第二章　基本遗传算法

2.1　基本遗传算法描述

基于对自然界中生物遗传与进化机理的模仿，针对不同的问题，很多学者设计了许多不同的编码方法来表示问题的可行解，开发出了许多种不同的遗传算子来模仿不同环境下的生物遗传特性。这样，由不同的编码方法和不同的遗传算子就构成了各种不同的遗传算法。但这些遗传算法都有共同的特点，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，来完成对问题最优解的自适应搜索过程。基于这个共同特点，Goldberg 总结出了一种统一的最基本的遗传算法——基本遗传算法（Simple Genetic Algorithms，简称 SGA）[5]。基本遗传算法只使用选择算子、交叉算子和变异算子这三种基本遗传算子，其遗传进化操作过程简单，容易理解，是其他一些遗传算法的雏形和基础，它不仅给各种遗传算法提供了一个基本框架，同时也具有一定的应用价值。

2.1.1　基本遗传算法的构成要素

（1）染色体编码方法。基本遗传算法使用固定长度的二进制符号串来表示群体中的个体，其等位基因是由二值符号集 {0，1} 所组成的。初始群体中各个个体的基因值可用均匀分布的随机数来生成。如：

$$X = 100111001000101101$$

就可表示一个个体，该个体的染色体长度是 $n = 18$。

（2）个体适应度评价。基本遗传算法按与个体适应度成正比

的概率来决定当前群体中每个个体遗传到下一代群体中的机会多少。为正确计算这个概率，这里要求所有个体的适应度必须为正数或零。这样，根据不同种类的问题，必须预先确定好由目标函数值到个体适应度之间的转换规则，特别是要预先确定好当目标函数值为负数时的处理方法。

(3) 遗传算子。基本遗传算法使用下述三种遗传算子：

●选择运算使用比例选择算子；

●交叉运算使用单点交叉算子；

●变异运算使用基本位变异算子或均匀变异算子。

(4) 基本遗传算法的运行参数。基本遗传算法有下述 4 个运行参数需要提前设定：

●M：群体大小，即群体中所含个体的数量，一般取为 20～100。

●T：遗传运算的终止进化代数，一般取为 100～500。

●p_c：交叉概率，一般取为 0.4～0.99。

●p_m：变异概率，一般取为 0.0001～0.1。

需要说明的是，这 4 个运行参数对遗传算法的求解结果和求解效率都有一定的影响，但目前尚无合理选择它们的理论依据。在遗传算法的实际应用中，往往需要经过多次试算后才能确定出这些参数合理的取值大小或取值范围。

2.1.2 基本遗传算法描述

下面我们给出基本遗传算法的伪代码描述。

2.1.3 基本遗传算法的形式化定义

基本遗传算法可定义为一个 8 元组：

$$\mathrm{SGA}=(C,\ E,\ P_0,\ M,\ \Phi,\ \Gamma,\ \Psi,\ T) \tag{2-1}$$

式中 C——个体的编码方法；

E——个体适应度评价函数；

P_0——初始群体；

M——群体大小；

Φ——选择算子；

Γ——交叉算子；

Ψ——变异算子；

T——遗传运算终止条件。

```
Procedure SGA
begin
    initialize P (0);
    t = 0;
    while (t≤T) do
        for i = 1 to M do
            Evaluate fitness of P (t);
        end for
        for i = 1 to M do
            Select operation to P (t);
        end for
        for i = 1 to M/2 do
            Crossover operation to P (t);
        end for
        for i = 1 to M do
            Mutation operation to P (t);
        end for
        for i = 1 to M do
            P (t + 1) = P (t)
        end for
        t = t + 1;
    end while
end
```

2.2 基本遗传算法的实现

根据上面对基本遗传算法构成要素的分析和算法描述，我们可以很方便地用计算机语言来实现这个基本遗传算法。附录 I 给出了其 C 语言源程序。现对具体实现过程中的问题作以下说明。

2.2.1 个体适应度评价

在遗传算法中，以个体适应度的大小来确定该个体被遗传到下一代群体中的概率。个体的适应度越大，该个体被遗传到下一代的概率也越大；反之，个体的适应度越小，该个体被遗传到下一代的概率也越小。基本遗传算法使用比例选择算子来确定群体中各个个体遗传到下一代群体中的数量。为正确计算不同情况下各个个体的遗传概率，要求所有个体的适应度必须为正数或零，不能是负数。

对于求目标函数最小值的优化问题，理论上只需简单地对其增加一个负号就可将其转化为求目标函数最大值的优化问题，即：

$$\min f(X) = \max(-f(X)) \tag{2-2}$$

当优化目标是求函数最大值，并且目标函数总取正值时，可以直接设定个体的适应度 $F(X)$ 就等于相应的目标函数值 $f(X)$，即：

$$F(X) = f(X) \tag{2-3}$$

但实际优化问题中的目标函数值有正也有负，优化目标有求函数最大值，也有求函数最小值，显然上面两式保证不了所有情况下个体的适应度都是非负数这个要求。所以必须寻求出一种通用且有效的由目标函数值到个体适应度之间的转换关系，由它来保证个体适应度总取非负值。

为满足适应度取非负值的要求，基本遗传算法一般采用下面两种方法之一将目标函数值 $f(X)$ 变换为个体的适应度 $F(X)$。

方法一：对于求目标函数最大值的优化问题，变换方法为：

$$F(X)=\begin{cases} f(X)+C_{\min}, & \text{if } f(X)+C_{\min}>0 \\ 0, & \text{if } f(X)+C_{\min}\leqslant 0 \end{cases} \tag{2-4}$$

式中，$C_{\min}$为一个适当地相对比较小的数，它可用下面几种方法之一来选取。

●预先指定的一个较小的数。

●进化到当前代为止的最小目标函数值。

●当前代或最近几代群体中的最小目标函数值。

方法二：对于求目标函数最小值的优化问题，变换方法为：

$$F(X)=\begin{cases} C_{\max}-f(X), & \text{if } f(X)<C_{\max} \\ 0, & \text{if } f(X)\geqslant C_{\max} \end{cases} \tag{2-5}$$

式中，$C_{\max}$为一个适当地相对比较大的数，它可用下面几种方法之一来选取。

●预先指定的一个较大的数。

●进化到当前代为止的最大目标函数值。

●当前代或最近几代群体中的最大目标函数值。

2.2.2 比例选择算子

选择算子或复制算子的作用是从当前代群体中选择出一些比较优良的个体，并将其复制到下一代群体中。最常用和最基本的选择算子是比例选择算子。所谓比例选择算子，是指个体被选中并遗传到下一代群体中的概率与该个体的适应度大小成正比。

比例选择实际上是一种有退还随机选择，也叫做赌盘(Roulette Wheel)选择，因为这种选择方式与赌博中的赌盘操作原理颇为相似。

如图 2-1 所示为一赌盘示意图。整个赌盘被分为大小不同的一些扇面，分别对应着价值各不相同的一些赌博物品。当旋转着的赌盘自然停下来时，其指针所指扇面上的物品就归赌博者所有。虽然赌盘的指针具体停止在哪一个扇面是无法预测的，但指

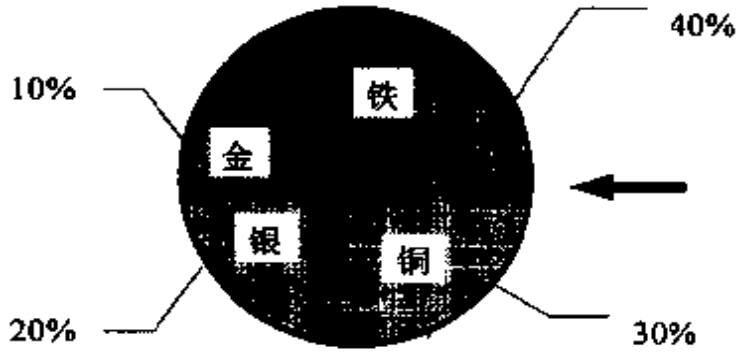


图 2-1　赌盘示意图

针指向各个扇面的概率却是可以估计的，它与各个扇面的圆心角大小成正比：圆心角越大，停在该扇面的可能性也越大；圆心角越小，停在该扇面的可能性也越小。与此类似，在遗传算法中，整个群体被各个个体所分割，各个个体的适应度在全部个体的适应度之和中所占比例也大小不一，这个比例值瓜分了整个赌盘盘面，它们也决定了各个个体被遗传到下一代群体中的概率。

比例选择算子的具体执行过程是：

(1) 先计算出群体中所有个体的适应度的总和。

(2) 其次计算出每个个体的相对适应度的大小，它即为各个个体被遗传到下一代群体中的概率。

(3) 最后再使用模拟赌盘操作（即 0 到 1 之间的随机数）来确定各个个体被选中的次数。

2.2.3　单点交叉算子

单点交叉算子是最常用和最基本的交叉操作算子。单点交叉算子的具体执行过程如下。

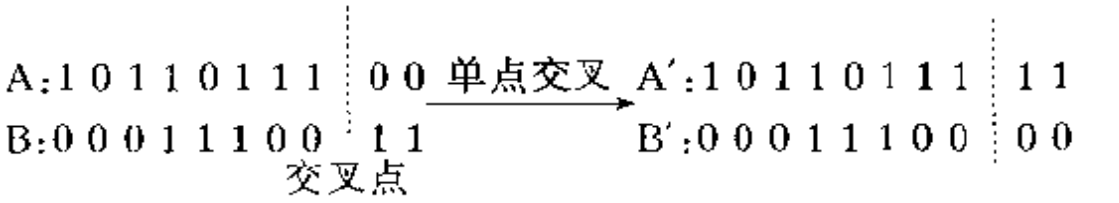
(1) 对群体中的个体进行两两随机配对。若群体大小为 M，则共有$\lfloor M/2 \rfloor$对相互配对的个体组。其中$\lfloor x \rfloor$表示不大于 x 的最大的整数。

(2) 对每一对相互配对的个体，随机设置某一基因座之后的位置为交叉点。若染色体的长度为 n，则共有 $(n-1)$ 个可能的

交叉点位置。

(3) 对每一对相互配对的个体，依设定的交叉概率 p_c 在其交叉点处相互交换两个个体的部分染色体，从而产生出两个新的个体。

单点交叉运算的示意如下所示：



2.2.4 基本位变异算子

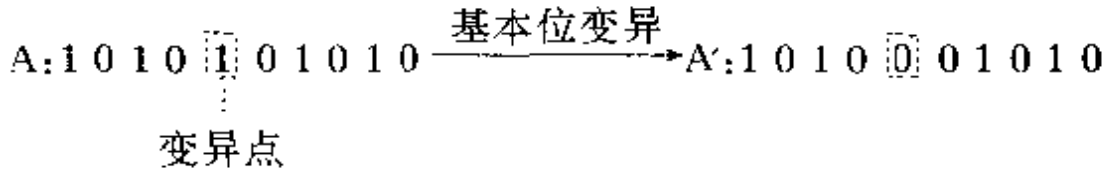
基本位变异算子是最简单和最基本的变异操作算子。对于基本遗传算法中用二进制编码符号串所表示的个体，若需要进行变异操作的某一基因座上的原有基因值为 0，则变异操作将该基因值变为 1；反之，若原有基因值为 1，则变异操作将其变为 0。

基本位变异算子的具体执行过程是：

(1) 对个体的每一个基因座，依变异概率 p_m 指定其为变异点。

(2) 对每一个指定的变异点，对其基因值做取反运算或用其他等位基因值来代替，从而产生出一个新的个体。

基本位变异运算的示意如下所示：



2.3 基本遗传算法应用举例

由前述我们可以知道，基本遗传算法是一个迭代过程，它模仿生物在自然环境中的遗传和进化机理，反复将选择算子、交叉算子、变异算子作用于群体，最终可得到问题的最优解或近似最优解。虽然算法的思想比较单纯，结构也比较简单，但它却也具

有一定的实用价值，能够解决一些复杂系统的优化计算问题。

2.3.1 遗传算法的应用步骤

遗传算法提供了一种求解复杂系统优化问题的通用框架，它不依赖于问题的领域和种类。对一个需要进行优化计算的实际应用问题，一般可按下述步骤来构造求解该问题的遗传算法。

第一步：确定决策变量及其各种约束条件，即确定出个体的表现型 X 和问题的解空间。

第二步：建立优化模型，即确定出目标函数的类型（是求目标函数的最大值还是求目标函数的最小值?）及其数学描述形式或量化方法。

第三步：确定表示可行解的染色体编码方法，也即确定出个体的基因型 X 及遗传算法的搜索空间。

第四步：确定解码方法，即确定出由个体基因型 X 到个体表现型 X 的对应关系或转换方法。

第五步：确定个体适应度的量化评价方法，即确定出由目标函数值 $f(X)$ 到个体适应度 $F(X)$ 的转换规则。

第六步：设计遗传算子，即确定出选择运算、交叉运算、变异运算等遗传算子的具体操作方法。

第七步：确定遗传算法的有关运行参数，即确定出遗传算法的 M、T、p_c、p_m 等参数。

由上述构造步骤可以看出，可行解的编码方法、遗传算子的设计是构造遗传算法时需要考虑的两个主要问题，也是设计遗传算法时的两个关键步骤。对不同的优化问题需要使用不同的编码方法和不同操作的遗传算子，它们与所求解的具体问题密切相关，因而对所求解问题的理解程度是遗传算法应用成功与否的关键。

图 2-2 所示为遗传算法的主要构造过程示意图。

2.3.2 基本遗传算法在函数优化中的应用举例

【例】Rosenbrock 函数的全局最大值计算。

$$\max \quad f(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2 \tag{2-6}$$

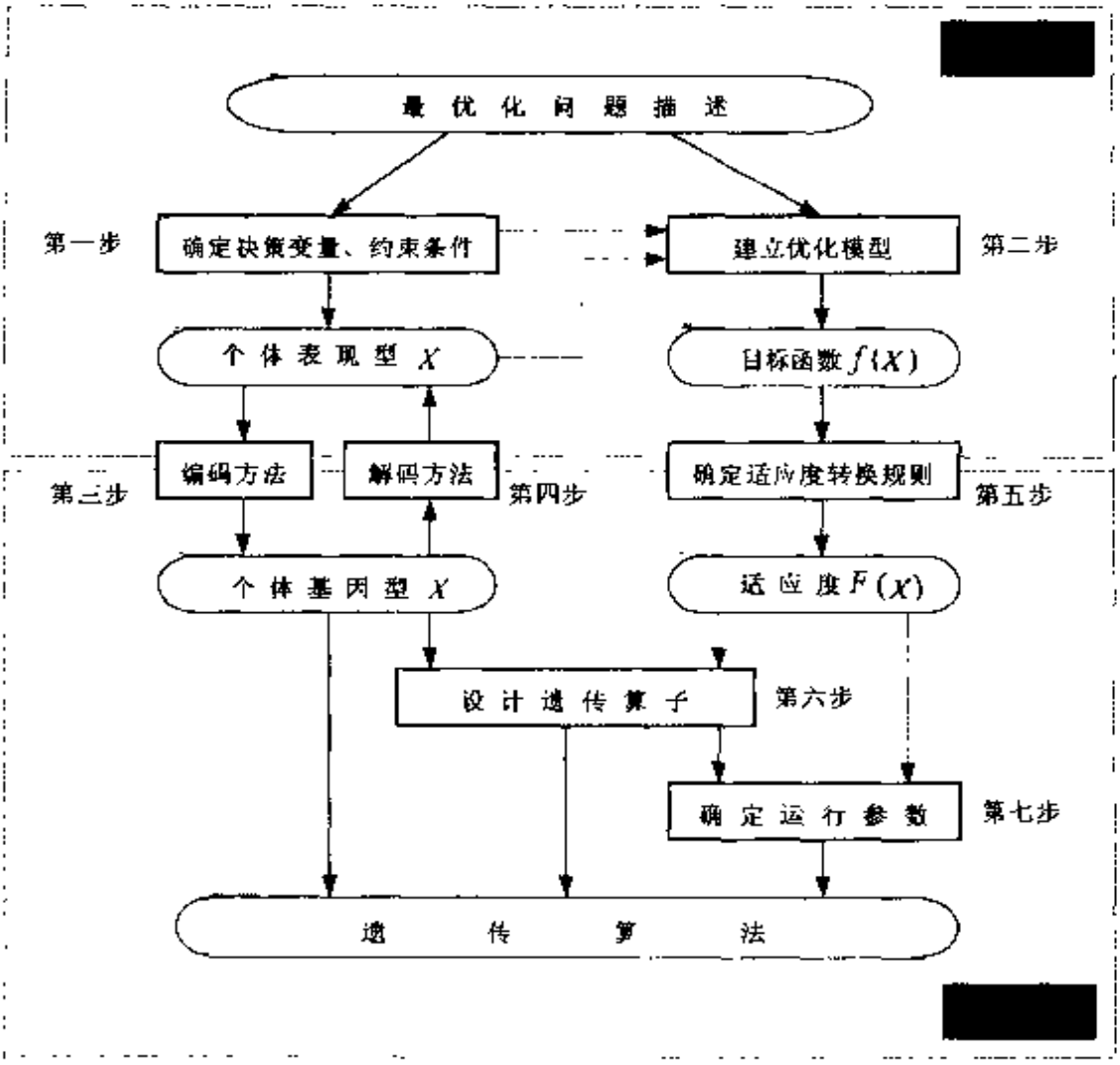


图 2-2　遗传算法的主要构造过程示意图

$$\text{s.t.} \quad -2.048 \leqslant x_i \leqslant 2.048 \quad (i=1,\ 2) \tag{2-7}$$

如图 2-3 所示，该函数有两个局部极大点，分别是 f（2.048, −2.048）= 3897.7342 和 f（−2.048, −2.048）= 3905.9262，其中后者为全局最大点。

下面介绍求解该问题的遗传算法的构造过程。

第一步：确定决策变量和约束条件。

式（2-7）已给出了该问题的决策变量及其约束条件。

第二步：建立优化模型。

式（2-6）已给出了该问题的数学模型。

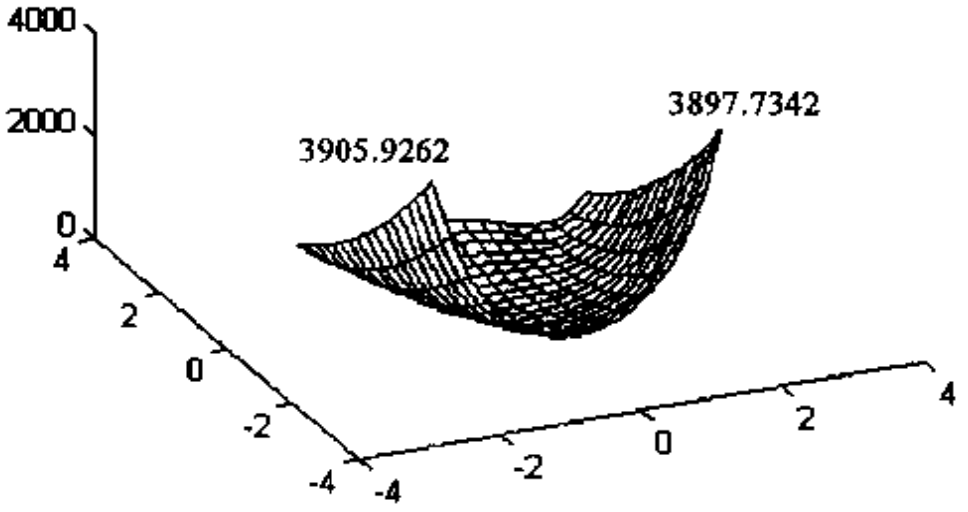


图 2-3　Rosenbrock 函数

第三步：确定编码方法。

用长度为 10 位的二进制编码串来分别表示二个决策变量 x_1、x_2。10 位二进制编码串可以表示从 0 到 1023 之间的 1024 个不同的数，故将 x_1、x_2 的定义域离散化为 1023 个均等的区域，包括两个端点在内共有 1024 个不同的离散点。从离散点 −2.048到离散点 2.048，依次让它们分别对应于从 0000000000 (0) 到 1111111111 (1023) 之间的二进制编码。再将分别表示 x_1、x_2 的二个 10 位长的二进制编码串连接在一起，组成一个 20 位长的二进制编码串，它就构成了这个函数优化问题的染色体编码方法。使用这种编码方法，解空间和遗传算法的搜索空间具有一一对应的关系。例如

$$X：\underbrace{0000110111}_{x_1}\ \underbrace{1101110001}_{x_2}$$

就表示一个个体的基因型，其中前 10 位表示 x_1，后 10 位表示 x_2。

第四步：确定解码方法。

解码时需先将 20 位长的二进制编码串切断为二个 10 位长的

二进制编码串，然后分别将它们转换为对应的十进制整数代码，分别记为 y_1 和 y_2。依据前述个体编码方法和对定义域的离散化方法可知，将代码 y_i 转换为变量 x_i 的解码公式为：

$$x_i = 4.096 \times \frac{y_i}{1023} - 2.048 \qquad (i = 1,\ 2) \qquad (2\text{-}8)$$

例如，对于前述个体

$$X：0000110111101110001$$

它由这样的两个代码所组成：

$$y_1 = 55$$

$$y_2 = 881$$

经过式（2-7）的解码处理后，可得到：

$$x_1 = -1.828$$

$$x_2 = 1.476$$

第五步：确定个体评价方法。

由式（2-6）可知，Rosenbrock 函数的值域总是非负的，并且优化目标是求函数的最大值，故这里可将个体的适应度直接取为对应的目标函数值，并且不再对它作其他变换处理，即有：

$$F\ (X) = f\ (x_1,\ x_2) \qquad (2\text{-}9)$$

第六步：设计遗传算子。

选择运算使用比例选择算子；

交叉运算使用单点交叉算子；

变异运算使用基本位变异算子。

第七步：确定遗传算法的运行参数。

对于本例，设定基本遗传算法的运行参数如下：

群体大小：$M = 80$

终止代数：$T = 200$

交叉概率：$p_c = 0.6$

变异概率：$p_m = 0.001$

通过上述七个步骤就可构成用于 Rosenbrock 函数优化计算

的基本遗传算法，图 2-4 所示为其进化过程示例及运行结果。图中横轴表示进化代数，纵轴表示适应度（也是目标函数值），图中的两条曲线分别为各代群体中个体适应度的最大值和平均值。

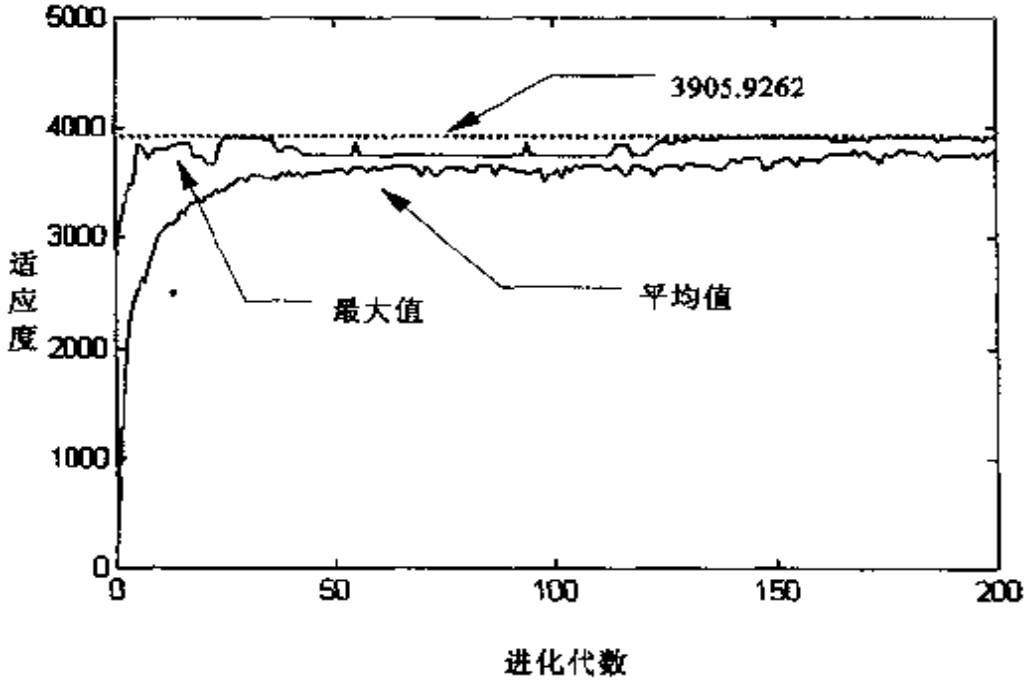
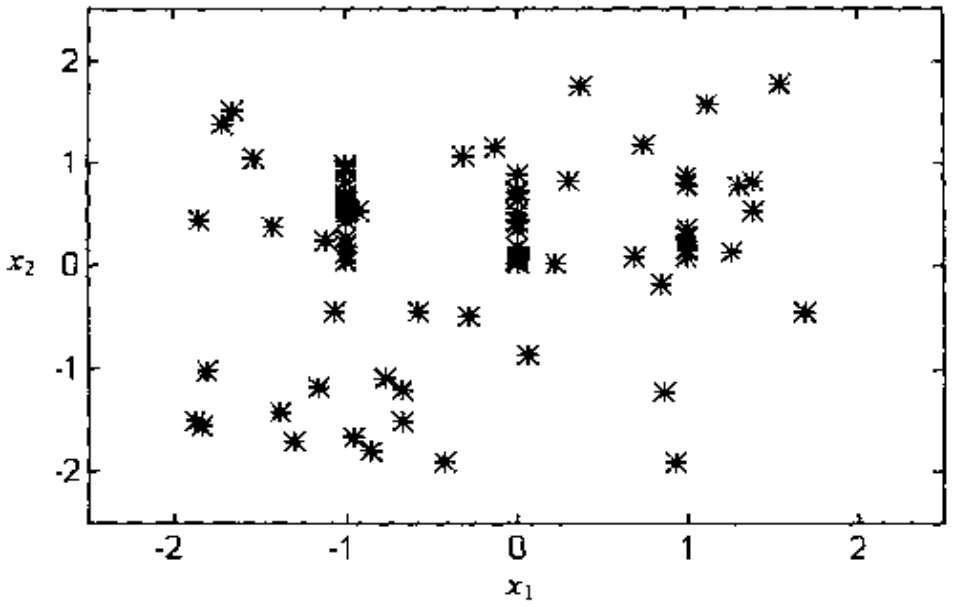


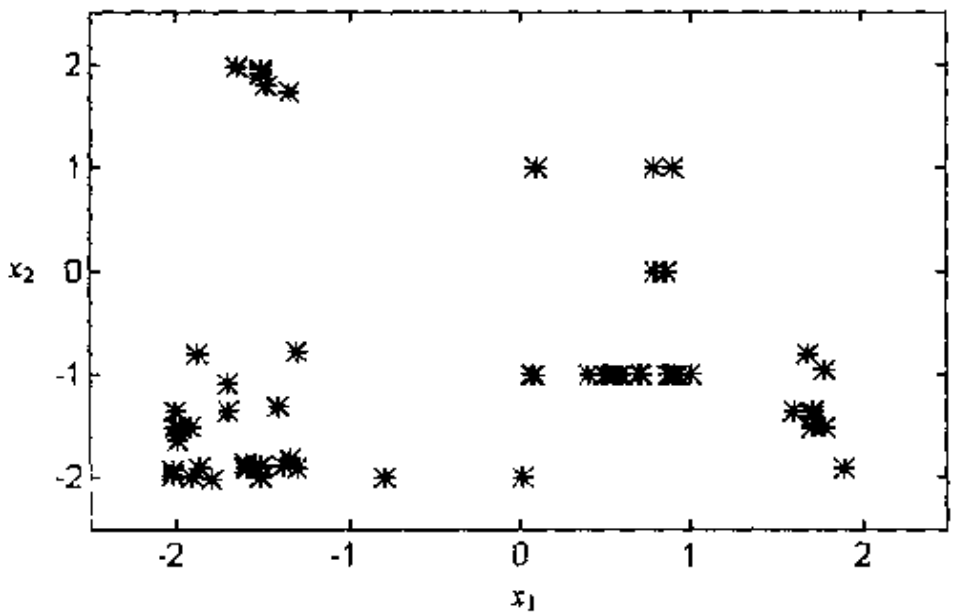
图 2-4　进化过程及运行结果

从图 2-4 中可以看出，在解的进化过程中，群体中个体适应度的最大值和平均值虽然有上下波动的情况，但总的来说却是呈现出一种上升的趋势。

图 2-5 所示分别为初始群体、第 5 代群体、第 10 代群体和第 100 代群体中个体的分布情况。在图 2-5（a）中各个个体分布得比较均匀；图 2-5（b）中大量的个体分布在最优点和次最优点附近；从图 2-5（c）中可以看出，次最优点也被淘汰；而从图 2-5（d）中可以看出，个体更加集中在最优点附近。由该组图形我们可以看出，随着进化过程的进行，群体中适应度较低的一些个体被逐渐淘汰掉，而适应度较高的一些个体会越来越多，并且它们都集中在所求问题的最优点附近，从而最终就可搜索到问题的最优解。



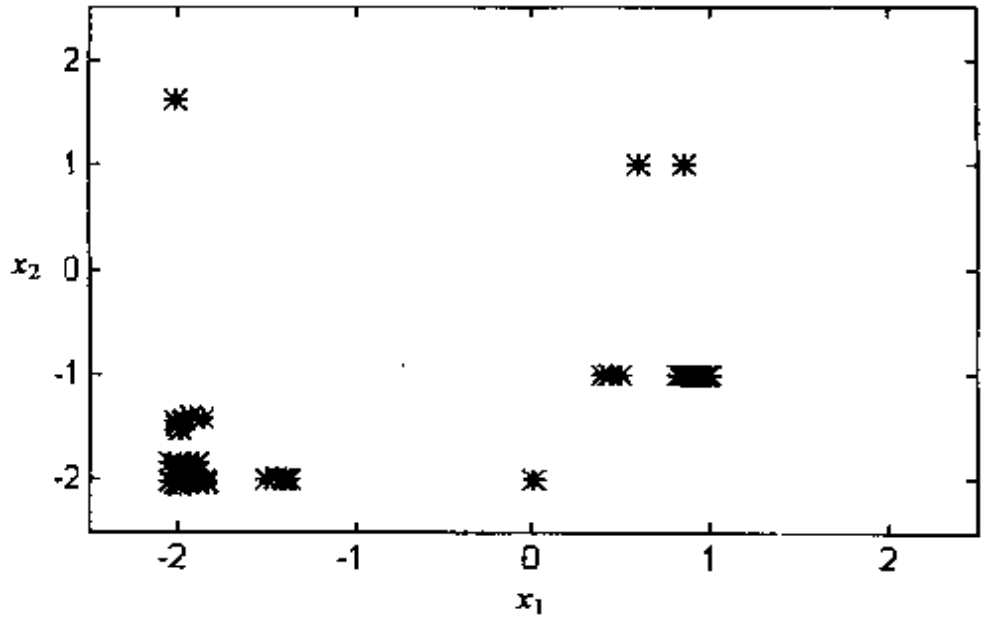
(a)



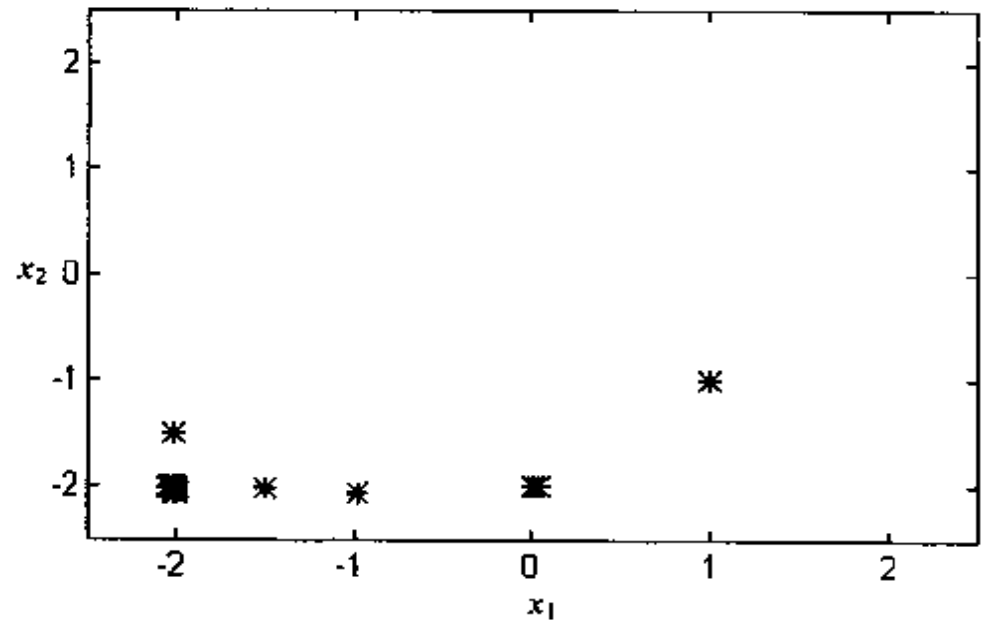
(b)

图 2-5　遗传算法搜索

(a) 初始群体；(b) 第 5 代群体；



(*c*)



(*d*)

过程中个体分布图

(*c*) 第 10 代群体；(*d*) 第 100 代群体。