第6章　遗传算法与数值优化

最优化问题是遗传算法经典应用领域，但采用常规方法对于大规模、多峰多态函数、含离散变量等问题的有效解决往往存在着许多障碍。遗传算法简单易行、高效性及其普遍适用性，已经赢得了许多应用，并且得到了长足的发展。本章和下一章将重点讨论遗传算法在这一领域的应用。本章首先概述最优化问题的常规方法及其分类；然后介绍 De Jong 的遗传算法应用于含上、下限约束的最优化问题方法以及多峰函数的优化，并引入分享机制；对于含一般线性约束条件的最优化问题，介绍 Michalewicz 的方法；最后考察一下多目标优化问题，引入 Pareto 最优解集的概念，介绍遗传算法在这方面的应用和研究进展。

6.1　最优化问题

一般地，最优化问题(optimization problem)由目标函数(objective function)和约束条件(constraints)两部分构成：

$$\begin{aligned} &\text{Minimize} \quad f(x) = f(x_1, x_2, \cdots, x_n) \\ &\text{subject to} \quad x = (x_1, x_2, \cdots, x_n) \in S \subset X \end{aligned} \tag{6.1}$$

将满足所有约束条件的解空间 S 称为可行域(feasible region)，可行域中的解称为可行解(feasible solution)；将可行域中使目标函数最小的解称为最优解(optimal solution)。对于最大化问题，可将目标函数乘以(-1)，转化为最小化问题求解。

当 $X = \mathbf{R}^n$ 时(n 元实空间)，目标函数和约束条件均为线性表达式，最优化问题(6.1)称为线性规划问题(linear programming problem)；否则称之为非线性规划问题(nonlinear programming problem)。线性规划问题对应于单纯形法(simplex method)和对偶理论求解。当目标函数 $f(x)$ 为二次函数、约束条件全部为线性表达式时称为二次规划(quadratic programming)，可以找到类似于线性规划的在有限步搜索的优化方法。当目标函数 $f(x)$ 为凸函数、可行域为凸空间时，该优化问题称为凸规划(convex programming)，依据连续性和可微性的假设，有最小平方和法、最速下降法以及牛顿法等经典无约束方法。对于非凸规划(non-convex programming)问题，虽然可以应用互补转轴理论(complementary pivot theory)的推广，但一直没有非常有效的最优化方法，在一定程度上，随机搜索方法是较好的选择。当 X 或 S 为离散集合构成的解空间时，这类最优化问题称为组合最优化(combinatorial optimization problem)。严格意义上的最优解求取非常困难，研究高速近似的算法是一重要的发展方向。对全局优化问题，目前存在确定性和非确定性两类方法。前者以 Brianin 的下降轨线法、Levy 的隧道法和 R.Ge 的填充函数法为代表。该类方法虽然收敛快、计算效率较高，但算法复杂，求得全局极值的概率不大。非确定性方法以 Monte-Carlo 随机试验法、Hartman 的多始点法、Solis 和 Wets 的结合梯度信息的搜索方法、模拟退火法(simulated annealing)等为代表。该类方法对目标函

数要求低、容易实现、稳定性好，但收敛速度较慢、求得全局极值的概率较低。遗传算法作为现代最优化的手段，实践证明，它应用于大规模、多峰多态函数、含离散变量等情况下的全局优化问题是合适的，在求解速度和质量上远超过常规方法，因而是一高速近似算法。本章和下一章将对遗传算法与常规的一般优化方法进行分析比较。

组合最优化问题，顾名思义，它研究的对象可以看作是在有限集合上定义的函数在各种条件下的极值问题。从理论上说，这类问题若有解，总是可以用枚举的方法找到。这意味着以枚举为基础发展起来的分枝定界法(branch and bound method)具有普遍适用性。但实际上对于大规模的问题分枝定界法通常是不能实际使用的，该方法的计算时间一般为输入数据量(长)的指数函数，计算时间会迅速增加到现代计算工具难以承受的地步。根据计算复杂性的理论，一个好的算法，其计算时间作为输入数据量(长)的函数应该有一个多项式作为上界，这样的算法被称为多项式时间算法，简称有效算法；在组合优化中，至今还没有似乎也不可能发现普遍适用的多项式时间算法。一个多项式时间算法问题被称为多项式时间可解问题，或者 P 问题。组合优化中多数问题属于一类不知道是否为 P 问题的问题，其中，包括所谓的 NP 完全(Nondeterministic Polynomial Completeness)问题。在这类问题里，只要有一个被证明是 P 问题，那么它们全部是 P 问题。至今已被证明是属于 NP 完全问题的数目有几千个之多。一般认为，NP 完全问题不会是 P 问题。因此对 NP 完全问题，既然没有准确的多项式时间算法，比较现实的妥协方法是采用多项式时间近似算法。近似方法分为启发式方法(heuristic method)和随机方法(random method)两种，启发式方法有有序搜索(贪心算法或称最好优先搜索)和最优 A^* 算法。与盲目搜索不同的是：启发式搜索运用启发信息，应用某些经验或规则来重新排列 Open 表中结点的顺序，使搜索沿着某个被认为最有希望的前沿区段扩展。启发式方法较多地依赖于对问题构造和性质的认识和经验，适用于解决不太复杂的问题。随机方法不依赖于问题的性质，从解空间 X 中随机地选择多个解，检查这些解的可行性，在可行解集中选择目标函数最小的解作为最优解。这种方法的算法简单明了，但因为需要检查相当多的可行解集，计算非常耗时，尤其对于大范围的搜索，很有可能遗漏最优解。

对于组合最优化问题，与常规方法比较，可以认为遗传算法处于随机方法与启发式方法之间，它在引入随机搜索的同时，在解的构成法和演算过程中考虑了问题的原有构造。

6.2 只含上、下限约束的最优化问题

只含上、下限约束的最优化问题可以描述为：

$$\begin{aligned} &\text{Minimize} \quad f(x_1, x_2, \cdots, x_n) \\ &\text{subject to} \quad l_i \leqslant x_i \leqslant u_i \quad i = 1, 2, \cdots, n \end{aligned} \tag{6.2}$$

遗传算法对于上述优化问题是比较容易的。$x = (x_1, x_2, \cdots, x_n)$的各个变量可以按二进制编码方法分别编码。因此，2.1 节介绍的单变量函数在闭区间上的函数优化方法，可以直接推广到多变量的情形。对于变量 x_i 的上、下限约束 $l_i \leqslant x_i \leqslant u_i (i = 1, 2, \cdots, n)$，依据解的精度要求(有效位数)求得各变量$(x_1, x_2, \cdots, x_n)$的二进制码位数$(m_1, m_2, \cdots, m_n)$，因此将 n 个二进制位串顺序连接起来，构成一个个体的染色体编码，编码的总位数 $m = m_1 + m_2 + \cdots + m_n$。

图 6.1 所示的是该优化问题的遗传编码构成。解码时仍按各个变量的编码顺序分别实现常规的二进制编码解码方法即可。除了采用标准的二进制编码方法外，还可以采用 Grey 编码

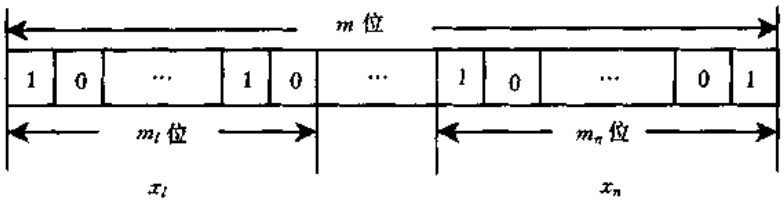

图 6.1 二进制遗传编码示意图

和实数编码方法。1975 年,De Jong 发表了题为"An Analysis of the Behavior of a Class of Genetic Adaptive Systems"的学位论文。主要论述了遗传算法应用于函数最优化的研究。函数最优化问题的复杂性依赖于:局部极小点的数目、局部极小点的分布、局部极小点的函数值的分布以及局部极小点的吸引域等。De Jong 根据函数优化相关的各种特性,如:①连续与非连续,②凸性与非凸性,③单峰性与多峰性,④二次函数与多次函数,⑤低维函数与高维函数,⑥确定的与随机的,从中精心挑选了五种函数最优化的测试例子,如表 6.1 所示。函数 F_1 是简单的平方求和函数,具有一个极小值在点 $x_i=0$;二维 Rosenbrock 函数 F_2 是单峰函数,但它是病态的且极难极小化;F_3 是不连续函数,是由整数阀值的和得到的,在五维空间中有一个极小值;F_4 是有噪声的四次函数;F_5 是多峰函数,具有 25 个局部极小值。图 6.2 是 $F_1 \sim F_5$ 的函数图形示意图。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发展的典范。

表 6.1 De Jong 的五种函数最小化问题

函数	上、下限约束
$F_1 = \sum_{i=1}^{3} x_i^2$	$-5.12 \leqslant x_i \leqslant 5.12$
$F_2 = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$	$-2.048 \leqslant x_i \leqslant 2.048$
$F_3 = \sum_{i=1}^{5} integer(x_i)$	$-5.12 \leqslant x_i \leqslant 5.12$
$F_4 = \sum_{i=1}^{30} ix_i^4 + Gauss(0,1)$	$-1.28 \leqslant x_i \leqslant 1.28$
$F_5 = 0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^2}$	$-65.536 \leqslant x_i \leqslant 65.536$

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

在环境 e 下策略 s 的在线性能(on-line performance)指标 $X_e(s)$定义为:

$$X_e(s) = \frac{1}{T}\sum_{t=1}^{T} f_e(t) \tag{6.3}$$

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。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。

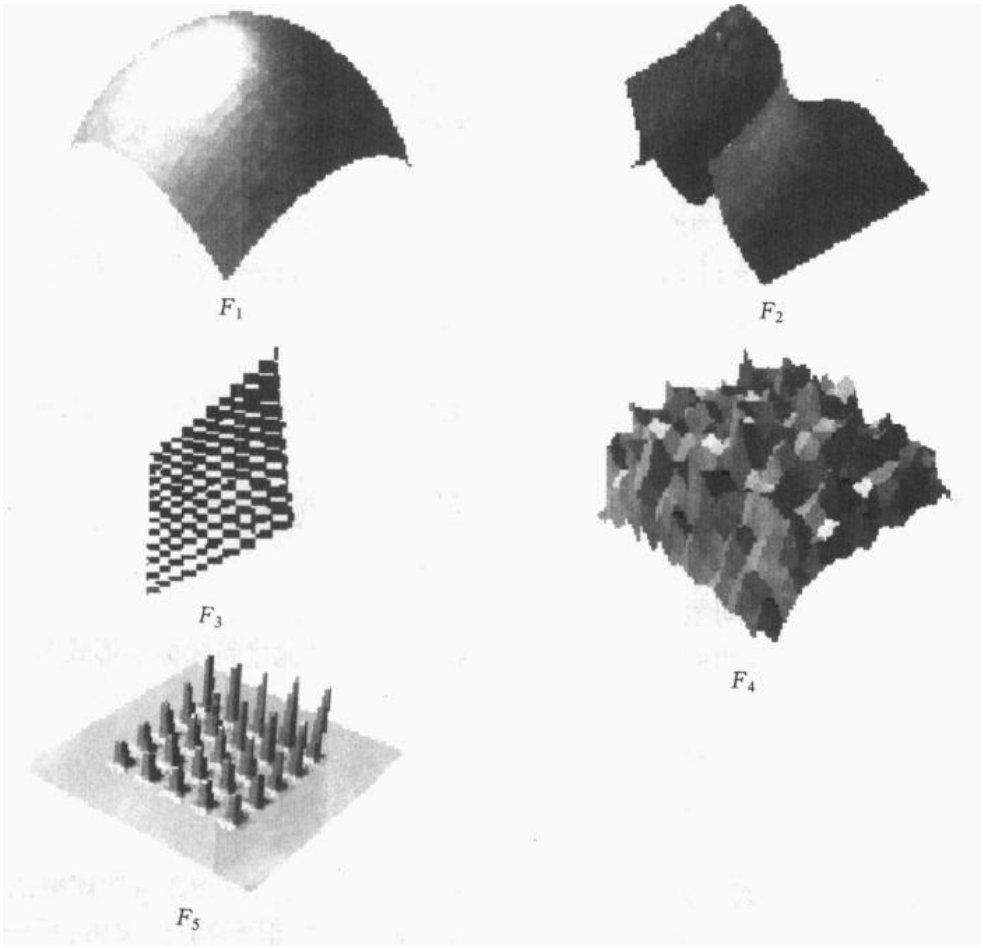

图 6.2　$F_1 \sim F_5$ 的函数图形示意图

在环境 e 下策略 s 的离线性能(off-line performance)指标 $X_e^*(s)$定义为：

$$X_e^*(s) = \frac{1}{T}\sum_{t=1}^{T} f_e^*(t) \tag{6.4}$$

式中，$f_e^*(t)$是在环境 e 下$[0, t]$时间内最好的目标函数或适应度，$f_e^*(t) = \text{best}\{f_e(1), f_e(1), \cdots, f_e(t)\}$。算法的离线性能表示了算法在运行过程中各进化世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 采用了以下一些研究方法：

(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**　包括了群体大小 M、交叉概率、变异概率、代沟 G，其中代沟 $G(0 < G < 1)$指子代与父代之间的重叠程度。

(3) **算法中的策略 s**　$s1$：基本遗传算法(比例选择、单点交叉、基本位变异)；

$s2$：精英保留模型(elitist model)；

$s3$：期待值模型(expected value model)；

$s4$：精英保留期待值模型(elitist expected value model)；

$s5$：排挤因子模型(crowding model)；

$s6$：广义交叉模型(generalized crossover model)。

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於随机搜索算法的性能。

结论 4　使用精英保留模型和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。

6.3　优化的约束问题解决

6.3.1　约束最优化问题

含不等式约束、等式约束及变量上、下限约束的约束最优化问题标准形式为

$$\begin{aligned}&\text{Minimize } f(x)\\&\left.\begin{aligned}&g_i(x)\leqslant 0,\quad i=1,\cdots,m\\&h_j(x)\leqslant 0,\quad j=1,\cdots,l\\&l_i\leqslant x_i\leqslant u_i\end{aligned}\right\}\end{aligned}\tag{6.6}$$

上述约束最优化问题(Constrained Optimization Problems, COPs)的常规解法可以分为两种途径：一种是把有约束问题化为无约束问题，再用无约束问题的方法去解；另一种是改进无约束问题的方法，使之能用于有约束的情况。第一种途径的历史很悠久，主要是罚函数法(penalty function method)，由 Couran 在 1949 年提出，后来由 Frish(1955)和 Carroll(1959)分别作了发展。罚函数法在实践中使用比较广泛。罚函数法的要点是把问题的约束函数以某种形式归并到目标函数上去，使整个问题变为无约束问题。这种方法对于非线性的约束，设计的算法常常因为迭代点要沿复杂的可行区域边界移动而花费大量的计算而可能导致失败。罚函数法根据解序列相对于原问题的可行性分为外部罚函数法(exterior penalty method)和内部罚函数法(interior penalty method 也称障碍法 barrier function method)。对序列罚函数法性质的进一步研究导出了一次求无约束问题来得到原问题解的恰当罚函数(exact penalty function)。罚函数法的更近发展是乘子法(multiplier method)或 Lagrange 法，以及投影 Lagrange 法。约束最优化问题的第二种途径发展较晚，20 世纪 60 年代 Rosen 对于带线性约束问题提出了著名的梯度投影算法(gradient projection method)。这类算法可以看成是无约束问题中最速下降法在含约束问题上的推广，其基本思想是把负梯度方向投影到可行方向集的一个子集上，取投影为可行下降方向。简约梯度法推广到非线性约束的情况，称为广义简约梯度法(generalized reduced gradient method, GRD)。

将遗传算法应用于约束最优化问题的关键是对约束条件的处理，由于等式约束可以包含到适应度函数中，所以仅需要考虑不等式约束。假设按无约束问题那样求解，在搜索过程中计算由算法产生的目标函数值，并检查是否有约束违反。如果没有违反，则表明是可行解，就根据目标函数值为其指定一个适应值；否则，就是不可行解，因而没有适应度值(适应度值为 0)。

除了许多实际问题是高度约束的之外，这样的处理实际上是行不通的，因为要找到一个可行解同样是很难的，因而需要从不可行解中得到一些信息。借鉴常规方法中的罚函数法是一方便的选择。将罚函数包含到适应度评价中，可以采用下列形式：

$$f(x) + rP(x) \tag{6.7}$$

式中，罚函数 $P(x)$ 为满足下列条件的连续函数：

$$P(x)\begin{cases} = 0, & x \in X \\ > 0, & x \notin X \end{cases} \tag{6.8}$$

r 为罚函数尺度系数，$r>0$。X 为问题的可行解域。

由于如何设计罚函数以有效地惩罚非可行解，对问题的解决至关重要。罚函数设计方面的研究一直被十分重视，对于不同的问题，人们提出了多种罚函数的形式，如静态罚函数、动态罚函数、退火罚函数、自适应罚函数、启发式罚函数、双重罚函数以及逐次惩罚罚函数等。此外，适于一些高级遗传操作算子也有一定的适用范围，如边界变异(boundary mutation)、启发式交叉(heuristic crossover)、几何交叉(geometrical crossover)、球面交叉(sphere crossover)等。总而言之，罚函数法对于不同的问题需要设计不同的罚函数，而且在约束数目及其复杂性小的情况下才比较适用；对于规模不大的线性约束最优化问题，有较好的应用效果。对于一般的约束处理，通常是很困难的。1997 年，Jan Paredis 提出了共同进化遗传算法(Coevolutionary Genetic Algorithm, CGA)解决一般的约束满足问题，其中一个种群由问题的解组成，另一个种群由约束组成。这两个种群协同进化，较好的解应满足更好的约束，而较优的约束则被更多的解所违背(见 8.4 节)。

下面将介绍 Michalewicz 在遗传算法应用于线性约束最优化问题方面的研究成果。

6.3.2 求解线性约束最优化问题的遗传算法

线性约束最优化问题的一般形式可以描述为：

$$\begin{aligned}
&\text{Minimize} \quad f(x_1, \cdots, x_n) \\
&\text{subject to} \\
&\left.\begin{aligned}
&a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\
&\qquad \vdots \\
&a_{m1}x_1 + \cdots + a_{mn}x_n = b_m \\
&c_{11}x_1 + \cdots + c_{1n}x_n \leqslant d_1 \\
&\qquad \vdots \\
&c_{l1}x_1 + \cdots + c_{ln}x_n \leqslant d_l \\
&l_i \leqslant x_i \leqslant u_i, \qquad i = 1, \cdots, n
\end{aligned}\right\}
\end{aligned} \tag{6.9}$$

以上问题的矩阵形式为：

$$\begin{aligned}
&\text{Minimize} \quad f(\boldsymbol{x}) \\
&\text{subject to} \quad \left.\begin{aligned}
&\boldsymbol{Ax} = \boldsymbol{b} \\
&\boldsymbol{Cx} \leqslant \boldsymbol{d} \\
&\boldsymbol{l} \leqslant \boldsymbol{x} \leqslant \boldsymbol{u}
\end{aligned}\right\}
\end{aligned} \tag{6.10}$$

式中，$\boldsymbol{x} = (x_1 \ \cdots \ x_n)^{\mathrm{T}}$，$\boldsymbol{A} = (a_{ij})$，$\boldsymbol{b} = (b_1 \ \cdots \ b_n)^{\mathrm{T}}$，$\boldsymbol{C} = (c_{kj})$，$\boldsymbol{d} = (d_1 \ \cdots \ d_n)^{\mathrm{T}}$，$\boldsymbol{l} = (l_1 \ \cdots \ l_n)^{\mathrm{T}}$，$\boldsymbol{u} = (u_1 \ \cdots \ u_n)^{\mathrm{T}}$，$(i = 1, \cdots, n;\ j = 1, \cdots, m;\ k = 1, \cdots, l)$。

上述约束优化的目标函数可以是线性函数或者非线性函数。1991 年 Michalewicz 等研究了这类约束最优化问题。通过消除可能的变量以减少变量数目，消除等式约束，并设计特别的遗传操作等手段，使线性约束最优化问题适合于遗传算法求解。这种遗传优化算法称之为 GENOCOP(genetic algorithm for numerical optimization for constrained problem 约束优化的遗传算法)。在介绍这种算法之前，我们给出一个简单的例子说明其基本思想。

下面是一个含 6 个优化变量的线性约束最优化问题：

$$\begin{aligned}&\text{Minimize} \quad f(x_1, x_2, x_3, x_4, x_5, x_6)\\&\text{subject to}\quad \left.\begin{aligned}&x_1 + x_2 + x_3 = 5\\&x_4 + x_5 + x_6 = 10\\&x_1 + x_4 = 3\\&x_2 + x_5 = 4\\&x_i \geqslant 0 \qquad i = 1, \cdots, 6\end{aligned}\right\}\end{aligned} \tag{6.11}$$

首先，根据 4 个独立的等式约束，将变量 x_3, x_4, x_5, x_6 用 x_1, x_2 来表示。

$$\left.\begin{aligned}&x_3 = 5 - x_1 - x_2\\&x_4 = 3 - x_1\\&x_5 = 4 - x_2\\&x_6 = 3 + x_1 + x_2\end{aligned}\right\} \tag{6.12}$$

目标函数表示为：

$$\bar{f}(x_1, x_2) = f(x_1, x_2, (5 - x_1 - x_2), (3 - x_1), (4 - x_2), (3 + x_1 + x_2)) \tag{6.13}$$

因此，通过消除多余变量和等式约束，约束条件转化为两个变量的不等式约束情形：

$$\left.\begin{aligned}&x_1 \geqslant 0, x_2 \geqslant 0\\&5 - x_1 - x_2 \geqslant 0\\&3 - x_1 \geqslant 0\\&4 - x_2 \geqslant 0\end{aligned}\right\} \tag{6.14}$$

即

$$\left.\begin{aligned}&0 \leqslant x_1 \leqslant 3\\&0 \leqslant x_2 \leqslant 4\\&x_1 + x_2 \leqslant 5\end{aligned}\right\} \tag{6.15}$$

然后，我们考虑遗传操作的方式。由于采用实数编码方式，变异操作比较简单，检查搜索空间中的一点 $\boldsymbol{x} = (x_1, x_2) = (1.8, 2.3)$，保持 x_2 的值使 x_1 发生变化(均匀变异)，变量 x_1 的变化区间为 $[0, 5 - x_2] = [0, 2.7]$。在考虑交叉操作时，假设搜索空间中存在两点 $\boldsymbol{x} = (x_1, x_2) = (1.8, 2.3)$，$\boldsymbol{x}' = (x_1', x_2') = (0.9, 3.5)$，其任意线性组合 $\lambda\boldsymbol{x} + (1 - \lambda)\boldsymbol{x}'(0 \leqslant \lambda \leqslant 1)$也为搜索空间中的一点，并且满足所有约束条件，从而实现了交叉操作。由于搜索空间是凸性的，设计这样的遗传操作使解向量限制在可行解域内。

一般地，GENOCOP 算法对约束条件处理步骤如下：

(1) **消除等式约束** 假设等式约束 $\boldsymbol{Ax} = \boldsymbol{b}$ 中有 m 个独立的等式，m 个变量 $x_{i_1}, x_{i_2}, \cdots, x_{i_m}(\{i_1, \cdots, i_m\} \subseteq \{1, 2, \cdots, n\})$，可以用剩余的 $n - m$ 个变量表示。等式消除的操作可以描述如下：

将矩阵 A 在第 j 列($j \in \{i_1, \cdots, i_m\}$)处分割为两个分矩阵 A_1 和 A_2，类似地分割矩阵 C、向量 l 和 u，对应分矩阵和向量加下标表示。这样，等式约束成为：

$$A_1 x^1 + A_2 x^2 = b \tag{6.16}$$

由于 ${A_1}^{-1}$存在，$x^1 = {A_1}^{-1} b - {A_1}^{-1} A_2 x^2$，这样，变量 $x_{i_1}, x_{i_2}, \cdots, x_{i_m}$ 可用剩余的变量的线性组合表示。对于变量 $x_{i_j}(j = 1, \cdots, m)$的上、下限约束 $l_{i_j} \leqslant x_{i_j} \leqslant u_{i_j}$，去掉其中所有的 x_{i_j}，有下列新的不等式成立：

$$l_1 \leqslant {A_1}^{-1} b - {A_1}^{-1} A_2 x^2 \leqslant u_1 \tag{6.17}$$

加上原问题中不等式 $Cx \leqslant d$，即：

$$C_1 x^1 + C_2 x^2 \leqslant d \tag{6.18}$$

将 x^1 代入上式转化为：

$$C_1({A_1}^{-1} b - {A_1}^{-1} A_2 x^2) + C_2 x^2 \leqslant d \tag{6.19}$$

因此，将 m 个变量 $x_{i_1}, x_{i_2}, \cdots, x_{i_m}$ 消除后，最终的约束由以下的不等式约束组成：

① 原上下限约束　　$l_2 \leqslant x^2 \leqslant u_2$　　(6.20)

② 新增加的不等式约束　　$A_1 l_1 \leqslant b - A_2 x^2 \leqslant A_1 u_1$　　(6.21)

③ 原不等式　　$(C_2 - C_1 {A_1}^{-1} A_2) x^2 \leqslant d - C_1 {A_1}^{-1} b$　　(6.22)

(2) GENOCOP 算法　为将 GENOCOP 算法与常规约束最优化方法比较，Michalewicz 等考虑了下面的 7×7 的运输规划问题作为实例。

$$\begin{aligned}
&\text{Minimize} \quad f(x) = f(x_1, x_2, \cdots, x_{49}) \\
&\text{subject to} \\
&\left.\begin{aligned}
&x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 = 27 \\
&x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} = 28 \\
&x_{15} + x_{16} + x_{17} + x_{18} + x_{19} + x_{20} + x_{21} = 25 \\
&x_{22} + x_{23} + x_{24} + x_{25} + x_{26} + x_{27} + x_{28} = 20 \\
&x_{29} + x_{30} + x_{31} + x_{32} + x_{33} + x_{34} + x_{35} = 20 \\
&x_{36} + x_{37} + x_{38} + x_{39} + x_{40} + x_{41} + x_{42} = 20 \\
&x_{43} + x_{44} + x_{45} + x_{46} + x_{47} + x_{48} + x_{49} = 20 \\
&x_1 + x_8 + x_{15} + x_{22} + x_{29} + x_{36} + x_{43} = 20 \\
&x_2 + x_9 + x_{16} + x_{23} + x_{30} + x_{37} + x_{44} = 20 \\
&x_3 + x_{10} + x_{17} + x_{24} + x_{31} + x_{38} + x_{45} = 20 \\
&x_4 + x_{11} + x_{18} + x_{25} + x_{32} + x_{39} + x_{46} = 20 \\
&x_5 + x_{12} + x_{19} + x_{26} + x_{33} + x_{40} + x_{47} = 20 \\
&x_6 + x_{13} + x_{20} + x_{27} + x_{34} + x_{41} + x_{48} = 20 \\
&x_7 + x_{14} + x_{21} + x_{28} + x_{35} + x_{42} + x_{49} = 20 \\
&x_i \geqslant 0, \quad i = 1, 2, \cdots, 49
\end{aligned}\right\} \qquad (6.23)
\end{aligned}$$

对于非线性目标函数的构造，考虑下面几种测试函数 $A(x) \sim F(x)$(如图 6.3 所示)。

① 函数 $A(x)$

$$A(x)=\begin{cases}0 & 0<x\leqslant S\\ c_{ij} & S<x\leqslant 2S\\ 2c_{ij} & 2S<x\leqslant 3S\\ 3c_{ij} & 3S<x\leqslant 4S\\ 4c_{ij} & 4S<x\leqslant 5S\\ 5c_{ij} & 5S<x\end{cases} \tag{6.24}$$

式中，S 小于一个典型的 x 值。

② 函数 $B(x)$

$$B(x)=\begin{cases}c_{ij}\dfrac{x}{S} & 0<x\leqslant S\\ c_{ij} & S<x\leqslant 2S\\ c_{ij}(1+\dfrac{x-2S}{S}) & 2S<x\end{cases} \tag{6.25}$$

式中，S 和典型的 x 具有同样的阶。

③ 函数 $C(x)$

$$C(x)=c_{ij}x^2 \tag{6.26}$$

④ 函数 $D(x)$

$$D(x)=c_{ij}\sqrt{x} \tag{6.27}$$

⑤ 函数 $E(x)$

$$E(x)=c_{ij}\left(\frac{1}{1+(x-2S)^2}+\frac{1}{1+(x-\frac{9}{4}S)^2}+\frac{1}{1+(x-\frac{7}{4}S)^2}\right) \tag{6.28}$$

式中，S 和典型的 x 具有同样的阶。

⑥ 函数 $F(x)$

$$F(x)=c_{ij}x\left(\sin\left(x\,\frac{5\pi}{4S}\right)+1\right) \tag{6.29}$$

式中，S 和典型的 x 具有同样的阶。

非线性运输问题的目标函数为：

$$\sum_{i,j}f(x_{ij})+P \tag{6.30}$$

这里，$f(x_{ij})$取测试函数 $A(x)\sim F(x)$中任一函数，P 为罚函数。

$$P=k\cdot\left(\frac{t}{T}\right)^p\cdot\bar{f}\cdot\sum_{i=1}^{14}d_i \tag{6.31}$$

式中，$\bar{f}$ 为第 t 代群体的平均适应度。k 和 p 为参数，取 $k=1$，$p=1/14$。T 为最大运行代数，d_i 为第 i 个约束的违反度。

对于约束 $\sum_{i\in W}x_i=val$，$W\subseteq\{1,\cdots,49\}$，个体的染色体表示为$(v_1,\cdots,v_{49})$，其约束违反度定义为：

$$d_i=\left|\sum_{i\in W}v_i-val\right| \tag{6.32}$$

7×7 费用参数 c_{ij}取表 6.2 所列数据。

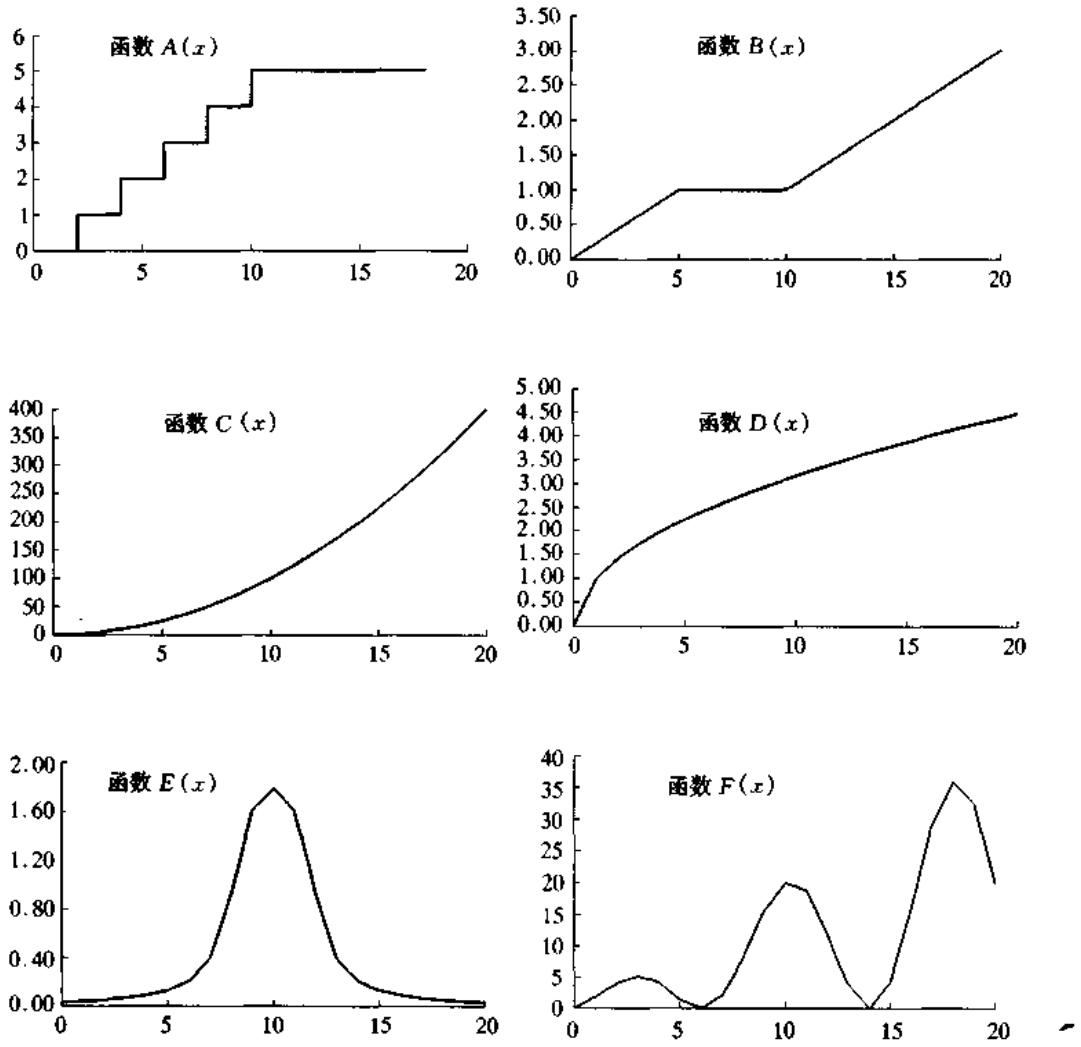

图 6.3　六种测试函数图形

表 6.2　7×7 费用参数表

	27	28	25	20	20	20	20
20	0	21	50	62	93	77	1 000
20	21	0	17	54	67	1 000	48
20	50	17	0	60	98	67	25
23	62	54	60	0	27	1 000	38
26	93	67	98	27	0	47	42
25	77	1 000	67	1 000	47	0	35
26	1 000	48	25	38	42	35	0

为推导 S，需估计 x 的典型值。可通过初步运行的方式估算 x_{ij}的数目和大小。用这种方式可估算每个弧线的平均流量，找到 S 的值。对于函数 A 使用 $S=2$，对于 B，E 和 F 使用 $S=5$。

由于该规划问题含有 13 个独立的等式约束，因此，可以消除 13 个变量。被消除的 x_1，x_2，…，x_8，x_{15}，x_{22}，x_{36}，x_{44}，其余的 36 个变量按顺序设定为 y_1，y_2，…，y_{36}，$y_1 = x_9$，$y_2 = x_{10}$，…，$y_{36} = x_{49}$。按照等式约束消除的方法，将原规划问题转化为：

$$0 \leqslant y_1 \leqslant 20 \tag{6.33}$$

$$93 - \sum_{i=2}^{30} y_i + y_{31} \leqslant y_1 \leqslant 113 + y_{31} - \sum_{i=2}^{30} y_i \tag{6.34}$$

$$\sum_{i=31}^{36} y_i - 20 - y_7 - y_{13} - y_{19} - y_{25} \leqslant y_1 \leqslant \sum_{i=31}^{36} y_i - 20 - y_7 - y_{13} - y_{19} - y_{25} \tag{6.35}$$

分别以 $A(x) \sim F(x)$测试函数构成的目标函数，设定群体大小 40，均匀变异概率 0.08，边界变异概率 0.03，非均匀变异概率 0.07，简单交叉概率 0.10，单一算术交叉概率 0.10，全体交叉概率 0.10，实现 GENOCOP 算法，进行演算 8 000 代的试验。表 6.3 所示列出了中间世代优化结果。

表 6.3 中间世代优化结果

函数	世代					
	1	500	1 000	2 000	4 000	8 000
A	1 085.8	273.4	230.5	153.0	94.5	24.15
B	932.4	410.6	258.4	250.3	209.2	205.60
C	83 079.1	14 122.7	4 139.0	2 944.1	2 772.7	2 571.04
D	1 733.6	575.3	575.3	480.2	480.2	480.16
E	225.2	204.9	204.9	204.9	204.9	204.82
F	3 799.3	1 719.0	550.8	320.9	166.4	119.61

对于非线性目标函数的运输问题，获取大范围全局最优解必须满足目标函数为凸性的要求，而采用常规方法容易收敛于局部最优解。Michalewicz 等将基于拟牛顿法的非线性最优化算法 GAMS 与 GENOCOP 算法进行比较，比较结果见表 6.4。

表 6.4 GAMS 与 GENOCOP 算法比较

函数	GAMS	GENOCOP	误差%
A	96.00	24.15	297.52
B	1 141.60	205.60	455.25
C	2 535.29	2 571.04	-1.41
D	565.15	480.16	17.70
E	208.25	204.82	1.67
F	43 527.54	119.61	36 291.22

从计算结果分析，GENOCOP 算法比商品化的软件包 GAMS 获得的结果要理想。有趣的是，对于费用函数 $A(x)$，$B(x)$以及特别"不规则的"费用函数 $F(x)$，GENOCOP 算法要好得多。而对于其他费用函数 $C(x)$，$D(x)$和 $E(x)$，两种方法的结果比较相近。

6.4　多目标优化问题

工程中经常会遇到在多准则或多设计目标下设计和决策的问题，如果这些目标是相背的，需要找到满足这些目标的最佳设计方案。解决含多目标和多约束的优化问题，即多目标优化(Multi-Objective Optimization, MO)。通常的做法是根据某效用函数将多目标合成单一目标来进行优化。但大多数情况下，在优化之前这种效用函数是难以确知的。这样为了使决策者深入掌握优化问题的特点，有必要提供多个解以便于作出合理的最终选择。

法国经济学家 V. Pareto(1848～1923)最早研究经济领域内的多目标优化问题，他的理论被称为 Pareto 最优性理论。用求单目标优化的方法求最优解，获得的所谓理想解往往在可行域之外。MO 问题需要优化一组费用函数，其解不是单一点，而是一组点的集合，称之为 Pareto 最优集(Pareto－optimal set)。Pareto 最优集定义如下：

对于最小化 MO 问题，n 个目标分量 $f_k(k=1,\cdots,n)$组成的向量 $\bar{f}(\bar{x})=(f_1(\bar{x}), f_2(\bar{x}), \cdots, f_n(\bar{x}))$其中 $\bar{x}_u \in U$ 为决策变量，若 $\bar{x}_u$ 为 Pareto 最优解满足：

当且仅当，不存在决策变量 $\bar{x}_v \in U$，$v=f(\bar{x}_v)=(v_1,\cdots,v_n)$支配 $u=f(\bar{x}_u)=(u_1,\cdots,u_n)$即不存在 $\bar{x}_v \in U$ 使得下式成立：

$$\forall i \in \{1,\cdots,n\}, v_i \leqslant u_i \wedge \exists i \in \{1,\cdots,n\} \mid v_i < u_i \tag{6.36}$$

常规 MO 问题求解方法有多目标加权法、层次优化法、ε－约束法、全局准则法、目标规划法等，其中以目标规划法(goal programming)最为著名。这些算法的特点是将多目标转化为单目标处理，往往只能得到一个解。除了预先获知目标函数最优值的情况外，不能保证 Pareto 最优性，即使最优化求解很成功。

图 6.4 所示的是 MO 问题解集空间示意图，Pareto 最优集的每个解都是 MO 问题的一个非劣解。遗传算法通过代表整个解集的种群进化，以内在并行的方式搜索多个非劣解，决策者可以在多个解中选择决策方案，这对于解决 MO 问题是非常诱人的。下面在分析适用于多目标优化的遗传算法基础上，讨论一种结合决策偏好与 Pareto 秩的多目标遗传算法(Multi－

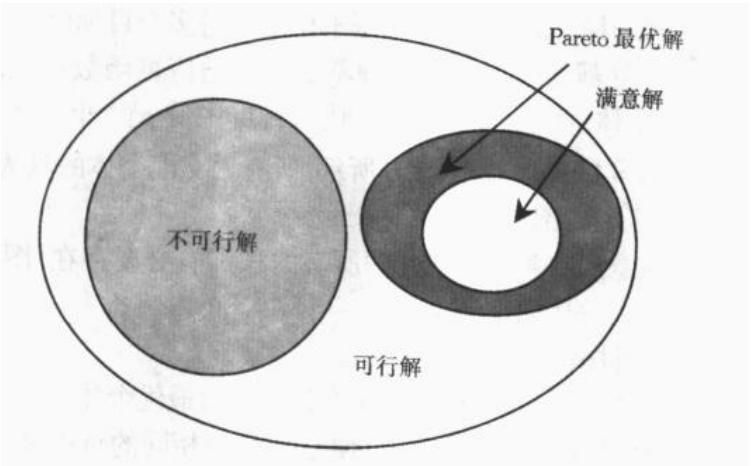

图 6.4　MO 问题解集空间示意图

Objective Genetic Algorithms, MOGA)。

6.4.1 多目标优化的遗传算法

1. 非 Pareto 方法

Schaffer(1985)研究多目标优化时，在扩展 SGA(Simple Genetic Algorithms)时提出了向量形式的适应度计算方法，称为向量评价的遗传算法(Vector Evaluated Genetic Algorithm, VEGA)。其选择方法作了修改，在每一代，基于各目标函数的计算，用适应度比例法产生一定数目的子种群。假定种群的大小为 N，目标函数个数为 q，将产生 q 个子种群，子种群大小为 N/q。然后将其混合起来形成新的一代，继续执行交叉和变异的遗传操作，VEGA 方法如图 6.5 所示。Richardson 等指出这种将所有个体混合起来的做法等价于将适应度函数线性求和，只不过权重取决于当前的世代。

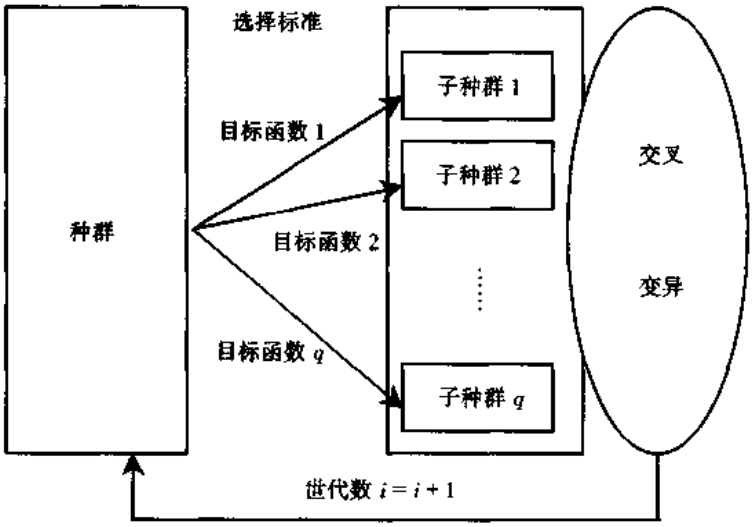

图 6.5 VEGA 方法

2. Pareto 方法

Goldberg [3]提出来一种 Pareto 方法，其基本想法是将多个目标值直接映射到适应度函数中，于是在 Pareto 最优集的基础上提出了一种基于秩的适应度函数形式，先将多目标函数值组成一个向量代表一个个体，假定个体 x_i 在 t 代时种群中有 $p_i^{(t)}$ 个个体支配于它，则它在种群内的秩为：$rank(x_i, t) = 1 + p_i^t$。如图 6.6 所示，所有无支配个体的秩为 1，个体 3 劣于个体 2，因为个体 2 落在折衷区域内部。

关于适应度计算，注意到在某一世代并非所有的秩都有必要存在，图中序位 4 是缺失的。因此，按秩对适应度计算方法作如下调整：

① 种群内个体进行排序；

② 按线性插值的方法计算个体适应度，序位为 1 者为最优个体。

计算具有相同秩的个体平均适应度值，以便它们具有相同的选中概率。这样在适当选择压力的前提下可以保持整个种群的适应度值为常量。这种方法的关键是求各个体的秩，在秩的基础上得到的适应度是均匀分布的，但求秩的过程一般十分复杂，运算量大，特别当种群规

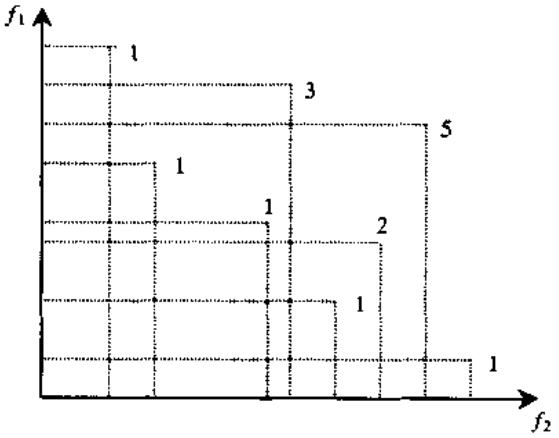

图 6.6　种群中个体的 Pareto 秩

模很大时计算耗时多。

J. Horn 和 N. Nafpliotis 提出了不同的选择方法，有点类似于竞争选择法。在种群中随机地挑选两个候选个体，在种群中另外随机地挑选一个个体参照集。然后候选个体一一与参照集中的个体作比较。如果一个个体劣于参照集，另一个优于参照集，则后者被选作为再生对象。如果两个个体均优于或均劣于参照集，必须采用分享法(sharing method)以确定哪一个是选中的个体。但并非所有情况下都要在给定的候选集中决定出一个个体。例如两个个体正好处于当前的无支配前沿(Non-dominated frontier)时两个个体之间不存在优劣。为了防止种群收敛到 Pareto 前沿的单一区域，当两个个体之间无偏好时引入一种适应度分享法，保持 Pareto 前沿的遗传多样性。分享操作目的是减少相似个体的复制量以尽可能保持种群的多样性，从而达到同时探索多个区域的目的。该算法被称为 Niched Pareto Genetic Algorithms，即 NPGA。

3. 结合目标值及其优先级偏好信息的 Pareto 方法

Charnes 和 Cooper(1961)以及后来的 Ijiri(1965)对目标规划法进行了系统研究。决策者预先为各目标函数确定好目标值，目标值可以是理想值也可以不是理想值，然后按照目标的重要程度给一个权重系数。优化的目标函数被表示成目标偏差值的最小化。目标约束区别于一般的约束条件，它实际上是决策者预期满足的，但微小的偏差也是可以接受的。受目标规划的影响，在 MO 问题中目标和目标优先级是比较容易得到的偏好信息。不同的目标给定相同的优先级，可以避免使用目标函数的距离测度，而距离测度不可避免地依赖于具体问题中给定目标值的大小。Carlos M. Fonseca 和 Peter J. Fleming 提出了结合偏好信息的关系算子，整个种群的个体排序按照定义的关系算子进行。

与单目标的情况相反，多目标排序不是唯一的。这是由于支配性和偏好性的概念只定义了全局秩没有定义部分秩。两个目标 f_1 和 f_2 的目标偏好值分别设置为 g_1 和 g_2，当两个目标优先级相同时，个体的排序结果如图 6.7(a)；当 f_2 比 f_1 的优先级高时，个体的排序结果如图 6.7(b)所示。这种方法依赖于决策者提供的目标值及其优先级偏好信息，在某种程度上仍然取决于决策者对问题的把握程度，最终得到的一组 Pareto 最优解在实际问题中不会是等价的，还需要决策者来遴选。

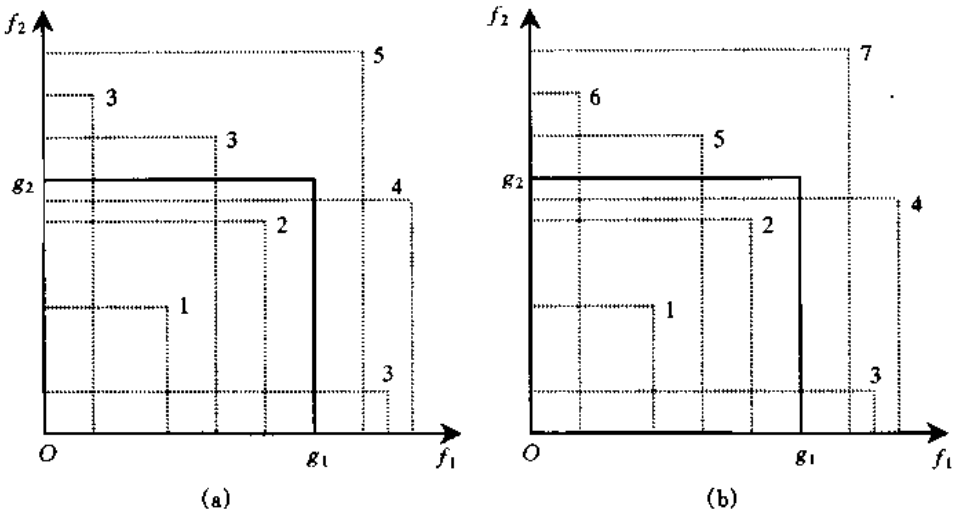

图 6.7 种群中个体的 Pareto 秩
(a) f_1 与 f_2 优先级相同； (b) f_2 比 f_1 优先级大

6.4.2 结合非精确目标权重偏好信息与 Pareto 秩的交互式多目标遗传算法

我们知道，多目标加权法将每个目标函数值乘上一个权重，然后加起来作为一个目标，采用单目标方法优化求取最优解。目标的权重作为决策者的一种偏好信息，一般很难预先确定。因此，我们提出由决策者交互优化过程产生非精确的目标权重引导遗传搜索的一种新的多目标遗传算法。根据决策者预先比较为数较少的可行解，获得非精确的权重信息，然后按照一般的 Pareto 秩构造适应度函数，进行遗传操作，逐代搜索获得无支配前沿，最终获得满足决策偏好的满意解，而不是一组无所适从的 Pareto 最优解。

对目标 $a_k(k=1,2,\cdots,q)$正规化处理，设 $a_k^{\max}$ 和 $a_k^{\min}$ 分别为目标 a_k 最大值和最小值，若 a_k 为效益型，线性正规化后目标值 $v_k(a_k)$为：

$$v_k(a_k)=\frac{a_k-a_k^{\min}}{a_k^{\max}-a_k^{\min}} \tag{6.37}$$

若 a_k 为成本型，线性正规化后目标值 $v_k(a_k)$为：

$$v_k(a_k)=\frac{a_k-a_k^{\max}}{a_k^{\min}-a_k^{\max}} \tag{6.38}$$

可行解 x 的多目标函数为：

$$v_x=\sum_k w_k v_k(a_k)_x \qquad \text{其中，} w_k>0,\ \sum_{k=1}^{q} w_k=1 \tag{6.39}$$

若已知两个可行解 x_1 和 x_2 且 $x_1 \succ x_2$ 则 $V_{x_1}>V_{x_2}$，即

$$\sum_{k=1}^{q} w_k[v(a_k)_{x_1}-v(a_k)_{x_2}]>0 \tag{6.40}$$

另一可行解 x_3，欲判断 x_3 与 x_1 之优劣，只需求解下面线性规划问题：

最小化
$$z=\sum_{k=1}^{q} w_k[v(a_k)_{x_3}-v(a_k)_{x_2}] \tag{6.41}$$

约束条件

$$\sum_{k=1}^{q} w_k [v(a_k)_{x_1} - v(a_k)_{x_2}] > 0 \tag{6.42}$$

$$w_k > 0, \quad \sum_{k=1}^{q} w_k = 1 \tag{6.43}$$

若以上规划问题解存在，则有：

$$z_{\min} \begin{cases} > 0 & x_3 \succ x_1 \\ = 0 & x_3 \sim x_1 \\ < 0 & x_3 \prec x_1 \end{cases} \tag{6.44}$$

在上述判别中权重作 $w_k (k = 1, 2, \cdots, q)$ 为一种非精确偏好信息，包含在可行解的比较中，我们称之为非精确偏好包含。据此构造一种交互式多目标遗传算法：

第 1 步　随机产生代表若干个可行解的初始种群，将个体的目标值作正规化处理。

第 2 步　有差异地挑选几个个体，由决策者进行比较判别优劣性，产生一组非精确偏好包含的约束。若不能进行有差异的挑选，即认为已经获得一组满意解，即停止。

第 3 步　建立线性规划模型的进行个体比较，对当前种群的个体进行排序。

第 4 步　依据个体的 Pareto 秩，计算适应度值，并进行分享操作。

第 5 步　选择个体，完成交叉、变异的遗传操作，产生新一代个体。

第 6 步　世代更迭，每隔一定代数，需要执行非精确偏好包含，转第 2 步。

第 7 步　若代数超过一定数目，则停止；否则转向第 3 步。

6.4.3　一个多目标优化问题计算实例

下面以 J. Horn 和 N. Nafpliotis 提出的一个简单多目标优化问题为例，假定 S_l 为固定长度 l 的二进制串，对 S_l 而言有两个目标，一是串中含 1 的个数，表示为 $U[S_l]$，二是串中 10 和 01 的对数，表示为 $P[S_l]$。例如 $S_8 = 11110101$，有 $U[S_8] = 6$，$P[S_8] = 4$。该问题是给定串长度，求取串变量使 $U[S_l]$ 和 $P[S_l]$ 同时最大化。取 Niche 半径为 5，选择方法为锦标赛选择，交叉率为 0.7，变异率为 0.3，按 NPGA 算法，第 200 代个体分布统计如图 6.8 所示（种群大小为 400），最终解几乎是 Pareto 最优解的全集。设定串长 28，种群大小为 100，开始在初始群体中选择

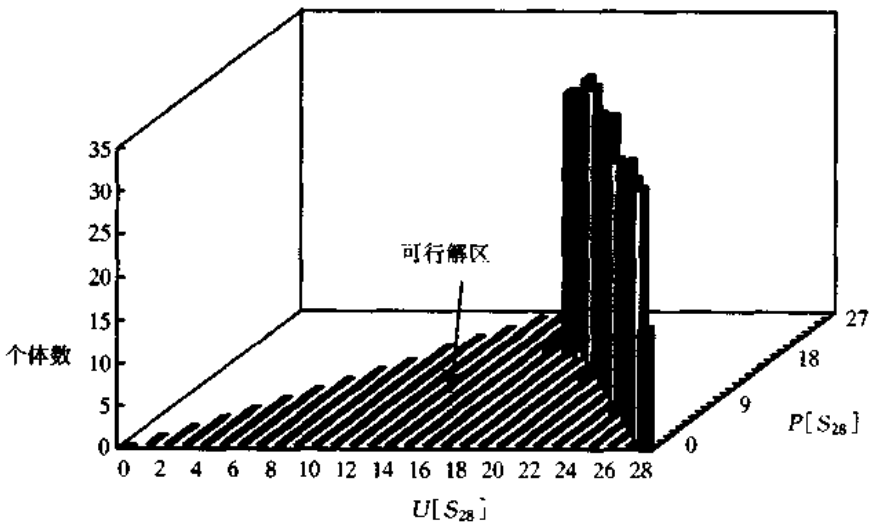

图 6.8　NPGA 算法，第 200 代个体分布

三个个体进行非精确偏好包含，如表 6.5 所示，表中前 3 个个体分别用 x_1、x_2 和 x_3 表示。

表 6.5 初始种群中的个体

NO.	个体	$U[S_{28}]$	$P[S_{28}]$	$v(U[S_{28}])$	$v(P[S_{28}])$
1	1011101101010101110110001001	16	18	0.571	0.667
2	0110010110111011010101111001	17	17	0.607	0.630
3	0010101000000110101101010101	12	20	0.429	0.740
4	1000001110001010110011001101	13	14	0.464	0.518
⋮	⋮	⋮	⋮	⋮	⋮
100	0100111110001010110011001111	16	13	0.571	0.481

如果选择 $x_3 \prec x_1 \prec x_2$，即包含 $P[S_{28}]$比 $U[S_{28}]$更重要的偏好信息，经过 200 代进化运算，获得的最终代 100 个个体统计如表 6.6 所示，其中 21 个个体的目标值 $P[S_{28}]=20$，$U[S_{28}]=18$；49 个个体目标值 $P[S_{28}]=22$，$U[S_{28}]=17$；30 个个体的目标值 $P[S_{28}]=24$，$U[S_{28}]=16$。显然，作为最终解的个体的 $P[S_{28}]$很大。

表 6.6 第 200 代种群个体统计($x_3 \succ x_1 \succ x_2$)

NO.	个体	$U[S_{28}]$	$P[S_{28}]$	$v(U[S_{28}])$	$v(P[S_{28}])$
1 2 ⋮ 49	1010101101010110101011101101 1101010101011010101110110101 ⋮ 1010101010110101011011101011	17	22	0.607	0.815
50 51 ⋮ 70	1101110101010110101101110101 0111011010101110101110101011 ⋮ 1011010101011010111110110101	18	20	0.642	0.740
71 72 ⋮ 100	1010101010101101010110110101 1101101010101010101010110101 ⋮ 1010110110101010101010110101	16	24	0.571	0.889

如果选择 $x_2 \succ x_1 \succ x_3$，即包含 $U[S_{28}]$比 $P[S_{28}]$更重要的偏好信息，经过 200 代进化运算，获得的最终代 100 个个体统计如表 6.7 所示，其中 8 个个体的目标值 $P[S_{28}]=0$，$U[S_{28}]=28$；92 个个体目标值 $P[S_{28}]=2$，$U[S_{28}]=27$。显然，作为最终解的个体的 $U[S_{28}]$很大。

表 6.7 第 200 代种群个体统计($x_2 \succ x_1 \succ x_3$)

NO.	个体	$U[S_{28}]$	$P[S_{28}]$	$v(U[S_{28}])$	$v(P[S_{28}])$
1 2 ⋮ 8	1111111111111111111111111111 1111111111111111111111111111 ⋮ 1111111111111111111111111111	28	0	1	0
9 10 ⋮ 100	1111111111101111111111111111 1110111111111111111111111111 ⋮ 1111101111111111111111111111	27	2	0.964	0.074

图 6.9 和图 6.10 所示，两种情况下最终解都处于 Pareto 前沿，而且与决策者的偏好非常吻合。实际上是由于遗传进化中保持了决策者非精确偏好信息而演化的结果。

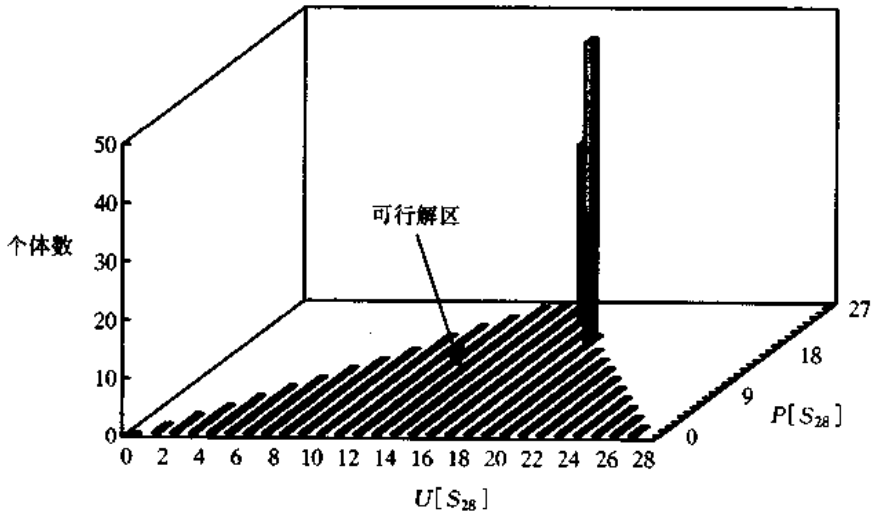

图 6.9　第 200 代个体分布($x_3 \succ x_1 \succ x_2$)

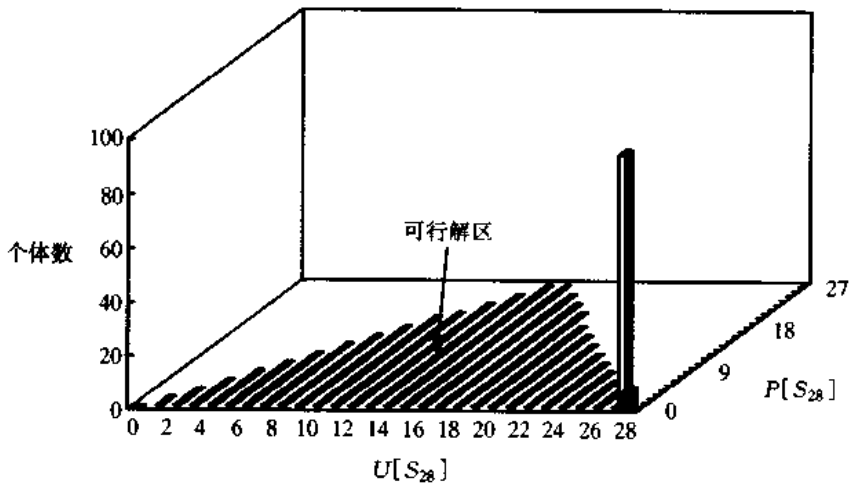

图 6.10　第 200 代个体分布($x_2 \succ x_1 \succ x_3$)

遗传算法的内在并行机制及其全局优化的特点适合于多目标优化问题的解决，特别是目标函数多、数学表达式非线性或者不明确、优化变量多、常规方法难以奏效的复杂场合，如控制系统优化设计。基于 Pareto 秩的 MOGA 有利于寻找最优解集，结合决策者的偏好信息构造适应度值，有利于寻求满意解集。本文提出的交互式多目标遗传算法以较容易的方式获得非精确偏好信息，据此世代进化获得与决策者意愿吻合的解集。此外，分享法(Sharing method)对于保持 Pareto 前沿的遗传多样性往往是必需的，通过减少相似个体的复制量达到同时探索多个区域的目的。

参考文献

[1] Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs. Springer－Verlag, Second, Extended Edition, 1994

[2] Michalewica Z, Janikow C. Handling Constraints in Genetic Algorithms. In: Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, 1991, 151～157

[3] De Jong K A. An Anlysis of the Behavior of a Class of Genetic Adaptive Systems. Doctoral Dissertation, University of Michigan, 1975

[4] Macready W G, Wolpert D H. What Makes an Optimization Problem Hard? SFI－TR－95－05－046, the Santa Fe Institute, 1995

[5] Houck C R, Joines J A. A Genetic Algorithm for Function Optimization: A MATLAB Implementation. NCSU－IE TR95－09, 1995

[6] Myung H, Kim J H. Lagrangian－Based Evolutionary Programming for Constrained Optimization. In: Simulated Evolution and Learning, First Asia－Pacific Conference, SEAL'96, Taejon, Korea, Springer, 1996, 35～44

[7] Pan Z, Kang L. An Adaptive Evolutionary Algorithm for Numerical Optimization. In: Simulated Evolution and Learning, First Asia－Pacific Conference, SEAL'96, Taejon, Korea, Springer, 1996, 27～34

[8] Krishnakumar K. Solving Large Parameter Optimization Problems Using A Genetic Algorithm with Stochastic Coding. In: Genetic Algorithms in Engineering and Computer Science, Winter G(ed.), Wiley, 1995, 287～301

[9] Smith A, Tate D. Genetic Optimization Using a Penalty Function, Proceedings of 5th International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, 1993, 499～503

[10] Beasley D, Bull D R. A Sequencing Niche Technique for Multimodal Function Optimization. Evolutionary Computation, 1993, 1(2): 101～125

[11] 徐光辉. 运筹学基础手册. 北京: 科学出版社, 1999

[12] 丁承民, 张传生等. 正交试验遗传算法及其在函数优化中的应用. 系统工程与电子技术, 1997(10): 57～60

[13] 樊叔维, 汪洁等. 遗传算法在电力变压器和电机全局优化设计中的应用研究. 西安交通大学学报, 1996, 30(6): 15～21

[14] 苟先太, 金炜东. 有约束优化中遗传算法的应用. 西南交通大学学报, 1997, 32(4): 433～437

[15] 鹿跃丽, 周力平. 遗传算法用于工程结构优化设计的研究. 郑州工业大学学报, 1998, 19(2): 35～39

[16] 侯格贤, 吴成柯. 遗传算法的性能分析. 控制与决策, 1999, 14(3): 257～260

[17] 刘铁南, 陈广义, 刘延力. 模拟生物种族形成的进化算法与多峰函数优化. 控制与决策, 1999, 14(3): 185～188