进化算法

云庆夏 编著

冶金工业出版社

国家自然科学基金资助项目

进 化 算 法

云庆夏 编著

北 京
冶 金 工 业 出 版 社
2000

内 容 简 介

进化算法是一种新兴的搜索寻优技术。它根据生物中遗传与进化的原理，仿效基因、染色体等物质表达所研究的问题，遵循达尔文"物竞天择，适者生存"原则，使随机生成的初始解通过复制、交换、突变等遗传操作不断迭代进化，逐步逼近最优解。从实质上讲，进化算法是生物科学与工程技术相结合的一门边缘学科。目前，它已成为继专家系统、人工神经网络之后有关人工智能学科的第三个研究热点。

进化算法包括遗传算法、遗传规划、进化策略、进化规划4种方法。本书分别介绍它们的基本知识、基本理论及实施技术。全书共分5章，分别为绪论、遗传算法、遗传规划、进化策略、进化规划。

本书论述深入浅出，先从简单的示例介绍每种算法的原理，然后深入讨论它们的基本理论及应用技术。书中图文并茂，便于自学，适合于程技术人员、科研人员阅读，也可作为大专院校的教材或参考书。

图书在版编目(CIP)数据

进化算法/云庆夏编著. —北京：冶金工业出版社

2000.5

ISBN 7-5024-2574-8

Ⅰ.进… Ⅱ.云… Ⅲ.遗传工程-计算方法

Ⅳ.Q78-32

中国版本图书馆CIP数据核字(2000)第16903号

出版人 卿启云(北京沙滩嵩祝院北巷39号，邮编100009)

责任编辑 田 锋 美术编辑 熊晓梅 责任校对 白 迅 责任印制 牛晓波

北京源海印刷厂印刷；冶金工业出版社发行；各地新华书店经销

2000年5月第1版，2000年5月第1次印刷

850mm×1168mm 1/32；6.5印张；174千字；199页；1-2500册

18.00元

冶金工业出版社发行部 电话：(010)64044283 传真：(010)64044283

冶金书店 地址：北京东四西大街46号(100711) 电话：(010)65289081

(本社图书如有印装质量问题，本社发行部负责退换)

前　言

1859 年达尔文创立的进化论，曾经作为生物界以及人类文明史上的一个里程碑，促进了科学技术的发展。20 世纪 60 年代以来，生物学的进化论又被推广应用于工程技术，形成一种新型的计算方法---进化算法(Evolutionary Algorithms)，又称进化计算(Evolutionary Computation)。

进化算法仿效生物学中进化和遗传的过程，遵从"生存竞争，优胜劣汰"的原则，从一组随机生成的初始可行群体出发，借助复制、交换(重组)、突变等遗传操作，逐步逼近所研究问题的最优解。从实质而言，进化算法是一种具有自适应调节功能的搜索寻优技术。

目前，进化算法已成为继人工智能专家系统、人工神经网络之后的又一个受人青睐的新学科。在美国、德国、法国等发达国家，它已被成功地用于机械、化工、建筑、计算机等领域，着重用于解决结构性优化、非线性优化、并行计算等复杂问题。近 10 年来，进化算法在我国也得到重视和推广，特别是遗传算法，已成功用于许多领域中。

通常，进化算法包括遗传算法(Genetic Algorithms)、遗传规划(Genetic Programming)、进化策略(Evolution Strategies)、进化规划(Evolutionary Programming)4 种典型方法。它们在进化原则上是一致的，但是在实施进化的手段上却各有特点，互不相同。尽管国内对遗传算法已有一些论著，但是缺乏对进化算法进行全面、深入阐述的普及性书籍。作者根据自己多年从事进化算法的教学和科研成果，特别是在国家自然科学基金项目 No. 59874019 的资助下，编写本书，力求向我国广大科学技术工作者全面介绍进化算法，为我国科技事业发展添砖加瓦。

本书共分 5 章，即

1. 绪论。在简单介绍生物学中有关进化与遗传概念的基础

上，扼要描述进化算法的特征，并进行历史的回顾与展望。

2. 遗传算法。在详细介绍遗传算法原理的基础上，讨论遗传算法的基本理论　　模式理论，进而全面论述遗传算法的各种实施技术。

3. 遗传规划。在详细介绍遗传规划工作原理及基本技术的基础上，讨论遗传规划常遇见的一些理论问题，进而论述遗传规划的一些最新技术。

4. 进化策略。先介绍进化策略的原理，然后详细论述其基本技术，进而分析有关进化策略的理论问题。

5. 进化规划。先介绍进化规划的原理，然后详细论述其基本技术，最后综合比较 4 种进化算法。

本书由西安建筑科技大学云庆夏编写，陈永锋、卢才武、侯煜参加部分章节编写，卢少华、高文伟校阅全稿。

由于作者学识有限，本书肯定有不足之处，恳请广大专家及读者批评指正。

作　者

2000 年 1 月

目　录

1 绪　论

1.1 生物的进化与遗传

进化算法(Evolutionary Algorithms)通常包括遗传算法(Genetic Algorithms)、遗传规划(Genetic Programming)、进化策略(Evolution Strategies)和进化规划(Evolutionary Programming)。它们都是借鉴生物界中进化与遗传的机理,用于解决复杂的工程技术问题。因此,我们先回顾生物学中进化与遗传的概念,以便以后深入讨论各种进化算法。

1.1.1 生物的进化

地球上的生物,都是经过长期进化而形成的。根据达尔文的自然选择学说,地球上的生物具有很强的繁殖能力。在繁殖过程中,大多数生物通过遗传,使物种保持相似的后代;部分生物由于变异,后代具有明显差别,甚至形成新物种。正是由于生物的不断繁殖后代,生物数目大量增加,而自然界中生物赖以生存的资源却是有限的。因此,为了生存,生物就需要竞争。生物在生存竞争中,根据对环境的适应能力,适者生存,不适者消亡。自然界中的生物,就是根据这种优胜劣汰的原则,不断地进行进化。

进化算法就是借用生物进化的规律,通过繁殖-竞争-再繁殖-再竞争,实现优胜劣汰,一步一步地逼近问题的最优解。进化算法中的"进化"二字,就是由此而冠名。

1.1.2 遗传物质

众所周知,细胞是生物结构和功能的基本单位,细胞通常由细胞膜、细胞质与细胞核三部分组成。细胞核位于细胞的最内层,由核膜、染色质、核液三者组成,是遗传物质贮存和复制的场所。

细胞核中的染色质,在细胞分裂时形成光学显微镜下可以看见的染色体。染色体主要由蛋白质和 DNA 组成。DNA 又称脱氧

核糖核酸，是一种高分子化合物，组成它的基本单位是脱氧核苷酸。DNA 可以传递遗传信息。生物上下代之间传递遗传信息的物质，称作遗传物质。绝大多数生物的遗传物质是 DNA。由于细胞里的 DNA 大部分在染色体上，因此，遗传物质的主要载体是染色体。

控制生物遗传的物质单元称作基因，它是有遗传效应的 DNA 片段。每个基因含有成百上千个脱氧核苷酸。它们在染色体上呈线性排列，这种排列顺序就代表遗传信息。

在进化算法中，为了形成具有遗传物质的染色体，就用不同字符组成的字符串表达所研究的问题。这种字符串相当于染色体，其上的字符就相当于基因。

1.1.3 遗传方式

生物的主要遗传方式是复制。遗传过程中，父代的遗传物质 DNA 分子被复制到子代，以此传递遗传信息。

生物在遗传过程中还会发生变异。变异方式有 3 种：基因重组、基因突变和染色体变异。基因重组是控制物种性状的基因发生重新组合。基因突变是指基因分子结构的改变。染色体变异是指染色体在结构上或数目上的变化。

进化算法中，仿效生物的遗传方式，主要采用复制、交换（重组）、突变这 3 种遗传操作，衍生下一代的个体。

1.2 进化算法简介

1.2.1 遗传算法简介

为了说明遗传算法的实质，本节以函数极值的求解过程为例。

设自变量介于 0～31，求其二次函数的最大值，即

$$\max f(x) = x^2 \qquad x \in [0,31] \tag{1-1}$$

当然，利用简单的代数运算，我们可以求出该问题的解为 $x=31$。现在改用遗传算法求解。遗传算法通常包括下述工作：

(1)编码。遗传算法首先要用字符串表达所研究的问题，这称作编码。表达问题的字符串相当于遗传学中的染色体。每个字符

串称作个体。每一遗传代次中个体的组合称作群体。为了便于计算机操作，通常字符串长度固定，字符用二进制码或为 0，或为 1。

本例中，用二进制数表示 x 值。由于 x 的最大值(31)只需 5 位二进制数，所以利用 5 位二进制数组成个体。

(2)形成初始群体。遗传算法中，常用随机的方法产生初始群体，即随机生成一组任意排列的字符串。群体中个体的数目通常也是固定的。

本例中，采用随机产生的方法，假设得出拥有 4 个个体的初始群体，即：01101、11000、01000、10011。它们的 x 值相应为：13、24、8、19(见表 1-1)。

表 1-1　遗传算法的第 0 代

个体编号	初始群体	x_i	适应度 $f(x_i)$	$f(x_i)/\Sigma f(x_i)$	$f(x_i)/\bar{f}$	下代个体数目
1	2	3	4	5	6	7
1	01101	13	169	0.14	0.58	1
2	11000	24	576	0.49	1.97	2
3	01000	8	64	0.06	0.22	0
4	10011	19	361	0.31	1.23	1
总计 $\Sigma f(x_i)$			1170			
平均值 $\bar{f}$			293			
最大值 f_{max}			576			
最小值 f_{min}			64			

(3)计算适应度。衡量字符串(染色体)好坏的指标是适应度(Fitness)，它通常也就是遗传算法的目标函数。适应度是今后优胜劣汰的主要判据。

在本例中，适应度比较简单，用 x^2 计算。当 x 值为 13、24、8、19 时的适应度分别为：169、576、64、361(表 1-1 第 4 列)。

表 1-1 中还列举当前适应度的总和 $\Sigma f(x_i)$ 及平均值 $\bar{f}$，即

$$\Sigma f(x_i) = f(x_1) + f(x_2) + f(x_3) + f(x_4) = 1170$$

$$\bar{f} = \Sigma f(x_i)/4 = 293$$

表 1-1 中第 6 列的 $f(x_i)/\bar{f}$ 表示每个个体的相对适应度，它反映个体之间的相对优劣性。例如，2 号个体的 $f(x_i)/\bar{f}$ 值最高(1.97)，为优良个体；而 3 号个体最低(0.22)，为不良个体。

(4)复制(Reproduction)。为了将已有的群体变为下一代群体，遗传算法仿效进化论中“自然选择，适者生存”的原则，从旧群体中选择优良个体予以复制，直接进入下一代群体。选择的依据是个体适应度的大小，适应度大的个体接受复制，使之繁殖；适应度小的个体则予删除，使之死亡。

本例中，根据相对适应度 $f(x_i)/\bar{f}$ 的大小对个体进行取舍。2 号个体性能最优($f(x_i)/\bar{f}=1.97$)，予以复制繁殖。3 号个性能最差($f(x_i)/\bar{f}=0.22$)，将它删除，使之消亡。表 1-1 第 7 列表示传递给下一代的个体数目，其中 2 号个体占 2 个，3 号个体为 0 个，1 号及 4 号个体仍保持为 1 个。

这样，就产生下一代新群体，如表 1-2 所示。新群体的 4 个个体分别是 01101、11000、11000、10011。从表 1-2 第 4 列可以看出，复制后产生的新一代群体，其平均适应度 $\bar{f}$ 明显增加，由原来的 293(表 1-1) 增至 421(表 1-2)。造成平均适应度增加的原因有二：一是淘汰原来最差的个体，使最小适应度由原来的 64(表 1-1) 增至 169(表 1-2)；另一个原因是增加优良个体(2 号)的个数，使适应度累计值增加。因此，复制体现优胜劣汰原则，使群体的素质不断得到改善。

表 1-2 遗传算法的复制与交换(第 1 代)

个体编号	复制初始群体	x_i	复制后适应度 $f(x_i)$	交换对象	交换位置	交换后群体	交换后适应度 $f(x_i)$
1	2	3	4	5	6	7	8
1	01101	13	169	2 号	3	01100	144
2	11000	24	576	1 号	3	11001	625

续表 1-2

个体编号	复制初始群体	x_i	复制后适应度 $f(x_i)$	交换对象	交换位置	交换后群体	交换后适应度 $f(x_i)$
3	11000	24	576	4 号	2	11011	729
4	10011	19	361	3 号	2	10000	256
总计 $\Sigma f(x_i)$			1682				1754
平均值 $\bar{f}$			421				439
最大值 f_{max}			576				729
最小值 f_{min}			169				256

(5)交换(Crossover)。通过复制产生的新群体,其总体性能得到改善,然而却不能产生新的个体。为了产生新个体,遗传算法仿照生物学中杂交的办法,对染色体(字符串)的某些部分进行交叉换位。被交换的母体都选自经过复制产生的新一代个体(优胜者)。

本例中,利用随机配对的方法,决定 1 号和 2 号个体、3 号和 4 号个体分别交换,如表 1-2 第 5 列所示。再利用随机定位的方法,确定这两对母体交叉换位的位置分别从字符串左数第三位字符及第二位字符之后。例如,3 号及 4 号个体如下式左侧所示,交换始于字符串左数第 2 位之后,交换开始的位置称交换点,用"┆"标记,所得的新个体如右侧所示:

$$\begin{Bmatrix} 父代\ 1:1\ 1 \mid 0\ 0\ 0 \\ 父代\ 2:1\ 0 \mid 0\ 1\ 1 \end{Bmatrix} \rightarrow \begin{Bmatrix} 子代\ 1:1\ 1 \mid 0\ 1\ 1 \\ 子代\ 2:1\ 0 \mid 0\ 0\ 0 \end{Bmatrix}$$

至于 1 号、2 号个体交换结果如表 1-2 第 7 列所示。

表 1-2 中最后一列表示交换后群体的适应度。从表中可以看出,交换后出现优异个体 3 号,其适应度高达 729,大大高于交换前的最大值(576)。与此同时,平均适应度也从原来的 421 提高到 439,说明交换后的群体正朝优良方向发展。

(6)突变(Mutation)。遗传算法模仿生物学中基因突变的方法,将个体字符串某位符号进行逆变,即由 1 变为 0 或由 0 变为 1。例如,下式左侧的个体于第 3 位突变,得到新个体如右侧所示:

{ 1 0 0 0 0 } → { 1 0 1 0 0 }

遗传算法中，个体是否进行突变以及在哪个字符突变，都由事先给定的概率决定。通常，突变概率很小，约为0.01，本例的第一代中就没有发生突变。

(7)终止。反复执行上述(3)～(6)项工作，直至得出满意的最优解。

通过上述函数极值求解的例子可以看出，遗传算法仿效生物进化和遗传的过程，从随机生成的初始可行解出发，利用复制、交换、突变等操作，遵循优胜劣汰的原则，不断循环执行，逐渐逼近全局最优解。

1.2.2 遗传规划简介

遗传算法是用字符串作为染色体去表达所研究的问题，而且字符串的长度常常是固定的。然而，现实中的问题往往很复杂，有时不能用简单的字符串表达问题的所有性质，于是就产生了遗传规划。遗传规划用广义的计算机程序形式表达问题，它的结构和大小都是可以变化的，从而可以更灵活地表达复杂的事物性质。

现以曲线拟合为例，说明遗传规划的基本原理。设图1-1的曲线表示某种实验的观测结果，现在要用遗传规划确定该实验结果的函数关系 $y=f(x)$。传统的曲线拟合，先要确定方程的结构形

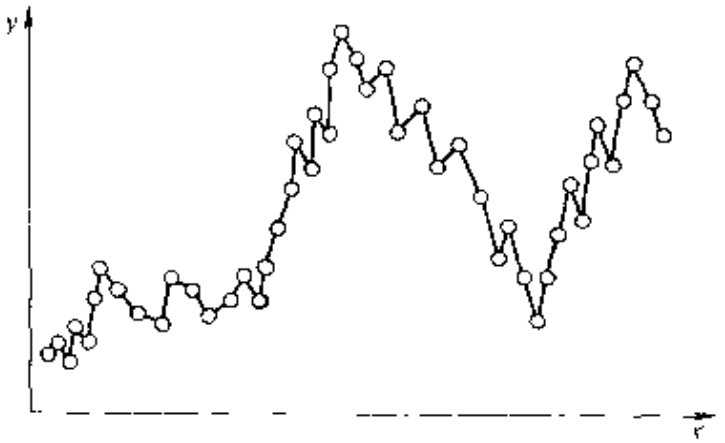

图1-1 曲线拟合

式：或线性形式、对数形式，或多项式等，然后再进行参数估计。采用遗传规划时，可以将结构估计和参数估计融合在一起。遗传规划的工作过程大致如下：

(1)确定表达结构。遗传规划用可变的层状计算机程序结构表达问题。这种广义的计算机程序结构通常由两部分组成：一为函数集 F，F 包括+、-、*、/、sin、cos、tan、log、exp、↑……等运算符号；另一为终止符集 T，T 包括变量 x 和随机数 A、B、C…… 等。遗传规划中的个体(染色体)将随机地由上述函数及终止符组成，例如：

$$A + B * x \quad 或 \quad B * \exp(A/\sin x)\cdots\cdots$$

(2) 形成初始群体。采用随机选取的方法，从函数集 F 及终止符集 T 中随机选择函数及其相应的终止符，组成下述 4 个个体：

个体 1： $y = A + B * x$

个体 2： $y = A + B * x + C * x^2$

个体 3： $y = x * \sin x$

个体 4： $y = C * x * \sin x$

尽管初始形成的个体不符合实际，但在以后的进化计算中会不断改进，逼近实际所需要的形式。

(3) 计算适应度。将不同的实验数据 x_i 代入上述 4 种初始表达式中，得出一组计算值 $\hat{y}_i$。将计算所得的 $\hat{y}_i$ 与实验数据 y_i 相比较，计算两者之差，然后用误差衡量 4 种初始表达式的优劣。假设第 3 个表达式最佳，第 1 个表达式最差。

(4) 复制。根据优胜劣汰的原则，复制效果最佳的第 3 个表达式，淘汰效果最差的第 1 个表达式。于是，新一代的群体由下述表达式组成：

个体 1： $y = x * \sin x$

个体 2： $y = A + B * x + C * x^2$

个体 3： $y = x * \sin x$

个体 4： $y = C * x * \sin x$

(5) 交换。为了产生新的表达式，需要使用交换。采用随机选

择的方法，假设第 2 和第 3 个表达式进行交换，交换位置在第 一项，则新的表达式为：

个体 1： $y = x * \sin x$

个体 2： $y = x + B * x + C * x^2$

个体 3： $y = A * \sin x$

个体 4： $y = C * x * \sin x$

(6) 突变。遗传规划也可采用突变产生新个体。例如，将表达式中的 $\sin x$ 变为 $\cos x$。不过，遗传规划中的突变远不及在遗传算法中那样重要。

(7) 终止。反复执行上述(3)～(6)项工作，使函数表达式 $y = f(x)$ 不断变化，逐渐逼近所要求的表达式。

从上述曲线拟合的简单例子可以看出，遗传规划和遗传算法相类似，都是从随机产生的初始群体出发，同样用适应度衡量个体优劣，都采用复制、交换、突变等操作，经过一代一代的优胜劣汰，逐步得出最优的数学表达式

遗传规划和遗传算法的差别，主要在问题的表达方式上。后者用定长的字符串，前者则是形式可变的计算机程序结构。很明显，遗传规划的表达式更加灵活多变，适用于复杂的课题。应该指出，遗传规划(Genetic Programming)可直译为遗传程序设计，它以计算机程序的层次结构形式表达问题，而绝不是执行遗传算法的计算机程序，为了避免误会，本书称作遗传规划。

1.2.3 进化策略简介

进化策略是最早出现的一种进化算法。它用传统的实型数表达问题，其表达形式如下：

$$X^{t+1} = X^t + N(0,\sigma) \tag{1-2}$$

式中 X^t —— 用实数表示的第 t 代个体；

X^{t+1} —— 用实数表示的第 $t+1$ 代个体；

$N(0,\sigma)$ —— 独立的随机数，服从正态分布，后者的数学期望为 0，标准差为 σ。

请注意，习惯上常用 $\sim N(0,\sigma^2)$ 表示正态分布。本书采用

$N(0,\sigma)$ 表示一个服从正态分布的随机数。

式(1-2)表明，新一代的个体 X^{t+1} 是在父代个体 X^t 的基础上添加一个随机量 $N(0,\sigma)$。因此，每个个体由 X 及 σ 两个变量决定，是一个二元组 (X,σ)。

进化策略中个体的进化主要采用突变，即对随机量中的标准差进行修正：

$$\begin{cases}\sigma' = \sigma \cdot e^{(N(0,1))} \\ X' = X + N(0,\sigma')\end{cases} \tag{1-3}$$

式中 (X,σ) —— 父代个体；

(X',σ') —— 子代个体。

这就是说，新一代的 X' 是在上一代的 X 基础上添加一个微小的随机量 $N(0,\sigma')$，后者服从数学期望为 0、标准差为 σ' 的正态分布。新一代的标准差 σ' 又是在上一代标准差 σ 的基础上乘以一个微小的随机量 $\exp(N(0,1))$。

在进化策略中，产生新个体的另一种方法是重组(Recombination)，它相当于遗传算法的交换。最简单的重组是随机交换两个个体的 x_i 及 σ_i。例如，设有两个体如下：

个体 1： $(X^1,\sigma^1) = ((x_1^1, x_2^1, \cdots, x_n^1),(\sigma_1^1, \sigma_2^1, \cdots, \sigma_n^1))$

个体 2： $(X^2,\sigma^2) = ((x_1^2, x_2^2, \cdots, x_n^2),(\sigma_1^2, \sigma_2^2, \cdots, \sigma_n^2))$

重组后的子代个体为：

新个体： $(X,\sigma) = ((x_1^{q_1}, x_2^{q_2}, \cdots, x_n^{q_n}),(\sigma_1^{q_1}, \sigma_2^{q_2}, \cdots, \sigma_n^{q_n}))$ (1-4)

其中 q_i 随机地或为 1，或为 2。

在进化策略中，复制隐含在选择(Selection)中。父代群体所有的 μ 个个体，经过突变、重组后生成 λ 个新个体，然后再从这些群体中按适应度选择 μ 个优良个体组成下一代群体，从而体现个体在竞争中的优胜劣汰原则。

同样，进化策略也是一个反复迭代的过程，它从随机产生的初始群体出发，经过突变、重组(交换)、选择等进化操作，改进群体的质量，逐渐得出最优解。

1.2.4 进化规划简介

进化规划与进化策略几乎同时出现，并平行发展。最早的进化策略只采用单个个体，而最早的进化规划则是采用多个个体组成群体共同进化。

进化规划也是用实型数表达问题，其表达形式为：

$$X^{t+1} = X^t + \sqrt{f(X^t)} \cdot N(0,1) \tag{1-5}$$

式中 X^t 用实数表示的第 t 代旧个体；

X^{t+1} 用实数表示的第 $t+1$ 代新个体；

$N(0,1)$ 独立的随机数，服从(0,1)标准正态分布；

$f(X^t)$ X^t 的适应度

早期的进化规划，只用上式实现个体的不断进化，它相当于突变。后期的进化规划，也仿效进化策略引入方差的调整作用，即

$$\begin{cases} X = X + \sqrt{\sigma} \cdot N(0,1) \\ \sigma = \sigma + \sqrt{\sigma} \cdot N(0,1) \end{cases} \tag{1-6}$$

在进化规划中，没有重组或交换，但有选择，它从 μ 个父代个体及 λ 个子代个体中择优选取 μ 个优良个体组成下一代群体。

近年来，由于进化策略在进化规划中的交叉渗透，两者的差别减少。

1.3 进化算法的特征

1.3.1 进化算法的实质

进化算法中，无论是遗传算法、遗传规划、进化策略或进化规划，都是从一组随机生成的初始个体出发，经过复制(选择)、交换(重组)、突变等操作，并根据适应度大小进行个体的优胜劣汰，提高新一代群体的质量，再经过多次反复迭代，逐步逼近最优解。从数学角度讲，进化算法实质上是一种搜索寻优的方法。

以图 1-2 为例。假设图中曲面表示某种复杂的数学关系 $f(x, y)$，现在要求它的最大值 $\max f(x,y)$。传统的寻优方法有三种：

(1)基于导数的方法，最常用的是利用数学分析中求函数极值的方法。令函数的一阶导数为零：

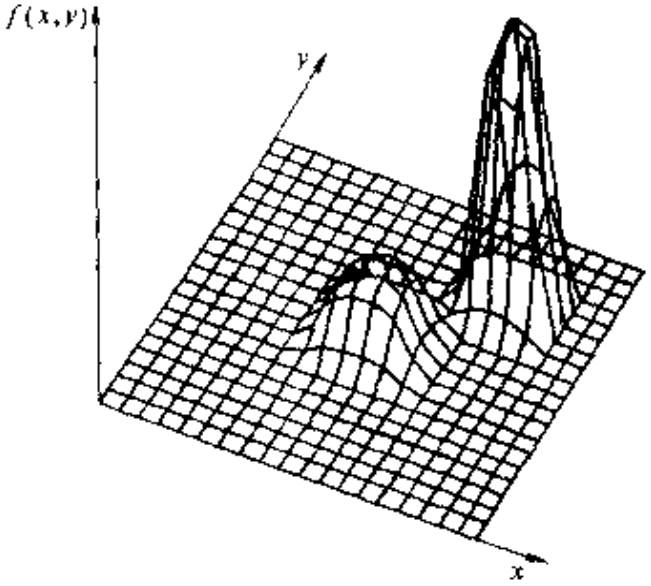

图 1-2　求函数极大值

$$\begin{bmatrix} \partial f/\partial x \\ \partial f/\partial y \end{bmatrix} = 0 \tag{1-7}$$

解上述方程得出驻点，若该点的海色矩阵之值

$$\begin{vmatrix} \partial^2 f/\partial x^2 & \partial^2 f/\partial x \partial y \\ \partial^2 f/\partial y \partial x & \partial^2 f/\partial y^2 \end{vmatrix} > 0$$

则该点为所求的极值点。

另一种方法就是爬山法。它根据函数的一阶导数确定梯度最大的方向，然后沿此方向逐步向上迁移，最终到达峰顶。每次爬坡，按下式计算：

$$f\begin{pmatrix} x^{k+1} \\ y^{k+1} \end{pmatrix} = \max_d f\left\{ \begin{pmatrix} x^k \\ y^k \end{pmatrix} + d \cdot \begin{pmatrix} \partial f/\partial x \\ \partial f/\partial y \end{pmatrix}_{(x^k, y^k)} \right\} \tag{1-8}$$

式中　x^k、y^k ---- 第 k 次迭代的起点坐标；

d ---- 每次迁移步长。

这些方法，不仅要求函数 $f(x,y)$ 连续、可导，而且对其一阶导数及二阶导数都有要求。对于复杂的问题，往往不能满足这些要求。

(2) 穷举法。在 x、y 平面内依次搜索所有 x_i、y_i 时的 $f(x_i, y_i)$ 值，以便求出函数的最大值位置。显然，这是一种费时费力的方法。

(3) 随机搜索法。在 x、y 平面内随机确定搜索点 x_i、y_i，再计算其函数值 $f(x_i, y_i)$，争取经过少量搜索找出函数的最大值点。显然，这种方法具有很大的偶然性。

进化算法尽管也是一种搜索寻优的方法，但是它和传统的方法有很大的不同，它不要求所研究的问题是连续、可导的，但是却可以很快地得出所要求的最优解。图 1-3 表示进化算法的基本流程。

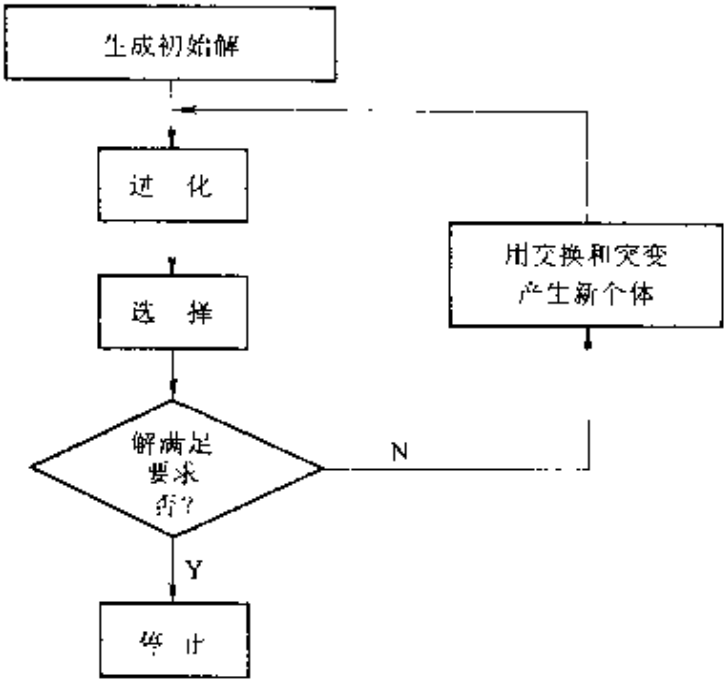

图 1-3 进化算法工作流程

1.3.2 进化算法的主要特征

进化算法的搜索方式，具有如下特征：

(1) 有指导搜索。进化算法的搜索策略，既不是盲目式的乱搜索，也不是穷举式的全面搜索，它是有指导的搜索。指导进化算法执行搜索的依据是适应度，也就是它的目标函数。在适应度的驱动下，使进化算法逐步逼近目标值。

(2)自适应搜索。进化算法在搜索过程中，借助复制(选择)、交换(重组)、突变等操作，体现"适者生存，劣者消亡"的自然选择规律，无需添加任何额外的作用，就能使群体的品质不断得到改进，具有自动适应环境的能力。

(3)渐进式寻优。进化算法从随机产生的初始可行解出发，一代一代地反复迭代，使新一代的结果优越于下一代，逐渐得出最优的结果，这是一个逐渐寻优的过程。

(4)并行式搜索。进化算法每一代运算都是针对一组个体同时进行，而不是只对单个个体。因此，进化算法是一种多点齐头并进的并行算法，大大提高了进化算法的搜索速度。并行式计算是进化算法的一个重要特征。

(5)黑箱式结构。进化算法根据所解决问题的特性，用字符串表达问题及选择适应度，一旦完成这两项工作，其余的复制(选择)、交换(重组)、突变等操作都可按固定方式进行。个体的字符串表达如同输入，适应度计算如同输出。因此，从某种意义讲，进化算法是一种只考虑输入与输出关系的黑箱问题。图 1-4 就是黑箱表达的形象解释，图中 5 个开关代表 5 位 0—1 变量，表示输入的字符串。图中的输出 $f(s)$ 是适应度计算结果，代表输出。进化算法只研究输入与输出的关系，并不深究造成这种关系的原因，因此便于处理因果关系不明确的问题。

图 1-4　黑箱结构

(6)全局最优解。进化算法由于采用多点并行搜索，而且每次迭代借助交换和突变产生新个体，不断扩大搜索范围，因此进化算法很容易搜索出全局最优解而不是局部最优解。以图 1-2 为例，假若搜索范围过于狭窄，有可能误以为低峰便是最优解，但是进化算法能达到最高峰。

(7)通用性强。传统的优化算法，需要将所解决的问题用数学

式子表达，而且要求该数学函数的一阶导数或二阶导数存在。采用进化算法，只用某种字符表达问题，然后根据适应度区分个体优劣，其余的复制（选择）、交换（重组）、突变等操作都是统一的，由计算机自动执行。因此，有人称进化算法是一种框架型算法，它只有一些简单的原则要求，在实施过程中无需额外的干预。

1.3.3 进化算法的生物学含义

进化算法仿效生物的进化和遗传，将生物学的基本原则用于优化计算中。因此，不仅进化算法的指导性原则与生物学吻合，而且基本术语也相似。

进化算法的运算基础是字符串或字符段，它就相当于生物学中的染色体。字符串（段）由一系列字符组成，每个字符都有特定的含义，它就相当于基因，即染色体 DNA 片断。

进化算法的迭代过程，相当于生物学的逐代进化。进化算法中的复制（选择），体现生物界中"自然竞争、优胜劣汰"。进化算法的交换（重组），相当于生物界的交配。进化算法的突变，类似于生物界的变异。

此外，生物学中的等位基因、显性性状、隐性性状、表现型、基因型等术语，在进化算法中都有相应的体现。

表 1-3 列举生物学术语在进化算法中的对应意义，从中可以进一步了解进化算法的生物学含义。

表 1-3 进化算法的术语在生物学的含义

序　号	进化算法	生物学
1	字符串、字符段	染色体
2	字符	基因
3	迭代	进化
4	复制（选择）	优胜劣汰
5	交换（重组）	交配
6	突变	变异
7	字符位置	基因位置
8	字符串结构	基因型
9	字符串含义	表现型

1.4 进化算法的发展与应用简介

1.4.1 进化算法的发展简介

进化算法与其他科学技术一样，都经历一段成长过程，逐渐发展壮大。此过程可大致分为三个时期：萌芽期、成长期和发展期。

1.4.1.1 萌芽期(50年代后期至70年代初期)

早在50年代后期，一些生物学家就着手采用电子计算机模拟生物的遗传系统，尽管这些工作纯粹是研究生物现象，但其中已使用现代遗传算法的一些标识方式。例如，美国A.S.Fraser于1960年为了建立生物的表现型方程，用3组5位(共15位)的0-1字符串表示方程的三个参数。

1965年，德国的I.Rechenberg等人正式提出进化策略的方法，当时的进化策略只有一个个体，而且进化操作也只有突变一种。1965年，美国的L.J.Fogel正式提出进化规划，在计算中采用多个个体组成的群体，而且只运用突变操作。60年代期间，美国芝加哥大学教授J.H.Holland在研究自适应系统时，提出系统本身与外部环境相互协调的遗传算法。1968年，J.H.Holland教授又提出模式理论，它成为遗传算法的主要理论基础。

第一次正式使用遗传算法(Genetic Algorithm)这个术语，是在1967年美国J.D.Bagay关于博奕论的论文中，他采用复制、交换、突变等手段研究国际象棋的对奕策略。

1.4.1.2 成长期(70年代中期至80年代末期)

1975年，J.H.Holland教授的专著《自然界和人工系统的适应性(Adaptation in Natural and Artificial Systems)》正式出版，全面地介绍了遗传算法，人们常常把这一事件视作遗传算法问世的标志，Holland也被视作遗传算法的创始人。

1975年，德国H.P.Schwefel在他的博士论文中发展了进化策略，采用多个个体组成的群体而不是单个个体参与进化，而且进化的手段包括突变及重组，使进化策略更加完善。

80年代期间，许多研究工作者积极从事遗传算法的研究，使

遗传算法成为美国人工智能领域的又一个研究热点。1987年，美国D. Lawrence总结人们长期从事遗传算法的经验，公开出版《遗传算法和模拟退火(Genetic Algorithm and Simulated Annealing)》一书，以论文集形式用大量实例介绍遗传算法。1989年，作为J. H. Holland的学生，D. E. Goldberg博士出版专著《遗传算法搜索、优化及机器学习(Genetic Algorithms in Search, Optimization and Machine Learning)》，全面、系统地介绍遗传算法，使这一技术得到普及与推广，该书被人们视为遗传算法的教科书。

1985年，在美国举行第一届遗传算法国际学术会议(International Conference on Genetic Algorithms)，与会者交流运用遗传算法的经验。随后，1987，1989，1991，1993，1995及1997年，每2年左右都举行一次这种会议。

80年代后期，D. B. Fogel改进父亲L. J. Fogel的进化规划，在突变中添加方差的变化，使进化规划与进化策略交叉渗透。

1.4.1.3 发展期(90年代以后)

这时期在进化算法方面最突出的成就，是遗传规划的出现。早在80年代后期，人们已开始努力改进遗传算法的表达形式，例如有人采用不定长的字符串，也有人使用一系列If-then语句表达问题。美国斯坦福大学的J. R. Koza于1989年提出遗传规划新概念，用层次化的计算机程序代替字符串表达问题。1992年，他出版专著《遗传规划——应用自然选择法则的计算机程序设计(Genetic Programming: on the Programming of Computer by Means of Natural Selection)》，该书全面介绍了遗传规划的原理及应用实例，标明遗传规划已成为进化算法的一个重要分支，Koza本人也被视作遗传规划的奠基人。

1994年，J. R. Koza又出版第二部专著《遗传规划Ⅱ：可再用程序的自动发现(Genetic Pragramming Ⅱ: Automatic Discovery of Reusable Programs)》，提出自动定义函数的新概念，在遗传规划中引入子程序的新技术。同年，K. E. Kinnear Jr主编《遗传规划

进展(Advances in Genetic Programming)》,汇集许多研究工作者有关应用遗传规划的经验和技术。1998 年,W. Banzhaf 等人出版专著《遗传规划:计算机程序的自动进化及其应用综述(Genetic Programming: An Introduction On the Automatic Evolution of Computer Programs and its Applications)》,除了全面总结遗传规划的方法外,还对各种改进技术进行评述。与此同时,有关遗传规划的定期国际学术会议也多次召开,标志着遗传规划已得到广泛的应用。

90 年代,遗传算法不断地向广度和深度发展。1991 年,D. Lawrence 出版《遗传算法手册(Handbook of Genetic Algorithms)》一书,详尽地介绍遗传算法的工作细节。1996 年 Z. Michalewicz 的专著《遗传算法+数据结构=进化程序(Genetic Algorithms + Data Structure = Evolution Programs)》深入讨论了遗传算法的各种专门问题。同年,T. Back 的专著《进化算法的理论与实践:进化策略、进化规划、遗传算法(Evolutionary Algorithms in Theory and Practice: Evolutionary Strategies, Evolutionary Programming, Genetic Algorithms)》深入阐明进化算法的许多理论问题。这一时期,国外有关进化算法的专著有十几本,说明进化算法已得到广泛的关注。

90 年代期间,有关进化算法的国际会议也比较活跃,除了前述的遗传算法会议外,还有(表 1-4):

(1) 进化规划年会(Annual Conference on Evolutionary Programming)。第一次会议 1992 年于美国圣地亚哥举行,随后于 1993,1994,1995,1996,1997,1998 年分别举行。

(2) 仿效自然界的并行算法国际会议 (International Conference on Parallel Problems Solving from Nature)。第一次会议于 1990 年召开,随后于 1992,1994,1996 年分别举行。

(3) 遗传算法基础会议(Workshop on the Foundation of Genetic Algorithms)。第一次会议于 1990 年召开,随后于 1992,1994 及 1997 年举行。

（4）遗传规划年会（Annual Genetic Programming Conference）。第一次会议1996年召开，随后于1997，1998年分别举行会议。

（5）关于进化计算的电学学会的国际会议（IEEE International Conference of Evolutionary Computation）。第一次会议1994年于美国举行，1995及1996年也举行过同样会议。

（6）欧洲人工生命会议（European Conference on Artificial Life）第一次会议于1992年召开，随后1995年也举行过。

国外有关进化算法的学术期刊有：进化计算（Evolutionary Computation，MIT Press），关于进化计算电学学会学报（IEEE Transaction on Evolutionary Computation，IEEE），关于系统、人和控制论的电学学会学报（IEEE Transaction of Systems，Man and Cybernetics，IEEE），自适应行为（Adaption Behavior，MIT Press）。

我国开展进化算法研究，主要在90年代。目前，已成为继专家系统、人工神经网络之后有关人工智能方面的第三个热点课题。

表1-4 进化算法的学术会议

时 间	地 点	论文集主编
1. 遗传算法国际会议（International Conference on Genetic Algorithms）		
1985	美国 Pittsbergh	Grefenstette
1987	美国 Cambridge	Grefenstette
1989	美国 George Mason 大学	Schaffer
1991	美国 San Diego	Belew and booker
1993	美国 Urbana Champaign	Forrest
1995	美国 Pittsbergh	Fishelman
1997	美国 Ann Arbor	

2. 仿效自然界的并行算法国际会议(International Conference on Parallel Problems Solving from Nature)

1990	德国 Dortmund	Schwelfel and Manner
1992	比利时 Brussels	Manner and Manderick
1994	以色列 Jerusalem	Davidor、Schwefel and Manner
1996	德国 Dortmund	Ebeling and Voigt

3. 遗传算法基础会议(Workshop on Foundation of Genetic Algorithms)

1990	美国 San Mateo	Rawlines
1992	美国 Vail	Whitley
1994	美国 San Mateo	Whitley and Vose
1997	美国 San Mateo	Belew and Vose

4. 进化规划年会(Annual Conference on Evolutionary Programming)

1992	美国 San Diego	Fogel and Atmar
1993	美国 La Jolla	Fogel and Atmar
1994	美国 San Diego	Sebald and Fogel
1995	美国 San Diego	McDonnell,Reynolds and Fogel
1996	美国 San Diego	Angeline and Back
1997	美国 San Diego	
1998	美国 San Diego	Porto,Saravanan,Waagen and Eiben

5. 关于进化计算的电学学会的国际会议(IEEE International Conference on Evolutionary Computation)

1994	美国 Orlando	Fogel
1995	澳大利亚 Perth	de Silva
1996	日本 Nagoya	Fogel

6. 遗传规划年会(Annual Genetic Programming Conference)

1996	美国 Stanford	Koza
1997	美国 Stanford	Koza
1998	美国 Stanford	Koza

续表 1-1

7. 欧洲遗传规划年会(Genetic Programming:European Workshop)		
1998	法国 Paris	Banzaf
1999	瑞典 Goteborg	Poli

1.4.2 进化算法的应用简介

进化算法是一种新型的优化技术，它仿效生物界的进化与遗传，弥补传统优化技术的不足。然而进化算法并不是万能的，它并不能代替已有的优化技术，各自都有自己的适用范围。进化算法在下述领域特别奏效。

(1) 结构性优化。通常，工程技术的优化包括结构优化和参数优化。对于后者，人们已成功地使用了许多方法，如运筹学、数理统计、有限元等数值计算。然而对于结构优化，还缺乏成熟、有效的方法。近年来，人们运用进化算法，成功地解决了建筑桁架结构、飞机结构设计、电网及管网等网络结构等结构性问题，充分显示进化算法在这一领域的广阔应用前景。

(2) 人工智能。进化算法继模糊数学、专家系统、人工神经网络之后，成为处理人工智能的又一个有力工具。许多研究工作者利用这种新技术，从事机器学习、自动程序设计、聚类分析、博奕对策等工作。在知识工程方面，进化算法发挥越来越重要的作用。

(3) 复杂问题的优化。当所要解决的问题具有非线性、多峰值、不确定性时，使用传统的优化方法常常不能奏效。进化算法由于是一种黑箱式的框架型技术，不要求有明确的因果关系数学表达式，因此它是解决这类问题的有力工具，可以解决诸如液体流动、气候变化以及军事战略等非线性动态系统问题。

(4) 复杂系统分析。多年来，人们应用进化算法从事聚类分析、模式识别、图像处理、调度组织等工作，将表面上杂乱无章的复杂事物条理化。对于这类复杂系统的分析和归纳，进化算法具有很大的应用价值。

(5) 综合应用。随着科学技术的发展，各种学科不断交叉渗

透，相互促进。同样，进化算法也要和其他技术手段相结合，各自发挥特长，综合解决问题。例如，人们已将遗传算法和人工神经网络相结合，成功地解决了机器学习等问题。

由于进化算法具有广阔的应用范围，我们无法一一列举具体的领用领域，只能概括地指明应用方向。随着时间的推移，它的应用范围还会不断扩大。

2 遗传算法

2.1 遗传算法的基本原理

遗传算法(Genetic Algorithms，简称 GA)是 60 年代后期由美国 J. H. Holland 教授首先提出的一种进化算法，目前使用最为广泛。本节结合一个复杂函数求极值的示例，介绍遗传算法的基本原理。

设目标函数是：

$$\max f(x_1,x_2) = 21.5 + x_1 \cdot \sin(4\pi x_1) + x_2 \cdot \sin(20\pi x_2) \tag{2-1}$$

约束条件为：

$$-3.0 \leqslant x_1 \leqslant 12.1 \tag{2-2}$$

$$4.1 \leqslant x_2 \leqslant 5.8 \tag{2-3}$$

此优化问题的几何图形如图 2-1 所示。从图中可以看出，该图形很复杂，用传统的方法很难求解，需要采用遗传算法。

2.1.1 编码

遗传算法的工作对象是字符串，因此编码是一项基础性工作。从生物学角度看，编码相当于选择遗传物质，每个字符串对应一个染色体。

遗传算法大多采用二进制的 0/1 字符编码。当问题比较简单，例如只描述贵/贱、大/小、好/坏等布尔型性质时，每一位 0/1 变量就代表一个性质。当问题的性质要用数值描述时，则涉及二进制数与十进制数的转换。对于长度(位数)为 L 的 0/1 字符串，按数学的排列组合计算，它可以表达 $2 \times 2 \times 2 \cdots 2 = 2^L$ 个数，扣除零点，为 $2^L - 1$，根据内插计算，十进制数与二进制数有如下关系：

$$x = x_{min} + \frac{x_{max} - x_{min}}{2^L - 1} \mathrm{Dec}(y) \tag{2-4}$$

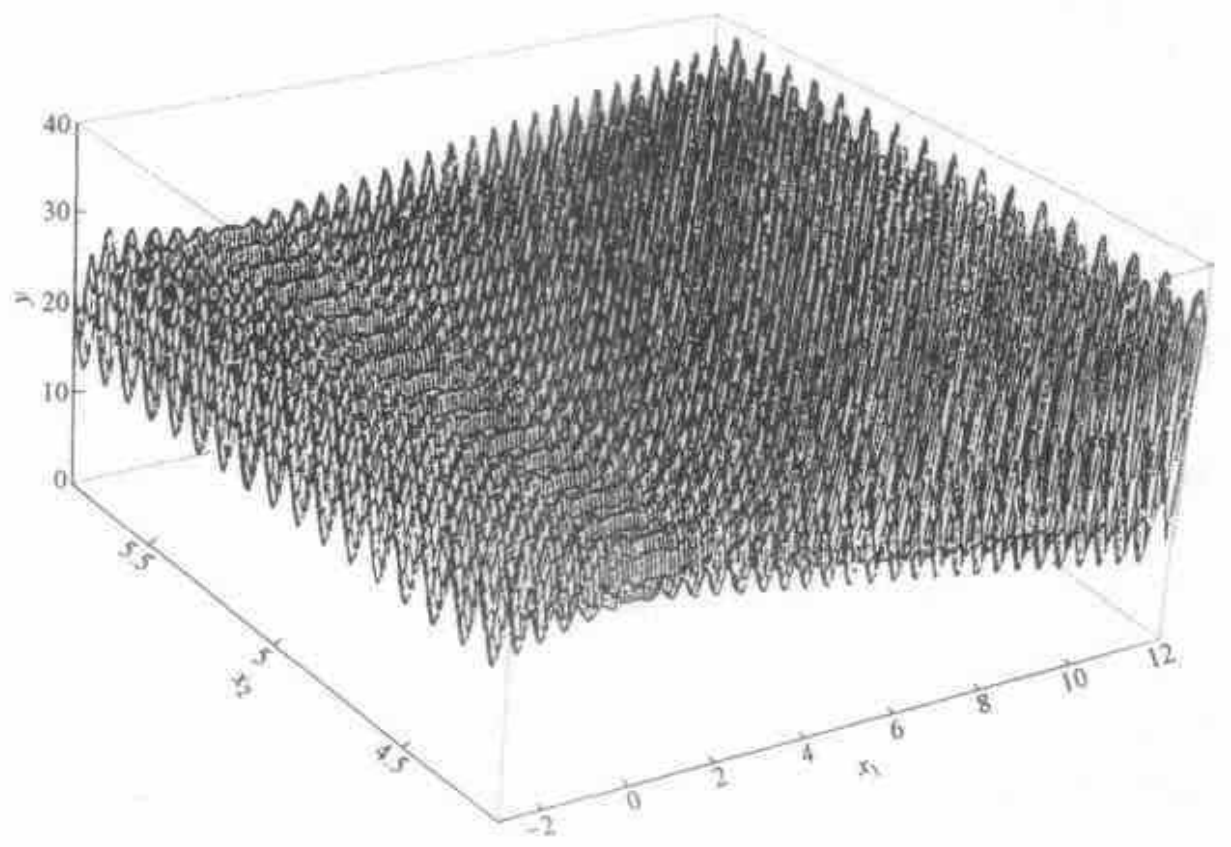

图 2-1　$f(x_1, x_2)$ 几何图形

式中　x_{min}、x_{max}——最小及最大的十进制数值；

y　—相应于 x 的二进制数；

Dec ——将二进制数转换为十进制数。

在这种换算关系下，二进制表示法的精度 δ 为：

$$\delta = \frac{x_{max} - x_{min}}{2^L - 1} \tag{2-5}$$

对于兼有多种性质的问题，可以将描述各种性质的字符串组合在一起，用一长字符串表达。例如，可选 25 位 0/1 字串表示物体的体积、重量及材质，其中前 10 位数表示体积量，中间 10 位表示重量，后 5 位表示材质。下式表示用 10 组 4 位字符串描述 10 个参数：

0001	0101	…	1100	1111
x_1	x_2		x_9	x_{10}

[例] 针对优化问题(2-1)～(2-3)，设精度要求 $\delta = 1/10000$，对于 x_1，由式(2-5)有：

$$2^l \geq \frac{x_{1\max} - x_{1\min}}{\delta} + 1 = \frac{12.1 - (-3.0)}{1/10000} + 1 = 151001$$

即

$$2^{17} < 151001 < 2^{18}$$

也就是说，x_1 需要 18 位 0/1 字符表示。同理，对于 x_2，有：

$$2^{11} < 17000 < 2^{15}$$

即 x_2 需要 15 位 0/1 字符。于是，本优化问题有：

$$L = 18 + 15 = 33$$

即字符串的总长度为 33，其中前 18 位代表 x_1，后 15 位代表 x_2，假设有一字符串：

(0 1 0 0 0 1 0 0 1 0 1 1 0 1 0 0 0 0 1 1 1 1 1 0 0 1 0 1 0 0 0 1 0)

根据式(2-4)，有

$$x_1 = -3.0 + \frac{12.1 - (-3.0)}{2^{18} - 1}\text{Dec}(0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0)$$

$$= -3.0 + \frac{12.1 - (-3.0)}{2^{18} - 1} \times 70352$$

$$= 1.052426$$

同理

$$x_2 = 4.1 + \frac{5.8 - 4.1}{2^{15} - 1}\text{Dec}(1\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 0)$$

$$= 5.755330$$

2.1.2 产生初始群体

初始群体是遗传算法搜索寻优的出发点。群体规模 M 越大，搜索的范围越广，但是每代的遗传操作时间越长。反之，M 越小，每代的运算时间越短，然而搜索空间也越小。通常，M 取 50～100。

初始群体中的每个个体是按随机方法产生的。根据字符串的长度 L，随机产生 L 个 0/1 字符组成初始个体。初始个体的数目为 M。

[例] 对于本节的示例，每次产生33位的0/1随机数，组成一个初始个体。假设产生20个初始个体，组成初始群体如下：

v_1=(1 0 0 1 1 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 0 1 1 0 1 1 1 1)

v_2=(1 1 1 0 0 0 1 0 0 1 0 0 1 1 0 1 1 1 0 0 1 0 1 0 1 0 0 0 1 1 0 1 0)

v_3=(0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 1 0 1 0 1 1 1 0 1 1 1 0 1)

v_4=(1 0 0 0 1 1 0 0 0 1 0 1 1 0 1 0 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0 1 0)

v_5=(0 0 0 1 1 1 0 1 1 0 0 1 0 1 0 0 1 1 0 1 0 1 1 1 1 1 0 0 0 1 0 1)

v_6=(0 0 0 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 1 1 1 1 1 0 1 1)

v_7=(0 0 1 0 0 0 1 0 0 0 0 0 1 1 0 1 0 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1)

v_8=(1 0 0 0 0 1 1 0 0 0 0 1 1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 1 0 0 1 1 1)

v_9=(0 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 1 1 0 0)

v_{10}=(0 0 0 0 0 1 1 1 1 0 0 0 1 1 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1 0 1 1)

v_{11}=(0 1 1 0 0 1 1 1 1 1 1 0 1 1 0 1 0 1 1 0 0 0 0 1 1 0 1 1 1 1 0 0 0)

v_{12}=(1 1 0 1 0 0 0 1 0 1 1 1 1 0 1 1 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0)

v_{13}=(1 1 1 0 1 1 1 1 1 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 1 1 0)

v_{14}=(0 1 0 0 1 0 0 1 1 0 0 0 0 0 1 0 1 0 1 0 0 1 1 1 1 0 0 1 0 1 0 0 1)

v_{15} = (1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 0 0 0 1 0 0 0 1 1 1 1 1 0 1 1 1 1 0)

v_{16} = (1 1 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 0 1 0 0 1 0 1 1)

v_{17} = (0 1 1 0 1 0 1 1 1 1 1 0 0 1 1 1 1 0 1 0 0 0 1 1 0 1 1 1 1 0 1)

v_{18} = (0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0 1 1 1 1 1 0 1 0 1 1 0 1)

v_{19} = (0 0 0 1 0 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0 1 1 0 0)

v_{20} = (1 0 1 1 1 0 0 1 0 1 0 0 1 1 1 1 0 0 1 1 0 0 0 1 0 1 1 1 1 1 1 0)

2.1.3 计算适应度

适应度是衡量个体优劣的标志，它是执行遗传算法“优胜劣汰”的依据。因此，适应度也是驱使遗传算法向前发展的动力。

通常，遗传算法中个体的适应度也就是所研究问题的目标函数，但是，有时适应度是目标函数转换后的结果。

为了讨论的方便，本章的遗传算法只研究目标变量 x 大于零的最大值问题。对于最小值问题，其适应度按下式转换：

$$f(x)=\begin{cases} C_{max}-g(x) & \text{当 } g(x)<C_{max} \\ 0 & \text{其他情况} \end{cases} \tag{2-6}$$

式中 $f(x)$ ——转换后的适应度；

$g(x)$ ——最小值问题下的适应度；

C_{max} ——足够大的常数，可取 $g(x)$ 的最大值。

为了保证适应度不出现负值，对于有可能产生负值的最大值问题，可采用下式变换：

$$f(x)=\begin{cases} U(x)+C_{min} & \text{当 } U(x)+C_{min}>0 \\ 0 & \text{其他情况} \end{cases} \tag{2-7}$$

式中 $f(x)$ ——变换后的适应度；

$U(x)$ ——最大值问题的适应度；

C_{min} ——足够小的常数。

[例] 对于本节的示例，对初始个体 $v_1 \sim v_{20}$ 的 0/1 表达式进行解码，求出相应的 x_1、x_2，再代入式(2.1)求目标函数，得适应度：

$f(v_1) = f(6.084492, \quad 5.652242) = 26.019600$

$f(v_2) = f(10.348434, \quad 4.380264) = 7.580015$

$f(v_3) = f(--2.516603, \quad 4.390381) = 19.526329$

$f(v_4) = f(5.278638, \quad 5.593460) = 17.406725$

$f(v_5) = f(--1.255173, \quad 4.734458) = 25.341160$

$f(v_6) = f(-1.811725, \quad 4.391937) = 18.100417$

$f(v_7) = f(-0.991471, \quad 5.680258) = 16.020812$

$f(v_8) = f(4.910618, \quad 4.703018) = 17.959701$

$f(v_9) = f(0.795406, \quad 5.381472) = 16.127799$

$f(v_{10}) = f(-2.554851, \quad 4.793707) = 21.278435$

$f(v_{11}) = f(3.130078, \quad 4.996097) = 23.410669$

$f(v_{12}) = f(9.356179, \quad 4.239457) = 15.011619$

$f(v_{13}) = f(11.134646, \quad 5.378671) = 27.316702$

$f(v_{14}) = f(1.335944, \quad 5.151378) = 19.876294$

$f(v_{15}) = f(11.089025, \quad 5.054515) = 30.060205$

$f(v_{16}) = f(9.211598, \quad 4.993762) = 23.867227$

$f(v_{17}) = f(3.367514, \quad 4.571343) = 13.696165$

$f(v_{18}) = f(3.843020, \quad 5.158226) = 15.414128$

$f(v_{19}) = f(-1.746635, \quad 5.395584) = 20.095903$

$f(v_{20}) = f(7.935998, \quad 4.757338) = 13.666916$

很明显，上述个体中 v_{15} 最佳(30.060205)，v_2 个体最差(7.580015)。

2.1.4 复制

在遗传算法中通过复制，将优良个体插入下一代新群体，体现“优胜劣汰”的原则。

选择优良个体的方法，通常采用 J. H. Holland 教授推荐的轮

盘法　轮盘法的基本精神是个体被选中的概率取决于个体的相对适应度：

$$p_i = f_i / \Sigma f_i \tag{2-8}$$

式中　p_i　　个体 i 被选中的概率；

f_i　　个体 i 的适应度；

Σf_i　　群体的累加适应度。

显然，个体适应度愈高，被选中的概率愈大。但是，适应度小的个体也有可能被选中，以便增加下一代群体的多样性。图 2-2 描述轮盘选择的原理。图中指针固定不动，外圈的圆环可以自由转动，圆环上的刻度代表各个个体的适应度。当圆环旋转若干圈后停止，指针指定的位置便是被选中的个体。从统计意义讲，适应度大的个体，其刻度长，被选中的可能性大；反之，适应度小的个体被选中的可能性小，但有时也会被"破格"选中。

表 2-1 详细描述这种选择方式。表中第一行说明有 10 个个体参与选择，第二行表示各个个体的适应度，第三行标记适应度的累加值，总值为 76。然后，在[0，76]区间中产生均匀分布的随机数，如第四行所示。依顺序将第三行的累计适应度与随机数相比较，其值大于或等于随机数的第一个个体为入选的复制对象。例如，第一个随机数是 23，除了 1 号、2 号个体外，其余个体的累计适应度均大于 23，然而 3 号个体累计值为 27，是第一个大于 23 的个体，所以它入选

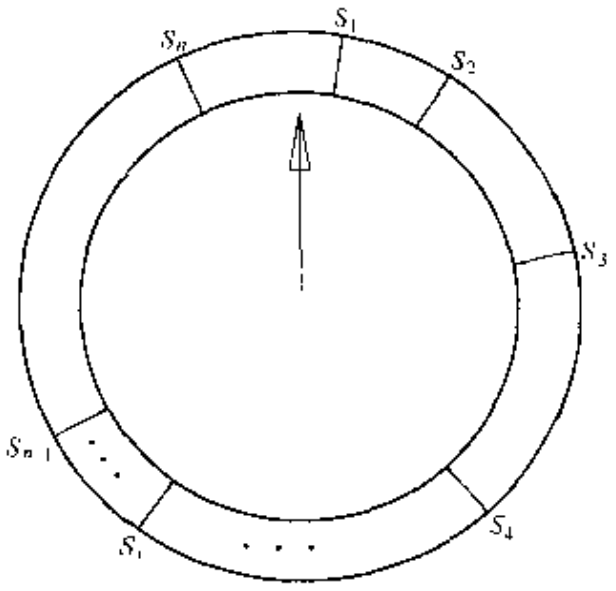

图 2-2　轮盘选择原理

表 2-1　轮盘选择示例

个体序号	1	2	3	4	5	6	7	8	9	10
适应度 f_i	8	2	17	7	2	12	11	7	3	7
适应度累计值 s_i	8	10	27	34	36	48	59	66	69	76
随机数 r	23	49	76	13	1	27	57			
被选中的个体号	3	7	10	3	1	3	7			

上述轮盘选择过程，可描述如下：

(1)顺序累计群体内各个体的适应度，得相应的累计值 S_i，最后一个累计值为 S_n；

(2)在 $[0, S_n]$ 区间内产生均匀分布的随机数 r；

(3)依次用 S_i 与 r 比较，第一个出现 S_i 大于或等于 r 的个体 i 被选为复制对象；

(4)重复(2)、(3)项，直到满足所需要的复制个体数目。

至于群体中被复制的个体数目，常用期望值法。这种方法规定复制概率 p_r，即

$$p_r = \frac{m}{M} \tag{2-9}$$

式中　M ——群体中个体数目；

　　m ——群体中被复制的个体数目。

由此得 $m = p_r \cdot M$。p_r 常取 0.1～0.2。也就是说，群体中有 10%～20%个体被淘汰，以保持群体规模。至于被淘汰的劣质个体的选择，可以是适应度最差的个体(确定型)，也可以是轮盘法(随机型)按下述概率选择：

$$p_i = \frac{f_{max} - f_i}{\sum f_i}$$

式中　f_{max} ——个体适应度的最大值。

近年来，在遗传算法中常常采用择优选择法。这种方法没有明显的复制操作，而是根据个体的相对适应度 $f_i/\Sigma f_i$ 反复地从群体中选择 M 个个体组成下一代群体。当然，个体的适应度愈高，它被

重复选中的可能性愈大，而重复选中的就相当复制。相反，适应度小的个体往往未能选中，它就被淘汰。这种选择方法和上述方法相类似，其操作过程为：

(1)顺序累计群体中各个体的适应度 f_i，得适应度的累加值 S_n：

$$S_n = \sum_i f_i$$

(2)用 S_n 除各个体的适应度 f_i，得相对适应度 p_i，它也就是该个体被选中的概率：

$$p_i = f_i / S_n$$

(3)累计 p_i 得累积概率 g_i：

$$g_i = \sum_{j=1}^{i} p_j$$

(4)产生[0,1]均匀分布的随机数 r；

(5)将 r 与 g_i 相比较，若 $g_{i-1} < r < g_i$，则选个体 i 进入下一代新群体；

(6)反复执行(4)及(5)项，直至新群体的个体数目等于父代群体规模。

[例] 在本节示例中，初始群体的累积适应度 S_n 为：

$$S_n = \sum_{i=1}^{21} f_i = 387.776822$$

各个个体的相对适应度 p_i、累积概率 g_i 及随机数 r 如表 2-2 所示。

表 2-2 择优选择法

个体 i	相对适应度 p_i	累积概率 g_i	随机数 r	选中的个体 v_i
1	0.067099	0.067099	0.513870	v_{11}
2	0.019547	0.086646	0.175741	v_4
3	0.050355	0.137001	0.308652	v_7
4	0.044889	0.181890	0.534534	v_{11}
5	0.065350	0.247240	0.947628	v_{19}

续表 2-2

个体 i	相对适应度 p_i	累积概率 g_i	随机数 r	选中的个体 v_i
6	0.046677	0.293917	0.171736	v_4
7	0.041315	0.335232	0.702231	v_{15}
8	0.046315	0.381546	0.226431	v_5
9	0.041590	0.423137	0.491773	v_{11}
10	0.054873	0.478009	0.124720	v_3
11	0.060372	0.538381	0.703899	v_{14}
12	0.038712	0.577093	0.389647	v_9
13	0.070444	0.647537	0.277226	v_6
14	0.051257	0.698794	0.368071	v_8
15	0.077519	0.776314	0.983437	v_{20}
16	0.061549	0.837863	0.005398	v_1
17	0.035320	0.873182	0.465682	v_{10}
18	0.039750	0.912932	0.646473	v_{13}
19	0.051823	0.964756	0.767139	v_{15}
20	0.035244	1.000000	0.780237	v_{16}

第 1 个随机数 $r = 0.513870$，它大于 g_{10} 而小于 g_{11}，于是个体 v_{11} 被选中进入下一代新群体。第 2 个随机数 $r = 0.175741$，它大于 g_3 而小于 g_4，使个体 v_4 入选。重复这个过程，最后有 20 个个体组成新群体，如表中最后一列所示。

从新群体中可以看出，个体 v_{15} 及 v_{11} 被选中 3 次，这是由于它们的适应度较高。个体 v_4 尽管适应度较低，但被选中两次，从而增加新群体的多样性。相反，个体 v_2、v_{12}、v_{14}、v_{17} 及 v_{18} 则被淘汰。

2.1.5 交换

在遗传算法中，交换是产生新个体的主要手段。它类似于生物学的杂交，使不同个体的基因互相交换，从而产生新个体。图 2-3 表示交换的基本原理，图中涂黑的数码表示交换的字符，左侧是两

个父代个体，右侧是两个子代个体。

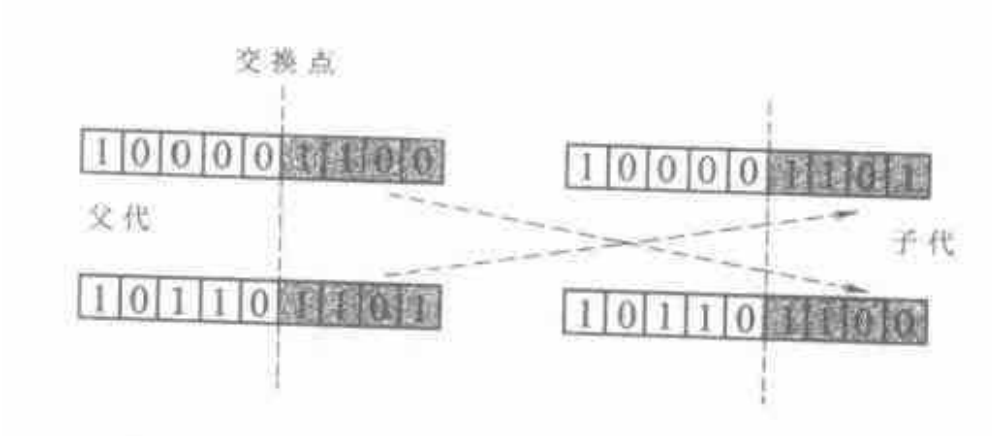

图 2-3　交换原理

通常，被交换的个体从复制后的新群体中随机选择。选择的方法也是轮盘法，使优良个体尽可能被选中，劣质个体被选中的可能性小，但也有可能被“破格”选中。

控制被交换个体数目 M_c 的参数是交换概率 p_c，即：

$$M_c = p_c \cdot M \tag{2-10}$$

式中　M —— 群体中个体的数目；

p_c —— 交换概率，群体中被交换个体的比例，常取 0.2～0.6。

假若 M_c 为奇数，则要增选一个交换个体或删去一个个体，使交换对象成对出现。

交换点的选择也是随机的。若字符串的长度为 L，则在[1，$L-1$]区间内产生均匀分布的随机整数，该整数便是交换点的位置。

通过交换，子代的字符串不同于父代。有时，这种差别很明显，表 2-3 的第一组交换便是这种差别明显的例子。有时，这种差别却不大，如表中第二组交换，只有最后一个字符发生变化。尽管如此，交换仍是遗传算法产生新个体的主要手段。正是有了交换操作，群体的性态才多种多样。

表 2 3　交换

序　号	1	2
交换前	父代 1：1 1 1 ⋮ 1 1 1	父代 1：1 0 1 ⋮ 1 0 1
	父代 2：1 0 0 ⋮ 0 0 0	父代 2：0 0 1 ⋮ 1 0 0
交换后	子代 1：1 1 1 ⋮ 0 0 0	子代 1：1 0 1 ⋮ 1 0 0
	子代 2：1 0 0 ⋮ 1 1 1	子代 2：0 0 1 ⋮ 1 0 1

［例］　本节示例中，假设 $p_c = 0.25$。由于群体规模 $M = 20$，故有 $p_c \cdot M = 5$ 个个体要被交换。然而 5 是奇数，在增加个体或删除个体中随机选择，决定删除个体。采用类似于复制中的择优选择法，选中 v_4 与 v_{15}、v_6 与 v_{13} 进行交换，交换点分别是第 9 及第 20 位，其交换过程如表 2-4。

表 2-4　交换结果

交换前	v_4：	1 0 0 0 1 1 0 0 0 ⋮ 1 0 1 1 0 1 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 1 0
	v_{15}：	1 1 1 0 1 1 1 0 1 ⋮ 1 0 1 1 1 0 0 0 0 1 0 0 0 1 1 1 1 1 0 1 1 1 1 0
交换后	v'_4：	1 0 0 0 1 1 0 0 0 ⋮ 1 0 1 1 1 0 0 0 0 1 0 0 0 1 1 1 1 1 0 1 1 1 1 0
	v'_{15}：	1 1 1 0 1 1 1 0 1 ⋮ 1 0 1 1 0 1 0 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0 1 0
交换前	v_6：	0 0 0 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 ⋮ 1 0 1 0 1 1 1 1 1 1 0 1 1
	v_{13}：	1 1 1 0 1 1 1 1 1 0 1 0 0 0 1 0 0 0 1 1 ⋮ 0 0 0 0 0 0 1 0 0 0 1 1 0
交换后	v'_6：	0 0 0 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 ⋮ 0 0 0 0 0 0 1 0 0 0 1 1 0
	v'_{13}：	1 1 1 0 1 1 1 1 1 0 1 0 0 0 1 0 0 0 1 1 ⋮ 1 0 1 0 1 1 1 1 1 1 0 1 1

2.1.6　突变

突变是遗传算法产生新个体的另一种方法。它对某个字符进行补运算，将字符 1 变为 0，或将 0 变为 1。

突变是针对字符执行的，因此突变概率 p_m 也是针对字符而言，即：

$$p_m = \frac{B}{M \cdot L} \tag{2-11}$$

式中　B — 每代中突变的字符数目；

M——每代中群体拥有的个体数目；

L ——个体中字符串长度。

通常，p_m 约为 0.005～0.01。

突变字符的位置也是随机确定的，如表 2-5 所示。某群体有 3 个个体，每个体含 4 个字符。针对每个个体的每个字符产生一个[0,1]区间具有 3 位有效数字的均匀随机数。假设突变概率 p_m = 0.01，则随机数小于 0.01 的对应字符产生突变。表中 3 号个体的第 4 位数的随机数为 0.001，小于 0.01，该字符产生突变，使 3 号个体由 0010 变为 0011。其余字符的随机数均大于 0.01，不产生突变。

表 2-5 突变示例

个体编号	10 个体	随机数				新个体
1	1 0 1 0	0.801	0.102	0.266	0.373	
2	1 1 0 0	0.120	0.796	0.105	0.840	
3	0 0 1 0	0.760	0.473	0.894	0.001	0 0 1 1

[例] 本节示例中，每代有 20 个个体，每个体有 33 个字符，即每代字符数为 20×33=660。若突变概率 p_m = 0.01，则每代平均有 6.6 个字符产生突变。为此，针对每个字符产生一个[0,1]区间的随机数。一旦该随机数小于 0.01，则该字符产生突变，表 2-6 列举突变的结果。表中说明，个体 v_4、v_{11}、v_{13} 及 v_{19} 等 4 个个体的 5 个字符产生突变。应该指出，实际产生突变的字符数少于平均数 6.6，而且 13 号个体有两个字符产生突变，这都是随机选择的结果。

表 2-6 突变结果

序 号	字符位置	随机数	个体编号	个体中字符位置	原来字符	突变后字符
1	1 1 2	0.000213	4	1 3	1	0
2	3 4 9	0.009945	11	1 9	1	0

续表 2-6

序　号	字符位置	随机数	个体编号	个体中字符位置	原来字符	突变后字符
3	4 1 8	0.008809	13	2 2	0	1
4	4 2 9	0.005425	13	3 3	0	1
5	6 0 2	0.002836	19	8	0	1

经过复制、交换及突变，本示例中第 1 代群体如表 2-7 所示。从表中可以看出，新一代群体的累计适应度 $S_n = 447.049688$，比初始群体的 387.776822 有了很大的提高。而且新一代的最优个体 v_{11} 的适应度为 33.351874，也比初始群体的最优适应度 (v_{15})30.060205 有所增长。

表 2-7　新一代群体

个体编号	字　　符　　串	适应度 f_i
v_1	0 1 1 0 0 1 1 1 1 1 0 1 1 0 1 0 1 1 0 0 0 1 1 0 1 1 1 1 0 0 0	23.410669
v_2	1 0 0 0 1 1 0 0 0 1 0 1 1 1 0 0 0 0 1 0 0 0 1 1 1 1 1 0 1 1 1 1 0	18.201083
v_3	0 0 1 0 0 0 1 0 0 0 0 0 0 1 1 0 1 0 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1	16.020812
v_4	0 1 1 0 0 1 1 1 1 1 1 0 0 1 0 1 0 1 1 0 0 0 0 1 1 0 1 1 1 1 0 0 0	23.412613
v_5	0 0 0 1 0 1 0 1 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0 1 1 0 0	20.095903
v_6	1 0 0 0 1 1 0 0 0 1 0 1 1 0 1 0 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0 1 0	17.406725
v_7	1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 0 0 0 1 0 0 0 1 1 1 1 1 0 1 1 1 1 0	30.060205
v_8	0 0 0 1 1 1 0 1 1 0 0 1 0 1 0 0 1 1 0 1 0 1 1 1 1 1 1 0 0 0 1 0 1	25.341160
v_9	0 1 1 0 0 1 1 1 1 1 1 0 1 1 0 1 0 1 1 0 0 0 0 1 1 0 1 1 1 1 0 0 0	23.410669
v_{10}	0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 1 0 1 0 1 1 1 0 1 1 1 0 1	19.526329
v_{11}	1 1 1 0 1 1 1 0 1 1 0 1 1 0 1 0 0 1 0 1 1 0 0 0 0 0 1 1 1 0 0 1 0	33.351874
v_{12}	0 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 1 1 0 0	16.127799
v_{13}	0 0 0 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 1 1	22.692462
v_{14}	1 0 0 0 0 1 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 1 0 0 1 1 1	17.959701

续表 2-7

个体编号	字　　符　　串	适应度 f_i
v_{16}	1 0 1 1 1 0 0 1 0 1 1 0 0 1 1 1 1 0 0 1 1 0 0 0 1 0 1 1 1 1 1 1 0	13.666916
v_{17}	1 0 0 1 1 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 0 1 1 0 1 1 1 1 1	26.019600
v_{18}	0 0 0 0 0 1 1 1 1 0 0 0 1 1 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1 0 1 1	21.278435
v_{19}	1 1 1 0 1 1 1 1 1 0 1 0 0 0 1 0 0 0 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1	27.591064
v_{20}	1 1 1 0 1 1 1 0 0 1 0 1 1 1 0 0 0 0 1 0 0 0 1 1 1 1 1 0 1 1 1 1 0	27.608441
v_{21}	1 1 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1 0 1 0 0 1 0 1 1	23.867227
合计		447.049688

2.1.7　终止

遗传算法是一个反复迭代的过程，每次迭代期间，要执行适应度计算、复制、交换、突变等操作，直至满足终止条件。

使遗传算法终止的方法有三种：

(1) 规定最大迭代次数 N。一旦遗传算法的迭代次数达到 N，则停止操作，输出结果。通常 N 取 200 ～ 500 次。

由于遗传算法中有许多随机因素影响，最后一代的结果不一定含有最优个体。为此，要经常记录每代的最优个体以便查询比较。例如，本节示例中，规定 $N = 1000$。然而第 1000 代的最优个体（1 1 0 1 0 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1 0 0 0 1 0 1 1 0 0 0 0）的适应度 $f = 35.477938$，不及第 396 代的最优个体 $f = 38.827553$。

(2) 规定最小的偏差 δ。对于适应度目标已知的遗传算法，如用方差作为适应度计算的曲线拟合问题，可用最小的偏差 δ 制定终止条件，即：

$$|f_{max} - f^*| \leqslant \delta \tag{2-12}$$

式中　f^* ——已知的适应度目标；

f_{max} ——每代最大的适应度。

(3) 观察适应度的变化趋势。在遗传算法的初期，最优个体的

适应度以及群体的平均适应度都较小，以后随着复制、交换、突变等操作，适应度值增加。到了遗传算法后期，这种增加已趋缓和或停止，一旦这种增加停止，即中止遗传算法。

2.2 遗传算法的表述

从前面 2.1 的示例可以看出，遗传算法的实施过程包括编码、产生初始群体、计算适应度、复制、交换、突变、反复迭代、终止等操作。

图 2-4 详细描述遗传算法的流程。图中 Gen 代表遗传(迭代)的代次，遗传算法从 Gen＝0 开始。根据所研究问题的表达方式确定字符串长度 L，接着随机产生 M 个初始群体。刚开始时，终止条件不会被满足，于是依次计算群体中各个个体的适应度。根据计算结果，依次执行复制、交换、突变等遗传操作。首先执行复制，其复制概率为 p_r。图中 j 代表遗传操作次数，从 $j=0$ 开始。根据适应度大小，选择优质个体作为复制对象，将复制结果添入新群体中。复制过程不断重复，直到复制次数满足 $p_r \cdot M$。与此同时，从父代群体中删除同样数目的劣质个体，将剩余的个体进入下一代，使新一代群体规模 M 保持不变。其次执行交换，其交换概率为 p_c。交换次数从 $j=0$ 开始。根据个体的适应度每次选择两个个体执行交换，然后将新形成的两个个体代替父代旧个体添入新群体中。交换过程反复执行，直至交换次数满足 $p_c \cdot M$ 为止。第三个遗传操作是突变，突变概率为 p_m。突变针对字符进行，发生突变的字符总数为 $p_m \cdot L \cdot M$。经过复制、交换及突变，产生新一代群体，代次 Gen 又增加 1 次。再对新一代群体计算适应度，执行遗传操作。如此重复迭代，直至满足终止条件。

概括地讲，遗传算法的工作步骤是：

(1)确定个体的字符串的组成及长度。

(2)随机建立初始群体。

(3)计算各个体的适应度。

(4)根据遗传概率，用下述操作产生新群体：

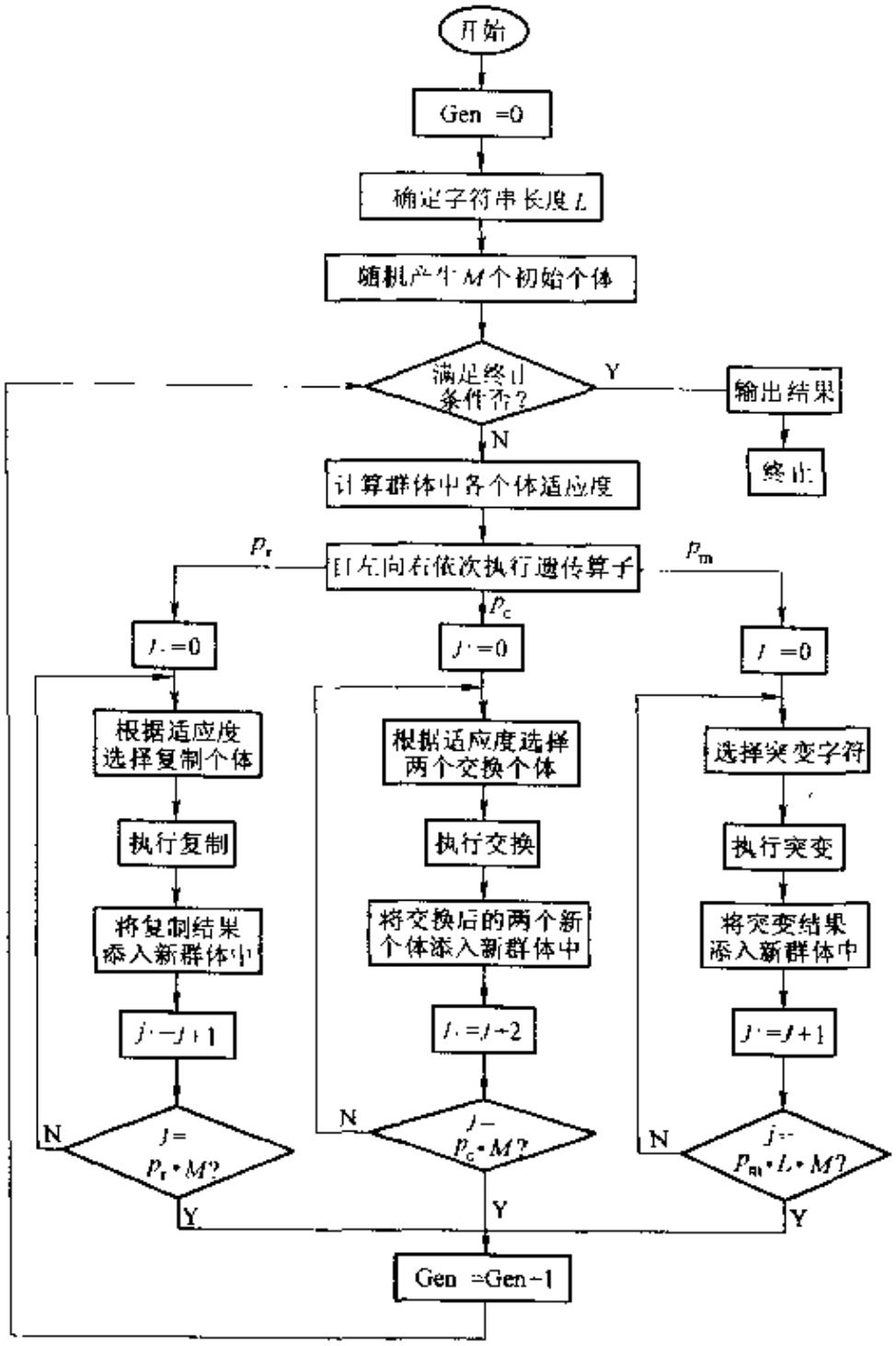

图 2-4　遗传算法流程图

1)复制，将已有的优良个体复制后添入新群体中，删除劣质个体。

2)交换。将选出的两个个体进行交换，所产生的新个体进入新群体。

3)突变。随机地改变某个体的某一字符后，将新个体添入新群体。

(5)反复执行(3)及(4)，直至达到终止条件，选择最佳个体作为遗传算法的结果。

遗传算法还可以用形式化语言表达。假设 $a \in I$ 记为个体，I 记为个体空间。适应度函数记为 $\Phi: I \to R$。在第 t 代，群体 $P(t) = \{a_1(t), a_2(t), \cdots, a_n(t)\}$ 经过复制 r、交换 c 及突变 m 转换成下一代群体。这里 r, c, m 均指宏算子，把旧群体变换为新群体。$l: I^n \to \{\text{True}, \text{False}\}$ 记为终止准则。利用上述符号，遗传算法可描述为：

```
t=0
initialize P(0) : = {a1(0), a2(0), ···, an(0)};
while ( l (P (t)) ≠ True ) do
    evaluate P(t): {Φ(a1(t)),Φ(a2(t)),···,Φ(an(t))};
    reproduction: P'(t) : =r(P(t));
    crossover: P''(t) : =c(P'(t));
    mutation: P(t+1) : =m(P''(t));
    t==t+1;
end
```

2.3 模式理论

指导遗传算法的基本理论，是 J. Holland 教授创立的模式理论。该理论揭示遗传算法的基本机理。

2.3.1 基本概念

2.3.1.1 问题的引出

回顾前面 1.2.1 关于最大值问题的例子。在寻找 $f(x) = x^2$，$x \in [0,31]$ 的最大值中，第 0 代初始个体的编码及适应度如表 2-8 所示。从表中可以看出，编码中左数第一位字符为 1 时，其适应

度较大，如 2 号个体（$f(x)=576$）及 4 号个体（$f(x)=361$），我们不妨记作 1＊ ＊ ＊ ＊，其中＊表示我们不关心的字符，可为 0 或 1。进一步观察，2 号个体的适应度最大，其编码的左数前两位字符都是 1，即 11＊ ＊ ＊。相反，左数第一位字符为 0 时，其适应度较小，如 1 号、3 号个体，可记作 0＊ ＊ ＊ ＊。

表 2-8 编码与适应度

序 号	编 码	适应度 $f(x)$
1	01101	169
2	11000	576
3	01000	64
4	10011	361

从这个例子可以看出，我们在分析字符串时，常常只关心某一位或某几位字符，而对其他字符不关心。换句话讲，我们只关心字符串的某些特定形式，如 1＊ ＊ ＊ ＊ ＊或 11＊ ＊ ＊，这种特定的形式就叫模式

2.3.1.2 模式、阶次及长度

模式(Schema)是指字符串中具有类似特征的子集。以五位二进制的字符串为例，模式＊111＊可代表 4 个个体：{01110，01111，11110，11111}；模式＊0000 则代表 2 个个体：{10000，00000}。

图 2-5 描述模式的几何意义。图中表示三位二进制字符串 $(x_1x_2x_3)$ 的各种组合，其中 x_i 在 {0，1，＊} 三种字符中选值。8 个顶点表示 8 个明确的 0/1 字符串，12 条边表示只有一个＊符号的模式，6 个平面表示含 2 个＊号的模式。至于含有三个＊号的模式（＊ ＊ ＊），则是立方体本身，对于位数大于 3 的字符串，要用超平面的几何概念解释模式。

模式的阶次(Order)是指模式中已有明确含意(二进制字符时指 0 或 1)的字符个数，记作 $O(H)$，式中 H 代表模式。例如，模式（011＊1＊ ＊）含有 4 个明确含意的字符，其阶次是 4，记作

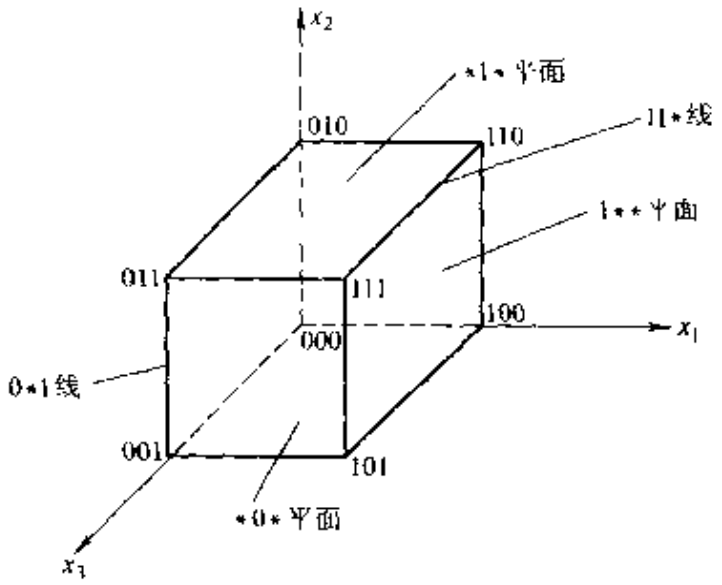

图 2.5　模式的几何意义

$O(0\ 1\ 1\ *\ 1\ *\ *)=4$。模式$(0\ *\ *\ *\ *\ *\ *)$的阶次是1。很明显，阶次越低，模式的概括性越强，所代表的字符串个体数也越多，反之亦然。当模式阶次为零时，它没有明确含义的字符，其概括性最强。

模式的定义长度(Defining Length)是指模式中最前面和最后面两个具有明确含意的字符之间的距离，记作$\delta(H)$。例如，模式$(0\ 1\ 1\ *\ 1\ *\ *)$的最前面字符为0，最后面字符为1，中间有3个字符，其定义长度为4，记作$\delta(011\ *\ 1\ *\ *)=4$。模式$(0\ *\ *\ *\ *\ *\ *)$的长度是0。一般地，有式子

$$\delta(H)=b-a \tag{2-13}$$

式中　b ——模式H中最后一个明确字符的位置；

a ——模式H中最前一个明确字符的位置。

模式的长度代表该模式在今后遗传操作(交换、突变)中被破坏的可能性。模式长度越短，被破坏可能性越小。例如，长度为0的模式最难被破坏。

2.3.1.3　字符串的模式数目

根据模式及阶次的定义，我们可以计算字符串所含有的模式数目。

(1) 模式总数。以二进制字符串为例，假设字符串的长度为 L，字符串中每一个字符可取{0，1，*}三个符号中任意一个，于是，可能组成的模式数目最多为：

$$3 \times 3 \times 3 \times \cdots \times 3 - (2+1)^L$$

一般情况下，假设字符串长度为 L、字符的取值为 k 种，则该字符串组成的模式数目 n_1 最多为：

$$n_1 = (k+1)^L \tag{2-14}$$

(2) 已知阶次的模式所含字符串的总数。对于二进制的字符串，若字符串长度为 L，某种模式的阶次为 $O(H)$，则在 L 个字符串中取 $O(H)$ 个字符的可能组合方式为 $C_L^{O(H)}$，而在 $O(H)$ 的字符中具体取0或1的可能性为 $2^{O(H)}$。于是，组成 H 的各种字符串数目最多为 $C_L^{O(H)} \times 2^{O(H)}$。

一般情况下，假设字符串长度为 L、字符的取值为 k 种，模式 H 的阶次为 $O(H)$，则组成 H 的数目 n_2 最多为：

$$n_2 = C_L^{O(H)} \cdot k^{O(H)} \tag{2-15}$$

(3) 字符串所含模式总数。对于长度为 L 的某二进制字符串，它含有的模式总数最多为 $2 \times 2 \times 2 \times \cdots \times 2 = 2^l$ 个。注意，这个数目是指字符串已确定为0或1，每个字符只能在已定值(0/1)或 * 中选取；前面所述的 n_1 是指字符串未确定，每个字符可在{0，1，*}三者中选取。

一般情况下，长度为 L、取值有 k 种的某一字符串，它可能含有的模式数目 n_3 最多为：

$$n_3 = k^L \tag{2-16}$$

2.3.2 遗传过程的模式数目及模式定理

上述的字符串数目 n_1、n_2 及 n_3，可以说是静态下的模式数目。下面我们从动态角度，讨论遗传过程中的模式数目。

2.3.2.1 复制时的模式数目

假设在第 t 次迭代时，群体 A 中有 n 个个体，其中 m 个个体属于模式 H，记作 $m(H,t)$。复制时，个体 a_i 是根据其适应度 f_i 的大小进行复制。从统计意义讲，个体 a_i 被复制的概率 p_i 是：

$$p_i = f_i / \Sigma f_i$$

因此复制后在下一代 $(t+1)$ 中，群体 A 内属于模式 H 的个体数目 $m(H,t+1)$ 可用平均适应度按下式近似计算：

$$m(H,t+1) = m(H,t) \cdot n \cdot f(H)/\Sigma f_i$$

式中 $f(H)$ —— 第 t 代属于模式 H 的所有个体之平均适应度；

n —— 群体中拥有的个体数目。

假设第 t 代所有个体（不论它属于何种模式）的平均适应度是 $\bar{f}$，有等式

$$\bar{f} = \Sigma f_i / n$$

综合上述二式，复制后模式 H 所拥有的个体数目可按下式近似计算：

$$m(H,t+1) = m(H,t) \frac{f(H)}{\bar{f}} \tag{2-17}$$

式(2-17)是模式计算的基本公式，它说明复制后下一代群体中属于模式 H 的个体数目，取决于该模式平均适应度 $f(H)$ 与群体的平均适应度 $\bar{f}$ 之比。换句话讲，只有当模式 H 的平均值 $f(H)$ 大于群体的平均值 $\bar{f}$ 时，H 模式的个体数目才能增长。否则，H 模式的数目要减小。模式 H 的这种增减规律，正好符合复制操作的“优胜劣汰”原则，这也说明模式的确能描述字符串的内部特征。请注意，$f(H)$ 及 $\bar{f}$ 都是平均适应度，但是前者是针对模式 H 拥有的个体，而 $\bar{f}$ 是指全部个体。

进一步，假设某一模式 H 在复制过程中其平均适应度 $f(H)$ 比群体的平均适应度 $\bar{f}$ 高出一个定值 $c\bar{f}$，其中 c 为常数，则(2-17)式改写作：

$$m(H,t+1) = m(H,t) \frac{\bar{f} + c\bar{f}}{\bar{f}} = (1+c) \cdot m(H,t)$$

从第一代开始，若模式 H 以常数 c 繁殖到第 $t+1$ 代，其个体数目为：

$$m(H,t+1) = m(H,1) \cdot (1+c)^t \tag{2-18}$$

从数学上讲，上式是一个指数方程，它说明模式 H 所拥有的

个体数目在复制过程中是以指数形式增加或减小。

2.3.2.2 交换时的模式数目

我们先看一个简单例子。假设有一个7位字长($L=7$)的二进制字符串A及其模式H_1、H_2:

$$\begin{aligned} A &= 0\ 1\ 1\ 1\ 0\ 0\ 0 \\ H_1 &= *\ 1\ *\ *\ *\ *\ 0 \\ H_2 &= *\ *\ *\ 1\ 0\ *\ * \end{aligned}$$

很明显,模式H_1及H_2均能代表字符串A。假设通过随机产生的方法,确定交换点在左数第3位字符后面,用分割线“ ”表示,有:

$$\begin{aligned} A &= 0\ 1\ 1 \quad 1\ 0\ 0\ 0 \\ H_1 &= *\ 1\ *\ \vdots\ *\ *\ *\ 0 \\ H_2 &= *\ *\ *\ \vdots\ 1\ 0\ *\ * \end{aligned}$$

从上式可以看出,模式H_1由于定义长度大($\delta(H_1)=5$),交换后其结构被破坏;模式H_2由于定义长度小($\delta(H_2)=1$),其模式结构保存下来。假设交换位置是用均匀分布的随机数确定,由于在最后一个字符后发生交换没有意义,所以模式(H_1)在长度为$L-1=7-1=6$个字符中任一字符开始发生交换的概率是$1/(L-1)=1/6$。交换发生在模式H_1的定义长度$\delta(H_1)$范围内的概率是:

$$p_d = \delta(H_1)/(L-1) = 5/6$$

p_d也就是模式H_1被破坏的概率。换句话讲,模式H_1存活下来的概率p_s为:

$$p_s = 1 - p_d = 1/6$$

同理,我们可以计算出模式H_2的$p_d=1/6$,$p_s=5/6$,它从数量上说明模式H_2比H_1容易保存。

上述计算过程没有考虑交换的概率p_c,即交换并非每个个体都使用。一般的,模式H经交换保存下来的概率是:

$$p_s = 1 - p_c \frac{\delta(H)}{(L-1)} \tag{2-19}$$

若 $p_c = 1.0$，式(2-19)也就是在示例中模式 H_1、H_2 的计算结果。

式(2-19)从数量上说明，模式的定义长度 $\delta(H)$ 对模式的存亡有很大影响。$\delta(H)$ 越长，H 存活的可能性越小，反之亦然。

2.3.2.3 突变时的模式数目

突变时字符串的每一个字符发生变化的概率是突变率 p_m。也就是说，每个字符存活的概率是$(1-p_m)$。根据模式的阶次 $O(H)$，可知模式中有明确含意的字符有 $O(H)$ 个，于是模式 H 存活的概率是：

$$p_{ms} = (1-p_m)(1-p_m)\cdots(1-p_m) = (1-p_m)^{O(H)}$$

通常，$p_m \ll 1$，上式用台劳级数展开取一次项，可近似表达为：

$$p_{ms} = 1 - O(H) \cdot p_m \tag{2-20}$$

式(2-20)从数量上说明，模式的阶次 $O(H)$ 越低，模式 H 存活的可能性越大，反之亦然。

2.3.2.4 模式定理

综合式(2-17)、(2-19)、(2-20)，可以得出遗传算法经复制、交换、突变操作后模式 H 在下一代群体中所拥有的个体数目，如下式所示：

$$m(H,t+1) = m(H,t)\frac{f(H)}{\bar{f}}\left[1 - p_c\frac{\delta(H)}{(L-1)} - O(H)p_m\right] \tag{2-21}$$

式(2-21)是遗传算法的基本理论公式，它说明所有长度短、阶次低、平均适应度高于群体平均适应度的模式 H 在遗传算法中呈指数形式增长。相反，凡是长度长、阶次高、平均适应度低于群体平均适应度的模式将呈指数形式消失。这个结论很重要，以至人们称之为模式定理(Schema Theorem)。

模式定理深刻地阐明遗传算法中发生“优胜劣汰”的原因。在遗传过程中能存活的模式都是定义长度短、阶次低、平均适应度高于群体平均适应度的优良模式。遗传算法正是利用这些优良模式

逐步进化到最优解。

2.3.3 模式定理示例

我们再次利用 1.2.1 关于最大值的例子。表 2-9 列举遗传算法中字符串和模式的变化情况。

表 2-9 字符串及模式的变化情况

[字符串变化]

序号	初始群体	x_i	适应度 $f(x_i)$	$f(x_i)/\sum f(x_i)$	$f(x_i)/\bar{f}$	实际数目	复制后的新群体	交换对象	交换点位置	新群体	x_i	适应度 $f(x_i)$
1	01101	13	169	0.14	0.58	1	01101	2#	3	01100	12	144
2	11000	24	576	0.49	1.97	2	11000	1#	3	11001	25	625
3	01000	8	64	0.06	0.22	0	11000	4#	2	11011	27	729
4	10011	19	361	0.31	1.23	1	10011	3#	2	10000	16	256
合计 $\sum f(x_i)$			1170	1.00	4.00	4.0						1754
平均值 $\bar{f}$			293	0.25	1.00	1.0						439
最大值			576	0.49	1.97	2.0						729

[模式变化]

	复制前			复制后			复制及交换后		
	拥有的个体号		模式的平均适应度 $f(H)$	预期数目	实际数目	拥有的个体号	预期数目	实际数目	拥有的个体
H_1	1 * * * *	2 号,4 号	469	3.20	3	2 号,3 号,4 号	3.20	3	2 号,3 号,4 号
H_2	* 10 * *	2 号,3 号	320	2.18	2	2 号,3 号	1.64	2	2 号,3 号
H_3	1 * * * 0	2 号	576	1.97	2	2 号,3 号	0.0	1	1 号,

表 2-9 的上半部[字符串变化],取自表 1-1 及 1-2,它说明初始群体经过复制、交换后变成新群体的过程。表 2-9 的下半部[模式变化],表示三种模式 H_1、H_2、H_3 的变化过程。

模式 H_1 的定义长度最小($\delta(H_1)=0$),阶次最低($O(H_1)=1$)。在初始群体中,H_1 代表 2 号及 4 号个体,其平均适应度 $f(H_1)=(576+361)/2=468.5$,高于初始群体适应度的平均值 $\bar{f}=293$。利用公式(2-17),可得复制后新群体中模式 H_1 所代表的个体

数目：

$$m(H_1,t+1)=2\times 468.5/293=3.20$$

实际上复制后的新群体中，2 号、3 号、4 号三个个体都属于模式 H_1，与理论计算值一致。在交换过程中，由于 $\delta(H_1)=0$，$p_c=1$，从公式(2 19)可以得出交换后 H_1 的存活概率是：

$$p_c=1-1\times 0/(5-1)=1$$

即交换操作对 $m(H_1,t+1)$ 没有影响。假设突变率 $p_m=0.004$，由公式(2 20) 可计算出突变存活概率为：

$$p_m=1-1\times 0.004=0.996$$

H_1 基本上不会破坏。总之，模式 H_1 由于长度小、阶次低、平均适应度高，属于优良模式，在遗传过程中不断增长。

对于模式 H_3，其定义长度最大($\delta(H_3)=4$)，阶次 $O(H_3)=2$，在初始群体中它仅代表 2 号个体。不过它的平均适应度较高($f(H_3)=576$)，超过群体的平均适应度 $\bar{f}=293$，因此复制后拥有的个体数目要增加。利用公式(2-17)，可算得：

$$m(H_3,t+1)=1\times 576/293=1.97\doteq 2$$

从表 2-9 可知，实际上 H_3 代表复制后新群体的 2 号、3 号个体，与理论结果相符。在交换时，由于 H_3 的长度大，由公式(2-19) 可计算存活率：

$$p_s=1-p_c\frac{\delta(H)}{(L-1)}=1-1\times 4/4=0$$

再代入公式(2-21)，当不考虑突变时有：

$$m(H_3,t+1)=1\times 576/293\times 0=0$$

表 2 9 中，H_3 在复制、交换后的新群体中只代表 4 号个体。总之，模式 H_3 因适应度高在复制中增长，但由于长度大在交换中却衰减。

同理，模式 H_2 的定义长度为 1、阶次为 2，在初始群体中代表 2 号、3 号个体，利用公式(2-17)，有：

$$m(H_2,t+1)=2\times 320/293=2.18$$

表 2-9 中实际上 H_2 代表复制后新群体中的 2 号、3 号个体，与计

算结果相符。利用公式(2-21)，当不考虑突变时有：

$m(H_2,t+1)=2\times 320/293\times(1-1\cdot 1/(5-1))=1.64$

表 2-9 在复制及交换后的新群体中 H_2 代表 2 号、3 号个体，与计算结果相近。

应该指出，表 2-9 中理论计算值与实际数目稍有误差的原因，是由于群体中的个体数目太少。公式(2-17) ～ (2-21) 都是从统计意义上推导出来的，一旦个体数目足够大，计算的精确度会明显提高。

2.3.4 构造块

根据模式定理，遗传算法在运算过程中，利用定义长度短、阶次低、平均适应度高的优良模式，不断建立高适应度的字符串，从而逐渐逼近问题的最优解。由于这种短定义长度、低阶、高适应度的模式在遗传算法中有如此重要的作用，我们特别称之为构造块(Building Block)。遗传算法正是利用这些构造块建立性能优越的个体。

为了形象地说明构造块的作用，我们仍用前述求函数 $f(x)=x^2$ 的最大值为例，x 用 5 位二进制字符串表示。

图 2-6 表示模式 $H_1=1****$ 的分布。图中横坐标表示

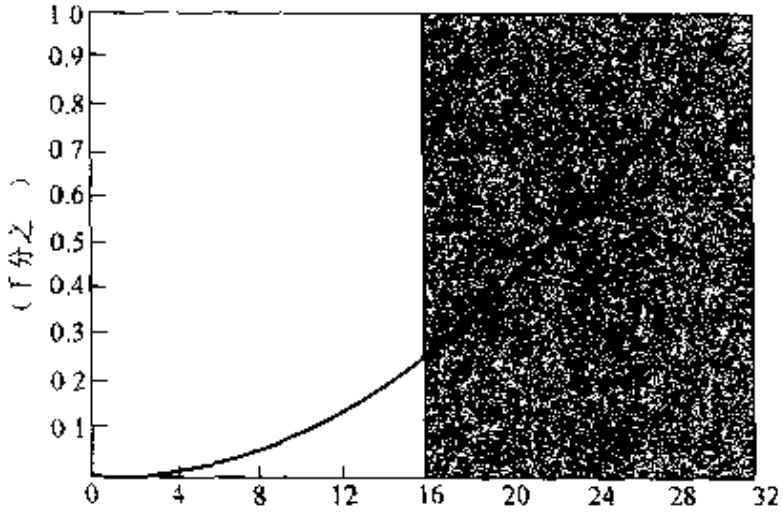

图 2-6 模式 1 * * * * 的分布

x，纵坐标代表适应度 $f(x)=x^2$，后者用千分数表示。图中弧线表示适应度曲线，网点区代表所有符合此模式的字符串集合。在此模式下，x 最小值的字符串是 1 0 0 0 0($x=16$)，其适应度 $f(x)=x^2=256$；最大值的字符串是 1 1 1 1 1 ($x=31$)，其适应度为 961。若初始群体中含有字符串 1 0 0 0 0，则此模式沿此网点区逐步向字符串 1 1 1 1 1 逼近。因此，模式 H_1 属于构造块，可产生性能优良的字符串。

图 2-6 的左半部空白区代表模式 $H_2=0\ *\ *\ *\ *$ 的分布。在此模式下，x 最小值的字符串是 0 0 0 0 0($x=0$)，其适应度为 0；最大值的字符串是 0 1 1 1 1($x=15$)，其适应度 225。很明显，这种模式不能达到最优解，在后期要消亡。

图 2-7 表示另一个特殊模式 $H_3=*\ *\ *\ *\ 1$ 的分布。在这种模式下，x 最小值的字符串是 0 0 0 0 1($x=1$)，其适应度为 1；最大值的字符串是 1 1 1 1 1，其适应度为 961。对于这种模式，x 值只能为奇数，所以模式分布在斑状条带内，如图中网点区所示。这种模式尽管是离散分布，它仍能达到最优解。

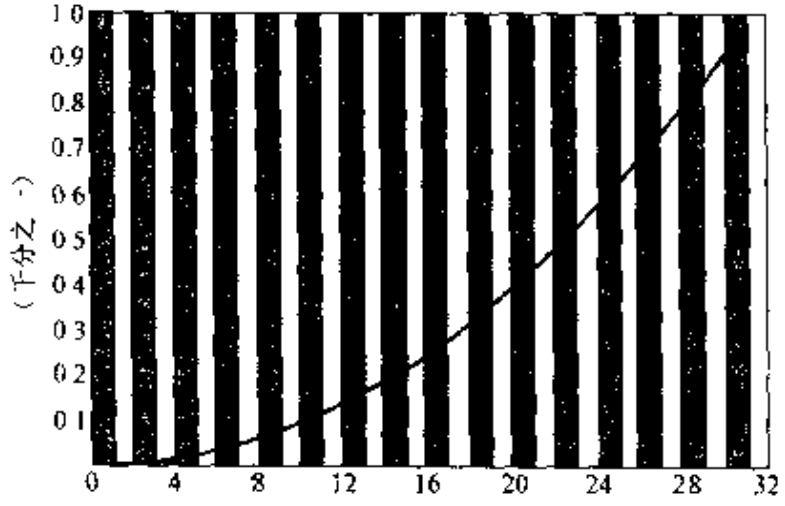

图 2-7　模式 * * * * 1 的分布

图 2-8 是模式 $H_4=*\ *\ *\ 0\ *$ 的分布。这时，字符串的最小值是 00000($x=0$)，最大值是 11101($x=29$)。在这种模式下，x 值只能在[0,2]、[4,6]、[8,10]… 区间内，所以模式呈较宽的斑状

条带分布。这种模式只能接近最优解($x = 30$),但不能达到 $x = 31$。

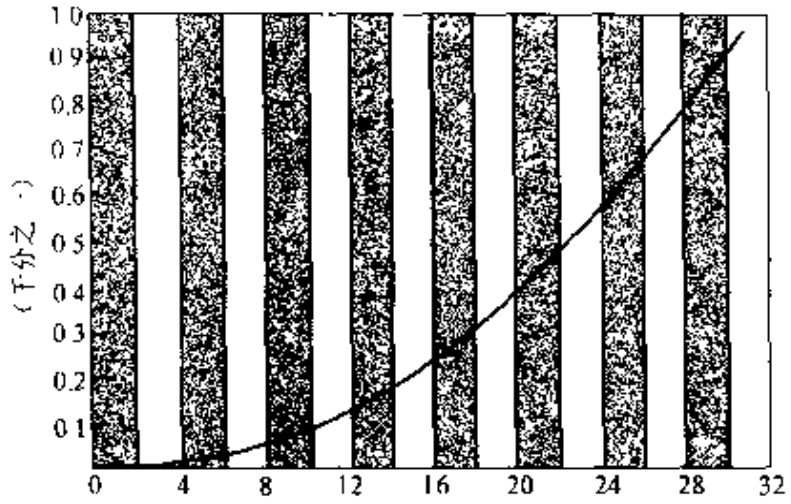

图 2-8 模式 * * * 0 * 的分布

图 2-9 是模式 $H_5 = 10 * * *$ 的分布图。这时,字符串介于 $10000(x = 16) \sim 10111(x = 24)$,分布在中间的宽条带中。很明显,这种模式无法到达最优解。

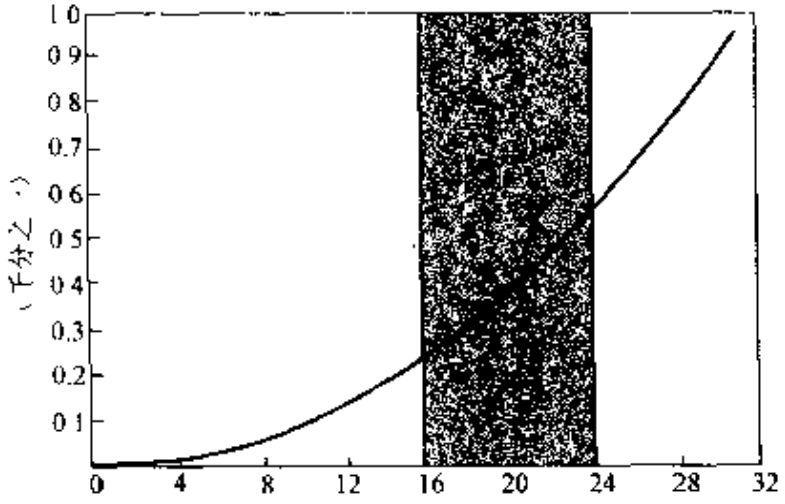

图 2-9 模式 1 0 * * * 的分布

图 2-10 是模式 $H_6 = * * 1 * 1$ 的分布图。其字符串分布在成对的斑状条带中。

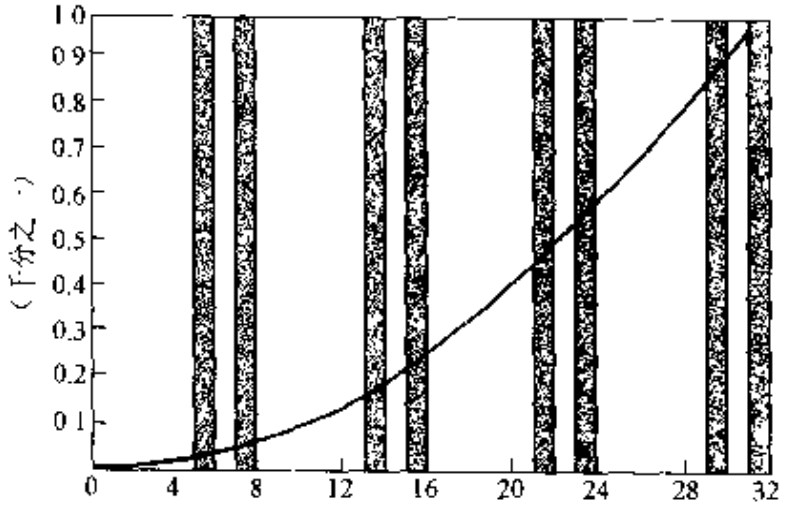

图 2-10　模式 * * 1 * 1 的分布

综观图 2-6～图 2-10 可以看出，符合模式 H_1、H_3、H_6 的字符串可以逼近最大值，模式 H_2、H_4、H_5—— 尤其是 H_2 及 H_5，不可能逼近最大值，H_2 及 H_5 的字符串在后期要消亡。

2.3.5　隐并行机理

隐并行机理(Implicit Parallelism)是模式理论的另一个重要内容。这一机理说明，在遗传算法中尽管每一代只处理 n 个个体，但实际上却是处理 n^3 以上模式。下面我们将证明这个机理的存在。

2.3.5.1　模式存活的最小字长 L_s

从模式定理中可以看出，模式在交换和突变时可能遭破坏。由于突变概率很小，我们先考虑交换的破坏。由公式(2-19)可知，对于字长为 L、模式定义长度为 $\delta(H)$ 的模式 H，当交换概率为 p_c，H 被破坏的概率 p_d 是：

$$p_d = 1 - p_s = p_c \frac{\delta(H)}{(L-1)}$$

即：

$$\frac{p_m}{p_c} = \frac{\delta(H)}{(L-1)} = \varepsilon$$

上式只考虑了交换，若再兼顾突变的破坏作用，令 ε 为模式 H 遭交换及突变破坏的可能性，则上式可写作：

$$\varepsilon \cdot (L-1) = \delta(H) \tag{2-22}$$

式中 $\delta(H)$ 是指破坏概率为 ε 的模式长度。根据模式定义长度的定义，模式不被破坏的最小字长 L_s 是：

$$L_s = \delta(H) + 1$$

代入式(2-22)有：

$$L_s - 1 < \varepsilon(L-1)$$

即：

$$L_s < \varepsilon(L-1) + 1$$

2.3.5.2 字符串中拥有字长为 L_s 的模式数目

我们首先看一个示例。假设下述字符串长度 $L = 10$：

1 0 1 1 1 0 0 0 1 0

若模式 H 的存活字长 $L_s = 5$。将它放置在字符串的最左侧，则有：

[1 0 1 1 1] 0 0 0 1 0

写成模式的形式，上述字符串变为：

[% % % % 1] * * * * *

方框中 % 可在{0，1，*}三者中任取一个。也就是说，% 可为固定值(0/1)或不固定值(*)两种情况。方框内的 1 表示有一个确定的模式，也可以选方框内的其他固定值表示。这时，可以组成的模式个数是 $2 \times 2 \times \cdots\cdots \times 2 = 2^{(L_s-1)}$。

将上述方框右移一位，有：

1 [0 1 1 1 0] 0 0 1 0

其模式表达式为：

* [% % % % 0] * * * *

在这种情况下又可以组成 $2^{(L_s-1)}$ 个模式。

上述方框可发生在 6 个不同位置上，即发生次数为 $L - L_s +$

1。于是，长度为 L 的字符串可组成字长为 L_s 的模式数目 n_{s1} 是：

$$n_s = 2^{(L_s-1)} \times (L - L_s + 1)$$

2.3.3.3 群体中的模式数目

当群体由 n 个字符串组成时，可能组成的字长为 L_s 的模式数目 n_{s2} 为：

$$n_{s2} = n \cdot n_{s1} = n \times 2^{(L_s-1)} \times (L - L_s + 1)$$

在复制中产生的个体完全相同。根据模式定理，它们都是一些不易被破坏的低阶模式。因此按上式计算的 n_{s2} 显然偏大，我们不妨保守地取 $n = 2^{L_s/2}$。

另一方面，由于遗传操作都是利用均匀随机数，模式数目服从二项分布，即模式中有一半的阶次高于 $L_s/2$，另一半小于 $L_s/2$。于是，计算时模式数目应取上述的 1/2。

综合上述各方面，可能存活的模式数目 n_s 为：

$$\begin{aligned} n_s &= n \times 2^{L_s-1} \times (L - L_s + 1) \times 1/2 \\ &= n \times n^2 \times 1/2 \times (L - L_s + 1) \times 1/2 \end{aligned}$$

即：

$$n_s = \frac{(L - L_s + 1)}{4} n^3 = Cn^3$$

上式说明，遗传算法中存活的模式数目 n_s 是群体中个体数目 n 的三次方。表面上遗传算法每一代只处理 n 个个体，但实质上却处理了 Cn^3 个能存活的模式。因此，遗传算法实际上是一种并行算法，隐藏在字符串的背后，故称隐并行算法。正是由于这种隐并行性，遗传算法的搜索效率很高。

2.4 遗传算法的实施技术

80 年代以后，遗传算法得到了广泛的使用，在实践过程中，人们对遗传算法的实施提出了许多改进。本节分别予以介绍。

2.4.1 编码

2.4.1.1 二进制编码的优越性

根据模式理论，二进制编码具明显的优越性。对于长度为 L 的

二进制字符串，它可以表达的个体数目 n_1 为：

$$n_1 = 2 \times 2 \times \cdots\cdots \times 2 = 2^L$$

对于长度为 L' 的 k 进制字符串，其表达的个体数目 n_2 为：

$$n_2 = k \cdot k \cdots\cdots k = k^L$$

对于同一问题，$n_1 = n_2$，故有：

$$2^L = k^L \tag{2-23}$$

从表达的模式数目看，二进制字符除了 0、1 外，还有 * 字符，所以它表达的模式数目 m_1 为：

$$m_1 = (2 + 1)^L = 2^L(1 + \frac{1}{2})^L$$

同理，k 进制字符串表达的模式数目 m_2 为：

$$m_2 = (k + 1)^{L'} = k^{L'}(1 + \frac{1}{k})^{L'}$$

由于 $k > 2$，由公式(2 23) 可知：

$$L > L$$

另有：

$$(1 + \frac{1}{2}) > (1 + \frac{1}{k})$$

即：

$$(1 + \frac{1}{2})^L > (1 + \frac{1}{k})^L$$

将上述不等式及公式(2-23)代入 m_1 或 m_2 的表达式，可得

$$m_1 > m_2$$

这就是说，二进制字符串拥有的内涵(模式数目)比 k 进制字符串大。

为了形象地说明问题，表 2-10 列举十进制、二进制及英文字母的字符串的比较。从表中可以明显看出，二进制字符串所表达的模式多于十进制及英文字母，在执行交换及突变时可以有更多的变化。

表 2 10　二进制编码的优越性

英文字母	A	B	C	D	E	F	G
十进制	1	2	3	4	5	6	7
二进制	00001	00010	00011	00100	00101	00110	00111
英文字母	H	I	J	K	…	Y	Z
十进制	8	9	10	11	…	25	26
二进制	01000	01001	01010	01011	…	11001	11010

2.4.1.2　格雷码

近年来，遗传算法中常常采用格雷码(Gray Code)。格雷码是一种循环的二进制字符串，它与普通二进制数的转换如下式所示：

由标准二进制码 a_i 转换到格雷码 b_i

$$b_i = \begin{cases} a_i & \text{若 } i = 1 \\ a_{i-1} \oplus a_i & \text{若 } i > 1 \end{cases} \tag{2-24}$$

由格雷码 b_i 转换到标准二进制码 a_i

$$a_i = \bigoplus_{j=1}^{i} b_j \tag{2-25}$$

式中⊕表示以 2 为模的加运算，即：

$$0 \oplus 0 = 0, \quad 0 \oplus 1 = 1$$
$$1 \oplus 0 = 1, \quad 1 \oplus 1 = 0$$

以标准二进制字符串 1 0 1 1 为例，转换为格雷码时，左数第 1 位同原来二进制为 1，第 2 位为 1⊕0=1，第 3 位 0⊕1=1，第 4 位为 1⊕1=0，最后格雷码为 1 1 1 0。反之，设格雷码字符串为 1011，转换为标准二进制时第 1 位仍为 1，第 2 位为 1⊕0=1，第 3 位为(1⊕0)⊕1=0，第 4 位为[(1⊕0)⊕1]⊕1=1，即 1 1 0 1，

表 2-11 列举格雷码与十进制整数及二进制码的比较。从表中可以看出，相邻两个格雷码只有一个字符的差别。通常，相邻两个二进制字符串中字符不同的数目称作海明距离。格雷码的海明距离总是 1。这样，在进行突变操作时，格雷码某个字符的突变很有可能使字符串变为相邻的另一个字符串，从而实现顺序搜索，避免

无规则的跳跃式搜索。

表 2-11　四位格雷码

十进制整数	标准二进制码	格雷码
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1
10	1 0 1 0	1 1 1 1
11	1 0 1 1	1 1 1 0
12	1 1 0 0	1 0 1 0
13	1 1 0 1	1 0 1 1
14	1 1 1 0	1 0 0 1
15	1 1 1 1	1 0 0 0

格雷码在交换中的作用可从下述示例中看出。假设在遗传算法的后期，有两个父代个体如下：

父代个体 1：　[… 1 0 0 0 0 0 0 0 …]=[… 1 2 8 …]

父代个体 2：　[… 0 1 1 1 1 1 1 1 …]=[… 1 2 7 …]

中间的二进制数表示一段有用的基因，其值分别为 128 及 127。假设交换点选在第二个二进制字符之后，则子代新个体为：

子代个体 1：　[… 1 0 0 1 1 1 1 1 …]=[… 1 5 9 …]

子代个体 2：　[… 0 1 1 0 0 0 0 0 …]=[… 9 6 …]

很明显，交换使个体发生明显变化，其值变为 159 及 96，不利于遗

传算法收敛。特别当字符串很长时，更是加剧这种离散现象。假若采用格雷码表达个体，上述父代个体变成：

父代个体 1：［… 1 1 0 0 0 0 0 0 …］=［… 1 2 8 …］

子代个体 2：［… 0 1 0 0 0 0 0 0 …］=［… 1 2 7 …］

若交换点仍在第三字符之后，则子代个体为：

子代个体 1：［… 1 1 0 0 0 0 0 0 …］=［… 1 2 8 …］

子代个体 2：［… 0 1 0 0 0 0 0 0 …］=［… 1 2 7 …］

新个体的数值没有改变，从而有利于遗传算法时收敛。不过，有人作过试验，采用格雷码后遗传算法的收敛速度只提高 10%～20%，作用不明显；但有人却宣称格雷码能明显提高收敛速度。

2.4.2 适应度

2.4.2.1 适应度的缩放

在遗传算法初始阶段，各个个体的性态明显不同，其适应度大小差别很大。个别优良个体的适应度有可能远远高于其他个体，从而增加被复制的次数。反之，个别适应度很低的个体，尽管本身含有部分有益的基因（字符），但却会被过早舍弃。这种不正常的取舍，对于个体数目不多的群体尤为严重，会把遗传算法的搜索引向误区，过早地收敛于局部最优解。这时，需要将适应度按比例缩小，减少群体中适应度的差别。另一方面，当遗传算法到了后期，群体逐渐收敛，各个体的适应度差别不大，为了更好地优胜劣汰，希望适当地放大适应度，突出个体之间的差别。无论是缩小或放大适应度，可用下式变换适应度：

$$f' = af + b \tag{2-26}$$

式中 f' ——缩放后的适应度；

f ——原来的适应度；

a、b ——系数。

图 2-11 描述适应度缩放原理。图中的缩放斜线采用两点连线的方法确定：一个点是适应度的平均点 A。从统计意义讲，缩放前及缩放后的平均适应度应该相同，即 A 点处有：

$$f_{avg} = f_{avg}'$$

式中 f_{avg} —— 原来的适应度平均值；

f'_{avg} —— 缩放后的适应度平均值。

连线的另一点 B 是适应度的最大值点，可用下式确定：

$$f'_{max} = C \cdot f_{avg}$$

式中 f'_{max} —— 缩放后的最大适应度；

C —— 系数。当群体中个体数目为 50 ～ 100 时，常取 $C = 1.2 \sim 2$。

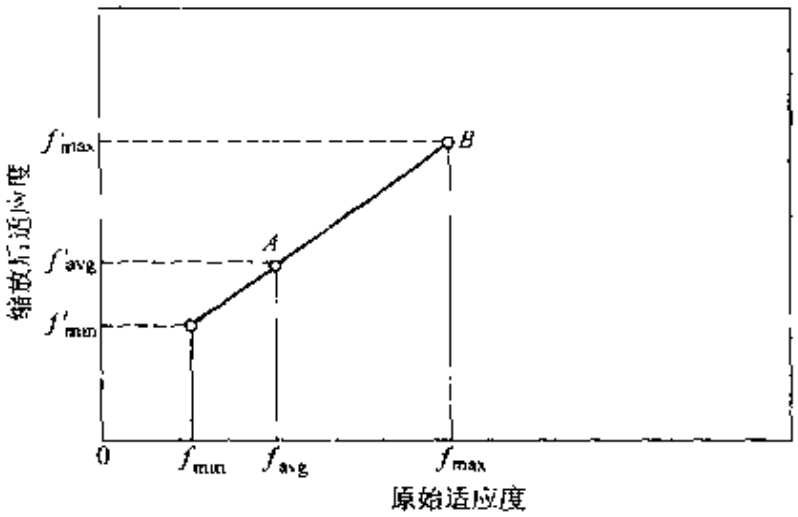

图 2-11 适应度缩放原理

在遗传算法执行的后期，个别劣质个体的适应度远远小于群体平均适应度及最大适应度，并且后两者比较接近。这时按上述方法缩放适应度会使低适应度变成负值，如图 2-12 的 C 点。为此，需要修正上述的两点连线法。平均点 A 的位置仍然同前，即 $f_{avg} = f'_{avg}$。B 点则用负值点的映射点代替，即：

$$f'_{min} = 0$$

综上所述，缩放斜线式(2-26)的参数 a、b 计算方法如下：

(1)计算适应度非负判别式：

$$f_{min} > \frac{C \cdot f_{avg} - f_{max}}{C - 1} \tag{2-27}$$

若不等式满足，执行(2)，否则则执行(3)。

(2)正常情况下的缩放：

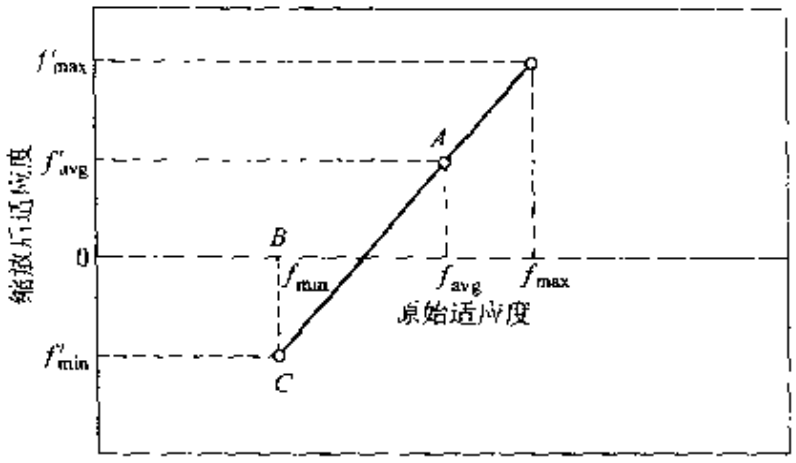

图 2-12　负值点的映射

$$a = \frac{(C-1)}{f_{max} - f_{avg}} \cdot f_{avg} \tag{2-28}$$

$$b = \frac{f_{max} - C \cdot f_{avg}}{f_{max} - f_{avg}} f_{avg} \tag{2-29}$$

(3)负适应度时的缩放

$$a = \frac{f_{avg}}{f_{avg} - f_{min}} \tag{2-30}$$

$$b = -\frac{f_{min} \cdot f_{avg}}{f_{avg} - f_{min}} \tag{2-31}$$

调整适应度的另一种方法是方差缩放技术，它根据适应度的离散情况进行缩放。对于适应度离散的群体，调整量要大一些。反之，调整量减少。具体调整方法如下：

$$f' = f + (\bar{f} - C \cdot \sigma) \tag{2-32}$$

式中　$\bar{f}$ ——适应度的平均值；

σ ——群体适应度的标准差；

C ——系数，介于 1～5 间的整数。

若计算出的 f' 为负值，令其为零。

上述调整公式，有人建议改为相对形式，即：

$$f' = f + \frac{\overline{f} - C \cdot \sigma}{2C \cdot \sigma}$$

也有人建议采用指数缩放方法，即：

$$f' = f^k \tag{2-33}$$

式中 k ——系数，在 1 附近。

上述调整适应度的各种方法，其目的都是修改各个体性能的差距，以便体现“优胜劣汰”的原则。例如，假若我们想多选择一些优良个体进入下一代，则尽量加大适应度之间的差距。

2.4.2.2 约束条件的处理

对于有约束的优化问题，目标函数中的目标变量要服从一定的约束条件，即：

$$\max \quad f(x_1, x_2, \cdots, x_n)$$

$$\text{s.t.}: \quad h_i(x_1, x_2, \cdots, x_n) \leqslant 0, \quad i = 1, 2, \cdots, m$$

对于这类问题，最常用的方法是采用罚函数的方法，将有约束的极值问题变为无约束的极值问题。在这种方法中，原来的问题变为下述一个新目标函数 F：

$$\max \quad F = f(x_1, x_2, \cdots, x_n) + \delta \sum_{i=1}^{m} \phi_i(h_i(x_1, x_2, \cdots, x_n)) \tag{2-34}$$

式中 ϕ ——惩罚函数；

δ ——惩罚系数。

最简单的惩罚函数形式是平方法，即：

$$F = f(x_1, x_2, \cdots, x_n) + \sum_{i=1}^{m} [h_i(x_1, x_2, \cdots, x_n)]^2 \tag{2-35}$$

也有人建议采用松紧法。在遗传算法初期，惩罚轻一些；到了后期，则惩罚重一些。具体的新目标函数 F 为：

$$F = f(x_1, x_2, \cdots, x_n) + K \cdot (\frac{t}{T})^\rho \cdot \overline{f} \cdot \sum_{i=1}^{m} d_i \tag{2-36}$$

式中 K —— 系数，可取 $1/m$；

t —— 当前的迭代代次；

T —— 计划的总迭代代次；

ρ —— 系数，1 左右；

$\bar{f}$ —— 个体适应度的平均值；

d_i —— 违背约束 i 的程度。

d_i 视问题的不同而定义。例如，对于下述约束条件：

$$\Sigma_{i\in w}x_i = f, \qquad w = \{1, 2, \cdots, n\}$$

则 d_i 可定义为：

$$d_i = |\Sigma_{i\in w}x_i - f|$$

不过，实践证明，这种罚函数并非永远奏效。对于运输问题这类具有众多约束条件及约束变量的情况，遗传算法执行过程中往往有不符合约束的解出现。

处理约束条件的第二种方法是解码修改法。这种方法将新产生的个体解码为十进制数，检查它是否满足约束条件。若违背约束条件，适当修正该个体的数码，使之符合约束。现以运输问题为例说明，假设 V_i、V_j、V_k、V_m 为个体的四个正值分量（图 2-13），其中 $i<j<k<m$。当前状态下 V_i 与 V_j、V_k 与 V_m 在同一行，V_i 与 V_k、V_j 与 V_m 在同一列。遗传算法执行过程中要求产生的 V_i、V_j、V_k、V_m 满足运输约束，即每行（列）分量之和为常数。

$$\cdots \cdots \cdots$$
$$\cdots V_i \cdots V_j \cdots$$
$$\cdots \cdots \cdots$$
$$\cdots V_k \cdots V_m \cdots$$
$$\cdots \cdots \cdots$$

图 2-13 修改法示例

假设突变操作使 V_i 变为 $V_i + C$ ($C>0$)。此时，其他三个分量要作下述修正：

$$V_j = V_j - C$$
$$V_k = V_k - C$$
$$V_m = V_m + C$$

然而有可能出现 $V_i < C$ 或 $V_k < C$ 而违背变量非负约束，或 $V_m > u_m$（u_m 为允许的最大值）而违背最大值约束，因此 C 应按下式修正：

$$C = \min (V_i, V_k, u_m - V_m)$$

显然，这种解码修正法费时费力，特别是对于交换算子，由于交换后的子代个体明显不同于父代，这种修正更感困难。

第二种处理约束条件的方法是简化约束条件，进而改进遗传算子。例如，有运输问题：

$$\begin{aligned}
\max \quad & f(x_1, x_2, x_3, x_4, x_5, x_6) \\
\text{s.t.:} \quad & x_1 + x_2 + x_3 = 5 \\
& x_4 + x_5 + x_6 = 10 \\
& x_1 + x_4 = 3 \\
& x_2 + x_5 = 4 \\
& x_1, x_2, x_3, x_4, x_5, x_6 \geqslant 0
\end{aligned}$$

首先，简化约束条件，仅用 x_1 及 x_2 两个变量。从约束方程组中得：

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

于是，目标函数 f 变为：

$$g(x_1, x_2) = f(x_1, x_2, (5 - x_1 - x_2), (3 - x_1), (4 - x_2), (3 + x_1 + x_2))$$

其约束条件是：

$$\begin{aligned}
& x_1 \geqslant 0, x_2 \geqslant 0 \\
& 5 - x_1 - x_2 \geqslant 0 \\
& 3 - x_1 \geqslant 0 \\
& 4 - x_2 \geqslant 0 \\
& 3 + x_1 + x_2 \geqslant 0
\end{aligned}$$

上述约束方程组进一步可化简为：

$$0 \leqslant x_1 \leqslant 3$$

$$0 \leqslant x_2 \leqslant 4$$

$$x_1 + x_2 \leqslant 5$$

其次，修改遗传算子，使衍生的新个体满足约束条件。假设在搜索空间内有一点 $X = (x_1, x_2) = (1.8, 2.3)$。若不改变 x_2 而只改变 x_1，则 x_1 的波动范围为 $[0, 5 - x_2] = [0, 2.7]$，从而限制了 x_1 的选择空间。进一步，在搜索空间中若有另外一点 $X' = (x_1', x_2) = (0.9, 3.5)$，则 X 与 X' 连线上的各点，即线性组合：

$$\alpha X + (1 - \alpha) X', \qquad 0 \leqslant \alpha \leqslant 1$$

也满足约束条件。根据上述单点波动范围和双点内插范围，可以改进突变和交换，使之满足约束要求。

2.4.2.3 减少适应度的计算

对于有些复杂的课题，其遗传算法的适应度计算需要占用大量的 CPU 时间。为了减少适应度的计算次数，可以采用下述方法。

(1) 控制初始群体。使初始个体散布在广阔的搜索空间，避免初始个体彼此相近，从而使今后的群体在大范围内散布，以加快遗传算法的搜索速度。为了使初始个体彼此有明显差别，最简单的方法是使字符串的前几位数字有规律地造成差别，而其后的字符则仍是随机选取。例如，有下述 8 个字符串作为初始群体：

0 0 0 1 0 1 0 1 0

0 0 1 1 0 1 0 1 0

0 1 0 1 0 1 0 1 0

0 1 1 0 0 1 1 0 0

1 0 0 1 1 0 0 1 1

1 0 1 0 0 0 1 1 1

1 1 0 1 1 0 0 0 0

1 1 1 0 1 1 0 1 1

这 8 个初始个体的前 3 位字符是故意顺序选取，而后 6 位则是随

机产生，从而保证各个体有明显的差别。取前3个个体为例，尽管后6位字符都是101010，但是由于前3位字符按000、001、010顺序排列，从而使它们有明显差别。

当个体的字符串是由不同含义的字段（基因）组成时，为了使初始群体离散分布，同样是使每个字段（基因）的初始字符有差别。例如，某个体由9个二进制字符组成，每3个字符代表一种性质（基因）。这时，可将第1、第4及第7个字符有意分散选取，而其他字符则是随机产生，它们的结果如下：

111**1**11**1**11
100**1**00**0**00
111**0**11**1**00
100**0**00**0**11
000**1**00**1**00
000**1**00**0**00
010**0**10**1**01
011**0**01**0**11

（2）只计算交换及突变产生的新个体。由于复制是将上一代的优良个体原封不动地移入下一代，其适应度已知，无需重新计算。因此，在每代群体中，只计算交换及突变所产生的新个体的适应度，以节约CPU时间。

（3）避免适应度重复计算。在遗传过程中，有时经交换或突变产生的新个体与群体的个体恰好是一样的，可以将原来的适应度赋予新个体，无需重复计算。因此，对交换及突变产生的新个体都要逐一检查是否已有相同的个体，避免适应度重复计算。

避免适应度重复计算的另一个方法是经常记录个体的字符串及适应度，并检查新个体是否过去曾经出现过。一旦发现相同，可将记录下来的适应度赋予新个体。

不过，这种方法需要增加查寻的时间，而且记录个体也占用存贮空间，只有当适应度计算时间的确很长时才宜采用。

（4）防止重复个体出现。在执行遗传算法时，新生的个体都要

逐个检查，一旦发现群体中已有相同的个体，则取消它进入群体的资格。此外，还要防止从事交换的两个个体相同。这样做不仅避免重复计算适应度，也增加群体的多样性，从而扩大搜索范围。

此外，还可以从简化适应度计算入手。例如，在遗传算法的初始阶段用简化方法粗略计算适应度，在后期再用精确方法计算适应度。

2.4.3 复制和选择

遗传算法在搜索过程中，一方面要选择优良个体，实现优胜劣汰，使搜索收敛于全局最优解；另一方面，在搜索过程要保持群体的多样性，使搜索空间不断扩大，避免收敛于局部最优解。因此，选择力度和群体多样性是遗传算法的一对矛盾。若加大选择力度，能提高收敛速度，但群体缺乏多样性。反之，减少选择力度，能保持群体多样性，但影响收敛速度。为此，人们对选择和复制作了许多改进。目前，常用的复制(选择)方法有三种。

(1) 适应度比例选择法(Fitness—Proportional Selection)。这种方法的选择依据是个体适应度的大小。适应度愈高，该个体被选中的可能性愈大，个体被选中的概率 p_i 为：

$$p_i = \frac{f_i}{\Sigma f_i}$$

式中 f_i —— 个体 i 的适应度；

Σf_i —— 群体中各个体适应度的累加值。

执行适应度比例选择法的手段是轮盘选择，如 2.1.4 所述。该节介绍了期望值法(有复制概率 p_i) 及择优选择法(无复制概率 p_i)。这里再推荐一种综合选择法。该方法先选择 r 个供交换、突变的个体，然后再选择剩余的个体，其具体步骤为：

1) 用轮盘法从第 t 代群体中选择 r 个个体，并标明供交换及突变用。其中数值 r 可以是常数，也可以随机变化的。

2) 再用轮盘法从第 t 代群体中选择 $M-r$ 个个体进入第 $t+1$ 代群体。其中 M 代表每代群体中的个体数。

3) 将标明的 r 个个体进行交换、突变操作，产生 r 个新个体。

4) 将 r 个新个体插入第 $t+1$ 代中。

(2)分级选择法(Ranking Selection)。遗传算法中个体适应度数值上的差别有时会很大，尤其是在算法的早期这种差别更是悬殊。因此，个别特优个体会多次被选中进行复制，经过几代后它们在群体中数目愈来愈多，冲淡了群体的多样性。因此，人们提出分级的概念，用连续渐变的分级代替数值悬殊的适应度。

设群体中有个体 a_1、a_2、…、a_m，它们按适应度大小排列：$f(a_{i-1}) \geqslant f(a_i)$，$i=1, 2, \cdots, m$，规定个体优劣的等级依次为 1，2，…，$I$，…，$M$，其中 M 为群体拥有的个体数目，也是最差个体的等级分。

最常用的分级方法是线性分级，使各个体被选中的可能性 $p(i)$ 有如下线性关系：

$$p(i) = q - (i-1)d \tag{2-37}$$

式中 q ——最优个体被选中的概率；

d ——相邻个体被选中概率之差。

上述线性关系使 $p(i)$ 构成等差级数，即：

$$q, \quad q-d, \quad q-2d, \quad q-id, \quad q-(M-1)d$$

由于概率定义要求 $\sum_{i=1}^{m} p(i) = 1$，按级数求和，有：

$$M \cdot q - \frac{M(M-1)d}{2} = 1$$

即：

$$q = \frac{(M-1)d}{2} + \frac{1}{M} \tag{2-38}$$

若 $d=0$，则 $q=1/M$ 为常数，即所有个体按相同概率选取，这是一种极端情况。另一方面，令最后(最坏)个体的选择概率为 0，即 $q-(M-1)d=0$ 代入(2-38)式有：

$$d = \frac{2}{M(M-1)} \quad 及 \quad q = \frac{2}{M}$$

这时各个体被选择的差别最大，从最大的 $2/M$ 递减为零。于是，q 可在这两种极端情况下选取，即

$$\frac{1}{M} \leqslant q \leqslant \frac{2}{M}$$

确定 q 之后，由式(2-38)即可计算 d：

$$d = \frac{2(M \cdot q - 1)}{M(M - 1)}$$

另一种分级方法是非线性分级，即：

$$p(i) = q(1 - q)^{i-1} \tag{2-39}$$

请注意，这一表达式满足概率定义的要求：

$$\sum_{i=1}^{m} p(i) \approx \sum_{i=1}^{m} q(1 - q)^{i} = 1$$

分级选择法的优点是消除个体适应度差别悬殊时的影响，代替适应度的缩放技术。然而它抹杀个体适应度的实际差别，未能充分运用遗传信息。

(3) 竞技选择法(Tournament Selection)。这种选择法通过相互竞争，优胜者成为下一代的个体。在每一代群体中，每次都随机选择 k 个个体构成一个小群体，然后从这 k 个个体中确定性地取适应度最大的个体复制，进入下一代群体。被复制后的个体仍返回父代群体中，参加下一次 k 个个体的随机选择。这种随机选择重复 M 次，产生 M 个下一代个体，具体操作步骤是：

1) 从第 t 代群体中随机选择 k 个个体；

2) 比较 k 个个体的适应度，复制适应度最大者进入第 $t+1$ 代，被复制的个体仍保留在第 t 代；

3) 重复执行 1)、2) M 次，直至产生同 t 代一样的个体数目。

竞技选择法中，选择的力度取决于 k 值的大小。k 值愈大，每次选出的优胜者具有很高的适应度。反之，k 值愈小，优胜者的适应度或高或低，随机性较强。

(4)波尔兹曼选择法(Boltzmann Selection)。这种选择方法参照模拟退火(Simulated annealing)的原理，通过控制温度 T 来调节选择力度，使选择力度随时间而变化。每个个体按下式转换其适应度：

$$Val(i,t)=\frac{e^{f_i/T}}{\left(\sum_{i=1}^{M}e^{f_i/T}\right)/M}$$

式中 $Val(i,t)$ — 第 i 个体于第 t 代的适应度转换值；

f_i — 第 i 个体的适应度值；

T —— 控制温度；

M — 群体中个体数目。

在遗传算法初期，T 值取大值，则 $Val(i,t)$ 之间的差值被缩小，性能欠佳的个体容易被选中参与复制。随着迭代过程的进行，T 值不断变小，优劣个体之间的差距增大，从而阻止劣质个体参与复制。

上述四种选择方法中，比例选择法的选择力度最小，选择所消耗的计算机时间最长，新群体的离散性最大。反之，大 k 值的竞技选择法是另一种极端情况，其选择力度最大，机时最短，新群体多为当时的优良个体。分级选择法则介于上述两者之间，波尔兹曼选择法则是一种动态选择法，其选择力度随时间而变化。

目前在遗传算法中，最常用的仍是比例选择法，其原因一是由于它是 Holland 教授最早提出的一种经典方法，二是由于它符合遗传算法的模式理论。不过，实际工作表明，比例选择法常常要作一些修正才能使优质个体确保被选中，因此近年来分级选择法、竞技选择法等新的选择方法日益被重视。

2.4.4 交换

交换是遗传算法产生新个体的主要手段。遗传算法实现交换的方法有四种。

(1)一点交换法。这种方法交换点只有一个。一旦交换点被随机选定后，交换点以后的字符串进行交换，如 2.1.5 所述。

(2) k 点交换法。若字符串较长，可选用 k 点交换。先用随机的方法选取 k 个交换点，凡是奇数交换点之后、偶数交换点之前的字符串，皆进行交换。如表 2-12 所示的两个十位字符串 p^0 及 p^1，假设随机选定 4 个交换点，则交换点 1～2、3～4 之间的字符串进行交换，产生 2 个新个体 C^0 及 C^1。

表 2-12 k 点交换

交换点		1				2		3		4	
父代	$p^0 = p_8^0 \quad p_9^0$	⋮	p_2^0	p_3^0	p_4^0	p_5^0	⋮	p_6^0	⋮	p_7^0	⋮	p_8^0	p_9^0
	$p^1 = p_4^1 \quad p_1^1$	⋮	p_2^1	p_3^1	p_1^1	p_5^1	⋮	p_6^1	⋮	p_7^1	⋮	p_8^1	p_9^1
子代	$C^0 = p_8^0 \quad p_9^0$	⋮	p_2^1	p_3^1	p_4^1	p_5^1	⋮	p_6^0	⋮	p_7^1	⋮	p_8^0	p_9^0
	$C^1 = p_6^1 \quad p_1^1$	⋮	p_2^0	p_3^0	p_4^0	p_5^0	⋮	p_6^1	⋮	p_7^0	⋮	p_8^1	p_9^1

k 点交换法通常选 $k = 3$ 或 $k = 2$，即成为 3 点交换或 2 点交换法。

k 点交换法的优点是可以使交换点分散在字符串的多个位置上，促进有益字符的交换，避免交换的字符串保持原样。以表 2-12 为例，若有益字符在 $p_5 \sim p_9$ 区段，如果采用一点交换且交换点在 p_4 之后，则交换后的子代个体的有益字符仍保持不变，而多点交换则改变这些有益字符的分布。

(3) k 点杂乱交换(Shuffle Crossover)。这种方法是在 k 点交换的基础上，将原有字符顺序打乱，以便增加各字符相互交换的概率。其方法步骤如下：

1)设有父代个体 p^0 及 p^1，将各字符按顺序编号，得：

$$p^0 = p_0^0 \quad p_1^0 \quad p_2^0 \quad p_3^0 \quad p_4^0 \quad p_5^0 \quad p_6^0 \quad p_7^0 \quad p_8^0 \quad p_9^0$$

$$p^1 = p_0^1 \quad p_1^1 \quad p_2^1 \quad p_3^1 \quad p_4^1 \quad p_5^1 \quad p_6^1 \quad p_7^1 \quad p_8^1 \quad p_9^1$$

2)用随机的方法打乱 p^0 及 p^1 的字符排列顺序，使第 i 个字符位于第 j 位，但是两个父代个体的 i、j 要相同，如：

$$p^0 = p_2^0 \quad p_5^0 \quad p_6^0 \quad p_9^0 \quad p_0^0 \quad p_7^0 \quad p_8^0 \quad p_4^0 \quad p_3^0 \quad p_1^0$$

$$p^1 = p_2^1 \quad p_5^1 \quad p_6^1 \quad p_9^1 \quad p_0^1 \quad p_7^1 \quad p_8^1 \quad p_4^1 \quad p_3^1 \quad p_1^1$$

3)用随机的方法产生 k 个交换点，并进行 k 点交换，如表 2-13 第 1～4 行所示的 3 点交换。

4)按原来父代个体字符顺序，恢复子代新个体的字符顺序，如表 2-13 第 5～6 行。

杂乱交换的优点是增加各字符(基因)被交换的可能性，以便后代具有更多的有益基因。

(4) 均匀交换。这种方法是对父代个体的每个字符都要进行是否交换的选择。如表 2-14，有父代个体 p^0 及 p^1，各有 10 个字符。对每个字符产生一个 0 或 1 的随机数，0 表示不变动，1 表示交换。于是，产生新个体 C^0 及 C^1。

表 2-13　3 点杂乱交换

序号	交换号		1	2	3
1	父代	$p^0 = p_2^0 \quad p_3^0$	$p_4^0 \quad p_5^0 \quad p_6^0 \quad p_7^0$	p_8^0	$p_9^0 \quad p_{10}^0 \quad p_{11}^0$
2		$p^1 = p_2^1 \quad p_3^1$	$p_4^1 \quad p_5^1 \quad p_6^1 \quad p_7^1$	p_8^1	$p_9^1 \quad p_{10}^1 \quad p_{11}^1$
3	子代	$C^0 = p_2^0 \quad p_3^1$	$p_4^1 \quad p_5^1 \quad p_6^1 \quad p_7^1$	p_8^0	$p_9^1 \quad p_{10}^1 \quad p_{11}^1$
4		$C^1 = p_2^1 \quad p_3^0$	$p_4^0 \quad p_5^0 \quad p_6^0 \quad p_7^0$	p_8^1	$p_9^0 \quad p_{10}^0 \quad p_{11}^0$
5	子代	$C' = p_0^0 \quad p_1^1 \quad p_2^0 \quad p_3^1 \quad p_4^1 \quad p_5^0 \quad p_6^1 \quad p_7^1 \quad p_8^0 \quad p_9^1$			
6		$C'' = p_0^1 \quad p_1^0 \quad p_2^1 \quad p_3^0 \quad p_4^0 \quad p_5^1 \quad p_6^0 \quad p_7^0 \quad p_8^1 \quad p_9^0$			

表 2-14　均匀交换

父　代	$p^0 = p_0^0$	p_1^1	p_2^0	p_3^0	p_4^1	p_5^0	p_6^0	p_7^1	p_8^0	p_9^1
	$p^1 = p_0^1$	p_1^0	p_2^1	p_3^1	p_4^0	p_5^1	p_6^1	p_7^0	p_8^1	p_9^0
随机数	0	1	1	0	0	0	1	0	1	1
子　代	$C^0 = p_0^0$	p_1^0	p_2^1	p_3^0	p_4^0	p_5^0	p_6^1	p_7^0	p_8^1	p_9^1
	$C^1 = p_0^1$	p_1^1	p_2^0	p_3^1	p_4^1	p_5^1	p_6^0	p_7^1	p_8^0	p_9^0

从概率的角度分析，均匀交换可使一半的字符产生交换。

2.4.5　突变

遗传算法产生新个体的方法有二：一为交换，另一为突变。在早期，人们对突变不够重视，突变概率 p_m 取得很小，约 0.005。实践表明，尽管在产生新个体方面突变不及交换重要，但它也是一个有力的手段，特别是加大选择力度使群体素质普遍提高后，突变的作用更加显著。

除了在 2.1.5 所述的传统突变方法，人们在突变方面还作了许多深入研究。

(1) 突变概率取决于字符串长度及群体规模。字符串长度 L 增大后，左侧字符所代表的数值远比右侧大，而字符串中每个字符产生突变的概率相同，一旦突变发生在左侧字符，会使新个体偏离旧个体太远；特别是在遗传算法的后期，会使结果无法收敛。因此，若字符串长度大时，突变概率 p_m 宜取小值。

突变概率还与群体规模 M 有关。当群体所含个体数目较多时，个体间的适应度差别会增大。为了使群体性态不致于过度离散，也希望突变概率随群体规模的增大而减小。

针对上述两种因素的考虑，可参照下式酌情选取突变概率 p_m：

$$p_m = \frac{1.75}{M \cdot \sqrt{L}} \tag{2-40}$$

式中 M —— 群体中拥有的个体数目；

L ——个体中字符串的长度。

(2)突变概率随时间而变化。尽管交换和突变都能产生新个体，然而前者的新个体是原有字符的重新组合，后者则添加崭新的字符。特别是临近遗传算法的后期，突变会使结果无法收敛。因此，在遗传算法的各个阶段，突变概率应该发生变动，早期的突变概率可大，后期的突变概率宜小。具体数值可参照下式选取：

$$p_m(t) = \sqrt{\frac{C_1}{C_2}} \frac{\exp(-C_3 \cdot t/2)}{M \cdot \sqrt{L}} \tag{2-41}$$

式中 $p_m(t)$ —— 第 t 遗传代次时的突变概率；

M —— 群体规模；

L —— 个体字符串长度；

C_1、C_2、C_3 —— 系数。

(3)突变概率是一个敏感参数。由于突变会产生新个体，而新个体的性态可能会变得更好，也会变得更坏，因此突变的效果很难预料。遗过多次试验，人们发现突变概率对计算结果影响很大。图 2-14 描述突变概率对平均最优适应度的影响。从图中可以看出，若突变概率变动0.05，其收敛效果会有很大差别。

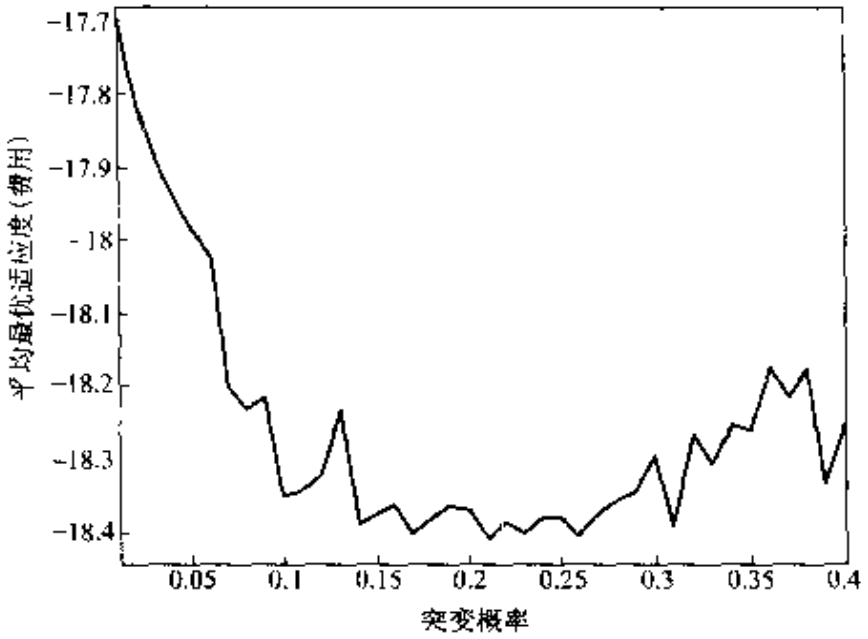

图 2-14　突变概率的影响

2.4.6　群体规模

群体规模 M 是遗传算法的一个重要参数，群体所拥有的个体数目少意味着每次搜索范围狭窄，会延长搜索时间。有人对群体数目作过试验，试验中针对初始群体的产生，采用下述四种方法：

(1) 均匀分布地产生初始个体(搜索点)；

(2) 随机分布地产生初始个体(搜索点)；

(3)50% 用均匀分布，50% 用随机分布产生搜索点；

(4)50% 用随机分布，50% 用补运算产生搜索点。例如，假设随机生成下述两个个体：

个体 1：　0 1 1 1 1 0 0 0 0

个体 2：　0 0 1 0 1 0 1 1 0

那么补运算产生的个体为：

个体 3：　1 0 0 0 0 1 1 1 1

个体 4：　1 1 0 1 0 1 0 0 1

图 2-15 表示 16 个个体的分布情况。可以看出，均匀分布产生的个体散布在整个搜索空间，第 4 种方法的效果也不错。图 2-16 表示 100 个个体时的分布情况。从图中可以看出，当个体数目足够大

时，上述四种方法差别不大，都能散布在整个搜索空间内。

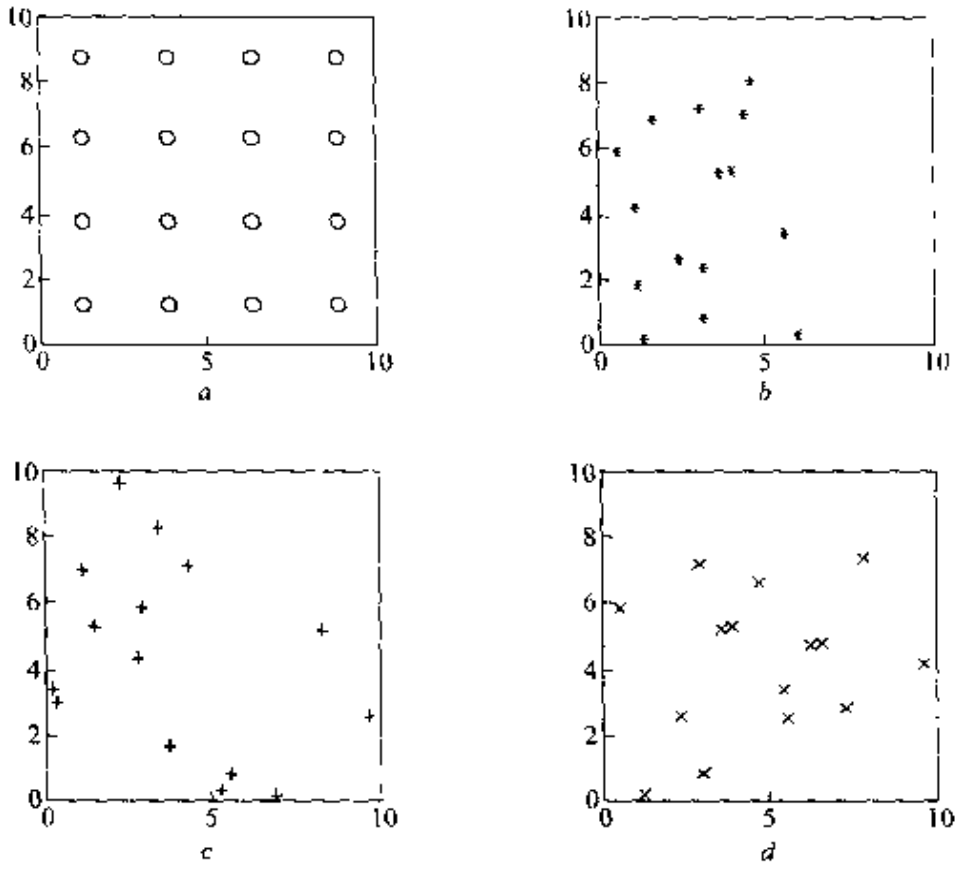

图 2-15　16 个个体的散布图

a--均匀分布；*b*—随机分布；*c*—半均匀、半随机分布；

d- 半随机、半补运算分布

然而，群体数目过大，不仅增加遗传运算的时间，而且会使群体形态过于分散，从而使群体的收敛困难。有人作过试验，试验中令群体数目 M 与最大迭代次数 N 的乘积为常数，6 次试验的 M 与 N 值如表 2-15 所示。试验中的目标函数为：

$$\min f(x,y) = x\sin(4x) + 1.1y\sin(2y)$$

图 2-17 表示最终试验结果。从图中可以看出，这时宜采用试验方案 2，即用 8 个个体迭代 80 次。

在群体规模这个参数上，还可以仿效自然界采用可变的群体规模。通常，遗传算法的群体规模是固定不变的，但也可以按下式方法改用可变的群体规模，即：

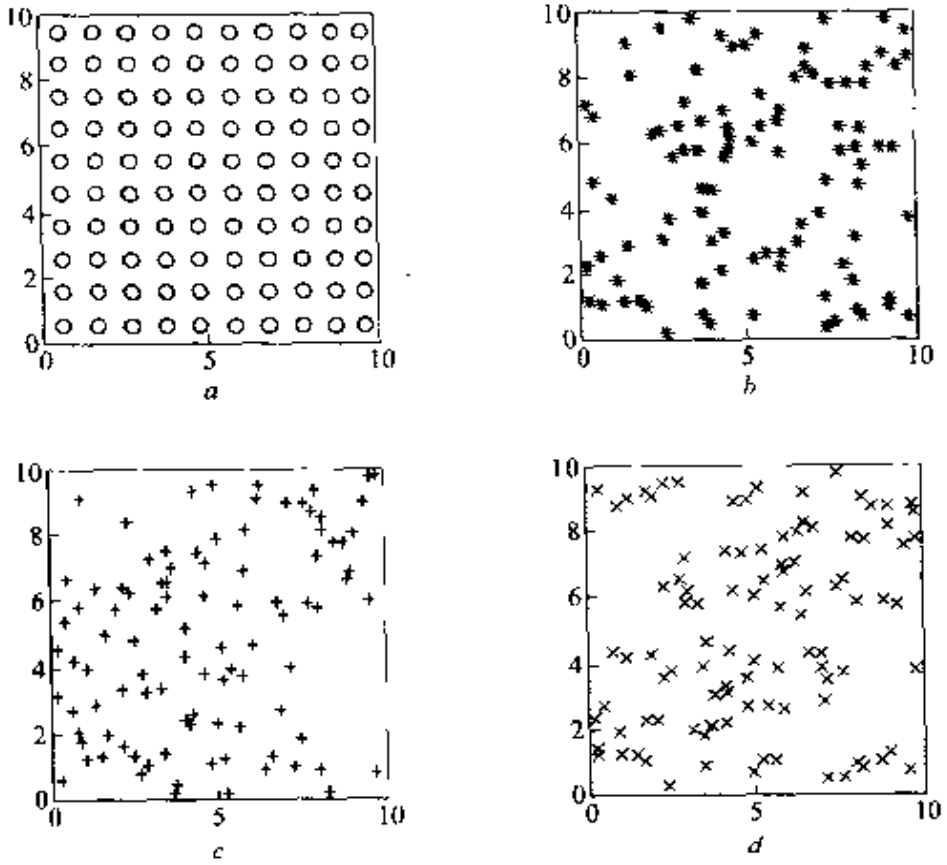

图 2-16　100 个个体的散布图

a—均匀分布；*b*—随机分布；*c*—半均匀、半随机分布；

d—半随机、半补运算分布

表 2-15　试验方案

试验方案	1	2	3	4	5	6
迭代次数 N	160	80	40	20	10	5
群体规模 M	4	8	16	32	64	128

$$M(t+1) = M(t) + A(t) - D(t) \tag{2-42}$$

式中　$M(t)$　— 第 t 遗传代次的群体规模；

$A(t)$　—— 第 t 遗传代次时增加的个体数目；

$D(t)$　—— 第 t 遗传代次时死亡的个体数目。

$A(t)$ 可按下式计算：

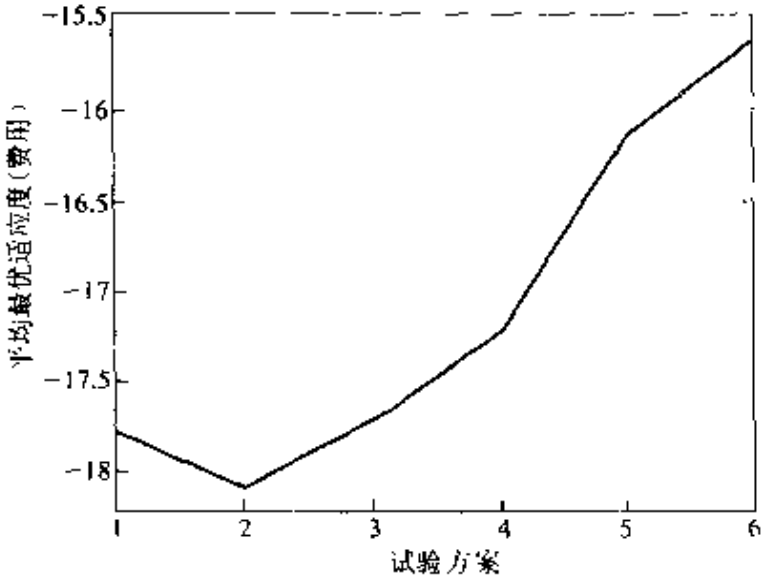

图 2-17　群体数目的试验

$$A(t) = M(t) \cdot p_r$$

式中　p_r —— 复制概率。

也就是说，$A(t)$ 都是复制产生的优良个体，$D(t)$ 则根据个体的寿命决定。在这种方法中，一旦新个体产生，便赋予它一个新参数 —— 个体寿命 Life(i)，该参数取决于个体适应度，优质个体的寿命可长，即：

$$\text{Life}(i) = \begin{cases} L_{\min} + \eta \dfrac{f_i - f_{\min}}{f_{\text{avg}} - f_{\min}} & \text{若 } f_{\text{avg}} \geqslant f_i \\ \dfrac{1}{2}(L_{\min} + L_{\max}) + \eta \dfrac{f_i - f_{\text{avg}}}{f_{\max} - f_{\text{avg}}} & \text{若 } f_{\text{avg}} < f_i \end{cases} \tag{2-43}$$

式中　$L_{\min}$，$L_{\max}$ —— 分别为最小及最大的允许寿命；

f_i —— 个体 i 的适应度；

$f_{\min}$，$f_{\max}$，f_{avg} —— 分别为群体中最小、最大及平均的适应度。

$$\eta = \frac{1}{2}(L_{\max} - L_{\min})$$

与此同时，还统计个体的年龄。个体每经历一个遗传代次，其年龄增加 1。一旦年龄等于寿命，该个体便死亡。

图 2-18 是这种新方法的执行结果。从图中可以看出，群体规模从初始时的 20 个增至 340 个，然后又减少至 20～50 个。

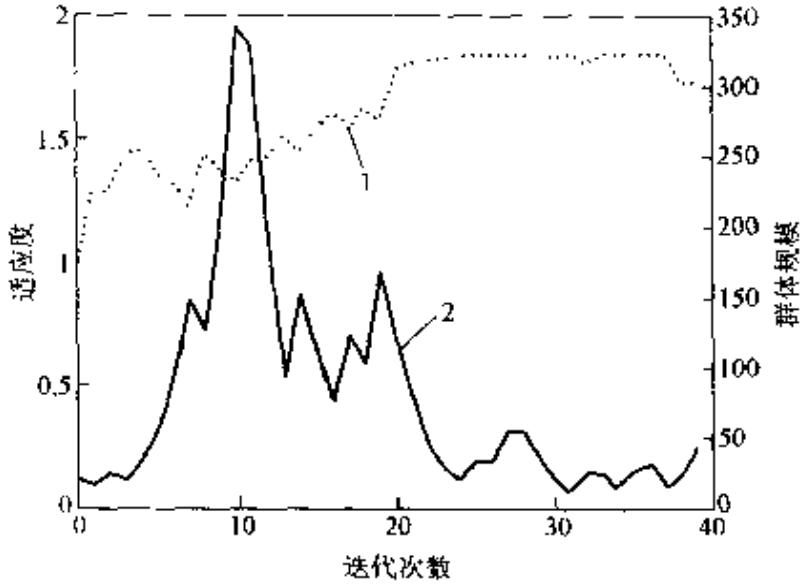

图 2-18　可变群体规模

1－适应度；2－群体规模

表 2-16 是群体规模可变与不变时的效果比较。其中 $G1$、$G2$、$G3$、$G4$ 代表下述四种目标函数(求最大值)：

$G1$：　$-x\sin(10\pi x)+1$　　$-2.0 \leqslant x \leqslant 1.0$

$G2$：　$\text{integer}(8x)/8$　　$0 \leqslant x \leqslant 1.0$

$G3$：　$x\text{sgn}(x)$　　$-1.0 \leqslant x \leqslant 2.0$

$$G4：\quad 0.5+\frac{\sin^2\sqrt{x^2+y^2}-0.5}{(1+0.001(x^2+y^2))^2} \qquad -100 \leqslant x, y \leqslant 100$$

表中衡量算法效果的指标有二：一为最优解(目标函数值)V，另一为运算时间 E，用目标函数的计算次数表示。从表中可以看出，两种方法的最优解基本相同，但新方法计算时间似乎较长。不过，表中旧方法的群体规模已选定为最优规模，从 $G1$ 至 $G4$ 依次为 75、15、75 及 100。例如，若 $G2$ 的规模改为 75，则运算时间增至 1035。

表 2-16　两种群体规模的比较

群体规模	G1		G2		G3		G4	
	V	E	V	E	V	E	V	E
固定不变	2.814	1467	0.875	345	1.996	1420	0.959	2186
可　变	2.813	1538	0.875	670	1.999	1555	0.972	2106

2.4.7　次要算子

遗传算法的主要遗传操作是复制、交换及突变，它们又称为主要算子。此外，人们仿照生物进化及遗传的机理，又提出一系列新的算子，如倒序算子(Inversion Operator)、生态算子(Niche Operator)、显性算子(Dominance Operator)、多重字符串结构(Multiple Chromosome Structure)、加倍(Duplication)与删除(Deletion)、性别区分(Sexual Differentiation)等。不过，这些次要算子使用较少，这里仅介绍使用较多的倒序算子。

倒序算子是使字符串中某一部分字符的顺序倒转。例如，在下述 8 位二进制字符串中：

$$1\ 0\ 1\ 1\ 0\ 0\ 1\ 1$$

随机选中第三及第六个字符(用下横线表示)，在这区间实施倒序操作，其结果如下：

$$1\ 0\ \underline{0}\ 0\ 1\ \underline{1}\ 1\ 1$$

倒序算子可使字符串产生急剧的变化，对于长度特别大的字符串会有积极的意义，通常的交换或突变不易取得这种效果。

倒序算子不只是简单地随意变动字符，而且还可以有目的地改进模式结构。为此，它常采用扩展形式表示，把字符(基因)的位置和字符本身分别用两行表达。例如，下述扩展形式中：

$$\begin{cases} 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8 \\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1 \end{cases}$$

第一行是字符的位置，第二行是字符本身。若上式在 3～6 之间进行倒序操作，则有：

$$\begin{cases} 1\ 2\ 6\ 5\ 4\ 3\ 7\ 8 \\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1 \end{cases}$$

这样，字符串中的字符总是和它们的位置连系在一起，即第 6 位字符总是 0，而第 4 位总是 1。因此，单纯的倒序操作不会改变该字符串的适应度，这正是扩展形式的优点。

扩展形式的另一特点是模式结构的调整。例如，下述字符串在第 4、3 位倒序，则有：

$$\begin{cases} 1\ 4\ 3\ 5\ 2\ 6\ 7\ 8\ 9\ 10 \\ *\ *\ 0\ *\ 0\ 1\ *\ *\ *\ * \end{cases} \longrightarrow \begin{cases} 1\ 3\ 4\ 5\ 2\ 6\ 7\ 8\ 9\ 10 \\ *\ 0\ *\ *\ 0\ 1\ *\ *\ *\ * \end{cases}$$

其模式长度由 3 增至 4。相反，若倒序操作发生在模式之内时，在扩展形式意义下模式没有被破坏。例如，上式改在第 5、2 位倒序，则：

$$\begin{cases} 1\ 4\ 3\ 5\ 2\ 6\ 7\ 8\ 9\ 10 \\ *\ *\ 0\ *\ 0\ 1\ *\ *\ *\ * \end{cases} \longrightarrow \begin{cases} 1\ 4\ 3\ 2\ 5\ 6\ 7\ 8\ 9\ 10 \\ *\ *\ 0\ 0\ *\ 1\ *\ *\ *\ * \end{cases}$$

其模式的阶次及长度都没有变化。

从数学上可以证明，字符串第 k 位的字符在倒序操作中被移动的概率为：

$$p(k) = \frac{2[k(L+1) - k^2 + 1]}{L^2}$$

式中 L — 字符串长度。

当字符串很长时，有：

$$p(k) = 2(X - X^2)$$

式中 $X = k/L$

根据一阶导数求极值，p 的最大值发生在 $X = 0.5$ 时，这说明在倒序操作中，位于字符串中间部分的字符被移动的可能性最大，两端处相对较小。因此，对于有用的基因组（倒如上例中第 3、5、2、6 位字符 0 * 01），应利用倒序操作迁移至中间，以便重新组合，形成有意义的基因组，即所谓构造块。

用扩展形式表示字符串会对交换操作带来困难。例如，下述 A、B 两字符串在左数第 4 个数字之后发生交换为：

$$A=\begin{pmatrix}1\ 2\ 3\ 4 & 5\ 6\ 7\ 8\\ 1\ 0\ 1\ 1 & 1\ 0\ 1\ 1\end{pmatrix} \quad \xrightarrow{\text{交换}} \quad A=\begin{pmatrix}1\ 2\ 3\ 4 & 4\ 3\ 7\ 8\\ 1\ 0\ 1\ 1 & 1\ 1\ 1\ 1\end{pmatrix}$$

$$B=\begin{pmatrix}1\ 2\ 5\ 6 & 4\ 3\ 7\ 8\\ 1\ 0\ 1\ 1 & 1\ 1\ 1\ 1\end{pmatrix} \qquad B=\begin{pmatrix}1\ 2\ 5\ 6 & 5\ 6\ 7\ 8\\ 1\ 0\ 1\ 1 & 1\ 0\ 1\ 1\end{pmatrix}$$

交换后 A 缺少第5、6位字符，B 缺少第3、4位。为此，可以选用下述交换方式：

(1) 对等交换法。只有当两个字符串的位置一一对应时才能交换，否则不允许交换。例如：

$$A=\begin{pmatrix}1\ 4\ 2\ 5\ 3\\ 1\ 1\ 0\ 0\ 1\end{pmatrix} \quad B=\begin{pmatrix}1\ 4\ 2\ 5\ 3\\ 0\ 0\ 1\ 1\ 0\end{pmatrix} \quad C=\begin{pmatrix}1\ 2\ 3\ 4\ 5\\ 1\ 0\ 0\ 1\ 1\end{pmatrix}$$

A 和 B 允许交换，而 A 和 C、B 和 C 则不允许。

(2) 生死交换法。允许任意两个字符串交换，但当交换结果所包含的位置符不完整时，取消其生存权，作为“死胎”处理。例如，上例中的 A、C 交换结果为死胎。

(3) 主从交换法。任意两个字符串交换时，以其中一个字符串为主序，另一个字符串按主串要求调整成对等关系，然后再进行交换。例如，上述 A、B、C 三个字符串中，假设 A 的适应度最高，被选作主串，则将 C 串倒序为：

$$C=\begin{pmatrix}1\ 4\ 2\ 5\ 3\\ 1\ 1\ 0\ 1\ 0\end{pmatrix}$$

然后才允许与 A 交换。

针对扩展形式上的交换，人们还提出其他一些算子，如 PMX 算子(Partially Matched Operator)、OX 算子(Order Crossover Operator)、CX 算子(Cycle Crossover Operator)，限于篇幅，这里不作介绍。

2.4.8 连续型遗传算法

前面讨论的遗传算法，都是用二进制字符串表达问题。当目标变量的有效数字很多时(如 $x=1469.72315$)，二进制编码的字符串长度很大，使得复制、交换、突变等操作占用计算机时间很长。相比之下，若改用十进制编码，不仅无需转换数制便于理解，而且节

省遗传操作的时间。人们针对二次型最优控制问题，分别用二进制码和十进制码进行遗传算法。实验结果如表 2-17 所示。从表中可以看出，十进制码的运算时间仅为二进制的 10%～25%。因此，十进制编码受到人们重视，被称作连续型遗传算法(Continuous Genetic Algorithms)，以区别于前述的二进制遗传算法。

表 2-17　十进制编码与二进制编码的 CPU 时间(s)

二进制编码	二进制编码的字符串长度					
	5	10	20	30	40	50
	4426	5355	7438	9219	10981	12734
十进制编码	1072					

连续型遗传算法的工作过程与二进制时相同，所不同的操作仅为下述 4 种：

(1)编码。连续型遗传算法采用十进制数表达问题，与传统的处理手法相同。例如，有下述优化问题：

$$\max \quad f(x,y) = x \cdot \sin(4x) + 1.1y \cdot \sin(2y)$$

$$\text{s.t.:} \quad 0 \leqslant x \leqslant 10$$

$$0 \leqslant y \leqslant 10$$

在连续型遗传算法中，每个个体由两部分组成，即：

$$\text{个体：}[x,y]$$

其中 x,y 用十进制数表示。今后处理时，x 和 y 这两个十进制数将作为两个独立的基因分别对待。

(2)产生初始群体。同样采用随机生成的方法建立初始群体。个体中每个十进制数用下述式子计算：

$$X = (X_{\max} - X_{\min}) \cdot \text{Rand} + X_{\min} \tag{2-44}$$

式中　X ——初始个体的一个目标变量；

$X_{\max}$、$X_{\min}$ ——目标变量允许的最大值及最小值。本例中为 10 及 0；

Rand ——在[0,1]区间内的随机数。

例如，在 $f(x,y)$ 问题中可产生(0.9073，0.6684)、(2.6932，7.6649)、(9.1382，5.2693)、(7.6151，9.1032)…… 等初始个体。

(3) 交换。在连续型遗传算法中，每个十进制数被视作一个基因整体，因此若采用二进制的交换方法，所产生的新个体与父代差别不大。例如将 (x_1,y_1) 及 (x_2,y_2) 进行交换，新个体仅为 (x_2,y_1) 和 (x_1,y_2)，为此需要采用新的交换方法使新个体有更大的差异。

最基本的方法是混合交换(Blending Crossover)。当随机确定交换点后，将交换点以后的每对父代个体的十进数按下式产生新基因：

$$p_i = \beta \cdot p_{i1} + (1 - \beta) p_{i2} \tag{2-45}$$

式中 p_i — 新个体的第 i 个十进数；

p_{i1}、p_{i2} — 父代两个个体的第 i 个十进数；

β —[0,1] 区间的随机数。

很明显，假如 $\beta = 0$ 或 $\beta = 1$，则新基因 p_i 等同于旧基因 p_{i2} 或 p_{i1}。当 $\beta = 0.5$，则新基因是旧基因的平均值。

第二种方法是启发式交换(Heuristic Crossover)。这种方法采用线性插值的方法产生新基因：

$$p_i = \beta(p_{i1} - p_{i2}) + p_{i1} \tag{2-46}$$

上述两种方法每次只产生一个新基因。为了成双地产生两个子代个体以便替换父代两个旧个体，需要重复执行两次计算，利用随机数 β 的差异形成两个新基因。

第三种方法称综合交换。它综合应用过去的交换方法。首先，随机确定交换点为 α，则父代个体写作：

$$\text{父代个体 1：}\ [p_{11}\ p_{12}\ \cdots\ p_{1\alpha}\ \cdots\ p_{1m}]$$

$$\text{父代个体 2：}\ [p_{21}\ p_{22}\ \cdots\ p_{2\alpha}\ \cdots\ p_{2m}]$$

其次，按下式计算交换点的新基因 p_1' 及 p_2'：

$$\begin{cases} p_1' = p_{1\alpha} - \beta(p_{1\alpha} - p_{2\alpha}) \\ p_2' = p_{2\alpha} + \beta(p_{1\alpha} - p_{2\alpha}) \end{cases} \tag{2-47}$$

随后，对交换点以后父代个体基因进行交换，从而得出两个新个体为：

子代新个体 1：$[p_{11}\ p_{12}\cdots\ p_i\cdots p_{2m}]$

子代新个体 2：$[p_{21}\ p_{22}\cdots\ p'_i\cdots p_{2m}]$

例如，在前述 $f(x,y)$ 问题中有两个父代个体：[5.2693，9.1382] 及[9.1032，7.6151]，假设 β 随机选定为 0.7147，交换点在第一个数，则新个体为：

新个体 1：$[5.2693-0.7147\times(5.2693-9.1032),7.6151]$
$=[8.0094,7.6151]$

新个体 2：$[9.1032+0.7147\times(5.2693-9.1032),9.1382]$
$=[6.3631,9.1382]$

(4) 突变。在连续遗传算法中，每个十进数(基因)的突变是在该数允许范围内变动，可按式(2-44)计算。

另一种突变方法称非均匀突变。假设第 t 代的个体为 $p^t=(p_1\cdots p_i\cdots p_m)$，若 p_i 被随机选中为突变点，则新个体为 $p^{t+1}=(p_1\cdots p'_i\cdots p_m)$，其中 p'_i 按下式计算：

$$p'_i=\begin{cases}p_i+\Delta(t,y) & \text{若随机数 } R \text{ 为 } 0\\ p_i-\Delta(t,y) & \text{若随机数 } R \text{ 为 } 1\end{cases}$$

式中 R——仅在 0 与 1 两者选择的随机数；

$$y=p_{max}-p_{min}$$

p_{max}，p_{min}——p_i 允许的最大值及最小值；

$$\Delta(t,y)=y\cdot(1-r^{(1-\frac{t}{T})^b})$$

r——[0,1] 区间内的随机数；

T——最大允许迭代次数；

b——系数；

t——当前的迭代次数。

$\Delta(t,y)$ 的目的是使迭代后期当 $t\to T$ 时突变愈来愈小，p'_i 值仅在原地附近变动；而在初期 $\Delta(t,y)$ 较大，p'_i 值在大范围内搜索。图 2-19 描述 t/T 分别为 0.5 及 0.9 且 $b=2$ 时 $\Delta(t,y)$ 的变化，可明显看出前者大于后者。

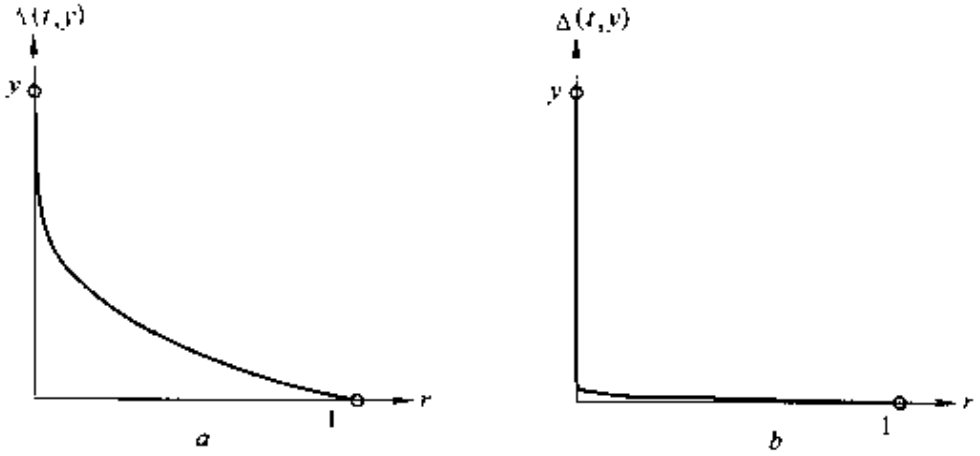

图 2-19　不同时期下 $\Delta(t,y)$ 的比较

a　$t/T=0.50$，b＝2；b　$t/T=0.90$，b＝2

3 遗传规划

3.1 遗传规划的原理

3.1.1 遗传算法的局限性

遗传算法是对字符串进行操作，逐步实现进化遗传，可以解决许多复杂的问题。但是，由于遗传算法常用定长的字符串表达问题，限制了它的应用范围。遗传算法的主要缺点是：

(1)不能描述层次化的问题。有许多问题，其描述往往是一种层次化的结构，而不是定长的字符串形式。例如，多项式 $f(x) = A_0 + A_1x + A_2x^2 + A_3x^3$，很难用定长的字符串表达，它实质上可用图 3-1 的层次化结构表达。图中从左到右、从下而上地读取各节点，可得 $f(x)$ 的表达式。例如，从最左边读起，有 $A_0 + (A_1 * x)$

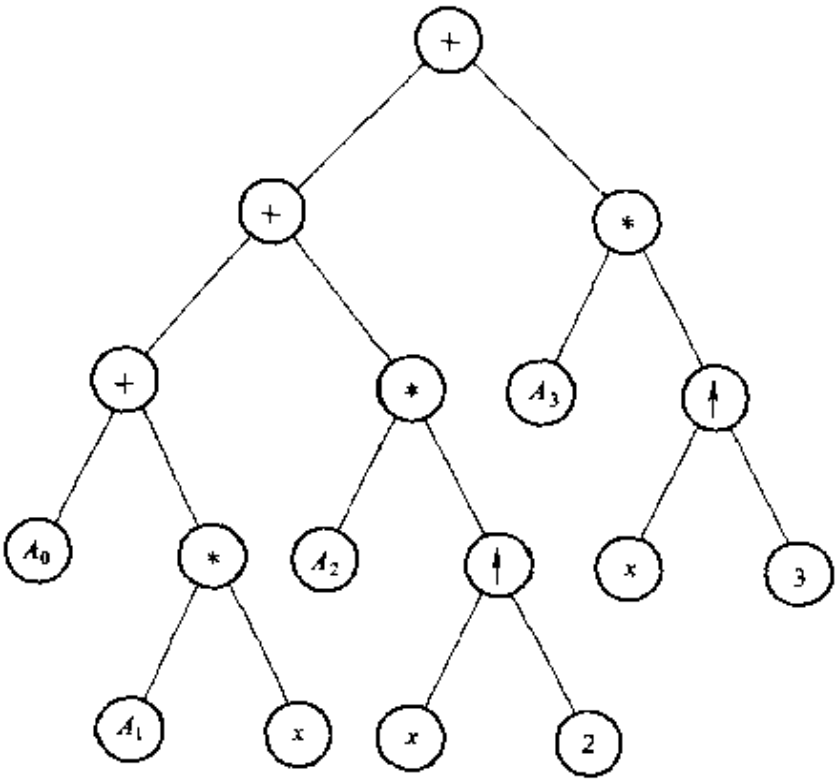

图 3-1 $f(x)$ 的层次化结构

$+ (A_2 * (x \uparrow 2)) + \cdots\cdots$

(2)不能描述计算机程序。在人工智能领域中,计算机自动编程--直是个热门课题。遗传算法通过进化和遗传,只能改变字符串的形式,不能形成层次结构的计算机程序。为此,需要改变问题的表达方法,以便实现计算机自动编程。

(3)缺乏动态可变性。遗传算法的定长字符串,不具备动态可变性。字符串长度一旦确定,就很难动态地表达状态或行为的变化,限制了问题的表述。

由此可见,遗传算法这种字符串表达形式,限制了对问题的结构和大小的灵活处理。因此,迫使人们寻求新的表达方法。1992 年,美国John R. Koza 正式提出遗传规划(Genetic Programming),用层次化的结构性语言表达问题。

3.1.2 遗传规划的工作原理

遗传规划的最大特点,是采用层次化的结构表达问题,它类似于计算机程序分行或分段地描述问题。这种广义的计算机程序能够根据环境状态自动改变程序的结构及大小。现以曲线拟合问题为例进行说明。

表 3-1 列举一组实测数据 x_i、$y_i(i = 1 \sim 17)$,现在要求出它们之间的函数关系 $y = f(x)$。

表 3-1 一组实测数据

序号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
x	97	47	95	84	96	113	24	16	47	37	42	14	23	35	27	85	96
y	34	17	28	25	31	52	8	6	14	9	13	3	14	10	9	36	48

通常,进行曲线拟合时首先要决定函数 $f(x)$ 的结构形式,如:

直线型　　$y = a + bx$

多项式　　$y = a + bx + cx^2 + dx^3$

对数型　　$y = a + \log x$

指数型　　$y = a + b^x$

……

然后再用最小二乘法求出参数 a、b、c、d……等。若采用遗传规划，可以将函数 $f(x)$ 的结构和参数综合在一起求解。这正是遗传规划的特点和优点。

3.1.2.1　个体的描述

类似于遗传算法，在遗传规划中每一代包含众多的个体，组成这一代的群体。然而，遗传规划中的个体用广义的层状计算机程序表达，它由函数集 F 及终止符集 T 组成。

函数集 F 包含 n 个函数：

$$F = \{f_1, f_2, \cdots\cdots, f_n\}$$

函数集内的函数 f_i 可以是＋、－、＊、／等算术运算符或 sin、cos、log、exp 等标准数学函数。

终止符集 T 包含 m 个终止符：

$$T = \{t_1, t_2, \cdots, t_m\}$$

终止符集内的终止符可以是 x，y，z 等变量或 a、b、π 等常量。

将函数 f_i 和终止符 t_j 组合，可得出类似于下述式子的各种表达式：

(1) $y_1 = a + b * x$

(2) $y_2 = a * \exp(b * x)$

(3) $y_3 = a + b * \log(x)$

(4) $y_4 = a * x^b = a * x \uparrow b$

这 4 种函数的层次化描述如图 3-2 所示，它们以树状结构的形式表达问题。

3.1.2.2　初始群体的形成

遗传规划的初始群体用随机的方法产生，从函数集 F 及终止符集 T 中随机选取 f_i 及 t_j，组成各种复杂的数学函数。其中，常数 a、b 可在允许范围内随机取值。

假设初始群体如下：

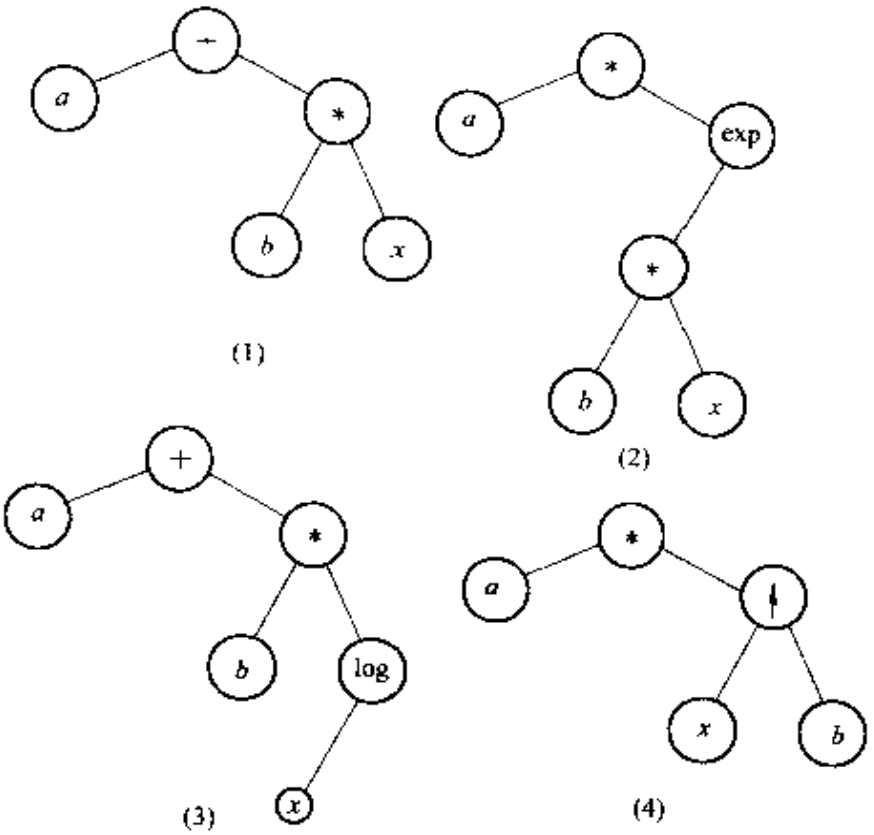

图 3-2　树状结构

（第 0 代群体）
$$\begin{cases} \text{个体 1：} & -4.3+1.21*x \\ \text{个体 2：} & 0.667*\exp(0.071*x) \\ \text{个体 3：} & -12.72+1.77*\log(x) \\ \text{个体 4：} & 1.242*x\uparrow 0.76 \end{cases}$$

3.1.2.3　个体适应度测定

个体的适应度是衡量个体优劣的主要尺度。在本例中，适应度也就是个体表达式逼近真实解的近似程度。表 3-2 列举初始个体计算值与实测值的比较，采用误差绝对值总和作为个体的适应度。

从表中可以看出，个体 2 的误差最大，其性能最差；个体 4 的误差最小，其性能最佳。第 0 代群体的平均适应度为 1613。

表 3-2　各个体计算值与实测值的比较

序号	自变量 x	实测值 y	计算返回值 $\bar{y}$			
			个体 1	个体 2	个体 3	个体 4
1	97	34	113.070	653.299	−4.634	40.185
2	47	17	52.570	18.766	−5.915	23.169
3	95	28	110.650	566.816	−4.671	39.554
4	84	25	97.340	259.572	−4.888	36.022
5	96	31	111.860	608.523	−4.652	39.870
6	113	32	132.430	2034.560	−4.364	45.130
7	24	8	24.740	3.666	−7.103	13.902
8	16	6	15.060	2.077	−7.819	10.215
9	47	14	52.570	18.766	−5.915	23.169
10	37	9	40.470	9.226	−6.337	19.318
11	42	13	46.520	13.158	−6.113	21.271
12	14	3	12.640	1.802	−8.055	9.230
13	23	14	23.530	3.415	−7.178	13.460
14	35	10	38.050	8.005	−6.436	18.519
15	27	9	28.370	4.536	−6.894	15.204
16	85	36	98.550	278.672	−4.867	36.348
17	96	48	111.860	608.523	−4.652	39.870
误差绝对值总和			753.280	5123.380	457.493	118.518

3.1.2.4　复制

类似于遗传算法，达尔文的“优胜劣汰、适者生存”的自然法则也用于遗传规划中。每一代群体的优良个体被复制进入下一代群体，而劣质个体被淘汰。为了选择复制对象，可采用遗传算法中的比例选择法（轮盘法），使适应度高的优良个体尽量被复制，但也不排除个别劣质个体被破格录用（详见 3.3）。

在本例中，个体 2 被淘汰，个体 4 被复制代替个体 2。于是，复制后的第 1 代群体变为：

$$\text{（复制的第 1 代群体）}\begin{cases}\text{个体 1：} -4.3 + 1.21 * x \\ \text{个体 2：} 1.242 * x \uparrow 0.76 \\ \text{个体 3：} -12.72 + 1.77 * \lg(x) \\ \text{个体 4：} 1.242 * x \uparrow 0.76\end{cases}$$

由于个体 2 变为优质个体，这时群体的平均适应度为 362，比初始群体有很大改善。

3.1.2.5　交换

交换是将两个个体的组分互换，繁衍出两个新的个体。在交换中，首先用轮盘选择法随机选择两个优良个体作为父代个体，然后在两个个体中各随机选取一个交换点，将交换点以后的部分进行交换（详见 3.3）。

在本例中，假设个体 1 和个体 4、个体 2 和个体 3 进行交换，交换点分别用下划线“—”标记，交换后的新群体为：

$$\text{（交换后的第 1 代群体）}\begin{cases}\text{个体 1：} -4.3 + 1.21 * \underline{x \uparrow 0.76} \\ \text{个体 2：} 1.242 * \lg(x) \\ \text{个体 3：} -12.72 + 1.77 * \underline{x \uparrow 0.76} \\ \text{个体 4：} 1.242 * x\end{cases}$$

经过复制和交换，第 1 代群体的计算结果与实测值的比较列于表 3-3。从表中可知，个体 2 经过复制和交换，其误差从第 0 代的 5123.380 减少至 276.225。第 1 代群体的平均适应度（327）也比第 0 代群体（1613）有很大的改善。

3.1.2.6　突变

在遗传规划中，突变的作用远不如在遗传算法中那样重要。不过，近年来突变已日益受到重视。

不断执行上述个体适应度测定 复制 交换 突变等操作，使群体的素质不断改善，最终可逐渐逼近最优解。

表 3-3　第 1 代群体的适应度

序号	自变量 x	实测值 y	计算返回值 y			
			个体 1	个体 2	个体 3	个体 4
1	97	31	31.850	5.682	44.471	120.474
2	47	17	18.273	4.782	20.254	58.374
3	95	28	34.235	5.656	43.572	117.990
4	84	27	30.794	5.503	38.546	104.328
5	96	31	34.543	5.669	44.022	119.232
6	113	52	39.667	5.871	51.307	140.346
7	24	8	9.244	3.947	7.065	29.808
8	16	6	5.652	3.444	1.818	19.872
9	47	11	18.273	4.782	20.254	58.374
10	37	9	14.520	4.485	14.772	45.954
11	42	13	16.423	4.642	17.552	52.164
12	14	3	4.692	3.278	0.415	17.388
13	23	14	8.813	3.894	6.435	28.566
14	35	10	13.742	4.416	13.635	43.470
15	27	9	10.512	4.093	8.918	33.534
16	85	36	31.111	5.518	39.009	105.570
17	96	48	34.543	5.669	44.022	119.232
误差绝对值总和			75.314	276.225	98.908	857.676

总之，遗传规划与遗传算法的工作过程基本类似，都经历初始群体生成、个体适应度计算、复制、交换、突变、反复迭代等过程。它们的差别主要体现在问题的表达上：遗传规划是用广义的层状结构的计算机程序表达问题，在遗传进化过程中个体不断动态变更结构及大小。

3.2　遗传规划的表述

遗传规划的工作步骤可归纳如下：

(1)确定个体的表达方式，包括函数集 F 及终止符集 T。

(2)随机产生初始群体。

(3)计算各个体的适应度。

(4)根据遗传参数，用下述操作产生新个体：

1) 复制。将已有的优良个体复制，加入新群体中，并相应删除劣质个体。

2) 交换。将选出的两个个体进行交换，所产生的两个新个体插入新群体中。

3) 突变。随机改变个体某一部分，将新个体插入新群体中。

(5)反复执行(3)及(4)，直至取得满意结果。

图 3-3 详细描述遗传规划的工作流程。图中 Gen 代表遗传代次。从第 0 代开始，随机产生 M 个初始个体组成初始群体，随之计算各个体的适应度，然后依次执行复制、交换及突变等操作。图中 p_r、p_c 及 p_m 分别代表复制概率、交换概率及突变概率。首先反复执行复制，将优质个体复制后添入新群体中，直至复制个体数目 j 达到复制概率要求，并删除同样数目的劣质个体。其次反复执行交换，每次选择两个父代个体，将交换后的两个子代个体添入新群体中，直至交换个体数目 j 达到交换概率要求。最后执行突变，突变概率是指每个个体发生突变的可能性，突变次数也要满足突变概率所规定的突变个体数。一旦完成复制、交换及突变，说明遗传规划已完成一个代次，将计数器 Gen 加 1。如此反复迭代下去，直至得出满意结果。

遗传规划也可以用形式化语言描述。假设 $a \in I$ 记为个体，I 记为个体空间。适应度函数记为 $\Phi: I \to R$。在第 t 代，群体 $P(t) = \{a_1(t), a_2(t), \cdots, a_n(t)\}$ 经过复制 r、交换 c 及突变 m 转换成新一代群体。这里 r、c、m 均指宏算子，把旧群体转换为群体。令 $l: I^n \to \{\text{True}, \text{False}\}$ 为终止准则。利用上述符号，遗传规划可描述为：

$t = 0$;

initialize $P(0) := \{a_1(0), a_2(0), \cdots, a_n(0)\}$;

while ($l(P(t)) \neq$ True) do

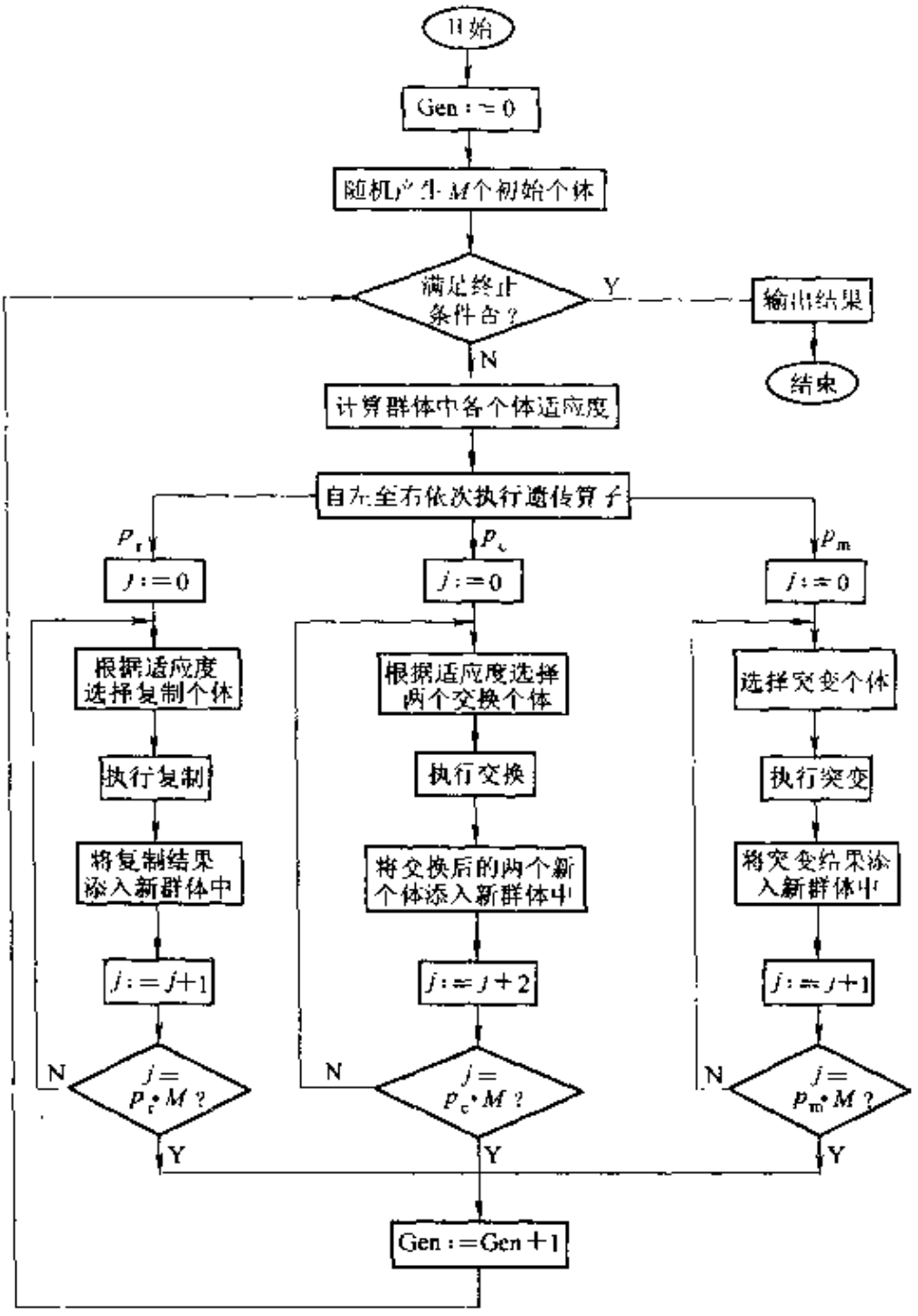

图 3 3　遗传规划流程图

evaluate p(t)：$\{\Phi(a_1(t)), \Phi(a_2(t)), \cdots, \Phi(a_n(t))\}$；

reproduction：P′(t)：=r (P (t))；

crossover: P''(t) := c(P(t));

mutation: P(t+1) := m(P''(t));

t=t+1;

end

3.3 遗传规划的基本技术

遗传规划自90年代初出现以来，有了飞速的发展。据初步统计，从1992～1998年短短的6年期间，有200多人就遗传规划发表了800多篇文章。本节将对这些技术进行综合介绍。

3.3.1 问题的表达

遗传规划是用层次结构可变的形式表达问题。在表达中主要用函数和终止符两类组分。简单地说，终止符表示问题的值，函数表示对值的处理。综合在一起，遗传规划的个体表示对各种值(终止符)的处理过程(函数)。

在函数集 $F=\{f_1, f_2, \cdots, f_n\}$ 中，函数 f_i 可以是运算符、函数、说明等，具体有：

(1) 算术运算符，如+、-、*、/等。其中除号为防止计算机溢出，规定不允许用零作分母，称保护性除法(Protected Division)，用%标记。一旦遇到分母为零时，最简单的处理方法是令其商为1，或是重新选择算术运算符。

(2) 超越函数，如sin、cos、tan、log、exp等。其中log要防止处理小于或等于零的数值，称保护性对数，记为Rlog，其处理方法类似于%。

(3) 布尔运算符，如AND、OR或NOT等。

(4) 条件表达式，如If-then-else、Swith-Case等。

(5) 循环表达式，如Do-until、While-do、For-do等。

(6) 控制转移说明，如Go to、Call、Jump等。

(7) 变量赋值函数，如a:=1、Read、Write等。

(8) 其他定义的函数。

终止符集 $T=\{t_1, t_2, \cdots, t_m\}$ 包括各种常数、输入、变量等，

具体有：

（1）常数，如 π、180° 等。

（2）变量，如 x，y，z 等。

（3）输入，如 a，b，c 等。

顾名思义，终止符是个体表达的终点，从属于函数。

将函数集和终止符集综合在一起，便可形成层次状个体。例如，有下列函数集：

$$F = \{\text{AND}, \text{OR}, \text{NOT}\}$$

和终止符集：

$$T = \{\text{D0}, \text{D1}\}$$

式中 D0、D1——布尔变量。

上述函数集和终止符集的并集 C 为：

$$C = \{\text{AND}, \text{OR}, \text{NOT}, \text{D0}, \text{D1}\}$$

C 中终止符 D0 及 D1 可视为具有 0 个变量的函数。于是并集 C 中各函数的变量个数分别为：2，2，1，0，0。从并集 C 中任选一个函数（假设是 OR），根据该函数的变量数目再从 C 集选取相应数目的函数（OR 要选两个函数）。如此重复，直至选出 0 个变量的函数，它也就是终止符。

为了形象地表达遗传规划的层次结构，通常采用算法树的形式。现用一个有两个变量的奇－偶判断函数为例。若该函数的变量数目为偶数，则该函数返回 T（真）；否则，该函数返回 F（假）。这种布尔型函数的符号表达式为：

(NOT(D0) AND NOT(D1)) OR (D0 AND D1)

图 3-4 描述上述符号表达式的层次化结构树。树中 5 个内结点为函数集 F 中的函数元素（OR、AND、AND、NOT、NOT），4 个外结点（叶子）为终止符集 T 中的布尔变量（D0、D1、D0、D1），根为函数集 F 中的一个函数，即 OR。该树即为数据结构中的算法树。这种算法树常用于表达遗传规划中的个体。

算法树的大小视问题而异。对于复杂的问题可含 256 个结点，其中函数结点约占 56 个，其余 200 个为终止符。

3.3.2 初始群体的生成

3.3.2.1 初始个体生成原理

初始群体由众多个体组成，每个初始个体都是用随机方法产生。

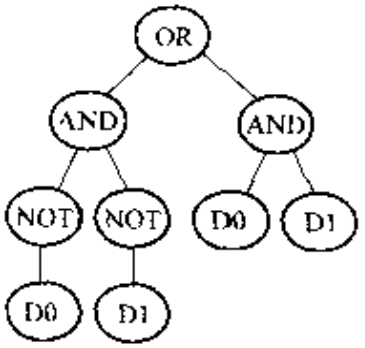

图 3-4 奇-偶判断函数的算法树

图 3-5 根结点为函数“+”的算法树(第一次选择)

生成初始个体时，第一步是从函数集 F 中按均匀分布用随机方法选取一个函数作为算法树的根结点。把根结点的选择限制在函数集 F 中是为了生成一个层次化的复杂结构；相反，若从终止符集 T 中选取根结点，则形成只有一个终止符的退化结构。图 3-5 描述函数“+”为根结点的算法树，函数“+”具有两个变量，可以连结两个函数或终止符。一般情况下，若函数 f 具有 $z(f)$ 个变量，那么从该结点发出 $z(f)$ 条线。

选出根结点后，根据所发出的线数(变量数目)，再从函数集 F 和终止符集 T 的并集 $C = F \cup T$ 中按均匀分布的随机方法选出一个元素作为该条线的尾结点。如果选出的仍是函数，则重复执行上述方程。如图 3-6 所示，若在根结点“+”(图中点 1)之后又从并集 C 中选出函数“*”(图中点 2)，函数“*”作为非根内结点。由于函数“*”具有两个变量，则又发出两条线。

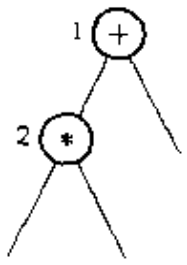

图 3-6 非根内结点为函数“*”的算法树(第二次选择)

在根结点之后从并集 C 随机选择的尾结点，假若为终止符，则该分支上的树就停止生长。例如，图 3-7 中终止符 a、b、c 分别从并集 C 中被选出，作为树中各分支的尾结点，这时算法树生成过程终止。

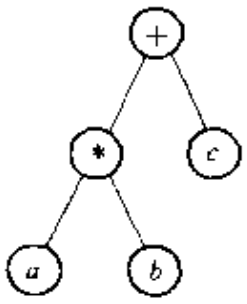

图 3-7　终止符 a、b、c 被选作叶子的算法树(第三～五次选择)

上述过程从上到下、从左到右不断重复，直至形成一棵完整的树为止。

3.3.2.2　初始个体的生成方法

初始个体的生成方法有三种：完全法、生长法和混合法。

(1) 完全法。用完全法产生的初始个体，每一叶子的深度都等于给定的最大深度。所谓叶子的深度，是指叶子距树根的层数。例如，图 3-7 的树深度为 3。

用完全法产生初始个体时，首先从函数集 F 中选取元素作根结点，然后根据给定的最大深度分别从函数集 F 及终止符集 T 中选取元素。假若待定结点的深度小于给定的最大深度，则该结点从函数集中随机选取，以便算法树延深；若待定结点的深度等于给定的最大深度，则该结点从终止符集中选取，以便终止算法树的生长。

图 3-8 是用完全法生成的深度为 3 的算法树。该树各分支都具有同样的深度。

(2)生长法。用生长法形成的初始个体，每一叶子的深度不一定都等于给定的最大深度，但是算法树的最大深度要等于给定值。

使用生长法时，同样首先从函数集 F 中选取根结点，然后再选取外延的待定结点。假若待定结点的深度小于给定的最大深度，则该结点在函数集 F 及终止符集 T 的并集 $C = F \cup T$ 中选取。假若待定结点的深度等于给定的最大深度，则该结点仅从终止符集内选取，以便终止算法树的生长。

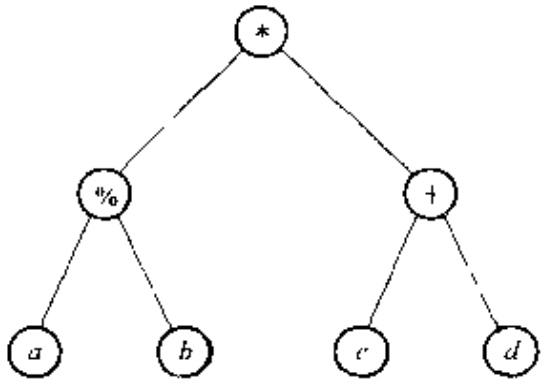

图 3 8　完全法生成的算法树

图 3-9 表示用生长法形成的最大深度为 4 的算法树。很明显，树中各分支长短不等。

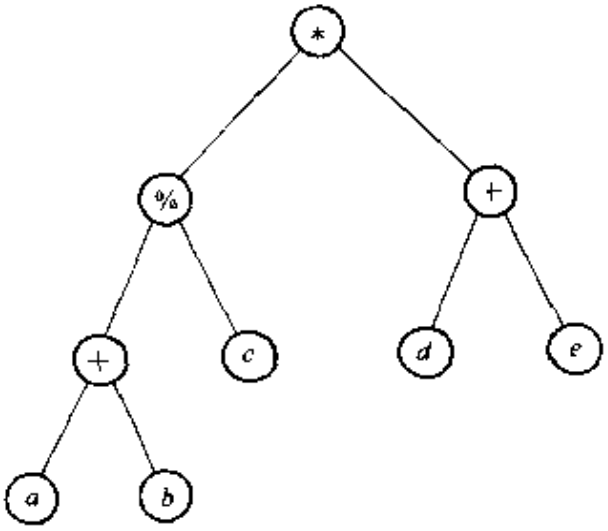

图 3-9　生长法形成的算法树

(3)混合法。单纯用完全法或生长法形成的初始群体，各个体在结构上(树的深度上)会很相似。为了增加群体的多样性，可采用混合法。混合法是完全法和生长法的综合应用。

首先，确定每个初始个体的算法树深度。初始个体的深度在 2 至给定的最大深度之间均匀选取，每一深度下初始个体数目所占百分比为：

$$h = \frac{1}{D-1} \times 100\%$$

式中 D —— 给定的最大深度。

例如，当最大深度 D 为 6 时，深度为 2、3、4、5、6 的个体各占 20%

其次，在每一深度中，50% 的初始个体用完全法产生，另外 50% 的初始个体用生长法产生。这样生成的初始群体形态各异，结构多样。

3.3.3 适应度计算

适应度是衡量个体优劣的尺度，也是遗传规划实现自然选择的主要依据。在遗传规划中，可用四种不同的适应度。

3.3.3.1 原始适应度

原始适应度(Raw Fitness)是问题的自然描述。对于成本、误差这类问题，优良个体的原始适应度小；对于赢利、捕食等问题，优良个体的原始适应度大。因此，用原始适应度衡量个体优劣，有时是愈小愈好，有时是愈大愈好。

原始适应度的选择，往往视问题而定。表 3-4 列举一个曲线拟合的例子。第 2、3 列是真实的输入及输出，它表达下述函数关系：

$$f(x) = x^4 + x$$

表中第 4 列是用 $f(x) = x^2$ 拟合所得的输出。第 5 列用绝对误差 f_1 表示原始适应度，即：

$$f_1 = \sum_i |p_i - O_i|$$

式中 p_i —— 用 $f(x) = x^2$ 拟合的第 i 次模拟输出；

O_i —— 按 $f(x) = x^4 + x$ 计算的第 i 次真实输出。

第 6 列是用方差 f_2 表示原始适应度，即

$$f_2 = \sum_i (p_i - O_i)^2$$

从表中可以看出，尽管绝对误差 f_1 及方差 f_2 都可以反映每次测试的优劣，但前者差别不及后者差别大。

表 3-4　原始适应度的选择

序号	输入 i	真实输出 O_i	模拟输出 p_i	绝对误差	方　差
1	2	3	4	5	6
1	1	1	1	1	2
2	2	0	1	2	1
3	4	20	10	1	25
4	7	50	19	7	20
5	9	90	81	9	81
适应度总和			$f_1=23$	$f_2=151$	

在遗传规划中，常用的原始适应度有：

（1）在模式识别问题中，用匹配的像素点数目衡量，数目愈大愈好；

（2）在机器人控制中，用撞击墙壁次数衡量，次数愈小愈好；

（3）在分类问题中，用正确分类的次数衡量，次数愈大愈好；

（4）在预测问题中，用预测值与实际值的离差衡量，离差愈小愈好；

（5）在博奕问题中，用所赢货币数衡量，货币数愈多愈好；

（6）在人工生命问题中，用所发现及吃掉的食物数衡量，食物数愈多愈好。

3.3.3.2　标准适应度

用原始适应度衡量个体，有时是愈小愈好，有时又是愈大愈好。为了统一衡量标准，于是引出标准适应度（Standarized Fitness）。用标准适应度衡量个体时，一律是愈小愈好；最好个体的标准适应度为 0。

为此，将原始适应度转换为标准适应度有两种情况：

（1）对于原始适应度为越大越好的情况（如博奕赢钱），按下式计算

$$s(i,t)=r_{\max}-r(i,t) \tag{3-1}$$

式中　$s(i,t)$　　第 t 代第 i 个体标准适应度；

$r(i,t)$ — 第 t 代第 i 个体的原始适应度；

r_{max} — 原始适应度所能达到的最大值。

(2) 对于原始适应度为越小越好的情况(如曲线拟合)，则标准适应度等于原始适应度，即：

$$s(i,t) = r(i,t) \tag{3-2}$$

3.3.3.3 调整适应度

在遗传规划的进化后期，个体之间的差别不大，为了更好地区分这种细微差别，希望"放大"适应度。于是，就产生调整适应度(Adjusted Fitness)。调整适应度的计算公式是：

$$a(i,t) = \frac{1}{1+s(i,t)} \tag{3-3}$$

式中 $a(i,t)$ — 第 t 代第 i 个体的调整适应度；

$s(i,t)$ — 第 t 代第 i 个体的标准适应度。

通常，$s(i,t) \geqslant 0$，所以 $a(i,t) \in [0,1]$，且调整适应度值愈大，个体愈优良。

调整适应度的放大功能可从下述例子看出。假设标准适应度 $s(i,t)$ 取值介于0(最佳)及64(最差)之间，标准适应度为64、63的两个较差个体的调整适应度分别为0.0154及0.0156，其差别为0.0002。但是，标准适应度为4、3的两个优良个体的调整适应度分别为0.20及0.25，其差别为0.05。

3.3.3.4 归一化适应度

归一化适应度(Normalized Fitness)是使适应度介于[0,1]区间，它可用下式计算：

$$n(i,t) = \frac{a(i,t)}{\sum_{k=1}^{M} a(k,t)} \tag{3-4}$$

式中 $n(i,t)$ —— 第 i 个体于第 t 代的归一化适应度；

$a(k,t)$ 第 k 个体于第 t 代的调整适应度；

M — 群体中的个体数目。

归一化适应度具有下述三个理想特征：

(1) $n(i,t) \in [0,1]$；

(2) $n(i,t)$ 愈大，个体越优良；

(3) $\sum_{k=1}^{M} n(k,t) = 1$。

上述特征特别有利于应用轮盘法这类基于适应度按比例选择的原则。这时，归一化适应度也就是个体被选择的概率。

对于原始适应度愈大其个体越好的问题，归一化适应度可直接从原始适应度计算，即

$$n(i,t) = \frac{r(i,t)}{\sum_{k=1}^{M} r(i,t)} \tag{3-5}$$

当然，从数学角度讲，上式中 $r(i,t)$ 不允许为无穷大。

3.3.4 基本算子

遗传规划中的基本算子包括复制、交换和突变。其中突变的作用不及遗传算法中那样重要。

3.3.4.1 复制

复制是将上一代优良个体未经任何改变地进入下一代群体，体现进化过程的优胜劣汰原则。为了选择优良个体，类似于遗传算法中的选择，有下述几种方法：

(1) 比例选择法。这种方法是使用最久的方法，其实质是个体适应度越高，被选中的概率愈大，即：

$$p_i = \frac{f_i}{\sum_{k=1}^{M} f_k} \tag{3-6}$$

式中 p_i —— 个体 i 被选中的概率；

f_i —— 个体 i 的适应度；

M —— 群体中个体的总数。

上式中个体的适应度 f_i 要按优良个体的适应度愈高愈好计算，可用调整适应度或归一化适应度。若为后者，归一化适应度值也就是该个体被选中的概率。

(2) 分级选择法，这种方法是将个体按适应度的大小分成许多级别，然后按照个体的级别而不是适应度进行选择，从而降低适应度特高的个体被优先选择的可能性。与此同时，级差选择法可扩大适应度相差不大的个体间的差别，利用分级的级分使适应度不很高的较优个体仍能被选中。

对于线性分级，个体 i 被选中的概率 p_i 为：

$$p_i = \frac{1}{M}\left[\eta^- - (\eta^- - \eta^+)\frac{i-1}{M-1}\right] \tag{3-7}$$

式中 M ——群体中个体的数目；

η^+/M ——最优个体被选中的概率；

η^-/M ——最差个体被选中的概率。

而且为使 M 保持常数，要求

$$1 \leqslant \eta^+ \leqslant 2 \qquad 及 \qquad \eta^- + \eta^+ = 2$$

对于指数分级，上式改为：

$$p_i = \frac{C-1}{C^M-1}C^{M-i} \tag{3-8}$$

式中 C——选择偏好系数，$0 < C < 1$。

应该指出，上述公式与式(2-37)及式(2-39)是一致的，仅仅表达形式稍有不同。

(3) 竞技选择法。这种方法是每次从群体中随机选出一组包含 k 个个体的小群体，然后从中挑选出适应度最高的个体作为复制的对象。被选中的个体经复制后，仍保留在父代群体中，有可能再次进入下一次随机选取的 k 个个体的小群体中。

通过调整 k 的大小，可以改变选择的力度。k 值愈大，被选出的个体愈优。反之，被选出的个体只是相对较优而已。

近年来，竞技选择法已成为主要的选择方法，其原因主要是由于它不需要对每代的所有个体进行集中式的适应度评价，这不仅明显加快群体的进化，而且也便于算法并行计算。

(4) 截取选择法(Truncation Selection)。这种方法源自进化策略(详见下一章)，又称(μ,λ)选择法，μ 代表父代群体拥有的个体总数(相当于本章常用的 M)，λ 是下一代需要产生的子代个体

数(相当于M乘复制概率p_c)。执行选择时,从μ个父代个体中确定性地按适应度大小择优选出λ个优良个体进入下一代群体。由于采用确定性选择,被选中的λ个个体都是名列前茅的优良个体,不会掺入个别不良个体。

截取选择法由于采用确定性选择,执行机时短,但是群体缺乏多样性。

3.3.4.2 交换

3.3.4.2.1 交换操作

交换是将两个父代个体的部分组分相互替代,形成两个新的子代个体,其方法步骤如下(参见图 3-10)。

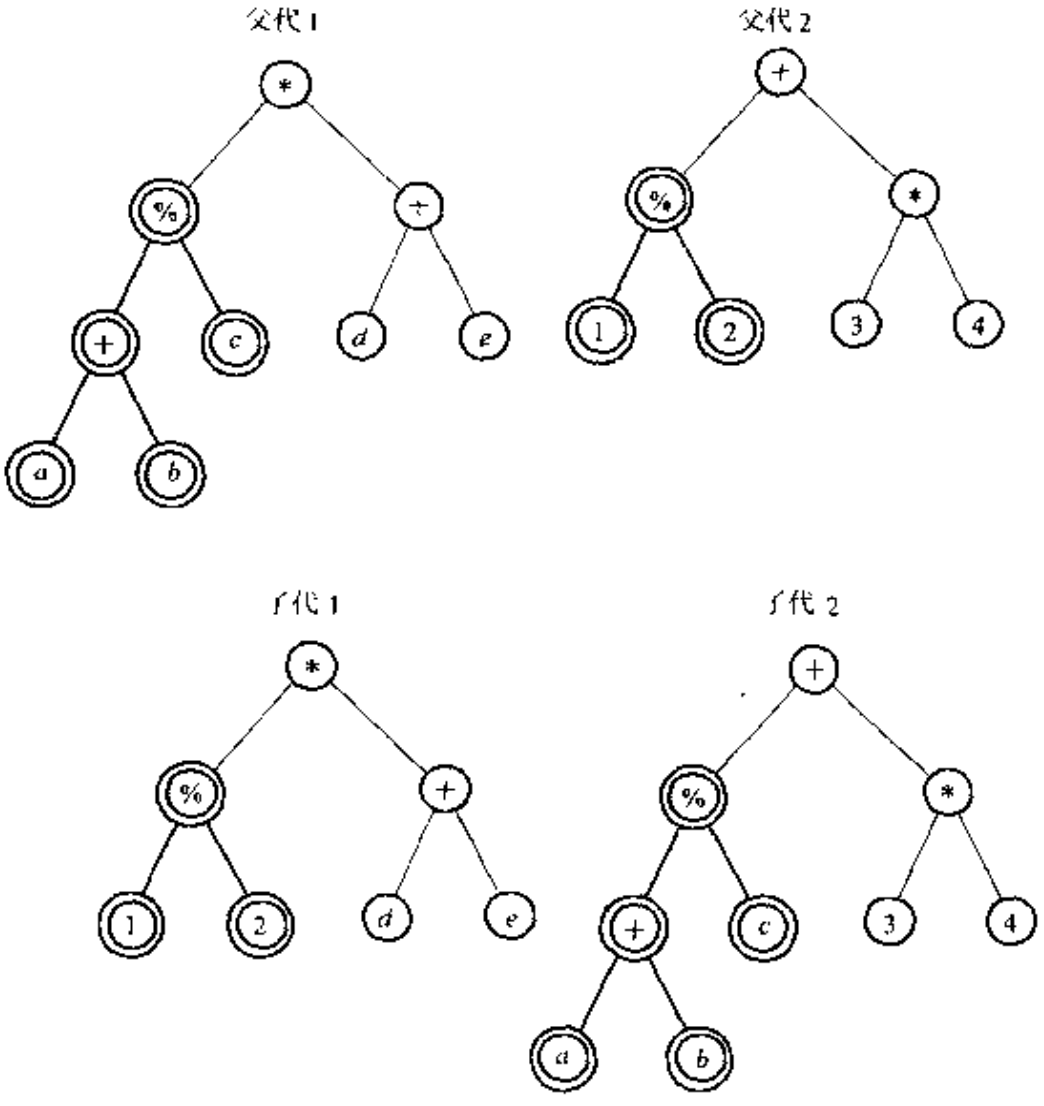

图 3-10 算法树的交换

(1)从父代群体中选择两个优良个体作为交换对象，其选择方法可采用复制中的任何一种择优选择法。

在图 3-10 中，两个父代个体分别是：

父代个体 1： $[(a+b)/c] * (d+e)$

父代个体 2： $1/2+(3 * 4)$

(2)在两个父代个体中用均匀分布的随机方法分别选取交换点，包括交换点在内的交换点以下的子树称为交换段。

在图 3-10 中，假设交换点恰好都是左侧的"%"点，用粗黑线及双圆环表示交换段，它们分别为：

父代个体 1 的交换段： $(a+b)/c$

父代个体 2 的交换段： $1/2$

(3) 交换两个父代个体的交换段，形成两个新的子代个体。

图 3-10 的下半部表示两个新形成的子代个体，它们为：

子代个体 1： $1/2 * (d+e)$

子代个体 2： $[(a+b)/c]+(3 * 4)$

3.3.4.2.2 交换的结果

遗传规划中的交换，是整个子树的交换，因此不论父代个体和交换点如何选择，交换所产生的子代新个体总是语法上合法的表达式。下面列举两个布尔表达式的交换。如图 3-11 所示，两个父代个体的布尔表达式为：

父代个体 1： NOT (D1) OR ((D0) AND (D1))

父代个体 2： ((D1) OR (NOT(D0))) OR((NOT(D0)) AND (NOT(D1)))

假定第一棵树的第 2 个结点(NOT) 被随机选中为交换点，第二棵树的第 6 个结点(AND) 被随机选中为交换点，则两个父代个体的交换段为：

父代个体 1 的交换段： NOT(D1)

父代个体 2的交换段： (NOT(D0)) AND (NOT(D1))

互换这两个交换段，得出图 3-12 描述的两个子代新个体，它们是：

子代个体 1： ((NOT(D0)) AND (NOT(D1))) OR ((D0)

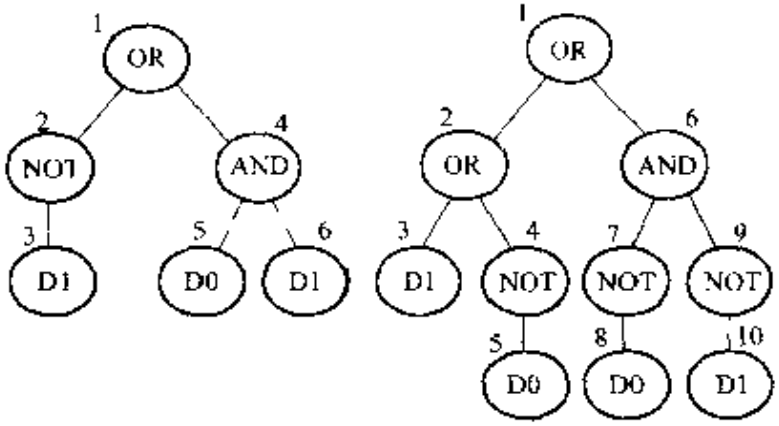

图 3-11　两个父代个体

AND (D1))

子代个体 2：((D1) OR (NOT(D0))) OR (NOT(D1))

它们都是符合语法的合法表达式。

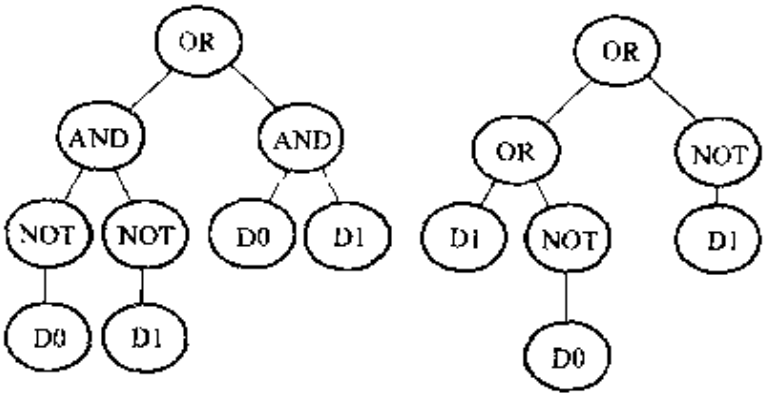

图 3-12　两个子代个体

在交换时，如果一个父代个体的交换段恰为一个终止符（叶子），而另一个父代个体的交换段为一棵子树，那么交换后第一个父代个体产生的子代个体将使算法树的该分支深度增加。

假若一个父代个体的交换点恰好为根，那么该个体的交换段便是整棵算法树。与此同时，若第二个父代个体的交换点是一个内结点，则第一棵算法树插入第二棵算法树中，前者成为后者所生成

新个体的一棵子树，而且新个体是深度明显增大的算法树。

为了防止交换产生巨形新个体，要用树的最大允许深度进行控制。如果有一个子代新个体的算法树深度超过允许值，则删除该巨形新个体，用任一个父代旧个体替代，但保留符合深度要求的另一棵子代算法树。如果两个子代新个体都违背深度要求，则该次交换作废，重新执行交换或用两个父代个体作为子代新个体。

如果两个父代个体的交换段均为终止符（叶子），那么交换仅仅表现为两片叶子的交换，算法树的结构和深度没有改变。

如果两个交换点都是根，那么交换退化为一次复制操作。

3.3.4.2.3 各种交换方法

由于交换是遗传规划的主要进化手段，因此人们对交换方法进行了许多深入的研究。概括起来，交换方法主要有三种：

（1）子树交换（Subtree Exchange Crossover）。这种方法也就是上述的基本交换方法。

图 3-13 描述这种交换。图中左侧为两棵父代个体的示意性算

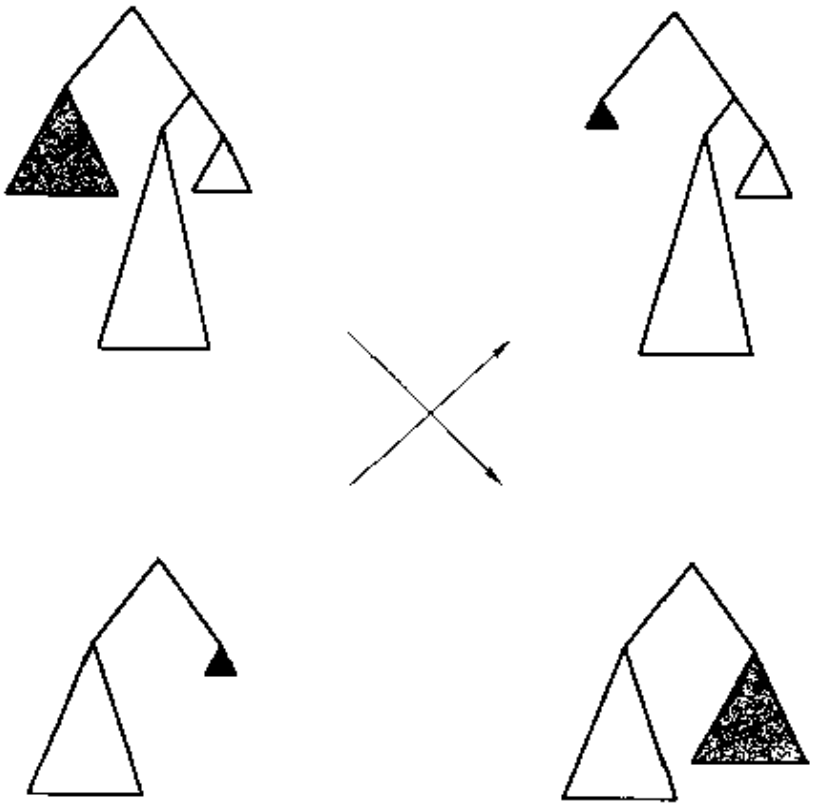

图 3-13 子树交换

法树，其中涂黑的三角形代表准备交换的示意性子树。图中右侧是交换生成的两个子代个体，涂黑的三角形示意交换的结果。

(2)自身交换(Selfcrossover)。这种交换是在单个父代个体上发生。该个体被视作两个相同的个体，然后随意划分为两部分相互交换。

图 3-14 描述这种交换 图中左侧示意性表示复制生成的两个相同个体，上方的浅黑色三角形代表一个交换对象，下方的深黑色三角形代表剩余的子树。图中右侧表示交换产生的两个子代新个体，其中深黑色三角形插入上方的个体，浅黑色三角形插入下方的个体。

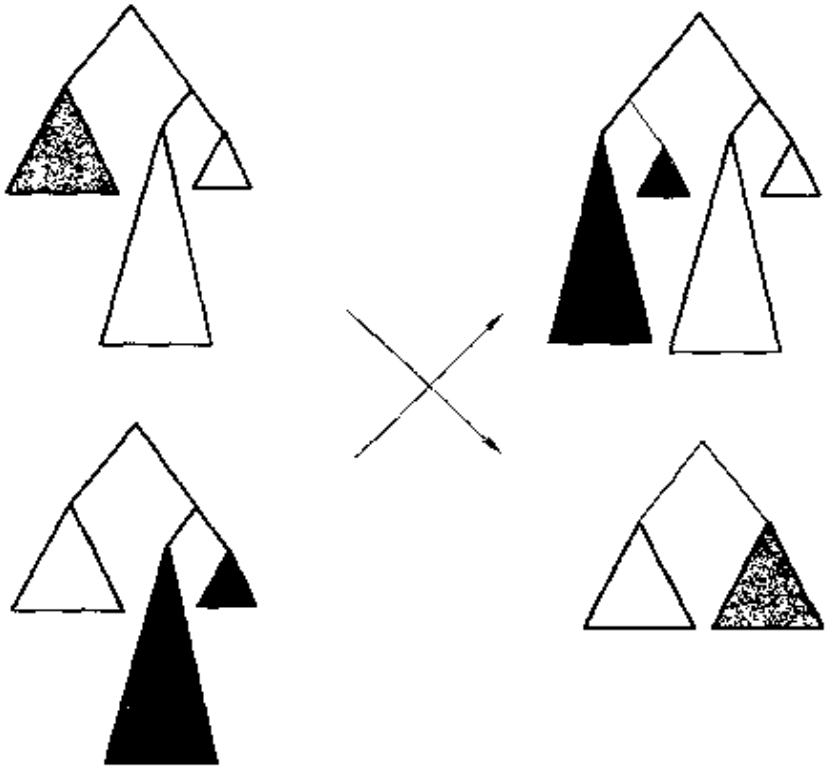

图 3-14 自身交换

(3)模块交换(Module Crossover)。这种交换类似于子树交换，但被交换的不是交换点以下的整棵子树，而仅仅是子树的一部分。

如图 3-15 所示，图中左侧是两个示意性父代个体算法树，其中涂黑的三角形表示拟交换的模块，它们是交换点下属子树的

部分。图中右侧表示交换后产生的子代新个体。

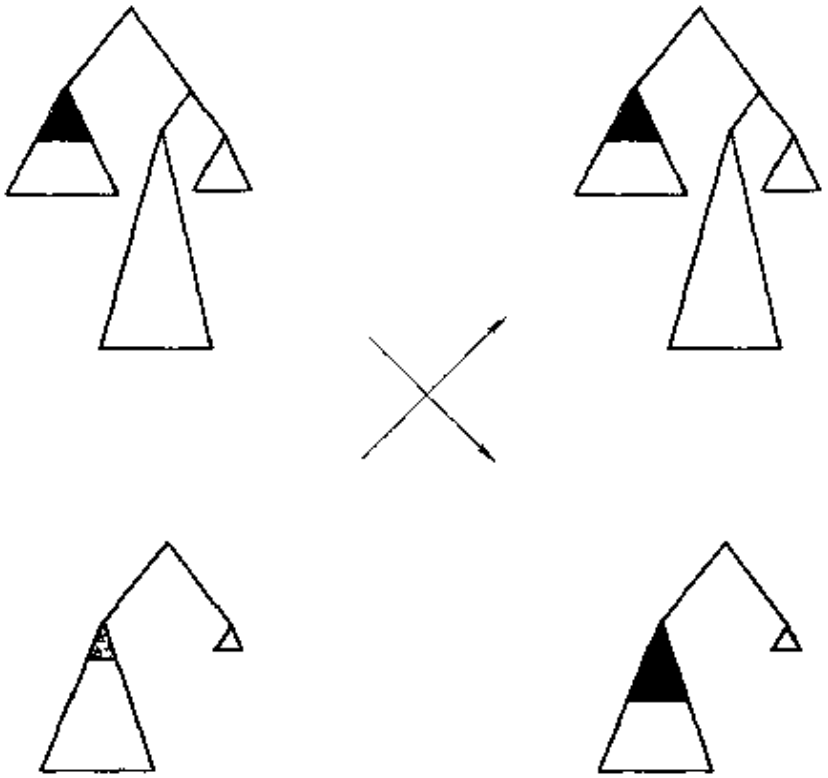

图 3-15　模块交换

3.3.4.3　突变

在遗传规划中，突变的作用比较小。通常，每代只有 5%的个体发生突变，突变概率很小。

执行突变时，先随机选取个体，然后在该个体的算法树上再随机选定一个结点作为突变点。突变点可以是树的内结点（函数点），也可以是树的叶子（终止符点）。接着，删除突变点和突变点以下的子树，用随机产生的一棵新子树插入突变点更换旧子树。对新子树有两点要求：

（1）新子树的产生方法要与初始个体产生的方法相同；

（2）新子树添加后的树的深度要小于允许的最大深度，或新子树的深度要小于某一允许深度。

如图 3-16*a*，点 3(D0) 被选为突变点。假设随机生成的子树为 NOT(D1)，突变后的树如图 3-16*b*。

除了上述基本突变形式外，近年来还有如下的突变形式：

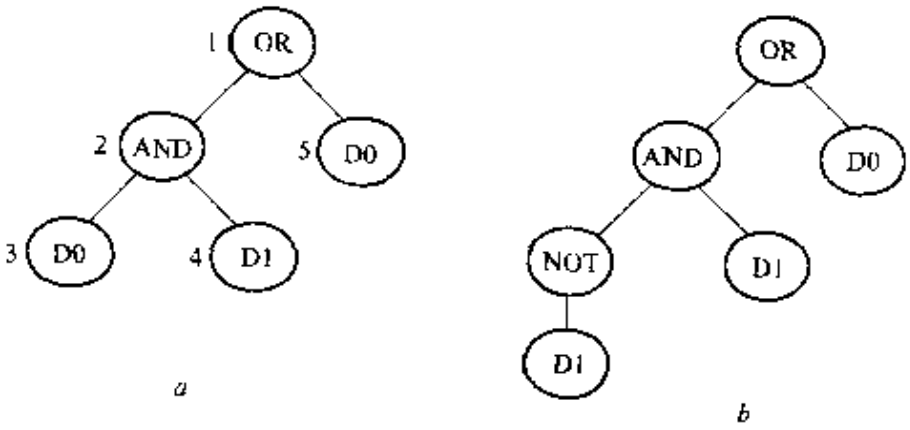

图 3-16　个体的突变

(1) 点突变(Point Mutation)。这种突变仅仅是突变点自身的突变，突变点下面的子树仍保持不变。假若突变点为函数，则在具有相同变量数目的函数集选取元素替换。假若突变点为终止符，则在终止符集选元素。

如图 3-17，上方为父代个体的算法树，即

$$[(x-1)*(x-1)]+\{[(x-1)*(x-1)]*(x-1)\}$$

突变点在算法树右侧分支的第三层"*"，它突变为"+"，如图 3-17 中下方算法树涂黑的结点。子代新个体为：

$$[(x-1)*(x-1)]+\{[(x-1)+(x-1)]*(x-1)\}$$

(2)排列突变(Permutation)。这种突变是将突变点直接下属的变量交换排列顺序。

如图 3-18 所示，上方为父代个体的算法树(同图 3-17)。假设随机选中的突变点在树的右分支第四层左数第二个结点"-"，则将该分支 $(x-1)$ 突变为 $(1-x)$，新的子代个体示于图中下方，涂黑的结点表示交换顺序的结果。

(3) 主从突变(Hoist Mutation)。这种突变是将突变点以下的分支子树，视作一棵独立的子代新个体。

如图 3-19 所示，上方为父代个体的算法树。假设随机选中该树右分支第三层的结点"*"为突变点，则该点下方的子树成为子代新个体，如图中下方涂黑的算法树。

(4) 扩张突变(Expansion Mutation)。这种突变是将父代个体

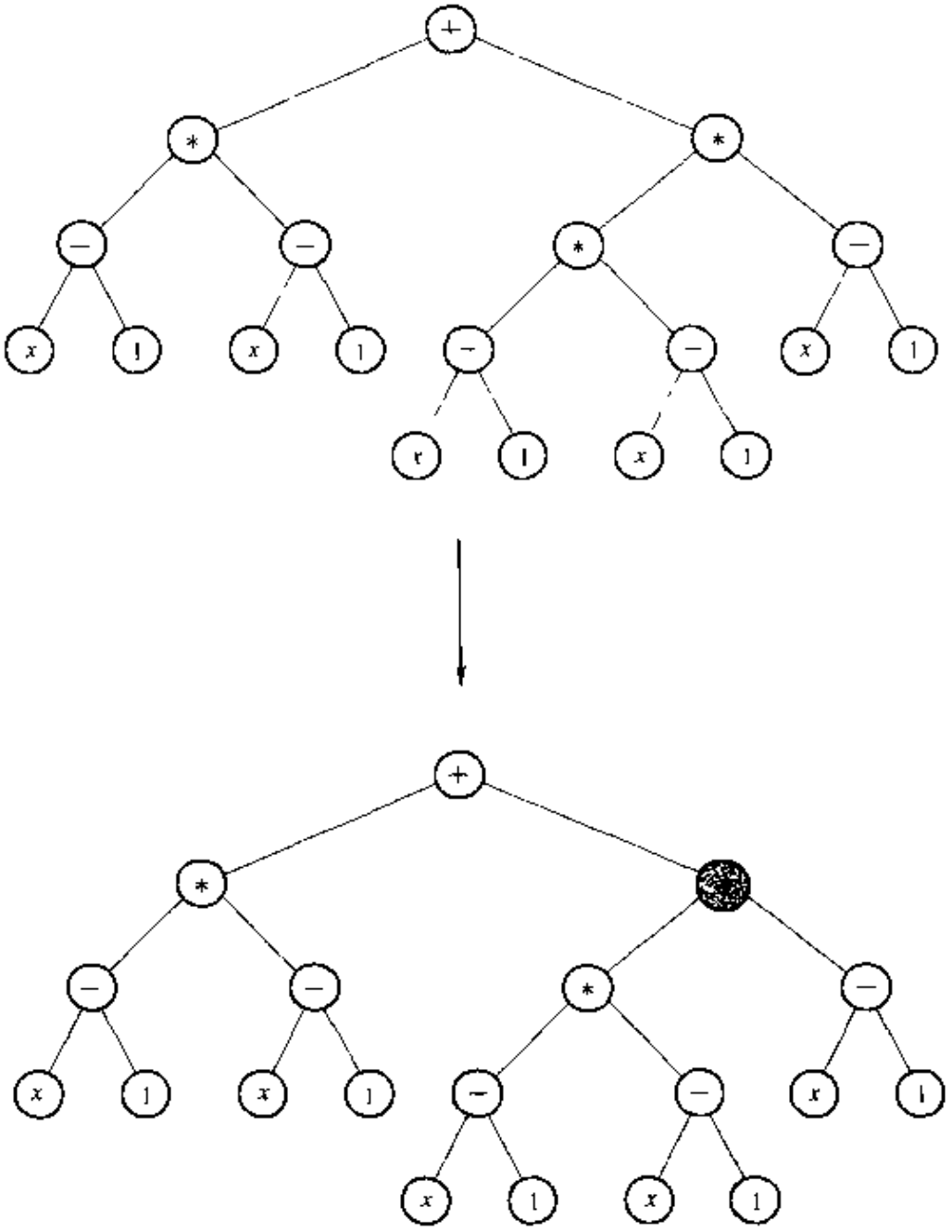

图 3-17　点突变

某一子树插入另一子树的叶子中，扩张为一棵更大的算法树作为子代新个体。

如图 3-20，上方仍为父代个体的算法树。假设突变点在随机选中的右侧第三层的左数第一个结点“*”点，则该结点下属的分支为：

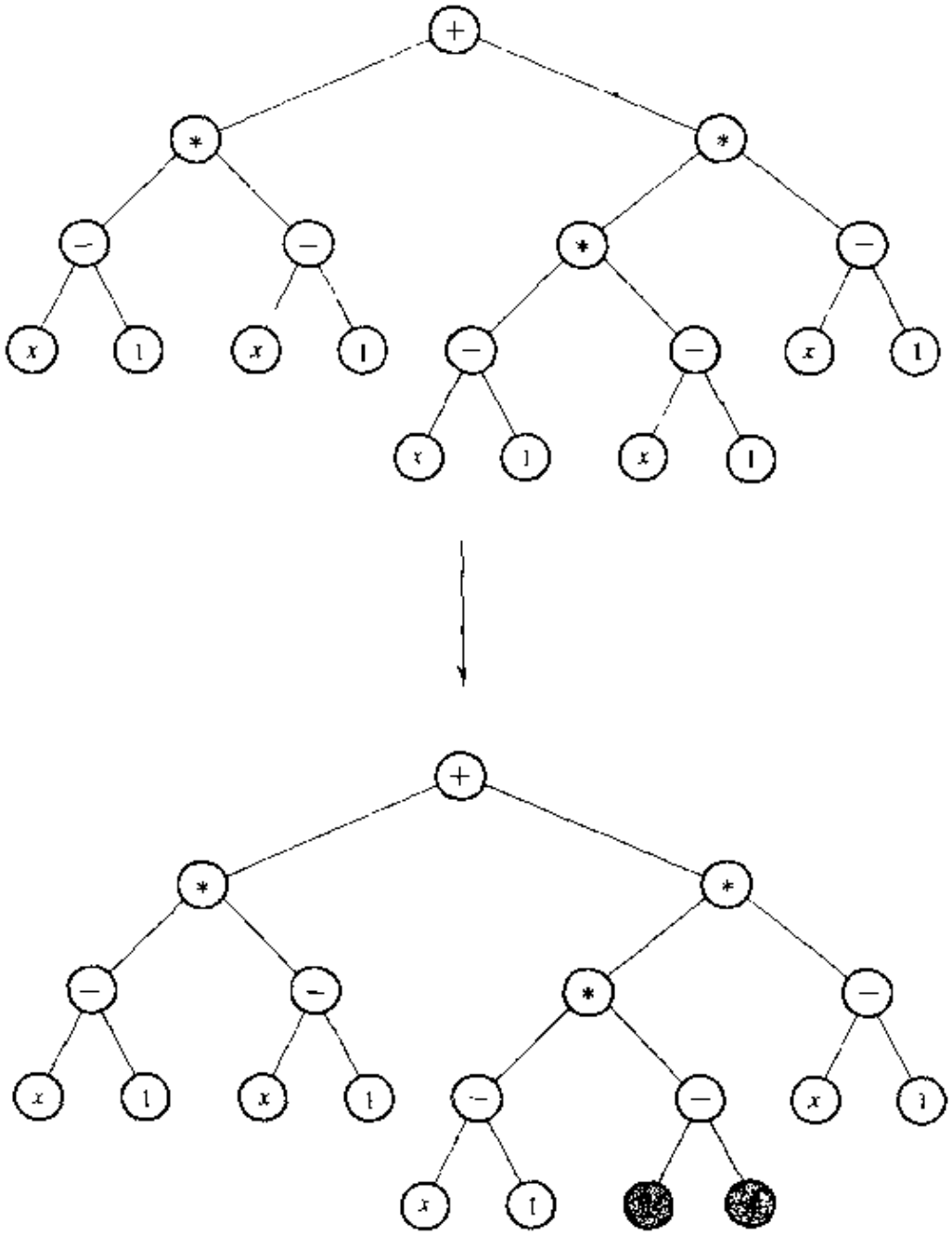

图 3-18　排列突变

$$(x-1)*(x-1)$$

随后又随机选择一个该分支以外的叶结点，假设是左侧分支第四层左数第二个结点“1”，则将上述分支插入该点，生成子代新个体。

(3) 收缩突变(Collapse Subtree Mutation)。这种方法恰好与扩张突变相反。当随机选定父代个体的突变点及其下属分支子树

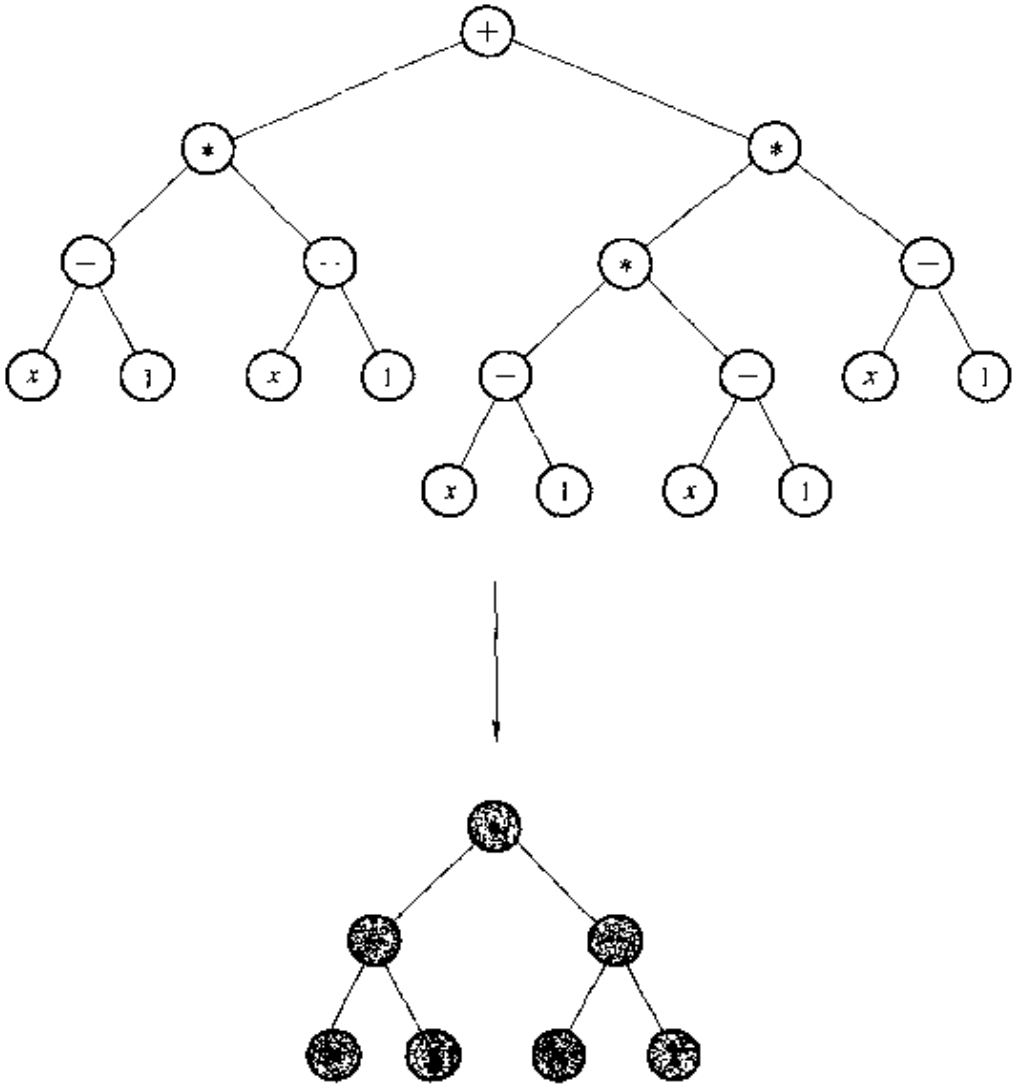

图 3.19　子从突变

后，删除该子树，用随机选出的终止符代替突变点，这样生成的子代新个体小于父代旧个体。

3.3.5　终止

遗传规划是一个不断进化、连续迭代的过程，需要制定准则使运行终止，终止准则通常有三种：

(1) 规定最大允许进化迭代次数 G。一旦迭代次数达到 G 值，运行过程立即停止。G 的大小视问题不同而异，可通过试验确定。

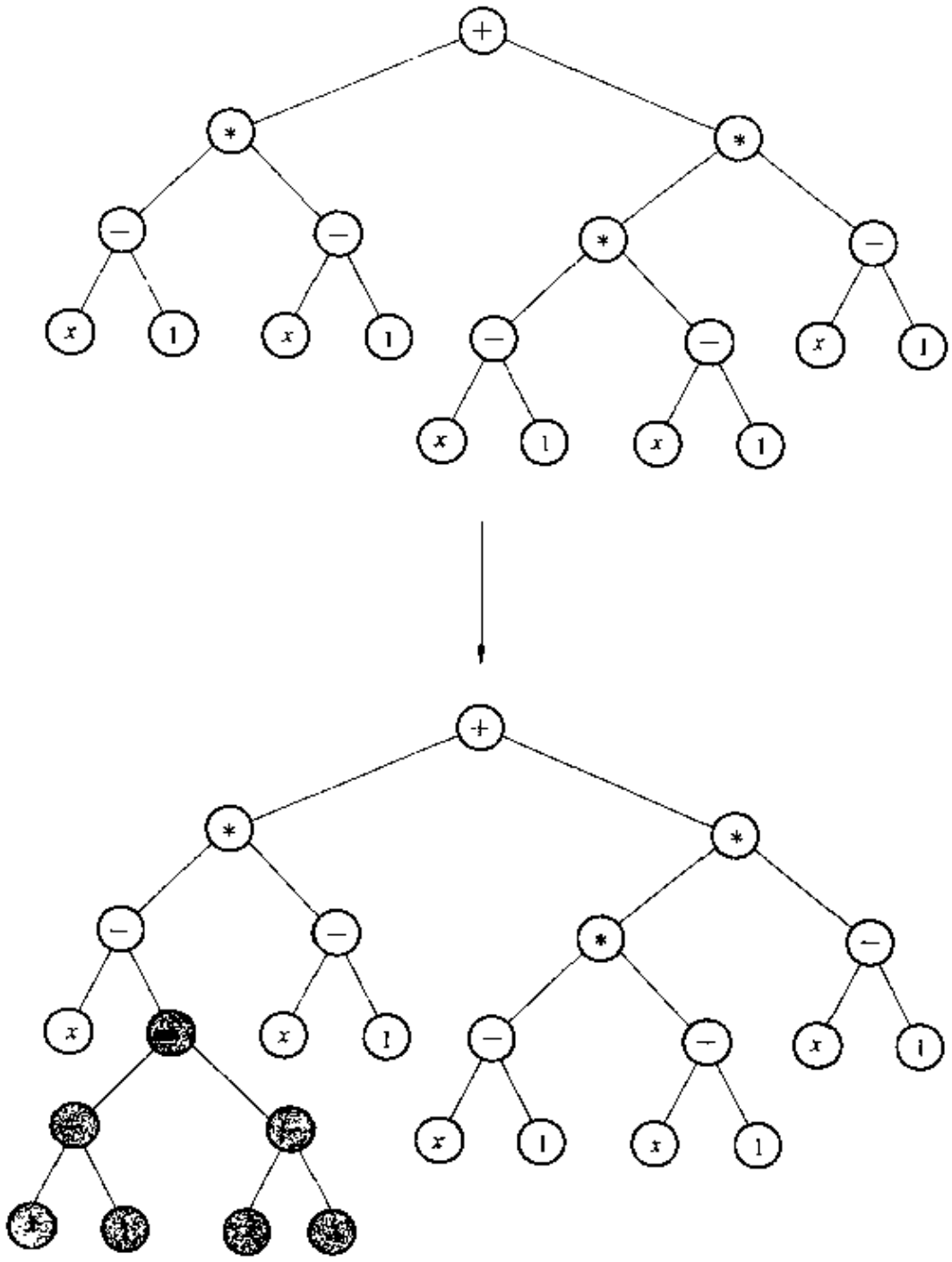

图 3-20　扩张突变

刚开始时 G 不妨选小一些，然后视运行效果逐步增大。

（2）规定允许误差 ζ。对于目标明确的问题，可以规定允许偏离目标的误差。当个体进化到允许误差之内，运行立即停止。为了

防止个别个体的误导，可以采用群体或群体中某一组个体作为测算误差的依据。

（3）观察适应度的变化情况。根据运行过程中个体适应度的变化情况，可在变化趋于平稳后中止遗传规划的运行。图 3 21 是某次图像识别的遗传规划运行情况，图中横坐标表示产生的个体数目，纵坐标表示适应度大小，实线表示平均适应度，虚线表示适应度方差。该遗传规划采用标准适应度，愈是优良的个体其适应度愈小。从图中可以看出，运行刚开始时，适应度急剧降低，达到一个局部最优解后又再次升高，然后逐渐下降，平均适应度慢慢平稳，趋于一个常数，这说明运行在中央部分（个体数为 1.5e + 06）即可终止。

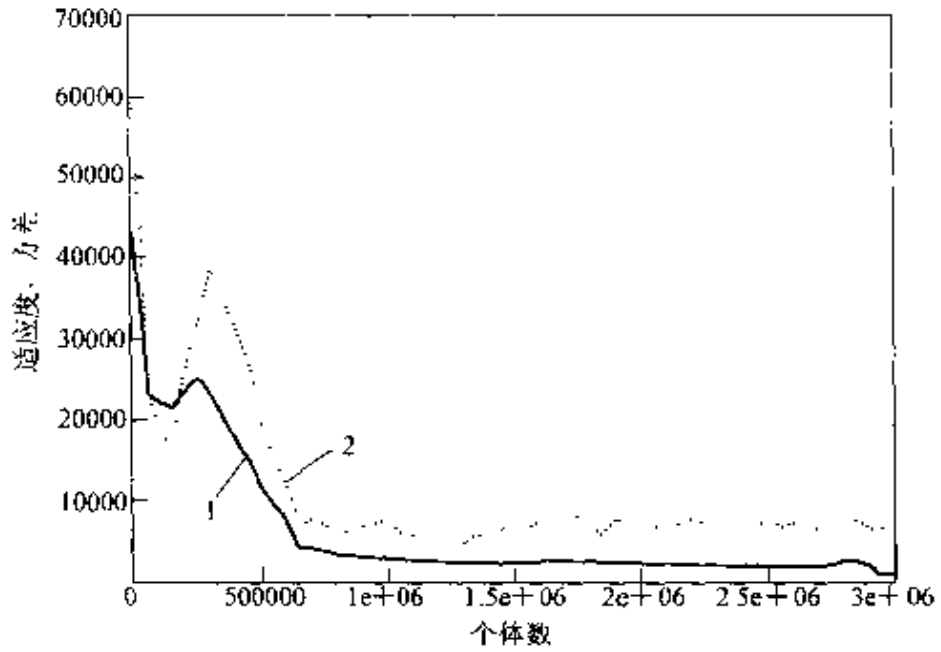

图 3-21　适应度的变化

1　平均适应度；2- 方差

3.3.6　结果标定

遗传规划运行终止后，要找出一个结果作为问题的解答。结果标定的方法有三：

（1）全局最优个体法。这种方法是要找出一个全局最优个体，该个体会隐藏在进化过程的某一代中。为此，在运行过程中每一代

的最优个体都要与前一代的最优个体相比较，若优于前一代，则输入缓冲区；否则，保留前一代最优个体。运行结束后，可从缓冲区输出全局最优个体。

(2) 末代最佳个体法。这种方法是将运行终止前最后一代的最优个体作为问题的解答。通常，末代最佳个体也就是全局最优个体，而且本方法不需要每代都进行最优个体的比较和更新，比上一种方法节省机时及缓冲区。

(3) 多种解答法。这种方法是选出一组优化结果作为问题的解答，其中包括最优解、次优解、次次优解……等等，可供操作者进一步选择。

3.3.7 示例

本节用一个简单的曲线拟合演示遗传规划的工作过程。采用遗传规划从事曲线拟合，可以一次性地同时确定表达式的结构形式及参数，因此又称符号回归(Symbolic Regression)。

表 3-5 列举已知 10 组试验数据，其实它是按下述函数关系计算而得：

$$y = f(x) = \frac{x^2}{2}$$

现在要用遗传规划进行符号回归，求出这些数据的表达式。

表 3-5 符号回归原始数据

序　号	输入 x	输出 y
1	0.000	0.000
2	0.100	0.005
3	0.200	0.020
4	0.300	0.045
5	0.400	0.080
6	0.500	0.125
7	0.600	0.180
8	0.700	0.245
9	0.800	0.320
10	0.900	0.405

3.3.7.1 准备工作

为了实现遗传规划，要做如下准备工作：

(1) 确定终止符集。对于本例，假设终止符集包括变量 x 及介于[-5，5]之间的整数。

(2) 确定函数集。对于本例，假定函数集包括算术运算符 +、-、*、%(预防性除法)。

(3) 确定适应度。本例选用标准适应度，用 10 组数据返回的方差衡量，最优个体的适应度为零。

(4) 确定遗传规划的基本参数及方法，如群体规模、算法树最大允许深度、选择方法、初始个体生成方法以及遗传概率等。表 3-6 列举所需的基本参数。

表 3-6 遗传规划基本参数

参 数	值
目标	确定拟合已知数据的符号表达式
终止符集	变量 x，介于[-5.5]的整数
函数集	+、-、*、%
群体规模	600
交换概率	0.90
突变概率	0.05
选择方法	竞技选择法，$k=4$
终止准则	规定最大允许代次
最大允许代次	100
交换后算法树最大深度	200
最大突变深度	4
初始群体产生方法	生长法

3.3.7.2 运行结果

在初始群体中，个体的适应度差别很大，图 3-22 表示第 0 代最好的个体，它是：

$$f_0(x) = x/(1.0 - x/x) = \frac{x}{3}$$

图 3-23 表示第 1 代最优的个体，此时算法树通过交换及突变而扩张，其实际表达式是：

$$f_1(x) = \frac{x}{6 - 3x}$$

图 3 24 及图 3 25 表示第 2 代最优的个体，其中图 3-25 是图 3 24 的延续，由此可见第 2 代的

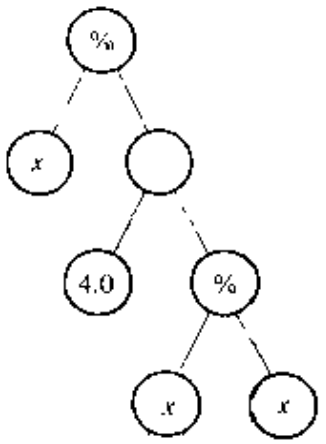

图 3-22 第 0 代最优个体

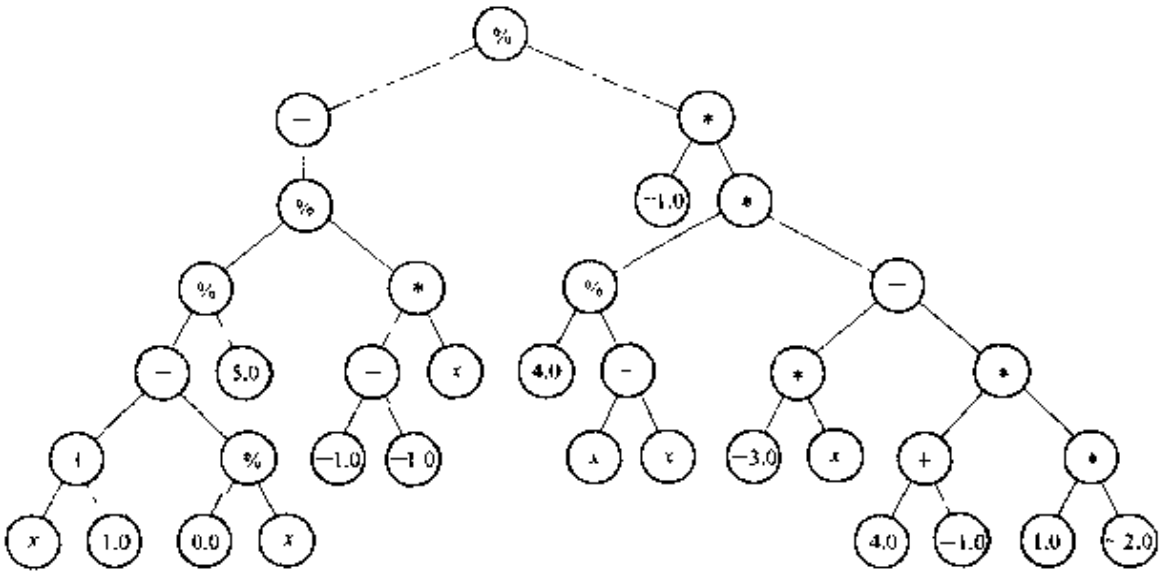

图 3 23 第 1 代最优个体

算法树已急剧扩大。此时最优个体的表达式可简化为：

$$f_2(x) = \frac{x}{x(x-4) - 1 + \frac{4}{x} - \frac{9(x+1)}{\frac{5x}{6-3x}} + x}$$

运行至第 3 代，出现正确结果，如图 3-26 所示，它为：

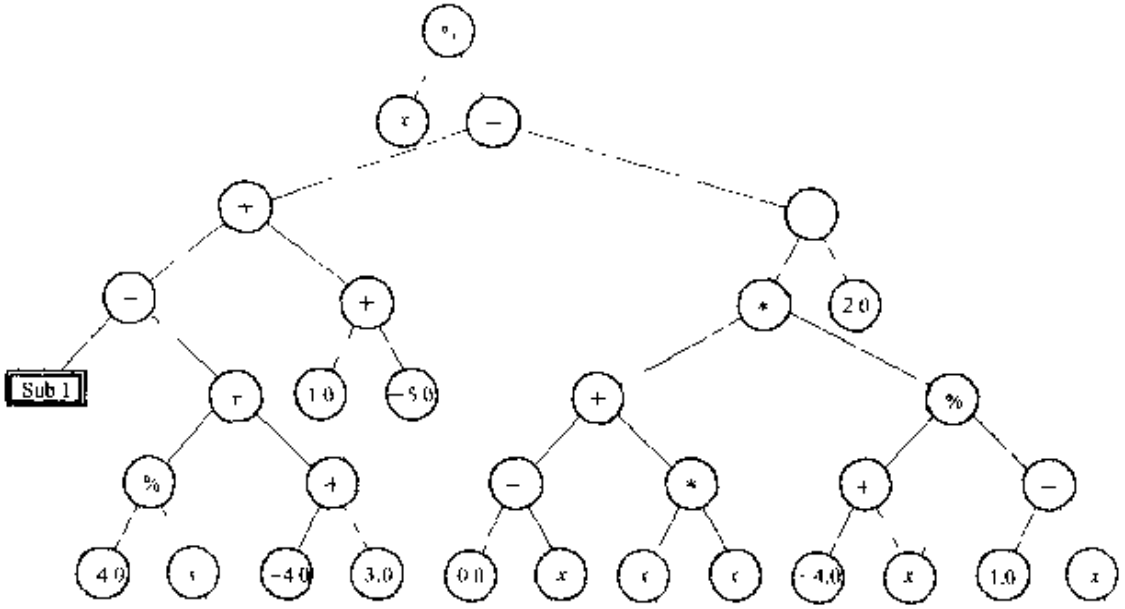

图 3-24　第 2 代最优个体(主体)

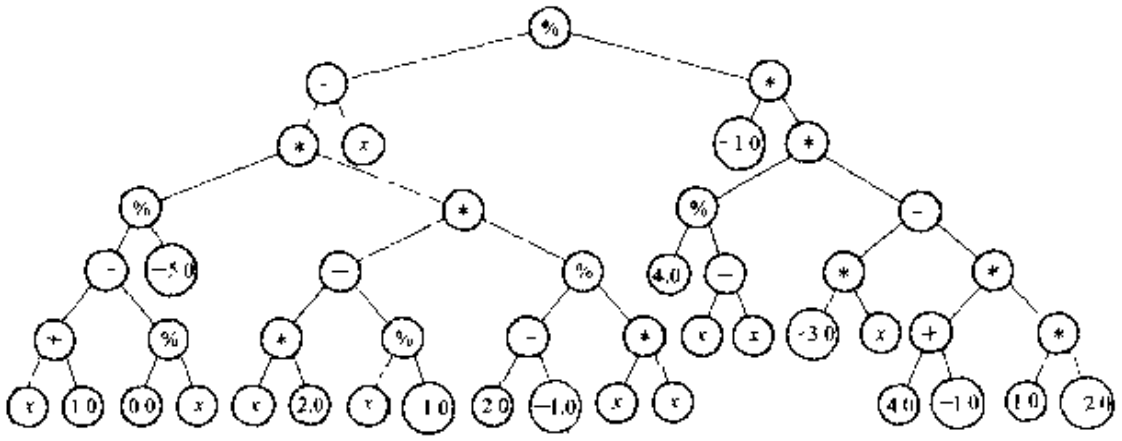

图 3-25　第 2 代最优个体(Sub1 分支)

$$f_3(x) = (x * x)/2 = \frac{x^2}{2}$$

表 3-7 列举上述各代最优个体的输出值 y。图 3-27 表示这些输出值的动态特征。

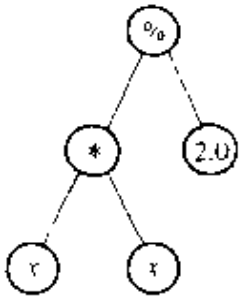

图 3-26　第 3 代最优个体

表 3-7　各代最优个体的输出值

序　号	目标输出	第 0 代	第 1 代	第 2 代	第 3 代
1	0.000	0.000000	0.000000	0.000000	0.000000
2	0.005	0.033333	0.017544	0.002375	0.005000
3	0.020	0.066667	0.037037	0.009863	0.020000
4	0.045	0.100000	0.058824	0.023416	0.045000
5	0.080	0.133333	0.083333	0.044664	0.080000
6	0.125	0.166667	0.111111	0.076207	0.012500
7	0.180	0.200000	0.142857	0.122140	0.180000
8	0.245	0.233333	0.179487	0.188952	0.24500
9	0.320	0.266667	0.222222	0.287024	0.320000
10	0.405	0.300000	0.272727	0.432966	0.405000

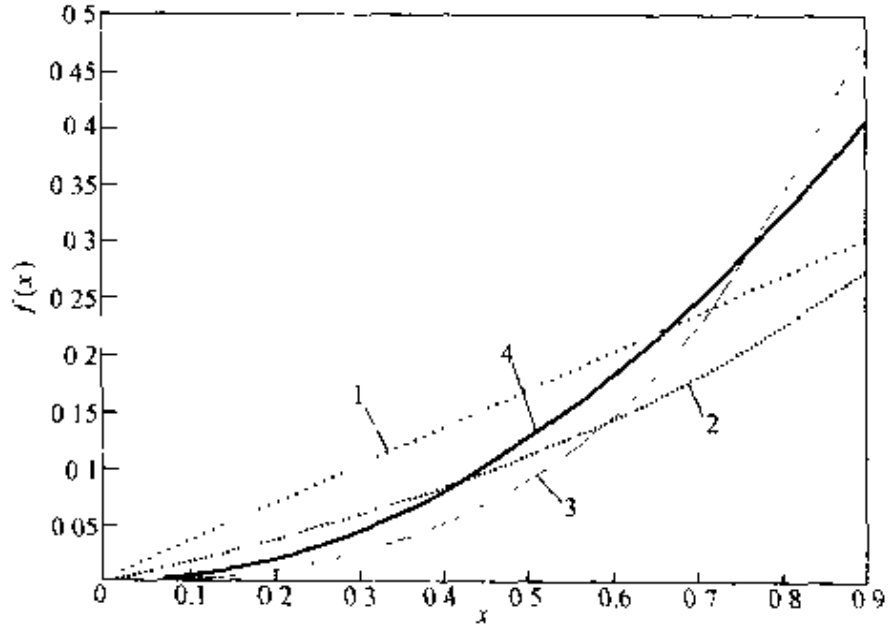

图 3-27　各代(0、1、2、3 代)最优个体

1—第 0 代;2—第 1 代;3—第 2 代;4—第 3 代

从第 3 代继续运行下去，最优个体规模再度扩张。图 3-28 表示第 5 代的最优个体，尽管算法树扩张，但仍是最优结果，即

$$f_5(x) = \frac{x^2}{2}$$

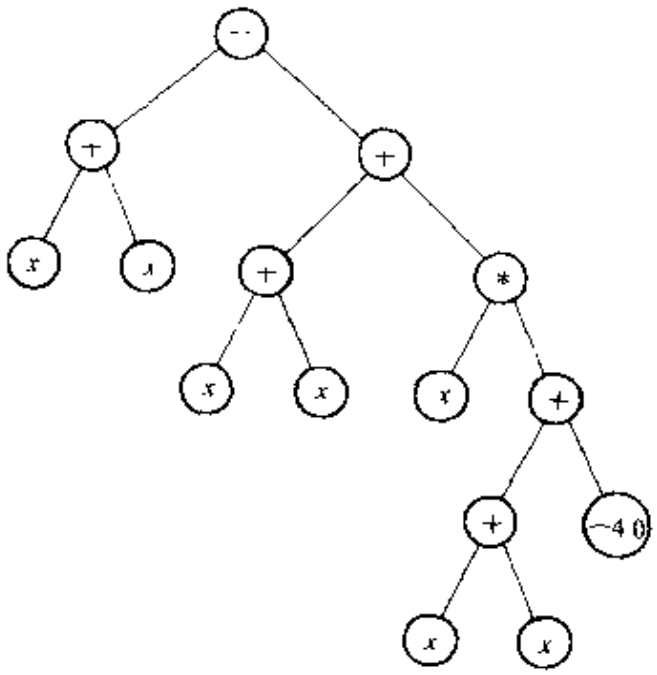

图 3-28　第 5 代的最优个体

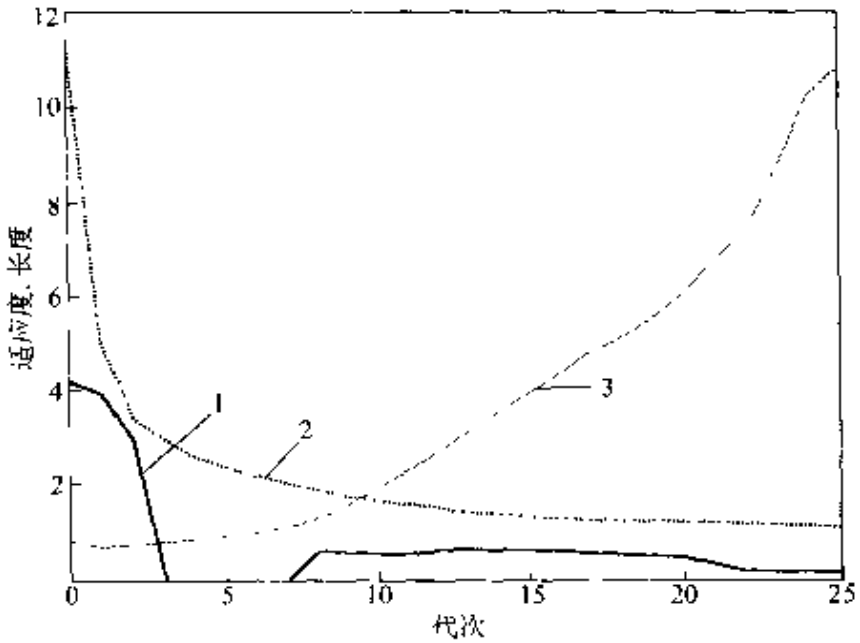

图 3-29　各代适应度及个体长度的变化

1　最优适应度(已放大 10 倍)；2　平均适应度；3　平均长度(已缩小 50 倍)

3.3.7.3 适应度及个体尺寸的变化

图 3-29 描述遗传规划运行 25 代次中各代次最优适应度(已放大 10 倍)、平均适应度以及个体平均长度(已缩小 50 倍)的变化情况。从图中可以看出,最优适应度(图中 1)在运行前期明显减小,于第 3～7 代为零,以后又稍有升高。平均适应度(图中 2)在初期明显减小后,继续缓慢降低。个体的平均长度(图 3)则在不断加大,这是由于多次交换及突变造成。

3.4 遗传规划的理论分析

3.4.1 模式理论

遗传规划中的模式理论,不如遗传算法那样成熟和完整,至今仍无法从理论上严密预测优良模式如何在运行中壮大发展。这主要是由于遗传规划的个体表达式经常变化其结构及大小。

3.4.1.1 模式

Koza 在 1992 年定义模式为:模式是群体中含有一个或多个特定子树的所有个体(树)的集合。也就是说,一个模式是一个具有某些共同特征的符号表达式的集合。例如,若定义一个模式 H_1 是下述符号表达式的集合:

$$H_1 = \{(2\ -\ y),(2\ +\ 3)\}$$

那么所有包含 $(2-y)$ 和(2+3)子树的个体都属于模式 H_1。Koza 关于模式的这种定义没有提及模式的阶、长度以及不关心的字符"#"。

1994 年 O' Reilly 等人重新定义遗传规划的模式,她在 Koza 的基础上加入不关心的字符"#"。因此,模式可定义为:模式是群体中含有同样数目特定子树的所有个体(树)的集合,其中子树中包含不关心的结点"#"。例如,可以定义如下一个模式 H_1:

$$H_1 = \{(\#\ -\ y),(2\ +\ \#)\}$$

那么,含有 $(2-y)$ 及(2+3),或者(5-y)及(2+6)子树的个体都属于模式 H_1。

很明显,O' Reilly 关于模式的定义更接近遗传算法的模式本

意，可以套用遗传算法的模式理论。

3.4.1.2 模式的阶次

模式的阶次是模式中含有明确意义的结点数目。

例如，在模式 $H=\{(\#-y),(2+\#)\}$ 中，具有明确意义的结点依次为："－"、"y"、"2"、"＋"，因此该模式的阶次为 4。很明显，阶次愈小，该模式的概括性愈强，而且被破坏的可能性愈小。阶次的这种性质和遗传算法是一致的。

3.4.1.3 模式的定义长度

模式的定义长度是模式中子树内连接弧数目以及连接子树的弧数目之总和。

例如，在模式 $H_1=\{(\#-y),(2+\#)\}$ 中，子树内的连接弧数目为：$2+2=4$，而连接$(\#-y)$及$(2+\#)$两棵子树的弧数目为 2，因而模式 H_1 的定义长度为：$4+2=6$。显然，模式的定义长度愈长，它被破坏的可能性愈大。

3.4.1.4 模式定理

仿效遗传算法的模式定理，O' Reilly 也列出下述公式：

$$E[m(H,t+1)]\geqslant m(H,t)\frac{f(H,t)}{f(t)}(1-p_c\,p_d(H,t)) \tag{3-9}$$

式中 $m(H,t)$ —— 模式 H 在第 t 代拥有的个体数目；

$E[m(H,t+1)]$ —— 模式 H 在第 $t+1$ 代拥有个体数目的期望值；

$f(H,t)$ —— 模式 H 在第 t 代所拥有的个体的平均适应度；

$f(t)$ —— 第 t 代群体平均适应度；

p_c —— 交换概率；

$p_d(H,t)$ —— 模式 H 在第 t 代被破坏的最大概率。

上述公式与遗传算法的模式定理是一致的。以复制为例，如果某个体的适应度是群体平均适应度的 2 倍。那么复制将产生 2 个这样的个体，使该个体所在模式拥有的个体数目 $m(H,t+1)$ 增加。对于交换，它是模式被破坏的主要原因。上式也反映模式的*仟*

活概率与交换概率 p_c 呈反比关系。

不过，上述公式没有明确表达模式的阶次及长度的影响，这主要是由于遗传规划的表达方式造成。在遗传规划中，个体的表达式呈层次状，其结构及大小经常发生变化，很难精确预测。因此，遗传规划的模式理论尚有待发展及完善。

3.4.2 交换的作用

交换是遗传规划产生新个体的主要算子，也是促进群体进化的主要动力。因此，人们对交换在遗传规划中的作用进行了深入的研究。

3.4.2.1 简单示例

图 3-30 的算法树，代表遗传规划运行中的一个个体。假设节点 7～9 组成构造块（深黑色），促使个体向优良方向发展。在此个体中有 19 个可进行交换的交换点，假设用均匀随机选择的方法确定交换点，则构造块（7～9 节点）被交换破坏的概率为 2/19，即 10.5%。

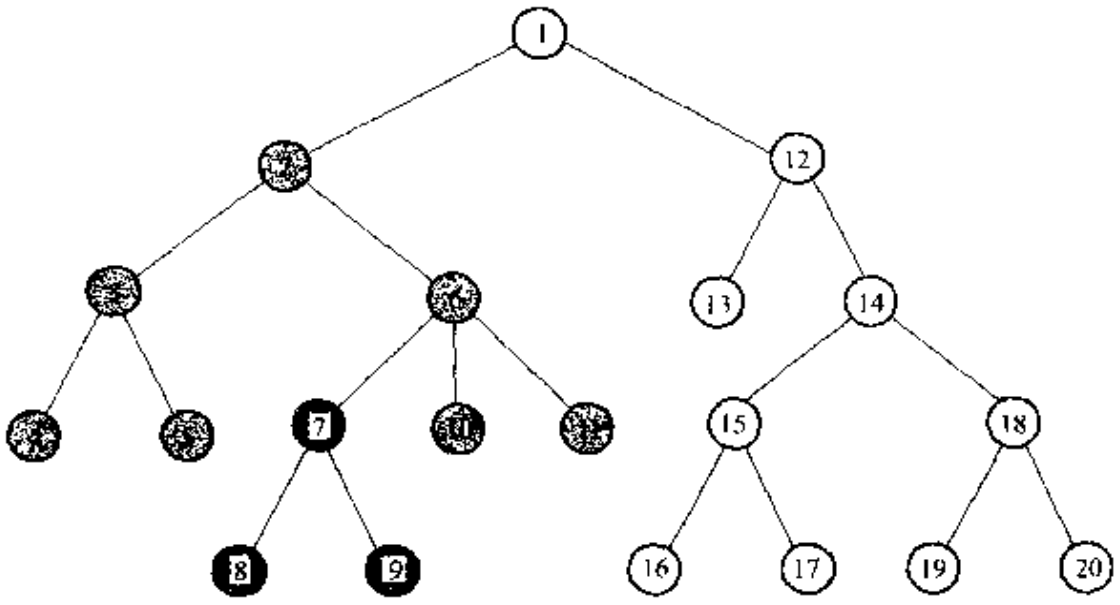

图 3-30 交换作用示例

假设交换后产生一个新的更大的构造块（结点 6～11），此构造块被交换破坏的概率为5/19，即 26.3%。

假设结点 1～11 组成另一新的构造块，此构造块被交换破坏的概率为 10/19，即 52.6%。

从上述三个构造块可知，随着遗传规划的进化，构造块会不断扩张，说明交换促进了个体的进化，但是与此同时新的构造块却愈来愈脆弱，容易遭受交换破坏。假设问题的最优解就是最后一个构造块（结点 1～11），那么当遗传规划得到图 3-31 的个体时，它只需要删除结点 12 便可得到最优解，然而这时构造块被交换被坏的概率达到 10/11，即 90.9%。

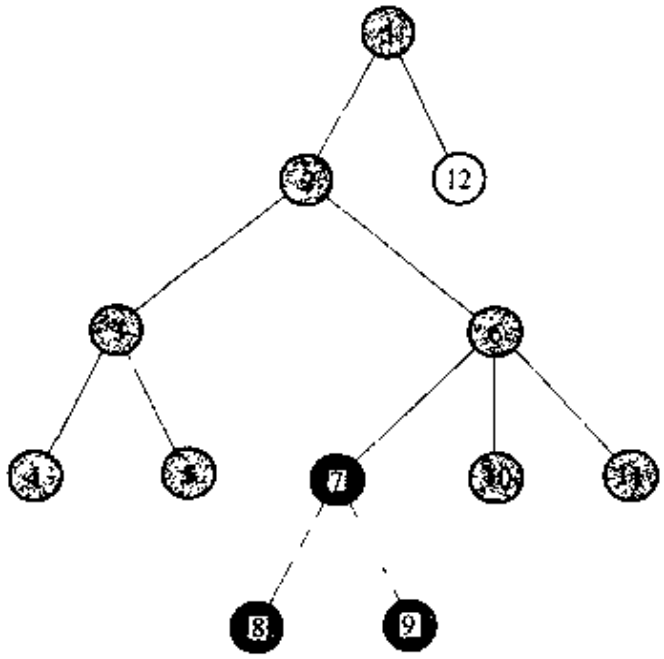

图 3-31　交换破坏作用

交换不仅通过破坏构造块降低个体的适应度，而且有时还可以由于将构造块插入不适当的位置而减小个体适应度。因此，在遗传规划中交换对进化既有积极的作用，又有消极的作用。

3.4.2.2　观测结果

为了进一步说明交换在遗传规划中的作用，人们对遗传规划进行具体观测。观测中用适应度变化的相对值 Δf 衡量交换效果，即：

$$\Delta f = \frac{f_b - f_a}{f_b} \tag{3-10}$$

式中　f_b —— 交换前的个体适应度；

f_a —— 交换后的个体适应度。

上述适应度按标准适应度计算，即最优个体的适应度为 0，最差个体为∞。

图 3-32 表示具体观测结果。图中最左侧是适应度降低超过 100%的个体数目累计数。根据该图，可将交换分成三类：

(1) 破坏性交换。交换使个体的适应度降低 2.5%以上，如图中左侧所示。

(2) 中性交换。交换使个体的适应度变化介于±2.5%，如图中中部的个体。

(3) 建设性交换。交换使个体的适应度提高 2.5%以上，图中很少。

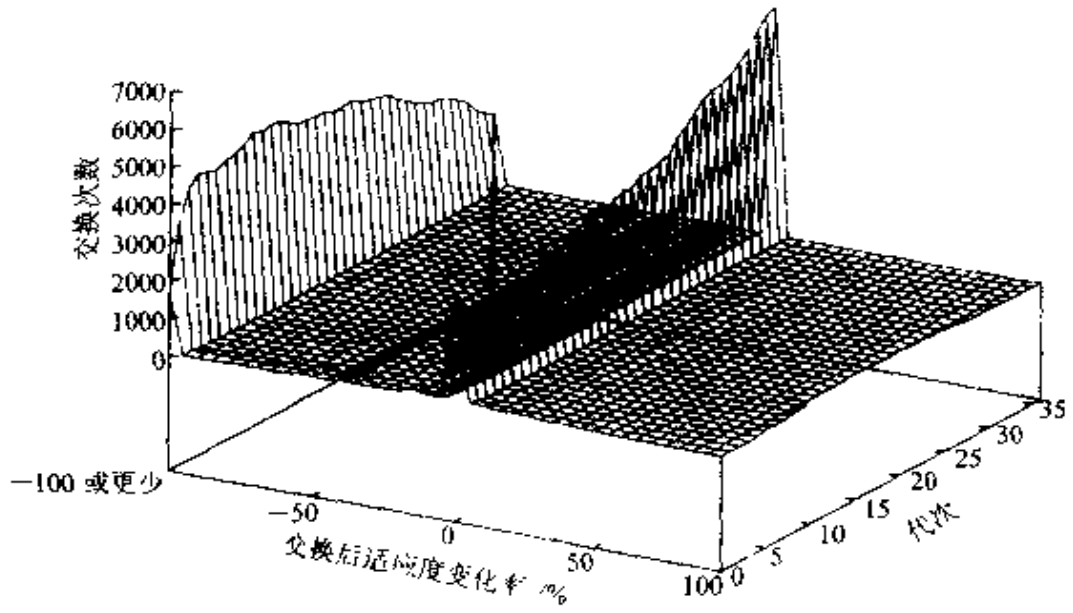

图 3-32　交换的破坏作用

其他人对交换的也有类似的观测结果。例如，有的观测指出，交换后 75%的个体的适应度比原来恶化 50%。另一个观测指出交换后只有 10%的个体改善了适应度。

3.4.2.3　交换的改进

尽管交换具有破坏性作用，然而它仍然是驱动遗传规划向良

好方向进化的主要动力，因此人们设计了许多新的交换方法。

（1）孵卵组合(Brood Recombination)。这种方法是从两个父代个体中经过多次交换产生多个子代新个体，然后从众多新个体中只选择最好的两个作为子代个体。其工作步骤如下(图 3 33)：

1）从群体中选取两个父代个体；

2）对父代个体实施 N 次交换(图中为 4 次)，产生 $2N$ 个子代个体(图中为 8 个)；

3）从 $2N$ 个新个体中挑选适应度最好的两个作为子代新个体，其余新个体舍弃。

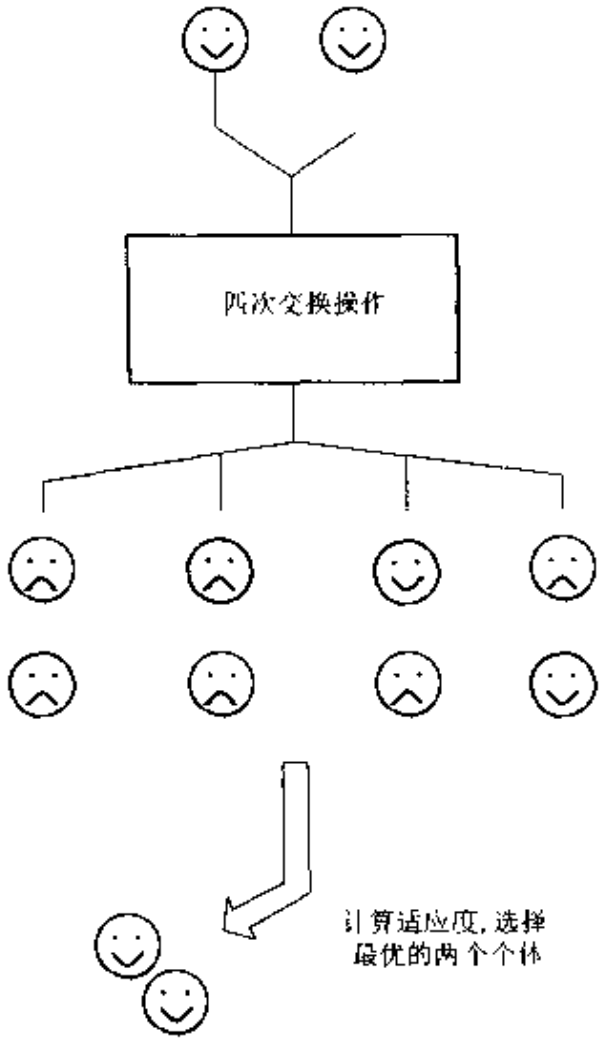

图 3-33　孵卵组合

（2）同源交换(Homologous Crossover)。这种方法力求促使相似交换体之间的交换，使新生的子代个体继承父代的特征。为了判别个体之间的相似性，采用结构相似性和功能相似性两种度量手段。

结构相似性用编辑距离(Edit Distance)衡量。对于算法树而言，编辑距离是将算法树 g 变成算法树 h 所需要删减或添加元素的最小次数，记作 $\delta(g,h)$。例如，表达式 $g=a+(b/c)$ 和 $h=(a*b)+c$ 写成前缀形式分别为：

$g=+\ a/bc$ 及 $h=+\ *abc$

使 g 变成 h 需交替删除或添加某个字符，记作 del(k) 或 add(k)，其中 k 代表结点。按深度优先顺序，上述替换依次为：

del(*b*),del(*c*),del(/),add(*c*),del(*a*),add(*),add(*a*),add(*b*)

因此，$\delta(h,g)=8$。

功能相似性用算法树的输出表示。将计算适应度的样本数据代入该算法树的表达式中，则得到输出。

本方法的具体工作步骤如下：

1)随机选择两个父代个体。

2)结构相似性计算。针对算法树较大的父代个体的每条弧 k，按深度优先顺序在另一个体(较小的算法树)上找具有最小编辑距离的子树，然后将所有最小距离相加，得：

$$\overline{D}_s = \sum_k D_s(k, i\min(k)) \tag{3-11}$$

式中 k ——较大算法树上的弧；

$i\min(k)$ ——较小算法树上针对 k 弧指向具有最小编辑距离的子树的弧；

$D_s(k,i\min(k))$ ——$i\min(k)$ 弧所指向子树的最小编辑距离；

$\overline{D}_s$——最大算法树上所有弧的最小编辑距离之和。

进一步，对 $D_s(k,i\min(k))$ 正则化，得 k 弧的结构相似性 $D_s^N(k,i\min(k))$，即：

$$D_s^N(k,i\min(k)) = \frac{D_s(k,i\min(k))}{\overline{D}_s} \tag{3-12}$$

3)功能相似性计算。针对较大的算法树的 k 子树，代入第 α 组输入样本值计算输出 $O_k^{(\alpha)}$，相应地也计算较小算法树上对应于 k 弧的子树的输出 $O_{j\min(k)}^{(\alpha)}$，再累加各组样本值下的输出误差，得：

$$D_F(k,j\min(k)) = \sum_\alpha |O_k^{(\alpha)} - O_{j\min(k)}^{(\alpha)}| \tag{3-13}$$

式中 $D_F(k,j\min(k))$ ——各组输入样本值下输出差的绝对值之和。

进一步，针对较大算法树的所有弧累计输出差之和：

$$\overline{D}_F = \sum_j D_F(j,j\min(j)) \tag{3-14}$$

对 $D_F(k,j\min(k))$ 正则化，得 k 弧的功能相似性 $D_s^N(k,i\min(k))$：

$$D_F^N(k,j\min(k)) = \frac{D_F(k,j\min(k))}{\overline{D}_F} \tag{3-15}$$

4)选择交换点。按表 3-8 任一种方法将相似性作为每条弧的交换概率。表中 n 为正则化系数，使交换概率 p 小于 1。

表 3 8　交换概率选择方法

序　号	方法简称	弧 k 的交换概率 p
1	SMD	$D_S^N(k,i\min(k))$
2	FMD	$1-D_F^N(k,j\min(k))$
3	FMD/SMD	$D_I^N(k,i\min(k))[(1-D_F^N(k,j\min(k))]/n$

3.4.3　基因内区

3.4.3.1　基因内区的定义

在生物学中，基因内区(Intron)是指染色体内对生物性态没有直接效用的基因。在遗传规划所产生的个体中，有些表达式也是多余的，如：

$$x+0,\ x*1,\ \mathrm{NOT}(\mathrm{NOT}\ X),\cdots$$

在进化过程中，这些多余的表达式附加在算法树上，使个体变得臃肿，但对输出结果没有影响。在遗传规划中，把这种多余的表达式也称作基因内区，或称作遗传规划的基因内区。

据统计，在遗传规划运行的初期和中期，群体中的基因内区可占全体代码的 40%～60%，随后仍可继续增长。图 3-34 表示某遗传规划运行过程中基因内区的统计结果。图中横坐标是累计的个体数，随着遗传规划的运行累计个体数不断增大；纵坐标表示代码的组成，其中黑线表示基因内区，空白部分表示有效的代码。从图中可明确看出，基因内区在早期呈现指数型快速增长。

3.4.3.2　基因内区的分析

为了说明基因内区的形成和作用，要从程序的复杂性说起。在计算机程序设计中，程序的复杂性可用程序的字节数、指令条数或算法树的结点数衡量(图 3-35)。为方便起见，我们不妨先用节点数目表示某一个体或个体内某一块段(程序段)的复杂度。

进一步，我们将个体内程序的复杂度分成两部分：绝对复杂

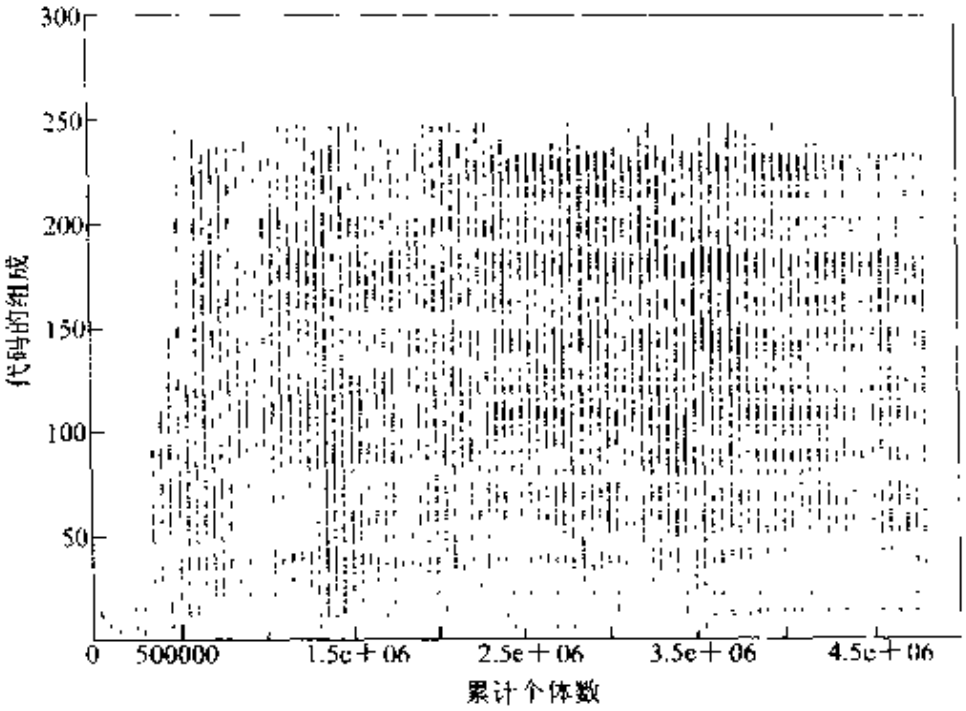

图 3-34　基因内区的分布

度及有效复杂度。绝对复杂度是指个体或个体内某一块段的结点总数，记作 C^a；有效复杂度是指个体或个体内某一块段中有效结点的数目，记作 C^e。很明显，C^a 与 C^e 之差便代表基因内区所包含的结点数。

仿照模式定理，针对复制和交换算子，有下式：

$$P_j^{t+1} \approx P_j^t \cdot \frac{f_j}{\bar{f}^t} \cdot \left\{1 - p_c \cdot \frac{C_j^e}{C_j^a} \cdot p_j^d\right\} \tag{3-16}$$

式中　P_j^{t+1} —— 个体或块段 j 在第 $t+1$ 代的出现概率；

f_j —— 个体或块段 j 的适应度；

$\bar{f}^t$ —— 个体或块段 j 在第 t 代时的平均适应度；

p_c —— 个体发生交换的概率；

C_j^e —— 个体或块段 j 的有效复杂度；

C_j^a —— 个体或块段 j 的绝对复杂度；

p_j^d —— 个体或块段 j 发生破坏性交换的概率。它使个体适应度恶化，但当个体全为基因内区时，此值为零。

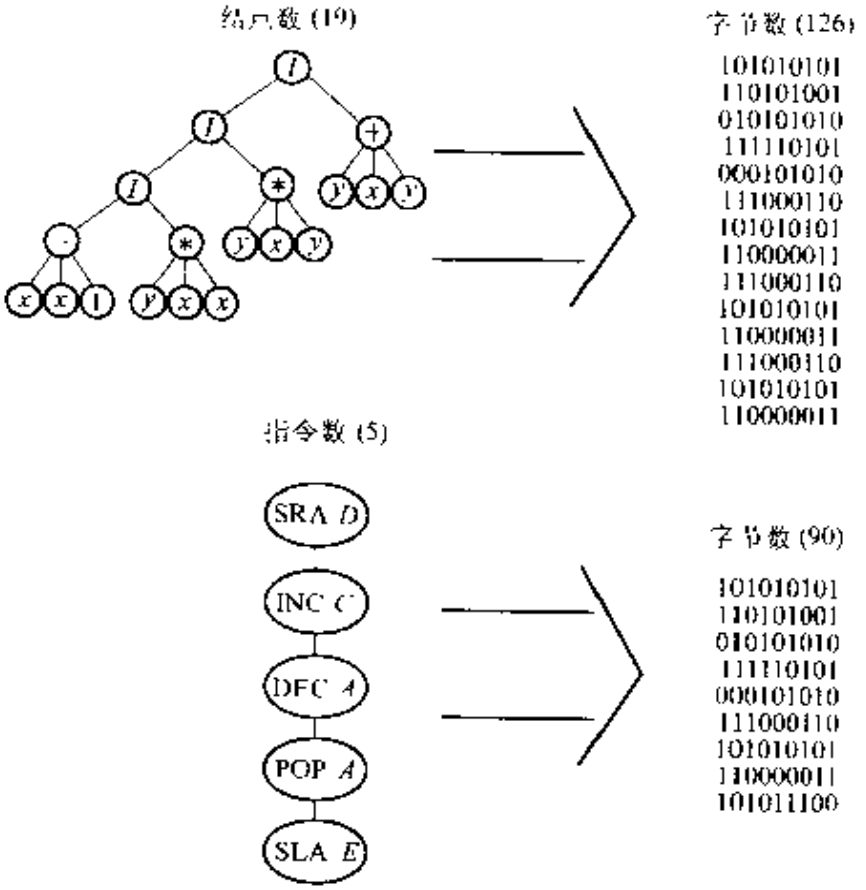

图 3-35　程序复杂性

上式说明，个体或块段 j 出现的概率取决于复制的作用。该式可改写作：

$$P_j^{t+1} \approx \left(\frac{f_j}{f^t} - p_c \cdot \frac{f_j}{f^t} \cdot p_j^d \cdot \frac{C_j^e / C_j^u}{f^t} \right) \cdot P_j^t \qquad (3\text{-}17)$$

若令：

$$f_j^e = f_j - p_c \cdot f_j \cdot \frac{C_j^e}{C_j^u} \cdot p_j^d \qquad (3\text{-}18)$$

(3-17)式变成：

$$P_j^{t+1} \approx \frac{f_j^e}{f^t} \cdot P_j^t \qquad (3\text{-}19)$$

上式说明个体或块段 j 出现的概率取决于 f_j^e / f^t 的比值，这如同轮盘法随机复制的效果。因此称 f_j^e 为个体或块段 j 的有效适应度。从

式(3-18)可以看出，欲增加个体或块段 j 的有效适应度 f_j，可增大 j 的绝对复杂度 C_j，或是减小 j 的有效复杂度 C_j^*。这两种措施都导致增加基因内区的数目。这也正是基因内区在遗传规划中的有益作用，它可改善个体的素质。

基因内区的有益作用可从下述例子中形象地看出。图 3-36 表示没有基因内区的一个“有效”个体，每个结点 E 对个体适应度都有积极作用。该个体肯定(100%)被交换所破坏。图 3-37 是添加 14 个基因内区的新个体，图中 I 表示基因内区结点，对个体适应度毫无作用。很明显，添加基因内区后个体被交换破坏的概率仅为 4/16，即 25%。实际观测也表明，遗传规划运行过程中，75%的交换会起破坏作用。当基因内区大量添加后，交换变成中性交换，不影响个体的适应度。

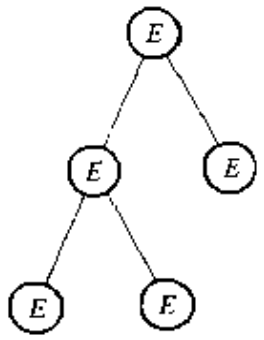

图 3-36 “纯有效”个体

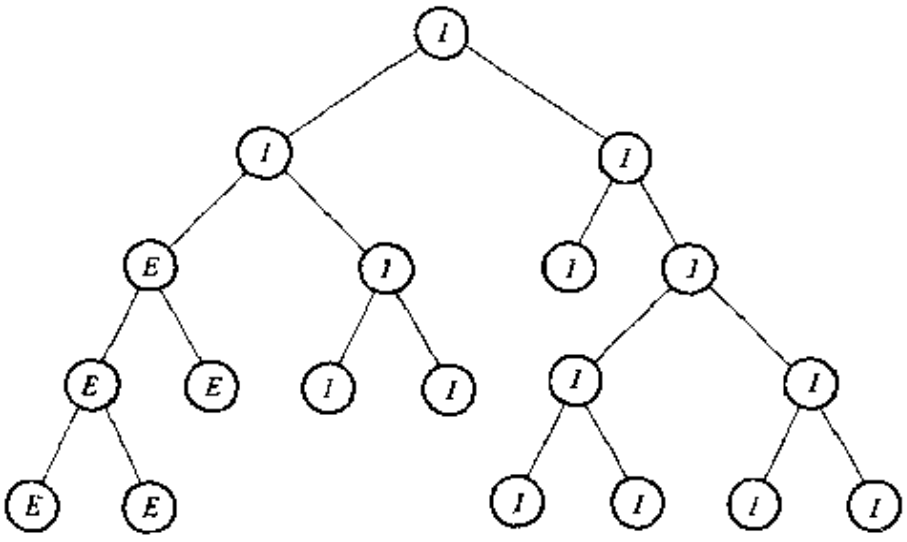

图 3-37 添加基因内区的个体

将公式(3-18)推广用于其他算子，可改写作：

$$f_j^e = f_j\left[1 - \frac{1}{C_j^a}\sum(p_r \cdot p_j^{r,d} \cdot (C_j^{r,r}))\right] \tag{3-20}$$

式中 r —— 算子，包括交换、突变等；

p_r —— 采用 r 算子概率；

$p_j^{r,d}$ —— 个体或块段 j 采用 r 算子的破坏概率；

$C_j^{r,r}$ —— 个体或块段 j 采用 r 算子后的有效复杂度。

将上式进一步改写，有：

$$f_j^e = f_j \cdot [1 - \sum(p_j^{p,r} \cdot (1 - (C_j^{r,r}/C_j^a)))] \tag{3-21}$$

式中 $p_j^{p,r}$ —— r 算子的应用概率及破坏概率的综合；

$C_j^{r,r}$ —— r 算子作用后所产生基因内区的复杂度。

从上式可以看出，若增加基因内区的复杂度 $C_j^{r,r}$，个体或块段 j 的有效适应度 f_j^e 会得到增加。又由于基因内区的复杂度总是小于个体 j 的复杂度，因此 $C_j^{r,r}/C_j^a$ 的比值总是小于 1，这说明基因内区可以不断地扩充。

应该指出，个体的适应度 f_j 与有效适应度 f_j^e 的增长过程是不同的：f_j 是有界的，它的最大值便是问题的最优解；f_j^e 是无界的，它可随基因内区而增大。

3.4.3.3 基因内区的作用

在遗传规划中，基因内区既有积极作用，也有消极作用，其积极作用表现在：

(1) 保护作用。基因内区的存在，扩大了个体的规模，可以引导交换或突变在基因内区内发生，不影响构造块及个体的适应度，从而缓解交换和突变的破坏作用。

保护作用可细分为两类：结构性保护和全局性保护。结构性保护是对构造块有效代码的保护，避免有用块段被交换或突变破坏。全局性保护是对整个个体的保护，使个体能够承受交换或突变的破坏作用，继续向良好个体的方向发展。

(2) 节俭作用(Parsimony)。由于基因内区的存在，促使个体的有效部分更加简洁精悍，这称作节俭作用或压缩作用。为了更好说明基因内区和节俭作用的关系，现将式(3-20)改写作：

$$f' = f \cdot [1 - p^D \cdot (C^e/C^a)] \qquad (3\ 22)$$

上式略去 j 和 r 的标识。从上式可以看出，假若有效复杂度 C^e 等于绝对复杂度 C^a，在个体中便没有基因内区，这时交换对个体的破坏作用最大。另一方面，假若个体中存在基因内区，C^e 不等于 C^a，这时减小 C^e 会增大有效适应度 f'，而 C^e 的减小正是基因内区的节俭作用。

由于基因内区的节俭作用，促使个体有效部分更加短小精悍，可以避开交换或突变的破坏作用，从而提高有用部分的抗干扰能力。

基因内区的消极作用表现在：

(1) 进化停滞。若基因内区大量充斥在群体中，交换操作往往只发生在基因内区，不能改善个体的素质，从而使遗传规划的进化过程处于停滞。

为了打破这种停滞，有人建议提高突变概率，破坏基因内区，以便减少基因内区的规模及数量。

(2) 结果欠佳。许多研究表明，基因内区会导致最优个体的适应度欠佳，这也是进化停滞的另一种表现。由于进化操作无意义地消耗在基因内区内，使群体的素质未能改善，因此最优个体的适应度也欠佳。

(3)计算臃余。基因内区的存在，也使计算机的存贮空间及计算时间白白浪费在对基因内区的无效处理上。

总之，基因内区既有积极作用，又有消极作用。为此，在实际操作中应该扬长避短，正确利用基因内区，其中比较主要的措施是：

(1) 充分利用节俭作用。由于短小的个体更能承受交换的破坏作用，在评价个体优劣时要兼顾个体的复杂度。对于复杂度小的个体，其适应度要适当增大；反之，对于复杂度大的个体要对其适应度施加惩罚。

图 3-38 和图 3 39 分别表示施加节俭压力前后的个体复杂度。图中横坐标表示迭代次数，纵坐标用字节长度表示个体的复杂度。图 3-38 是普通遗传规划的结果，个体的复杂度不断增大。图 3-

39 是将适应度的大小与个体复杂性连系在一起，鼓励精悍个体的进化。图中个体的复杂度得到控制。

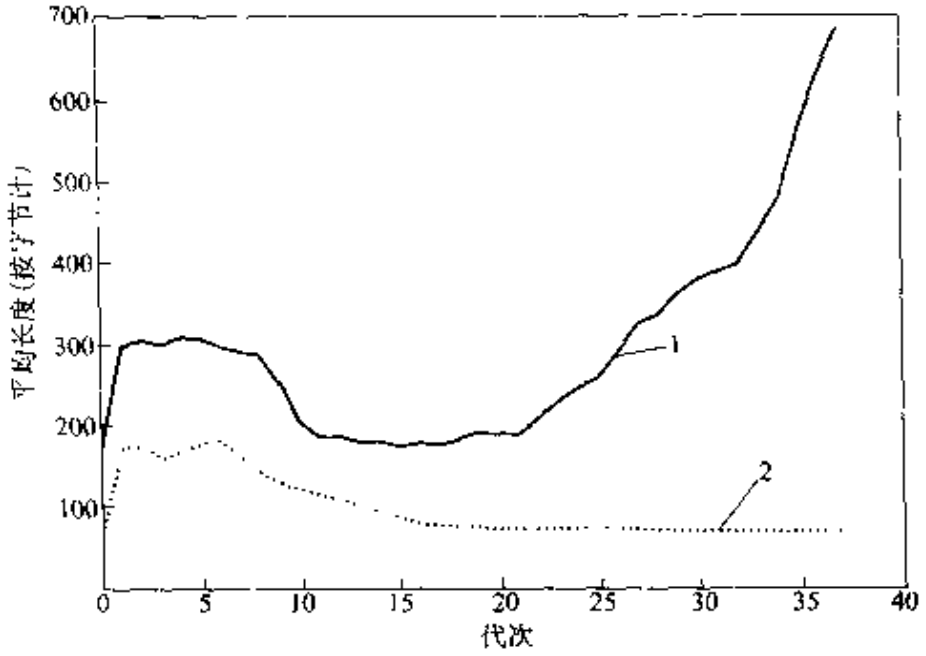

图 3-38　普通的遗传规划

1- 绝对长度；2—有效长度

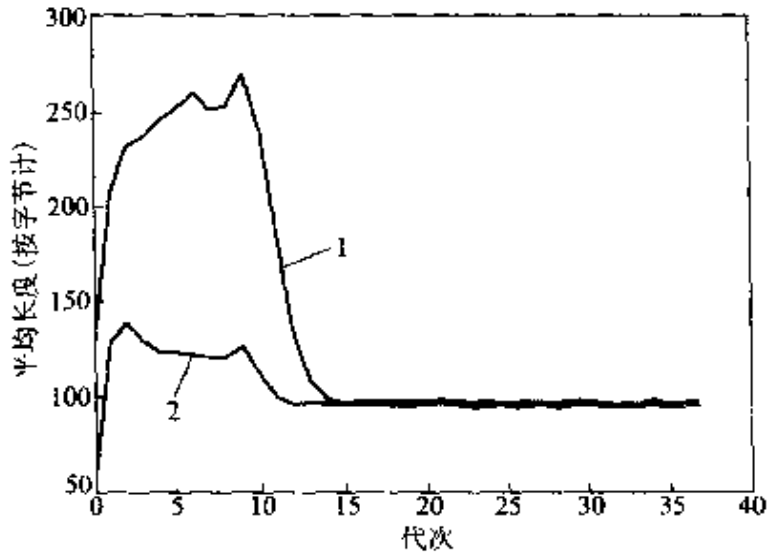

图 3-39　施加节俭的遗传规划

1—绝对长度；2—有效长度

（2）改造适应度方程。使适应度的计算和个体的基因内区联系起来，防止基因内区无休止地任意泛滥。例如，采用可变的或非

线性的适应度函数，促使遗传规划努力搜寻更优良个体，避免进化停滞。

(3) 减少交换的破坏作用。式(3.20)表明，减小交换的破坏作用 p_j^d，在相同的有效适应度 f_j 条件下，可加大个体 j 的有效适应度 f_j'，亦即减少基因内区或推迟基因内区的发展。

3.5 遗传规划的新进展

遗传规划由于其灵活的表达形式，很快得到广泛的应用。在具体实践中又有许多发展，本节介绍比较突出的一些进展。

3.5.1 程序结构

遗传规划的个体由函数和终止符两部分组成。为了将这两部分有机地组成可执行的计算机程序，其结构有三：树状结构、线状结构及网状结构。

3.5.1.1 树状结构

这是最常用的结构型式，它以树状形式表示遗传规划的层次结构，前面已多次使用。关于算法树中各结点的执行顺序，有两种方法：

(1) 后缀法。这种方法从算法树的最左侧结点开始，自左而右，自下而上逐个子树执行各结点。这是最常用的一种方法，因为它先读出终止符(值)，然后才执行函数(算子)，和人们的常规习惯一致。

如图 3.40 所示的算法树，其执行顺序是：

$$d \to e \to \mathrm{OR} \to a \to b \to c \to + \to * \to$$

其表达式为：

$$(d \vee e) - (a * (b + c))$$

(2)前缀法。这种方法的执行顺序恰好相反，它是从算法树的根结点开始，自上而下，自左而右逐个子树顺序执行。

按前缀法，图 3.40 算法树的执行顺序是：

$$\to \mathrm{OR} \to d \to e \to * \to a \to + \to b \to c$$

前缀法的优点是对于含有 IF/THEN 结点的分支树，THEN

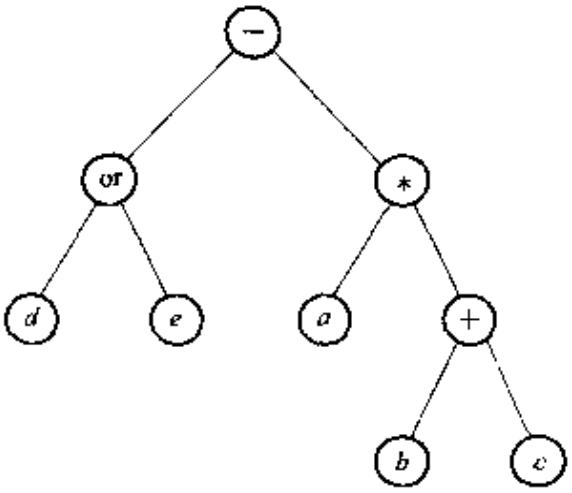

图 3-40　算法树结构

结点下层的支树只有满足 IF 支树后才执行，从而不会因盲目执行 THEN 语言而浪费计算时间。

对于树状结构，执行时采用局部存贮，该存贮专门为树结点建立。例如在图 3-40 中，b 和 c 隶属于结点“+”，因此 b 和 c 的数值不供算法树的其他结点使用。

3.5.1.2　线状结构

线状结构以链状形式表示遗传规划的层状结构，如图 3-41 所示。

图 3-41 的表达结果等同于图 3-40 的算法树，所不同的是线状结构中没有为函数提供直接的输入途径。例如，线状结构中的方框(结点)仅仅表示一个函数方程，例如第一个方框为 $b = b + c$，至于函数方程的值 b 及 c 要另外提供。

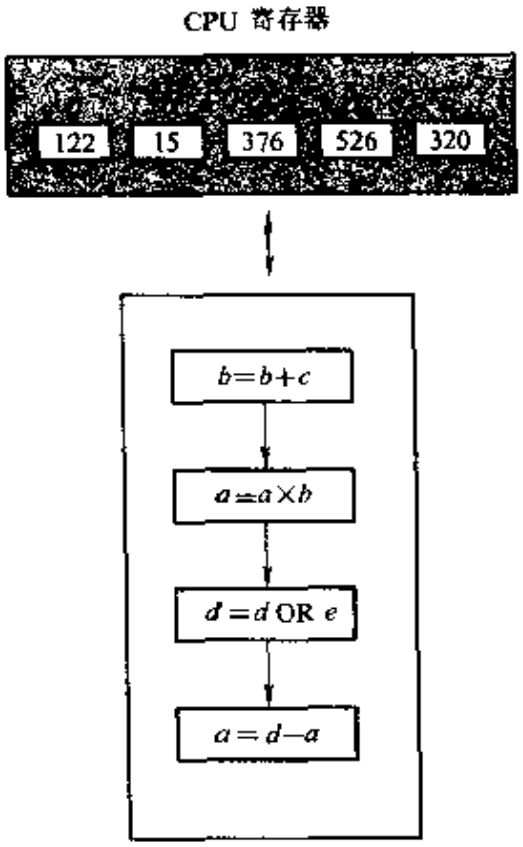

图 3-41　线状结构

在遗传规划中，有多种数

据提供方式，其中最常用的是寄存器，因为计算机的CPU中的存贮寄存器也按线状指令执行。如图3-41，第一条指令是$b=b+c$，于是计算机从寄存器b和c取值相加，其和再放入寄存器b中。

遗传规划的线状结构，通常是自上而下逐条执行，仅仅当遇到转移、跳跃等指令时才偏离这种顺序，因此线状结构在执行顺序上更加灵活多变。执行的结果体现在最后的指令中，如图3-41其结果存于寄存器a。

线状结构与树状结构的另一差别是数据存贮的性质。如前所述，树状结构的数据是局部存贮，从属于具体的结点；而线状结构是全局性数据，寄存器的数据供整个遗传规划使用。例如，图3-41中寄存器a的值就是随着各条指令的执行而变化。

3.5.1.3 网状结构

网状结构用结点和弧组成的复杂有向图表示遗传规划。从数学上讲，网是树的推广，因此网状结构可表示复杂的问题，包括循环及递归等结构。

图3-42表示网状结构。计算机首先执行Start结点，终止于End结点或其他预先设定的结点，中间的执行过程由图中的有向弧指挥。

网状结构的数据存放在堆栈中，每个结点从(往)堆栈读(写)数据。例如，图3-42的结点A从RAM中读取数据，然后放入堆栈。结点6则将数值6放入堆栈。结点MUL(乘)从堆栈读取两个数值，相乘后再放回堆栈。因此，网状结构的数据属于局部存贮。

网状结构的数据还可以采用索引存贮。例如，图3-42的结点Write从堆栈中取出两个数值，然后将第一个数值写入按第二个数值标记的索引存贮器中。结点Read的功能恰好与Write相反因此，索引存贮属于全局性存贮。

网状结构中每个结点要执行两项作业：

(1) 对堆栈和(或)索引存贮执行某项动作；

(2) 确定下一步要转向哪一个结点。

后一项作业指明遗传规划的执行顺序。例如，图3-42中结点

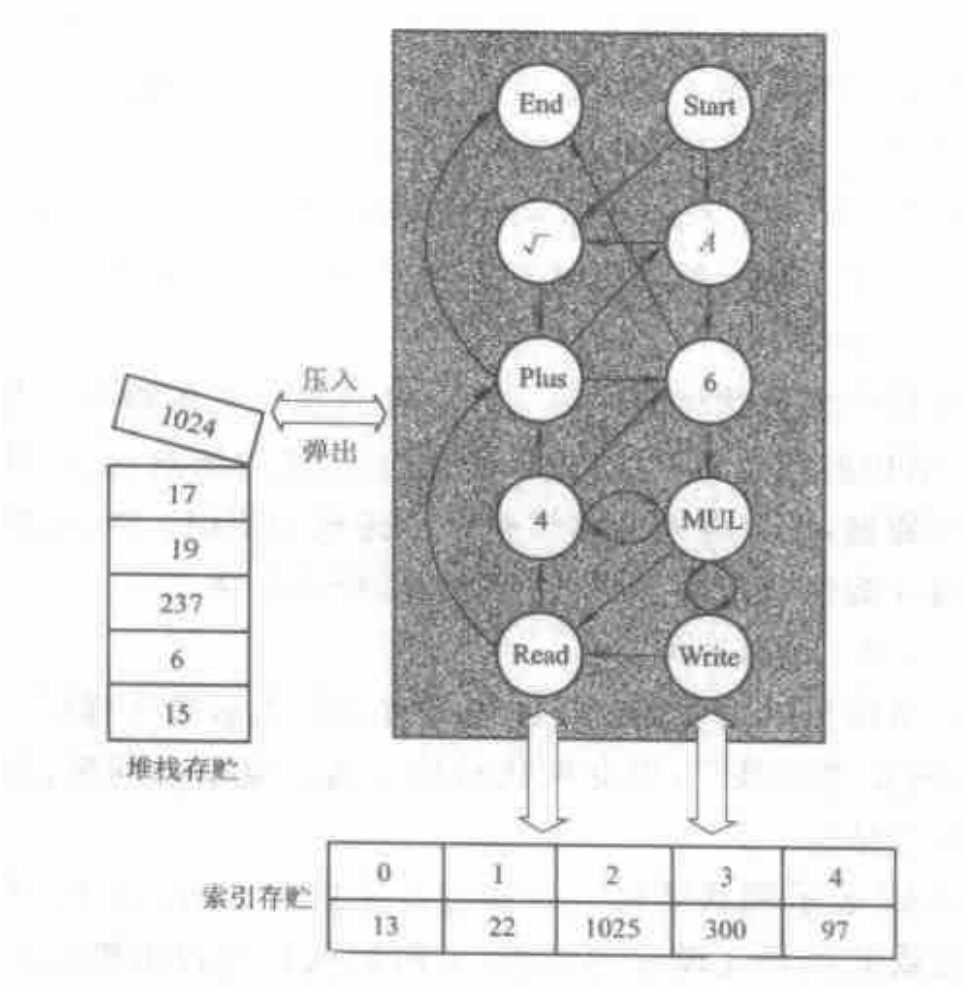

图 3-42　网状结构

MUL(乘),下一步可能执行 Write,或者执行 Read,或是结点 4,具体由系统内部的决策逻辑确定。

3.5.1.4　遗传操作

采用线状结构和网状结构后,其遗传操作也有别于前述树状结构。现以线状结构为例说明交换及突变。

线状结构的交换比树状结构简单。如图 3-43 所示,其工作步骤为:

(1) 按前述的选择原则,随机选取两个父代个体,图中父代个体位于左侧,用 a、b、c… 表示各指令。

(2) 在父代个体中各随机选取一段指令作为交换体,图中涂黑的方框表示交换体。

(3) 对父代个体实行交换，产生两个子代新个体，如图中位于右侧的个体。

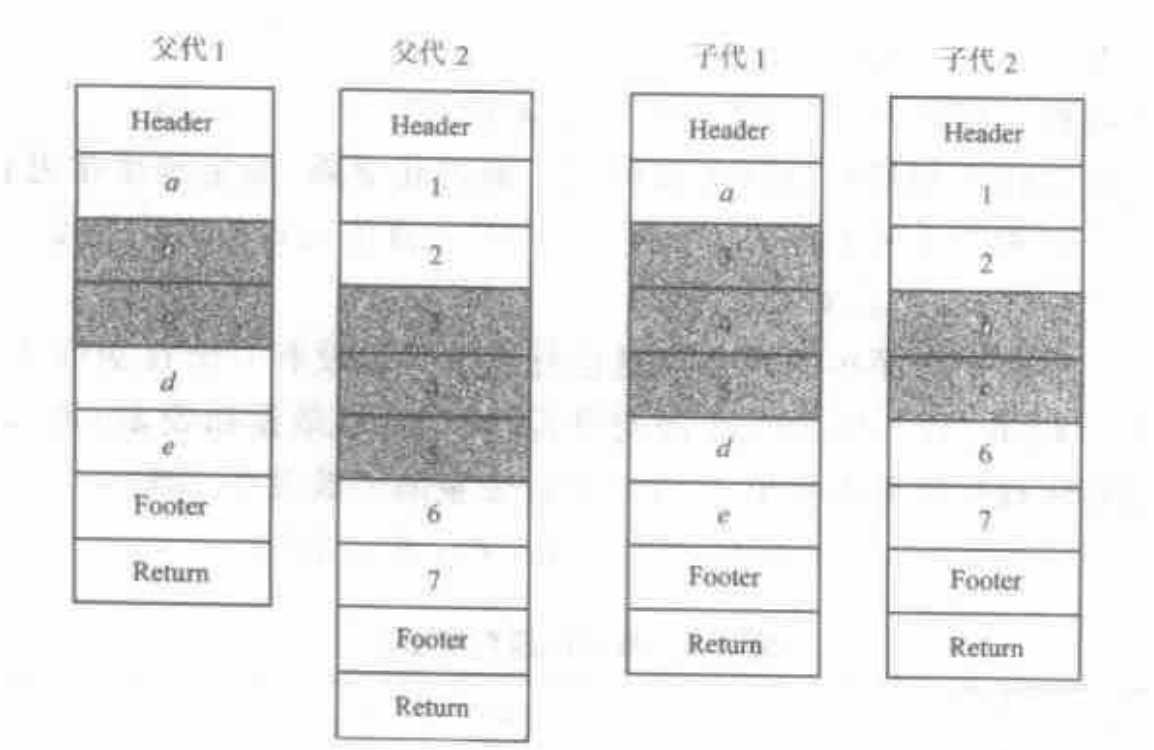

图 3-43 线状结构的交换

线状结构的突变较为复杂。当随机选定突变个体后，再随机选定某条指令(方框)进行突变，具体的突变内容从下述三者中随机选择：

(1) 指令中的指示符可相互随机替换；

(2) 指令中的操作符可从函数集中随机选取；

(3) 指令中的常数可在允许值中随机选取。

例如，若原来的指令为：

$$r_0 = r_1 + r_2 + 3$$

可突变指示符为：

$$r_1 = r_0 + r_1 + 3$$

或突变操作符为：

$$r_0 = r_1 * r_2 + 3$$

或突变常数为：

$$r_0 = r_1 + r_2 + 4$$

3.5.2 自动定义函数

自动定义函数(Automatically Defined Function,简称 ADF)是 Koza 首先提出来的。它类似 FORTRAN 语言的子程序概念,可以在主程序中调用,便于表达复杂的大问题。然而,自动定义函数在遗传规划执行过程中又可以不断进化更新,使主程序在进化过程中调用不断更新的子程序,明显提高遗传规划的进化速度。

3.5.2.1 基本原理

现以一个简单的两盒问题进行说明。假设有 6 个自变量 L_0、W_0、H_0、L_1、W_1、H_1 及 1 个因变量 D,10 次观察数据如表 3-9 所示。现要从观察数据中找出 D 与 6 个自变量的函数关系,即:

$$D = f(L_0, W_0, H_0, L_1, W_1, H_1)$$

表 3-9 两盒问题观测数据

序 号	L_0	W_0	H_0	L_1	W_1	H_1	O
1	3	4	7	2	5	3	54
2	7	10	9	10	3	1	600
3	10	9	4	8	1	6	312
4	3	9	5	1	6	4	111
5	4	3	2	7	6	1	-18
6	3	3	1	9	5	4	−171
7	5	9	9	1	7	6	363
8	1	2	9	3	9	2	−36
9	2	6	8	2	6	10	−24
10	8	1	10	7	5	1	45

若采用传统的遗传规划,最终可找出上述函数关系为两长方体体积之差,即:

$$D = L_0W_0H_0 - L_1W_1H_1$$

为了验算上式的 D 值,可编写一段计算机程序。最简单的方

法是先写一段子程序计算 D，然后，在主程序中针对 10 组自变量数据分别调用子程序，如图 3-14 所示。图中根结点 progn 代表主程序，左侧分支表示子程序，右侧分支表示对子程序的调用。在子程序分支中，结点 defun 对子程序作三方面说明：一为子程序名称，为最左边的结点 VOLUME；二为子程序包括的变量表，即中间结点表示的 ARG0、ARG1 及 ARG2；三为子程序执行后的返回值 values。values 的计算用虚线以下的子树表示，即：

$$\text{value} = \text{ARG}1 * (\text{ARG}1 * \text{ARG}2)$$

在 prog 结点的右侧分支中，结点 values 表示执行虚线以下主程序的返回值，它包括两次调用子程序 VOLUME 并代入不同的 L、W、H 值，然后求两次子程序执行结果之差，即：

$$\text{values} = \text{VOLUME}_0 - \text{VOLUME}_1$$

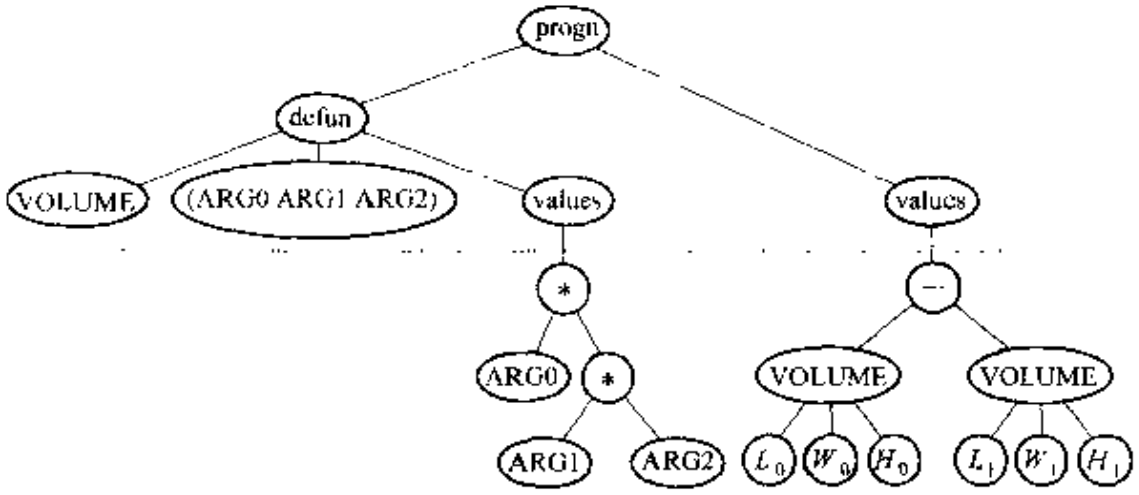

图 3-14　两盒问题的程序结构

上述的"子程序"是 FORTRAN 语言的称呼，在 LISP 语言中则称作函数。把这种概念推广到遗传规划中，使子程序也可以自动进化更新，这便是自动定义函数，如图 3-15 所示。图中左分支描述自动定义函数(结点 2)，规定自动定义函数的名称为 ADF0(结点 3)，列举函数的变量表 Argument List(结点 4)，并说明 ADF0 的返回值 values(结点 5)是如何用虚线以下的函数体(结点 7)求算。右分支是产生结果分支，在结点 8 中反复调用自动定义函数，往结

点 6 返回结果。在自动定义函数中，虚线以上的 1～6 结点保持不变，虚线以下的结点 7 和 8 经常进化更新，它们是两分支的主体部分。简而言之，带有自动定义函数的遗传规划，在不断进化更新的主程序(结点 8)中又包含也在不断进化更新的子程序(结点 7)。

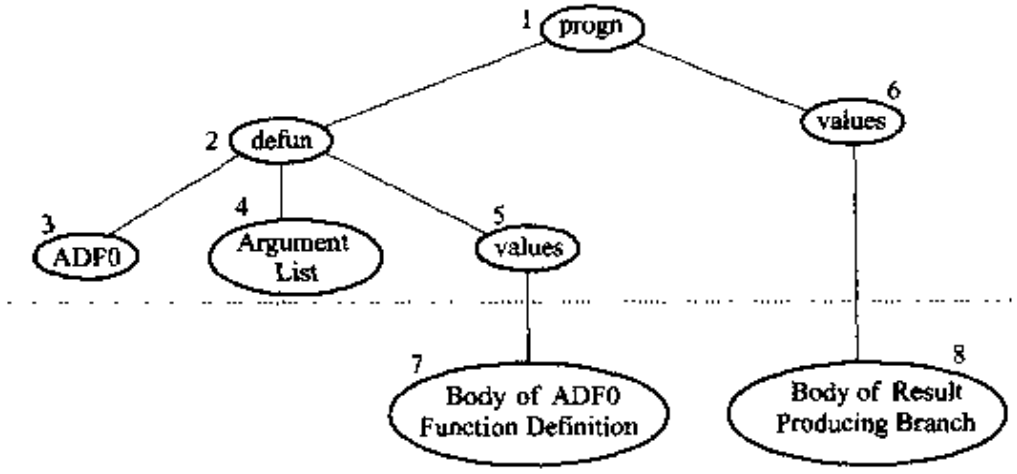

图 3-45　自动定义函数的结构

在遗传规划中，可以有多个自动定义函数，每一个自动定义函数又可以引用已定义过的其它自动定义函数，或者递归调用自己。

总之，自动定义函数的算法树，其结点可分为下列 8 类(见图 3-45)：

(1) 根结点(progn)，它返回总表达式的值；

(2) 自动定义函数的根结点(defun)，其下属分支描述自动定义函数；

(3) 自动定义函数的函数名(ADF0)；

(4) 自动定义函数的变量表(Argument List)；

(5) 自动定义函数的返回值(values)，它返回结点 7 的计算结果；

(6) 结果分支的返回值(values)，它返回结点 8 的计算结果；

(7) 自动定义函数的函数体部分(Body of ADF0 Function Definifion)，它本身又是一棵可变动的算法树；

(8) 结果分支的主体部分(Body of Result Producing Branch),它本身也是一棵调用自动定义函数的可变动的算法树。

3.5.2.2 结构保护操作

遗传规划中的交换及突变,使个体(树)的结构会产生巨大变化。当使用自动定义函数时,为了使遗传操作后的个体(树)不破坏自动定义函数的句法结构,必须对遗传操作采取一些限制,使新产生的个体仍然有效。这种限制,就称为结构保护(Structure Preserving)。

采用结构保护时,自动定义函数的前6类结点(图3-45虚线以上结点)保持不动,仅允许变动虚线以下的第7类、第8类结点。

现以结构保护的交换为例说明。如图3-46所示,执行交换时,虚线以上的第1～6结点保持不变,只能交换虚线以下的5个第7类结点及6个第8类结点,而且交换时不破坏算法树的结点类型。为此,第一个父代个体的交换点可在虚线以下的第7类或第8类结点中任意选择。一旦交换点选定后,第二个个体的交换点只能选择与第一个个体交换点类型相同的点。例如,图中假设第一个个体随机选中第7类结点之"+",由于它属于自动定义函数分支,那么第二个个体的交换点也只能从其自动定义函数分支(即第7类结点)中选取,不允许从产生结果分支的第8类点中选取。同样,若第一个个体的交换点属于产生结果分支的第8类结点,则第二个个体的交换点也只能是第8类结点。总之,结构保护的交换有两种限制:一是限制交换点只能是虚线以下的第7类、第8类结点;二是限制第二个父代个体的交换点类型要等同于第一个父代个体。正是由于这种限制,才能保证产生的子代新个体是有效、合法的。

对于突变,自动定义函数也需要结构保护。同样,虚线以上的第1～6类结点不允许突变,突变点只能是虚线以下的第7、第8类结点(图3-46)。一旦突变点选定后,还要判别它的类型,假若属于自动定义函数分支的第7类点,则新插入的子树只能从自动定义函数的终止符集和函数集中随机选择元素生成;否则,用产生结果分支的终止符集和函数集的元素随机生成子树。用新生成的子

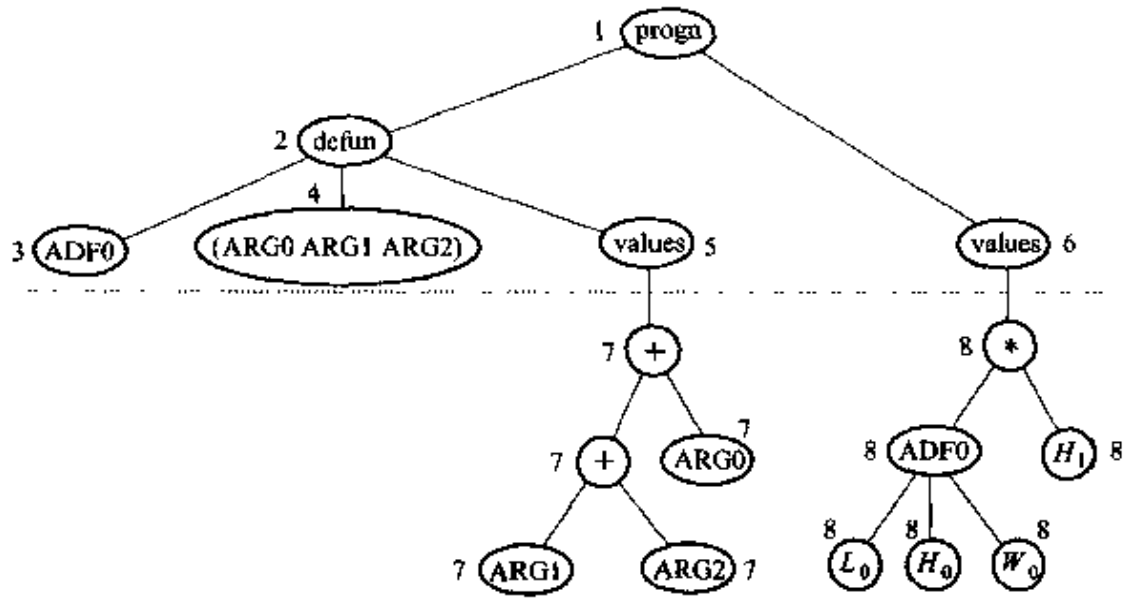

图 3-46　结构保护的交换

树代替以突变点为根的旧子树，使新产生的子代个体符合语法要求。

3.5.2.3　工作流程

带自动定义函数的遗传规划，要额外添加下述工作：

(1)选择自动定义函数的数目；

(2)确定每个自动定义函数的变量表(变量数目)；

(3)若有多个自动定义函数，确定它们之间的相互关系；

(4)确定自动定义函数分支及生成结果分支的函数集及终止符集；

(5)确定适应度计算方法、遗传计算参数及终止准则。

显然，引用自动定义函数后，遗传规划变得很复杂，计算时间也增长。因此，自动定义函数主要用于结构很复杂的大型问题中，此时利用自动定义函数可简化问题的表达。

3.5.3　模块类算子

为了保存个体优良部分，遗传规划仿照传统计算机程序设计中的模块化原则，也提出一些新的算子，以提高进化速度。严格地讲，上述自动定义函数也属于模块类算子。

3.5.3.1　封装

封装(Encapsulation)的目的是把个体中的有用部分保存下来，以便今后作为一个整体予以调用。

封装的工作步骤如下：

(1)从群体中选择一个个体；

(2)在该个体中选择一个内结点(函数)；

(3)将内结点下属的分支树移出，视作一个叶结点，并命名；

(4)将新命名的叶结点作为一个独立终止符使用。

如图 3-47，上方的个体代表下述表达式：

$$(x-1)^2 \mid (x-1)^4$$

若采用封装，得新的叶结点：

$$E_0 = x - 1$$

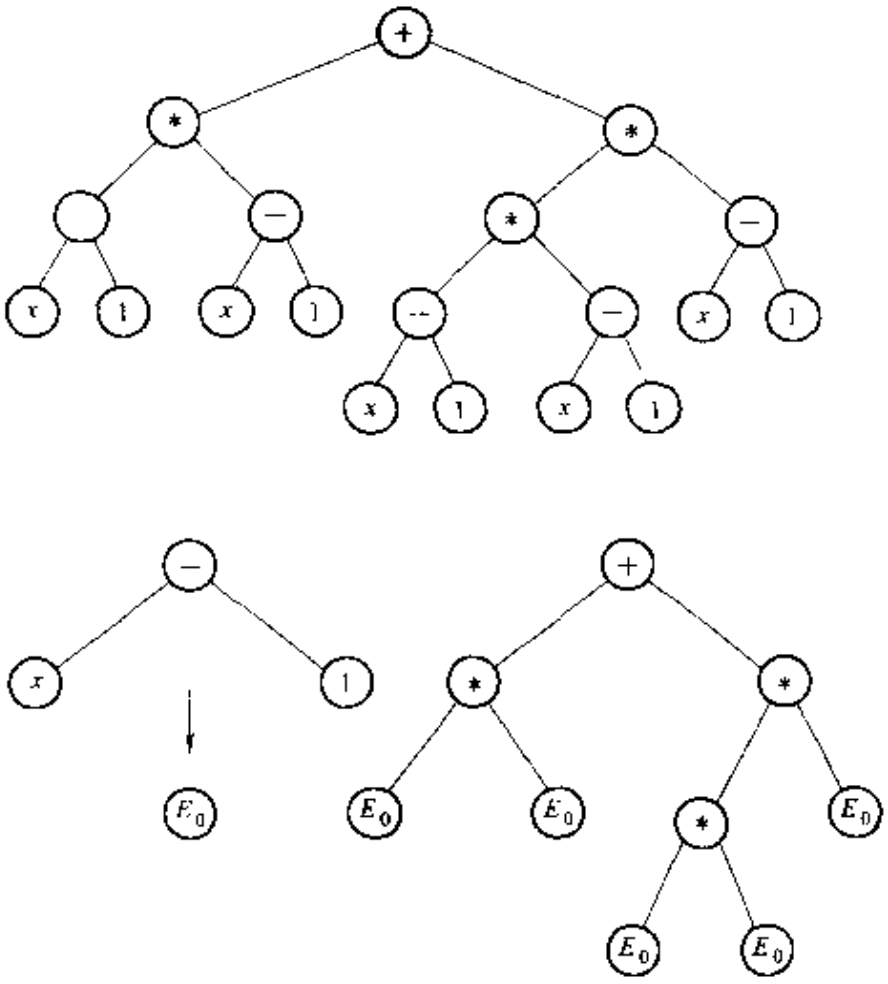

图 3-47　封装

则原个体可简化为图中右下方的算法树。

封装的作用在于保护被封装的子树，使它在交换操作中不被瓦解。事实上，被封装的子树通常应是一个潜在构造块，可供后代繁衍生息。

3.5.3.2 模块化

模块化(Module Aquisition)类似于封装，但是模块化只封装所游离出的子树的一部分而不是全体，而且被封装部分要置于实体库中调用。

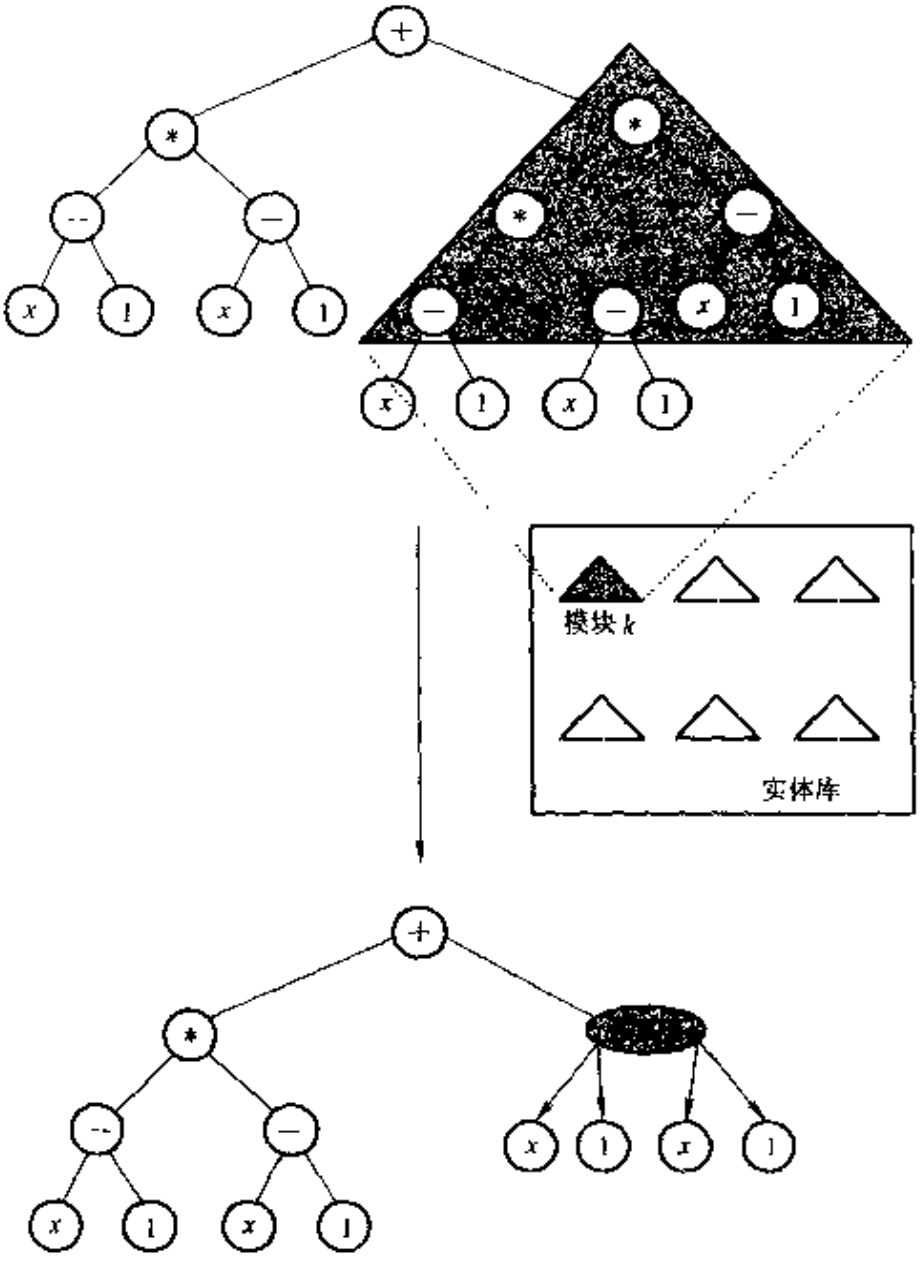

图 3-48 模块化

如图 3.48 所示，模块化的工作过程如下：

(1)从群体中选择一个个体(图中上方的算法树)；

(2)从该个体中选择一个内结点(图中为"*")；

(3)将该结点下属子树在某一规定深度内的三角形部分，视作一个模块，并命名(图中用阴影线标记，并命名为模块 k)；

(4)将该模块存入模块库中；

(5)调用该模块(图中用模块 k 重新表达原算法树)。

模块化的作用类似于封装，它将个体中的有价值部分保护起来，避免交换操作破坏。模块化的参数是子树划归模块的最大深度，但有时用两个参数：子树划归模块的最小深度及最大深度，从而使模块呈梯状。

4 进化策略

4.1 进化策略的基本原理

4.1.1 (1+1)—ES

1963 年，德国柏林技术大学的 I.Rechenherg 和 H.P.Schwefel 为了研究风洞中的流体力学问题，提出进化策略(Evolution Strategies，简称 ES)。当时提出的这种优化方法只有一个个体，并由此衍生同样仅为一个的下一代新个体，故称为(1+1)—ES。

进化策略中的个体用传统的十进制实型数表示，即：

$$X^{t+1} = X^t + N(0,\sigma) \tag{4-1}$$

式中 X^t —— 第 t 代个体的数值；

$N(0,\sigma)$ —— 服从正态分布的随机数，其均值为零，标准差为 σ。

因此，进化策略中的个体含有两个变量，为二元组 $\langle X,\sigma\rangle$。新个体的 X^{t+1} 是在旧个体 X^t 的基础上添加一个独立随机变量 $N(0,\sigma)$。假若新个体的适应度优于旧个体，则用新个体代替旧个体；否则，舍弃性能欠佳的新个体，重新产生下一代新个体。在进化策略中，个体这种进化方式称作突变。很明显，突变产生的新个体与旧个体性态差别不大，这符合生物进化的基本状况：生物的微小变化多于急剧变化。

下面仍用 2.1 节遗传算法的实例说明(1+1)—ES 的执行过程。设目标函数为：

$$\max f(x_1,x_2) = 21.5 + x_1 \cdot \sin(4\pi x_1) + x_2 \cdot \sin(20\pi x_2)$$

约束条件为：

$$-3.0 \leqslant x_1 \leqslant 12.1$$

$$4.1 \leqslant x_2 \leqslant 5.8$$

在此优化问题中，$X = (x_1,x_2)$，令 $\sigma = (\sigma_1,\sigma_2)$，即：

$$x_1^{t+1} = x_1^t + N_1(0,\sigma_1)$$
$$x_2^{t+1} = x_2^t + N_2(0,\sigma_2)$$

假设在第 t 代，有：

$$(X^t,\sigma) = ((5.3,\ 4.9),(1.0,\ 1.0))$$

则突变后的新个体为：

$$x_1^{t+1} = x_1^t + N(0,1.0) = 5.3 + 0.4 = 5.7$$
$$x_2^{t+1} = x_2^t + N_2(0,1.0) = 4.9 - 0.3 = 4.6$$

式中 $N(0,1.0)$ 产生两个服从正态分布的随机数 0.4 及 -0.3。

由于

$$f^{(t)} = f(5.3,4.9) = 18.383705$$
$$f^{(t+1)} = f(5.7,4.6) = 24.849532 > f^{(t)}$$

因此 x_1^{t+1} 及 x_2^{t+1} 被接纳，用新个体 X^{t+1} 代替父代个体 X^t。

为了控制收敛速度，Rechenberg 还提出著名的“1/5 成功定律”，他用 φ 表示突变次数中成功突变（新个体被采纳）的比率。他认为 φ 应为 1/5，若 φ 大于 1/5，适当加大 σ；反之，减少 σ，即：

$$\sigma^{t+1} = \begin{cases} C_d \cdot \sigma & \text{若 } \varphi < 1/5 \\ \sigma & \text{若 } \varphi = 1/5 \\ C_i \cdot \sigma & \text{若 } \varphi > 1/5 \end{cases} \tag{4-2}$$

式中 φ—— 经历 k 次迭代后突变成功的次数与总突变次数之比，通常 $k > 10$；

C_d —— 小于 1 的系数；

C_i —— 大于 1 的系数。

(1+1)—ES 仅仅使用一个个体，进化操作只有突变一种，亦即用独立的随机变量修正旧个体，以求提高个体素质。显然，这是最简单的进化策略。

4.1.2 $(\mu+1)$—ES

早期的(1+1)—ES，没有体现群体的作用，只是单个个体在进化，具有明显的局限性。随后，Rechenberg 又提出$(\mu+1)$—ES，在这种进化策略中，父代有 μ 个个体($\mu>1$)，并且引入重组(Recombination)算子，使父代个体组合出新的个体。在执行重组时，

从 μ 个父代个体中用随机的方法任选两个个体：

$$(X^1, \sigma^1) = ((x_1^1, x_2^1, \cdots, x_n^1), (\sigma_1^1, \sigma_2^1, \cdots, \sigma_n^1))$$

$$(X^2, \sigma^2) = ((x_1^2, x_2^2, \cdots, x_n^2), (\sigma_1^2, \sigma_2^2, \cdots, \sigma_n^2))$$

然后从这两个个体中组合出如下新个体：

$$(X, \sigma) = ((x_1^{q_1}, x_2^{q_2}, \cdots, x_n^{q_n}), (\sigma_1^{q_1}, \sigma_2^{q_2}, \cdots, \sigma_n^{q_n}))$$

式中 $q_i = 1$ 或 2，它以相同的概率针对 $i = 1, 2, \cdots\cdots n$ 随机选取。

对重组产生的新个体执行突变操作，突变方式及 σ 的调整与 (1+1)- ES 相同。

将突变后的个体与父代 μ 个个体相比较，若优于父代最差个体，则代替后者成为下一代 μ 个个体新成员；否则，重新执行重组和突变产生另一新个体。

$(\mu+1)$-ES 和(1+1)-ES 具有相同的策略：只产生一个新个体。$(\mu+1)$-ES 的特点在于：

(1) 采用群体，其中包含 μ 个个体；

(2) 增添重组算子，它相当于遗传算法中的交换算子，从父代继承信息构成新个体。

显然，$(\mu+1)$-ES 比(1+1)-ES 有了明显的改进，为进化策略这种新的进化算法奠下良好的基础 。

4.1.3 $(\mu+\lambda)$-ES 及(μ,λ)-ES

1975 年，Schwefel 首先提出$(\mu+\lambda)$-ES，随后又提出(μ,λ)-ES。这两种进化策略都采用含有 μ 个个体的父代群体，并通过重组和突变产生 λ 个新个体。它们的差别仅仅在于下一代群体的组成上。$(\mu+\lambda)$-ES 是在原有 μ 个个体及新产生的 λ 个新个体中(共 $\mu+\lambda$ 个体)再择优选择 μ 个个体作为下一代群体。(μ,λ)-ES 则是只在新产生的 λ 个新个体中择优选择 μ 个个体作为下一代群体，这时要求 $\lambda > \mu$。总之，在选择子代新个体时若需要根据父代个体的优劣进行取舍，则使用“+”记号，如(1+1)、$(\mu+1)$及$(\mu+\lambda)$；否则，改用逗号分隔，如(μ,λ)。

这两种进化策略中，都采用重组、突变、选择三种算子，其中重组算子类似于 $(\mu+1)$ - ES，而突变算子有了新的发展，标准差 σ

既不是固定的常数，也不是按 1/5 成功规则的确定型变化，而是可自适应地调整，即：

$$\begin{cases}\sigma = \sigma \cdot e^{\Delta\sigma \cdot N(0,\sigma)} \\ X = X + N(0,\sigma)\end{cases} \tag{4-3}$$

式中 (X,σ) —— 父代个体；

(X',σ') —— 子代新个体；

$\Delta\sigma$ —— 参数；

$N(0,\sigma')$ —— 独立的服从正态分布的随机变量，其均值为 0，标准差为 σ'。

近年来，(μ,λ) - ES 得到广泛的应用，这是由于这种进化策略使每个个体的寿命只有一代，更新进化很快，特别适合于目标函数有噪声干扰或优化程度明显受迭代次数影响的课题。因此，本章着重论述 (μ,λ) - ES。

4.2 进化策略的基本技术

4.2.1 问题的表达

进化策略不同于遗传算法，采用传统的十进制实数表达问题。为了与突变操作相适应，进化策略有两种表达方式。

(1) 二元表达方式。这种表达方式中个体由目标变量 X 和标准差 σ 两部分组成，每部分又可以有 n 个分量，即：

$$(X,\sigma) = ((x_1, x_2, \cdots, x_i, \cdots, x_n), (\sigma_1, \sigma_2, \cdots, \sigma_i, \cdots, \sigma_n))$$

X 和 σ 之间的关系是：

$$\begin{cases}\sigma_i' = \sigma_i \cdot \exp(\tau \cdot N(0,1) + \tau \cdot N_i(0,1)) \\ x_i' = x_i + \sigma_i' \cdot N_i(0,1)\end{cases} \tag{4-4}$$

式中 (x_i,σ_i) —— 父代个体的第 i 个分量；

(x_i',σ_i) —— 子代新个体的第 i 个分量；

$N(0,1)$ —— 服从标准正态分布的随机数；

$N_i(0,1)$ —— 针对第 i 分量重新产生一次符合标准正态分布的随机数；

τ —— 全局系数，常取 1；

τ—— 局部系数，常取 1。

上式表明，新个体是在旧个体基础上随机变化而来。

式(4-4)实际上是式(4-3)的具体化，两式的基本原则是一样的。

二元表达方式简单易行，得到广泛的应用。

(2) 三元表达方式。为了改善进化策略的收敛速度，Schwefel在二元表达的基础上引入第三个因子——坐标旋转角度 α。个体的描述扩展为 (X,σ,α)，即：

$$(X,\sigma,\alpha)=((x_1,x_2,\cdots,x_i,\cdots,x_n),(\sigma_1,\sigma_2,\cdots,\sigma_i,\cdots,\sigma_n),$$
$$(\alpha_1,\alpha_2,\cdots,\alpha_j,\cdots,\sigma_m))$$

三者的关系为：

$$\begin{cases}\sigma_i'=\sigma_i\cdot\exp(\tau'\cdot N(0,1)+\tau\cdot N_i(0,1))\\ \alpha_j'=\alpha_j+\beta\cdot N_j(0,1)\\ x_i'=x_i+z_i\end{cases}\tag{4-5}$$

式中 α_i—— 父代个体 i 分量与 j 分量间坐标的旋转角度；

α_j'—— 子代新个体 i 分量与 j 分量间坐标的旋转角度；

β—— 系数，常取 0.0873；

z_i—— 取决于 σ' 及 α' 的正态分布随机数。

其余符号同式(4-4)。

旋转角度 α_j 表面上是分量 i 与 j 间坐标的旋转角度，实质上它是分量 i 与分量 j 之间协方差的体现(详见 4.4.3)。

随机数 z_i 服从正态分布，其数学期望为零，其方差 σ_a^2 取决于突变方差 σ'^2 及旋转角度 α，即：

$$\sigma_a=\left(\prod_{i=1}^{n-1}\prod_{j=i+1}^{n}R(r_{ij})\right)\cdot\sigma'\tag{4-6}$$

式中矩阵 $R(r_{ij})$ 各元素按下式计算：

$$r_{ii}=r_{jj}=\cos\alpha_{ij}$$
$$r_{ij}=-r_{ji}=-\sin\alpha_{ij}\tag{4-7}$$

因此，随机数 z_i 可表示为 $N(0,\sigma_a)$。

图 4-1 表示旋转的作用。该图是针对 $n=2$，即 $X=((x_1,x_2),(\sigma_1,\sigma_2))$ 的简单情况。图中横坐标和纵坐标分别代表 x_1 及 x_2，五个带十字线的圆或椭圆表示五个子代新个体可能出现的等概率分布曲线，其余的细曲线表示目标函数的等值线。最左侧的图表示 $\sigma_1=\sigma_2$ 且没有旋转的情况，子代新个体出现的等概率分布呈圆形。中间的图表示 $\sigma_1\neq\sigma_2$ 且没有旋转的情况，其等概率分布呈椭圆形，说明新个体有可能向有利的方向发展，加快收敛速度。不过这时椭圆的长、短轴的方向都与原来的坐标轴方向一致。右图是 $\sigma_1\neq\sigma_2$ 且存有旋转角 α 的情况，其等概率分布呈椭圆形，而且长、短轴的方向不同于坐标轴，说明新个体的发展更加自由。例如，左上角、右上角及右下角三个个体（椭圆）沿长轴方向发展可横跨多条目标函数等值线，从而加快收敛。

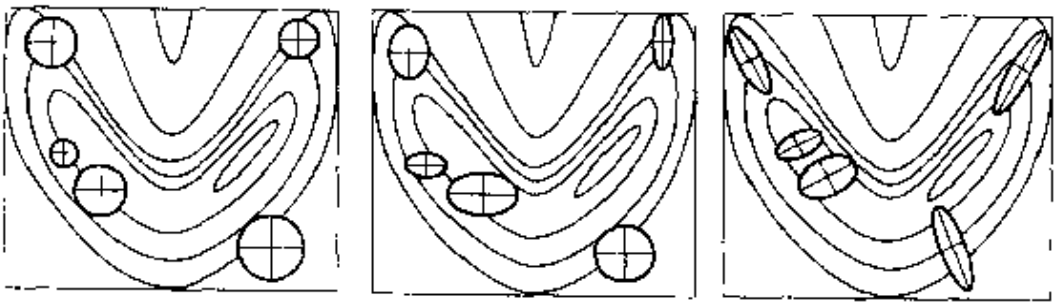

图 4-1　旋转因子的作用

4.2.2　初始群体的产生

进化策略中初始群体由 μ 个个体组成，每个个体 (X,σ,α) 内又可以包含 n 个 x_i、σ_i 分量及 $\frac{n\cdot(n-1)}{2}$ 个 α_j 分量。产生初始个体的方法是随机生成。为了便于和传统的方法比较，可以从某一初始点 $(X(0),\sigma(0),\alpha(0))$ 出发，通过多次突变产生 μ 个初始个体，该初始点从可行域中用随机方法选取。

初始个体的标准差 $\sigma(0)$，可用下式计算：

$$\sigma(0)=\Delta X/\sqrt{n} \tag{4-8}$$

式中　ΔX —— 初始点与最优点的距离；

n ——个体中所含分量个数。

由于 ΔX 在初始时不便确定，可取 $\sigma(0)=3.0$。$\sigma(0)$ 不宜取太大，若 $\sigma(0)$ 太大而且 μ 也大时，选择力度不够，易使群体过于分散。尽管 $\sigma(0)$ 较小，但在进化过程中通过个体的自适应调整仍可使搜索点很快散布在整个可行域内。

4.2.3 适应度计算

适应度是衡量个体优劣的尺度。由于进化策略采用十进制的实数表达问题，因此适应度的计算更加直观、简便。相对于遗传算法和遗传规划，进化策略中的适应度计算更易于执行。

进化策略中对于约束条件的处理，主要是采用重复试凑法。每当新个体生成，将其代入约束条件中检验是否满足约束条件。若满足，则接纳新个体；否则，舍弃该新个体，藉助重组、突变再产生另一个新个体。由于进化策略采用实数编码，这种检验比较直观和简单易行。

应该指出，当采用 $(\mu,\lambda)-ES$ 时，旧群体不参加选择，因此初始个体可以略去适应度计算，直接执行重组、突变等操作，然后再计算新群体的适应度。然而对于 $(\mu+\lambda)-ES$，旧群体参与选择，则需要计算初始群体的适应度。

4.2.4 重组

进化策略中的重组(Recombination)算于相当于遗传算法的交换，它们都是以两个父代个体为基础进行信息交换。进化策略中，重组方式主要有三种：

(1)离散重组。先随机选择两个父代个体

$$\begin{cases}(X^1,\sigma^1)=((x_1^1,x_2^1,\cdots,x_n^1),(\sigma_1^1,\sigma_2^1,\cdots,\sigma_n^1))\\(X^2,\sigma^2)=((x_1^2,x_2^2,\cdots,x_n^2),(\sigma_1^2,\sigma_2^2,\cdots,\sigma_n^2))\end{cases}\tag{4-9}$$

然后将其分量进行随机交换，构成于代新个体的各个分量，从而得出如下新个体：

$$(X,\sigma)=((x_1^{q_1},x_2^{q_1},\cdots,x_n^{q_n}),(\sigma_1^{q_1},\sigma_2^{q_1},\cdots,\sigma_n^{q_n}))$$

式中 $q_i=1$ 或 2。新个体的分量是从两个父代个体随机选取，而且 x_i 分量的 q_i 不一定要等于 σ_i 分量的 q_i。

(2)中值重组。这种重组方式也是先随机选择两个父代个体如式(4-9),然后将父代个体各分量的平均值作为子代新个体的分量,构成的新个体为:

$$(X,\sigma)=((x_1^1+x_1^2)/2,(x_2^1+x_2^2)/2,\cdots,(x_n^1+x_n^2)/2),$$
$$((\sigma_1^1+\sigma_1^2)/2,(\sigma_2^1+\sigma_2^2)/2,\cdots,(\sigma_n^1+\sigma_n^2)/2))$$

这时,新个体的各个分量兼容两个父代个体信息,而在离散重组中则只含有某一个父代个体的因子。

(3) 混杂(Panmictic)重组。这种重组方式的特点在于父代个体的选择上。混杂重组时先随机选择一个固定的父代个体,然后针对子代个体每个分量再从父代群体中随机选择第二个父代个体。也就是说,第二个父代个体是经常变化的。至于父代两个个体的组合方式,既可以采用离散方式,也可以采用中值方式,甚至可以把中值重组中的 1/2 改为[0,1]之间的任一权值。

将上述三组方法排列组合,进化策略的重组方式有以下各种:

$$x_i'=\begin{cases}x_{S,i} & \text{无重组}\\ x_{S,i}\text{ 或 }x_{T,i} & \text{离散型}\\ x_{S,i}\text{ 或 }x_{T_i,i} & \text{混杂离散型}\\ x_{S,i}+(x_{T,i}-x_{S,i})/2 & \text{中值型}\\ x_{S,i}+(x_{T_i,i}-x_{S,i})/2 & \text{混杂中值型}\\ x_{S,i}+\eta(x_{T,i}-x_{S,i}) & \text{广义中值型}\\ x_{S,i}+\eta(x_{T_i,i}-x_{S,i}) & \text{广义混杂中值型}\end{cases}\qquad(4\text{-}10)$$

式中 x_i' —— 新个体目标变量 X 的第 i 个分量;

$x_{S,i}$ —— 固定的父代个体的目标变量 X 的第 i 个分量;

$x_{T,i}$ —— 离散重组时另一固定父代个体的目标变量 X 的第 i 个分量;

$x_{T_i,i}$ —— 随机选择的另一个父代个体的第 i 个分量;

η —— 权值,在[0,1]区间内。

上式为简便起见,只列举 X 因子。事实上对 σ、α 因子都可以采用上述 7 种重组方式,这样进化策略的重组算子便有 $7^3=343$ 种。

图 4 2 描述两维目标变量的重组结果。图中 P_1 和 P_2 代表两个父代个体的目标变量。P_1 和 P_2 所构成的长方形的 4 个顶点(图中用(1)标记)代表用离散重组产生的子代个体位置。长方形对角线的中点(用(2)标记)则是中值重组的结果。对角线上的其它点(用(3)标记)则代表广义中值重组的子代个体。至于广义混杂中值重组的结果,可位于长方形内的任一点,如(4)标记。

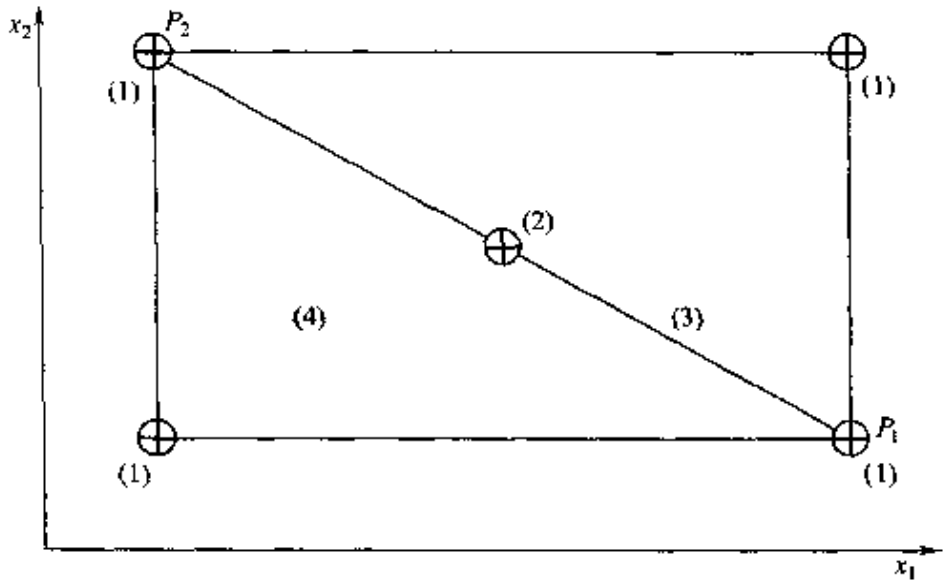

图 4-2 进化策略的重组机理

研究表明,进化策略采用重组后,明显增加算法的收敛速度。Schwefel 建议,对于目标变量 X 宜用离散重组,对子策略因子 σ 及 α 宜用中值重组或混杂中值重组。

4.2.5 突变

进化策略的突变是在旧个体基础上添加一个随机量,从而形成新个体。突变的过程如(4-5)式所述,为:

$$\begin{cases} \sigma_i' = \sigma_i \cdot \exp\left(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)\right) \\ \alpha_j' = \alpha_j + \beta \cdot N_j(0,1) \\ x_i' = x_i + z_i \end{cases}$$

式中系数 τ'、τ 及 β 按 Schwefel 建议,可按下式计算:

$$\tau = (\sqrt{2\sqrt{n}})^{-1}$$

$$\tau' = (\sqrt{2n})^{-1}$$

$$\beta \approx 0.0873$$

式中 n——个体中所含分量数目。

τ' 及 τ 分别为全局步长系数及局部步长系数，用于对随机量 $N(0,1)$ 进行缩放，它们相当于人工神经网络中的“学习率”。通常，τ' 及 τ 取为 1。系数 β 涉及旋转角度，建议取 5°，按弧度计算则为 $\pi \times 5/180 \approx 0.0873$。调整 τ'、τ 及 β 数值最终影响目标变量 X。

若取消旋转因子，进化策略的表达由三元组变为二元组，则突变转化为：

$$\begin{cases} \sigma_i' = \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)) \\ x_i' = x_i + \sigma_i' \cdot N_i(0,1) \end{cases}$$

若进一步简化，σ_i 都相同时，上式变为：

$$\begin{cases} \sigma' = \sigma \cdot \exp(\tau_0 \cdot N_i(0,1)) \\ x_i' = x_i + \sigma' \cdot N_i(0,1) \end{cases} \tag{4-11}$$

式中 $\tau_0 = 1/\sqrt{n}$。

这也就是进化策略最简单的突变。

进化策略在进行突变时，一要防止 σ_i 在进化过程中变为零，从而使 x_i 的进化停止；另一个是防止旋转角 α_j 在可行域$[-\pi,\pi]$之外。为此，要经常检查 σ_i 及 α_j，使之符合规定。在算法中执行下述操作：

If $|\alpha_j'| > \pi$ then $\alpha_j' := \alpha_j' - 2\pi \cdot \mathrm{sign}(\alpha_j')$

及 If $\sigma_i' < \varepsilon_0$ then $\sigma_i' := \varepsilon_0$

式中 ε_0 为大于 0 的一个小数值。

近年来，有些人提出用柯西(Cauchy)分布代替突变中的正态分布。一维柯西概率密度函数集中在原点附近，其定义为：

$$f_t(x) = \frac{1}{\pi} \frac{t}{t^2 + x^2}, \quad -\infty < x < +\infty$$

式中 $t > 0$ 为比例系数，相应的概率分布函数为：

$$F_t(x) = \frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{x}{t}\right)$$

从概率密度函数看，柯西分布类似于正态分布，但在垂直方向上柯西分布较小，而在水平方向上柯西分布愈靠近水平轴变得愈宽，可视作是无限的(图 4-3)。因此，采用柯西分布进行变异，使个体的变化更宽广，更容易跳出局部最优解。当然，柯西分布的中央部分较小又是它的一个弱点。

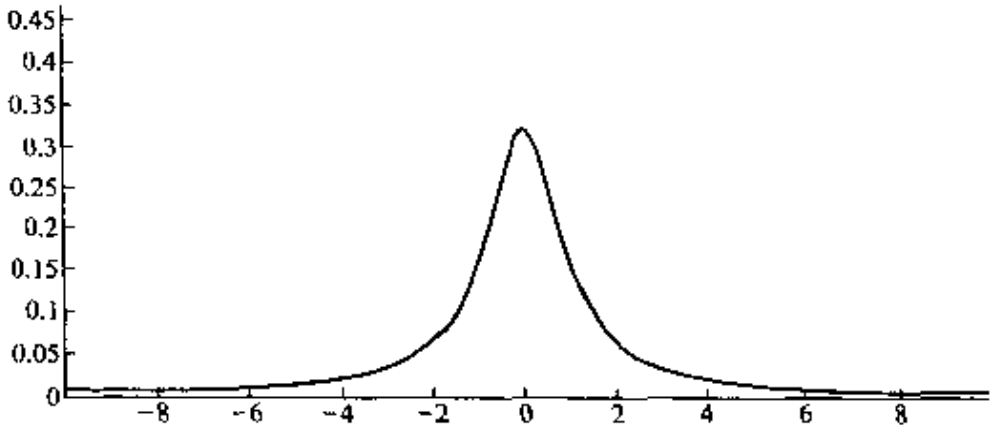

图 4-3　柯西分布

4.2.6　选择

进化策略中的选择(Selection)类似于遗传算法的复制，它们都体现达尔文的“物竞天择、适者生存”的原则。但是，进化策略中的选择是确定型操作，它严格根据适应度的大小，将劣质个体完全淘汰。选择中不采用轮盘法那种随机方式，而是使优良个体 100% 地被保留，劣质个体 100%地被淘汰。这是进化策略不同于遗传算法的主要特征之一。

进化策略的选择有两种：一为 $(\mu + \lambda)$ 选择，另一为(μ,λ) 选择。$(\mu + \lambda)$ 选择是从 μ 个父代个体及 λ 个子代新个体中确定性地择优选出 μ 个个体组成下一代新群体。(μ,λ) 选择是从 λ 个子代新个体中确定性地择优挑选 μ 个个体(要求 $\lambda > \mu$) 组成下一代群体，每个个体只存活一代，随即被新个体顶替。粗略地看，似乎$(\mu + \lambda)$

选择最好，它可以保证最优个体存活，使群体的进化过程呈单调上升趋势。但是，深入分析后发现$(\mu+\lambda)$选择具有下述缺点：

(1) $(\mu+\lambda)$选择保留旧个体，它有时会是过时的可行解，妨碍算法向最优方向发展。(μ,λ)选择全部舍弃旧个体，使算法始终从新的基础上全方位进化。

(2) $(\mu+\lambda)$选择保留旧个体，有时是局部最优解，从而误导进化策略收敛于次优解而不是最优解。(μ,λ)选择舍弃旧的优良个体，容易进化至全局最优解。

(3) $(\mu+\lambda)$选择在保留旧个体的同时，也将进化参数σ保留下来，不利于进化策略中的自适应调整机制。(μ,λ)选择则恰恰相反，可促进这种自适应调整。

实践也证明，(μ,λ)－ES优于$(\mu+\lambda)$－ES，前者已成为当前进化策略的主流。

在(μ,λ)－ES中，为了控制群体的多样性和选择的力度，比值μ/λ是一个重要参数，它对算法的收敛速度有很大影响。一方面，μ不能太小，否则群体太单调(例如$\mu=1$的极端情况)；另一方面，μ也不能太大，否则计算工作量过大。通常，μ取15或更多一些。λ数值的大小，类似于μ的作用，也要适当。某些研究表明，比值μ/λ宜取1/7左右。也就是说，若$\mu=15$，则λ宜取100。

4.2.7 终止

进化策略经过多次的迭代进化，算法逐渐收敛。判别算法需否中止的准则，可以参照遗传算法的终止准则(2.1.7)，即规定迭代次数、规定最小偏差、观察适应度变化趋势等三种方法。

针对进化策略，Schwefle提出用最优个体和最差个体之比较来决定算法是否终止。一旦最优适应度与最差适应度的差值小于允许值，则令算法终止。这种终止条件可表述为：

$$(f_{\max}-f_{\min})\leqslant\zeta$$

式中 $f_{\max},f_{\min}$ —— 分别为适应度的最大值及最小值；

ζ —— 大于零的定值。

上述终止条件也可改写作相对差，即

$$\frac{f_{max}-f_{min}}{f_{max}}\leqslant\zeta$$

式中　ζ—— 大于零的定值。

4.3　进化策略的表述

进化策略的工作过程包括表达问题、产生的初始群体、计算适应度、执行重组、突变、选择新群体等工作，经过反复迭代，逐渐得出最优解。

图 4-4 表示进化策略的工作流程。图中 Gen 表示进化的代次，在第 0 代，根据问题表达是二元组或三元组方式，随机产生 μ 个初始个体，并计算它们的适应度。然后依次执行重组和突变操作，产生新个体。图中 j 统计新个体数目，重组和突变执行 λ 次，产生 λ 个新个体。随后计算新个体的适应度，再根据选择策略，从 $(\mu+\lambda)$ 个体中或 λ 个新个体中选择 μ 个个体组成新群体。这样，便完成一代进化。重复这种进化过程，直至满足终止条件。

概括地讲，进化策略的工作步骤是：

(1)确定问题的表达方式。

(2)随机生成初始群体，并计算其适应度。

(3)根据进化策略，用下述操作产生新群体：

1) 重组。将两个父代个体交换目标变量和随机因子，产生新个体。

2) 突变。对重组后的个体添加随机量，产生新个体。

3) 计算新个体适应度。

4) 选择。根据选择策略，挑选优良个体组成下一代群体。

(4)反复执行(3)，直至达到终止条件，选择最佳个体作为进化策略的结果。

进化策略还可以用形式化语言表达。令 $a\in I$ 标记个体，I 为个体空间。适应度函数记为 $\Phi:I\to R$。在第 t 代，群体 $\mathrm{P}(t)=\{a_1(t),a_2(t),\cdots,a_n(t)\}$ 经过重组 r、突变 m 及选择 s 转换成下一代群体。这里 r、m、s 均指宏算子，把旧群体转换为新群体。$l:I^{\mu}\to$

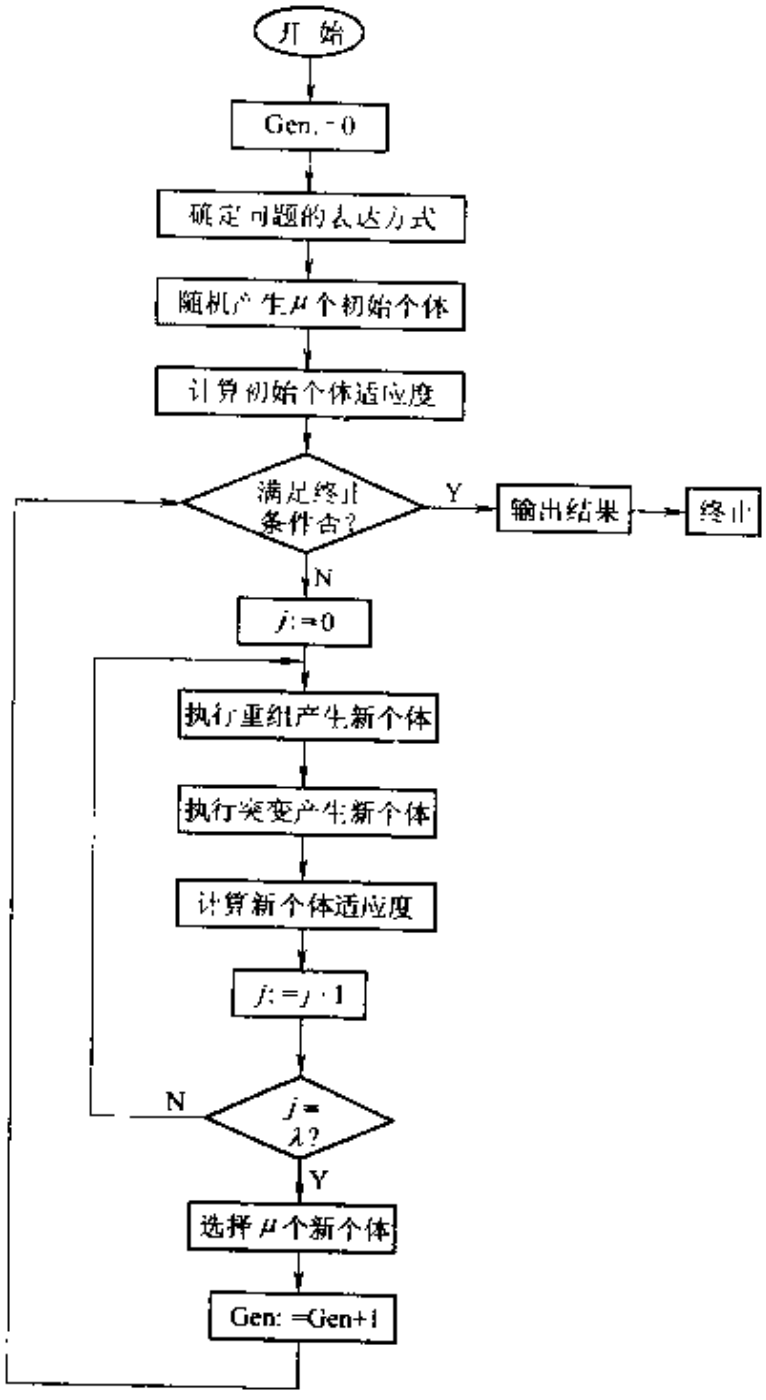

图 4-4　进化策略工作流程

{True,False} 记为终止条件。利用上述符号，进化策略可描述为：

t=0;

initialize P(0) : = $\{a_1(0), a_2(0), \cdots, a_\mu(0)\}$;

evaluate P(0): $\{\Phi(a_1(0)), \Phi(a_2(0)), \cdots, \Phi(a_\mu(0))\}$;

```
while  (l(P(t))≠True)  do
    recombination: P'(t) := r(P(t));
    mutation: P''(t) := m(P'(t));
    evaluate P''(t): {Φ(P''(t))};
    selection: If (μ,λ) - selection
                then s_(μ,λ)(P''(t))
                else s_(μ+λ)(P(t) ∪ P''(t));
    t=t+1;
end
```

4.4 进化策略的理论分析

4.4.1 (1+1)-ES的理论分析

(1+1)-ES中父代个体及子代都只有一个个体，情况简单，便于开展讨论。为统一起见，今后用随机变量Z表示目标变量$X=(x_1,x_2,\cdots,x_n)$突变后产生的偏移量，且目标函数为最小值问题。

4.4.1.1 个体的进化

设$Z=(z_1,z_2,\cdots,z_n)$中对应于x_i的每个偏移量z_i相互独立，服从$(0,\sigma_i^2)$的正态分布，则Z的概率密度函数为：

$$w(z_1,z_2,\cdots,z_n)=\prod_{i=1}^{n}w(z_i)=\frac{1}{(2\pi)^{n/2}\prod_{i=1}^{n}\sigma_i}\exp\left(-\frac{1}{2}\sum_{i=1}^{n}\left(\frac{z_i}{\sigma_i}\right)^2\right) \tag{4-12}$$

当$\sigma_i=\sigma$ $(i=1\sim n)$时，上式可简化为：

$$w(Z)=\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n\exp\left(\frac{-Z^TZ}{2\sigma^2}\right) \tag{4-13}$$

式中 $Z=(z_1,z_2,\cdots,z_n)^T$

随机偏移的总距离S可视作：

$$S=\sqrt{\sum_{i=1}^{n}z_i^2} \tag{4-14}$$

很明显，S^2服从χ^2分布，当n足够大时，它近似为$(\sigma\sqrt{n-\frac{1}{2}},\frac{\sigma^2}{2})$

的正态分布。也就是说，S 的数学期望可视为 $\sigma\sqrt{n}$，它随 n 及 σ 而变化；S 的方差为 $\sigma^2/2$，只与 σ 有关。

图 4-5 表示 $n=2$ 时目标变量 $X=(x_1,x_2)$ 的进化过程。图中用等值线表示不同 X 值时的目标函数 $F(X)$；用矢量 $X_E^{(g)}$ 指向初始点 $E^{(g)}$，其中 g 代表迭代的代次。图中以 $E^{(g)}$ 为中心、偏移量 $Z^{(g)}$ 为半径作一圆，圆周上各点表示可能的偏移点。假设 $N^{(g)}$ 是子代个体，其目标变量 $X_N^{(g)}$ 为：

$$\vec{X}_N^{(g)}=\vec{X}_F^{(g)}+\vec{Z}^{(g)}$$

由于 $N^{(g)}$ 点所处的目标函数等值线劣于 $E^{(g)}$ 点，舍弃新个体 $N^{(g)}$，让父代个体 $E^{(g)}$ 为下一代新个体 $E^{(g+1)}$。同理，从 $E^{(g+1)}$ 开始，加上偏离量 $Z^{(g+1)}$ 得子代个体 $N^{(g+1)}$。由于 $N^{(g+1)}$ 点所处的目标函数等值线优于 $E^{(g+1)}$，故选择 $N^{(g+1)}$ 为下一代新个体 $E^{(g+2)}$，其目标变量为 $X_E^{(g+2)}$。

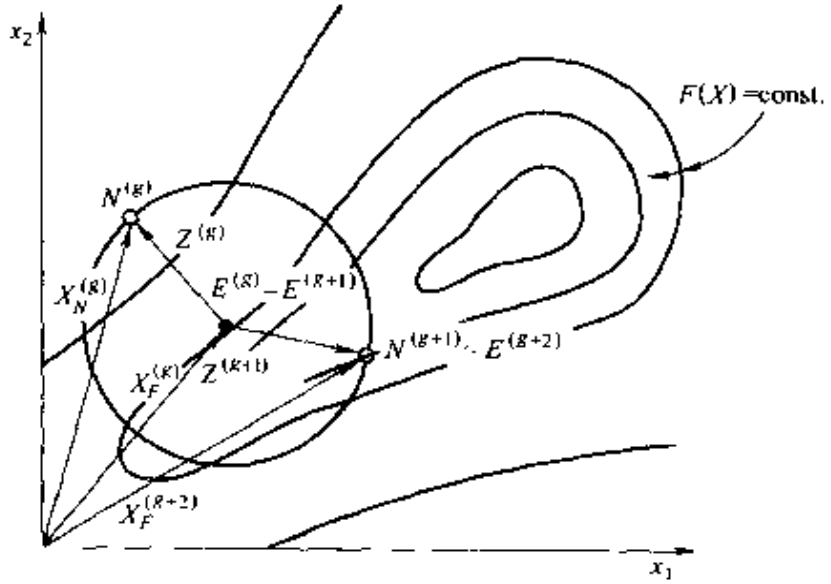

图 4-5 (1 + 1) — ES 的进化过程

4.4.1.2 步长控制

进化策略利用方差 σ^2 使目标变量 X 不断进化。因此，方差 σ^2 可视作每次进化的步长。控制 σ^2 就相当于控制进化过程。现以图 4-6 表示进化过程的控制。图中目标函数 $F(X)$ 为：

$$F(X) = x_1^2 + x_2^2 = \text{const.}$$

最优点 $X = (0, 0)$，以此点为中心的一系列同心圆表示 $F(X)$ 为不同值时的等值线图。假设初始点为 $E^{(g)}$，它距最优点的距离为 $r^{(g)}$。由 $E^{(g)}$ 衍生出 $E^{(g+1)}$，其距离为 $r^{(g+1)}$。对于这一次突变，算法前进了一步，令收敛率 $\varphi^{(g)}$ 为：

$$\varphi^{(g)} = r^{(g)} - r^{(g+1)} \tag{4-15}$$

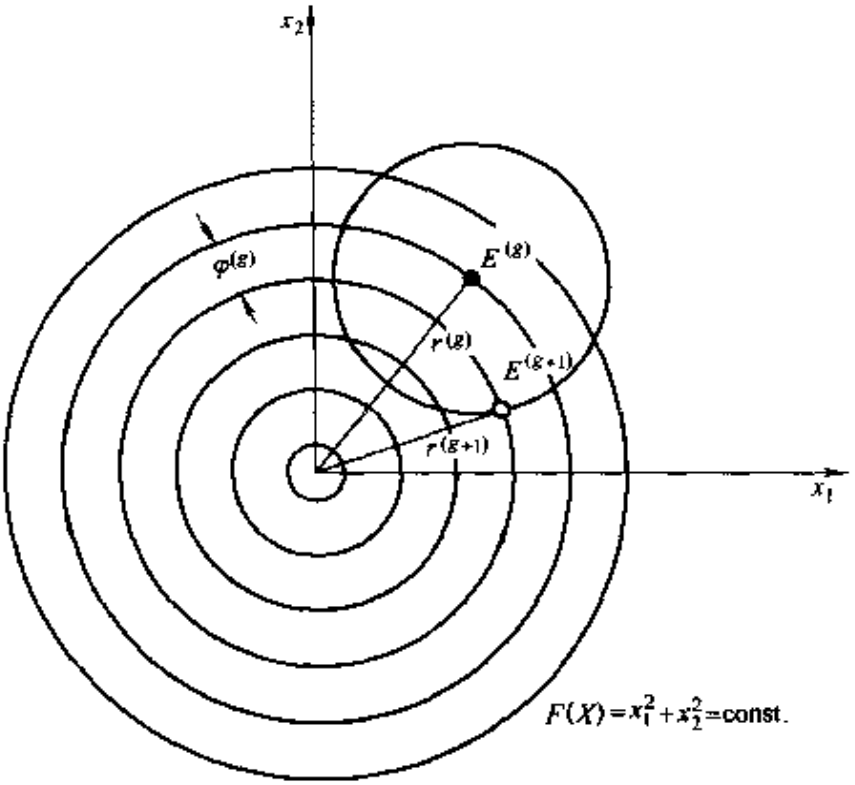

图 4-6　球状模型的收敛率

Rechenberg 通过研究，认为最优的收敛率 φ_{max} 为：

$$\varphi_{max} = k_1 \frac{r}{n}, \quad k_1 \approx 0.2025 \tag{4-16}$$

相应的标准差常为最优标准差 σ_{opt}，可按下式计算：

$$\sigma_{opt} = k_2 \frac{r}{n}, \quad k_2 \approx 1.224 \tag{4-17}$$

式中　r —— 前进距离；

n —— 个体目标变量 $X = (x_1, x_2, \cdots, x_n)$ 的分量数目。

在进化策略中，每次进化都是由方差 σ^2 产生，因此方差的变化是衡量算法收敛的一个主要标准。由(4-15)～(4-17)式，有：

$$\frac{\sigma_{opt}^{(g+1)}}{\sigma_{opt}^{(g)}} = \frac{r^{(g-1)}}{r^{(g)}} = 1 - \frac{k_1}{n}$$

对于 n 次迭代，则为：

$$\frac{\sigma_{opt}^{(g-n)}}{\sigma_{opt}^{(g)}} = \left(1 - \frac{k_1}{n}\right)^n \tag{4-18}$$

当 n 足够大时，上述方差比值趋于一个常数，即：

$$\lim_{n\to\infty}\left(1 - \frac{k_1}{n}\right)^n = e^{-k_1} \approx 0.817 \approx \frac{1}{1.224} \tag{4-19}$$

请注意，上述公式推导中为了计算方便，将迭代次数的 n 等同于 X 的分量数目 n。根据(4-16)式，当 n 很大时收敛率 φ 变小。也就是说，此时 φ 对 σ 的变化很不敏感，要经过多次迭代才能确定成功概率。因此，1/5 成功定律要修正为：

每隔 n 次突变，检查 $10n$ 次突变的成功率。若成功率小于 $2n$，将步长 σ 乘以系数 0.85；若成功率大于 $2n$，将步长 σ 除以 0.85。具体数学表达式为：

$$\sigma^{(g)} = \begin{cases} \sigma^{(g-n)} \cdot \delta & \text{当 } p_s < 0.2 \\ \sigma^{(g-n)} & \text{当 } p_s = 0.2 \\ \sigma^{(g-n)}/\delta & \text{当 } p_s > 0.2 \end{cases} \tag{4-20}$$

式中 $\delta = 0.85$；

p_s——n 次突变的成功概率。

此外，为了避免步长 σ 趋于零，防止进化停滞，还要求 σ 有下限，即：

$$\sigma_i^{(g)} \geqslant \varepsilon_a \qquad \text{对 } i = 1 \sim n$$

$$\sigma_i^{(g)} \geqslant \varepsilon_b |x_i^{(g)}| \qquad \text{对 } i = 1 \sim n$$

式中 ε_a、ε_b —— 大于零的数，取决于计算精度。

4.4.2 (μ,λ)－ES 的理论分析

4.4.2.1 (μ,λ)－ES 的进化

(μ,λ)－ES 是个群体进化问题，比(1＋1)－ES 复杂得多。

图 4-7 表示 (2,4) － ES 的进化过程。图中 $E_1^{(g)}$ 表示一个父代个体，它产生的两个子代个体 $N_1^{(g)}$ 及 $N_2^{(g)}$。$E_2^{(g)}$ 表示另一个父代个体，它衍生两个子代个体 $N_3^{(g)}$ 及 $N_4^{(g)}$。从 4 个子代个体中择优选取 $N_3^{(g)}$ 及 $N_2^{(g)}$ 为下代个体 $E_1^{(g+1)}$ 及 $E_2^{(g+1)}$。

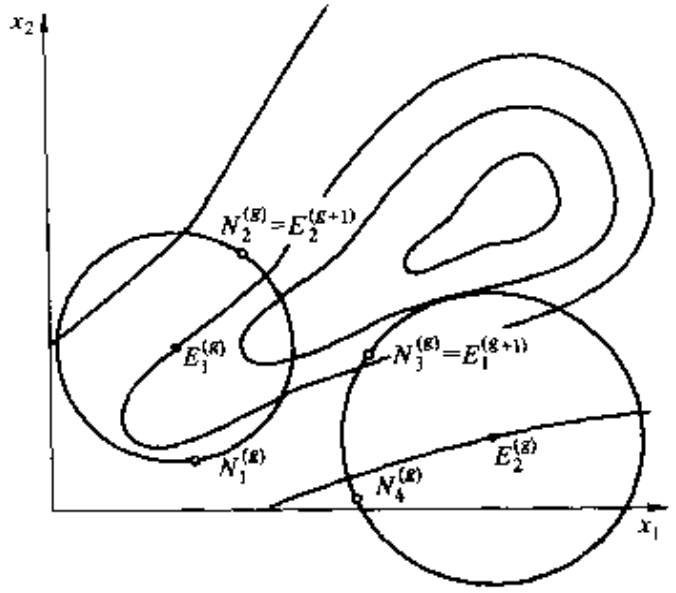

图 4-7 (2,4)－ES 的进化过程

4.4.2.2 (μ,λ) －ES 的收敛

仿照 (1 + 1) － ES 的 φ，(μ,λ) － ES 的收敛率用数学期望值衡量，即

$$\varphi = E\{\| \hat{X} - \overline{X}^{(g)} \| - \| \hat{X} - \overline{X}^{(g-1)} \|\}$$

式中 E —— 取数学期望；

$\hat{X}$ —— 目标变量的最优值；

$\overline{X}^{(g)}$ —— 第 g 代目标变量的平均值。

对于目标变量 $X = (x_1, x_2, \cdots, x_n)^T$，假设每个分量 x_i 对应的方差 σ_i^2 均等于 σ^2，则从始点 E 偏移至 N 点的概率密度函数是：

$$w(E \to N) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_{E,i} - x_{N,i})^2\right) \tag{4-21}$$

式中 $X_E = (x_{E,1}, x_{E,2}, \cdots, x_{E,n})^T$

$$X_N = (x_{N,1}, x_{N,2}, \cdots, x_{N,n})^T$$

距离 $\| X_E - X_N \| = \sqrt{\sum_{i=1}^{n} (x_{E,i} - x_{N,i})^2}$

用距离 $\| X_E - X_N \|$ 衡量，有些 N 点因远离目标被舍弃，有些则因靠近目标被采纳。现假设某一被采纳的距离 $s = f(X_E, X_N)$，式中 f 称权衡函数，它不同于目标函数 $F(X)$，于是 s 发生的概率密度是下述多重积分：

$$w(s) = \int\cdots\int_{f(X_E, X_N) = s} w(E \to N)\, \mathrm{d}x_{N,1}\, \mathrm{d}x_{N,2} \cdots \mathrm{d}x_{N,n} \tag{4-22}$$

上式仅仅针对一次随机偏离得出。然而进化策略却是从 λ 个子代个体中优选 μ 个新个体，其中每个子代个体的偏离距离用 s_i 标记。按优劣顺序排列，某代中第 ν 个优良子代个体偏离 s' 距离的概率密度 $w_\nu(s')$ 应是下述三部分的组合：

(1) 概率密度 $w(s_{i_1} = s)$，即某些新个体 N_{i_1} 恰好前进 s' 距离而靠近目标；

(2) 概率 $p(s_{i_2} > s')$，即某些新个体 N_{i_2} 的前进距离超过 s' 而更靠近目标；

(3) 概率 $p(s_{i_3} < s')$，即某些新个体 N_{i_3} 前进少于 s' 而远离目标。

由于比第 ν 个体更靠近目标的个体数为 $\nu - 1$，而比 ν 差的个体数为 $\lambda - \nu$，因此概率密度 $w_\nu(s')$ 为：

$$\begin{aligned} w_\nu(s') = \sum_{i_1=1}^{\lambda} \{ & w(s_{i_1} = s') \cdot \sum_{\substack{i_2=1 \\ i_2 \neq i_1}}^{\lambda-\nu+2} \{ p(s_{i_2} > s') \cdot \sum_{\substack{i_3=i_2+1 \\ i_3 \neq i_1}}^{\lambda-\nu+3} \{ p(s_{i_3} > s') \\ & \cdot \sum_{\substack{i_4=i_3+1 \\ i_4 \notin \{i_2, i_1\}}}^{\lambda-\nu+4} \{ p(s_{i_4} > s') \cdots \sum_{\substack{i_\nu=i_{\nu-1}+1 \\ i_\nu \notin \{i_1, i_2, \cdots, i_{\nu-2}\}}}^{\lambda} \{ p(s_{i_\nu} > s') \\ & \cdot \prod_{\substack{i_{\nu+1}=1 \\ i_{\nu+1} \notin \{i_1, i_2, \cdots, i_\nu\}}}^{\lambda} p(s_{i_{\nu+1}} < s') \} \cdots \} \} \} \} \end{aligned} \tag{4-23}$$

对 μ 个新个体取平均值，得平均概率密度 $w(s')$：

$$w(s') = \frac{1}{\mu}\sum_{\nu=1}^{\mu} w_\nu(s') \tag{4-24}$$

由此可得收敛率 φ:

$$\varphi = \int_{s=s_u}^{\infty} s' \cdot w(s') \cdot \mathrm{d}s' \tag{4-25}$$

很明显,解算上式很复杂。为了简化,用 X_k 表示所有父代个体,用 $\sigma_{l,i}$ 表示所有子代个体($l = 1 \sim \lambda$)的各个分量($i = 1 \sim n$)的标准差,则式(4-23)变为:

$$w_\nu(s') = \lambda \cdot C_{\lambda-1}^{\nu-1} \cdot w(s_l = s')[p(s_l < s')]^{\lambda-\nu}[p(s_l > s')]^{\nu-1} \tag{4-26}$$

由于 $$p(s_l > s') + p(s_l < s') = 1$$

及 $$C_{\lambda-1}^{\nu-1} = \frac{(\lambda-1)!}{(\nu-1)!(\lambda-\nu)!}$$

因此式(4-26)变为:

$$w_\nu(s') = \frac{\lambda!}{(\nu-1)!(\lambda-\nu)!} w(s_l = s')[p(s_l < s')]^{\lambda-\nu}[1 - p(s_l < s')]^{\nu-1} \tag{4-27}$$

当 $\mu = 1$ 时即成为$(1,\lambda)$ − ES,上式变成:

$$w(s') = w_1(s') = \lambda \cdot w(s_l = s') \cdot [p(s_l < s')]^{\lambda-1} \tag{4-28}$$

式中:

$$w(s_l = s') = \int\cdots\int_{f(X_E, X_N) = s} \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_{E,i} - x_{N,i})^2\right) \mathrm{d}x_{N,1}\,\mathrm{d}x_{N,2}\cdots\mathrm{d}x_{N,n} \tag{4-29}$$

及

$$p(s_l < s') = \int_{s_l=-\infty}^{s'} w(s_l = s')\,\mathrm{d}s_l \tag{4-30}$$

现在再返回讨论式(4-25)。该式实质上是对 $w(s')$ 在 $s_u < s' \leqslant \infty$ 区间内求均值(数学期望)。下限 s_u 有两种情况:

(1) 新衍生个体的性能(适应度)都不及父代个体,下一代要从父代个体挑选,即相当于$(\mu + \lambda)$ − ES 情况。这时 $S_u = 0$。

(2) 新衍生个体的性能优于父代个体，父代不参与新个体选择，即相当于(μ,λ) — ES_o这时$s_w = -\infty$。

1.4.2.3 球状模型

球状模型是指目标函数具有以下型式的优化问题：

$$F(X) = \sum_{i=1}^{n} x_i^2 = \text{const.}$$

同样，进化策略主要讨论最小值问题，即最优解在零点。

图 4-8 表示 $n = 2$ 的球状模型的进化过程。图中以原点为中心的同心圆表示目标函数的等值线。E 点为初始点，它恰好在 x_1 轴上。N_l 表示衍生的新个体，用 r_l 表示它距最优点的距离。图中 N_{l_2} 比 E 点优越，其前进距离 $s_{l_2} > 0$；相反，N_{l_1} 不及 E 点，其前进距离 $s_{l_1} < 0$。

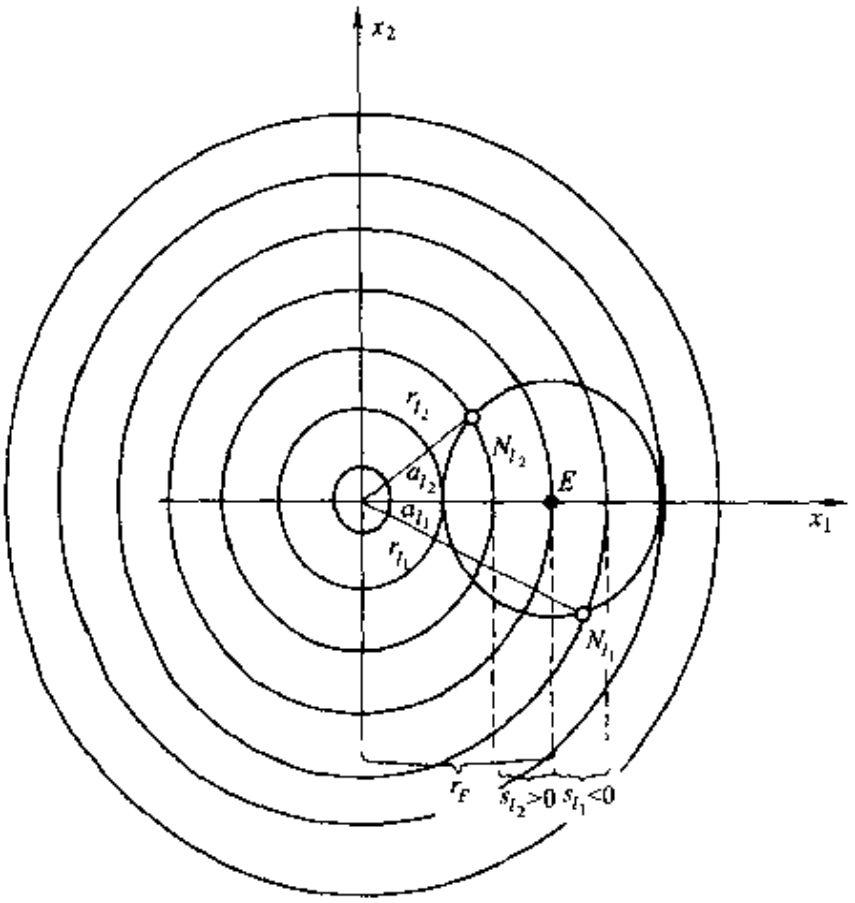

图 4-8 球状模型

一般地，对于n维空间，令父代个体E的目标变量$X_E = (x_{E,1},$

$x_{E,2}, \cdots x_{E,n})^T$ 所有分量对应的方差(偏离)都是 σ^2。令距离 r_I 为：

$$r_I = \sqrt{\sum_1^n x_{I,i}^2}$$

为今后解算方便，令初始点 E 位于第一个坐标轴上，即 $X_E = (r_E, 0, \cdots, 0)^T$，则式(4-21)可写作：

$$w(E \to N_I) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{1}{2\sigma^2}(r_I^2 + r_E^2 - 2r_E \cdot x_{I_1})\right)$$

令前进距离 s_I 为：

$$s_I = r_E - r_I$$

则 s_I 发生的概率密度 $w(s_I)$ 为

$$w(s_I) = \int_{r_E}\cdots\int_{r_I=r_I} \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{1}{2\sigma^2}(r_I^2 + r_E^2 - 2r_E \cdot x_{I_1})\right) \mathrm{d}x_{I_1}\,\mathrm{d}x_{I_2}\cdots\mathrm{d}x_{I_n}$$

将上述直角坐标系转换为球坐标系，得：

$$w(s_I) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \frac{\pi^{\frac{n-1}{2}}}{\Gamma\left(\frac{n-1}{2}\right)} \exp\left(-\frac{r_E^2 + r_I^2}{2\sigma^2}\right) r_I^{n-1} \int_{\alpha=0}^{2\pi} \exp\left(\frac{r_E \cdot r_I \cdot \cos\alpha}{\sigma^2}\right) \sin^{n-2}\alpha \,\mathrm{d}\alpha$$

将积分用贝塞耳函数(Bessel function)表示，则上式为：

$$w(s_I) = \frac{r_I^{\frac{n}{2}} \cdot r_E^{1-\frac{n}{2}}}{\sigma^2} \exp\left(-\frac{r_E^2 + r_I^2}{2\sigma^2}\right) I_{\frac{n}{2}-1}\left(\frac{r_E \cdot r_I}{\sigma^2}\right) \quad (4\text{-}31)$$

为了简化上式，令：

$$\nu = \frac{n}{2}, \quad a = \frac{r_E^2}{\sigma^2}, \quad v = \frac{r_I}{r_E}$$

则式(4-31)变为：

$$w(s_I) = \frac{a}{r_E} \cdot e^{-\frac{a}{2}} \cdot v^{\nu} \cdot e^{-\frac{av^2}{2}} \cdot I_{\nu-1}(a \cdot v)$$

其中 $s_I = r_E(1 - v)$

为了用(4-28)式计算总的概率，λ 个子代个体最大前进距离 s' 为：

$$s = \max_{l}\{s_l \mid l = 1 \sim \lambda\} - r_E - r$$

而且有：

$$w(s_l = s') = \frac{a}{r_E} \cdot e^{-\frac{au}{2}} \cdot u^{\nu} \cdot e^{-\frac{au^2}{2}} \cdot I_{\nu-1}(a \cdot u)$$

式中 $u = \dfrac{r'}{r_E}$ 及 $s' = r_E(1-u)$

且 $p(s_l < s) = 1 \quad p(s_l \geqslant s) = 1 - \int_{s_l=r_E}^{s'} w(s_l)\,\mathrm{d}s_l$

$$= 1 - \int_{v=0}^{au} a \cdot e^{-\frac{av}{2}} \cdot v^{\nu} \cdot e^{-\frac{av^2}{2}} \cdot I_{\nu-1}(a \cdot v)\,\mathrm{d}v$$

最后，可得某次突变中前进 s 距离的概率密度函数：

$$w(s) = \frac{a}{r_E} \cdot e^{-\frac{au}{2}} \cdot u^{\nu} \cdot e^{-\frac{au^2}{2}} \cdot I_{\nu-1}(a \cdot u)$$

$$\left[1 - a \cdot e^{-\frac{au}{2}} \int_{v=0}^{au} v^{\nu} \cdot e^{-\frac{av^2}{2}} \cdot I_{\nu-1}(a \cdot v)\,\mathrm{d}v\right]^{\lambda-1}$$

由于很难计算上述概率分布的数学期望，我们转而确定它的最大值以得到近似的收敛率 $\bar{\varphi}$，由极值条件：

$$\left.\frac{\partial w(s)}{\partial s'}\right|_{s'=\varphi} = 0$$

且令

$$D(y) = a \cdot e^{-\frac{ay}{2}} \cdot y^{\nu} \cdot e^{-\frac{ay^2}{2}} \cdot I_{\nu-1}(a \cdot y)$$

可得

$$\lambda = 1 + \left.\frac{\partial D(u)}{\partial u}\right|_{u=1-\bar{\varphi}/r_E} \left[D(1 - \bar{\varphi}/r_E)\right]^{-2}\left[1 - \int_{v=0}^{\bar{\varphi}-r_E} D(v)\,\mathrm{d}v\right] \tag{4-32}$$

利用 Debye 级数，有：

$$\left.\frac{\partial D(u)}{\partial u}\right|_{u=1-\bar{\varphi}/r_E} = D(1-\bar{\varphi}/r_E)\left[a \cdot \exp\left\{a(1-\bar{\varphi}/r_E)\right\}^{\nu} - \frac{1}{1-\bar{\varphi}/r_E} - a(1-\bar{\varphi}/r_E)\right]$$

将上式代入式(4-32)，得到一个复杂表达式，可简写作：

$$\lambda = \lambda(\bar{\varphi}, \sigma, r_E, n)$$

由于我们对出发点 r_I 的具体位置不感兴趣，可用相对值表达，令：

$$\varphi^* = \frac{\bar{\varphi} \cdot n}{r_E} \quad 及 \quad \sigma^* = \frac{\sigma_n \cdot n}{r_E}$$

用 φ^* 及 σ^* 代替 $\bar{\varphi}$ 及 σ，求极限：

$$\lim_{n \to \infty} \lambda(\varphi^*, \sigma^*, r_E, n)$$

得

$$\lambda = \lambda(\varphi^*, \sigma^*) = 1 + \sqrt{\pi}\left(\frac{\varphi^*}{\sqrt{2}\sigma^*} + \frac{\sigma^*}{\sqrt{8}}\right)$$

$$\exp\left[\left(\frac{\varphi^*}{\sqrt{2}\sigma^*} + \frac{\sigma^*}{\sqrt{8}}\right)^2\right]\left[1 + \operatorname{erf}\left(\frac{\sigma^*}{\sqrt{8}}\right)\right] \qquad (4\ 33)$$

式中 $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x \exp(-t^2)\mathrm{d}t$，称误差函数(Error Function)。

在上述求极限过程中，变量 n 和 γ_E 被消去，余留的 φ^* 和 σ^* 可视作“通用”变量。图 4-9 就是利用(4-33)式表示下述函数关系：

$$\varphi^* = \varphi^*(\sigma^*, \lambda)$$

从图中可以看出，当 $\sigma^* \to 0$ 时，$\varphi^* \to 0$，即进化没有改善，这和人们想象一致。当 $\lambda = 1$ 时 φ^* 总是负值，这是因为(1，λ)－ES 意味着每次突变不论结果好坏，都一律接纳新个体。然而对于球状模型，除了 $\sigma^* = 0$ 之外，收敛率 φ 总是小于目标变量 X 的变动范围之半。随着 σ^* 的增加，此差距更大，因此，改善的概率小于恶化的概率，φ^* 总是小于 0，而且 σ^* 愈大 φ^* 更向负值方向扩大。

当 $\lambda \geqslant 2$，由于性能不良的子代个体可以被父代个体取代，提高了改善的可能性。因此，φ^* 可为正值，在初始时随着 σ^* 的增加而增加，逐渐达到 φ^* 的极值。

假设 λ 为常数，利用 φ^* 对 σ^* 求一阶导数并令之为零，从式(4-33)可得：

$$\sigma^+(\varphi^+ + \sigma^+)\exp(-\sigma^{+2}) + \sqrt{\pi}(\sigma^- - \varphi^+)$$

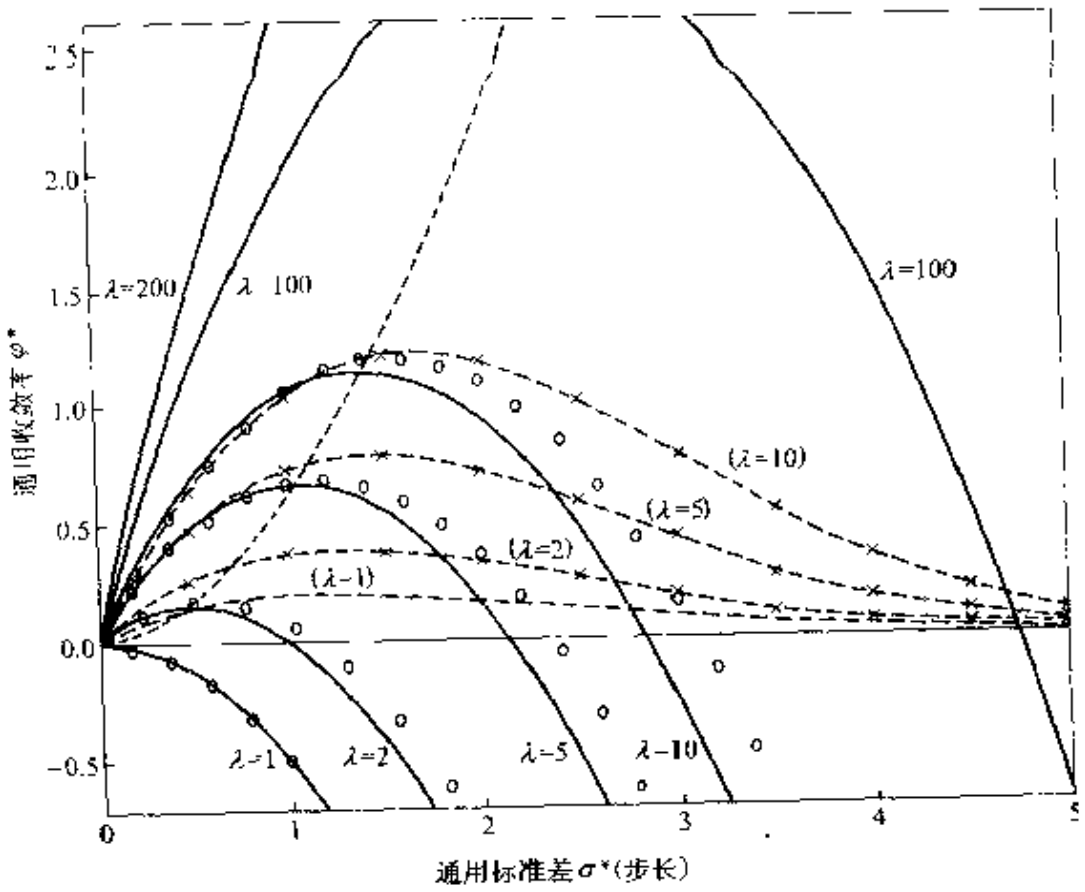

图 4-9　球状模型的收敛率

$$\left[\frac{1}{2}+(\varphi^{+}+\sigma^{+})^{2}\right](1+\operatorname{erf}(\sigma^{+}))=0 \tag{4-34}$$

式中 $\dfrac{\sigma^{*}_{opt}}{\sqrt{8}}=\sigma^{+}$ 及 $\dfrac{\varphi^{*}_{max}}{\sqrt{2}\,\sigma^{*}_{opt}}=\varphi^{+}$

根据不同的 λ 常数，由式(4-33)及(4-34)可求出 $\varphi^{*}_{max}=\varphi^{*}(\sigma^{*}=\sigma^{*}_{opt})$，从而得出图 4-10 的关系曲线。

进一步，假设：

$$\varphi^{+}=\sigma^{+}，即\ 2\varphi^{*}_{max}\approx\sigma^{*2}_{opt}$$

可近似得出：

$$\lambda\approx 1+\sqrt{\pi\varphi^{*}_{max}}\exp(\varphi^{*}_{max})\left[1+\operatorname{erf}\left(\frac{1}{2}\sqrt{\varphi^{*}_{max}}\right)\right]$$

将上式对 φ_{max} 求导，可得 λ 的最大值：

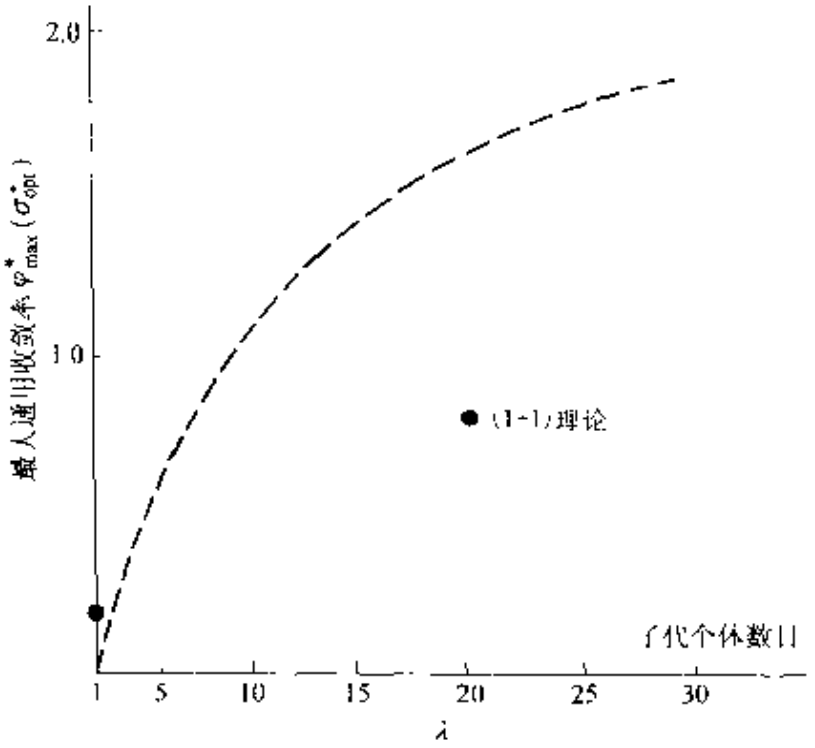

图 4-10　球状模型的最大收敛率

$$\lambda_{opt} = 2\sqrt{\pi}\left[\frac{1}{2} + (\varphi^{-} + \sigma^{-})^{2}\right]\exp\left[\varphi^{-} + \sigma^{+})^{2}\right]$$
$$\left[1 + \mathrm{erf}(\sigma\ )\right]\varphi^{1} \qquad (1\text{-}35)$$

最后，有：

$$\lambda_{opt} \approx 4.7 \quad 即 \quad \lambda_{opt} \approx 5$$

4.4.2.4　步长控制

为了今后公式推导的方便，令目标变量 X 的方差 σ^2 具有以下形式：

$$\sigma_N^{(g)} = \sigma_E^{(g)} \cdot Z(g) \qquad (4\ 36)$$

式中　$\sigma_N^{(g)}$ —　第 g 代新个体的标准差；

$\sigma_E^{(g)}$　　第 g 代旧个体的标准差；

$\overline{Z}(g)$　· 第 g 代产生的随机数，按下式计算：

$$\overline{Z} = e^{Y} \qquad (4\text{-}37)$$

式中　Y— — 服从 $(0,\tau^2)$ 正态分布的随机数。

$\bar{Z}$ 的概率密度函数为：

$$w(\bar{Z}) = \frac{1}{\sqrt{2\pi}\tau}\cdot\frac{1}{\bar{Z}}\exp\left(-\frac{(\ln\bar{Z})^2}{2\tau^2}\right)$$

下面讨论 τ 的选择。首先假定 X 的所有分量 x_i 具有相同的 σ^2，仿照 4.4.1.2 的方法，有：

$$\frac{\sigma_{opt}^{(g+1)}}{\sigma_{opt}^{(g)}} = \exp\left(-\frac{\varphi_{max}^*}{n}\right) \tag{4-38}$$

针对随机变量 $\bar{Z}$ 的乘积，得如下随机变量 W：

$$W = \left(\prod_{g=1}^{n}\bar{Z}^{(g)}\right)^{1/n} = \exp\left(\frac{1}{n}\sum_{g=1}^{n}Y^{(g)}\right)$$

其中 $Y^{(g)}$ 的平均值为：

$$V = \frac{1}{n}\sum_{g=1}^{n}Y^{(g)}$$

由于 $Y^{(g)}$ 服从 $(0,\tau^2)$ 正态分布，因此 V 服从 $(0,\tau^2/n)$ 正态分布，即每代平均变化步长为 $\exp(\tau/\sqrt{n})$。联系到式(4-38)，可近似得：

$$\exp\left(\frac{\varphi_{max}^*}{n}\right) \approx \exp\left(\frac{\tau}{\sqrt{n}}\right)$$

即：

$$\tau \approx \frac{\varphi_{max}^*}{\sqrt{n}} \tag{4-39}$$

进一步，假设每个分量 x_i 有独立的 $\sigma_i^2(i=1\sim n)$，式(4-36)改造为：

$$\sigma_{N,i}^{(g)} = \sigma_{E,i}^{(g)}\cdot\bar{Z}_i^{(g)}\cdot\bar{Z}_0^{(g)}$$

式中 $\bar{Z}_i^{(g)}$ ——第 g 代针对分量 i 的偏移；

$\bar{Z}_0^{(g)}$ ——第 g 代针对整个目标变量的偏移。

相应地，可令：

$$\tau_0 \approx \frac{\varphi^*}{\sqrt{2n}} \qquad \text{对 } \bar{Z}_0 \text{ 而言}$$

$$\tau \approx \frac{\varphi^*}{\sqrt{2\sqrt{n}}} \qquad \text{对 } \bar{Z}_i(i=1\sim n) \text{ 而言}$$

4.4.3 坐标旋转

由式(4-12)已知，对应于目标变量 $X=(x_1,x_2,\cdots x_n)^T$ 的随机偏移量 $Z=(z_1,z_2,\cdots z_n)^T$，其概率密度函数为：

$$w(Z)=\frac{1}{(2\pi)^{n/2}\prod_{i=1}^{n}\sigma_i}\exp\left(-\frac{1}{2}\sum_{i=1}^{n}\left(\frac{z_i}{\sigma_i}\right)^2\right) \tag{4-40}$$

将上式写成更一般的形式，应为：

$$w(Z)=\frac{1}{(2\pi)^{n/2}\sqrt{\mathrm{Det}\,C}}\exp\left(\frac{1}{2}(Z-\zeta)^T\cdot C^{-1}\cdot(Z-\zeta)\right) \tag{4-41}$$

式中 ζ 可视为随机偏移因子的确定性部分。然而进一步研究表明，即使采用 ζ 作为进化策略的一个可变动因素，对改善算法的收敛性无明显意义。因此，改善算法的另一个途径便是 C。假设 C^{-1} 是对角矩阵，且对角线上的元素为 $\sigma_i(i=1\sim n)$。由于有等式：

$$\sqrt{\mathrm{Det}\,C}=\prod_{i=1}^{n}\sigma_i$$

因此当 $\zeta=0$ 时，式(4-41)便等同于式(4-40)。假设 C^{-1} 不是对角矩阵，则 C^{-1} 变为：

$$C^{-1}=\begin{bmatrix}\sigma_{11} & C_{12} & \cdots & C_{1n}\\ C_{21} & \sigma_{22} & \cdots & C_{2n}\\ \cdots & & & \\ C_{n1} & C_{n2} & \cdots & \sigma_{nn}\end{bmatrix}$$

矩阵中对角线上的元素 σ_{ij} 实质上仍是标准差(方差的平方根)σ_i，而对角线以外的元素则是协方差 C_{ij}。很明显，假设目标变量 X 有 n 个分量，则协方差的最大个数 m 为：

$$m=\frac{1}{2}n(n-1)$$

在进化策略中，控制参数 σ_i 最多可以为 n 个，因此添加协方差 C_{ij} 后可供控制的参数变为：

$$m+n=\frac{1}{2}n(n+1)$$

也就是说，将原来的控制参数由方差 σ_i^2 扩展到协方差 C_{ij}，可以增强进化策略的适应能力。这时，偏离量的概率密度函数变为：

$$w(Z)=\frac{1}{(2\pi)^{n/2}\sqrt{\mathrm{Det}\ C}}\exp\left\{-\frac{1}{2}Z^T\cdot C^{-1}\cdot Z\right\} \quad (4\text{-}42)$$

然而，为了保持直角坐标系且使矩阵 C^{-1} 有确定的正值，对 C_{ij} 及 σ_i^2 有严格要求。以 $n=2$ 为例，当

$$C^{-1}=\begin{bmatrix}\sigma_1 & C_{12}\\ C_{21} & \sigma_2\end{bmatrix}$$

要求

$$C_{12}^2=C_{21}^2<\sigma_1^2\sigma_2^2$$

且相关系数 ρ 的绝对值小于 1，即：

$$\rho_{12}=\frac{C_{12}}{\sqrt{\sigma_1\sigma_2}},\qquad -1<\rho_{12}<+1$$

由于对协方差 C_{ij} 有如此复杂要求，为此改用坐标轴旋转的方法。在传统的二元表达方式 (X,σ) 的基础上扩展出三元表达方式 (X,σ,α)，其中旋转角度 α 的进化为：

$$\alpha_{N,j}^{(g)}=\alpha_{E,j}^{(g)}+\hat{z}_j^{(g)}$$

式中 $\alpha_{N,j}^{(g)}$ —— 第 g 代第 j 分量的新衍生值；

$\alpha_{E,j}^{(g)}$ —— 第 g 代第 j 分量的原有旋转角度；

$\hat{z}_j^{(g)}$ —— 第 g 代第 j 分量的旋转角度的随机偏移量，可按正态分布产生。

图 4-11 表示 $n=2$ 的二维情况。根据坐标旋转原理，有下述关系：

$$\Delta x_1=\Delta x_1'\cos\alpha-\Delta x_2'\sin\alpha$$

$$\Delta x_2=\Delta x_1\sin\alpha-\Delta x_2'\cos\alpha$$

式中 α —— 坐标 (x_1,x_2) 变为 (x_1',x_2') 的旋转角度；

$\Delta x_1'$，$\Delta x_2'$ —— 在 (x_1',x_2') 坐标系下的坐标值；

Δx_1，Δx_2 —— 在 (x_1,x_2) 坐标系下的坐标值。

旋转角度和协方差、方差的关系可用下式表达：

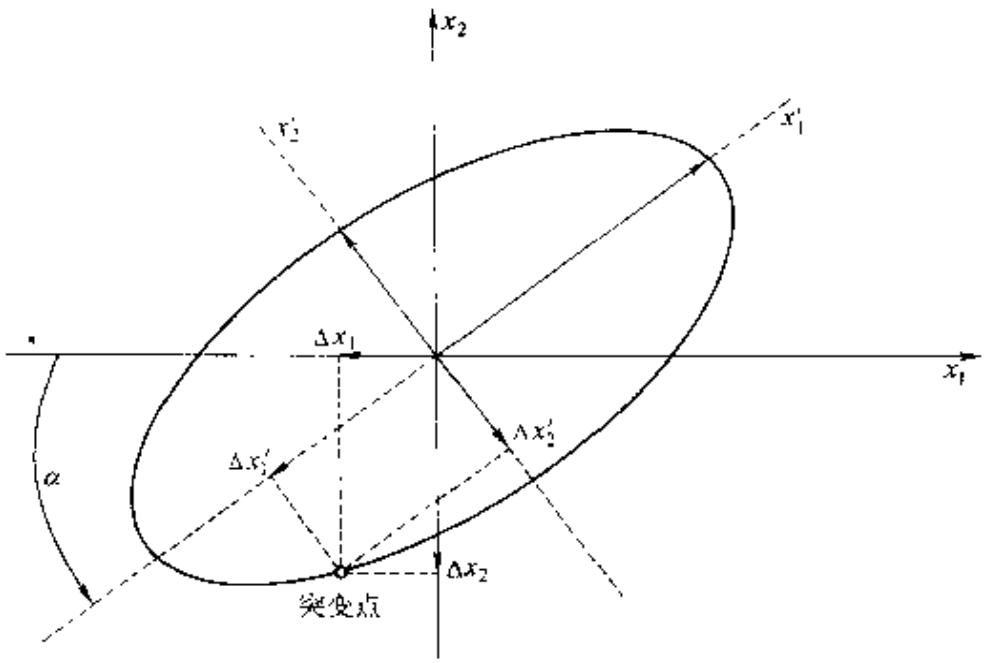

图 4-11　坐标旋转

$$\tan(2\alpha_{ij}) = \frac{2C_{ij}}{\sigma_i^2 - \sigma_j^2}$$

4.5　进化策略与遗传算法的比较

进化策略和遗传算法都属于进化算法，它们有许多相同的地方，但也有明显的差别。

4.5.1　相同

进化策略和遗传算法的相同点，首先表现在两者都体现生物进化过程中的“物竞天择、优胜劣汰”的原则，从随机产生的初始可行解出发，经过进化择优，逐渐逼近最优解。

第二个相同的地方，是两者都是渐进式搜索寻优，经过多次的反复迭代，不断扩展搜索范围，最终找出全局最优解。

第三个相同点是两种进化算法都采用群体的概念。尽管早期的进化策略 $(1+1)-ES$ 及 $(\mu+1)-ES$ 是基于单个个体，但它最后也发展为 $(\mu+\lambda)-ES$ 或 $(\mu,\lambda)-ES$，同时驱动多个搜索点，体现并行算法的特点。

此外，两种算法在自适应搜索、有指导的搜索、全局式寻优以

及黑箱式结构见(1.3.2)等方面，都有许多相似之处。

4.5.2 差别

进化策略有别于遗传算法，主要表现在：

(1) 表达方式上差别。进化策略采用十进制的实型数表达问题，便于处理连续的优化计算类课题。遗传算法常采用二进制编码表达问题，更适宜处理离散型问题。

进化策略源于函数优化处理，它是一种类似于爬山问题的优化方法，方差 σ 相当于爬山的步长，旋转角 α 可比作山坡坡角，遗传算法起源于自适应搜索，强调全方位探查。

(2) 算子的差别。进化策略的重组算子，不仅可以复制父代个体的部分信息(x_i，σ_i 或 α_i)，而且还可以通过中值计算产生新的信息。与之相对应的遗传算法的交换算子，仅仅是交换父代个体的部分字符，不能产生新的字符。

突变是进化策略和遗传算法都采用的算子名称，然而实际应用上有明显差别。进化策略的突变是在旧个体基础上添加一个正态分布的随机数，从而产生新个体。遗传算法的突变是对旧个体的某个字符进行补运算，将 0 变为 1 或将 1 变为 0。此外，突变是进化策略的主要进化手段，每个新个体都经历突变。在遗传算法中，仅仅某些个体的个别字符产生突变，它不及复制、交换那样重要。

选择是进化策略和遗传算法差别最明显的一种操作。进化策略从 λ 个新个体或从 $(\mu+\lambda)$ 个个体中挑选 μ 个个体组成新群体，而且挑选方法是确定型的。遗传算法的选择体现在复制中，它从旧群体中择优插入新群体，而且挑选方法是随机的轮盘法，优良个体入选概率高，但劣质个体有时也会“破例”入选。

(3) 执行顺序的差别。进化策略的进化顺序是先执行重组，随之为突变，最后才是选择。遗传算法最先执行选择及复制，其次为交换，最后是突变。图 4-12 是遗传算法与进化策略在选择顺序上的差别，图 4-12*a* 表示遗传算法，图 4-12*b* 表示进化策略。

进化策略的重组和突变都是针对同一旧个体依次进行。遗传算法的交换和突变不一定会发生在同一旧个体上。

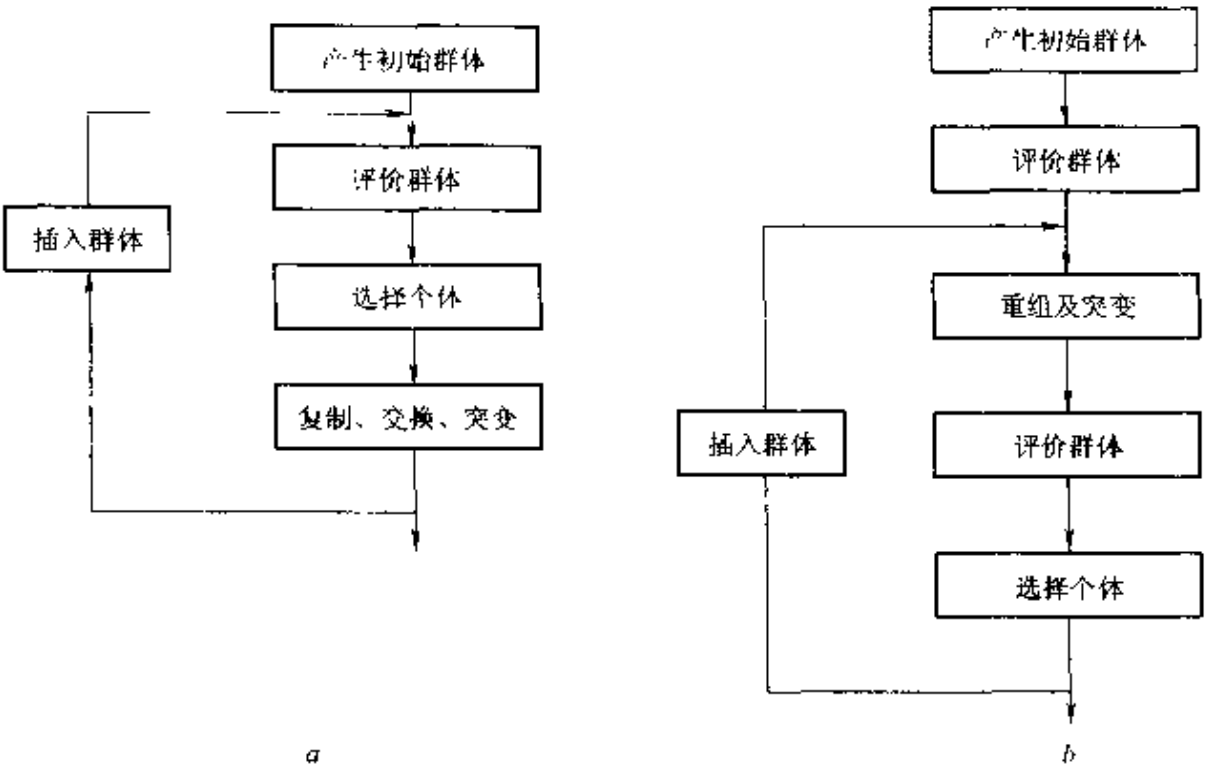

图 4-12　遗传算法与进化策略的比较

(4) 参数的差别。遗传算法操作参数指复制概率 p_r、交换概率 p_c 及突变概率 p_m，它们在运行过程中原则上是固定的。进化策略的控制因子包括标准差 σ 及旋转角度 α，它们在运行过程中不断发生自适应调整。

4.5.3　相互借鉴

进化策略和遗传算法作为进化算法的两个分支，是独立出现及平行发展，在长期实践中它们又相互借鉴，不断完善。

在问题的表达方式上，遗传算法已从原来的二进制编码扩展为用十进制实数表达问题，并相应改变突变等算子。

在算子方面，最早的进化策略只有突变算子，然后才添加重组算子，这是进化策略借鉴遗传算法中的交换算子。

对于突变算子，遗传算法不如进化策略那样重视，通常突变概率 p_m 的取值都很小(0.01)。近年来不少人提高突变概率，p_m 取值达 0.2。

遗传算法的交换算子也有向进化策略的重组算子靠拢的迹

象。例如，有人建议交换算子采用下述线性组合。

$$y_1 = \alpha \cdot x_1 + (1 - \alpha)x_2$$

$$y_2 = (1 - \alpha) \cdot x_1 + \alpha \cdot x_2$$

式中 y_1、y_2 ——新个体的目标变量分量；

x_1、x_2 ——旧个体的目标变量分量；

α —— 系数，介于[0,1]。

实质上这种交换就是进化策略的广义中值重组。

在参数的自适应调整方面，进化策略的这一特征也开始渗透到遗传算法中。例如，遗传算法中的群体规模 M、突变概率 p_m 已不再是常数，它们可随时间而变化。又如，有人提出在遗传算法内再运行一个参数遗传算法，后者负责调整遗传算法的操作参数。

本来，科学技术的发展要依靠各学科的交叉渗透。进化策略和遗传算法的相互借鉴正是科学技术发展的正确途径。

5 进化规划

5.1 进化规划的基本原理

1962 年美国 L. J. Fogel 首先提出进化规划(Evolutionary Programming，简称 EP)，当时并未得到足够的重视。30 多年后，其子 D. B. Fogel 改善了这种方法，从而使进化规划作为进化算法的一个分支得到广泛应用。1992 年在美国圣地亚哥举行第一届进化规划年度会议(First Annual Conference on Evolutionary Programming)，以后每年举行一次。

应该指出，尽管进化规划与进化策略十分相似，但是它们一直是相互独立地平行发展，前者主要在美国，后者主要在欧洲，双方缺乏交流。一直到 1992 年，进化规划和进化策略的科技人员才相互接触，促使两者相互渗透。

5.1.1 标准进化规划

进化规划用传统的十进制实数表达问题。在标准进化规划(Standard EP)中，个体的表达形式为：

$$x_i' = x_i + \sqrt{f(X)} \cdot N_i(0,1) \tag{5-1}$$

式中 x_i——旧个体目标变量 X 的第 i 个分量；

x_i'——新个体目标变量 X' 的第 i 个分量；

$f(X)$——旧个体 X 的适应度；

$N_i(0,1)$——针对第 i 分量发生的随机数，它服从标准正态分布。

上式表明，新个体是在旧个体的基础上添加一个随机数，添加值的大小与个体的适应度有关：适应度大的个体添加值也大，反之亦然。

根据这种表达方式，进化规划首先产生 μ 个初始个体，这也就是突变。接着从 μ 个旧个体及 μ 个新个体(2μ 个个体)中根据适应度挑选出 μ 个个体组成新群体。如此反复迭代，直至得到满意结果。进化规划的工作流程类似于其他进化算法，同样经历产生初始

群体 - 突变 - 计算个体适应度 - 选择 - 组成新群体，然后反复迭代，一代一代地进化，直至达到最优解。

应该指出，进化规划没有重组或交换这类算子，它的进化主要依赖突变。在标准进化规划中这种突变十分简单，它只需参照个体适应度添加一个随机数。很明显，标准进化规划在进化过程中的自适应调整功能主要依靠适应度 $f(X)$ 来实现。

5.1.2 元进化规划

为了增加进化规划在进化过程中的自适应调整功能，人们在突变中添加方差的概念。类似于进化策略，在进化规划中个体的表达采用下述方式：

$$\begin{cases} x_i' = x_i + \sqrt{\sigma_i} \cdot N_i(0,1) \\ \sigma_i' = \sigma_i + \sqrt{\sigma_i} \cdot N_i(0,1) \end{cases} \tag{5-2}$$

式中 x_i —— 旧个体目标变量 X 的第 i 个分量；

x_i' —— 新个体目标变量 X' 的第 i 个分量；

σ_i —— 旧个体第 i 分量的标准差；

σ_i' —— 新个体第 i 分量的标准差；

$N_i(0,1)$ —— 针对第 i 分量产生的服从标准正态分布的随机数。

从上式可以看出，新个体也是在旧个体的基础上添加一个随机数，该添加量取决于个体的方差，而方差在每次进化中又有自适应调整。这种进化方式已成为进化规划的主要手段，因此在进化规划前冠以“元”这个术语以表示它为基本方法。

元进化规划(Meta EP)的突变尽管类似于进化策略，但是它们有下述差别(可对照式(4-4))：

(1) 执行顺序不同。进化规划中首先计算新个体的目标变量 x_i'，计算中沿用旧个体的标准差 σ_i；其次才计算新个体的标准差 σ_i'，新的标准差留待下次进化时才用。与之相反，进化策略是先调整标准差 σ，然后再用新的标准差 σ' 去更改个体的目标变量 X。

(2) 计算式的不同。进化规划的计算式比进化策略的计算式(4-4)简单。

总之，元进化规划使 EP 靠近 ES。

5.1.3 旋转进化规划

旋转进化规划(Rmeta EP)进一步扩展进化规划，在表达个体时添加第三个因子——协方差，用三元组表示个体，即(X,σ,ρ)，具体计算如下：

$$\begin{cases} X' = X + N(0,C) \\ \sigma_i' = \sigma_i + \sqrt{\sigma_i} \cdot N_i(0,1) \\ \rho_j' = \rho_j + \sqrt{\rho_j} \cdot N_j(0,1) \end{cases} \tag{5-3}$$

式中 X——旧个体的目标变量，其内含 n 个分量；

X'——新个体的目标变量，其内含 n 个分量；

$N(0,C)$——遵从正态分布的随机数，其数学期望为 0，其标准差与协方差有关；

ρ_j——相关系数，$\rho_j = \dfrac{c_{ij}}{\sqrt{\sigma_i \sigma_j}}$。

旋转进化规划仿效三元组表示的进化策略，在控制因子中除了方差 σ 外，还添加相关系数 ρ，增加算法的自适应调整能力。

不过，旋转进化规划比较复杂，目前尚未得到充分认可，最常用的还是元进化规划，本章以它为重点进行介绍。

5.2 进化规划的基本技术

5.2.1 表达方法

和其他进化算法一样，进化规划也是一种反复迭代、不断进化的过程。进化规划和进化策略一样，都是采用十进制的实型数表达问题。每个个体的目标变量 X 可以有 n 个分量，即

$$X = (x_1, x_2, \cdots, x_i, \cdots, x_n)$$

相应地，每个个体的控制因子 σ_i 和 x_i 是一一对应的，n 个 x_i 要有 n 个 σ_i，这点不像进化策略那样自由，后者允许某些 x_i 分量没有 σ_i。

由 X 和 σ 组成的二元组(X,σ)是进化规划最常用的表达形式。有人建议将进化规划再增加一个控制因子 ρ，构成三元表达式(X,σ,ρ)，其中

$$\rho = (\rho_1, \rho_2, \cdots, \rho_j, \cdots, \rho_m)$$

ρ_j 是相关系数的单下标表达，它表示 x_i 和 x_j 之间的协方差，即：

$$\rho_j = \rho_{ij} = \frac{c_{ij}}{\sqrt{\sigma_i \sigma_j}} \qquad \rho_j \in [-1, +1] \tag{5-4}$$

式中 c_{ij} 是协方差，它构成协方差矩阵 $C^{n\times n} = (c_{ij})$。ρ 的个数 m 最多为 $n(n-1)/2$。因此，进化规划控制因子 (σ, ρ) 的最大组合数为：

$$n(n-1)/2 + n = n(n+1)/2$$

5.2.2 产生初始群体

进化规划中产生初始群体的方法类似于进化策略，从可行解中随机选择 μ 个个体作为进化计算的出发点。

5.2.3 计算适应度

进化规划采用十进制实数表达问题，计算适应度比较简单直观。

5.2.4 突变

突变是进化规划产生新群体的唯一方法，它不采用重组或交换算子。

(1) 对于标准进化规划，突变操作为：

$$x_i' = x_i + \sqrt{\beta_i \cdot \Phi(X) + r_i} \cdot N_i(0,1) \tag{5-5}$$

式中 x_i ——旧个体目标变量 X 的第 i 个分量；

x_i' ——新个体目标变量 X' 的第 i 个分量；

$\Phi(X)$ ——旧个体 X 的适应度之函数；

$N_i(0,1)$ ——针对 i 分量产生的随机数，它服从标准正态分布；

β_i ——系数，常取 1；

r_i ——系数，常取 0。

若 $\beta_i = 1$ 及 $r_i = 0$，则上式变为：

$$x_i = x_i + \sqrt{\Phi(X)} \cdot N_i(0,1) \tag{5-6}$$

上式突出表明 $\Phi(X)$ 的作用。若个体 X 的适应度 $f(X)$ 很大，直接用 $f(X)$ 乘 $N_i(0,1)$ 会使 x_i 远离 x_i，在算法的后期无法平稳收敛。为此，采用平方根 $\sqrt{f(X)}$ 或按比例缩小的方法变换适应度，于是

统一写作 $\Phi(X)$ 的形式。

不过，即使采用 $\Phi(X)$ 而不是 $f(X)$，目标变量 X 的变动还是比较剧烈，很难准确地收敛在最优点上，这是标准进化规划的致命弱点。

(2) 对于元进化规划，突变是：

$$\begin{cases} x_i' = x_i + \sqrt{\sigma_i'} \cdot N_i(0,1) \\ \sigma_i' = \sigma_i + \sqrt{\eta \cdot \sigma_i} \cdot N_i(0,1) \end{cases} \tag{5-7}$$

式中 σ_i ——旧个体第 i 分量的标准差；

σ_i' ——新个体第 i 分量的标准差；

η ——系数。

其他符号同式(5-5)。

这种进化规划增加一个控制因子——方差，它使 x_i 可以在小范围内变动，有利于算法的收敛。在上式中系数 η 的目的是增加 σ 的变动范围，若 η 取 1 则(5-7)式与(5-2)式一样。本节有关突变的描述与 5.1 及 1.2 节是一样的，仅仅在表达方式上循序渐进、由简到繁。

类似于进化策略，要使标准差大于零。为此要经常检查 σ_i'，及时予以纠正，即

$$\text{If} \quad \sigma_i' \leqslant 0 \qquad \text{then} \quad \sigma_i' = \zeta_0$$

式中 ζ_0 为大于 0 的小数值。

(3) 对于旋转进化规划，突变是：

$$\begin{cases} X' = X + N(0,C) \\ \sigma_i' = \sigma_i + \sqrt{\eta \cdot \sigma_i} \cdot N_i(0,1) \\ \rho_j' = \rho_j + \sqrt{\zeta \cdot \rho_j} \cdot N_j(0,1) \end{cases} \tag{5-8}$$

上式与(5-3)式基本一样，只是添加系数 η 及 ζ，增大 σ 及 ρ 的变动范围。不过这种三元突变方式尚未得到广泛应用。

5.2.5 选择

进化规划中没有重组或交换那样的算子，突变以后便执行选择。在进化规划中，新群体的个体数目 λ 等于旧群体的个体数目 μ，

即 $\lambda = \mu$。选择便是在 2μ 个个体中选择 μ 个个体组成新群体。按照进化策略中的术语，进化规划的选择是 $(\mu + \mu)$ 选择。

进化规划的选择采用随机型的 q 竞争选择法。遗传算法(2.4.3)所描述的竞技选择法就是来源于进化规划。在这种选择方法中，为了确定某一个体 i 的优劣，我们从新、旧群体的 2μ 个个体中任选 q 个个体组成测试群体。然后将个体 i 的适应度与 q 个个体的适应度进行比较，记录个体 i 优于或等于 q 内各个体的次数，此数便是个体 i 的得分 W_i，即

$$W_i = \sum_{j=1}^{q} \begin{cases} 1 & \text{If } f_i \text{ 优于或等于 } f_j \\ 0 & \text{其他} \end{cases} \tag{5-9}$$

式中 f_i —— 个体 i 的适应度；

f_j ——q 个测试群体中第 j 个体的适应度。

上述得分测试分别对 2μ 个个体进行，每次测试时重新选择 q 个个体组成新的测试群体。最后，按个体的得分选择分值高的 μ 个个体组成下一代新群体。q 竞争选择法的工作步骤如下：

(1) 从 $1 \sim 2\mu$，依次选择个体 i；

(2) 从 2μ 个个体中随机选择 q 个测试个体；

(3) 按适应度比较个体 i 与 q 个测试个体的优劣，记录 i 优于或等于 q 的次数，即为个体 i 的得分 W_i；

(4) 重复(1)、(2)、(3)，直至 2μ 个个体都有得分 W；

(5) 按得分 W_i 的大小，挑选得分大的前 μ 个个体组成新群体。

q 竞争选择法是一种随机选择，总体上讲，优良个体入选的可能性较大。但是由于测试群体 q 每次都是随机选择的，当 q 个个体都不甚好时，有可能使较差的个体因得分高而入选。这正是随机选择的本意。

q 竞争选择法中 q 的大小是一个重要参数。若 q 很大，极端地设 $q = 2\mu$，则选择变为确定性选择。反之，若 q 很小，则选择的随机性太大，不能保证优良个体入选。通常 q 在 10 以上，可取 0.9μ。

5.2.6 终止

进化规划在进化过程中，每代都执行突变、计算适应度、选择

等操作，不断反复执行，使群体素质得到改进，直至取得满意的结果。

进化规划的终止准则与进化策略相同，即根据最大进化代次、最优个体与期望值的偏差、适应度的变化趋势以及最优适应度与最差适应度之差等四个判据。

5.3 进化规划的表述

进化规划的工作流程包括表达问题、产生初始群体、计算适应度、执行突变、选择、反复迭代，直到取得满意结果。

图 5-1 表示进化规划的工作流程。图中 Gen 表示进化代次。在第 0 代，根据问题的表达方式，随机产生初始群体，并计算其适应度。随后，进入正常循环。每次迭代中，执行突变，计算新个体适应度。当新个体达到 μ 个后，从 2μ 个新、老个体中择优选出 μ 个个体组成新群体。如此反复迭代，直至满足终止条件。

进化规划的算法，可归纳为：

(1)确定问题的表达方式。

(2)随机产生初始群体，并计算其适应度。

(3)用下述操作产生新群体：

1)突变。对旧个体添加随机量，产生新个体；

2)计算新个体适应度；

3)选择。挑选优良个体组成新群体。

(4)反复执行(3)，直至满足终止条件，选择最佳个体作为进化规划的最优解。

若用形式化语言表述进化规划，可令 $a \in I$ 标记个体，I 为个体空间。适应度函数记为 $\Phi: I \to R$。在第 t 代，群体 $P(t) = \{a_1(t), a_2(t), \cdots, a_n(t)\}$ 经过突变 m、选择 s 转换成下一代群体。这里 m、s 均指宏算子，把旧群体转换为新群体。$l: I^{\mu} \to \{\text{True}, \text{False}\}$ 记为终止条件。利用上述符号，进化规划可描述为：

$t=0$;

initialize $P(0) := \{a_1(0), a_2(0), \cdots, a_\mu(0)\}$;

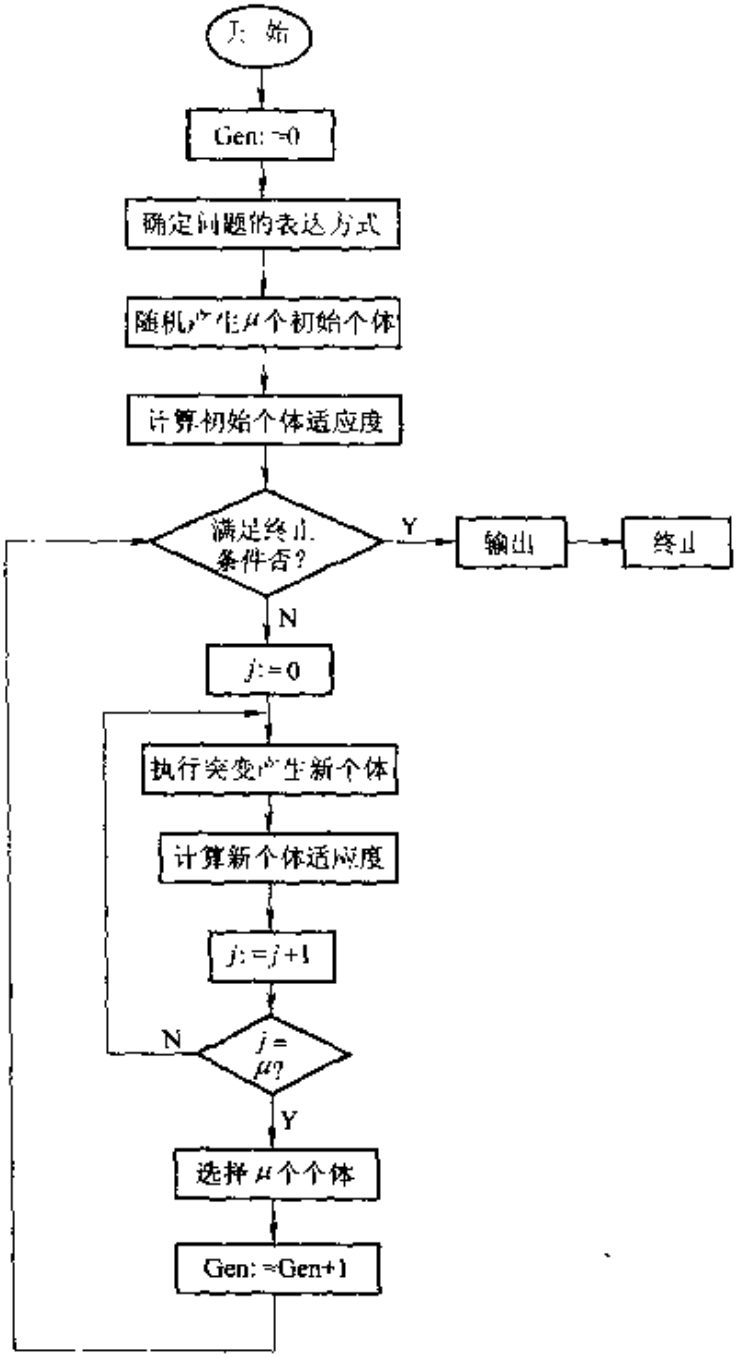

图 5-1

evaluate P(0): $\{\Phi(a_1(0)), \Phi(a_2(0)), \cdots, \Phi(a_n(0))\}$;

while (l(P(t)) $\neq$True) do

mutation: P′(t) := m(P(t));

evaluate P′(t): $\{\Phi(P'(t))\}$;

selection: s(P(t) $\bigcup$ P′(t));

$t=t+1$;

end

5.4 进化规划的理论分析

相对而言，进化规划的应用不如遗传算法、遗传规划及进化策略，因此相应的理论研究也比较薄弱。

5.4.1 q-竞争选择

q-竞争选择可以用于各种进化算法中，然而它源于进化规划。因此，在这里我们深入分析 q-竞争选择的机理。

假设在进化规划中父代有 μ 个个体，经过突变又产生 μ 个新个体，利用 q-竞争选择，要从这 $\mu+\mu$ 个个体中筛选出 μ 个子代个体。为此，将 2μ 个个体按适应度大小顺序排列，其中优胜者置于前面，即

$$a_1, a_2, \cdots, a_\mu, a_{\mu-1}, \cdots, a_{2\mu}$$

且 $$\Phi(a_i) \leqslant \Phi(a_{i+1})$$

式中 a_i —— 第 i 个个体；

$\Phi(a_i)$—— 第 i 个个体的适应度函数值，按最小值问题处理。

令事件 B 表示个体 a_j 不劣于随机选出的 q 个个体，即

$$B = [\Phi(a_j) \leqslant \Phi(a_q')]$$

式中 a_q'—— 随机选出的 q 个个体。

那么，补事件 $\bar{B}$ 表示个体 a_j 劣于随机选出的 q 个个体，即

$$B = [\Phi(a_j) > \Phi(a_q')]$$

对于 q 内某一次比较而言，B 发生的概率 $p(B)$ 取决于 j 在 $(1,2,\cdots,2\mu)$ 中的位置，即

$$p(B) = \frac{2\mu - j + 1}{2\mu} = p$$

相应地，补事件 $\bar{B}$ 发生的概率 $p(\bar{B})$ 为：

$$p(B) = 1 - p$$

由于 q 竞争选择是在 2μ 个个体中任选 q 个个体再与 a_j 比较以确定 a_j 的得分 W_{a_j}，即

$$W_{jq} = \begin{cases} 1 & \text{If } \Phi(a_j) \leqslant \Phi(a_{qi}) \\ 0 & \text{其他情况} \end{cases}$$

因此，为了使得分 $W_{jq} = k$，需要 B 事件发生 k 次，B 事件发生 q k 次，再考虑到从 q 个个体中选取 k 个个体的可能组合，于是 $W_{jq} = k$ 发生的概率为：

$$p(W_{jq} = k) = C_q^k \cdot p^k \cdot (1 - p)^{(q-k)}$$

进一步，事件 B 在 q 竞争中发生的绝对频率 $H_{jq}(B)$ 也是一个随机变量，它服从参数为 q 和 $p(B)$ 的二项分布。对于相对频率 $h_{jq}(B) = H_{jq}(B)/q$ 而言，它也是一个同样的二项分布。因此，其数学期望值为：

$$E(h_{jq}(B)) = p$$

其方差为：

$$V^2(h_{jq}(B)) = \frac{1}{q} \cdot p \cdot (1 - p)$$

当 $q \to \infty$ 时，有：

$$\lim_{q \to \infty} V^2(h_{jq}(B)) = 0 \qquad j \in \{1, 2, \cdots, 2\mu\}$$

上式说明随着 q 的增大，$h_{jq}(B)$ 的方差不断减小；当 q 是够大时，$h_{jq}(B)$ 的方差可忽略不计，即没有离散。这个数学期望的最大值发生在最前面的个体上，即 $j = 1$，有：

$$E(h_{1q}(B)) = \frac{2\mu - 1 + 1}{2\mu} = 1$$

对于第 μ 个个体，有：

$$E(h_{\mu q}(B)) = \frac{2\mu - \mu + 1}{2\mu} = \frac{\mu + 1}{2\mu}$$

对于最后一个个体，即 $j = 2\mu$，有：

$$E(h_{2\mu q}(B)) = \frac{1}{2\mu}$$

因此，从 h_{1q} 经 $h_{\mu q}$ 至 $h_{2\mu q}$ 是一个单调递减的数列，说明前 μ 个个体得分的概率较大。这样，当 $q \to \infty$ 时，在 $(a_1, a_2, \cdots, a_\mu, a_{\mu+1}, \cdots, a_{2\mu})$ 群体中前 μ 个个体肯定被择优选取，换句话讲，当 $q \to \infty$ 时进化规

划中的q 竞争选择(随机选择)等同于$(\mu+\mu)$进化策略中的确定性选择。

为了验证上述理论分析，图 5-2 用数字模拟表示相对频率$h_{\mu q}(B)$ 与 j 的关系。试验中 $\mu=50$，即 $j\in\{1,2,\cdots,100\}$；$q=10$(左半图)或$q=100$(右半图)。图中实线表示试验得出的$h_{\mu q}(B)$值，总的来讲是从左上角至右下角的一条斜线(图中用点划线表示)，说明随着 j 的增大，B 事件发生的频率不断减小，即个体 a_j 不

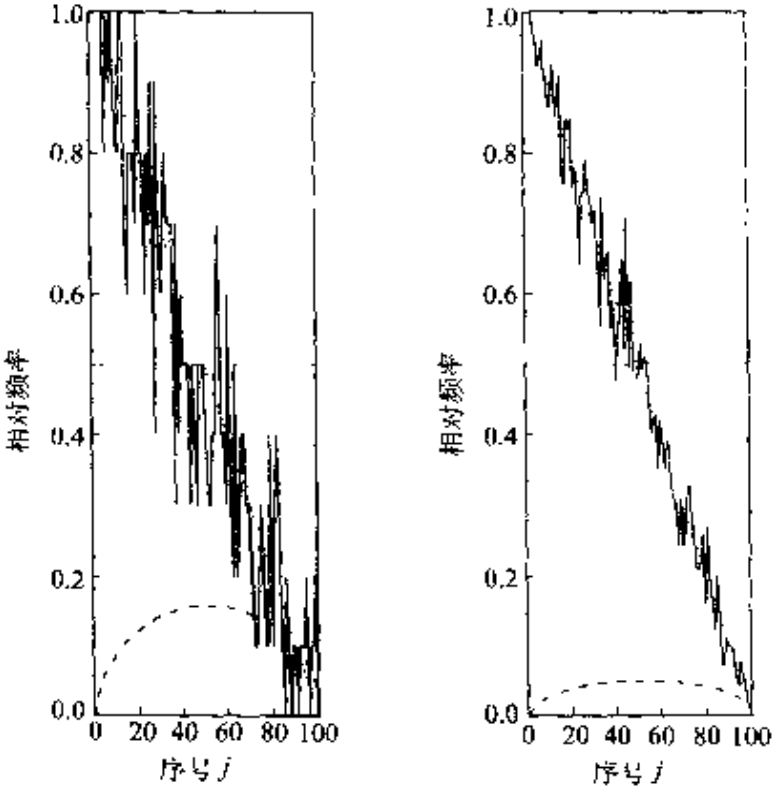

图 5-2　q-竞争选择的试验结果

易被选中。至于标准差：

$$\sigma_{h_{\mu q}(B)}=\sqrt{V^2(h_{\mu q}(B))}$$

它用短划线表示在图中底部。试验中$\sigma_{h_{\mu q}}(B)$的最大值恰好与一阶导数求出的最大值位置一致，即在 $j=\mu-1$ 时方差达到其最大值：

$$\max V^2(h_{\mu q}(B))=1/(2q)$$

此外，从图中可以看出，当 $q = 10$ 时（左图）$h_{pq}(B)$ 波动很大，个别劣质个体有可能被选中。相反，当 $q = 100$ 时（右图），$h_{pq}(B)$ 的波动较小，表明进化规划的选择方法趋近于进化策略的确定性选择。

5.4.2 收敛率

为了讨论进化规划的收敛机理，我们用最简单的标准进化规划说明。

在标准进化规划中，突变按下式执行：

$$x_i' = x_i + \sqrt{f(X)} \cdot N_i(0,1) \tag{5-10}$$

式中 x_i ——原目标变量 X 的第 i 个分量；

x_i' ——新目标变量 X' 的第 i 个分量；

$N_i(0,1)$ ——针对第 i 个分量产生的随机数，它服从标准正态分布；

$f(X)$ ——目标变量 X 的适应度。

若令标准差

$$\sigma = \sqrt{f(X)} \tag{5-11}$$

再令 $\mu = 1$，使原来的 q- 竞争选择变为确定性精英选择，于是上述的标准进化规划可视作（1 + 1）进化策略，可以借用进化策略的理论分析。

现针对球形问题，设目标函数为：

$$\min f(X) = \sum_{i=1}^{n} x_i^2 = r^2 \tag{5-12}$$

根据进化策略的式(4-17)，式(5-11)可写作：

$$\sigma = \sqrt{f(X)} = r\frac{\sigma_{opt} \cdot n}{1.224} \tag{5-13}$$

也就是说，由进化规划衍生的标准差 σ 总是比最优的标准差 σ_{opt} 大 $n/1.224$ 倍。因此，当目标变量 X 的分量数目 n 增加时，这种进化规划的执行效果肯定差于能使标准差维持在 σ_{opt} 的进化策略。

利用进化策略中式(4-25)有关收敛率的积分式，根据球状模型目标函数式(5-12)，可得出上述收敛率 φ 为（推导从略）：

$$\varphi = \frac{\sigma}{\sqrt{2\pi}}\exp\left(-\frac{1}{8}\left(\frac{\sigma \cdot n}{r}\right)^2\right) - \frac{\sigma^2 \cdot n}{4r}\left(1 - \mathrm{erf}\left(\frac{\sigma \cdot n}{r\sqrt{8}}\right)\right) \tag{5-14}$$

由于上述(1＋1)－ES(即简化后的进化规划)有$\sigma = r$,将它代入(5 14)式,可得进化规划的收敛率$\tilde{\varphi}$为:

$$\tilde{\varphi} = \left(\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{n^2}{8}\right) - \frac{n}{4}\left(1 - \mathrm{erf}\left(\frac{n}{\sqrt{8}}\right)\right)\right) \cdot r \tag{5-15}$$

从上式可以看出,进化规划的收敛率$\tilde{\varphi}$与r之间呈现线性关系,这与式(4-16)表示的φ_{max}与r之间的线性关系是一致的。将式(4-16)代入式(5-15),可得相对收敛率$\Phi(n)$:

$$\Phi(n) = \frac{\tilde{\varphi}}{\varphi_{max}} = \frac{n}{0.2025}\left(\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{n^2}{8}\right) - \frac{n}{4}\left(1 - \mathrm{exf}\left(\frac{n}{\sqrt{8}}\right)\right)\right) \tag{5-16}$$

图 5-3 表示上式中$\Phi(n)$与n之间的关系。从图中可以看出,$\Phi(n)$随n变化明显,有一最大值。对式(5-16)求最大值,即令:

$$\frac{d\Phi(n)}{dn} = 0$$

可得$n \approx 1.48$时$\Phi(n)$达到最大值。也就是说,当目标变量X的分量数目n为1或2时,进化规划的收敛率$\tilde{\varphi}$近似等于最优收敛率φ_{max},算法效率最高。随着n的增大,算法效率明显降低。

从上述分析可以看出,进化规划的收敛率$\tilde{\varphi}$对分量数目n很敏感,因此,当分量大于2时,Fogel建议将突变式(5-10)中适应度进行折减,即

$$x_i' = x_i + \sqrt{\beta \cdot f(X)} \cdot N_i(0,1) \tag{5-17}$$

式中 $\beta = n^{-2}$。

以球状模型为例,此时标准差σ为:

$$\sigma = \sqrt{\frac{f(X)}{n^2}} = \frac{r}{n} = \frac{\sigma_{opt}}{1.224} \tag{5-18}$$

从而使标准差σ不受n的影响,近似等于其最优值σ_{opt}。这时,进化规划的运算效果近似等于进化策略。

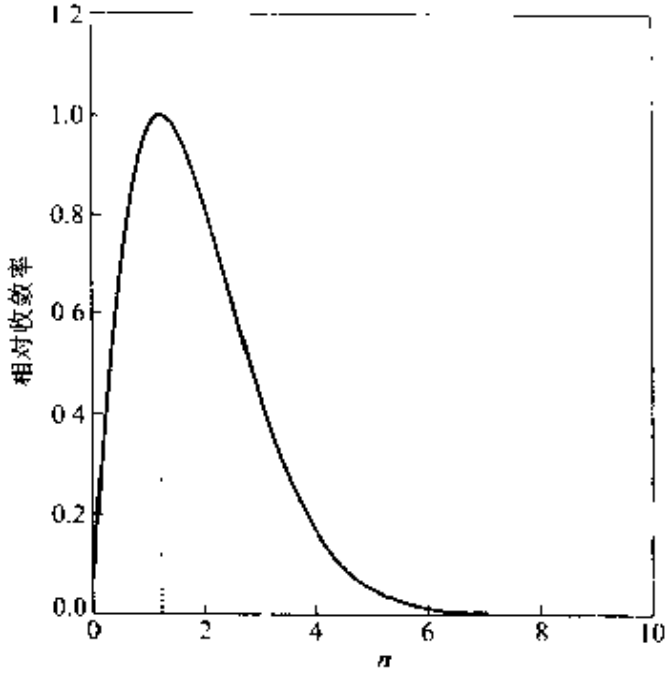

图 5-3　进化规划的相对收敛率

5.5　进化规划与进化策略的比较

进化规划和进化策略分别在美国和德国单独出现，随后又各自平行发展。这两种进化算法既有差别，又有相似之处。

5.5.1　相同

进化规划与进化策略都属于进化算法，它们同样仿效生物进化的过程，在优胜劣汰的竞争中不断进化，逐渐逼近最优解。

无论是进化规划或是进化策略，它们都是采用传统的十进制实型数表达问题。个体的表达都含有目标变量 X，反映方差的控制因子 σ 以及反映协方差的另一个控制因子 α 或 ρ。

进化规划和进化策略都通过突变使旧个体变为新个体，利用选择算子形成新一代群体。

总之，进化规划和进化策略都属于自适应搜索寻优算法，它们在搜索方式、搜索手段、搜索过程等方面都很相似。

5.5.2　差别

进化规划和进化策略的差别主要表现在：

(1) 重组算子的有无。进化规划没有重组算子，只依靠突变产生新个体。进化策略采用重组和突变两种手段产生新个体。关于重组和突变的作用一直存在争论，有人强调突变的作用，有人重视重组，还有人主张兼容二者。

(2) 选择的性质。两种算法都使用选择操作使旧群体进化为新群体。然而，进化规划采用随机型选择，进化策略则是确定型选择。前者用 q-竞争选择法使优良个体尽可能入选，但也允许少数欠佳的个体入选；后者用适应度排序的方法只允许优良个体进入新群体，劣质个体 100% 地被淘汰。

(3) 突变顺序。尽管两种算法都采用突变，但突变顺序不同。在进化策略中，首先突变控制因子 σ 及 α，然后才突变目标变量 X。在进化规划，先突变目标变量 X，后突变控制因子 σ 及 ρ，控制因子的突变效果滞后一代才起作用。

(4) 群体属性。进化规划中只使用突变，它是无性别的进化，使群体类于生物的种群，因为种群之间不存在性别交配。进化策略中使用重组，它属于有性别算子，因此产生同种的个体。

5.6 进化算法综述

进化算法包含遗传算法、遗传规划、进化策略、进化规划四种典型方法。它们在进化的原则上是一致的，但是在实施进化的方法上却各有特点，本节以表格形式列举它们的特点。

(1)表达方式。进化算法的表达方式见表 5-1。

表 5-1 进化算法的表达方式

算 法	表 达 方 式
遗传算法	常用固定长度的 0/1 字符串，也可用十进制数的字符串
遗传规划	用层次状的计算机程序格式，由函数及终止符表示
进化策略	十进制表示的实型数向量
进化规划	十进制表示的实型数向量

(2)初始群体的产生。进化算法的初始群体产生方法见表5-2。

表 5-2 进化算法的初始群体产生方法

算 法	初始群体的产生方法
遗传算法	随机从 0/1 中选取字符组成字符串
遗传规划	随机从函数集及终止符集选取结点组成层次状表达式
进化策略	从某一可行的实型数向量出发用突变产生初始群体
进化规划	从某一可行的实型数向量出发用突变产生初始群体

(3)适应度。进化算法的适应度见表5-3

表 5-3 进化算法的适应度

算 法	适 应 度
遗传算法	个体的性能指标，常作适当变换以便调节选择力度
遗传规划	常用归一化适应度，还有原始适应度、标准适应度、调整适应度
进化策略	个体的性能指标
进化规划	个体的性能指标

(4)遗传操作。进化算法的遗传操作见表5-4。

表 5-4 进化算法的遗传操作

算 法	遗 传 操 作
遗传算法	复制、交换、突变
遗传规划	复制、交换、较少的突变
进化策略	方差突变、重组、确定性选择
进化规划	方差突变、随机性选择

(5)复制(选择)算子。进化算法的复制(选择)算子见表5-5。

表 5-5 进化算法的复制(选择)算子

算　法	复制(选择)算子
遗传算法	随机性选择,限于从父代群体选取
遗传规划	随机性选择,限于从父代群体选取
进化策略	确定性选择,常从子代群体选取,也可包括父代群体
进化规划	随机性选择,从父代及子代群体中选取

(6)交换(重组)算子。进化算法的交换(重组)算子见表 5-6。

表 5-6 进化算法的交换(重组)算子

算　法	交换(重组)算子
遗传算法	字符之间的交换
遗传规划	子树之间的交换
进化策略	分量之间的交换
进化规划	无

(7)突变算子。进化算法的突变算子见表 5-7。

表 5-7 进化算法的突变算子

算　法	突　变
遗传算法	字符的逆变
遗传规划	结点的突变
进化策略	方差变化
进化规划	方差变化

参 考 文 献

1 Goldberg, D. E., Genetic algorithms in search, optimization and machine learning. New York: Addison-Wesley Publishing Company, Inc. 1989

2 Lawrence, D., Handbook of genetic algorithms. New York: Van Nostrand Reinhold. 1991

3 Koza, J. R., Genetic programming: on the programming of computers by means of natural selection. Cambridge: The MIT Press. 1992

4 Koza, J. R., Genetic programming Ⅱ: automatic discovery of reusable program. Cambridge: The MIT Press. 1994

5 Michalewicz, Z., Genetic algorithms + data structure = evolution programs. Berling: Springer—Verlag. 1996

6 Back, T., Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Ner York: Oxford University Press. 1996

7 Banzhaf, W., et al., Genetic programming—an introduction on the automatic evolution of computer programs and its applications. Morgan Kaufmann Publishers, Inc. 1998

8 Haupt, R. L. and Haupt, S. E., practical genetic algorithms. John Wiley & Sons, Inc. 1998

9 Schwefel. H—P., Evolution and optimum seeking. John Wiley & Sons, Inc. 1995

10 云庆夏，黄光球，王战权，遗传算法和遗传规划——一种搜索寻优技术，北京：冶金工业出版社，1997