考虑下面的优化问题：

$$\max f(x_1,x_2) = 21.5 + x_1\sin(4\pi x_1) + x_2\sin(20\pi x_2),$$
$$-3.0 \leqslant x_1 \leqslant 12.1,\quad 4.1 \leqslant x_2 \leqslant 5.8.$$

该目标函数有许多局部最优解，其三维图形如图 2.1 所示.

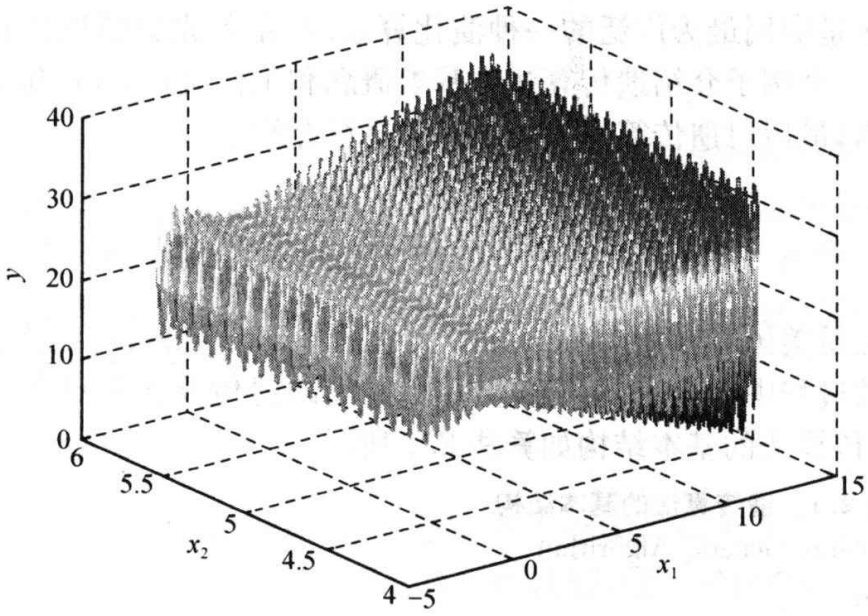

图 2.1　目标函数的三维图形

1. 个体的编码

问题的可能解为实数对(x_1,x_2)的形式，将问题的可能解编码为二进制位串. 为此，首先将每个变量编码为二进制位串，然后将表示各个变量的二进制数串连接起来便得到问题可能解的一种表示. 表示每个变量的二进制位串的长度取决于变量的定义域和所要求的精度.

设 $a_j \leqslant x_j \leqslant b_j$，所要求的精度为小数点后 t 位. 这要求将区间$[a_j,b_j]$划分为至少$(b_j-a_j)10^t$ 份. 假设表示变量 x_j 的位串的长度用 l_j 表示，则 l_j 可取为满足下列不等式的最小正整数 m：

$$(b_j-a_j)10^t \leqslant 2^m - 1,$$

即有

$$2^{l_j-1} - 1 < (b_j-a_j)10^t \leqslant 2^{l_j} - 1.$$

将 x_j 的二进制表示转换为十进制表示可按下式计算：

$$x_j = a_j + \text{decimal}(\text{substring}_j) \times \frac{b_j-a_j}{2^{l_j}-1},$$

其中，$\text{decimal}(\text{substring}_j)$表示变量 x_j 的二进制子串 substring_j 对应的十进制数.

对于上面的优化问题，假定所要求的精度为小数点后 4 位，那么由

$$(12.1-(-3.0))\times 10000 = 151000,$$

$$2^{17}-1 < 151000 \leqslant 2^{18}-1$$

知表示变量 x_1 的二进制位串的长度为 $l_1=18$. 由

$$(5.8-4.1)\times 10000 = 17000,$$

$$2^{14}-1 < 17000 \leqslant 2^{15}-1$$

知表示变量 x_2 的二进制位串的长度为 $l_2=15$.

计算出表示变量 x_1 和 x_2 的二进制位串长度 l_1 和 l_2 后，便可以得到表示问题可能解(x_1,x_2)的二进制位串的长度 $l=l_1+l_2=18+15=33$. 表示问题解(x_1,x_2)的二进制位串如图 2.2 所示，其中，前 18 位表示变量 x_1，后 15 位表示变量 x_2.

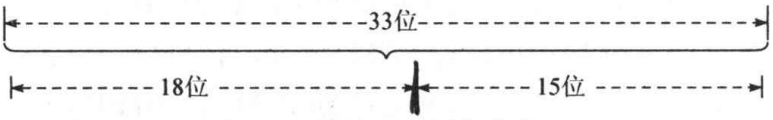

图 2.2 解的二进制位串表示

例如，给定下列 33 位二进制位串：

$$010001001011010000111110010100010,$$

那么前 18 位所表示变量 x_1 的值为

$$x_1 = -3.0 + \text{decimal}(010001001011010000)\times\frac{12.1-(-3.0)}{2^{18}-1}$$

$$= -3.0 + 70352\times\frac{15.1}{2^{18}-1} = -3.0+4.052426 = 1.052426,$$

而后 15 位所表示变量 x_2 的值为

$$x_2 = 4.1 + \text{decimal}(111110010100010)\times\frac{5.8-4.1}{2^{15}-1}$$

$$= 4.1 + 31906\times\frac{1.7}{2^{15}-1} = 4.1+1.655330 = 5.755330.$$

所以，二进制位串 010001001011010000111110010100010 所表示问题的可能解为 $(x_1,x_2)=(1.052426,5.755330)$.

2. *产生初始种群*

假定初始种群的规模为 $N=20$. 随机产生初始种群如下：

$$v_1 = (100110100000001111111010011011111),$$

$$v_2 = (111000100100110111001010100011010),$$

$$v_3 = (000010000011001000001010111011101),$$

$$v_4 = (100011000101101001111000001110010),$$

$$v_5 = (000111011001010011010111111000101),$$

$$v_6 = (000101000010010101001010111111011),$$

$$v_7 = (00100010000011010111011011111011),$$
$$v_8 = (10000110000111010001011010110011 1),$$
$$v_9 = (01000000010110001011000000111 1100),$$
$$v_{10} = (00000111100011000001101000011 1011),$$
$$v_{11} = (01100111111011010110000110111 1000),$$
$$v_{12} = (11010001011110110100010101000 0000),$$
$$v_{13} = (11101111101000100011000000100 0110),$$
$$v_{14} = (01001001100000101010011110010 1001),$$
$$v_{15} = (11101110110111000010001111101 1110),$$
$$v_{16} = (11001111000001111110000110100 1011),$$
$$v_{17} = (01101011111100111101000110111 1101),$$
$$v_{18} = (01110100000000111010011111010 1101),$$
$$v_{19} = (00010101001111111111000011000 1100),$$
$$v_{20} = (10111001011001111001100010111 1110).$$

3. 计算适应值

本例是一个极大化问题，直接取目标函数 $f(x_1,x_2)$作为适应函数 $\mathrm{eval}(x_1,x_2)$.

计算一个染色体的适应值的过程由下面两步组成：

(1) 将该染色体转换为所表示的问题可能解 $x=(x_1,x_2)$；

(2) 计算个体 $x=(x_1,x_2)$的适应值 $\mathrm{eval}(x_1,x_2)$.

上述种群中染色体的适应值分别如下：

$$\mathrm{eval}(v_1) = f(6.084492,5.652242) = 26.019600,$$
$$\mathrm{eval}(v_2) = f(10.348434,4.380264) = 7.580015,$$
$$\mathrm{eval}(v_3) = f(-2.516603,4.390381) = 19.526329,$$
$$\mathrm{eval}(v_4) = f(5.278638,5.593460) = 17.406725,$$
$$\mathrm{eval}(v_5) = f(-1.255173,4.734458) = 25.341160,$$
$$\mathrm{eval}(v_6) = f(-1.811725,4.391937) = 18.100417,$$
$$\mathrm{eval}(v_7) = f(-0.991471,5.680258) = 16.020812,$$
$$\mathrm{eval}(v_8) = f(4.910618,4.703018) = 17.959701,$$
$$\mathrm{eval}(v_9) = f(0.795406,5.381472) = 16.127799,$$
$$\mathrm{eval}(v_{10}) = f(-2.554851,4.793707) = 21.278435,$$
$$\mathrm{eval}(v_{11}) = f(3.130078,4.996097) = 23.410669,$$
$$\mathrm{eval}(v_{12}) = f(9.356179,4.239457) = 15.011619,$$
$$\mathrm{eval}(v_{13}) = f(11.134646,5.378671) = 27.316702,$$

$$\text{eval}(v_{14}) = f(1.335944, 5.151378) = 19.876294,$$
$$\text{eval}(v_{15}) = f(11.089025, 5.054515) = 30.060205,$$
$$\text{eval}(v_{16}) = f(9.211598, 4.993762) = 23.867227,$$
$$\text{eval}(v_{17}) = f(3.367514, 4.571343) = 13.696165,$$
$$\text{eval}(v_{18}) = f(3.843020, 5.158226) = 15.414128,$$
$$\text{eval}(v_{19}) = f(-1.746635, 5.395584) = 20.095903,$$
$$\text{eval}(v_{20}) = f(7.935998, 4.757338) = 13.666916.$$

从染色体的适应值可以看出染色体 v_{15} 是最好的，v_2 是最差的.

4. 父体选择

这一过程采用轮盘赌选择(roulette wheel selection). 轮盘赌选择是遗传算法中使用最多的选择策略之一.

轮盘赌选择模拟博彩游戏中的轮盘赌. 一个轮盘被划分为 N 个扇形，每个扇形表示种群中的一个染色体，而每个扇形的面积与它所表示的染色体的适应值成正比，如图 2.3 所示. 为了选择种群中的个体，设想有一个指针指向轮盘，转动轮盘，当轮盘停止后，指针所指向的染色体被选择. 因为一个染色体的适应值越大表示该染色体的扇形面积就越大，因此它被选择的可能性也就越大.

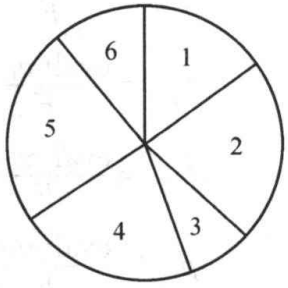

图 2.3 表示 6 个染色体的轮盘

轮盘赌选择可以如下实现：

(1) 计算种群中所有染色体适应值之和，

$$F = \sum_{k=1}^{N} \text{eval}(v_k);$$

(2) 计算每个染色体的选择概率，

$$p_k = \frac{\text{eval}(v_k)}{F}, \quad k = 1, 2, \cdots, N;$$

(3) 计算每个染色体的累计概率，

$$q_k = \sum_{j=1}^{k} p_j, \quad k = 1, 2, \cdots, N;$$

(4) 转动轮盘 N 次，从中选出 N 个染色体.

选择过程可如下实现：

用[0,1]中的一个随机数 r 来模拟转动一次轮盘，轮盘停止转动后指针所指向的位置. 若 $r \leqslant q_1$，这说明指针指向第一个扇形，这时选择第一个染色体 v_1，一般若 $q_{k-1} < r \leqslant q_k$，这说明指针指向第 k 个扇形，这时选择第 k 个染色体 v_k.

上述种群中染色体的适应值之和为

$$F=\sum_{k=1}^{20}\text{eval}(v_k)=387.776822.$$

种群中各染色体的选择概率如下：

$$p_1=\frac{\text{eval}(v_1)}{F}=0.067099,\quad p_2=\frac{\text{eval}(v_2)}{F}=0.019547,$$

$$p_3=\frac{\text{eval}(v_3)}{F}=0.050355,\quad p_4=\frac{\text{eval}(v_4)}{F}=0.044889,$$

$$p_5=\frac{\text{eval}(v_5)}{F}=0.065350,\quad p_6=\frac{\text{eval}(v_6)}{F}=0.046677,$$

$$p_7=\frac{\text{eval}(v_7)}{F}=0.041315,\quad p_8=\frac{\text{eval}(v_8)}{F}=0.046315,$$

$$p_9=\frac{\text{eval}(v_9)}{F}=0.041590,\quad p_{10}=\frac{\text{eval}(v_{10})}{F}=0.054873,$$

$$p_{11}=\frac{\text{eval}(v_{11})}{F}=0.060372,\quad p_{12}=\frac{\text{eval}(v_{12})}{F}=0.038712,$$

$$p_{13}=\frac{\text{eval}(v_{13})}{F}=0.070444,\quad p_{14}=\frac{\text{eval}(v_{14})}{F}=0.051257,$$

$$p_{15}=\frac{\text{eval}(v_{15})}{F}=0.077519,\quad p_{16}=\frac{\text{eval}(v_{16})}{F}=0.061549,$$

$$p_{17}=\frac{\text{eval}(v_{17})}{F}=0.035320,\quad p_{18}=\frac{\text{eval}(v_{18})}{F}=0.039750,$$

$$p_{19}=\frac{\text{eval}(v_{19})}{F}=0.051823,\quad p_{20}=\frac{\text{eval}(v_{20})}{F}=0.035244.$$

种群中各染色体的累计概率如下：

$$q_1=0.067099,\quad q_2=0.086647,\quad q_3=0.137001,\quad q_4=0.181890,$$
$$q_5=0.247240,\quad q_6=0.293917,\quad q_7=0.335232,\quad q_8=0.381546,$$
$$q_9=0.423137,\quad q_{10}=0.478009,\quad q_{11}=0.538381,\quad q_{12}=0.577093,$$
$$q_{13}=0.647537,\quad q_{14}=0.698794,\quad q_{15}=0.776314,\quad q_{16}=0.837863,$$
$$q_{17}=0.873182,\quad q_{18}=0.912932,\quad q_{19}=0.964756,\quad q_{20}=1.000000.$$

现在转动圆盘20次，每次选择一个染色体. 假定所产生[0,1]中的20个随机数分别如下：

0.513870，0.175741，0.308652，0.534534，0.947628，
0.171736，0.702231，0.226431，0.494773，0.424720，
0.703899，0.389647，0.277226，0.368071，0.983437，
0.005398，0.765682，0.646473，0.767139，0.780237.

第一个随机数 $r=0.513870$ 比 q_{10} 大，但比 q_{11} 小，故所选择的第一个染色体为 v_{11}；第二个随机数 $r=0.175741$ 比 q_3 大，但比 q_4 小，故所选择的第二个染色体为 v_4；如此进行下去，所选择的 20 个染色体如下：

$$
\begin{aligned}
v'_1 &= (01100111111011010110000110111000)(v_{11}),\\
v'_2 &= (10001100010110100111100000111010)(v_4),\\
v'_3 &= (00100010000011010111101101111011)(v_7),\\
v'_4 &= (01100111111011010110000110111000)(v_{11}),\\
v'_5 &= (00010101001111111111000011000100)(v_{19}),\\
v'_6 &= (10001100010110100111100000111010)(v_4),\\
v'_7 &= (11101110110111000010001111011110)(v_{15}),\\
v'_8 &= (00011101100101001101011111000101)(v_5),\\
v'_9 &= (01100111111011010110000110111000)(v_{11}),\\
v'_{10} &= (00001000001100100000101011011101)(v_3),\\
v'_{11} &= (11101110110111000010001111011110)(v_{15}),\\
v'_{12} &= (01000000010110001011000000111100)(v_9),\\
v'_{13} &= (00010100001001010100101011111011)(v_6),\\
v'_{14} &= (10000110000111010001011010110011)(v_8),\\
v'_{15} &= (10111001011001111001100010111110)(v_{20}),\\
v'_{16} &= (10011010000000111111101001101111)(v_1),\\
v'_{17} &= (00000111100011000001101000011011)(v_{10}),\\
v'_{18} &= (11101111101000100011000000100011 0)(v_{13}),\\
v'_{19} &= (11101110110111000010001111011110)(v_{15}),\\
v'_{20} &= (11001111000001111110000110100 1011)(v_{16}).
\end{aligned}
$$

轮盘赌选择如算法 2.2 所示.

算法 2.2 轮盘赌选择

```
Procedure Roulette-Wheel-Selection
begin
 for i←1 to N do
  r←random;
  j←1;
  while (r>q_j)do j←j+1;
  选择第 j 个染色体;
  end while
 end for
end
```

其中，random 生成[0,1]上的一个随机数.

5. 遗传算子

遗传算子有两种：杂交算子和变异算子.

(1) 杂交算子. 使用单点杂交，该方法对两个父体进行杂交，杂交后产生两个后代个体.

单点杂交过程如下：

设二进制位串的长度为 L，首先随机地产生一个整数 pos 作为杂交点的位置，pos$\in[1,L-1]$，然后将两个父体在该杂交点右边的子串进行交换，产生两个后代个体.

单点杂交的示意图如图 2.4 所示.

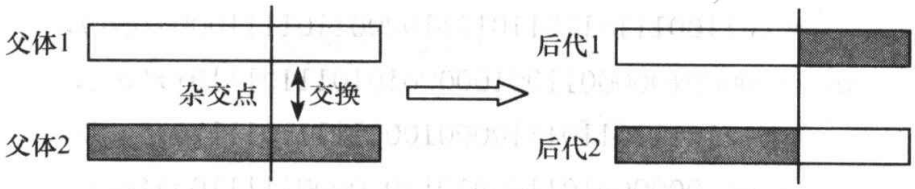

图 2.4　单点杂交示意图

例如，给定两个父体如下：

$$v_1 = (10011011010010110 \mid 1000000010111001),$$
$$v_2 = (00101101010000110 \mid 0010110011001100).$$

假设杂交点的位置为 17，那么交换两个父体第 17 个基因右边的子串后，所得到的两个后代如下：

$$v_1' = (10011011010010110 \mid 0010110011001100),$$
$$v_2' = (00101101010000110 \mid 1000000010111001).$$

杂交算子的应用过程如下：

从上面产生的父体中，按一定的概率 p_c 从中挑选出若干个染色体进行杂交. 首先从[0,1]中产生 N 个随机数 $r_k(k=1,2,\cdots,N)$，若 $r_k<p_c$，则挑选父体中的第 k 个染色体. 若这样挑选出来的染色体是奇数个，则或从父体中随机地再挑选一个染色体，或从已挑选的染色体中删除一个. 这样共挑选出来偶数个染色体，再将这偶数个染色体随机地两两配对进行杂交.

对本例而言，假定 $p_c=0.25$，所产生的 20 个随机数分别如下：

0.822951，　0.151932，　0.625477，　0.314685，　0.346901，
0.917204，　0.519760，　0.401154，　0.606758，　0.785402，
0.031523，　0.869921，　0.166525，　0.674520，　0.758400，
0.581893，　0.389248，　0.200232，　0.355635，　0.826927.

这意味着 v_2'，v_{11}'，v_{13}'，v_{18}'被选择进行杂交. 随机地对这 4 个染色体配对，如 v_2'，v_{11}'为一对，v_{13}'，v_{18}'为一对.

首先对 v_2'，v_{11}'进行杂交，随机地生成[1,32]中的一个整数 pos，设 pos=9，那么交换 v_2'和 v_{11}'第 9 个基因的右边部分，便得到下面两个后代：

$$v_2''=(100011000\mid 101110000100011111011110),$$
$$v_{11}''=(111011101\mid 101101001111000001110010).$$

然后对 v_{13}'，v_{18}'进行杂交，随机地生成[1,32]中的一个整数 pos，设 pos=20，那么交换 v_{13}'和 v_{18}'第 20 个基因的右边部分，便得到下面两个后代：

$$v_{13}''=(00010100001001010100\mid 0000001000110),$$
$$v_{18}''=(11101111101000100011\mid 1010111111011).$$

经过杂交算子的作用后，便得到当前种群如下：

$$v_1'=(01100111111011010110000110111 1000),$$
$$v_2''=(10001100010111000010001111101 1110),$$
$$v_3'=(00100010000011010111101101111 1011),$$
$$v_4'=(01100111111011010110000110111 1000),$$
$$v_5'=(00010101001111111111000011000 1100),$$
$$v_6'=(10001100010110100111100000111 0010),$$
$$v_7'=(11101110110111000010001111101 1110),$$
$$v_8'=(00011101100101001101011111100 0101),$$
$$v_9'=(01100111111011010110000110111 1000),$$
$$v_{10}'=(00001000001100100000101011101 1101),$$
$$v_{11}''=(11101110110110100111100000111 0010),$$
$$v_{12}'=(01000000010110001011000000111 1100),$$
$$v_{13}''=(00010100001001010100000000100 0110),$$
$$v_{14}'=(10000110000111010001011010110 0111),$$
$$v_{15}'=(10111001011001111001100010111 1110),$$
$$v_{16}'=(10011010000000111111101001101 1111),$$
$$v_{17}'=(00000111100011000001101000011 1011),$$
$$v_{18}''=(11101111101000100011101011111 1011),$$
$$v_{19}'=(11101110110111000010001111101 1110),$$
$$v_{20}'=(11001111000001111110000110100 1011).$$

(2) 变异算子. 变异算子的目的在于引入种群中染色体的多样性，防止算法的过早收敛.

变异算子以某一预先指定的概率 p_m对种群中染色体的每个基因进行变异. 当

染色体的某一基因被选择进行变异时，若该位为1，则变为0，否则变为1.

例如，给定下列染色体 v_1：

$$v_1 = (10011011010010110101000001011100 1).$$

若选择 v_1 的第18个基因进行变异，因为该位基因为1，所以将其变为0，得到一个新的染色体 v_1'，

$$v_1' = (10011011010010110001000001011100 1).$$

概率 p_m 是期望改变的种群中染色体的基因个数与基因总数的百分比. 例如，对本例而言，种群中的基因总数为 $L \times N = 33 \times 20 = 660$，若 $p_m = 0.01$，则每一代平均有6.6个基因发生改变.

在遗传算法中应用变异算子过程如下：

对种群中的每一个染色体的每一基因，产生一个随机数 r，若 $r < p_m$，那么该基因进行变异，否则不进行变异.

设变异概率 $p_m = 0.01$，所产生的660个随机数中有5个比0.01小. 假设这5个个体如表2.1所示.

表 2.1

随机数序号	随机数	染色体号	染色体中基因的位置
112	0.000213	4	13
349	0.009945	11	19
418	0.008809	13	22
429	0.005425	13	33
602	0.002836	19	8

经变异算子的作用后，得到新一代种群如下：

$$v_1 = (01100111111011010110000110111 1000),$$
$$v_2 = (10001100010111000010001111101 1110),$$
$$v_3 = (00100010000011010111101101111 1011),$$
$$v_4 = (01100111111001010110000110111 1000),$$
$$v_5 = (00010101001111111111000011000 1100),$$
$$v_6 = (10001100010110100111100000111 0010),$$
$$v_7 = (11101110110111000010001111101 1110),$$
$$v_8 = (00011101100101001101011111100 0101),$$
$$v_9 = (01100111111011010110000110111 1000),$$
$$v_{10} = (00001000001100100000101011101 1101),$$
$$v_{11} = (11101110110110100101100000111 0010),$$
$$v_{12} = (01000000010110001011000000111 1100),$$

$$v_{13}=(00010100001001010100010000 1000111),$$
$$v_{14}=(10000110000111010001011010 1100111),$$
$$v_{15}=(10111001011001111001100010 1111110),$$
$$v_{16}=(10011010000000111111101001 1011111),$$
$$v_{17}=(00000111100011000001101000 0111011),$$
$$v_{18}=(11101111101000100011101011 1111011),$$
$$v_{19}=(11101111110111000010001111 1011110),$$
$$v_{20}=(11001111000001111110000110 1001011).$$

新一代种群中个体的适应值如下：

$$\text{eval}(v_1)=f(3.130078,4.996097)=23.410669,$$
$$\text{eval}(v_2)=f(5.279042,5.054515)=18.201083,$$
$$\text{eval}(v_3)=f(-0.991471,5.680258)=16.020812,$$
$$\text{eval}(v_4)=f(3.128235,4.996097)=23.412613,$$
$$\text{eval}(v_5)=f(-1.746635,5.395584)=20.095903,$$
$$\text{eval}(v_6)=f(5.278638,5.593460)=17.406725,$$
$$\text{eval}(v_7)=f(11.089025,5.054515)=30.060205,$$
$$\text{eval}(v_8)=f(-1.255173,4.734458)=25.341160,$$
$$\text{eval}(v_9)=f(3.130078,4.996097)=23.410669,$$
$$\text{eval}(v_{10})=f(-2.516603,4.390381)=19.526329,$$
$$\text{eval}(v_{11})=f(11.088621,4.743434)=33.351874,$$
$$\text{eval}(v_{12})=f(0.795406,5.381472)=16.127799,$$
$$\text{eval}(v_{13})=f(-1.811725,4.209937)=22.692462,$$
$$\text{eval}(v_{14})=f(4.910618,4.703018)=17.959701,$$
$$\text{eval}(v_{15})=f(7.935998,4.757338)=13.666916,$$
$$\text{eval}(v_{16})=f(6.084492,5.652242)=26.019600,$$
$$\text{eval}(v_{17})=f(-2.554851,4.793707)=21.278435,$$
$$\text{eval}(v_{18})=f(11.134646,5.666976)=27.591064,$$
$$\text{eval}(v_{19})=f(11.059532,5.054515)=27.608441,$$
$$\text{eval}(v_{20})=f(9.211598,4.993762)=23.867227.$$

新一代种群中个体的适应值之和为 447.049688，比初始种群中个体适应值之和 387.776822 有了很大的提高，而且新一代中个体的最大适应值 33.351874 也要比初始种群中个体的最大适应值 30.060205 要大.

至此，完成了种群的一次演化. 重复上述过程若干代，如 1000 代之后，可以得到下述种群：

$$v_1 = (11101111011001101110010101011011),$$
$$v_2 = (11100110011000010001010101011000),$$
$$v_3 = (11101111011101101110010101011011),$$
$$v_4 = (11100110001000011000010101011001),$$
$$v_5 = (11101111011101101110010101011011),$$
$$v_6 = (11100110011000010000010001010001),$$
$$v_7 = (11010110001001001000110001011000),$$
$$v_8 = (11110110001000101000110101001001),$$
$$v_9 = (11100110001001001000110001011001),$$
$$v_{10} = (11101111011101101110010101011011),$$
$$v_{11} = (11010110000001001000110001011000),$$
$$v_{12} = (11010110001001001000110001011001),$$
$$v_{13} = (11101111011101101110010101011011),$$
$$v_{14} = (11100110011000010000010101011011),$$
$$v_{15} = (11100110101011100101010011011001),$$
$$v_{16} = (11100110011000010100010001010001),$$
$$v_{17} = (11100110011000010000010101011011),$$
$$v_{18} = (11100110011000010000010101011001),$$
$$v_{19} = (11110110001000101000111000001001),$$
$$v_{20} = (11100110011000010000010101011001),$$

而种群中个体的适应值分别如下：

$$\text{eval}(v_1) = f(11.120940, 5.092514) = 30.298543,$$
$$\text{eval}(v_2) = f(10.588756, 4.667358) = 26.869724,$$
$$\text{eval}(v_3) = f(11.124627, 5.092514) = 30.316575,$$
$$\text{eval}(v_4) = f(10.574125, 4.242410) = 31.933120,$$
$$\text{eval}(v_5) = f(11.124627, 5.092514) = 30.316575,$$
$$\text{eval}(v_6) = f(10.588756, 4.214603) = 34.356125,$$
$$\text{eval}(v_7) = f(9.631066, 4.427881) = 35.458636,$$
$$\text{eval}(v_8) = f(11.518106, 4.452835) = 23.309078,$$
$$\text{eval}(v_9) = f(10.574816, 4.427933) = 34.393820,$$
$$\text{eval}(v_{10}) = f(11.124627, 5.092514) = 30.316575,$$
$$\text{eval}(v_{11}) = f(9.623693, 4.427881) = 35.477938,$$
$$\text{eval}(v_{12}) = f(9.631066, 4.427933) = 35.456066,$$

$$\begin{aligned}
\text{eval}(v_{13}) &= f(11.124627, 5.092514) = 30.316575,\\
\text{eval}(v_{14}) &= f(10.588756, 4.242514) = 32.932098,\\
\text{eval}(v_{15}) &= f(10.606555, 4.653714) = 30.746768,\\
\text{eval}(v_{16}) &= f(10.588814, 4.214603) = 34.359545,\\
\text{eval}(v_{17}) &= f(10.588756, 4.242514) = 32.932098,\\
\text{eval}(v_{18}) &= f(10.588756, 4.242410) = 32.956664,\\
\text{eval}(v_{19}) &= f(11.518106, 4.472757) = 19.669670,\\
\text{eval}(v_{20}) &= f(10.588756, 4.242410) = 32.956664.
\end{aligned}$$

然而，如果仔细观察演化进程，则会发现在较早代种群中某些个体的适应值要比 1000 代后种群中最好个体的适应值 35.477938 要大. 例如，第 396 代种群中最好个体的适应值为 38.827553. 这是由于采样的随机误差所引起的.

在实现遗传算法时，一个常用的方法是将到当前代为止演化的最好个体单独存放起来，在遗传算法结束后，将演化过程中发现的最好个体作为问题的最优解或近似最优解.

2.3 遗传算法的实现技术

虽然遗传算法的结构非常简单，但用遗传算法求解问题的功效常常依赖于一些具体的实现技术，与所采用的编码方案、选择策略、遗传算子等有密切的关系. 本节介绍遗传算法的一些常用实现技术.

2.3.1 编码

怎样将问题的可能解编码为染色体是用遗传算法求解问题的一个关键问题. 编码对于算法的性能影响很大，与遗传算子的设计等问题密切相关. 在 Holland 最初提出的遗传算法中，是用二进制位串对问题的解进行编码的，但对许多实际问题来说，用二进制位串对问题的可能解进行编码是不自然的，用来求解问题的效果也不是很好. 目前，已经提出了许多非二进制位串编码的方法，本节介绍一些常用的编码方法.

1. 二进制编码

二进制编码将原问题的解空间映射到位串空间 $\{0,1\}^L$ 上，其中，L 为某个固定常数. 对有些优化问题，二进制编码是一个自然的选择.

例 2.1 0-1 背包问题. 给定 n 种物品和一个背包，假设第 i 种物品重 w_i，价值为 p_i，背包可容纳的重量为 W，求将物品放入背包的一种放法，使得背包所装的物品价值最大，这里假定物品不可分割，即一种物品要么完整地放入背包，要么不放