第三章　遗传算法的基本实现技术

3.1　编码方法

在遗传算法的运行过程中，它不对所求解问题的实际决策变量直接进行操作，而是对表示可行解的个体编码施加选择、交叉、变异等遗传运算，通过这种遗传操作来达到优化的目的，这是遗传算法的特点之一。遗传算法通过这种对个体编码的操作，不断搜索出适应度较高的个体，并在群体中逐渐增加其数量，最终寻求出问题的最优解或近似最优解。在遗传算法中如何描述问题的可行解，即把一个问题的可行解从其解空间转换到遗传算法所能处理的搜索空间的转换方法就称为编码。

编码是应用遗传算法时要解决的首要问题，也是设计遗传算法时的一个关键步骤。编码方法除了决定了个体的染色体排列形式之外，它还决定了个体从搜索空间的基因型变换到解空间的表现型时的解码方法，编码方法也影响到交叉算子、变异算子等遗传算子的运算方法。由此可见，编码方法在很大程度上决定了如何进行群体的遗传进化运算以及遗传进化运算的效率。一个好的编码方法，有可能会使得交叉运算、变异运算等遗传操作可以简单地实现和执行。而一个差的编码方法，却有可能会使得交叉运算、变异运算等遗传操作难以实现，也有可能会产生很多在可行解集合内无对应可行解的个体，这些个体经解码处理后所表示的解称为无效解。虽然有时产生一些无效解并不完全都是有害的，但大部分情况下它却是影响遗传算法运行效率的主要因素之一。

针对一个具体应用问题，如何设计一种完美的编码方案一直

是遗传算法的应用难点之一，也是遗传算法的一个重要研究方向。可以说目前还没有一套既严密又完整的指导理论及评价准则能够帮助我们设计编码方案。作为参考，De Jong 曾提出了两条操作性较强的实用编码原则（又称为编码规则）[4]：

●**编码原则一**（有意义积木块编码原则）：应使用能易于产生与所求问题相关的且具有低阶、短定义长度模式的编码方案。

●**编码原则二**（最小字符集编码原则）：应使用能使问题得到自然表示或描述的具有最小编码字符集的编码方案。

第一个编码原则中，模式是指具有某些基因相似性的个体的集合，而具有短定义长度、低阶且适应度较高的模式称为构造优良个体的积木块或基因块，这点后面再详细叙述。这里可以把该编码原则理解成应使用易于生成适应度较高的个体的编码方案。

第二个编码原则说明了我们为何偏爱于使用二进制编码方法的原因，因为它满足这条编码原则的思想要求。事实上，理论分析表明，与其他编码字符集相比，二进制编码方案能包含最大的模式数，从而使得遗传算法在确定规模的群体中能够处理最多的模式。

需要说明的是，上述 De Jong 编码原则仅仅是给出了设计编码方案时的一个指导性大纲，它并不适合于所有的问题。所以对于实际应用问题，仍必须对编码方法、交叉运算方法、变异运算方法、解码方法等统一考虑，以寻求到一种对问题的描述最为方便、遗传运算效率最高的编码方案。

由于遗传算法应用的广泛性，迄今为止人们已经提出了许多种不同的编码方法。总的来说，这些编码方法可以分为三大类：二进制编码方法、浮点数编码方法、符号编码方法。下面我们从具体实现的角度出发介绍其中的几种主要编码方法。

3.1.1 二进制编码方法

二进制编码方法是遗传算法中最常用的一种编码方法，它使用的编码符号集是由二进制符号 0 和 1 所组成的二值符号集 {0, 1}，它所构成的个体基因型是一个二进制编码符号串。

二进制编码符号串的长度与问题所要求的求解精度有关。假设某一参数的取值范围是 $[U_{min}, U_{max}]$，我们用长度为 l 的二进制编码符号串来表示该参数，则它总共能够产生 2^l 种不同的编码，若使参数编码时的对应关系如下：

$$\begin{aligned}
&00000000\cdots00000000=0 &&\longrightarrow && U_{min}\\
&00000000\cdots00000001=1 &&\longrightarrow && U_{min}+\delta\\
&\quad\vdots\quad\vdots\quad\vdots && && \vdots\quad\vdots\\
&11111111\cdots11111111=2^l-1 &&\longrightarrow && U_{max}
\end{aligned}$$

则二进制编码的编码精度为：

$$\delta=\frac{U_{max}-U_{min}}{2^l-1} \tag{3-1}$$

假设某一个体的编码是：

$$X:\ b_l b_{l-1} b_{l-2}\cdots b_2 b_1$$

则对应的解码公式为：

$$x=U_{min}+\left(\sum_{i=1}^{l} b_i\cdot 2^{i-1}\right)\cdot\frac{U_{max}-U_{min}}{2^l-1} \tag{3-2}$$

例如，对于 $x\in[0, 1023]$，若用 10 位长的二进制编码来表示该参数的话，则下述符号串：

$$X:\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1$$

就可表示一个个体，它所对应的参数值是 $x=175$。此时的编码精度为 $\delta=1$。

二进制编码方法有下述一些优点：

(1) 编码、解码操作简单易行。

(2) 交叉、变异等遗传操作便于实现。

(3) 符合最小字符集编码原则。

(4) 便于利用模式定理对算法进行理论分析。

3.1.2 格雷码编码方法

二进制编码不便于反映所求问题的结构特征，对于一些连续函数的优化问题等，也由于遗传运算的随机特性而使得其局部搜索能力较差。为改进这个特性，人们提出用格雷码 (Gray code)

来对个体进行编码。

格雷码是这样的一种编码方法，其连续的两个整数所对应的编码值之间仅仅只有一个码位是不相同的，其余码位都完全相同。例如十进制数 0～15 之间的二进制码和相应的格雷码分别如表 3-1 所示。

表 3-1　二进制码与格雷码

十进制数	二进制码	格雷码
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

假设有一个二进制编码为 $B = b_m b_{m-1} \cdots b_2 b_1$，其对应的格雷码为 $G = g_m g_{m-1} \cdots g_2 g_1$。由二进制编码到格雷码的转换公式为：

$$\begin{cases} g_m = b_m \\ g_i = b_{i+1} \oplus b_i, \quad i = m-1, \ m-2, \ \cdots, \ 1 \end{cases} \tag{3-3}$$

由格雷码到二进制码的转换公式为：

$$\begin{cases} b_m = g_m \\ b_i = b_{i+1} \oplus g_i, \quad i = m-1, \ m-2, \ \cdots, \ 1 \end{cases} \tag{3-4}$$

上面两种转换公式中，⊕表示异或运算符。

格雷码有这样一个特点：任意两个整数的差是这两个整数所对应的格雷码之间的海明距离（Hamming distance）。这个特点是遗传算法中使用格雷码来进行个体编码的主要原因。

遗传算法的局部搜索能力不强，引起这个问题的主要原因是，新一代群体的产生主要是依靠上一代群体之间的随机交叉重组来完成的，所以即使已经搜索到最优解附近，而想要达到这个最优解，却要费一番功夫，甚至需要花费较大的代价。对于用二进制编码方法表示的个体，变异操作有时虽然只是一个基因座的差异（个体基因型 X 的微小差异），而对应的参数值却相差较大（个体表现型 X 相差较大）。但是，若使用格雷码来对个体进行编码，则编码串之间的一位差异，对应的参数值也只是微小的差别。这样就相当于增强了遗传算法的局部搜索能力，便于对连续函数进行局部空间搜索。

例如，对于区间 [0, 1023] 中两个邻近的整数 $x_1 = 175$ 和 $x_2 = 176$，若使用长度为 10 位的二进制编码，它们可分别表示为：

X_1：0 0 1 0 1 0 1 1 1 1

X_2：0 0 1 0 1 1 0 0 0 0

而使用同样长度的格雷码，它们可分别表示为：

X_1：0 0 1 1 1 1 1 0 0 0

X_2：0 0 1 1 1 0 1 0 0 0

显见，使用格雷码时，两个编码串之间只有一位编码值不同；而使用二进制编码时，两个编码串之间却相差较大。

格雷码编码方法是二进制编码方法的一种变形，其编码精度与相同长度的二进制编码的精度相同。

格雷码编码方法的主要优点是：

(1) 便于提高遗传算法的局部搜索能力。

(2) 交叉、变异等遗传操作便于实现。

(3) 符合最小字符集编码原则。

(4) 便于利用模式定理对算法进行理论分析。

3.1.3 浮点数编码方法

对于一些多维、高精度要求的连续函数优化问题，使用二进制编码来表示个体时将会有一些不利之处。

首先是二进制编码存在着连续函数离散化时的映射误差。个体编码串的长度较短时，可能达不到精度要求；而个体编码串的长度较长时，虽然能提高编码精度，但却会使遗传算法的搜索空间急剧扩大。例如，若使用二进制编码方法来处理一个含有 100 个决策变量的优化问题，其中每个决策变量的取值范围是 [−250, 250]，要求精度取小数点后 5 位小数，为达到这个精度要求，每个变量必须用 26 位长的二进制编码符号串来表示，这是因为：

$$2^{25} = 33554432 < \frac{500}{0.00001} = 50000000 < 67108864 = 2^{26}$$

这样每个个体必须用 100×26 位长的二进制编码符号串表示。亦即此时遗传算法的搜索空间大约是 2^{2600}。在如此之大的搜索空间寻优肯定会使得遗传算法的运行性能相当差，甚至可能无法进行下去。

其次是二进制编码不便于反映所求问题的特定知识，这样也就不便于开发针对问题专门知识的遗传运算算子，人们在一些经典优化算法的研究中所总结出的一些宝贵经验也就无法在这里加以利用，也不便于处理非平凡约束条件。

为改进二进制编码方法的这些缺点，人们提出了个体的浮点数编码方法。所谓浮点数编码方法，是指个体的每个基因值用某一范围内的一个浮点数来表示，个体的编码长度等于其决策变量的个数。因为这种编码方法使用的是决策变量的真实值，所以浮点数编码方法也叫做真值编码方法。

例如，若某一个优化问题含有 5 个变量 x_i（$i=1$，2，…5），每个变量都有其对应的上下限 $[U^i_{min}, U^i_{max}]$，则

X：	5.80	6.90	3.50	3.80	5.00

就表示一个体的基因型，其对应的表现型是：$x=[5.80, 6.90, 3.50, 3.80, 5.00]^T$。

在浮点数编码方法中，必须保证基因值在给定的区间限制范围内，遗传算法中所使用的交叉、变异等遗传算子也必须保证其运算结果所产生的新个体的基因值也在这个区间限制范围内。再者，当用多个字节来表示一个基因值时，交叉运算必须在两个基因的分界字节处进行，而不能在某个基因的中间字节分隔处进行。

浮点数编码方法有下面几个优点[14, 15]：

(1) 适合于在遗传算法中表示范围较大的数。

(2) 适合于精度要求较高的遗传算法。

(3) 便于较大空间的遗传搜索。

(4) 改善了遗传算法的计算复杂性，提高了运算效率。

(5) 便于遗传算法与经典优化方法的混合使用。

(6) 便于设计针对问题的专门知识的知识型遗传算子。

(7) 便于处理复杂的决策变量约束条件。

3.1.4 符号编码方法

符号编码方法是指个体染色体编码串中的基因值取自一个无数值含义、而只有代码含义的符号集。这个符号集可以是一个字母表，如｛A，B，C，D，…｝；也可以是一个数字序号表，如｛1，2，3，4，5，…｝；还可以是一个代码表，如｛A1，A2，A3，A4，A5，…｝等等。

例如，对于旅行商问题，假设有 n 个城市分别记为 C_1、C_2、…、C_n，将各个城市的代号按其被访问的顺序连接在一起，就可构成一个表示旅行路线的个体。如

$$X：[C_1, C_2, \cdots, C_n]$$

就表示顺序访问城市 C_1、C_2、…、C_n。若将各个城市按其代号的下标进行编号，则这个个体也可表示为：

$$X:[\,1,\ 2,\ \cdots,\ n\,]$$

符号编码的主要优点是：

(1) 符合有意义积木块编码原则。

(2) 便于在遗传算法中利用所求解问题的专门知识。

(3) 便于遗传算法与相关近似算法之间的混合使用。

但对于使用符号编码方法的遗传算法，一般需要认真设计交叉、变异等遗传运算的操作方法，以满足问题的各种约束要求，这样才能提高算法的搜索性能。

3.1.5 多参数级联编码方法

一般常见的优化问题中往往含有多个决策变量。例如六峰值驼背函数（Six-hump Camel Back Function）：

$$f\ (x,\ y)\ =\left(4-2.1x^2+\frac{x^4}{3}\right)x^2+xy+\ (-4+4y^2)\ y^2 \tag{3-5}$$

就含有两个变量 x 和 y。对这种含有多个变量的个体进行编码的方法就称为多参数编码方法。

其实，在我们前面的例子中已遇到过多参数编码的一种最常用和最基本的方法：将各个参数分别以某种编码方法进行编码，然后再将它们的编码按一定顺序联接在一起就组成了表示全部参数的个体编码。这种编码方法称为多参数级联编码方法。

在进行多参数级联编码时，每个参数的编码方式可以是二进制编码、格雷码、浮点数编码或符号编码等编码方式中的一种，每个参数可以具有不同的上下界，也可以有不同的编码长度或编码精度。例如，假设一种个体含有 n 个参数，每个参数用 l_i（$i=1,\ 2,\ \cdots,\ n$）位的二进制编码来表示，则该个体可表示为：

$$\underbrace{b_{11}b_{12}\cdots b_{1l_1}}_{x_1}\ \vdots\ \underbrace{b_{21}b_{22}\cdots b_{2l_2}}_{x_2}\ \vdots\ \cdots\ \vdots\ \underbrace{b_{n1}b_{n2}\cdots b_{nl_n}}_{x_n}$$

该编码串的总长度为 $l=\sum_{i=1}^{n} l_i$。这也是多参数二进制级联编码的一种通用形式。

3.1.6 多参数交叉编码方法

多参数交叉编码方法的基本思想是[16]：将各个参数中起主要作用的码位集中在一起，这样它们就不易于被遗传算子破坏掉。

在进行多参数交叉编码时，可先对各个参数进行分组编码(假设共有 n 个参数，每个参数都用长度为 m 的二进制编码串来表示)；然后取各个参数编码串中的最高位联接在一起，以它们作为个体编码串的前 n 位编码，再取各个参数编码串中的次高位联接在一起，以它们作为个体编码串的第二组 n 位编码，……，取各个参数编码串中的最后一位联接在一起，以它们作为个体编码串的最后 n 位。这样所组成的长度为 $m\times n$ 位的编码串就是多参数的一个交叉编码串，如下所示：

参数编码 $\overbrace{b_{11}b_{12}b_{13}\cdots b_{1m}}^{x_1}\ \overbrace{b_{21}b_{22}b_{23}\cdots b_{2m}}^{x_2}\cdots\overbrace{b_{n1}b_{n2}b_{n3}\cdots b_{nm}}^{x_n}$

↓

个体编码串

$$b_{11}b_{21}\cdots b_{n1}\ \vdots\ b_{12}b_{22}\cdots b_{n2}\ \vdots\ b_{13}b_{23}\cdots b_{n3}\ \vdots\ \cdots\ \vdots\ b_{1m}b_{2m}\cdots b_{nm}$$

在前述多参数的级联编码方法中，各个参数的编码值集中在一起，这样各个参数的局部编码结构就不易被遗传算子破坏掉，它适合于各参数之间的相互关系较弱，特别是某一个或少数几个参数起主要作用时的优化问题。而多参数的交叉编码方法特别适合于各个参数之间的相互关系较强、各参数对最优解的贡献相当时的优化问题，因为在这种交叉编码方法中，用来表示各个参数值的二进制编码的最高位被集中在了一起，它们就不易被遗传算子破坏掉，而这些最高位在表示各个参数值时所起的作用最强，这样就可以尽量地维持各参数之间的相互关系。

3.2 适应度函数

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其生存环境的适应程度。对生存环境适应程度较高的物种将有更多的繁殖机会；而对生存环境适应程度较低的物种，其繁殖机会就相对较少，甚至会逐渐灭绝。与此相类似，遗传算法中也使用适应度这个概念来度量群体中各个个体在优化计算中有可能达到或接近于或有助于找到最优解的优良程度。适应度较高的个体遗传到下一代的概率就较大；而适应度较低的个体遗传到下一代的概率就相对小一些。度量个体适应度的函数称为适应度函数（Fitness Function）。

3.2.1 目标函数与适应度函数

遗传算法的一个特点是它仅使用所求问题的目标函数值就可得到下一步的有关搜索信息。而对目标函数值的使用是通过评价个体的适应度来体现的。评价个体适应度的一般过程是：

(1) 对个体编码串进行解码处理后，可得到个体的表现型。

(2) 由个体的表现型可计算出对应个体的目标函数值。

(3) 根据最优化问题的类型，由目标函数值按一定的转换规则求出个体的适应度。

最优化问题可分为两大类，一类为求目标函数的全局最大值，另一类为求目标函数的全局最小值。对于这两类优化问题，第二章中已经介绍过由解空间中某一点的目标函数值 $f(X)$ 到搜索空间中对应个体的适应度函数值 $F(X)$ 的转换方法：

●对于求最大值的问题，作下述转换：

$$F(X)=\begin{cases} f(X)+C_{\min}, & \text{if } f(X)+C_{\min}>0 \\ 0, & \text{if } f(X)+C_{\min}\leqslant 0 \end{cases} \tag{3-6}$$

式中，$C_{\min}$为一个适当地相对较小的数。

●对于求最小值的问题，作下述转换：

$$F(X)=\begin{cases}C_{\max}-f(X), & \text{if} \quad f(X)<C_{\max}\\ 0, & \text{if} \quad f(X)\geqslant C_{\max}\end{cases} \tag{3-7}$$

式中，$C_{\max}$为一个适当地相对较大的数。

遗传算法中，群体的进化过程就是以群体中各个个体的适应度为依据，通过一个反复迭代过程，不断地寻求出适应度较大的个体，最终就可得到问题的最优解或近似最优解。

3.2.2 适应度尺度变换

在遗传算法中，各个个体被遗传到下一代群体中的概率是由该个体的适应度来确定的。应用实践表明，仅使用式（3-6）或式（3-7）来计算个体适应度时，有些遗传算法会收敛得很快，也有些遗传算法会收敛得很慢。由此可见，如何确定适应度对遗传算法的性能有较大的影响。

例如，在遗传算法运行的初期阶段，群体中可能会有少数几个个体的适应度相对其他个体来说非常高。若按照常用的比例选择算子来确定个体的遗传数量时，则这几个相对较好的个体将在下一代群体中占有很高的比例，在极端情况下或当群体规模较小时，新的群体甚至完全由这样的少数几个个体所组成。这时产生新个体作用较大的交叉算子就起不了什么作用，因为相同的两个个体不论在何处进行交叉操作都永远不会产生出新的个体，如下所示：

$$\begin{array}{l} A:\ 1\,0\,1\,0\,1\,0 \ \vdots\ 1\,0\,1\,0 \\ B:\ 1\,0\,1\,0\,1\,0 \ \vdots\ 1\,0\,1\,0 \end{array} \xrightarrow{\text{单点交叉}} \begin{array}{l} A':\ 1\,0\,1\,0\,1\,0 \ \vdots\ 1\,0\,1\,0 \\ B':\ 1\,0\,1\,0\,1\,0 \ \vdots\ 1\,0\,1\,0 \end{array}$$

交叉点

这样就会使群体的多样性降低，容易导致遗传算法发生早熟现象（或称早期收敛），使遗传算法所求到的解停留在某一局部最优点上。为了克服这种现象，我们希望在遗传算法运行的初期阶段，算法能够对一些适应度较高的个体进行控制，降低其适应度与其他个体适应度之间的差异程度，从而限制其复制数量，以维护群

体的多样性。

又例如，在遗传算法运行的后期阶段，群体中所有个体的平均适应度可能会接近于群体中最佳个体的适应度。也就是说，大部分个体的适应度和最佳个体的适应度差异不大，它们之间无竞争力，都会有以相接近的概率被遗传到下一代的可能性，从而使得进化过程无竞争性可言，只是一种随机的选择过程。这将导致无法对某些重点区域进行重点搜索，从而影响遗传算法的运行效率。为了克服这种现象，我们希望在遗传算法运行的后期阶段，算法能够对个体的适应度进行适当的放大，扩大最佳个体适应度与其他个体适应度之间的差异程度，以提高个体之间的竞争性。

由此看来，不能仅仅依靠式（3-6）或式（3-7）就完全确定出个体的适应度，有时在遗传算法运行的不同阶段，还需要对个体的适应度进行适当的扩大或缩小。这种对个体适应度所做的扩大或缩小变换就称为适应度尺度变换（Fitness Scaling）。

目前常用的个体适应度尺度变换方法主要有三种：线性尺度变换、乘幂尺度变换和指数尺度变换。

（1）线性尺度变换。线性尺度变换的公式如下：

$$F' = aF + b \tag{3-8}$$

式中 F——原适应度；

F'——尺度变换后的新适应度；

a 和 b——系数。

线性比例尺度变换的正常情况如图 3-1 所示。由该图可见，系数 a、b 直接影响到这个尺度变换的大小，所以对其选取有一定的要求，一般希望它们满足下面两个条件[5]：

条件一：尺度变换后全部个体的新适应度的平均值 F'_{avg}要等于其原适应度平均值 F_{avg}。即：

$$F'_{avg} = F_{avg} \tag{3-9}$$

这条要求是为了保证群体中适应度接近于平均适应度的个体能够有期待的数量被遗传到下一代群体中。

条件二：尺度变换后群体中新的最大适应度 F'_{max}要等于其

原平均适应度 F_{avg}的指定倍数。即：

$$F'_{max} = C \cdot F_{avg} \tag{3-10}$$

式中，C 为最佳个体的期望复制数量，对于群体规模大小为 50～100 个个体的情况，一般取 $C=1.2\sim2$。这条要求是为了保证群体中最好的个体能够期望复制 C 倍到新一代群体中。

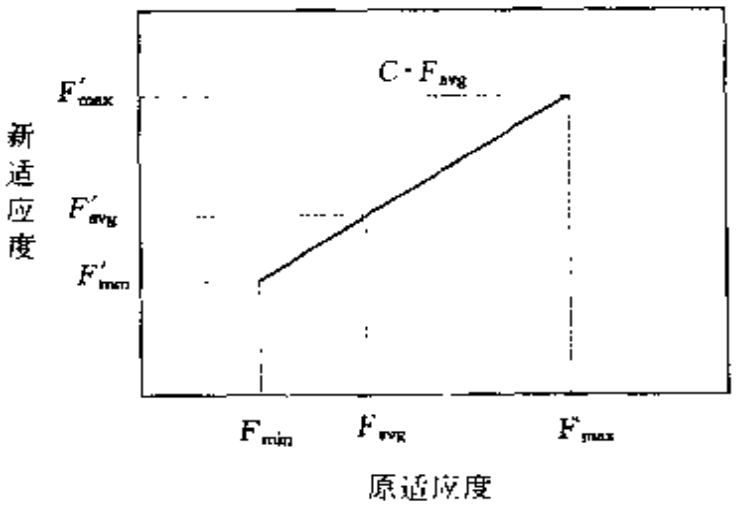

图 3-1　线性尺度变换的正常情况

使用线性尺度变换时，如图 3-1 所示，群体中少数几个优良个体的适应度按比例缩小，同时几个较差个体的适应度也按比例扩大。但在搜索过程的后期阶段，随着个体适应度从总体上的不断改进，群体中个体的最大适应度和全部个体的平均适应度较接近，而少数几个较差的个体的适应度却远远低于最大适应度，这时若想维持 F'_{max}和 F_{avg}的指定倍数关系，将有可能会使较差个体的适应度变换为负值，如图 3-2 所示。这将会给后面的处理过程带来不便，必须避免这种情况的发生。解决这个问题的方法是：把原最小适应度 F_{min}映射为 $F'_{min}=0$，并且保持原平均适应度 F_{avg}与新的平均适应度 F'_{avg}相等。

（2）乘幂尺度变换。乘幂尺度变换的公式为：

$$F' = F^k \tag{3-11}$$

即新的适应度是原有适应度的某个指定乘幂。幂指数 k 与所求解的问题有关，并且在算法的执行过程中需要不断对其进行修正

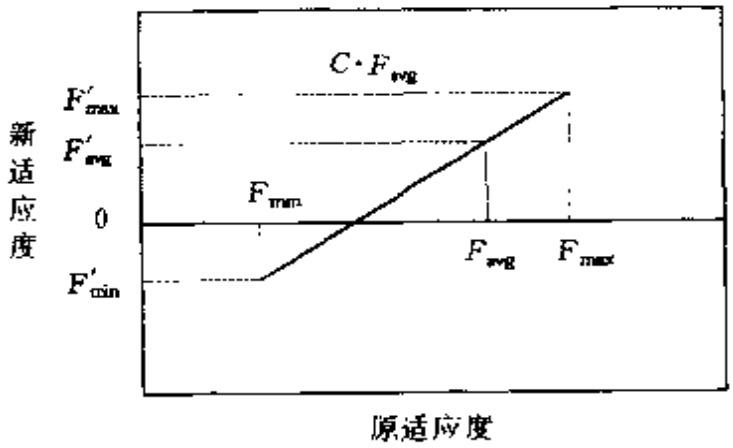

图 3-2　线性尺度变换的异常情况

才能使尺度变换满足一定的伸缩要求。

(3) 指数尺度变换。指数尺度变换的公式为：

$$F' = \exp(-\beta F) \tag{3-12}$$

即新的适应度是原有适应度的某个指数。式中系数 β 决定了选择的强制性，β 越小，原有适应度较高的个体的新适应度就越与其他个体的新适应度相差较大，亦即越增加了选择该个体的强制性。

3.3　选 择 算 子

在生物的遗传和自然进化过程中，对生存环境适应程度较高的物种将有更多的机会遗传到下一代；而对生存环境适应程度较低的物种遗传到下一代的机会就相对较少。模仿这个过程，遗传算法使用选择算子（或称复制算子，Reproduction Operator）来对群体中的个体进行优胜劣汰操作：适应度较高的个体被遗传到下一代群体中的概率较大；适应度较低的个体被遗传到下一代群体中的概率较小。遗传算法中的选择操作就是用来确定如何从父代群体中按某种方法选取哪些个体遗传到下一代群体中的一种遗传运算。

选择操作建立在对个体的适应度进行评价的基础之上。选择

操作的主要目的是为了避免基因缺失、提高全局收敛性和计算效率。

最常用的选择算子是基本遗传算法中的比例选择算子。但对于各种不同的问题，比例选择算子并不是最合适的一种选择算子，所以人们提出了其他一些选择算子。下面介绍几种常用选择算子的操作方法。

3.3.1 比例选择

比例选择方法（Proportional Model）是一种回放式随机采样的方法。其基本思想是[4]：各个个体被选中的概率与其适应度大小成正比。由于是随机操作的原因，这种选择方法的选择误差比较大，有时甚至连适应度较高的个体也选择不上。

设群体大小为 M，个体 i 的适应度为 F_i，则个体 i 被选中的概率 p_{is} 为：

$$p_{is} = F_i \Big/ \sum_{i=1}^{M} F_i \qquad (i = 1, 2, \cdots, M) \tag{3-13}$$

由上式可见，适应度越高的个体被选中的概率也越大；反之，适应度越低的个体被选中的概率也越小。

比例选择方法的具体操作过程已在第二章中做过介绍，此处不再赘述。

3.3.2 最优保存策略

在遗传算法的运行过程中，通过对个体进行交叉、变异等遗传操作而不断地产生出新的个体。虽然随着群体的进化过程会产生出越来越多的优良个体，但由于选择、交叉、变异等遗传操作的随机性，它们也有可能破坏掉当前群体中适应度最好的个体。而这却不是我们所希望发生的，因为它会降低群体的平均适应度，并且对遗传算法的运行效率、收敛性都有不利的影响。所以，我们希望适应度最好的个体要尽可能地保留到下一代群体中。为达到这个目的，可以使用最优保存策略进化模型（Elitist Model）来进行优胜劣汰操作，即当前群体中适应度最高的个体不参与交叉运算和变异运算，而是用它来替换掉本代群体中经过

交叉、变异等遗传操作后所产生的适应度最低的个体[4]。

最优保存策略进化模型的具体操作过程是：

(1) 找出当前群体中适应度最高的个体和适应度最低的个体。

(2) 若当前群体中最佳个体的适应度比总的迄今为止的最好个体的适应度还要高，则以当前群体中的最佳个体作为新的迄今为止的最好个体。

(3) 用迄今为止的最好个体替换掉当前群体中的最差个体。

最优保存策略可视为选择操作的一部分。该策略的实施可保证迄今为止所得到的最优个体不会被交叉、变异等遗传运算所破坏，它是遗传算法收敛性的一个重要保证条件。但另一方面，它也容易使得某个局部最优个体不易被淘汰掉反面快速扩散，从而使得算法的全局搜索能力不强。所以该方法一般要与其他一些选择操作方法配合起来使用，方可有良好的效果。

另外，最优保存策略还可加以推广，即在每一代的进化过程中保留多个最优个体不参加交叉、变异等遗传运算，而直接将它们复制到下一代群体中。这种选择方法也称为稳态复制。

3.3.3 确定式采样选择

确定式采样选择方法（Deterministic Sampling）的基本思想是按照一种确定的方式来进行选择操作[17]。其具体操作过程是：

(1) 计算群体中各个个体在下一代群体中的期望生存数目 N_i：

$$N_i = M \cdot F_i / \sum_{i=1}^{M} F_i \qquad (i = 1, 2, \cdots, M)$$

(2) 用 N_i 的整数部分 $\lfloor N_i \rfloor$ 确定各个对应个体在下一代群体中的生存数目。其中 $\lfloor x \rfloor$ 表示取不大于 x 的最大的整数。由该步共可确定出下一代群体中的 $\sum_{i=1}^{M} \lfloor N_i \rfloor$ 个个体。

(3) 按照 N_i 的小数部分对个体进行降序排序，顺序取前 $M -$

$\sum_{i=1}^{M} \lfloor N_i \rfloor$ 个个体加入到下一代群体中。至此可完全确定出下一代群体中的 M 个个体。

这种选择操作方法可保证适应度较大的一些个体一定能够被保留在下一代群体中，并且操作也比较简单。

3.3.4 无回放随机选择

这种选择操作方法也叫做期望值选择方法(Expected Value Model)，它的基本思想是根据每个个体在下一代群体中的生存期望值来进行随机选择运算[17]。其具体操作过程是：

(1) 计算群体中每个个体在下一代群体中的生存期望数目 N_i：

$$N_i = M \cdot F_i \Big/ \sum_{i=1}^{M} F_i \qquad (i = 1, 2, \cdots, M)$$

(2) 若某一个体被选中参与交叉运算，则它在下一代中的生存期望数目减去 0.5，若某一个体未被选中参与交叉运算，则它在下一代中的生存期望数目减去 1.0。

(3) 随着选择过程的进行，若某一个体的生存期望数目小于 0 时，则该个体就不再有机会被选中。

这种选择操作方法能够降低一些选择误差，但操作不太方便。

3.3.5 无回放余数随机选择

无回放余数随机选择(Remainder Stochastic Sampling with Replacement)[17] 的具体操作过程是：

(1) 计算群体中每个个体在下一代群体中的生存期望数目 N_i：

$$N_i = M \cdot F_i \Big/ \sum_{i=1}^{M} F_i \qquad (i = 1, 2, \cdots, M)$$

(2) 取 N_i 的整数部分 $\lfloor N_i \rfloor$ 为对应个体在下一代群体中的生存数目。这样共可确定出下一代 M 个群体中的 $\sum_{i=1}^{M} \lfloor N_i \rfloor$ 个个体。

(3) 以 $F_i - \lfloor N_i \rfloor \cdot \sum_{i=1}^{M} F_i / M$ 为各个个体的新的适应度，用比例选择方法(赌盘选择方法) 来随机确定下一代群体中还未确定的 $M - \sum_{i=1}^{M} \lfloor N_i \rfloor$ 个个体。

这种选择操作方法可确保适应度比平均适应度大的一些个体一定能够被遗传到下一代群体中，所以它的选择误差比较小。

3.3.6 排序选择

在前面所介绍的一些选择操作方法中，其选择依据主要是各个个体适应度的具体数值，一般要求它取非负值，这就使得我们在选择操作之前必须先对一些负的适应度进行变换处理。而排序选择方法(Rank-based Model)[18]的主要着眼点是个体适应度之间的大小关系，对个体适应度是否取正值或负值以及个体适应度之间的数值差异程度并无特别要求。

排序选择方法的主要思想是：对群体中的所有个体按其适应度大小进行排序，基于这个排序来分配各个个体被选中的概率。其具体操作过程是：

(1) 对群体中的所有个体按其适应度大小进行降序排序。

(2) 根据具体求解问题，设计一个概率分配表，将各个概率值按上述排列次序分配给各个个体。

(3) 以各个个体所分配到的概率值作为其能够被遗传到下一代的概率，基于这些概率值用比例选择(赌盘选择) 的方法来产生下一代群体。

例如，表 3-2 所示为进行排序选择时所设计的一个概率分配表。由该表可以看出，各个个体被选中的概率只与其排列序号所对应的概率值有关，即只与个体适应度之间的大小次序有关，而与其适应度的具体数值无直接关系。

该方法的实施必须根据对所研究问题的分析和理解情况预先设计一个概率分配表，这个设计过程无一定规律可循。另一方面，虽然依据个体适应度之间的大小次序给各个个体分配了一个选中

表 3-2　排序选择时的概率分配表

个体排列序号	适应度	选择概率
1	108	0.25
2	90	0.19
3	88	0.17
4	55	0.15
5	51	0.10
6	10	0.08
7	− 10	0.03
8	− 50	0.03

概率，但由于具体选中哪一个个体仍是使用了随机性较强的比例选择方法，所以排序选择方法仍具有较大的选择误差。

3.3.7　随机联赛选择

随机联赛选择(Stochastic Tournament Model)[17] 也是一种基于个体适应度之间大小关系的选择方法。其基本思想是每次选取几个个体之中适应度最高的一个个体遗传到下一代群体中。在联赛选择操作中，只有个体适应度之间的大小比较运算，而无个体适应度之间的算术运算，所以它对个体适应度是取正值还是取负值无特别要求。

联赛选择中，每次进行适应度大小比较的个体数目称为联赛规模。一般情况下，联赛规模 N 的取值为 2。

联赛选择的具体操作过程是：

(1) 从群体中随机选取 N 个个体进行适应度大小的比较，将其中适应度最高的个体遗传到下一代群体中。

(2) 将上述过程重复 M 次，就可得到下一代群体中的 M 个个体。