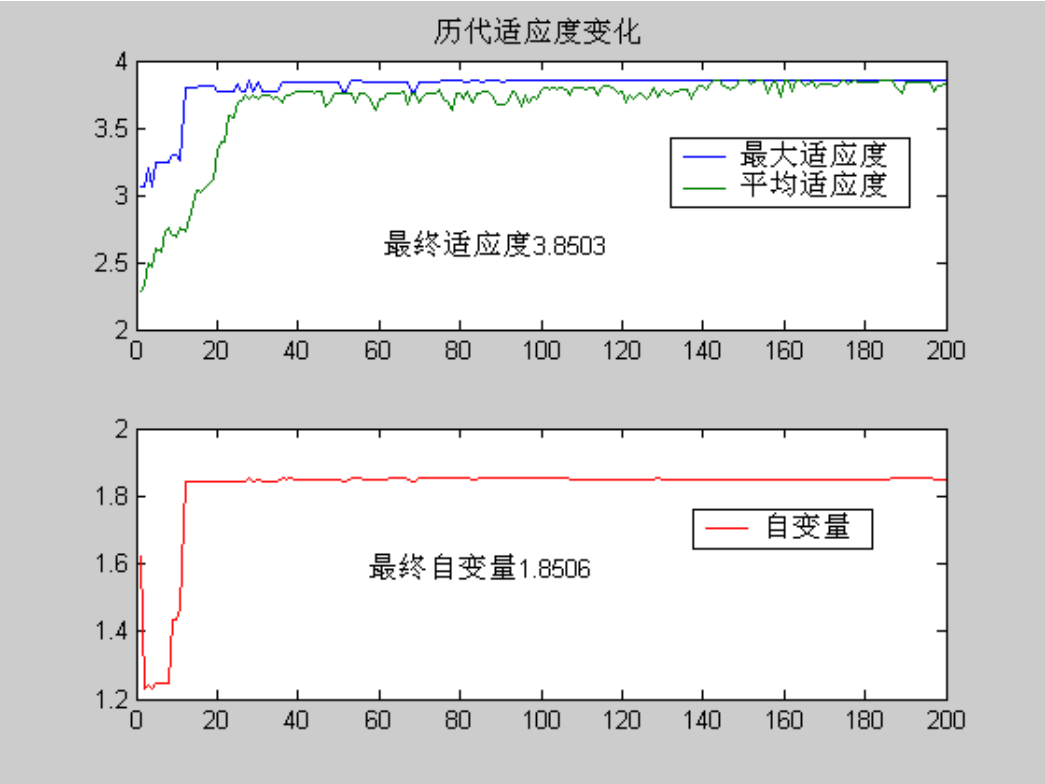
  s_{max}=<111100110011011111100>;

  x_{max}=1.8506;

  $f(x_{max})$=3.8503;

❖ 模拟结果

进化的过程：



世代数	自变量	适应度
1	1.4495	3.4494
9	1.8395	3.7412
17	1.8512	3.8499
30	1.8505	3.8503
50	1.8506	3.8503
80	1.8506	3.8503
120	1.8506	3.8503
200	1.8506	3.8503

❖考虑问题：能否用遗传算法解决以下问题

1、 $\max(f(x,\ y)) = x^2 y^2 \sin(10\pi * (x + e^y)) + 2.0 \quad x,\ y \in [-1,2]$

2、 $$\begin{aligned} &Minimize \quad f(x) \\ &\begin{cases} g_i(x) \le 0, & i = 1, \cdots, m \\ h_j(x) \le 0, & j = 1, \cdots, n \\ l_i \le x_i \le u_i \end{cases} \end{aligned}$$

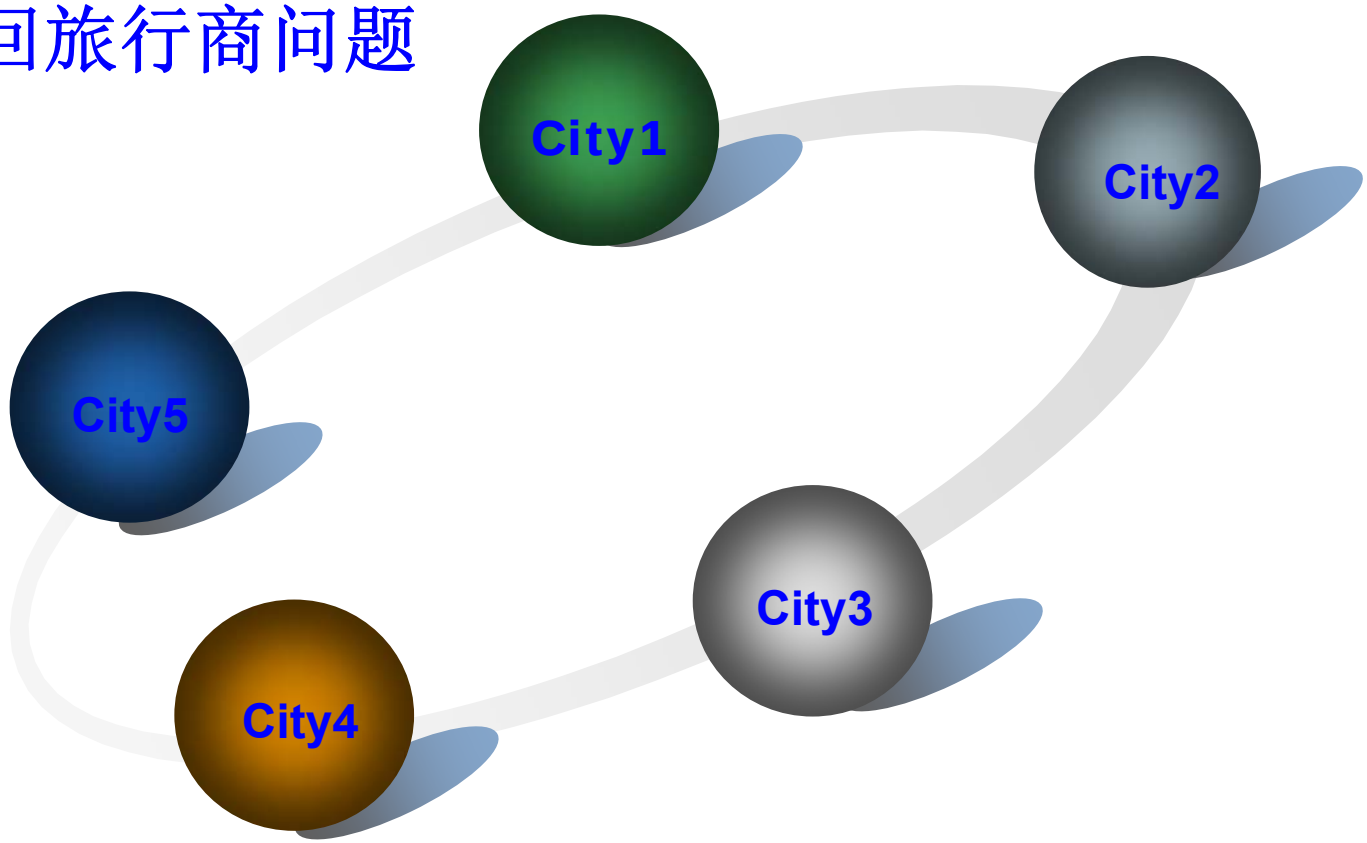
3、 $$\begin{aligned} &Minimize \ [f_1(x), f_2(x), \cdots, f_k(x)] \\ &\begin{cases} g_i(x) \le 0, & i = 1, \cdots, m \\ h_j(x) \le 0, & j = 1, \cdots, n \\ l_i \le x_i \le u_i \end{cases} \end{aligned}$$

Contents of Section 4

4　遗传算法求解巡回旅行商问题

4 遗传算法求解巡回旅行商问题

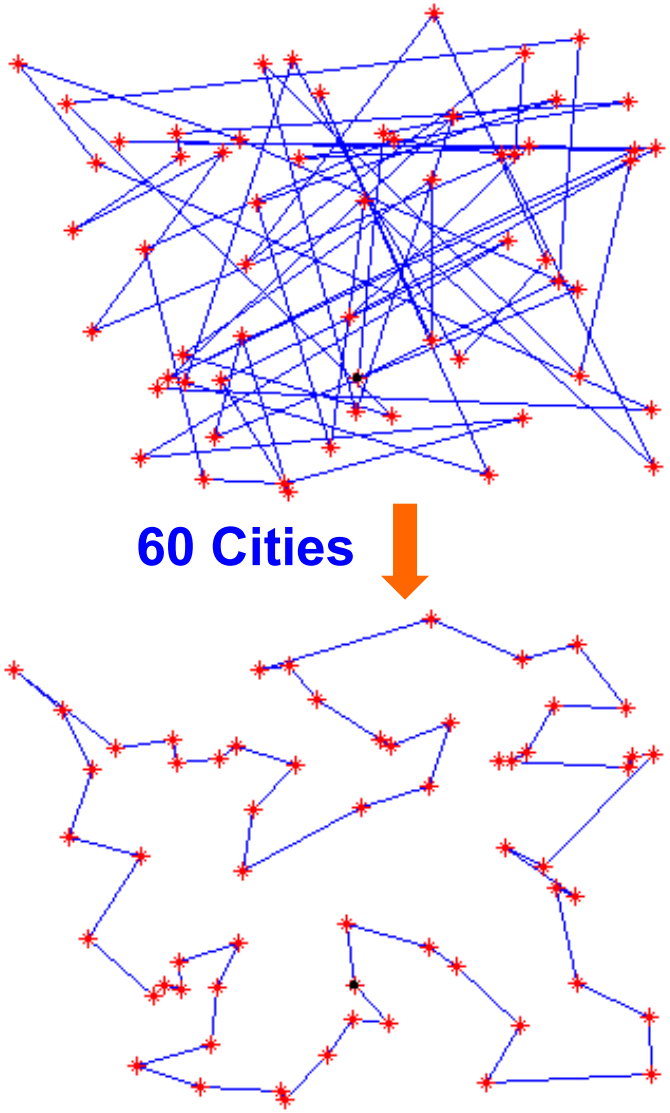
4.1 巡回旅行商问题

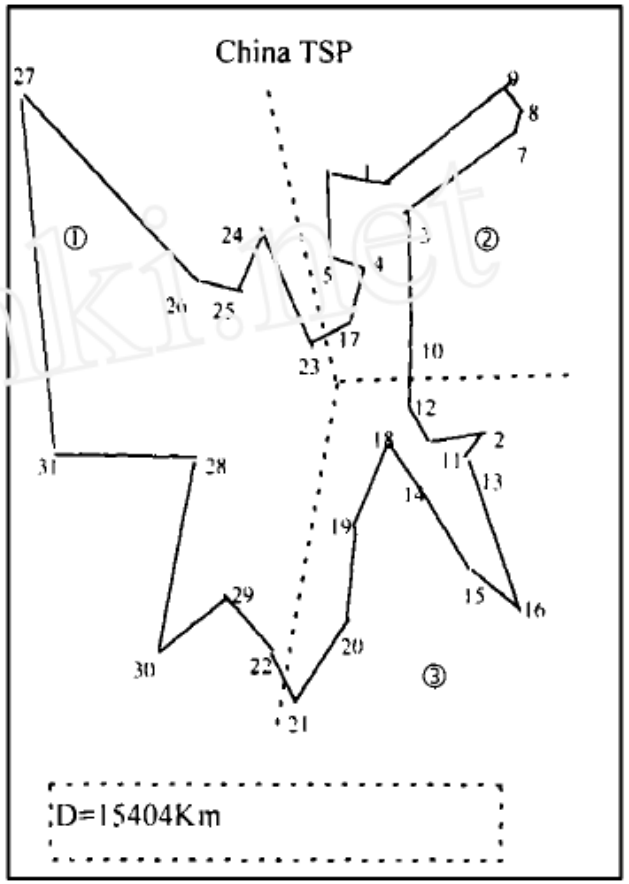


巡回旅行商问题（**Traveling salesman problem, TSP**）可描述如下：已知**N**个城市之间的相互距离，现有一推销员必须遍历这**N**个城市，并且每个城市只能访问一次，最后又必须返回出发城市，如何安排他访问这些城市的次序，使其旅行路线总长度最短？

4 遗传算法求解巡回旅行商问题

An optimal TSP tour through Germany's 15 largest cities. It is the shortest among 43 589 145 600[1] possible tours visiting each city exactly once.

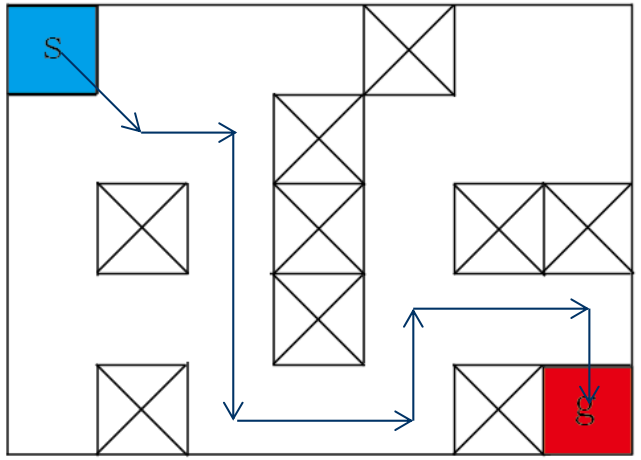




CHN31问题

4 遗传算法求解巡回旅行商问题

TSP具有广泛的应用背景和重要理论价值，特别在机器人运动规划中得到许多应用，例如：移动机器人的全局路径规划问题（如右上图）、焊接机器人的任务规划问题（如右下图）等等。



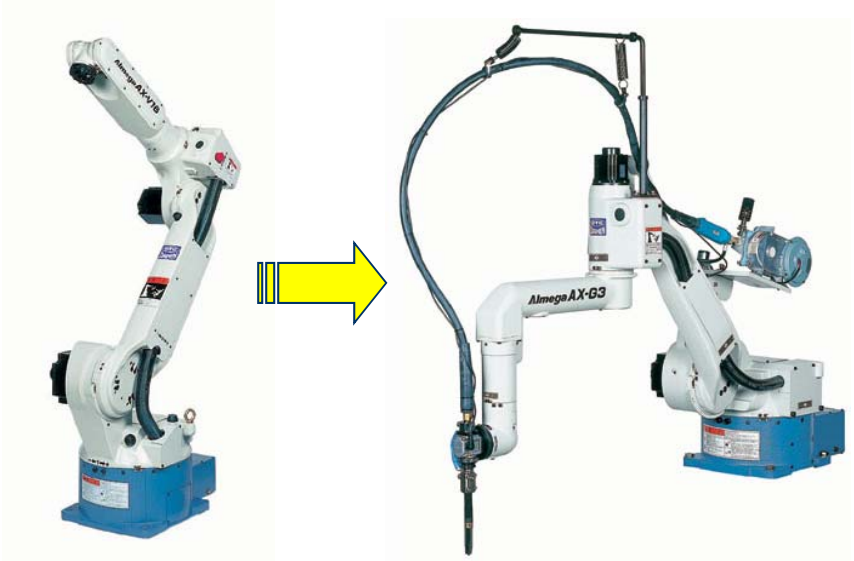
但是，巡回旅行商问题中可能的路径数目与城市数目**N**呈指数型增长，所有的旅程路线组合数为

$$\frac{(n-1)!}{2n}$$

其已经被证明属于**NP**难题。例如**30**个城市的路线对应约有

$$\frac{(30-1)!}{2\times 30}=1.4736\times 10^{29} \quad 条$$

对庞大的搜索空间寻求最优解，常规方法和现有的计算工具存在着诸多的计算困难。



4 遗传算法求解巡回旅行商问题

本例所用30个城市的XY坐标：

序号	X坐标	Y坐标	序号	X坐标	Y坐标	序号	X坐标	Y坐标
1	18	54	11	71	44	21	7	64
2	87	76	12	64	60	22	22	60
3	74	78	13	68	58	23	25	62
4	71	71	14	83	69	24	62	32
5	25	38	15	58	69	25	87	7
6	58	35	16	54	62	26	91	38
7	4	50	17	51	67	27	83	46
8	13	40	18	37	84	28	41	26
9	18	40	19	41	94	29	45	21
10	24	42	20	2	99	30	44	35

4 遗传算法求解巡回旅行商问题

4.2 基本操作

（1）编码与解码

采用对访问城市序列进行排列组合的方法编码，即某个巡回路径的染色体是该巡回路径的城市序列。对于N（N为城市总数）进制编码，即每个基因仅从1到N得整数里面取一个值，每个个体的长度为N。

利用矩阵来存储：

$$\begin{bmatrix} 8 & \cdots & 17 & 546.7 \\ 26 & \cdots & 2 & 493.6 \\ \vdots & \vdots & \vdots & \vdots \\ 10 & \cdots & 9 & 632.1 \end{bmatrix}_{s\times 31}$$

一行的前30个元素为一个个体

30个城市的访问次序

该种访问次序路径的距离

根据编码方法，一次求解得出的最优解（个体）是所访问的城市的次序，需要转换成相应的城市坐标进行输出，则只需将个体的染色体值作为存储30个城市坐标的矩阵的下标来引用，输出对应的矩阵元素，便可实现解码。

4 遗传算法求解巡回旅行商问题

（2）适应度函数：在TSP问题中，用路径的总长度作为适应度函数来衡量求解结果是否最优，路径越短对应的个体越优，其适应度值应越大。

两城市间的距离为：$d_{ij}=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}$

个体代表的路径的总长度为：

$$L=\sum_{k=1}^{29}d(k)$$

则可采用倒数法将适应度函数取为：$fitness=1/L$

（3）选择操作：将群体中适应度较大的C个个体直接替换适应度较小的C个个体。其中C值与选择算子和群体规模的关系在这里取为：

$$C=Select\_Operator\times Population\_Size$$

4 遗传算法求解巡回旅行商问题

（4）交叉操作

本例中采用有序交叉执行交叉操作。有序交叉能够有效地继承双亲的部分基因成分，达到了进化的遗传功能，使该遗传算法并不盲目搜索，二是趋向于使群体具有更多的优良基因。交叉后，考察父个体与子个体的适应度来决定是否更新种群。具体操作过程如下（以0~9编码为例）：

父个体：（“|”表示交叉点）

A_1：（0 1 2 | 3 4 5 6 | 7 8 9）

A_2：（4 2 9 | 0 8 5 3 | 1 7 6）

第三步：保留对应位置的间部分

A_1：（× × 9 | 3 4 5 6 | 1 7 ×）

A_2：（8 1 2 | 0 8 5 3 | 7 4 9）

4 遗传算法求解巡回旅行商问题

（5）变异操作

变异操作中，若变异后子代的适应度值更加优异，则保留子代染色体，否则仍保留父代染色体。本例中，采用倒置变异法。例如：假设当前个体为“5678412390”，如果当前随机值$p \in [0, 1] \geq p_{mutation}$，则随机选择来自同一个体的两个点（设为“8”和“2”），执行变异操作，即倒置该两点的中间部分。产生的新个体为“5672148390”。

变异前父代个体：**5 6 7 | 8 4 1 2 | 3 9 0**

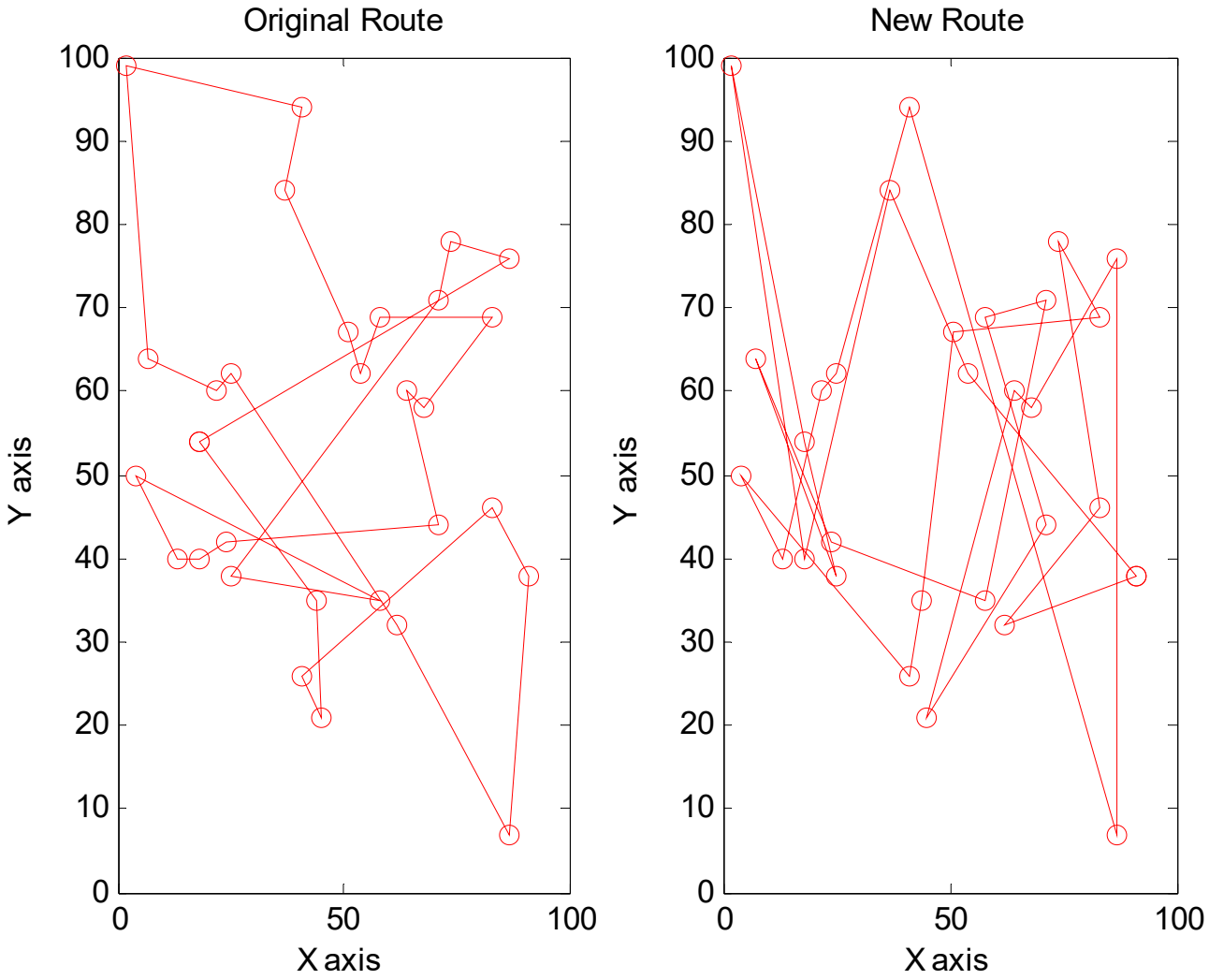
变异后子代个体：**5 6 7 | | 3 9 0**

（6）群体初始化

```
pop=zeros(s,t);
for i=1:s
      pop(i,1:t-1)=randperm(t-1);
end
```

4 遗传算法求解巡回旅行商问题

4.3 计算仿真结果



迁移代数：

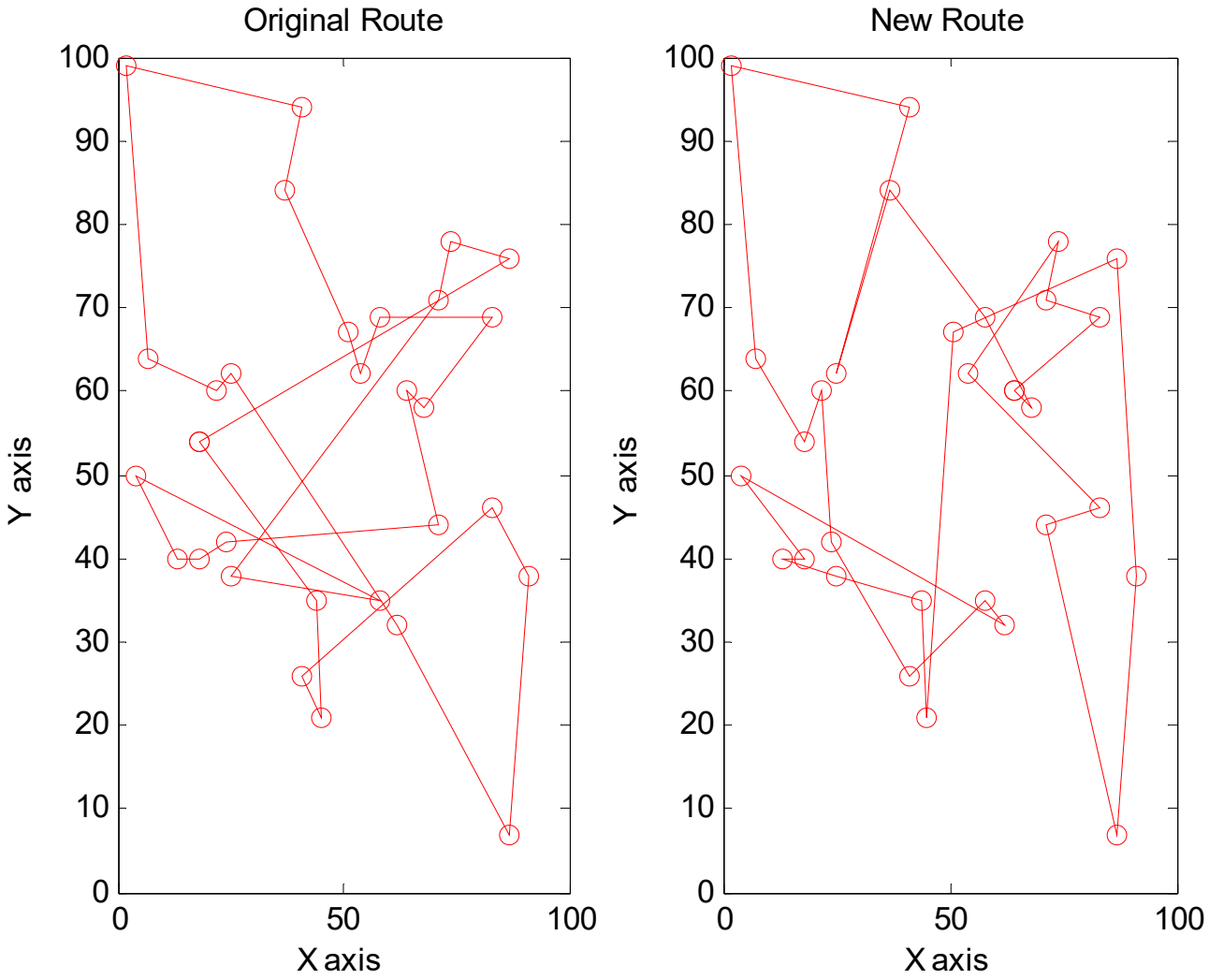
50

路径长度：

990.0829

4遗传算法求解巡回旅行商问题

4.3 计算仿真结果



迁移代数：

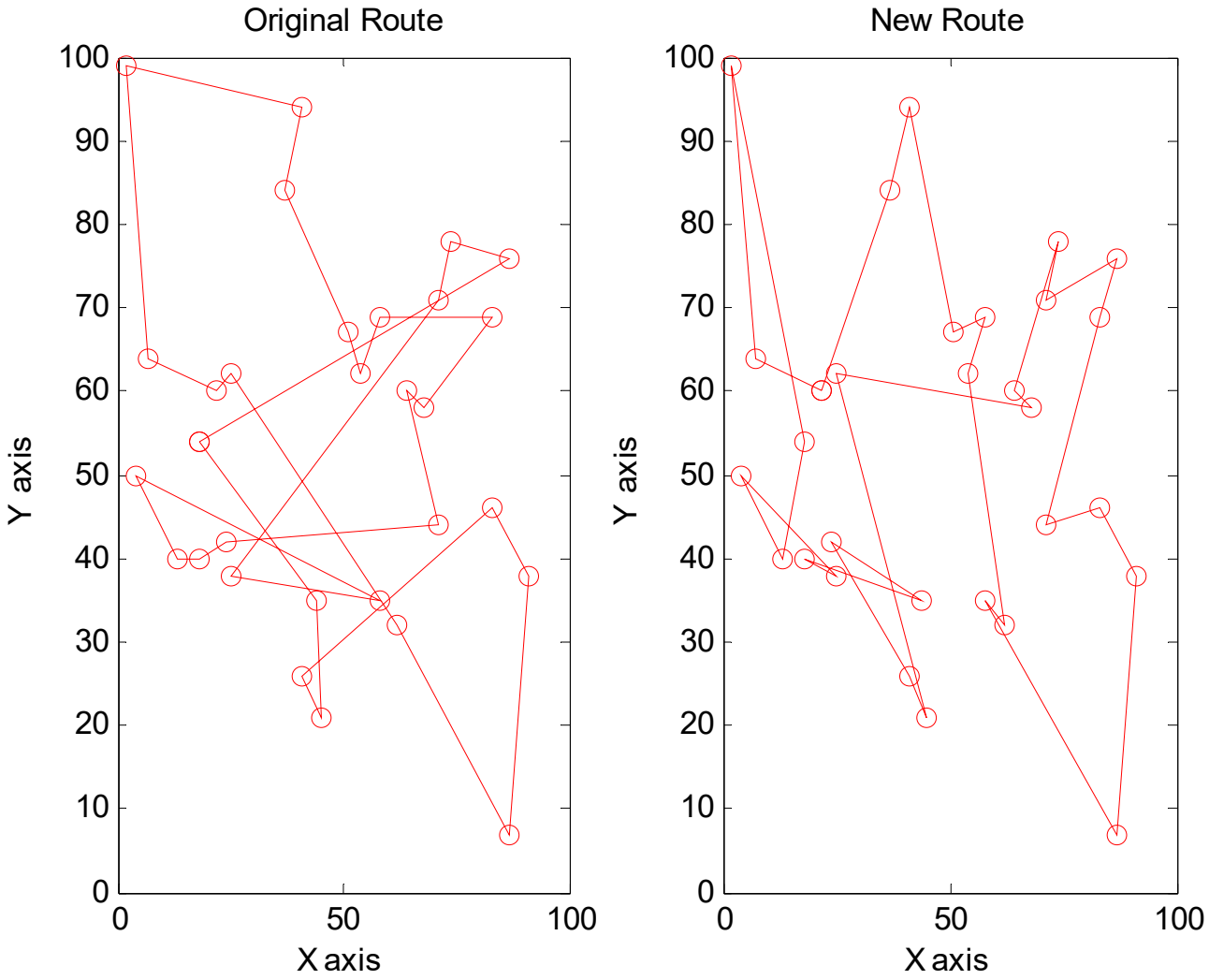
100

路径长度：

701.7754

4 遗传算法求解巡回旅行商问题

4.3 计算仿真结果



迁移代数：

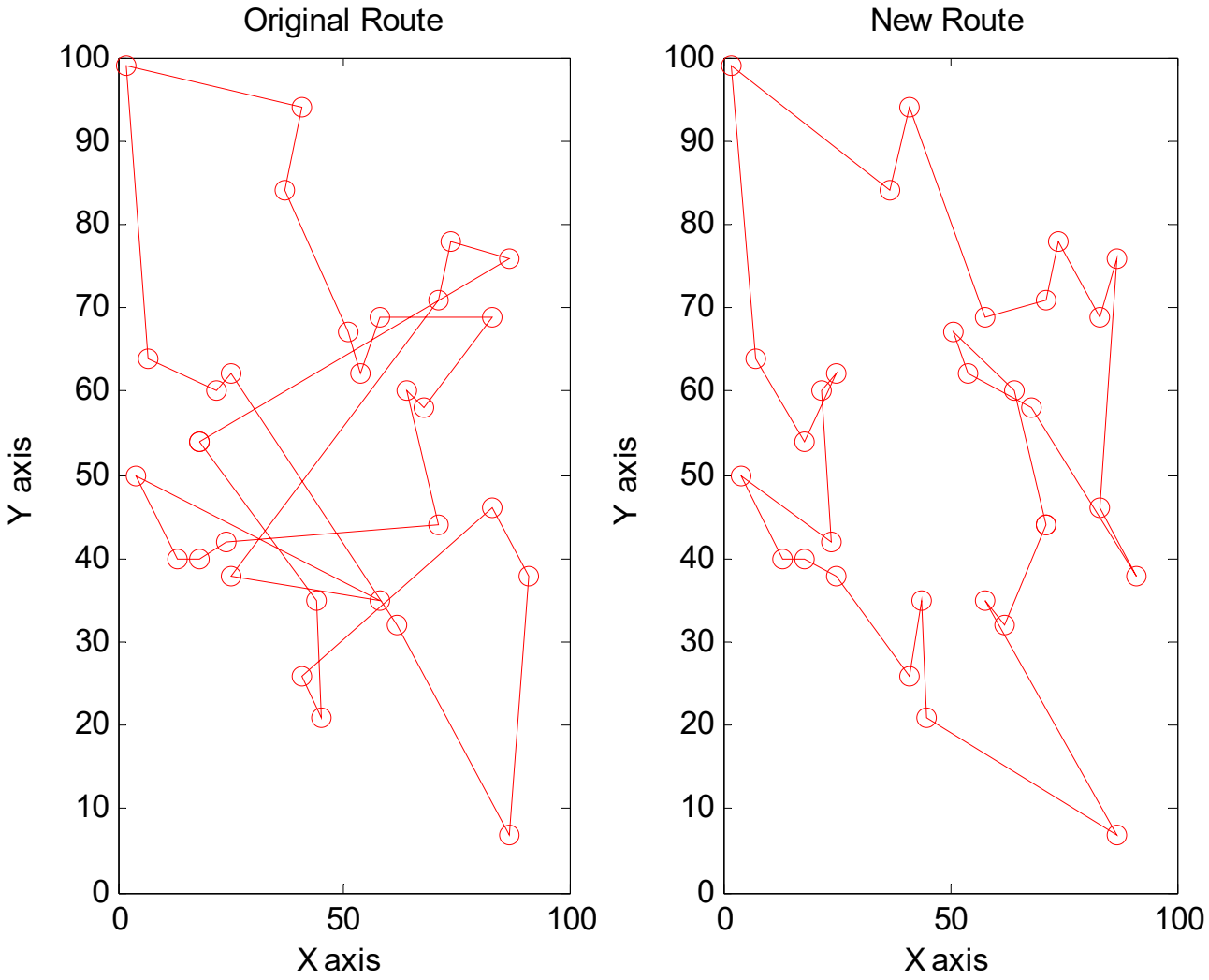
150

路径长度：

624.1821

4 遗传算法求解巡回旅行商问题

4.3 计算仿真结果



迁移代数：

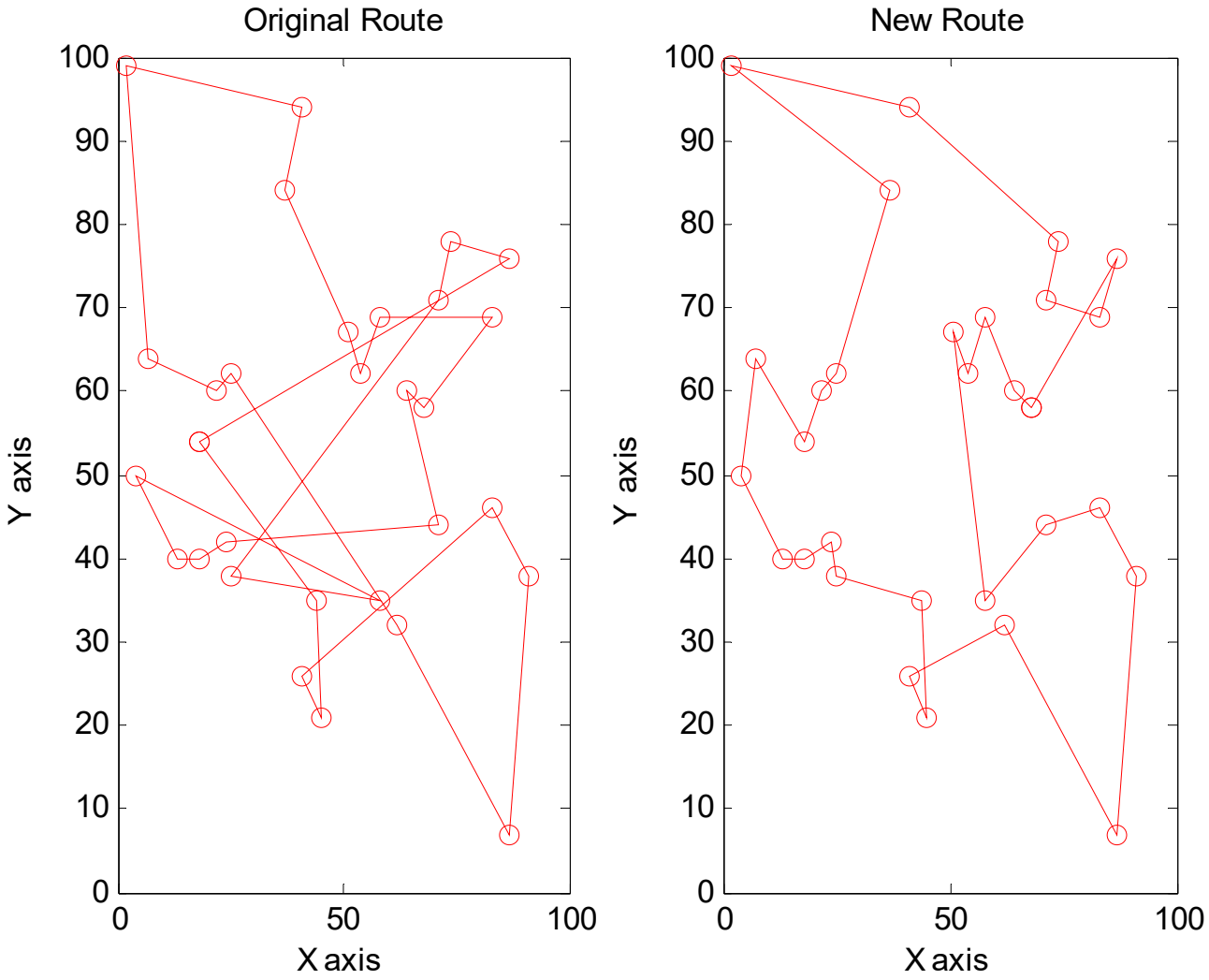
200

路径长度：

523.2674

4 遗传算法求解巡回旅行商问题

4.3 计算仿真结果



迁移代数：

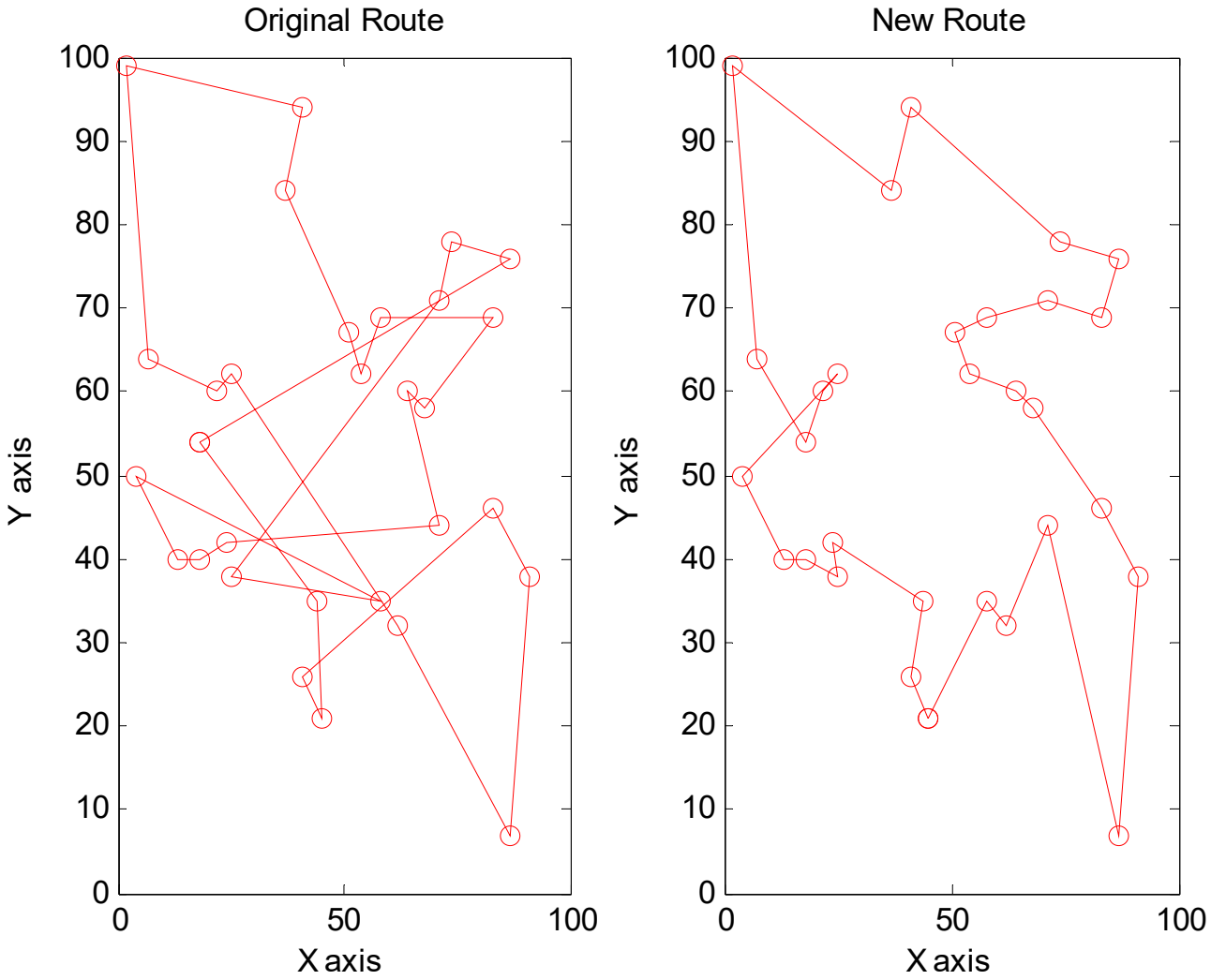
250

路径长度：

491.4063

4 遗传算法求解巡回旅行商问题

4.3 计算仿真结果



迁移代数：

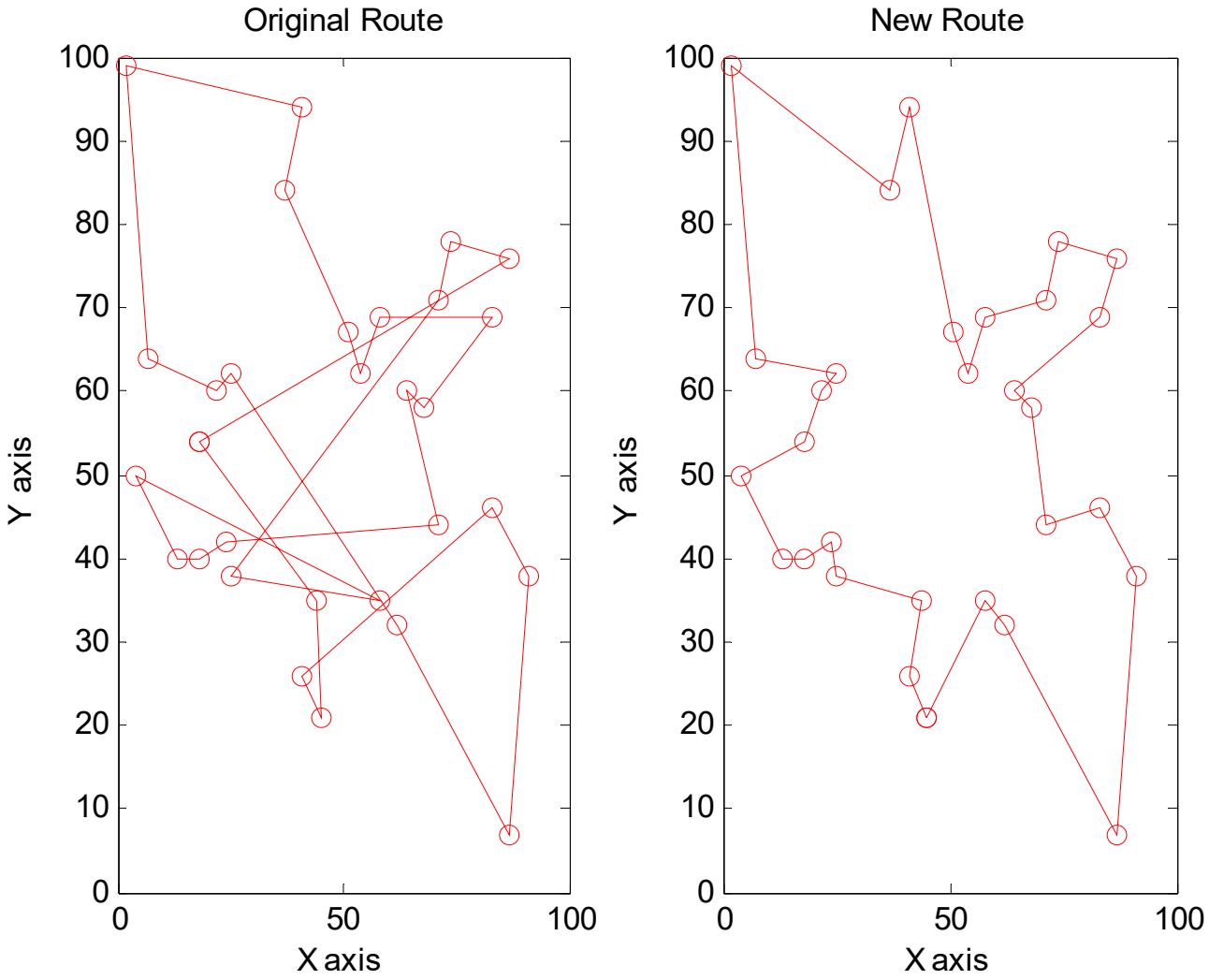
300

路径长度：

453.1959

4 遗传算法求解巡回旅行商问题

4.3 计算仿真结果



迁移代数：

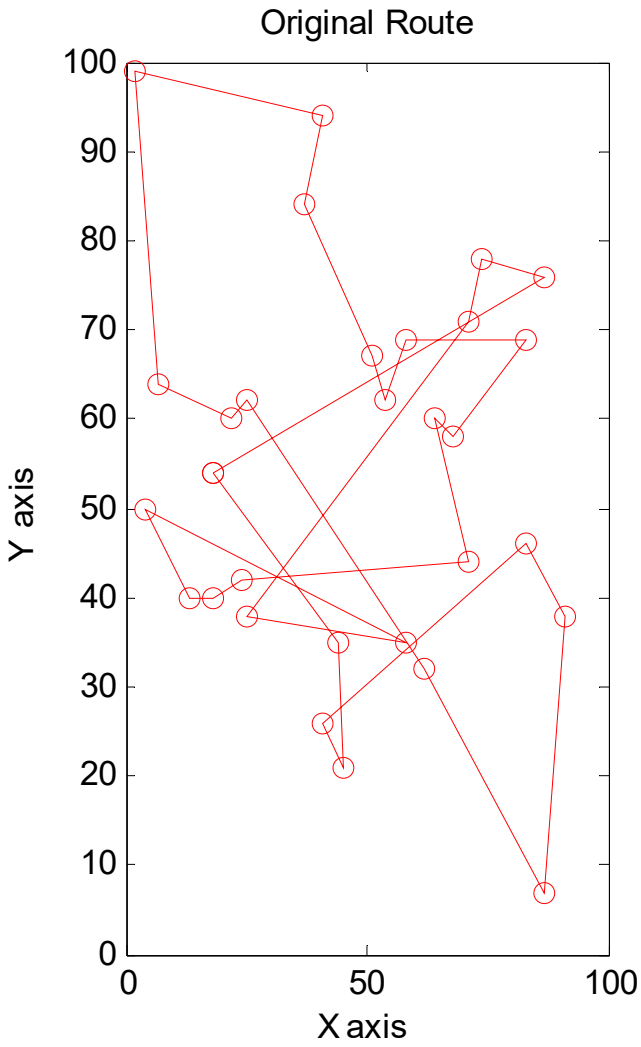
350

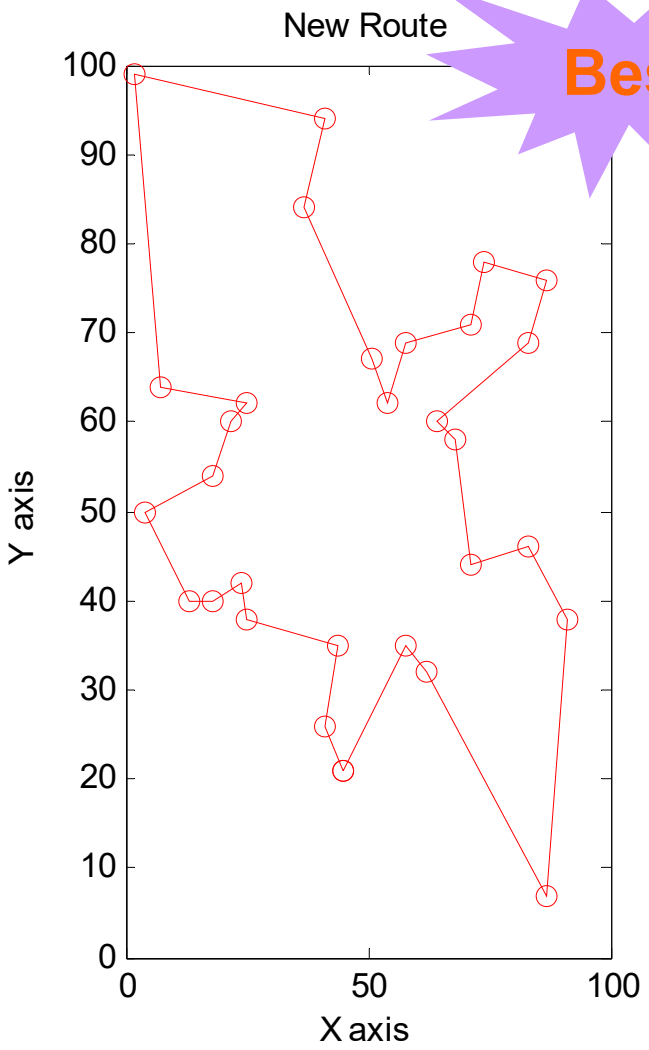
路径长度：

430.3986

4 遗传算法求解巡回旅行商问题

4.3 计算仿真结果





Best

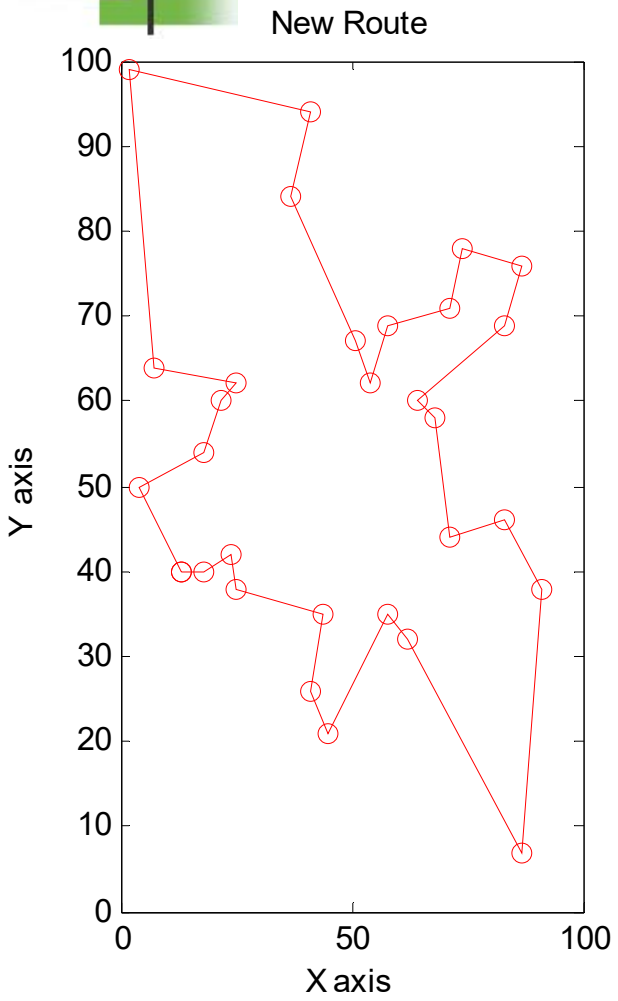
迁移代数：

400

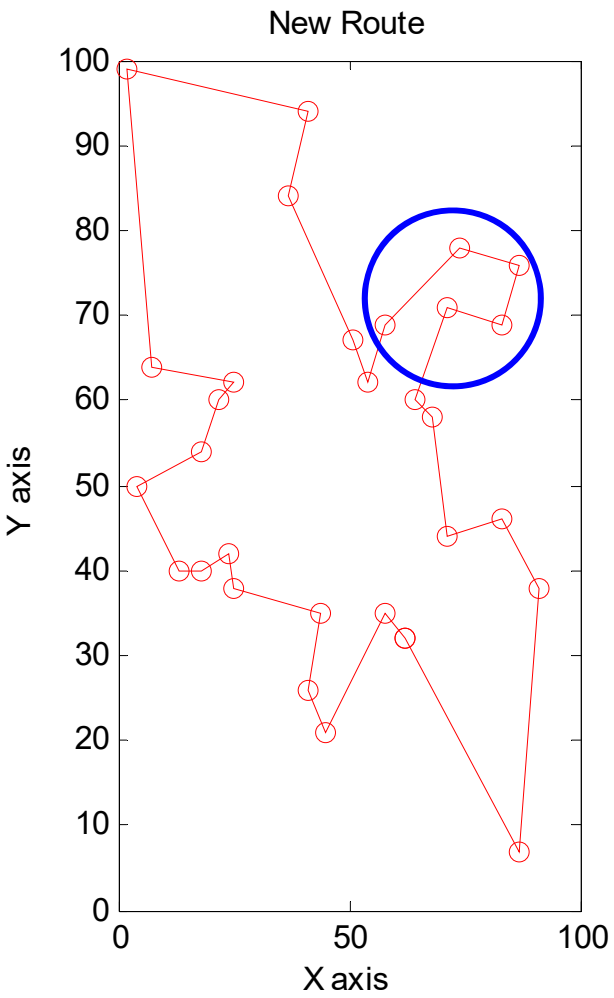
路径长度：

424.8693

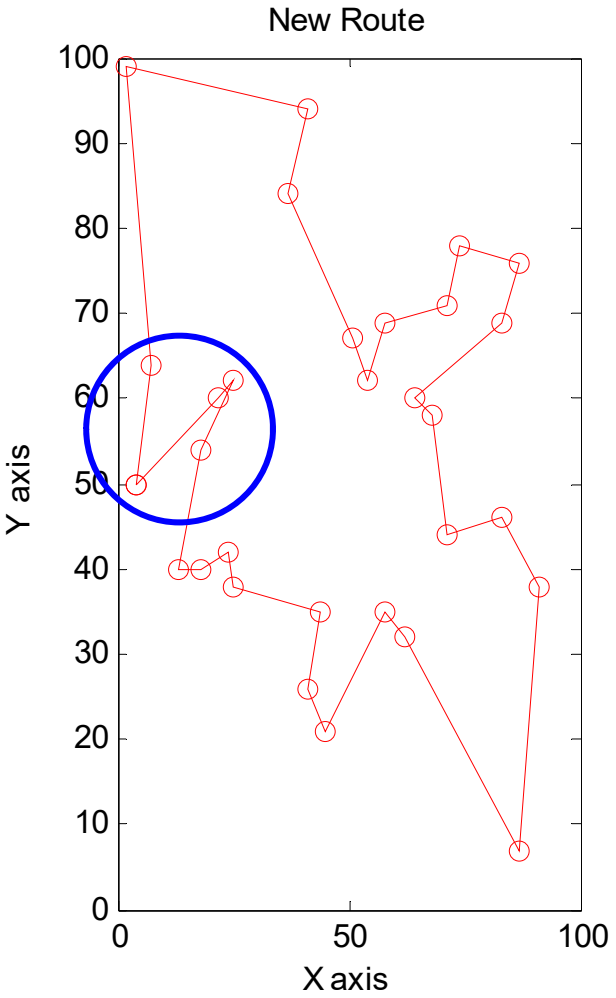
4 遗传算法求解巡回旅行商问题



距离为**424.78Km**的
访问次序（最优）

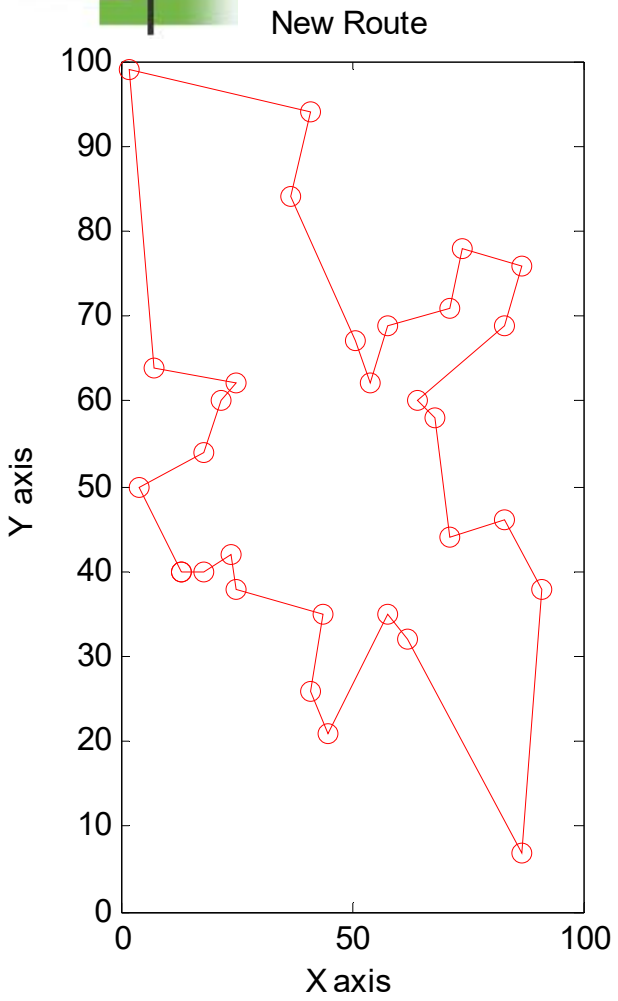


距离为**426.64Km**
的访问次序

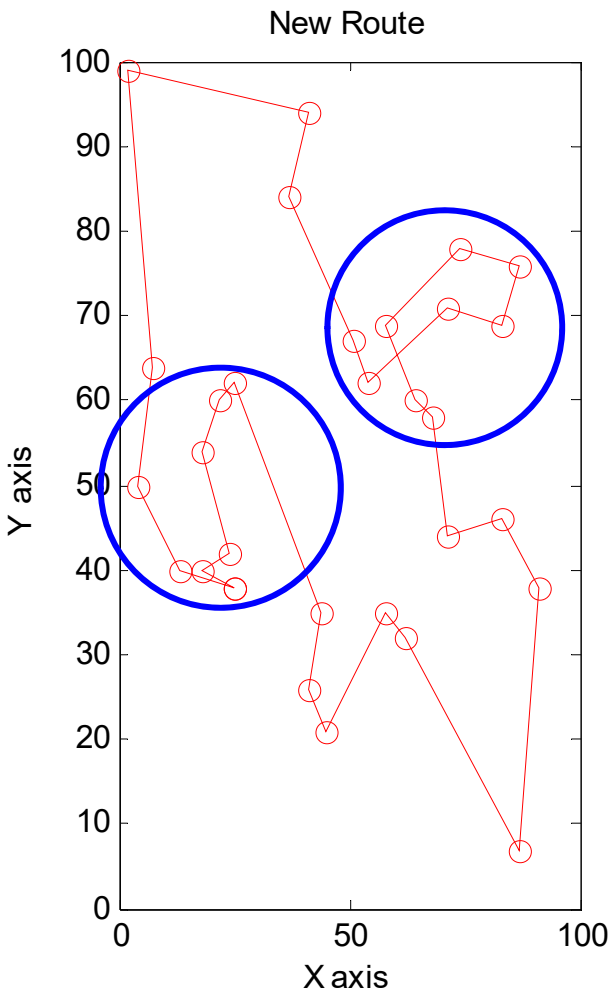


距离为**431.94Km**
的访问次序

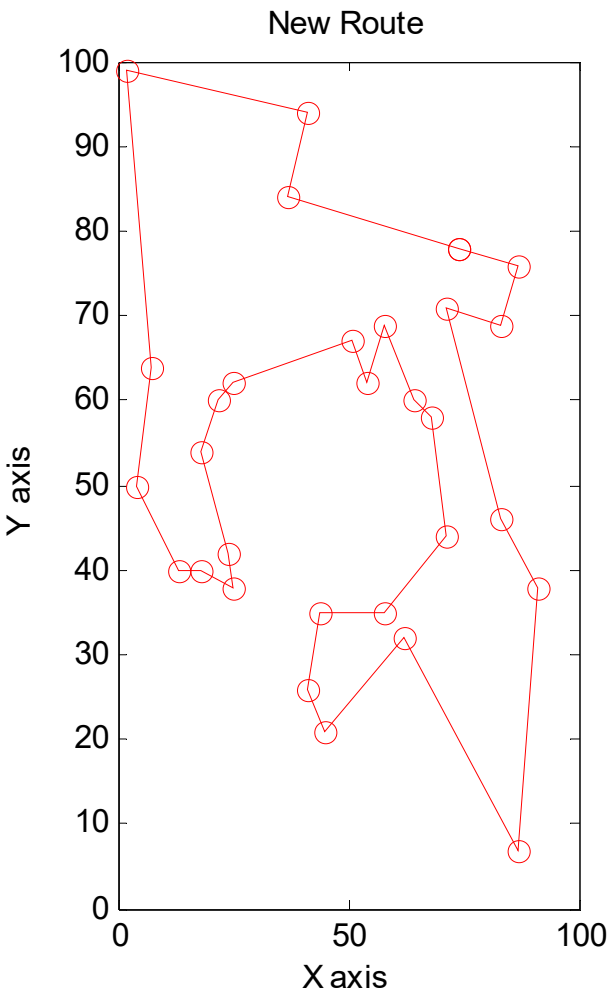
4 遗传算法求解巡回旅行商问题



距离为**424.78Km**的访问次序（最优）



距离为**454.75Km**的访



距离为**466.30Km**的访

4 遗传算法求解巡回旅行商问题

4.4 关于遗传算法操作算子的验证

Series No.	Population Size		Selection operator		Crossover Rate		Mutation Rate		Results	
	A		B		C		D		X_i	Y_i
1	1	100	1	1%	1	40%	1	0.1%	440.67	15.80
2	1	100	2	2%	2	65%	2	1.0%	440.61	15.75
3	1	100	3	3%	3	90%	3	10.0%	497.59	72.72
4	2	300	1	1%	2	65%	3	10.0%	431.12	6.25
5	2	300	2	2%	3	90%	1	0.1%	480.92	56.05
6	2	300	3	3%	1	40%	2	1.0%	429.06	4.19
7	3	500	1	1%	3	90%	2	1.0%	470.76	45.89
8	3	500	2	2%	1	40%	3	10.0%	426.22	1.35
9	3	500	3	3%	2	65%	1	0.1%	430.59	5.72
K_{1j}	104.27		67.95		21.34		77.57		sum(Y_i)	
K_{2j}	66.48		73.14		27.72		65.82		223.71	
K_{3j}	52.96		82.62		174.66		80.32			
W_{1j}	9.90		-2.21		-17.74		1.00			
W_{2j}	-2.70		-0.48		-15.62		-2.92		aver(Y_i)	
W_{3j}	-7.20		2.68		33.36		1.92		24.86	
R_j	17.10		4.89		51.11		4.83			
Fators	Primary C-A-B-D Secondary									
Optimal Plan	A3-B2-C1-D3									

“实验数据”课程所做的正交试验极差分析结果（迁移**500**代后退出的结果）。

4 遗传算法求解巡回旅行商问题

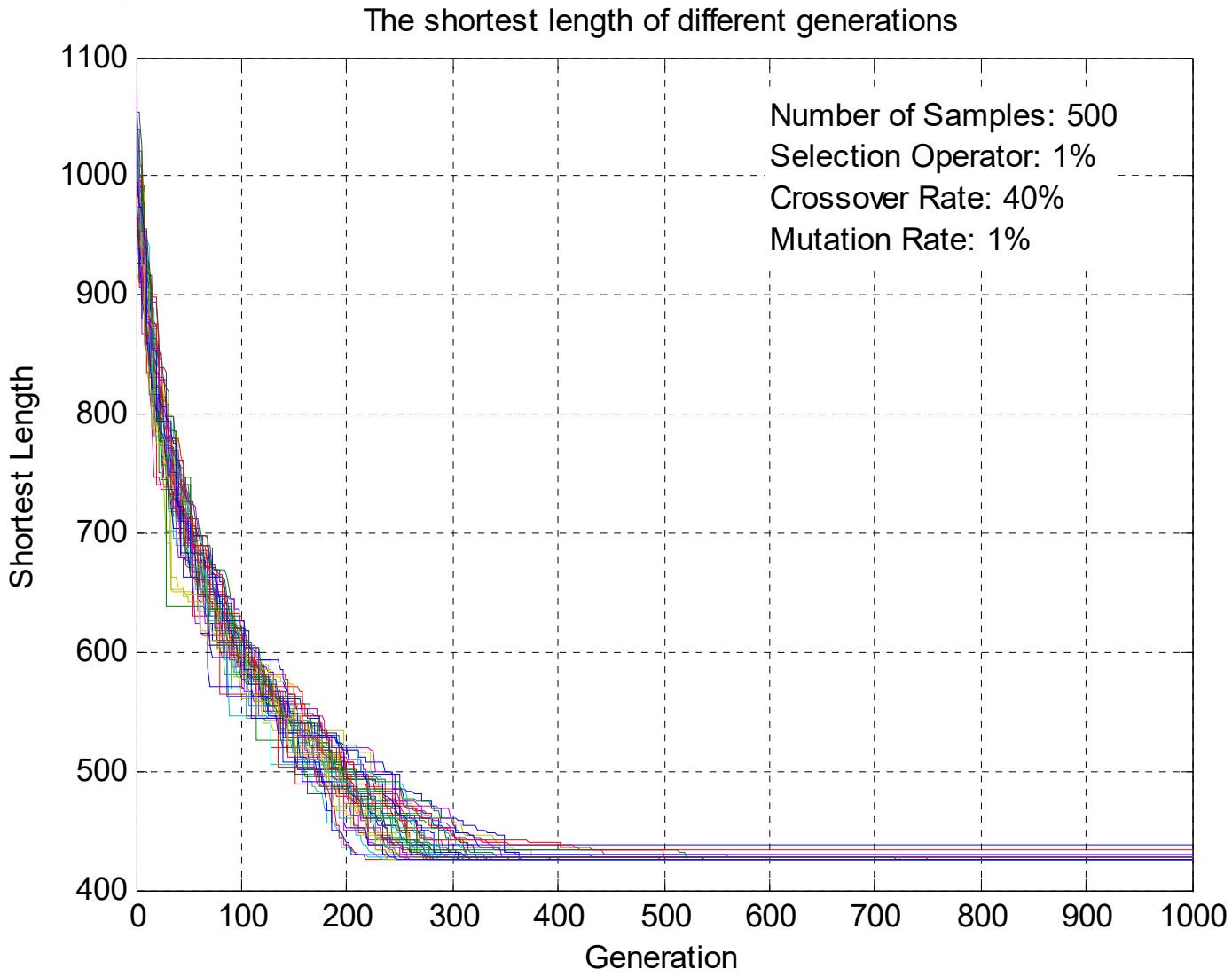
对于上表，有（验证）以下基本结论：

（1）遗传算法搜索求解能力与四个因素有关：群体规模、选择算子、交叉率和变异率 。

（2）从主到次依次为：交叉率——群体规模——选择算子——变异率。

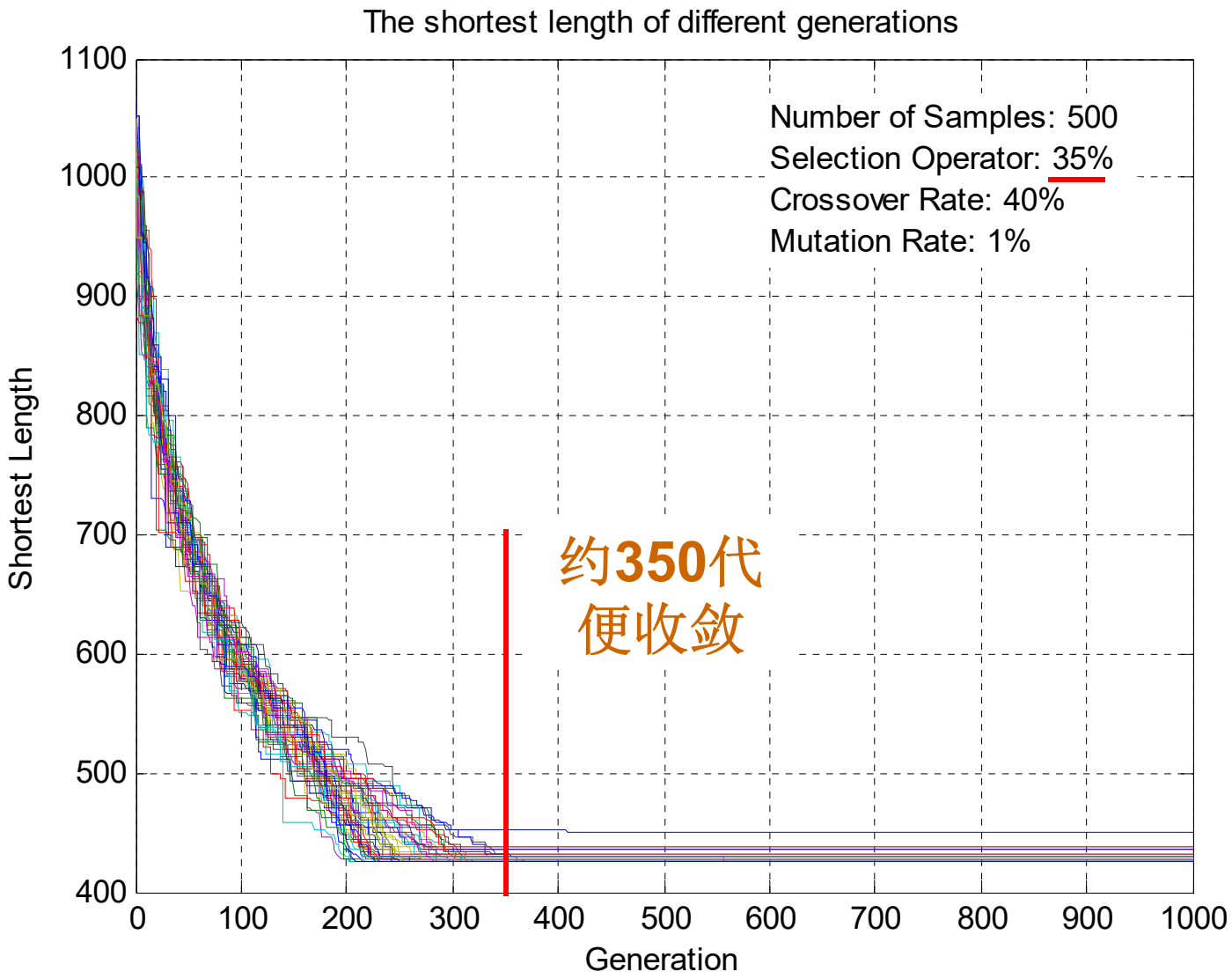
（3）A3-B2-C1-D3是优选方案。

4 遗传算法求解巡回旅行商问题



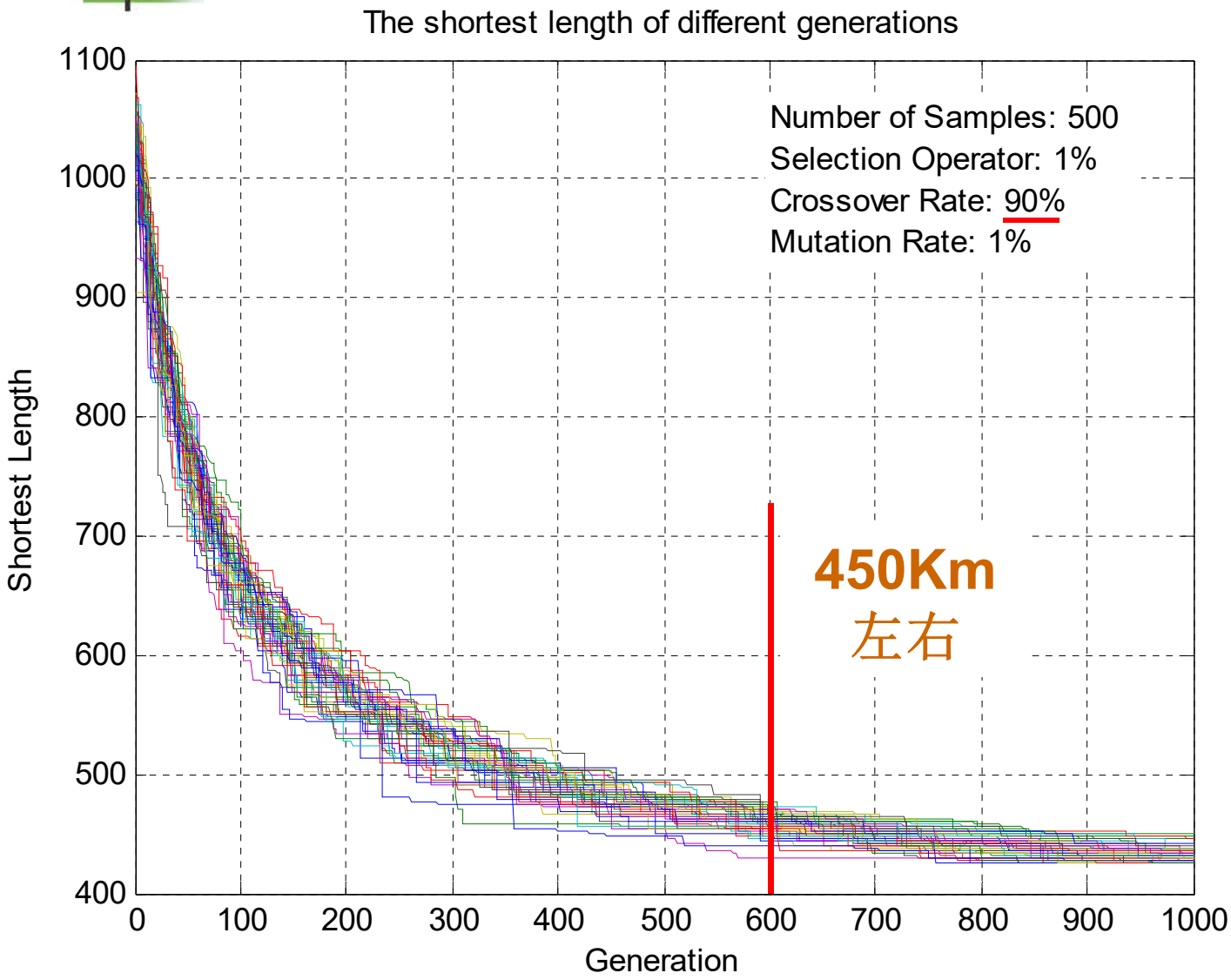
左图（进行**50**次独立运算求解，每次迁移**1000**代，有**36**次能收敛到全局最优解）是比较优的参数组合。实际上可看出，迭代进行到**450**代之后，所得到得最优个体基本不再发生变化，且其最优路径与真实的最有路径差距非常小。

4 遗传算法求解巡回旅行商问题



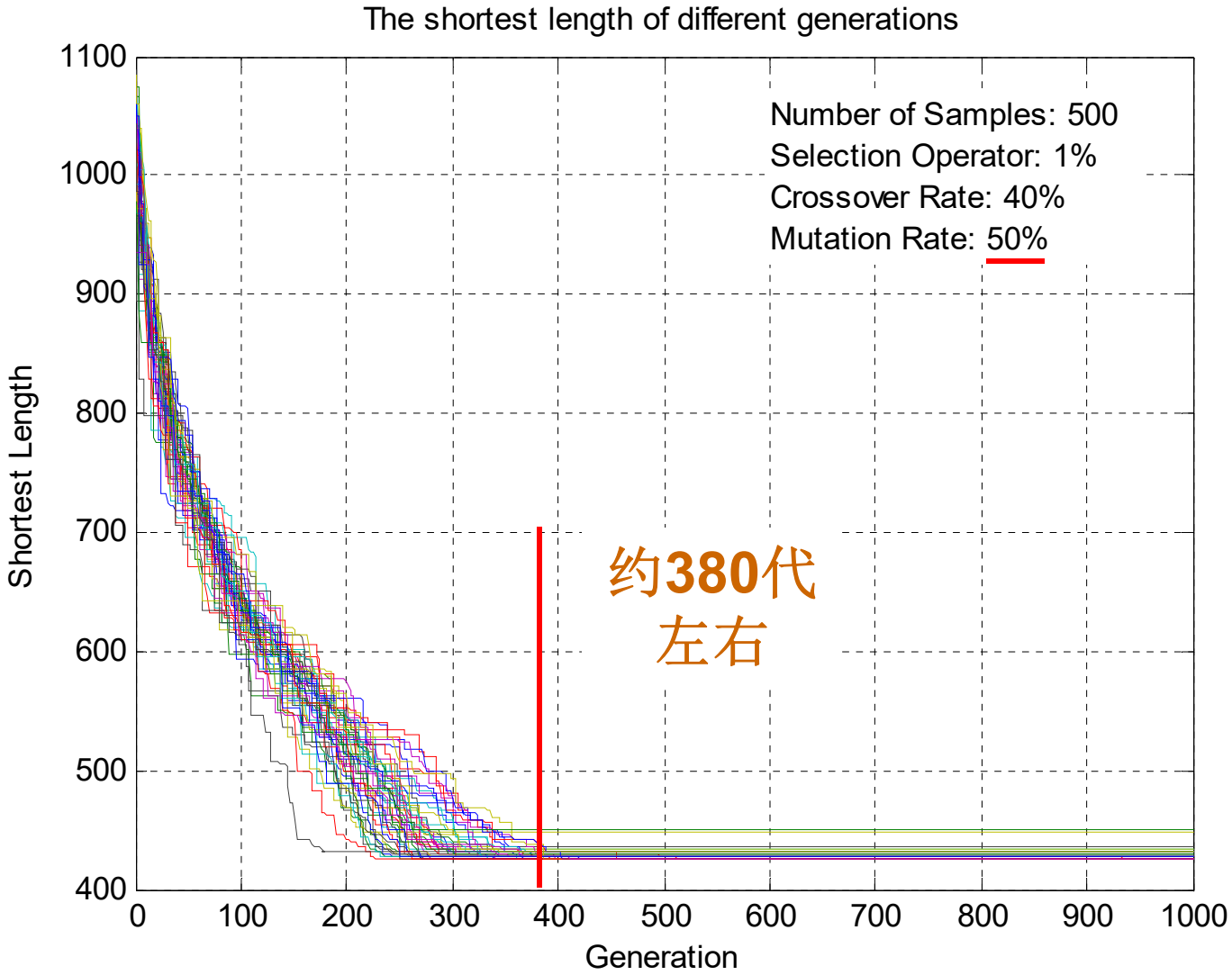
左图（进行**50**次独立运算求解，每次迁移**1000**代，有**24**次能收敛到全局最优解）表明：选择算子取值太大，收敛速度很快，但陷入局部最优解的可能性大大提高，而基本上不可能再跳出来。

4 遗传算法求解巡回旅行商问题



左图（进行**50**次独立运算求解，每次迭代**1000**代，仅有**6**次能收敛到全局最优解）表明：交叉率选取太大，导致群体中的优良模式遭到破坏，产生较大的代沟，从而使搜索走向随机化。

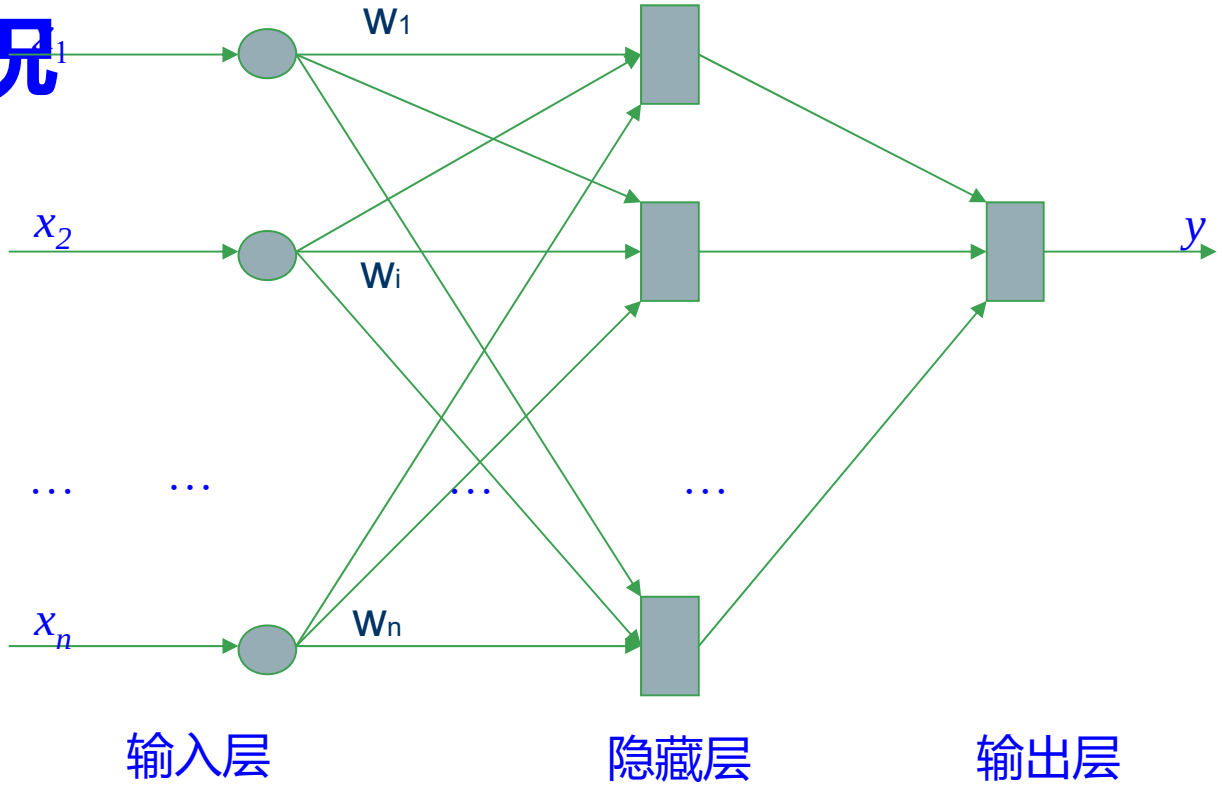
4 遗传算法求解巡回旅行商问题



左图（进行**50**次独立运算求解，每次迁移**1000**代，有**18**次能收敛到全局最优解）表明：变异率选取太大，遗传算法几乎退化为随机搜索，陷入局部最优解后比较难跳出来。

5遗传算法优化神经网络

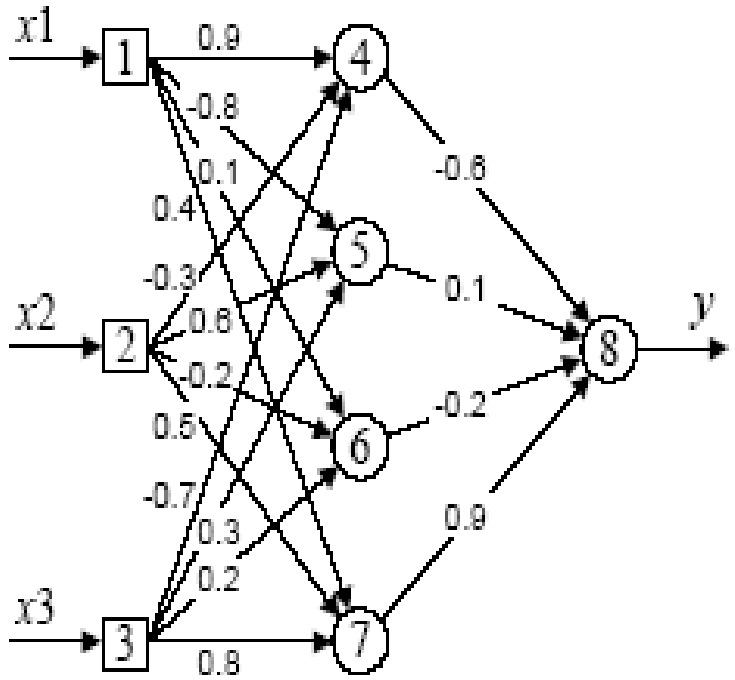
- GA优化NN的权重（结构确定）
- GA优化NN的结构：神经元数和连接状况



1、GA优化NN的权重

- 编码（实数编码）

		1	2	3	4	5	6	7	8
From neuron:									
To neuron:	1	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0
	4	0.9	-0.3	-0.7	0	0	0	0	0
	5	-0.8	0.6	0.3	0	0	0	0	0
	6	0.1	-0.2	0.2	0	0	0	0	0
	7	0.4	0.5	0.8	0	0	0	0	0
	8	0	0	0	-0.6	0.1	-0.2	0.9	0



Chromosome:	0.9	-0.3	-0.7	-0.8	0.6	0.3	0.1	-0.2	0.2	0.4	0.5	0.8	-0.6	0.1	-0.2	0.9

1、GA优化NN的权重

❖定义适应度

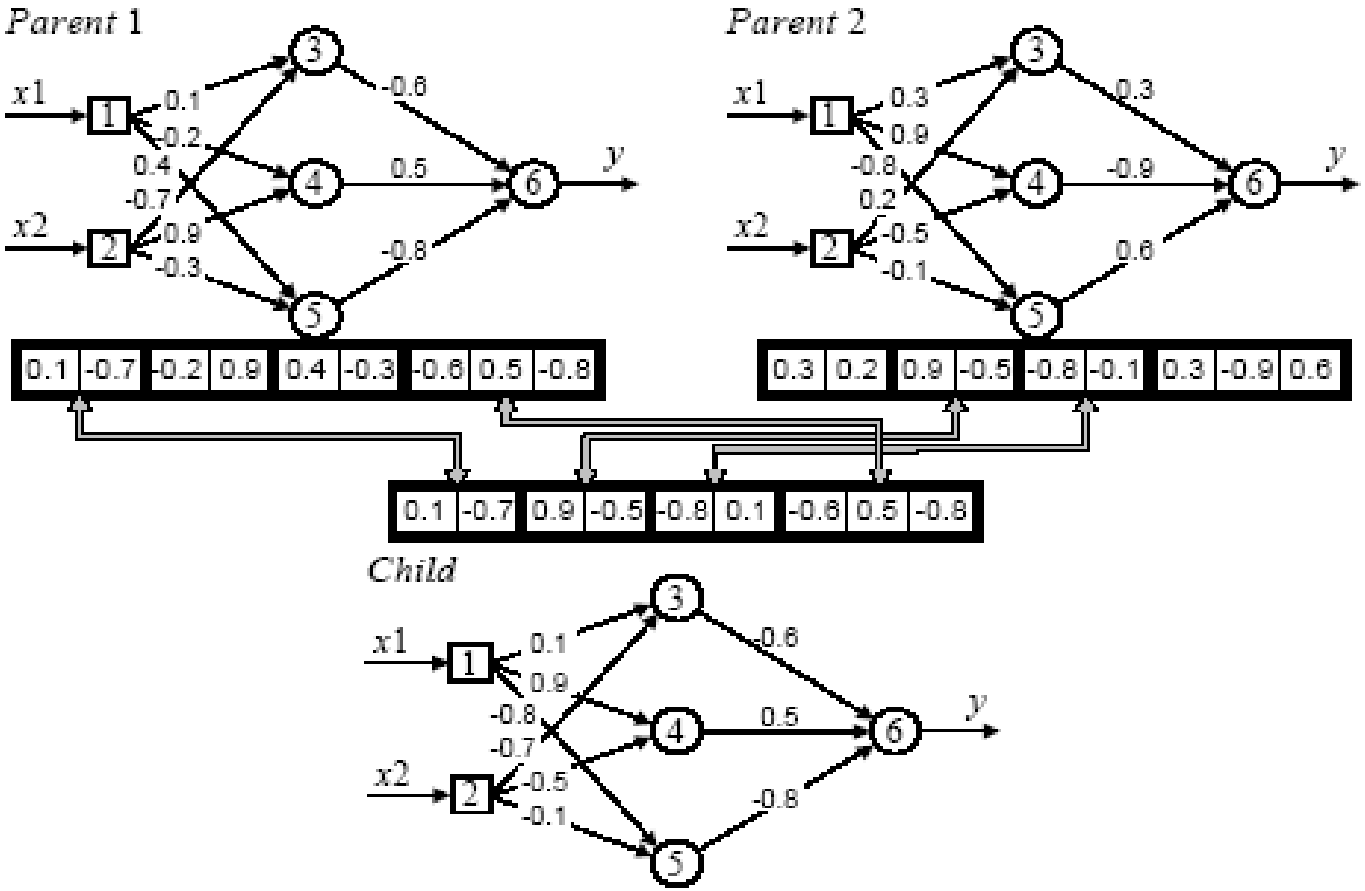
- 直接采用方差和.

$$E_p = \frac{1}{2}\sum_k \left(t_{pk} - o_{pk}\right)^2 \qquad E = \frac{1}{l}\sum_p E_p$$

- 评估过程：对每个解进行样本测试（前向计算），计算其实际输出和期望输出的误差（适应度）。如果有一个个体的适应度满足精度要求，则结束

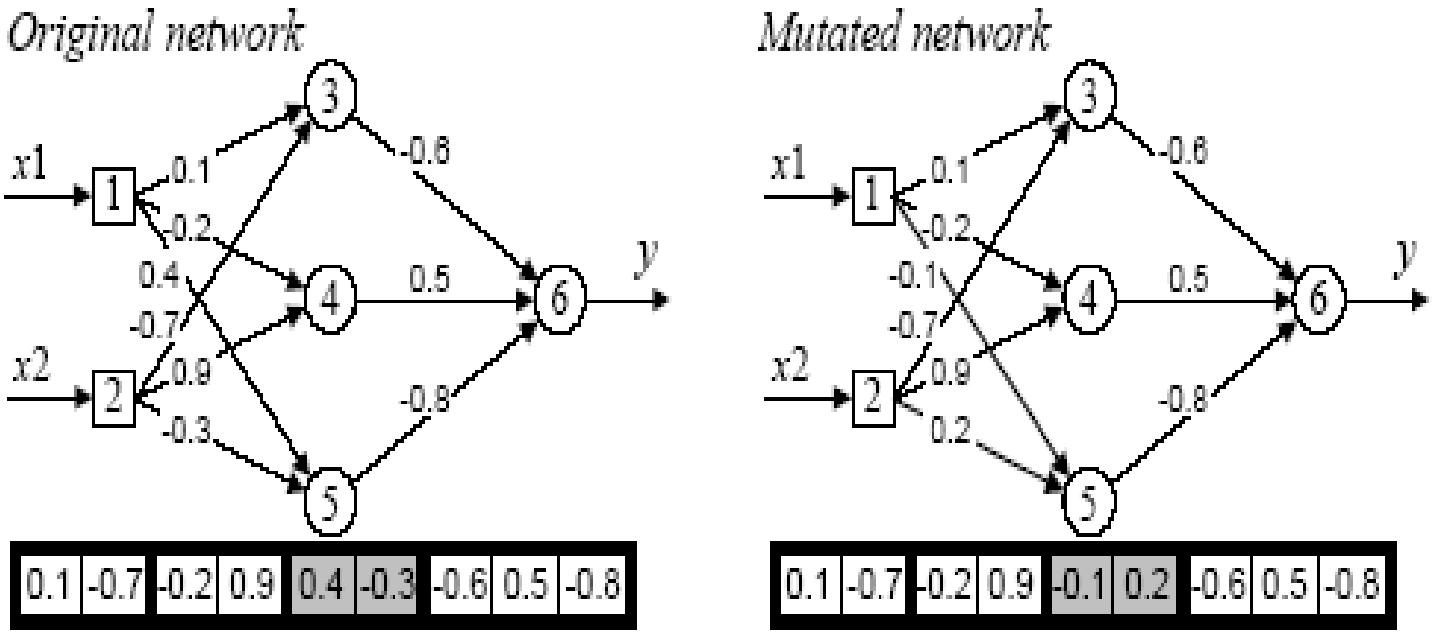
1、GA优化NN的权重

■遗传操作：交叉

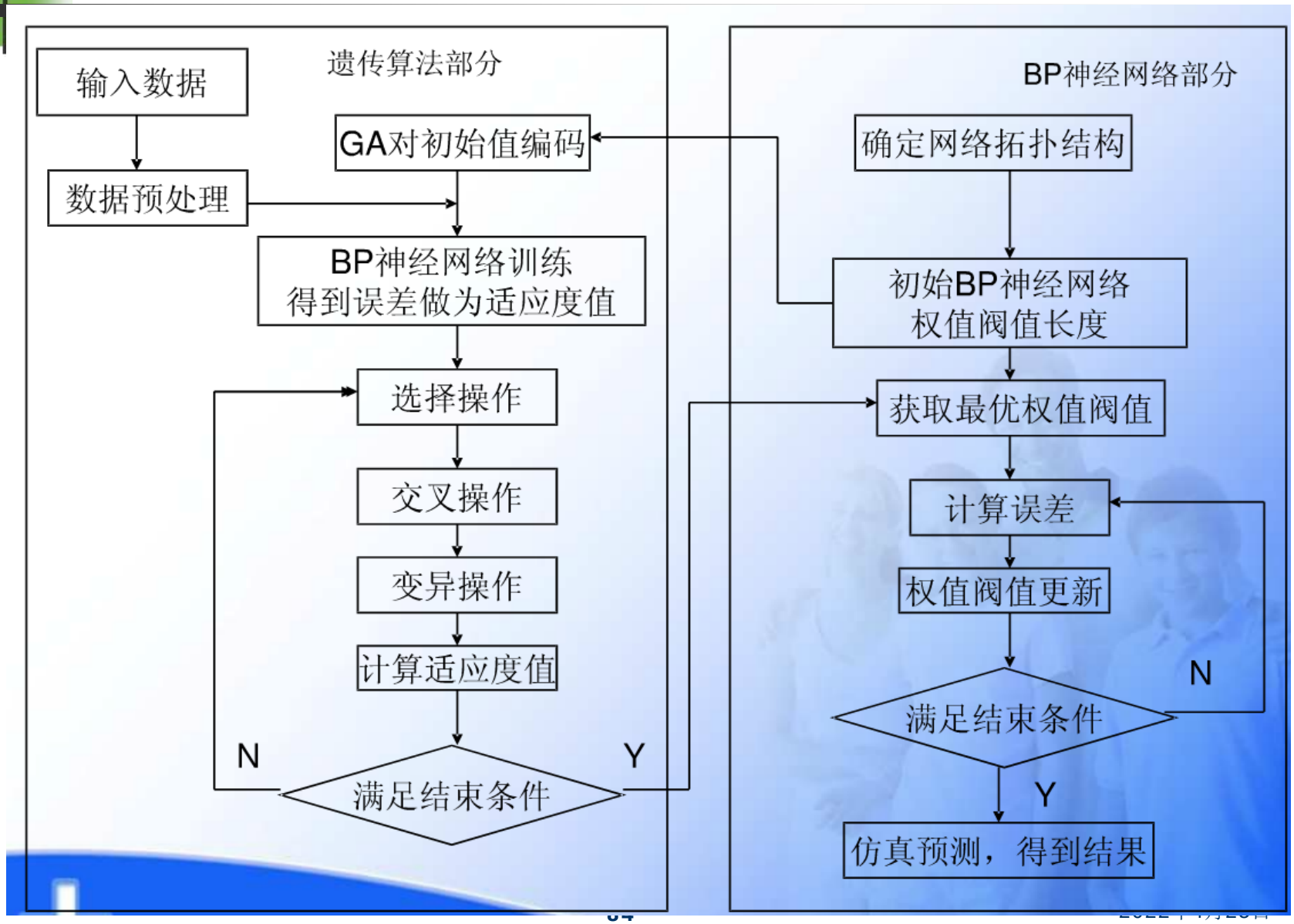


1、GA优化NN的权重

■遗传操作：变异



1、GA优化NN的权重


遗传算法部分
输入数据
数据预处理
GA对初始值编码
BP神经网络训练
得到误差做为适应度值
选择操作
交叉操作
变异操作
计算适应度值
满足结束条件
N
Y
BP神经网络部分
确定网络拓扑结构
初始BP神经网络
权值阀值长度
获取最优权值阀值
计算误差
权值阀值更新
满足结束条件
N
Y
仿真预测，得到结果

1、GA优化NN的权重

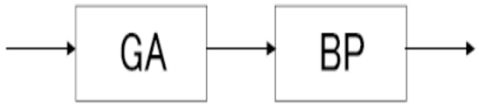
■BP算法和GA算法结合

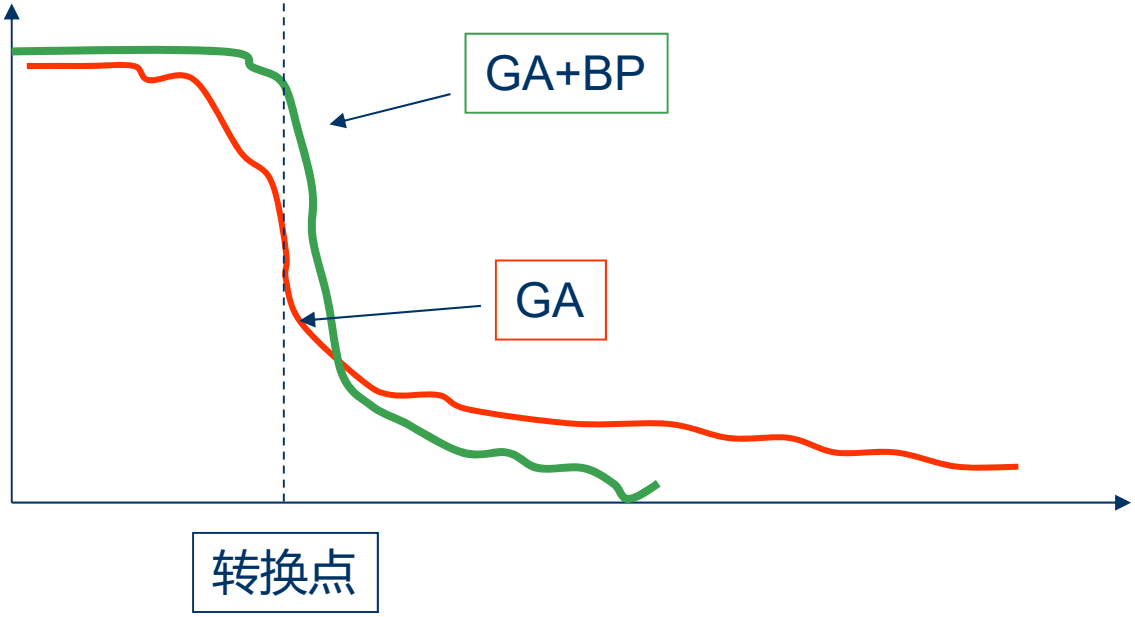
GA擅长全局优化搜索，BP擅长局部优化搜索；两者结合，可提高收敛速度。克服GA过早收敛的问题

1、GA优化NN的权重

■BP算法和GA算法结合：方法一

- 先用GA找出全局最优解的大概位置，然后采用BP算法微调得到全局最优解。





1、GA优化NN的权重

■BP算法和GA算法结合：方法二

交替使用BP和GA

第一步:利用GA找出一个最优解;

第二步：利用BP对该最优解进行微调;

第三步：如果发现微调后的最优解不够理想，则返回第一步；否则停止。

1、GA优化NN的权重

❖ **算法：**

1.随机产生一个具有N个个体的初始群体；

2.计算每个个体的适应值，如果有一个个体的适应度满足精度要求，则结束，否则进行下一步；

3.按适应度比例挑选出父本群体；

4.对父本群体进行杂交、变异操作得到新的群体；

5.找出当前群体中适应度最大的个体best；

6.对best用BP进行一到二次学习，得best’；

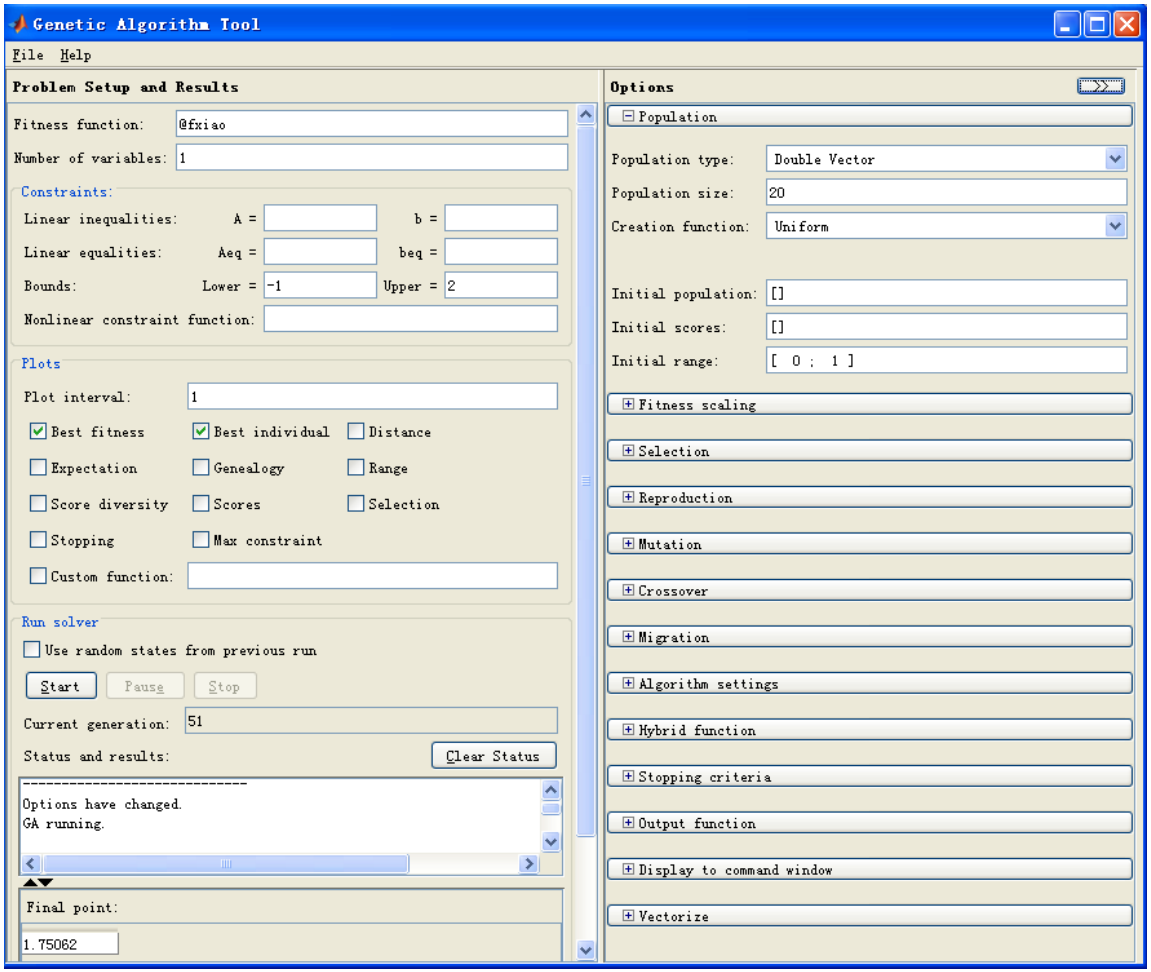
7.用best’替代best，转向2；

6 遗传算法的实现

- Matlab的GA工具箱
- Matlab的GA函数调用
- 根据原理编写属于自己的GA

6 遗传算法的实现：

❖Matlab的GA工具箱



6 遗传算法的实现

Matlab的GA函数调用

- 第一步：编写适应度函数；
- 第二步：对GA参数进行设置；

options = gaoptimset('参数名', 参数值, …, '参数名', 参数值)

例:options = gaoptimset('PopulationSize', 100)

- 第三步：调用GA函数；

[x fval] = ga(@fitnessfun, nvars)

[x fval exitflag output population scores] = ga(@fitnessfcn, nvars)

[x fval] = ga(@fitnessfun, nvars, [],[],[],[],[],[],[],options);

specify any linear equality, linear inequality,
or nonlinear constraints

6 遗传算法的实现

❖Matlab的GA函数调用

[x fval] = ga(@fitnessfun, nvars, [],[],[],[],[],[],[], options);

fitnessfcn — Fitness function

nvars — Number of variables for the problem

Aineq — Matrix for inequality constraints

Bineq — Vector for inequality constraints

Aeq — Matrix for equality constraints

Beq — Vector for equality constraints

LB — Lower bound on x

UB — Upper bound on x

nonlcon — Nonlinear constraint Function

options — Options structure

6 遗传算法的实现

- 根据原理编写属于自己的GA

参考资源

[1]王小平，曹立明. 遗传算法——理论、应用与软件实现. 西安交通大学出版社，**2002.1**

[2]朱福喜，朱三元，伍春香. 人工智能基础教程. 清华大学出版社，**2006.3**

[3]刘金琨. 机器人控制系统的设计与**MATLAB**仿真. 清华大学出版社，**2008.6**

[4] 雷英杰，张善文，李旭武. **MATLAB**遗传算法工具箱及应用. 西安电子科技大学出版社，**2005.4**

[5]求是科技. **MATLAB7.0**从入门到精通. 人民邮电出版社，**2006.3**

[6]**http://en.wikipedia.org/wiki/Genetic_algorithm**

[7]**http://en.wikipedia.org/wiki/Traveling_salesman_problem**

❖课后作业：

编程实现用遗传算法求取以下目标函数的极大值。

$$X \cdot cos2\pi Y + Y \cdot \sin 2\pi \cdot X$$

要求：种群初始随机，范围分别为[-2, 2]、[-2, 2]，染色体长度为20，交叉概率0.7，变异概率为0.01