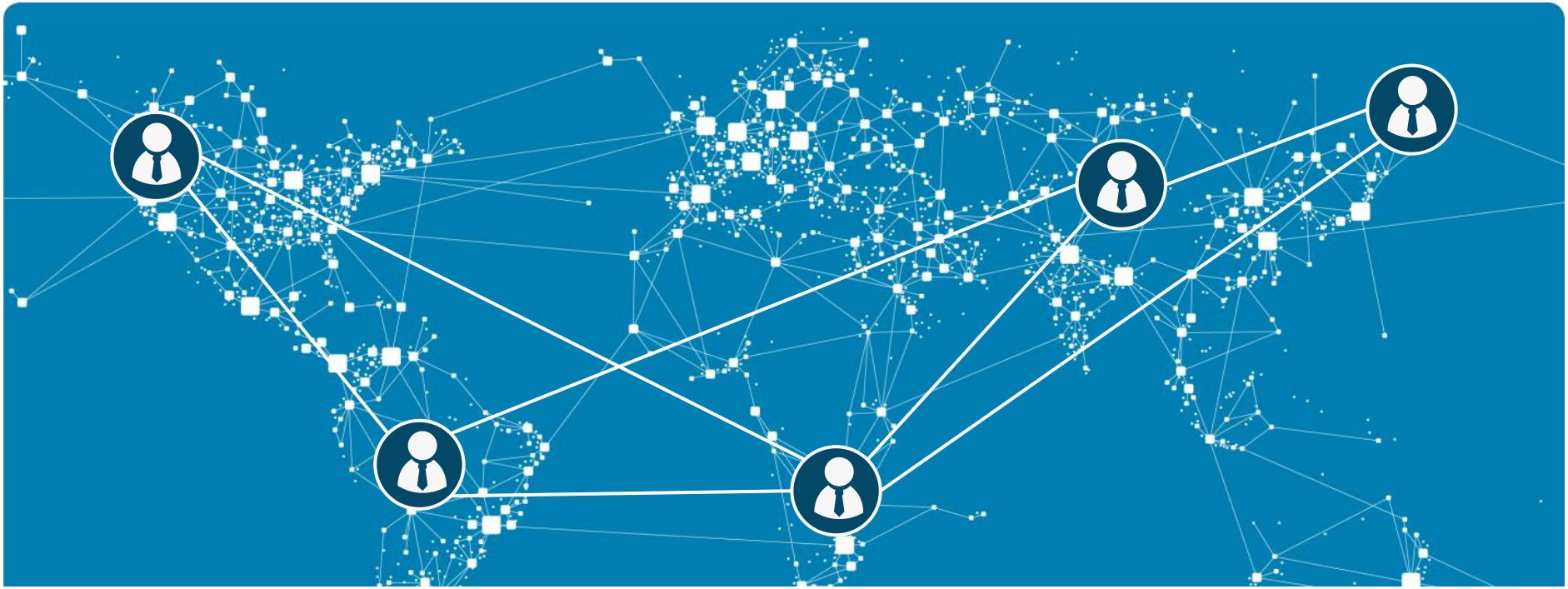


管网优化设计快速入门
第八节-模型参数校核

主讲人：小木
东华大学

课程大纲

- 1.水量预测
- 2.管网建模
- 3.监测点布置
- 4.水泵优化调度
- 5.爆管分析
- 6.管网水质
- 7.管网分区
- 8.模型校核

概述

- 1.遗传算法（GA）
  - 1.1.标准遗传算法（SGA）
  - 1.2.混乱遗传算法（MGA）
  - 1.3.快速混乱遗传算法（FMGA）
  - 1.4.非控制排序遗传算法（NSGA-II）
- 2.模型校核
- 3.总结

1.遗传算法

什么叫遗传算法

函数：$y=ax^2+bx+c$

最优解为（$-2a/b, 4ac-b^2/4a$）

这是二次函数，那么三次函数呢，多次幂的函数呢，非线性函数呢？这些我们没有一个确定的公式来找到它的最优解（最大或最小值），那么我们怎么办？

于是我们就引入了优化算法

优化算法是求最优值的方法，其大致包括三大类，遗传算法，模拟退火法，粒子群算法三种。其余的方法，如布谷鸟算法，都是按照这些方法改造而成的。

什么叫遗传算法

迭代法的基本结构(最小化$f(\mathbf{x})$)

1. 选择一个初始点，设置一个convergence tolerance ε，计数$k=0$
2. 决定搜索方向$\mathbf{d}_k$，使得函数下降。(核心)
3. 决定步长α_k使得$f(\mathbf{x}_k+\alpha\mathbf{d}_k)$对于$\alpha\geq 0$最小化，构建$\mathbf{x}_{k+1}=\mathbf{x}_k+\alpha_k\mathbf{d}_k$ (Dichotomous search, Golden section search)
4. 如果$\|\alpha_k\mathbf{d}_k\|<\varepsilon$，则停止输出解$\mathbf{x}_{k+1}$；否则继续重复迭代。

什么叫遗传算法

遗传算法（Genetic Algorithm）是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法。

GA是借鉴生物界的进化规律（适者生存，优胜劣汰遗传机制）演化而来的随机化搜索方法。它是由美国的J.Holland教授1975年首先提出，其主要特点是直接对结构对象进行操作，不存在求导和函数连续性的限定；具有内在的隐并行性和更好的全局寻优能力；采用概率化的寻优方法，能自动获取和指导优化的搜索空间，自适应地调整搜索方向，不需要确定的规则。遗传算法的这些性质，已被人们广泛地应用于组合优化、机器学习、信号处理、自适应控制和人工生命等领域。它是现代有关智能计算中的关键技术。

什么叫遗传算法

目前为止，遗传算法的种类十分的多，应用最广泛的是美国的J.Holland教授的标准遗传算法（SGA）、其它的如混乱遗传算法、非控制排序遗传算法等等。

SGA属于简单遗传算法，是遗传算法的最初级算法，其对于优化算法的发展具有里程碑的意义。

SGA由编、解码、个体适应度评估、遗传运算组成。遗传运算包括复制、交叉、变异、倒位等方法。

1.1 SGA

SGA

SGA的几个概念

1.种群：所有编码后的染色体集合；个体：一整条染色体

2.复制：被选中繁殖下一代的个体，就叫做复制体

3.编码：设自变量为X（X属于（L,U）），使用长度为k的二进制编码

000000000000000000

000000000000000001

000000000000000010

000000000000000011

SGA

SGA的几个概念

3.编码：长度k的确定方式：假如我这个数想要精确到小数点后面a位

$$2^{m_i-1} < (U-L) \times 10^a \leqslant 2^{m_i} - 1$$

举个例子：设-3.0≤x≤12.1 保留小数点后四位

那么 (U-L)*10^a=(12.1-(-3.0))*10^4=15100

131702=2^17≤15100≤2^18-1=262143

所以，个体的染色体长度为18

SGA的几个概念

4.解码：

解码是将二进制编码变成十进制的实际数据，其公式为：

$$x = L + \left(\sum_{i=1}^{k} b_i 2^{i-1} \right) \frac{U - L}{2^k - 1}$$

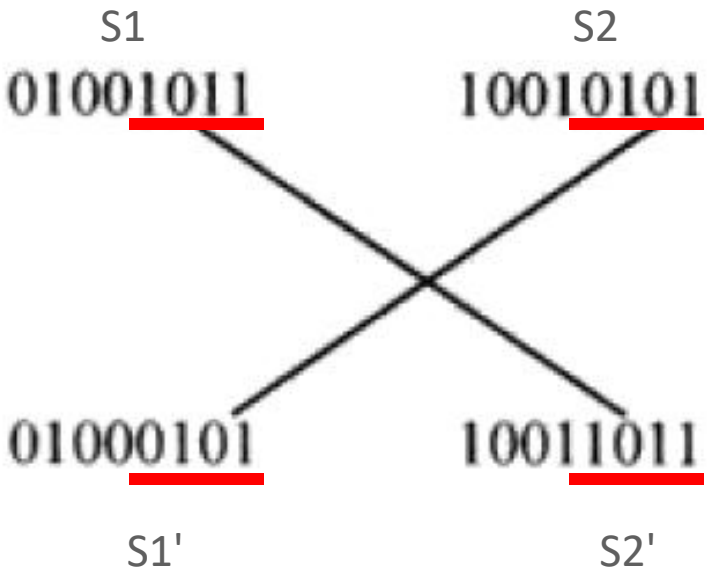
举个例子：x取值范围为[2,4]，若保留1位小数点，用5位2进制数来表示如取值为10101，其十进制值为：

$$\sum_{i=1}^{5} b_i 2^{i-1} = 1 \times 2^0 + 0 \times 2 + 1 \times 2^2 + 0 \times 2^3 + 1 \times 2^4 = 21$$

带入上述公式：$x = 2 + 21 \times \frac{4 - 2}{2^5 - 1} = 3.4$

SGA的几个概念

5.交配：交配运算是使用单点或多点进行交叉的算子，首先用随机数产生一个交配点位置，然后两个个体在交配点位置互换基因码。例如：



SGA的几个概念

6.突变：实行基因码0，1小概率的翻转.若达到突变概率，则1变为0，0变为1。

1 0 1 0 1

达到突变概率，且
第四位进行突变

1 0 1 1 1

SGA

SGA的几个概念

7.倒位：倒位在遗传算法中用的不是很多，是将染色体某个区段的正常排序顺序颠倒180度。

如：11001，颠倒下划线位置后为： 11100

8.适应度函数：y=f(x)，需要求最优值的函数

如想要求最优值y_{opt}，则$y=ax^2+bx+c$为适应度函数，任意取一个x1，求出的值y1叫做适应度。

SGA举例说明：卓金武老师的例子

卓金

【例

表 4-1　染色体编码

自变量	二进制	十进制	实际值
x_1	000001010100101001	5 417	−2.687 969
x_2	101111011111110	24 318	5.361 653

首先要进行编码工作，即将变量转换成二进制数串。数串的长度取决于所要求的精度。例如，变量 x 的区间是(L,U)，要求的精度是小数点后 4 位，也就意味着每个变量应该被分成至少$(L,U)\times 10^4$ 个部分。对一个变量的二进制数串位数用以下公式计算：

$$2^{m_i-1} < (U-L)\times 10^4 \leqslant 2^{m_i}-1$$

本例精度要求保留小数点后 4 位，则目标函数的两个自变量 x_1 及 x_2 所构成的染色体数串可以表示如下：

$$\begin{cases} -3.0 \leqslant x_1 \leqslant 12.1 \\ [12.1-(-3.0)]\times 10^4 = 151\,000 \\ 2^{m_1-1} < 151\,000 \leqslant 2^{m_1}-1 \\ m_1 = 18 \\ 4.1 \leqslant x_2 \leqslant 5.8 \\ (5.8-4.1)\times 10^4 = 17\,000 \\ 2^{m_2-1} < 17\,000 \leqslant 2^{m_2}-1 \\ m_2 = 15 \\ m = m_1 + m_2 = 18+15 = 33 \end{cases}$$

本例中任一染色体数串都是 33 位，即 000001010100101001101111011111110。

以上编码前 18 位表示 x_1，后 15 位表示 x_2，如表 4-1 所列。

SGA

【例 4－1】 求 $\max f(x_1,x_2)=21.5+x_1\sin(4\pi x_1)+x_2\sin(20\pi x_2)$

$$\text{s.t.}\begin{cases}-3.0\leqslant x_1\leqslant 12.1\\ 4.1\leqslant x_2\leqslant 5.8\end{cases}$$

卓金武老师的例子

则二进制转化成十进制为

$$x_1=-3.0+5\,417\times\frac{12.1-(-3.0)}{2^{18}-1}=-2.687\,969$$

$$x_2=4.1+24\,318\times\frac{5.8-4.1}{2^{15}-1}=5.361\,653$$

假设初始种群中有 10 个个体，其染色体可随机生成如下：

$U_1=[000001010100101001\ 101111011111110]$

$U_2=[001110101110011000\ 000010101001000]$

$U_3=[111000111000001000\ 010101001000110]$

$U_4=[100110110100101101\ 000000010111001]$

$U_5=[000010111101100010\ 001110001101000]$

$U_6=[111110101011011000\ 000010110011001]$

$U_7=[110100010111110001\ 001100111011101]$

$U_8=[001011010100001100\ 010110011001100]$

$U_9=[111110001011101100\ 011101000111101]$

$U_{10}=[111101001110101010\ 000010101101010]$

相对应的十进制的实际值$[x_1,x_2]$为

$U_1=[-2.687\,969,5.361\,653]$，　$U_2=[0.474\,101,4.170\,144]$

$U_3=[10.419\,457,4.661\,461]$，　$U_4=[6.159\,951,4.109\,598]$

$U_5=[-2.301\,286,4.477\,282]$，　$U_6=[11.788\,084,4.174\,346]$

$U_7=[9.342\,067,5.121\,702]$，　$U_8=[-0.330\,256,4.694\,977]$

$U_9=[11.671\,267,4.873\,501]$，　$U_{10}=[11.446\,273,4.171\,908]$

SGA

【例 4－1】 求 $\max f(x_1,x_2)=21.5+x_1\sin(4\pi x_1)+x_2\sin(20\pi x_2)$

$$\text{s.t.}\begin{cases}-3.0\leqslant x_1\leqslant 12.1\\ 4.1\leqslant x_2\leqslant 5.8\end{cases}$$

卓金武老师的例子

(2) 评价个体适应度

对一个染色体数串的适应度的评价由下列三个步骤组成。

① 将染色体串进行反编码(解码),转换成真实值,即

$$x^k=(x_1^k,x_2^k),\qquad k=1,2,3,\cdots$$

② 评价目标函数 $f(x^k)$。

③ 将目标函数值转为适应度。对于极大值问题,适应度可作为目标函数值:

$$\text{eval}(U_k)=f(x^k),\qquad k=1,2,3,\cdots$$

在遗传算法中,评价函数扮演自然进化中环境的角色,它通过染色体的适应度对其进行评价。上述染色体的适应度值如下:

$$\text{eval}(U_1)=f(-2.687969,5.361653)=19.805119$$
$$\text{eval}(U_2)=f(0.474101,4.170144)=17.370896$$
$$\text{eval}(U_3)=f(10.419457,4.661461)=9.590546$$
$$\text{eval}(U_4)=f(6.159951,4.109598)=29.406122$$
$$\text{eval}(U_5)=f(-2.301286,4.477282)=15.686091$$
$$\text{eval}(U_6)=f(11.788084,4.174346)=11.900541$$
$$\text{eval}(U_7)=f(9.342067,5.121702)=17.958717$$
$$\text{eval}(U_8)=f(-0.330256,4.694977)=19.763190$$
$$\text{eval}(U_9)=f(11.671267,4.873501)=26.401669$$
$$\text{eval}(U_{10})=f(11.446273,4.171908)=10.252480$$

SGA

【例 4－1】 求 $\max f(x_1,x_2)=21.5+x_1\sin(4\pi x_1)+x_2\sin(20\pi x_2)$

$$\text{s.t.}\begin{cases}-3.0\leqslant x_1\leqslant 12.1\\ 4.1\leqslant x_2\leqslant 5.8\end{cases}$$

卓金武老师的例子

依照染色体的适应度值进行新种群的复制，步骤如下：

① 计算染色体 U_k 的适应度值

$$\text{eval}(U_k)=f(x^k),\qquad k=1,2,\cdots$$

② 计算种群的适应度值总和

$$F=\sum_{k=1}^{\text{pop\_size}}\text{eval}(U_k)$$

③ 计算每个染色体被复制的概率

$$P_k=\frac{\text{eval}(U_k)}{F}$$

④ 计算每个染色体被复制的累积概率

$$Q_k=\sum_{j=1}^{k}P_k$$

表 4－2 种群每条染色体的适应度、被复制概率和被复制的累积概率

染色体	适应度值	P_k	Q_k
U_1	19.805 119	0.111 180	0.111 180
U_2	17.370 896	0.097 515	0.208 695
U_3	9.590 546	0.053 839	0.262 534
U_4	29.406 122	0.165 077	0.427 611
U_5	15.686 091	0.088 057	0.515 668
U_6	11.900 541	0.066 806	0.582 475
U_7	17.958 717	0.100 815	0.683 290
U_8	19.763 190	0.110 945	0.794 234
U_9	26.401 669	0.148 211	0.942 446
U_{10}	10.252 480	0.057 554	1.000 000

SGA

【例 4－1】 求 $\max f(x_1,x_2)=21.5+x_1\sin(4\pi x_1)+x_2\sin(20\pi x_2)$

$$\text{s.t.}\begin{cases}-3.0\leqslant x_1\leqslant 12.1\\ 4.1\leqslant x_2\leqslant 5.8\end{cases}$$

卓金武老师的例子

（3）新种群复制

依照轮盘选择法，转动轮盘 10 次（种群中有 10 条染色体），每次选择一个作为新种群的染色体。假设 10 次中产生的 0～1 随机数序列如下：

0.301 431　0.322 062　0.766 503　0.881 893　0.350 871

0.538 392　0.177 618　0.343 242　0.032 685　0.197 577

第 1 个随机数为 0.301 431，大于 Q_3 小于 Q_4，所以 U_4 被选中；

第 2 个随机数为 0.322 062，大于 Q_3 小于 Q_4，所以 U_4 再次被选中；

第 3 个随机数为 0.766 503，大于 Q_7 小于 Q_8，所以 U_8 被选中；

……

第 10 个随机数为 0.197 577，大于 Q_1 小于 Q_2，所以 U_2 被选中；

依照轮盘选择法，新种群的染色体组成如下：

$U_1=[1001101101001011010000000010111001](U_4)$　$U_6=[1101000101111100010011001110111101](U_7)$

$U_2=[1001101101001011010000000010111001](U_4)$　$U_7=[0011101011100110000001010100100 0](U_2)$

$U_3=[0010110101000011000101100110011 00](U_8)$　$U_8=[1001101101001011010000000010111001](U_4)$

$U_4=[1111100010111011000111010001111 01](U_9)$　$U_9=[0000010101001010011011110111111 10](U_1)$

$U_5=[1001101101001011010000000010111001](U_4)$　$U_{10}=[0011101011100110000001010100100 0](U_2)$

这种轮盘选择法的机理是：染色体的适应度大意味着$[Q_k,Q_{k+1}]$区间跨度就大，随机数发生器产生的均匀随机数就会有更大的概率落在较大长度的$[Q_k,Q_{k+1}]$区间里，这样具有较大 P_k 值的染色体自然更有机会复制到下一代。

【例 4－1】 求 $\max f(x_1,x_2)=21.5+x_1\sin(4\pi x_1)+x_2\sin(20\pi x_2)$

$$\text{s.t.}\begin{cases}-3.0\leqslant x_1\leqslant 12.1\\ 4.1\leqslant x_2\leqslant 5.8\end{cases}$$

(4) 新种群交配

① 交配染色体数量的确定

交配染色体的数量等于染色体总量乘以交配概率。这里假设交配概率 P_c 为 0.25，染色体总量为 10 条，所以参加交配的染色体数量为[2.5]条。符号[ ]表示取整，这里取整数 2，即交配的染色体数目为 2 条。

② 交配染色体对象的确立

用计算机产生[0,1]区间的 10 个随机数如下：

0.625 721	0.266 823	0.288 644	0.295 114	0.163 274
0.567 461	0.085 940	0.392 865	0.770 714	0.548 656

假定其分别对应 $U_1\sim U_{10}$ 这 10 个个体，则其中低于交配概率 0.25 的 U_5 和 U_7 参加交配。这样操作的原因是：交配概率越低，低于交配概率以下的随机数的数量就越少，所以参加交配的染色体数量与交配概率可能会成正比。

③ 在交配池发生交配

染色体 U_5 和 U_7 被选中作为交配的父辈，交配点的选择以随机数产生。交配的种类有单点交配和多点交配，这里取单点交配。计算机随机生成一个介于 0～32 的整数(因为整个染色体数串的长度为 33)。假设所产生的整数为 1，那么两个染色体自 1 位置(即二进制串的第二位)开始分割，在染色体 1 位置右端部分进行交换而生成新的子辈染色体，即

$$U_5=[1\ \underline{0011\ 0110\ 1001\ 0110\ 1000\ 0000\ 1011\ 1001}]$$

$$U_7=[0\ \underline{0111\ 0101\ 1100\ 1100\ 0000\ 0101\ 0100\ 1000}]$$

$$\Downarrow$$

$$U_5^*=[1\ \underline{0111\ 0101\ 1100\ 1100\ 0000\ 0101\ 0100\ 1000}]$$

$$U_7^*=[0\ \underline{0011\ 0110\ 1001\ 0110\ 1000\ 0000\ 1011\ 1001}]$$

【例 4－1】 求 $\max f(x_1,x_2)=21.5+x_1\sin(4\pi x_1)+x_2\sin(20\pi x_2)$

$$\text{s.t.}\begin{cases}-3.0\leqslant x_1\leqslant 12.1\\ 4.1\leqslant x_2\leqslant 5.8\end{cases}$$

(5) 基因突变

依照突变运算规则并假设突变几率 P_m 为 0.01，亦即种群内所有基因都有 0.01 的概率进行突变。在本例中共有 33×10＝330 个基因，即希望每一代中有 3.3 个突变基因，每个基因的突变几率是均等的。因此，将产生 330 个介于 0～1 之间的随机数(需编号)，然后将该随机数小于 0.01 者选出，并将其对应的基因值加以翻转，假设 330 次中产生 0～1 之间的随机数，其值小于 0.01 者如表 4－3 所列。

表 4－3 基因突变位置

基因位置	染色体位置	基因位数	随机数
105÷33＝4…6	4	6	0.009 857
164÷33＝5…32	5	32	0.003 113
199÷33＝7…1	7	1	0.000 946
329÷33＝10…32	10	32	0.001 282

表 4－3 第 1 列显示的是具体在哪些染色体以及在染色体的什么位置进行了突变。例如第 1 行“105÷33＝4…6”表示的是在第 4 条染色体第 6 个基因上发生了突变，因为第 4 条染色体第 6 个基因对应的基因编号是 33×(4－1)＋6＝105。

SGA

【例 4－1】 求 $\max f(x_1,x_2)=21.5+x_1\sin(4\pi x_1)+x_2\sin(20\pi x_2)$

$$\text{s.t.}\begin{cases}-3.0\leqslant x_1\leqslant 12.1\\ 4.1\leqslant x_2\leqslant 5.8\end{cases}$$

在突变后，最终新种群的染色体组成如下：

$U_1=[100110110100101101\ 000000010111001]$

$U_2=[100110110100101101\ 000000010111001]$

$U_3=[001011010100001100\ 010110011001100]$

$U_4=[111111001011101100\ 011101000111101]$

$U_5=[101110101110011000\ 000010101001010]$

$U_6=[110100010111110001\ 001100111011101]$

$U_7=[100110110100101101\ 000000010111001]$

$U_8=[100110110100101101\ 000000010111001]$

$U_9=[000001010100101001\ 101111011111110]$

$U_{10}=[001110101110011000\ 000010101001010]$

新一代的相对应实际值$[x_1,x_2]$和适应度值如下：

$\text{eval}(U_1)=f(6.159\,951,4.109\,598)=29.406\,122$

$\text{eval}(U_2)=f(6.159\,951,4.109\,598)=29.406\,122$

$U_{\text{best}}=[111110000000111000\ 111101001010110]$

相应实际值$[x_1,x_2]=[11.631\,407,5.724\,824]$，适应度值 $\text{eval}(U)=f(11.631\,407,5.724\,824)=38.818\,2$

$\text{eval}(U_6)=f(9.342\,067,5.117\,020)=17.958\,717$

$\text{eval}(U_7)=f(6.159\,951,4.109\,598)=29.406\,122$

$\text{eval}(U_8)=f(6.159\,951,4.109\,598)=29.406\,122$

$\text{eval}(U_9)=f(-2.687\,969,5.361\,653)=19.805\,119$

$\text{eval}(U_{10})=f(0.474\,101,4.170\,248)=17.370\,896$

至此，已完成遗传算法的第一代流程，一次迭代，至451次得到对应最大目标函数值的染色体

【例 4－1】 求 $\max f(x_1,x_2)=21.5+x_1\sin(4\pi x_1)+x_2\sin(20\pi x_2)$

$$\text{s.t.}\begin{cases}-3.0\leqslant x_1\leqslant 12.1\\ 4.1\leqslant x_2\leqslant 5.8\end{cases}$$

SGA

其余的也按照这个例子计算可以了，这就是最简单的SGA方法。

但是SGA方法也有一些缺点，

1.2 MGA

MGA

MGA

J.Holland在提出了SGA之后，Goldberg等在1989，改进了SGA，提出了混乱遗传算法（messy Genetic algorithm），该方法解决了更大型的非线性问题。

相比SGA，其优点为：

染色体位置是松散的，不是非得从头到尾的排序。

同时染色体的长度不是固定不变的，而是不断变长的。

最优值作为模版，并不断改进。

也就是说，MGA能更好地找到最优解。像GA到最后，所有的染色体都不变，然后陷入局部最优不动了，MGA出现这样的情况的概率大大降低了。

MGA

MGA

1.messy codes

MGA之所以叫做MGA那就是因为染色体的长度和顺序是变化的，比如我们SGA里面代码，每个位置有一个基因，但是在MGA中那就不是了，很可能某个位置没有基因，也可能某个位置有多个基因。

比如 正常的代码是{(1,0) (2,1) (3,1)}

在MGA中有三种情况，一种是正常的代码

打乱顺序的正常代码：{(2,1) (3,1) (1,0) }

underspecified: {(2,1) (3,1) } 某个位置少元素

overspecified: {(2,1) (3,1) (2,0) (1,1)} 某个位置多元素

MGA

MGA

2.templates

MGA中因为under-overspecified两者都要去计算适应度函数，那么少东西或者多东西怎么办呢。

（1）多东西：

从左到右往下走，选择第一个发现的基因作为变量，舍弃后面的，如：{(2,1) (3,1) (2,0) (1,1)}，第二个位置有两个元素，舍弃（2，0），保留第一个变量（2，1）

即：{(2,1) (3,1) (1,1)}

（2）少东西：

在生成MGA的染色体之前，先随机生成一个正常的template，如{(1,0) (2,1) (3,1)}，若underspecified是{(2,1) (3,1) }，用template中的(1,0)弥补缺少的变量。

即：{(2,1) (3,1) (1,0)}

MGA

3.Messy 控制

（1）门槛控制

sga选择基因进行复制时，随机生成一个数，然后比较大小选择复制个体，但是在MGA中就不能这样了，因为SGA中染色体位置长度不确定，如果选择相似度不足的来对比失去了意义，需要选两个差不多的比较，然后保留好的那个复制，比较两个染色体差不多的方法就是用一个门槛公式：

$$\theta = \left[\frac{l_1 l_2}{l} \right]$$

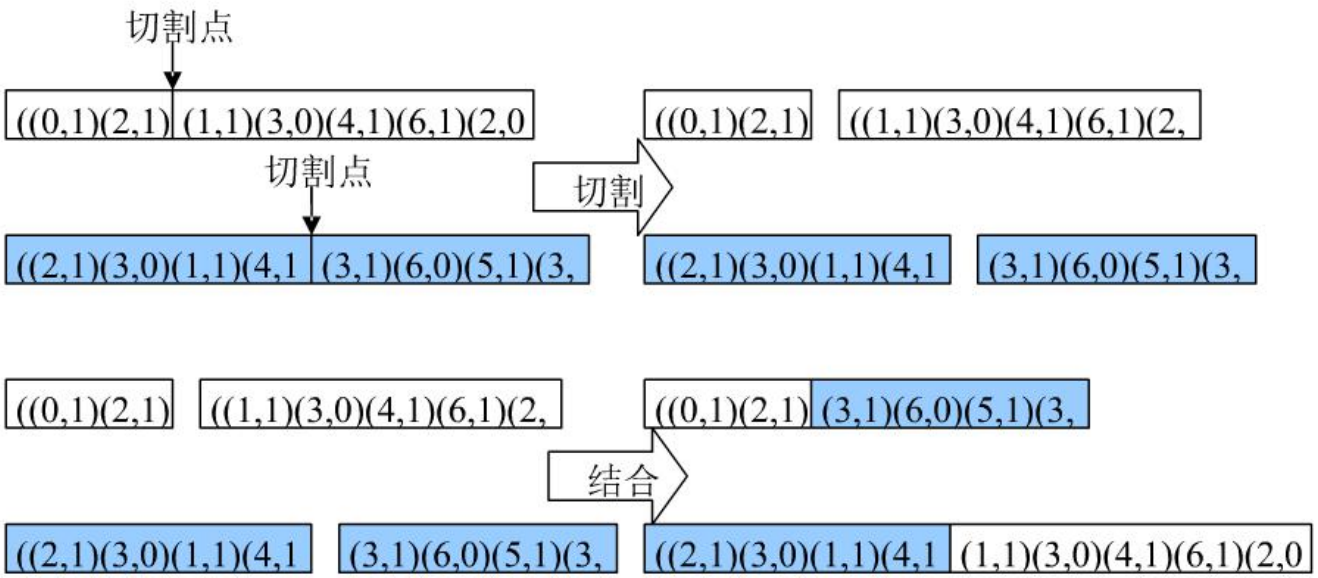
如果相似度大于θ，就可以比较。比如：{(0,1) (3,1) (5,0) (2,1)}以及{(3,0) (5,1) (0,0) (6,1)}它有三个相同位置，0、3、5.所以相似度是3，θ=[4*4/10]=1≤3，所以可以比较。

MGA

3.Messy 控制

（2）交配控制

根据交配概率P求出需要交配的染色体数量，然后随机选择两个染色体进行交配，交配方式是随机生成两个切割点，然后切割，之后相互结合。



MGA

4.Outer loop begins （外循环，舒诗湖论文译）

在Outer loop begins 中，首先确定染色体中包含基因数k值，然后在确定一个正常染色体template，长度为正常情况下的*l*。

5.Inner loop

（1）初始化阶段：随机生成基因数k的染色体，且计算适应度

（2）Primordial 阶段：门槛选择，然后复制染色体

（3）Juxtapositional 阶段：门槛选择，交配，变异，计算适应度

（4）Inner loop end 阶段：把最优适应度解当作模版

6. Outer loop ends

结束条件一般有两种，一种是k值与l值相等，或者达到最优值

MGA

MGA的 计算步骤如下所示：level即为k值，t值是自己设定的，注意，上下两个t不是一回事儿！

```
/* Initialization */
level = 1; template = random_string; // Random initialization of the template
While(mga_termination_criterion == TRUE) )
{
  t = 0; // Set the generation number to zero.
  Initialize(Pop(t), level); // Initialize the population at random
  Evaluate(Pop(t), template); // Evaluate the fitness values
  Repeat // Enter primordial phase
  {
    Selection(Pop(t)); // Select better strings
    Evaluate(Pop(t)); // Evaluate fitness
    t = t + 1; // Increment generation counter
  }
  Until (primordial_termination_criterion == TRUE)

  t = 0;
  Repeat // Enter juxtapositional phase
  {
    Selection(Pop(t));     Cut_and_Splice(Pop(t)); // Apply cut and splice operator
    Mutation(Pop(t)); // Apply mutation
    Evaluate(Pop(t), template);
    t = t + 1;
  }
  Until ( juxtapositional_termination_criterion == TRUE)

  template = optimal_string(Pop(t), level); // template remains locally optimal
  level = level + 1;
}
```

1.3 FMGA

FMGA

FMGA是MGA的一个修改的版本，中文名字叫做快速混乱遗传算法，相比MGA，其在初始化染色体时利用的方法是probabilistically complete initialization，生成的长度是l'，是一个大于k的数，并不是MGA的k。之后依靠了一个叫做 building-block filter的一种方法，将k阶以上的染色体进行裁剪。

FMGA

1.probabilistically complete initialization

MGA中，若初始化长度为k的染色体，那么其共有2^k*Cl_k种组合，且我们随机生成之时，都是在这些组合里面选择，这就有一点问题，初始化的染色体是什么已经定下来了（就是在这些组合中选择）。

在FMGA中，我们初始化长度变为l'=l-k，这样我们可以把生成长度为k的染色体包括进去，然后在生成几个额外的基因。那么里面有k情况下生成的基因的概率为：

$$p(\ell', k, \ell) = \frac{\binom{\ell - k}{\ell' - k}}{\binom{\ell}{\ell'}}.$$

把概率反过来，则表示几个染色体中才能包含有k情况下生成的基因。

例如染色体中拥有某k基因的概率为P=3/10，反过来就是大概3.33个染色体中会出现一次拥有某k基因。

FMGA

1.probabilistically complete initialization

把概率的倒数用Ng表示：

$$n_g = \binom{\ell}{\ell'} / \binom{\ell-k}{\ell'-k}$$

我们按照Ng的序列来生成长度为l染色体，那么平均Ng个染色体将会有一个k基因。为了是的其初始化的样本随机性增加，在列举Ng序列之时，我们引入population-size方程，以实现我们初始化的种群数不同，序列也不同。该方程如下所示：

$$n_a = 2c(\alpha)\beta^2(m-1)2^k,$$

用Ng与Na做乘法，即可得出序列的总数：

$$\begin{aligned} n &= n_g n_a \\ &= \frac{\binom{\ell}{\ell'}}{\binom{\ell-k}{\ell'-k}} 2c\beta^2(m-1)2^k. \end{aligned}$$

FMGA

$$
\begin{aligned}
n &= n_g n_a \\
&= \frac{\binom{\ell}{\ell'}}{\binom{\ell-k}{\ell'-k}} 2c\beta^2(m-1)2^k
\end{aligned}
$$

其中，l'为染色体基因字串长度，一般采用$l'=l-k$；$c(\alpha)$为常态分布对应α尾端几率平方值；β 为基因块中的最佳适应度值与次佳适应度值的比值；m 为子基因块数量，一般采用问题长度与基因块长度的比值。

一般令c(a)=3，直接作为一个常数即可。

FMGA

2. building-block filter

快速混乱遗传算法的原生阶段中，由于初始随机产生的母体的染色体长度为 l'，而不是基因快长度 k，因此需要通过逐步且随机地进行基因剔除，以将母体中所有染色体字串均缩减为合理长度的基因块。此外，为保留具有较佳适应度值的染色体，在剔除的过程中，每次剔除基因字串后均需再度进行门槛选择。其中，每次剔除基因字串的长度可由剔除比率 ρ 来决定，且 ρ 小于 1。ρ 值的决定必须依据问题不同而进行测试改变，而门槛选择的进行次数则可依式（3-8）来决定

$$\gamma = (\lambda^{(i-1)} / \lambda^{(i)})^k = \rho_i^{-k}$$

$$\ell' / \rho^{t_r} = \zeta k \qquad (3\text{-}8)$$

其中，γ 为进行门槛选择的次数，λ_{i-1} 为第 i-1 次剔除基因后的染色体长度，λ_i 为第 i 次剔除基因后的染色体长度，k 为基因块长度，ρ 为基因字串剔除率。

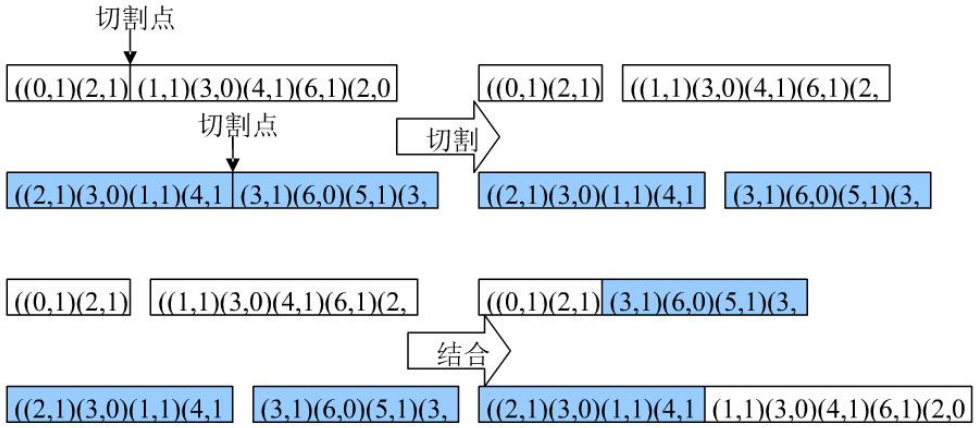
FMGA

2. building-block filter

一般来说规定一次删除一条基因，直到数量减小到k位置，这样的话，循环的次数仅仅为l'-k。

3.交配控制

其与MGA一样，增加的是这样两个基因切割后有概率不结合，切割后有可能生成2个，有可能3个，有可能4个



FMGA

4. 门槛控制

在MGA的基础上，加入了标准偏差与正态分布概率，公式如下：

$$\theta = \lceil \frac{\ell_1 \ell_2}{\ell} + c'(\alpha')\sigma \rceil,$$

$$\sigma^2 = \frac{\ell_1(\ell - \ell_1)\ell_2(\ell - \ell_2)}{\ell^2(\ell - 1)}.$$

FMGA的 计算步骤如下所示：level即为k值，t值是自己设定的，注意，上下两个t不是一回事儿！

```
/* Initialization */
level = 1;
template = random_string; // Random initialization of the template
While(fmga_termination_criterion ≠ TRUE) )
{
  Repeat // Iterative application of fmGA within each level
  t = 0; // Set the generation number to zero.
  Probabilistic_Initialize(Pop(t), level); // Initialize the population
  Evaluate(Pop(t), template); // Evaluate the fitness values
  Repeat // Enter primordial phase
  {
    episode = 0;
    Repeat
    {
      Thresholding_Selection(Pop(t)); // Select better strings
      episode = episode + 1;
    }
    Until (episode < episode_max(t))
    GeneDeletetion(Pop(t)); // Delete Genes
    Evaluate(Pop(t)); // Evaluate fitness
    t = t + 1; // Increment generation counter
  }
  Until (primordial_termination_criterion == TRUE)

  t = 0;
  Repeat // Enter juxtapositional phase
  {
    Selection(Pop(t));
    Cut_and_Splice(Pop(t)); // Apply cut and splice operator
    Mutation(Pop(t)); // Apply mutation
    Evaluate(Pop(t), template);
    t = t + 1;
  }
  Until ( juxtapositional_termination_criterion == TRUE)

  template = optimal_string(Pop(t), level); // template remains locally optimal
 }
  level = level + 1;
}
```

1.4 NSGA-II

NSGA-II

NSGA-II

NSGA-II中文名字叫做非控制排序遗传算法2，它是求解多目标函数最优值的一种算法，是 Srinivas 和 Deb 于 2000 年在 NSGA 的基础上提出的

与上述算法不同之处在于：

（1）利用非控制排序，实现多个值的计算，之前我们函数都是y=f(x1,x2,..xn)，这次我们变成了(y1,y2,..,ym)=f(x1,x2,...,xn)

（2）采用拥挤距离的概念，保持样本多样性，防止陷入局部最优

（3）对于NSGA-I而言，其降低了复杂度，并提高计算速度。

NSGA-II

1.非支配排序：计算每个个体的Rank值，且p支配q的意思是p的适应度函数值优于q的函数值，即(yp1,yp2..,ypn) are all better than (yq1,yq2,...,yqn).

（1）定义一个集合Sp=ϕ，表示被p所支配的集合。p指的是其中一条染色体。

（2）定义变量np=0，表示支配p个体的数目。

（3）在种群P中，找出被p支配的个体集合，以及求出支配p的个体的数目。

（4）若np=0的话，那么就说明没有个体能够支配p，则把p放入到first front集合F1当中，令F1=F1 U｛p｝并令其rank值等于1（p_{rank}=1）。

（5）同理，对于种群中的所有个体ki都按照类似的方法，找去S_{ki}，n_{ki}。最后把n_{ki}=0的个体放入F1中。

（6）令i=1，假如说Fi不等于空集时候，执行（7）否则结束。

（7）设定一个集合Q为空集，为了暂时保存个体。对于每一个属于F1的个体p进行遍历。找出在p的集合Sp中的元素ki，然后对每一个元素ki的n_{ki}值减一（n_{ki}=n_{ki}-1），如果nki值为0的话，就说明这个个体只是被p支配，令ki_{rank}=ki_{rank}+1，且把ki放入到Q之中（Q=Q U {ki}）

（8）把Q的集合中的个体放入Fi中，然后令i=i+1,循环下去，直到Fi集合为空集为止。

NSGA-II

NSGA-II

1.非支配排序：程序框图如下所示：

- for each individual p in main population P do the following
  - Initialize $S_p = \emptyset$. This set would contain all the individuals that is being dominated by p.
  - Initialize $n_p = 0$. This would be the number of individuals that dominate p.
  - for each individual q in P
    - if p dominated q then
      - add q to the set S_p i.e. $S_p = S_p \cup \{q\}$
    - else if q dominates p then
      - increment the domination counter for p i.e. $n_p = n_p + 1$
  - if $n_p = 0$ i.e. no individuals dominate p then p belongs to the first front; Set rank of individual p to one i.e $p_{rank} = 1$. Update the first front set by adding p to front one i.e $F_1 = F_1 \cup \{p\}$
- This is carried out for all the individuals in main population P.
- Initialize the front counter to one. $i = 1$
- following is carried out while the i^{th} front is nonempty i.e. $F_i \neq \emptyset$
  - $Q = \emptyset$. The set for storing the individuals for $(i+1)^{th}$ front.
  - for each individual p in front F_i
    - for each individual q in S_p (S_p is the set of individuals dominated by p)
      - $n_q = n_q - 1$, decrement the domination count for individual q.
      - if $n_q = 0$ then none of the individuals in the subsequent fronts would dominate q. Hence set $q_{rank} = i + 1$. Update the set Q with individual q i.e. $Q = Q \cup q$.
  - Increment the front counter by one.
  - Now the set Q is the next front and hence $F_i = Q$.

NSGA-II

2.拥挤距离：程序框图如下：

- For each front F_i, n is the number of individuals.
  - initialize the distance to be zero for all the individuals i.e. $F_i(d_j) = 0$, where j corresponds to the j^{th} individual in front F_i.
  - for each objective function m
    - Sort the individuals in front F_i based on objective m i.e. $I = sort(F_i, m)$.
    - Assign infinite distance to boundary values for each individual in F_i i.e. $I(d_1) = \infty$ and $I(d_n) = \infty$
    - for $k = 2$ to $(n - 1)$
      - $I(d_k) = I(d_k) + \dfrac{I(k+1).m - I(k-1).m}{f_m^{max} - f_m^{min}}$
      - $I(k).m$ is the value of the m^{th} objective function of the k^{th} individual in I

The basic idea behind the crowing distance is finding the euclidian distance between each individual in a front based on their m objectives in the m dimensional hyper space. The individuals in the boundary are always selected since they have infinite distance assignment.

NSGA-II

NSGA-II

3.复制

在我们计算了rank和拥挤距离之后，我们下一步要进行复制，怎么复制？通过一个叫做（crowed-comparison-operator）$(\prec_n)$的方法进行实现。

这个方法主要基于2种情况

（1）每一个个体都有一个rank值

（2）每一个个体都有一个拥挤距离。

实现：

首先，随机地选择种群中的两个个体，然后进行比较，如果满足 $p \prec_n q$

则选择p进行复制。所以刚才说了，为啥边缘的元素被选择的概率大，因为它的距离是无穷的。一般复制染色体数目是原始种群的数目的一半，但也可以多也可以少。

- $p \prec_n q$ if
  - $p_{rank} < q_{rank}$
  - or if p and q belong to the same front F_i then $F_i(d_p) > F_i(d_q)$ i.e. the crowing distance should be more.

NSGA-II

NSGA-II

4.遗传控制：交叉，变异的方式和SGA一样，运用的方法叫做：Simulated Binary Crossover及polynomial mutation。

1.交叉

（1）首先，设置一个交叉指数ηc，然后随机选择两个复制后的种群中的个体，随机生成一个0~1之间的数u，通过如下公式进行计算β(u)

$$\beta(u) = (2u)^{\frac{1}{(\eta+1)}} \quad u <= 0.5 \qquad \beta(u) = \frac{1}{[2(1-u)]^{\frac{1}{(\eta+1)}}} \quad u <= 0.5$$

（2）接下来计算新的子代c，其中1、k指的是第一个染色体中的第k个变量

$$c_{1,k} = \frac{1}{2}[(1-\beta_k)p_{1,k} + (1+\beta_k)p_{2,k}]$$

$$c_{2,k} = \frac{1}{2}[(1+\beta_k)p_{1,k} + (1-\beta_k)p_{2,k}]$$

（3）计算新的子代的适应度函数

NSGA-II

NSGA-II

4.遗传控制

2.变异

（1）计算 δ_k

$$\delta_k = (2r_k)^{\frac{1}{\eta_m + 1}} - 1, \quad \text{if } r_k < 0.5$$

$$\delta_k = 1 - [2(1 - r_k)]^{\frac{1}{\eta_m + 1}} \quad \text{if } r_k \geq 0.5$$

首先，设置一个变异指数ηc，然后随机选择一个个体，随机生成一个随机生成一个0~1之间的数rk，然后即可计算

（2）计算变异的新的子代 $c_k = p_k + (p_k^u - p_k^l)\delta_k$

公式中，pku,pkl分别是自变量k的最大、最小取值范围

NSGA-II

NSGA-II

5.合并

把子代和父代合并，然后重新非支配排序、拥挤度计算，最后筛选出和初始种群数目一样的染色体。

6.循环

交叉变异，不断循环，直到最优值不变或者到达迭代次数为止

NSGA-II

NSGA-II
流程图如下：

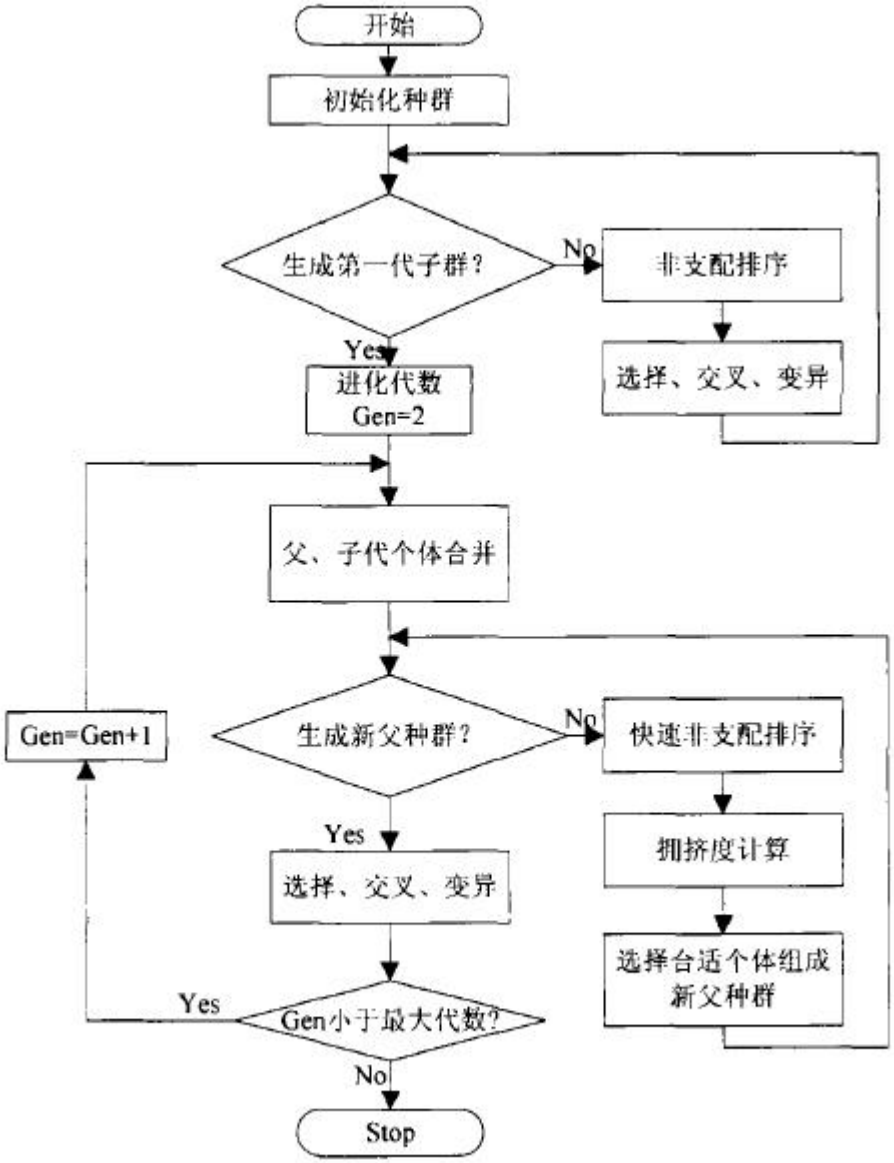


图 3.2 NSGA-II 基本流程

2.1 水力模型校核

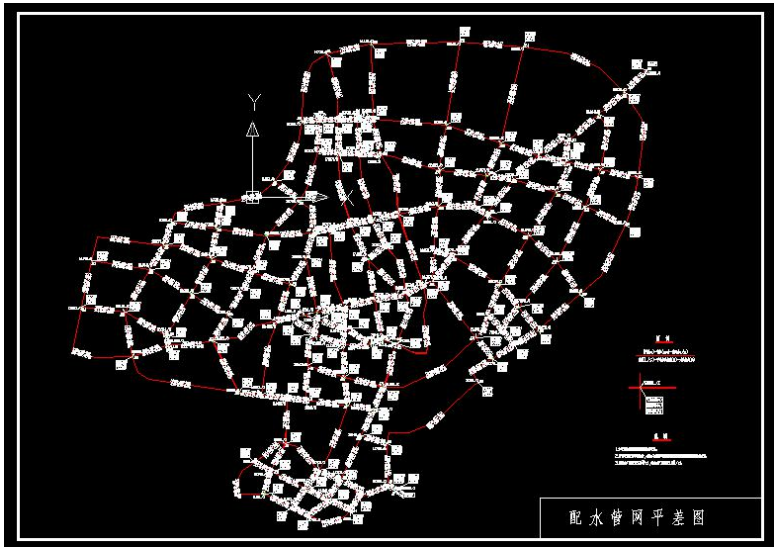
摩阻系数校核

摩阻系数

为什么要测定摩阻系数？

通过对供水管道比阻的测量,可以更好地了解供水管网中每条管道的通水能力,使整个供水管网工作更趋于合理，同时能够指出管网中哪些管道的比阻过大,需要更换或清洗。因此,对供水管网中管道比阻进行测量是十分必要的。虽然现场实测花费很大,但可以得到可信的结果。

在管网模拟当中，为了更好的使得模拟的数据与实际贴合，那么摩阻系数校核是必不可少的一个环节。



摩阻系数校核

摩阻系数

摩阻系数的取值大小关系到沿程水头损失计算结果的准确程度，在特定流体状态下选择适宜的摩阻系数计算公式是衡量计算参数准确性的主要理论依据。

海曾-威廉系数Ch

海曾-威廉公式表达形式为：

$$h_f = 10.67 \frac{Q^{1.85}}{C_h^{1.85} D^{4.87}} L$$

h_f——沿程水头损失（m）

Q——流量（m3/s）；

L——管长（m）；

D——管径（m）；

C_h——海曾-威廉系数。

摩阻系数校核

摩阻系数

系数Ch与管材、使用年限、水质、管中水压及水流速度等有关,其影响因素复杂,系数Ch、D从表面上看是确定的简单参数,但对旧管而言,由于使用年限不同,水质不同,结垢情况不同,它是变化的、难以确定的,甚至是非圆截面的。在实际管道中,真实管径D与粗糙系数Ch共同作用形成了比阻K.

$$h_f = 10.67 \frac{Q^{1.85}}{C_h^{1.85} D^{4.87}} L \Rightarrow i = \frac{h}{l} = KQ^{\alpha} \quad K = \frac{i}{Q^{\alpha}} \Rightarrow K = f(D, C).$$

为了计算水头损失，我们就需要设定一个系数Ch，如果设定的很好，那么水力模型中模拟的就与实际的相差很小，否则就会不符合实际。

若给水管很少，比如就有5条的话，那么我们自己手动调节这5个管段的摩阻系数，然后和实际进行对比，看看压力值是否能和实际值对得上，对得上就说明成功了。若给水管很多，成百上千个的话，我们一个个的尝试就不好使了，必须通过计算机来加快计算速度，节省人力物力。

摩阻系数校核

摩阻系数

那么我们就开始说一下如何计算。

问题，现在某小区的管网，有N个管段，其中有k条管段的摩阻系数未知，我们需要测定。

（1）收集数据阶段

我们要收集的数据是SCADA监测点的压力数据和流量数据，比如有6个检测点，我们每隔3小时取一次数据的话，那么数据就会8组流量和8组压力的数据。

收集完实际数据后，我们通过EPANET或者WATERGEMS软件进行水力建模，把管网导入到计算机中，然后输入各个管段、节点、阀门、水箱、水泵的参数，建立一个完整的水力模型。

（2）摩阻系数校正阶段

我们选取7组数据作为训练数据，最后一组数据作为测试数据。将k条管段的摩阻系数作为自变量因素，且根据经验及资料设定各个管段系数C的的取值范围，首先随机生成母体染色体，然后带入到GA或者其变种的算法之中。在遗传算法中，适应度函数选择如下：

$$\sum |\Delta H| = \sum_{i=1}^{M} \sum_{j=1}^{N} |H_{ij} - H_{ij0}|$$

摩阻系数校核

摩阻系数

（2）摩阻系数校正阶段

当我们遗传算法运行完成之后，我们把最终获得的摩阻系数值带入到测试数据的流量情况之中，然后计算：

$$|\Delta H| = \sum_{j=1}^{N'} |H_{ij} - H_{ij0}|$$

若|ΔH|很小，则说明计算的不错，若不行的话，返回GA，重新进行校核，直到满足|ΔH|很小为止。

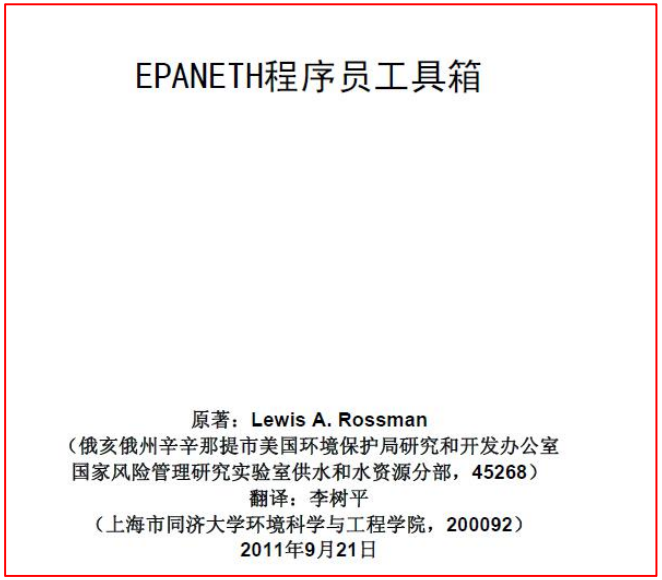
摩阻系数校核

摩阻系数

有人会问，适应度函数里面没有摩阻系数C，都是H，我们怎么计算呢？

$$\sum|\Delta H| = \sum_{i=1}^{M}\sum_{j=1}^{N}|H_{ij} - H_{ij0}|$$

在随机生成摩阻系数后，我们将其输入到管网模型中，通过管网平差计算，即可得出各个节点的压力H。因此该地方需要对EPANET进行二次开发，开发的方式有很多，一般应用C语言是最佳的方式，大家感兴趣的话可以参考：

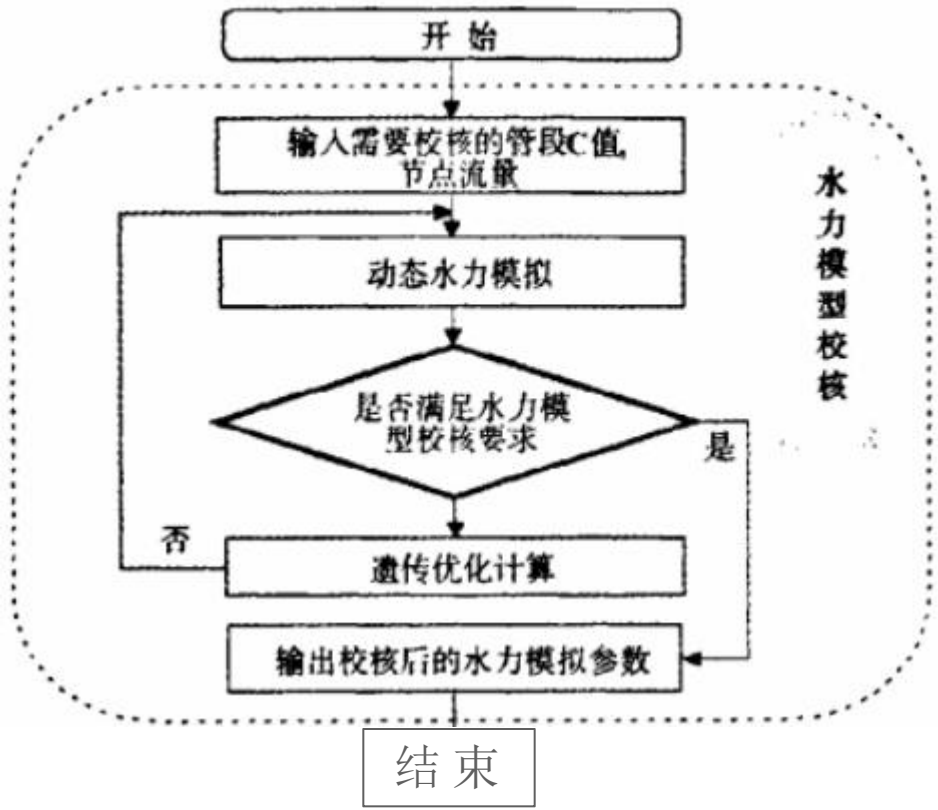

EPANETH程序员工具箱

原著：Lewis A. Rossman
（俄亥俄州辛辛那提市美国环境保护局研究和开发办公室
国家风险管理研究实验室供水和水资源分部，45268）
翻译：李树平
（上海市同济大学环境科学与工程学院，200092）
2011年9月21日

摩阻系数校核

摩阻系数

摩阻系数校核的流程图如下所示：



2.2 校核拓展

摩阻系数校核

摩阻系数校核拓展：

一般校核摩阻系数的时候，同时带上管网水质系数的校核,水质校核分为两个部分，一个部分是管段的校核，另一部分是管壁的校核。

下面是管段的校核：

主体水反应校核

主体水反应是在管道流动的水里或在水池里发生的反应，不被管壁反应所影响。水质模型用 n 级动力学模拟这些反应，假设瞬间反应速率（R，单位：质量/体积/时间）取决于浓度：

$$R(c_j) = \lambda_j c_j^n \quad (1)$$

式中 λ_j 是主体水反应速率系数；c_j 是节点 j 处的反应物浓度；n 是反应级数；λ_j 是单位浓度上升到 $(1-n)$ 次方除以时间。当最终的生成物或衰减物有一个限制浓度时，

$$R(c_j) = \lambda_j (c_L - c_j) c_j^{n-1} \quad (2)$$

式中 c_L 是限制浓度。衰减反应时用 $(c_j - c_L)$ 取代 $(c_L - c_j)$。

因此，参数 λ_j、c_L 和 n 是主体水反应动力学模型的关键。主体水反应校核主要是调整这三个参数。

下面是管壁的校核：

管壁反应校核

管壁反应动力学模型可表示为：

$$R(c_j) = (A/V) K_w c_j^n \quad (3)$$

式中 K_w 是管壁反应速率系数；(A/V) 是管道中单位体积水接触的管壁表面积；n 是管壁反应级数，管壁反应一般分零级和一级反应，n 的取值为0或1。管壁反应校核主要是调整 K_w 和 n 这两个参数。

管壁反应校核主要有直接校核法和关联校核法。直接校核法就是直接优化 K_w 和 n 这两个参数。一般认为相同属性（管龄、管材等）的管道发生管壁反应的动力学是一样的，可以把不同属性的管道分成几个不同的组进行校核。关联校核法是通过调整关联系数进行校核。Vasconelos 等人的研究表明，金属管道管壁反应速率系数 K_w 和管道海曾－威廉系数 C 值是成函数关系的[2]。随着管龄的逐渐增大，C 值逐渐变小，管壁和化学物质的反应活性也随之增大。所以，通过调整 C 值可以间接调整 K_w。

摩阻系数校核

摩阻系数校核拓展：

在EPANET的水力模型中当中不必校核这么多参数，只需要两个，一个是管段的反应速率系数，另外一个是管壁的反应速率系数。

校核的时候，我们的适应度函数就变成两个了：

$$\sum|\Delta H|=\sum_{i=1}^{M}\sum_{j=1}^{N}|H_{ij}-H_{ij0}|$$

$$\sum|\Delta C|=\sum_{i=1}^{M}\sum_{j=1}^{T}|c_{ij}-c_{ij0}|$$

式中 $H_{i,t}$——第 i 个测压点第 t 个时段的压力值，m；

$C_{j,t}$——第 j 个水质监测点第 t 个时段某水质指标浓度为 $C_{j,t}$，mg/L；

$H_{i,t}^{sim}$——根据水力模型计算出的第 i 个测压点第 t 个时段的压力值，m；

$C_{j,t}^{sim}$——根据水质模型计算出的第 j 个水质监测点第 t 个时段某水质指标浓度为 $C_{j,t}$，mg/L；

n——测压点总个数；

T——水质监测点总个数。

摩阻系数校核

摩阻系数校核拓展：

对于2个适应度函数，计算的方式有两种，一种是用普通的遗传算法，把两个适应度函数权值加和：ie:y=k1y1+k2y2 (k1+k2=1)

第二种方法是利用NSGA-II，来计算多目标函数优化求解。

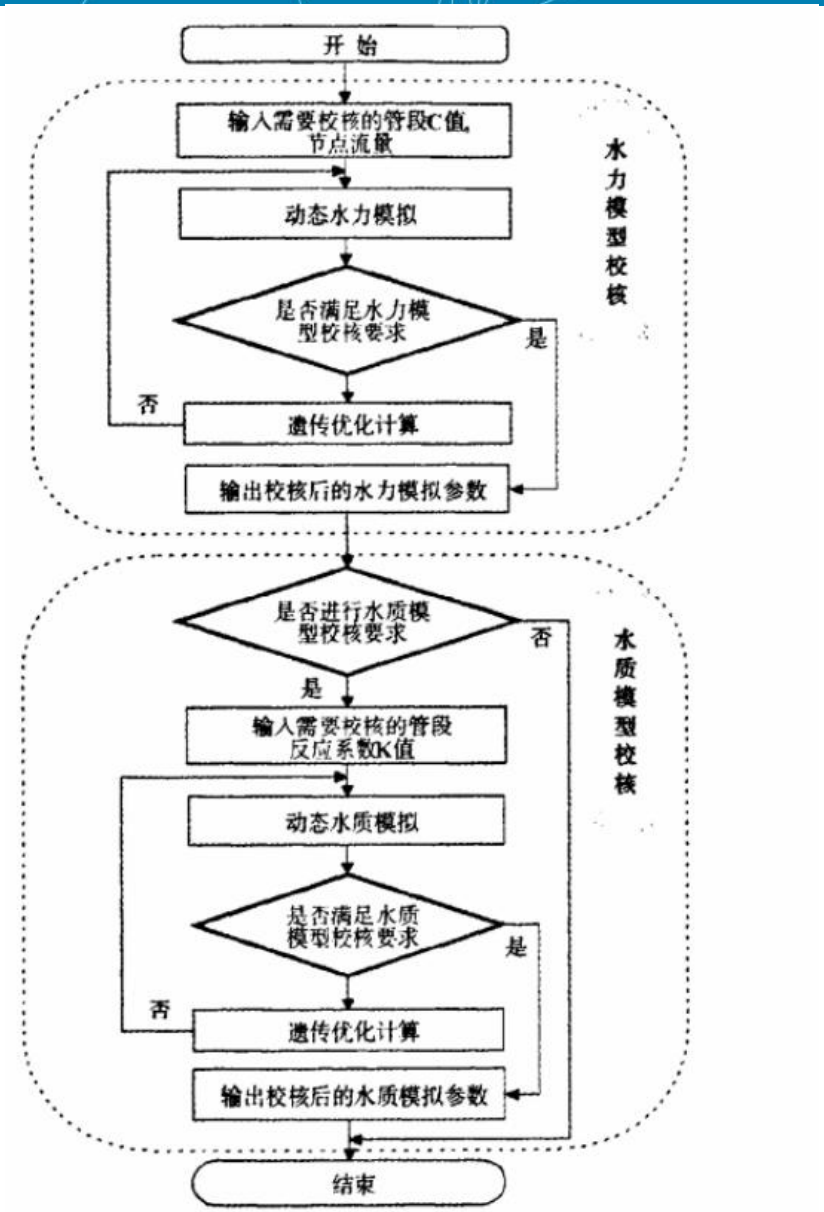
计算水质的时候，除了系数取值在一定的范围内之外，还需要有一些限定的条件，正如我们第七讲-管网水质中讲到的，我们利用EPANET可以直接满足，不用自己再添加。

通过计算可以得到水质及摩阻系数的最优解，之后把最优解带入到测试数据中进行检验，若|ΔH|、|ΔC|很小，则说明计算的不错，若不行的话，返回重新进行校核，直到满足|ΔH|、|ΔC|很小为止。

摩阻系数校核

摩阻系数校核拓展：

流程图如图所示：



3 结语

结语（大结局）

参考文献：

【1】Rossman, L. A. (2002). EPANET 2.0 user’ s manual, National Risk Management Research Laboratory, U.S. EPA, Cincinnati.

【2】Rossman L A. The EPANET water quality model[M]// Integrated computer applications in water supply (vol. 2). John Wiley and Sons Ltd. 1994:79-93.

【3】Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Transactions Evolutionary Computation, 2002, 6(2):182-197.

【4】Goldberg D E, Deb K, Kargupta H, et al. RapidAccurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms[J]. Proc.of the 5th ICGA, 1993:56-64.

【5】Kargupta H. Search, polynomial complexity, and the fast messy genetic algorithm[M]. University of Illinois at Urbana-Champaign, 1996.

【6】Shu S, Liu S, Shu S, et al. Water Quality Model Calibration of Water Distribution Network Using Parallel FmGA Algorithm[C]// International Conference on Electrical and Control Engineering. IEEE Computer Society, 2010:5801-5804.

【7】舒诗湖. 基于fmGA的供水管网系统模型自动校核及模型应用[D]. 哈尔滨工业大学, 2009.

【8】卓敏. 给水管网管道摩阻的灵敏度分析及其校正研究[D]. 浙江大学, 2005.

【9】卓金武. MATLAB在数学建模中的应用[M]. 北京航空航天大学出版社, 2011.

结语（大结局）

到这里我的管网八讲全部都讲完了，这个课程我因为时间及懒惰的原因，录制过程足足拖拖拉拉了半年，算是抱歉了。真心感谢像追番追剧一样看着我讲课的那些朋友们，你们是我的粉丝，你们都是我的家人。小木我今后还会录制更多的课程，并且创立网站，一直坚持免费，公益的教学，把正能量发扬光大。谢谢大家！

希望各位事业有成，学业顺利!

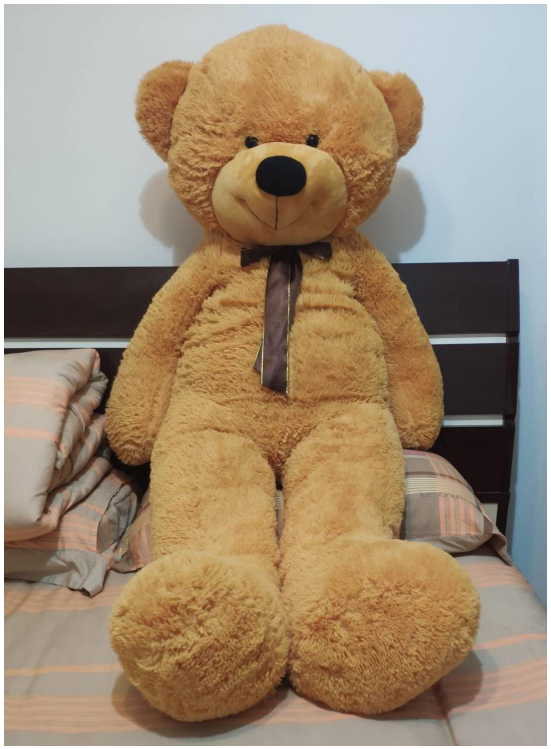
——牟天蔚

2018.04.12

结语（大结局）

最后在结束的时候，我给大家放一首歌，超好听的，作为我们离别的礼物吧，希望我们稍后会再见面的哦！

あの頃~ジンジンバオヂュオニー



请各位同学老师批评指正

小木

http://blog.csdn.net/u013631121

求三连！！！