第七章　遗传算法的应用

7.1　数值函数优化计算

7.1.1　遗传算法与纯数值函数优化计算

在遗传算法的研究工作中，纯数值函数优化问题受到了较大的重视。在遗传算法产生的早期，就有一批学者（如 Hollstien、De Jong 等）在研究这个问题，而直到如今，仍有不少的人还在不断地探索能用来求解数值函数优化问题的性能更好的遗传算法。之所以如此，主要是由于下面两条原因：

首先，对很多实际问题进行数学建模后，可将其抽象为一个数值函数的优化问题。由于问题种类的繁多，影响因素的复杂，这些数学函数会呈现出不同的数学特征。如有些函数是连续的，而有些函数却是离散的；有些函数是凸函数，而有些函数却不是凸函数；有些函数是单峰值的，而有些函数却是多峰值的。更经常遇到的函数是这些不同数学特征的组合。除了在函数是连续、可求导、低阶的简单情况下可解析地求出其最优解外，大部分情况下需要通过数值计算的方法来进行近似优化计算。尽管人们对这个问题进行了多年的研究，而至今仍尚无一种既能处理各种不同的复杂函数，又具有良好求解结果的数值计算方法。特别是当问题的规模比较大时，优化计算时的搜索空间也急剧扩大，人们逐渐意识到要严格地求出其最优解既不可能、也不现实。所以需要研究出一种能够在可接受的时间和可接受的精度范围内求出数值函数近似最优解的方法或通用算法。遗传算法提供了一种求解这种优化问题的通用框架。遗传算法通过对群体所施加的迭代进

化过程，不断地将当前群体中具有较高适应度的个体遗传到下一代群体中，并且不断地淘汰掉适应度较低的个体，从而最终寻求出适应度最大的个体。这个适应度最大的个体经解码处理之后所对应的个体表现型就是这个实际应用问题的最优解或近似最优解。

其次，如何评价一个遗传算法的性能优劣程度一直是一个比较难的问题，这主要是因为现实问题种类繁多，影响因素复杂，若对各种情况都加以考虑进行试算，其计算工作量势必太大。虽然人们提出了一些评价指标，如 De Jong 的在线性能指标和离线性能指标，但这都只是反映了算法在某一方面的性能，尚不足以完全反映算法在各种实际应用中的效果。而另一方面，由于纯数值函数优化问题不包含有某一具体应用领域中的专门知识，它们便于不同应用领域中的研究人员能够进行相互理解和相互交流，并且能够较好地反映算法本身所具有的本质特征和实际应用能力。所以人们专门设计了一些具有复杂数学特征的纯数学函数，通过遗传算法对这些函数的优化计算情况来测试各种遗传算法的性能。

利用遗传算法进行数值函数优化计算时，若精度要求不是太高，自变量的个数不是太多时，可以采用二进制编码来表示个体；若精度要求较高，自变量的个数较多时，可以采用浮点数编码来表示个体。

7.1.2 评价遗传算法性能的常用测试函数

在设计用于评价遗传算法性能的测试函数时，必须考虑实际应用问题的数学模型中所可能呈现出的各种数学特性，以及可能遇到的各种情况和影响因素。这里所说的数学特性主要包括：

- 连续函数或离散函数；
- 凹函数或凸函数；
- 二次函数或非二次函数；
- 低维函数或高维函数；
- 确定性函数或随机性函数；

●单峰值函数或多峰值函数，等等。

下面是一些在评价遗传算法性能时经常用到的测试函数[81,82]：

（1）De Jong 函数 F1：

$$\begin{cases} f_1(x_1,x_2,x_3) = \sum_{i=1}^{3} x_i^2 \\ -5.12 \leqslant x_i \leqslant 5.12 \qquad (i=1,2,3) \end{cases} \tag{7-1}$$

这是一个简单的平方和函数，只有一个极小点 f_1（0，0，0）=0。

（2）De Jong 函数 F2：

$$\begin{cases} f_2(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2 \\ -2.048 \leqslant x_i \leqslant 2.048 \qquad (i=1, 2) \end{cases} \tag{7-2}$$

这是一个二维函数，它具有一个全局极小点 f_2（1，1）=0。该函数虽然是单峰值的函数，但它却是病态的，难以进行全局极小化。

（3）De Jong 函数 F3：

$$\begin{cases} f_3(x_1,x_2,\cdots,x_5) = \sum_{i=1}^{5} \text{integer}(x_i) \\ -5.12 \leqslant x_i \leqslant 5.12 \qquad (i=1,2,\cdots,5) \end{cases} \tag{7-3}$$

这是一个不连续函数，对于 $x_i \in [-5.12, -5.0]$区域内的每一个点，它都取全局极小值 $f_3(x_1,x_2,x_3,x_4,x_5) = -30$。

（4）De Jong 函数 F4：

$$\begin{cases} f_4(x_1,x_2,\cdots,x_{30}) = \sum_{i=1}^{30} ix_i^4 + \text{Gauss}(0,1) \\ -1.28 \leqslant x_i \leqslant 1.28 \qquad (i=1,2,\cdots,30) \end{cases} \tag{7-4}$$

这是一个含有高斯噪声的 4 次函数，当不考虑噪声的影响时，它具有一个全局极小值 f_4（0，0，…，0）=0。

（5）De Jong 函数 F5：

$$\begin{cases} f_5(x_1,x_2) = 0.002 + \sum_{j=1}^{25} \dfrac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6} \\ -65.536 \leqslant x_i \leqslant 65.536 \qquad (i = 1,2) \\ [a_{ij}] = \\ \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & \cdots & 32 & 32 & 32 \end{bmatrix} \end{cases} \tag{7-5}$$

这是一个多峰值函数，它总共有 25 个局部极小点，其中有一个是全局极小点，全局极小值为 $f_5(-32, -32) = 0.998$。

(6) Schaffer 函数 F6：

$$\begin{cases} f_6(x_1, x_2) = 0.5 + \dfrac{\sin^2\sqrt{x_1^2 + x_2^2} - 0.5}{[1.0 + 0.001(x_1^2 + x_2^2)]^2} \\ -100 \leqslant x_i \leqslant 100 \qquad (i = 1, 2) \end{cases} \tag{7-6}$$

该函数在其定义域内只具有一个全局极小点 $f_6(0, 0) = 0$。

(7) Schaffer 函数 F7：

$$\begin{cases} f_7(x_1,x_2) = (x_1^2 + x_2^2)^{0.25}[\sin^2(50(x_1^2 + x_2^2)^{0.1}) + 1.0] \\ -100 \leqslant x_i \leqslant 100 \qquad (i = 1,2) \end{cases} \tag{7-7}$$

该函数在其定义域内只具有一个全局极小点 $f_7(0, 0) = 0$。

(8) Goldstein-Price 函数：

$$\begin{cases} f(x_1,x_2) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 \\ \qquad\qquad - 14x_2 + 6x_1x_2 + 3x_2^2)] \cdot \\ \qquad\qquad [30 + (2x_1 - 3x_2)^2(18 - 32x_1 \\ \qquad\qquad + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)] \\ -2 \leqslant x_i \leqslant 2 \qquad (i = 1,2) \end{cases} \tag{7-8}$$

该函数在其定义域内只具有一个全局极小点 $f(0,-1)=3$。

(9) Shubert 函数：

$$\begin{cases} f(x_1,x_2)=\sum_{i=1}^{5} i\cos[(i+1)x_1+i]\cdot\sum_{i=1}^{5} i\cos[(i+1)x_2+i] \\ -10\leqslant x_i\leqslant 10 \qquad (i=1,2) \end{cases} \tag{7-9}$$

这是一个多峰值函数，在其定义域内它总共有 760 个局部最小点，其中的 18 个点是全局最小点，全局最小值为 $f=-186.731$。

(10) 六峰值驼背函数 (Six-hump Camel Back Function)：

$$\begin{cases} f(x,y)=\left(4-2.1x^2+\frac{x^4}{3}\right)x^2+xy+(-4+4y^2)y^2 \\ -3\leqslant x\leqslant 3 \\ -2\leqslant y\leqslant 2 \end{cases} \tag{7-10}$$

该函数共有六个局部极小点，其中 $(-0.0898, 0.7126)$ 和 $(0.0898, -0.7126)$ 为全局最小点，最小值为 $f(-0.0898, 0.7126)=f(0.0898, -0.7126)=-1.031628$。

(11) 带有复杂约束条件的函数 (之一)：

$$\begin{cases} \min \quad f(x,y)=5x_1+5x_2+5x_3+5x_4-5\sum_{i=1}^{4}x_i^2-\sum_{i=1}^{9}y_i \\ \text{s.t.} \quad 2x_1+2x_2+y_6+y_7\leqslant 10 \quad 2x_1+2x_3+y_6+y_8\leqslant 10 \\ \qquad 2x_2+2x_3+y_7+y_8\leqslant 10 \quad -8x_1+y_6\leqslant 10 \\ \qquad -8x_2+y_7\leqslant 0 \qquad -8x_3+y_8\leqslant 0 \\ \qquad -2x_4-y_1+y_6\leqslant 0 \qquad -2y_2-y_3+y_7\leqslant 0 \\ \qquad -2y_4-y_5+y_8\leqslant 0 \qquad 0\leqslant x_i\leqslant 1(i=1,2,3,4) \\ 0\leqslant y_i\leqslant 1(i=1,2,3,4,5,9) \qquad 0\leqslant y_i(i=6,7,8) \end{cases} \tag{7-11}$$

该函数的全局最小点为：$f(1, 1, 1, 1, 1, 1, 1, 1, 1,$

3，3，3，1）＝－15。

（12）带有复杂约束条件的函数（之二）：

$$\begin{cases} \max f(x_1, x_2, x_3) = \dfrac{3x_1 + x_2 - 2x_3 + 0.8}{2x_1 - x_2 + x_3} + \dfrac{4x_1 - 2x_2 + x_3}{7x_1 + 3x_2 - x_3} \\ \text{s.t.} \quad x_1 + x_2 - x_3 \leqslant 1 \\ \qquad -x_1 + x_2 - x_3 \leqslant -1 \\ \qquad 12x_1 + 5x_2 + 12x_3 \leqslant 34.8 \\ \qquad 12x_1 + 12x_2 + 7x_3 \leqslant 29.1 \\ \qquad -6x_1 + x_2 + x_3 \leqslant -4.1 \\ 0 \leqslant x_i \quad (i = 1, 2, 3) \end{cases} \tag{7-12}$$

该函数的全局最大点为：f（1，0，0）＝2.471428。

7.1.3 De Jong 的研究结论[4]

De Jong 结合模式定理，对遗传算法在纯数值函数优化中的应用进行了细致深入的研究工作，他通过对算法在线性能指标和离线性能指标的定义，使用精心挑选出的 5 个测试函数，在计算机上经过大量的计算实践，得到了一些在遗传算法的发展和应用过程中具有重要指导意义的结论，其研究方式已成为遗传算法研究和开发中的典范。

De Jong 提出的两个评价遗传算法性能的指标是：在线性能指标和离线性能指标。

【定义 7.1】 在环境 e 下策略 s 的在线性能 $X_e(s)$ 定义为：

$$X_e(s) = \frac{1}{T}\sum_{t=1}^{T} f_e(t) \tag{7-13}$$

式中，$f_e(t)$ 是在环境 e 下第 t 时刻的平均目标函数值或平均适应度。

由定义 7.1 可知，算法的在线性能指标表示了算法从开始运行一直到当前为止的时间段内性能值的平均值，它反映了算法的动态性能。

【定义 7.2】 在环境 e 下策略 s 的离线性能 $X_e^*(s)$ 定义为：

$$X_e^*(s) = \frac{1}{T}\sum_{t=1}^{T} f_e^*(t) \tag{7-14}$$

式中，f_e^* (s) 是在环境 e 下 $[0, t]$ 时间段内最好的目标函数值或最大的适应度。

由定义 7.2 可知，算法的离线性能表示了算法运行过程中各进化代的最佳性能值的累积平均，它反映了算法的收敛性能。

De Jong 用来进行纯数值函数优化问题研究的研究对象是在 7.1.2 节中所介绍的 De Jong 测试函数 F1～F5。他采用了下面的一些研究方法：

1. 编码方法

用二进制编码符号串来表示个体。

2. 算法的影响参数

- 群体大小 M；
- 交叉概率 p_c；
- 变异概率 p_m；
- 代沟 G。

3. 算法种类（子群体复制策略）

- R1：基本遗传算法（比例选择、单点交叉、基本位变异）；
- R2：保留最佳个体模型；
- R3：期望值模型；
- R4：保留最佳期望值模型；
- R5：排挤因子模型；
- R6：广义交叉模型。

经过仔细分析和计算，De Jong 得到了下述几条重要的结论：

结论 1

群体的规模越大，遗传算法的离线性能越好，越容易收敛。

结论 2

规模较大的群体，遗传算法的初始在线性能较差；而规模较小的群体，遗传算法的初始在线性能较好。

结论 3

虽然变异概率的增大也会增加群体的多样性，但它却降低了遗传算法的离线性能和在线性能，并且随着变异概率的增大，遗传算法的性能越来越接近于随机搜索算法的性能。

结论 4

使用保留最佳个体模型或期望值模型的遗传算法比基本遗传算法的性能有明显的改进。

结论 5

对于广义交叉算子，随着交叉点数的增加会降低遗传算法的在线性能和离线性能。

这里之所以介绍 De Jong 的研究结论，是因为这些结论在遗传算法的开发研究和实际应用中具有重要的指导意义。

7.2 多目标优化

7.2.1 多目标优化的基本概念

前面讨论的都是单个目标在给定区域上的最优化问题，而在实际应用中，会更多地遇到需要使多个目标在给定区域上都尽可能地好的优化问题。例如在设计新产品时，我们既要考虑使产品具有较好的功能，又要考虑使其制造成本最低，同时还要考虑产品的可制造性、可靠性、可维修性等，这些设计目标的改善有可能是相互抵触的（如好的可维修性有可能会引起可靠性的降低），这就需要在这些设计目标之间取一折衷结果。再例如投资问题，一般我们都是希望所投入的资金量最少，风险最小，并且所获得的收益最大。这种多于一个的数值目标在给定区域上的最优化问题就称为多目标优化（Multi-object Optimization）。

多目标优化问题一般可描述为下面的数学模型：

$$\begin{cases} \mathrm{V}-\min \quad f(x)=[f_1(x), f_2(x), \cdots, f_p(x)]^T \\ \text{s.t.} \qquad\quad x \in X \\ \qquad\quad\ X \subseteq R^n \end{cases} \tag{7-15}$$

式中，V-min 表示向量极小化，即向量目标 $f(x)=[f_1(x), f_2(x), \cdots f_p(x)]^T$中的各个子目标函数都尽可能地极小化的意思。

多目标优化问题的本质在子，在很多情况下，各个子目标有可能是相互冲突的，一个子目标的改善有可能会引起另一个子目标性能的降低，也就是说，要同时使这多个子目标都一起达到最优值是不可能的，而只能是在它们中间进行协调和折衷处理，使各个子目标函数都尽可能地达到最优。

多目标优化问题的最优解与单目标优化问题的最优解有着本质上的不同，所以为了正确地求解多目标优化问题，必须对其最优解的概念进行定义[83]。

【定义 7.3】 设 $X\subseteq R^m$ 是多目标优化模型的约束集，$f(x)\in R^p$ 是多目标优化时的向量目标函数，$x_1\in X$，$x_2\in X$。若

$$f_k(x_1)\leqslant f_k(x_2) \qquad (\forall k=1, 2, \cdots, p)$$

并且

$$f_k(x_1)<f_k(x_2) \qquad (\exists k=1, 2, \cdots, p)$$

则称解 x_1 比解 x_2 优越。

【定义 7.4】 设 $X\subseteq R^m$ 是多目标优化模型的约束集，$f(x)\in R^p$ 是向量目标函数。若 $x^*\in X$，并且 x^* 比 X 中的所有其他点都优越，则称 x^* 是多目标极小化模型的最优解。

由该定义可知，多目标优化问题的最优解 x^* 就是使向量目标函数 $f(x)$ 的每一个子目标函数都同时到达最优点的解，如图 7-1 所示。

显然，在大多数情况下，多目标优化问题的最优解是不存在的。

【定义 7.5】 设 $X\subseteq R^m$ 是多目标优化模型的约束集，$f(x)\in R^p$ 是向量目标函数。若 $\tilde{x}\in X$，并且不存在比 $\tilde{x}$ 更优越的 x，则称 $\tilde{x}$ 是多目标极小化模型的 Pareto 最优解，或称为非劣解。

由该定义可知，多目标优化问题的 Pareto 最优解仅仅只是它的一个可以接受的“不坏”的解，并且通常的多目标优化问题

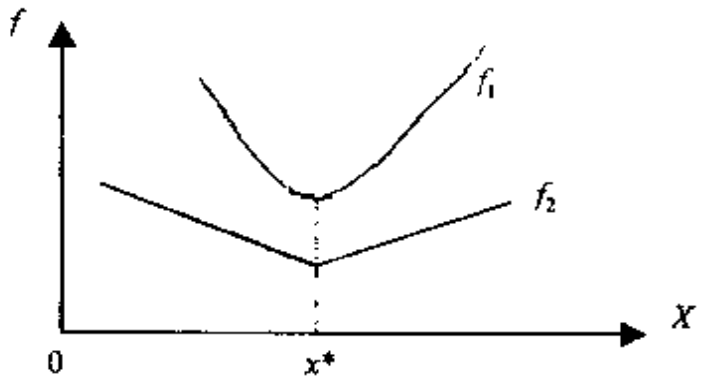

图 7-1　多目标优化问题的最优解

大多都具有很多个 Pareto 最优解，如图 7-2 所示。

由上述三个定义可知，若一个多目标优化问题存在最优解的话，则这个最优解必定是 Pareto 最优解，并且 Pareto 最优解也只由这些最优解所组成，再不包含有其他解。所以可以这么说，Pareto 最优解是多目标优化问题的合理的解集合。

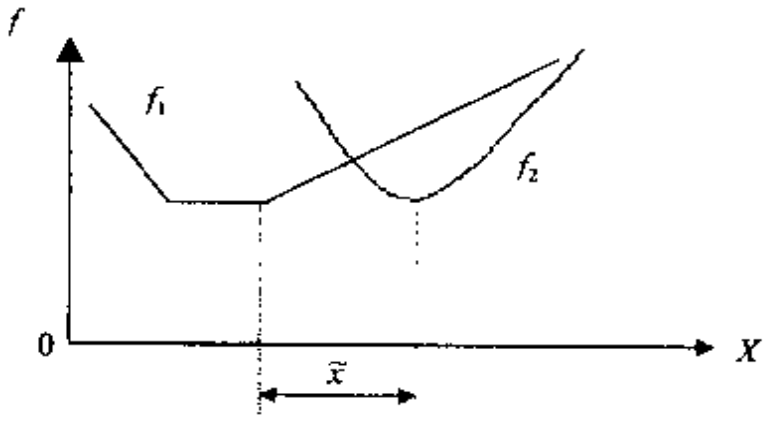

图 7-2　多目标优化问题的 Pareto 最优解

一般来说，多目标优化问题的 Pareto 最优解是一个集合。对于实际应用问题，必须根据对问题的了解程度及决策人员的个人偏好，从多目标优化问题的 Pareto 最优解集合中挑选出一个或一些解作为所求多目标优化问题的最优解。所以求解多目标优化问题的首要步骤和关键步骤是求出其所有的 Pareto 最优解。

由于多目标优化问题是实际中会经常遇到的问题，所以人们

对这个问题投入了较多的精力，研究出了多种求解方法，如线性加权和法、极大极小法、理想点法、逐次线性加权和法等[83]。这些求解方法各有各的优点，也各有各的应用限制。

7.2.2 多目标优化与遗传算法

由于遗传算法是对整个群体所进行的进化运算操作，它着眼于个体的集合，而多目标优化问题的 Pareto 最优解一般也是一个集合，因而可以预计遗传算法是求解多目标优化问题的 Pareto 最优解集合的一个有效手段。

求解多目标优化问题的遗传算法的基本结构与求解单目标优化问题的遗传算法的基本结构相类似。另一方面，在利用遗传算法进行多目标优化问题求解时，需要考虑如何评价 Pareto 最优解，如何设计适合于多目标优化问题的选择算子、交叉算子、变异算子等问题，所以算法在实现时也有其独特的地方。在算法的实现中，我们可以基于各个子目标函数之间的优化关系进行个体的选择运算；也可以对各个子目标函数进行独立的选择运算；也可以运用小生境技术；还可以把原有的多目标优化问题求解方法与遗传算法相结合构成混合遗传算法。对于具体的应用问题，如何选用哪种方法，仍是取决于对该问题的理解程度及决策人员的偏好。

7.2.3 求解多目标优化问题的遗传算法

对于如何求多目标优化问题的 Pareto 最优解，目前已经提出了多种基于遗传算法的求解方法。下面介绍其中几种主要的方法。

1. 权重系数变化法

对于一个多目标优化问题，若给其各个子目标函数 $f_i(x)$ $(i=1, 2, \cdots, p)$ 赋予不同的权重 w_i $(i=1, 2, \cdots, p)$，其中各 w_i 的大小代表相应子目标 $f_i(x)$ 在多目标优化问题中的重要程度。则各个子目标函数的线性加权和可表示为：

$$u(f(x)) = \sum_{i=1}^{p} w_i f_i(x) \tag{7-16}$$

若以这个线性加权和作为多目标优化问题的评价函数，则多目标优化问题就可转化为单目标优化问题。权重系数变化法[84]就是在这个评价函数的基础上，对每个个体取不同的权重系数，就可以利用通常的遗传算法来求出多目标优化问题的多个 Pareto 最优解。

2．并列选择法

并列选择法的基本思想是[85]：先将群体中的全部个体按子目标函数的数目均等地划分为一些子群体，对每个子群体分配一个子目标函数，各个子目标函数在其相应的子群体中独立地进行选择运算，各自选择出一些适应度较高的个体组成一个新的子群体，然后再将所有这些新生成的子群体合并为一个完整的群体，在这个完整的群体中进行交叉运算和变异运算，从而生成下一代的完整群体，如此这样不断地进行“分割—并列选择—合并”过程，最终可求出多目标优化问题的 Pareto 最优解。

这种方法很容易产生个别子目标函数的极端最优解，而要找到所有目标函数在某种程度上较好的协调最优解却比较困难。

3．排序选择法

排序选择法的基本思想是[5]：基于“Pareto 最优个体”的概念来对群体中的各个个体进行排序，依据这个排列次序来进行进化过程中的选择运算，从而使得排在前面的 Pareto 最优个体将有更多的机会遗传到下一代群体中。如此这样经过一定代数的循环之后，最终就可求出多目标优化问题的 Pareto 最优解。

这里所谓的 Pareto 最优个体，是指群体中的这样一个或一些个体，群体中的其他个体都不比它或它们更优越。需要说明的是，在群体进化过程中所产生的 Pareto 最优个体并不一定就对应于多目标优化问题的 Pareto 最优解。当然，当遗传算法运行结束时，我们需要取排在前面的几个 Pareto 最优个体，以它们所对应的解来作为多目标优化问题的 Pareto 最优解。

对群体中的所有个体进行 Pareto 最优个体排序的算法是：

算法 ParetoIndividual

①设置初始序号 $r \leftarrow 1$。

②求出群体中的 Pareto 最优个体，定义这些个体的序号为 r。

③从群体中去掉 Pareto 最优个体，并更改序号 $r \leftarrow r+1$。

④转到第②步，直到处理完群体中的所有个体。

由上述 Pareto 最优个体排序算法可知，排序选择法仅仅度量了各个个体之间的优越次序，而未度量各个个体的分散程度，所以它易于生很多个相似的 Pareto 最优解，而难于生成分布较广的 Pareto 最优解。

4．共享函数法

求解多目标优化问题时，一般希望所得到的解能够尽可能地分散在整个 Pareto 最优解集合内，而不是集中在其 Pareto 最优解集合内的某一个较小的区域上。为达到这个要求，可以利用小生境遗传算法的技术来求解多目标优化问题。这种求解多目标优化问题的方法称为共享函数法，它将共享函数的概念引入求解多目标优化问题的遗传算法中。

在利用通常的遗传算法求解最优化问题时，算法并未限制相同个体或类似个体的数量。但当在遗传算法中引入小生境技术之后，算法对它们的数量就要加以限制，以便能够产生出种类较多的不同的最优解。对于某一个个体 X 而言，在它的附近还存在有多少种、多大程度相似的个体，这是可以度量的，这种度量值称之为小生境数（Niche Count）。小生境数有很多种不同的度量计算方法，一般可定义为[5]：

$$m_X = \sum_{Y \in P} s(d(X, Y)) \tag{7-17}$$

式中，$s(d)$ 为共享函数，它是个体之间距离 d 的单调递减函数。

例如，共享函数 $s(d)$ 的一种定义是：

$$s(d) = \begin{cases} 1 - \dfrac{d}{\sigma} & \text{if} \quad 0 \leqslant d \leqslant \sigma \\ 0 & \text{if} \quad d > \sigma \end{cases} \tag{7-18}$$

式中，$d(X, Y)$ 是两个个体 X、Y 之间的海明距离，$\sigma>0$ 是预先指定的一个表示小生境范围的参数。

在计算出各个个体的小生境数之后，可以使小生境数较小的个体能够有更多的机会被选中遗传到下一代群体中，即相似个体较少的个体能够有更多的机会被遗传到下一代群体中，这样也就增加了群体的多样性，相应地也会增加解的多样性。

下面描述一种遗传算法中的选择操作方法，它综合运用联赛选择和共享函数的思想来选择当前群体中的优良个体遗传到下一代群体中[86]。

算法 TournamentSharingSelection

① 从群体中随机选取 k 个个体组成个体比较集合 C，其中 k 是预先指定的一个表示比较集合规模的常数。

② 从群体中随机选择 2 个个体组成个体联赛集合 T。

③ 分别比较个体联赛集合 T 中的 2 个个体与个体比较集合 C 中各个个体之间的优越关系，根据这个比较结果，按下述方法从个体联赛集合 T 中选择出一个个体遗传到下一代群体中。

- 如果集合 T 中的一个个体（记为 X）比集合 C 中的所有个体都优越，而集合 T 中的另一个个体都不比集合 C 中的所有个体优越，则将个体 X 遗传到下一代群体中；
- 如果由上面的一种情况无能选择出一个个体，则利用共享函数的概念从集合 T 中选择出一个小生境数较小的个体遗传到下一代群体中。

使用这个选择操作方法的遗传算法可用于求解多目标优化问题的 Pareto 最优解。该方法的优点是它能够得到多种不同的 Pareto 最优解，但另一方面，由于每次进行选择操作时都需要进行大量的个体之间优越关系的评价和比较运算，所以使得算法的搜索效率较低。

5. 混合法

前面所介绍的几种求解多目标优化问题的遗传算法各有各的优点、也各有各的缺点。例如，并列选择法易于生成单个目标函数的极端最优解，而较难生成一种多个目标在某种程度上都比较满意的折衷解；共享函数法虽然易于生成分布较广的 Pareto 最

优解集合，但其搜索效率却比较低。于是会很自然地意识到，如果混合使用上述几种求解多目标优化问题的方法，将有可能尽量地克服各自的缺点，而充分地发挥各自的优点。

下面介绍一种使用遗传算法求解多目标优化问题的混合方法。该方法的主要思想是：选择算子的主体使用并列选择法，然后通过引入保留最佳个体和共享函数的思想来弥补仅仅只使用并列选择法的不足之处。算法的主要过程是[87]：

算法 HybridSelection

①并列选择过程：

按所求多目标优化问题的子目标函数的个数，将整个群体均等地划分为一些子群体，各个子目标函数在相应的子群体中产生其下一代子群体。

②保留 Pareto 最优个体过程：

对于各个子群体中的 Pareto 最优个体，不让其参与个体的交叉运算和变异运算，而是将这个或这些 Pareto 最优个体直接保留到下一代子群体中。

③共享函数处理过程：

若所得到的 Pareto 最优个体的数量已超过规定的群体规模，则需要利用共享函数的处理方法来对这些 Pareto 最优个体进行挑选，以形成规定规模的新一代群体。

7.2.4 算例

【例】考虑下述含有二个优化目标的多目标优化问题：

$$\begin{aligned} \min \quad & f_1 = \frac{x_1^2}{4} \\ \min \quad & f_2 = x_1 (1 - x_2) + 5 \\ \text{s.t.} \quad & 1 \leqslant x_1 \leqslant 4 \\ & 1 \leqslant x_2 \leqslant 2 \end{aligned}$$

对于该多目标优化问题，分别用并列选择法、排序选择法、共享函数法和混合法进行了求解。各种方法的运行结果分别如图 7-3、图 7-4、图 7-5、图 7-6 所示，在各个图中，表示出了在遗传算法运行结束时的最终群体中所含全部个体的分布情况。

在求解时所使用的遗传算法中，各个变量用 8 位长的二进制编码串来表示，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，各个运行参数为：

$$|M,\ T,\ p_c,\ p_m| = |100,\ 100,\ 0.8,\ 0.01|$$

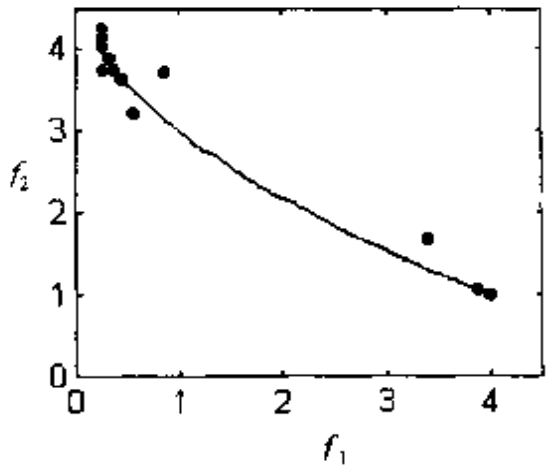

图 7-3　并列选择法的运行结果

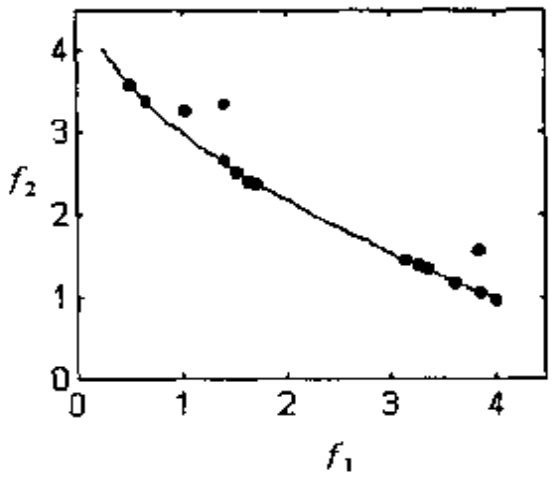

图 7-4　排序选择法的运行结果

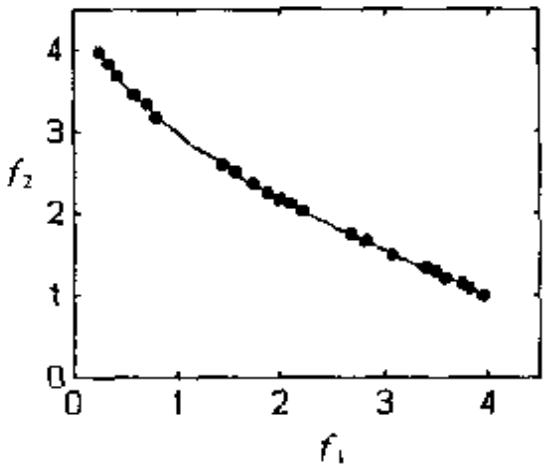

图 7-5　共享函数法的运行结果

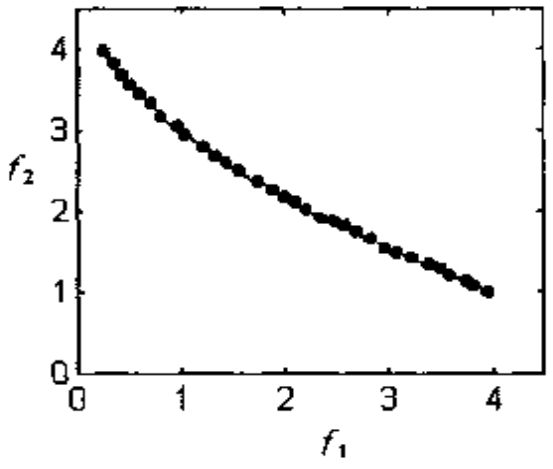

图 7-6　混合法的运行结果

7.3　求解装箱问题的遗传算法

7.3.1　装箱问题的描述

管理工程、工业工程等领域中的一些问题（例如人力资源分配、运输计划等）均可建模为装箱问题（Bin Packing），对其求