EE6227号
遗传算法与机器学习第二学期

P. N. Suganthan博士

电子邮件：

epnsugan@ntu.edu.sg

电话：6790 5404

办公室：S2-B2a-21

介绍

- 教科书：
  - D. E. Goldberg，“搜索，优化和机器学习中的遗传算法”。
  - Andries Engelbrecht，“计算智能：入门”，Wiley，第二版或更高版本。
  - 注释/幻灯片中提供了其他参考。
- 该主题有两个作业。
- 评定：两份（每份两份）40%；期末考试60%。
- 笔记和演示幻灯片可从NTULearn获得。如果您无法访问它们，请给我发送电子邮件以通过电子邮件接收它们。

第1-7周的大纲

- **组合学与概率论的综述**：组合学排列和组合；样本空间和事件；可能性;条件概率贝叶斯规则；随机变量。
- **GA简介**：GA的说明；简要讨论优化和优化算法；简单的GA（SGA）；模拟;相似性模板（方案或架构）。
- **计算机实现**：总体结构如流程图；再生产;交叉和变异；主程序测试问题将目标函数映射到适应形式；编码;多变量问题。
- **功能优化**：历史应用；德容和功能优化；随机通用抽样；两点均匀交叉替代策略；例子。

第1-7周的大纲

- **GA的数学基础**：图式，基本定理；两臂和第K臂强盗问题；构建块假设；最小的欺骗性问题。
- 适应度缩放，基于实参的运算符，
- 实参进化算法，例如粒子群优化，微分进化等。
- 约束处理
- 多目标进化算法
- 多模式优化。

自然进化

- 正如查尔斯·达尔文（Charles Darwin）所述，自然进化的特征是适者生存。
- 这也意味着，比那些体弱的个体，适应性强的个体存活时间更长，繁殖出更多的后代。
- 当个体繁殖时，他们的遗传信息就会传给后代（或孩子）。
- 如果父母是更健康的个体，那么后代也很可能（但并非总是）也更健康。

现实生活中的一些观察

- 为什么某些物种灭绝（即在这个世界上再也找不到它们了）？不适合与更合适的物种竞争并生存……
- 古代国王的生活方式有很多...
- 研究方面的发展影响：
  - 农业和畜牧业的人工进化（基因工程）。
  - 减慢甚至逆转人类的自然进化。
- 其他观察，例如某些国家的社会保障，法律，医学进步等。
- 达尔文奖：　　**https://darwinawards.com/**

气体与进化

- 如查尔斯·达尔文（Charles Darwin）所描述的，遗传算法试图模仿（过度简化）物种的自然进化。
- 1970年代初，GA首次由J Holland及其同事在密歇根大学研究。
- 基于自然演化的方法的一个更通用的名称是进化计算/算法（EA）。
- 进化计算及其变体用于求解*困*难的优化问题。
- GA与生命科学完全无关！！

进化计算技术

- 加油站： 由J Holland和Ken DeJong博士于1970年代初首次提出
- 进化编程： L Fogel博士于1960年代首次提出
- 进化策略（可能是最古老的）
  - 由Rechenberg博士和Schwefel博士于1960年代初首次提出
- 基因编程：对于Alan Turing（1950年代），Richard Forsyth（1980年代），John Koza（1980年代）不断发展的计划。
- 在本模块中（第1-7周），我们将研究GA，粒子群优化（PSO），约束处理，演化策略（ES），差分演化（DE），多目标演化算法和多峰优化。
- 集合名称是“进化算法”或“群智能算法”。
- 近年来，有数百种受群体启发的算法。

采用（人工）演化进行优化的原理

- 自然进化的过程具有从种群中选择合适个体的能力。
- 以上选择更好/更适合的个人的过程与优化过程非常相似。
- 如果我们用总体表示特定问题的解决方案实例，然后将进化的人工版本应用于这组候选解决方案，则可以期望这种人工进化会使代的数量更好增加。

从自然进化到遗传算法

- 在现实生活中，我们可以观察到选择2个父母来繁殖后代以及后代的特征在某种程度上是随机的。
- 当进化计算试图模仿自然进化时，在进化计算算法中适当地（也经常）使用随机性。
- GA基本上是 *基于人群的引导随机搜索。这是因为：*
  - 我们与一群人一起工作，以代表要解决的问题的潜在解决方案
  - 搜索以进化论为指导
  - 繁殖后代的父母的选择和后代的特征在某种程度上是随机的。

用于优化的GA

- 总体中的每个字符串都代表要解决的问题的解决方案的一个实例。
- 对解决方案进行适当的编码，以便可以应用进化计算运算符，例如交叉，变异，再现等。
- 个体人口成员（即解决方案实例）的适合度反映了该解决方案对于要解决的问题的良好程度。
- 其他问题：人口中有多少成员（即人口规模）？有几代人进化种群？
- 稍后将详细讨论……。

优化介绍

- 优化是试图找到最好的过程
  或正在考虑的问题的最佳解决方案。
- 各个领域的大量问题可以表述为优化问题。
- 我们通常处理如下定义的有限维确定性问题：
  - 决策变量的描述（假设有n个变量），例如。 x_1，x_2，…x_n。
  - 对允许或允许或可行区域的描述
    决策变量，例如 ^{n}。通常 通过以下方式隐式定义
    功能关系称为约束或明确地由上/下限表示。
  - 一种衡量绩效的方法，称为目标函数。

优化介绍

目标函数可以描述如下：现在可以将问题描述 $f : \Omega \to \Re$ 如下：

最小化 $f(x)$ $x \in \Omega$
超过

请注意，最大化问题可以转换为 通过乘以–1 来最小化问题。

有离散，连续，混合（连续+离散）决策变量。

存在数值和组合优化问题。

离散优化

- 决策变量采用有限数量的离散值。
- 运输，分配和最短路径计算属于这种类型。
- 通过量化决策变量，我们可以通过离散问题来近似连续问题。
- 考虑一个简单的连续优化问题：
  - 最小化 $f(x) = x^2$ $\quad -1 <= x <= 1$
  - 例如，可以使用32位加一个符号位来离散化此问题。
  - x有2^{33}个可能的选择，离散解可以看作是连续问题的近似解。
  - 效率不高...最好使用实参数进化算法。

离散优化

- 可以通过穷举搜索决策变量的所有可能值来解决离散优化问题。对于实际问题，这不是合适的方法，因为它们可能是大量枚举并且可能具有复杂的约束。
- 因此，我们需要比显式枚举更有效的算法。
- 一些传统算法包括单纯形法，最短路径算法，最大流量算法等（请参阅 D. G. Luenberger撰写的“线性和非线性编程”一书，详细介绍了问题和传统的解决方法。

全球最优与计算成本

- 对于一些困难的优化问题，我们没有有效的算法来在合理的时间内找到最优解。
- 在实践中，在合理的时间内获得的良好解决方案要好于通过占用过多时间和/或计算资源而获得的全局最优解决方案。
- 在这些情况下，可以使用启发式方法，例如GA或模拟退火。这些方法利用随机性来逃脱局部最优解。
- 启发式方法没有牢固的数学基础（与数学编程方法相比）。
- 没有任何方法可以保证产生全局最优解（对于复杂的多峰问题）。

非线性规划问题

- 用连续的非线性函数（通常是可微函数）描述客观和等式/不等式约束的问题可以通过微积分法解决。一个一般的非线性规划问题：

$$\text{最小化} \quad f(x), \quad x \in \Re^n$$

$$\text{服从} \quad h_j(x) \quad j = 1,2,\ldots, m$$

$$0$$

$$g_i(x) \quad 0 \qquad i = 1,2, \ldots, p$$

- 通常很难通过解析来解决实际的非线性规划问题。通常，开发迭代数值算法以获得可能产生或可能不产生全局最优解的解。

非线性规划问题

- 最受欢迎的方法是顺序二次编程。 （查看Matlab优化工具箱中的几种优化算法）。
- 大多数常规方法需要目标函数和约束函数的连续性。有时，也可能需要一阶导数的连续性。
- 这些是限制性条件，可能无法始终满足。
- 也有称为直接搜索方法的无导数自由优化算法（无进化类型）。

GA的优势

- 从概念上讲　简单的算法。
- 随着使用大量解决方案，GA给出了许多（不同的）解决方案。替代/多种解决方案在某些应用中很有用，例如计划，调度，多目标优化等。
- GA不需要将搜索空间设置为
  - 连续
  - 可微的
  - 单峰（即没有局部最优解，只有全局最优）。
- 也可能不需要基于显式方程的目标函数。可以进行实验以获得目标值。
- 我们需要规则来比较两种解决方案，以确定哪个更好。不同的规则可能对相同的两种解决方案的排序不同，
  这不是问题！！

遗传算法和传统优化算法

- GA会使用大量解决方案实例，而不仅仅是搜索空间中的单个点/解决方案。当需要多种解决方案时，人口是一大优势。
- GA使用的是概率转换规则，而不是大多数传统算法所使用的确定性规则。
- 遗传算法直接作用于目标函数，而不作用于其梯度，近似目标或其他辅助信息。
- GA通常是对编码变量进行操作，而不是直接对决策变量进行操作（诸如PSO，DE，ES等实参演化算法可以直接对决策变量进行操作，通常效果更好）。
- 当决策变量为二进制时，GA的效果会更好。

术语：天然和人工

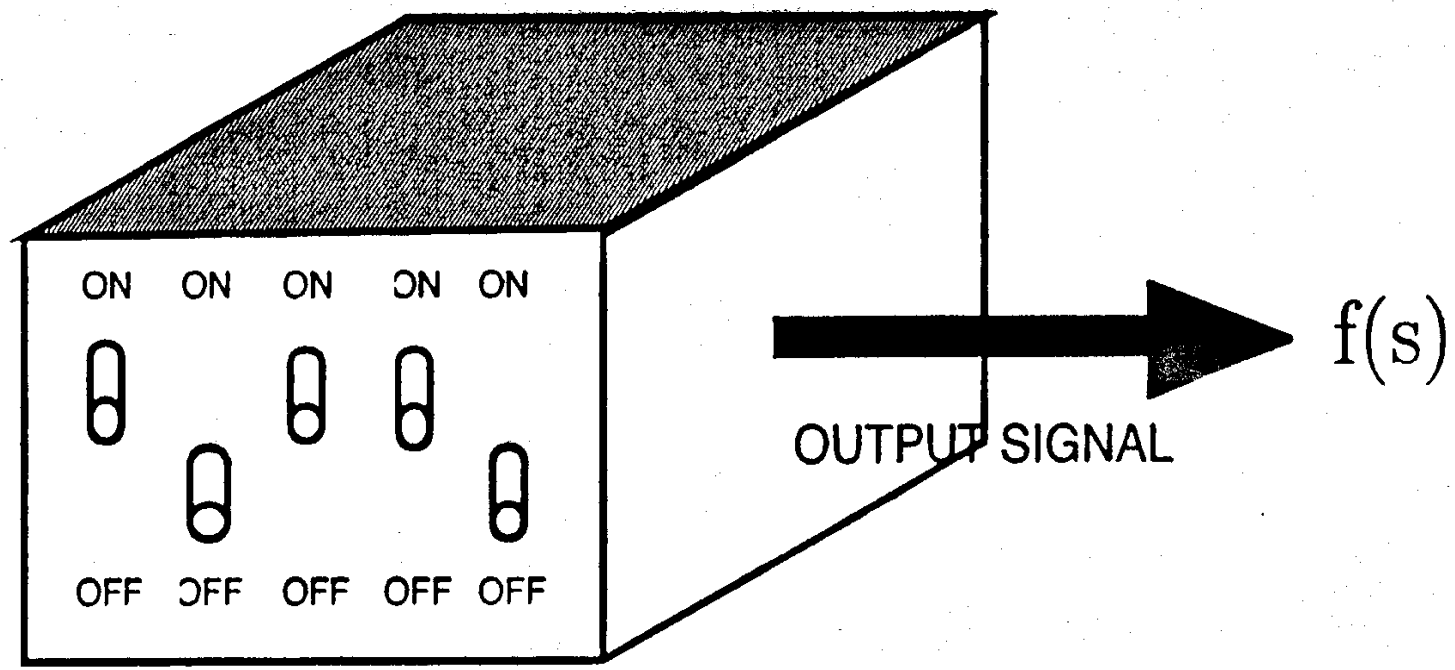
自然	遗传算法
染色体	字符串/解决方案实例
基因	特征/字符/检测器
等位基因	特征值
轨迹	弦位置
基因型	结构体
表型	参数集，替代解决方案，解码结构

不必记住所有这些术语。

近来，进化算法已经不同于自然进化。

简单遗传算法（SGA）：简单问题和解决方案编码

- 让我们考虑一个简单的黑匣子切换问题。
- 有5个输入开关，对于每个开关设置，都有一个输出信号f（s）。
- 目的是设置开关以获得最大输出信号值“ f（s）”。
- 开关位置为“开”或“关”。我们可以用1表示“开”，用0表示“关”，即解决方案的编码。
- 表格11110的特定解决方案表示前四个开关为“开”，而最后一个开关为“关”。


ON ON ON ON ON
OFF OFF OFF OFF OFF
f(s)
OUTPUT SIGNAL

SGA公司：初始人口

- 我们需要产生一个初始种群。我们还应该决定人口数量。
- 总体上，各个世代的人口规模都保持不变。
- 让我们假设我们在人口中选择4个成员。
- 初始种群通常是随机产生的，例如。通过抛硬币20次（4个成员，每个5位长）。或调用随机数生成器20次：如果rand <0.5　0。如果兰特
  > = 0.5　1，这里0 <= rand <= 1.0。
- 假设初始人口为01101，　　11000，01000，10011
- 通过实验，您可以观察到初始化在最终解决方案中的（显着）影响（尤其是当总体很小且决策变量数量少时…）。

SGA公司：初始人群的适应性（客观）值

- 通常，下一步是计算随机生成的初始解字符串的适应性（或目标）值。
- 通过将每个解决方案字符串替换为目标函数，可以轻松完成此操作。
- 在这里，没有必要，因为我们的黑盒给出的输出也是目标函数。

柱：1 2 3 4 5

Number	String	Fitness	% of Total	No. selected
1	01101	169	14.4	1
2	11000	576	49.2	2
3	01000	64	5.5	0
4	10011	361	30.9	1
Total		1170	100.0	4

Table 1.1 Sample problem strings and fitness values

SGA公司：进化与遗传算子

- 现在，我们定义简单的遗传算子并将其应用于初始种群，从而将初始种群演化为后代，同时期望种群的总体适应水平提高。
- 基本的遗传操作是：
  - 复制/选择（适用于当前人口）
  - 分频（适用于再现操作的结果）
  - 变异（应用于交叉操作的结果）
- 这些运算符按给定顺序依次应用，以获得下一代解决方案字符串（在SGA中）。

SGA公司：再生产

- 复制操作员会考虑到各个解决方案字符串的适用性或客观价值，将当前一代的解决方案字符串复制到配对池中。
- 具有较高适应性值的字符串可能会在交配池中以较高的数量表示，从而将其遗传信息传递给后代。
- 该运算符是达尔文自然选择过程的人工版本。
- 交配池通常与人口规模相同。

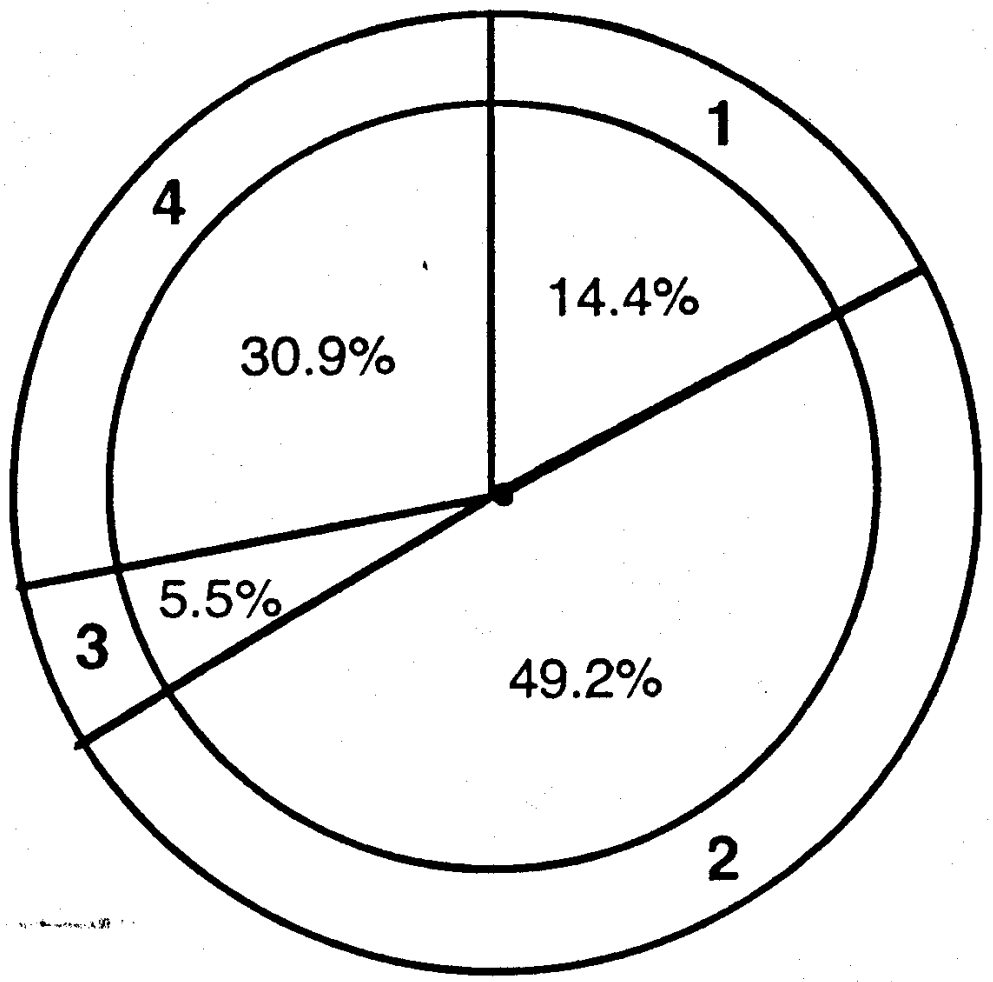
SGA公司：实施复制

- 有几种替代方法可以实现复制运算符。在SGA中，我们将使用轮盘赌法。我们稍后再讨论其他一些。
- 在这种方法中，每个字符串都根据其适应性水平分配了车轮周长的一部分。对于此选择方法，适应性值应为+ve。
- 然后旋转轮子n次以选择n个字符串到交配池中，n是种群大小。在我们的简单示例中为4。
- 在我们的简单示例中，假设已经从黑盒的输出中观察到适应度值并且它们为正。

SGA公司：轮盘赌轮选择

- 假设初始总体具有适应性值（第3列）以及适合度百分比值（第4列），如表1.1所示（幻灯片26）。
- 轮盘赌轮与百分比适应值成比例地显示分配给总体中每个字符串的区域。
- 接下来，将轮盘旋转4次，以从配对池中选择4根琴弦（表1.1中第5列显示了一个特定的选择）。

SGA公司：轮盘



^

| is这是标记

SGA公司：轮盘赌选择的计算机实现

- 我们可以使用随机数（0到1之间）生成器来模拟轮盘赌的操作。
- 串联的适合度百分比值：
  - 字符串1：在0和0.144之间
  - 字符串2：在0.144和0.636之间
  - 字符串3：在0.636和0.691之间
  - 字符串4：在0.691和1.0之间
- 让生成的随机数为：0.127, 0.282, 0.832, 0.496。因此，选择的字符串：字符串1，字符串2，字符串4和字符串2。

SGA公司：交叉操作

- 交叉算子将应用于所选的交配池。根据表1.1，以下字符串位于匹配池中：01101， 11000, 11000, 10011.
- 在2之间进行交叉 交配池中的父串，并产生2个后代。
- 交叉概率p_c可用于确定2个父字符串之间是否存在交叉。
- 从交配池中随机选择2个（通常是不同的）父字符串。
- 交叉操作交换每个字符串中的部分信息，因此后代具有父母双方的不同部分。在此步骤中，也存在随机性的元素。
- 如果种群为n，则重复n / 2次以获得n个后代。

SGA公司：交叉操作

- 在执行交叉操作之前，我们必须决定是否进行交叉。
- 通过使用交叉概率p_c做出此决定。
- 由于我们将执行n / 2次交叉，因此我们可以生成一个随机矢量，其中n / 2个元素分别在0和1之间。
- 如果生成的随机数小于p_c，则执行交叉操作。否则，将不执行任何交叉操作。
- 如果未执行交叉操作，则将选定的父代从帧开始传递，由变异算子处理。

SGA公司：交叉操作

- 在确定是否发生xover之后，将按以下方式执行点交叉：
  - 沿字符串1 <= k随机选择一个整数位置k <= L-1，其中L是字符串的长度。在我们的示例中，有4个潜在的单点交叉点：0.1.1.0.1
  - 通过在位置k分解父字符串并交换信息来生成两个新的字符串（后代）。
- 如果两个父字符串是01101和11000，整数交叉位置k是4，那么交叉操作的结果是2个后代：**0110**<u>**0**</u>和11001
- 在上面的示例中，一个字符串带有下划线，而另一个则带有粗体，这只是为了提高对交叉操作的理解。

繁殖和Xover的有效性

- 尽管复制和交叉是简单的操作，但它们可以导致强大的搜索机制。
- 稍后我们将调查这种情况。

- 观察：在此SGA版本中，父母不大可能不变地出现在下一代中。现代的进化/遗传算法保留了下一代的一些最佳解决方案。

SGA公司：突变

- 一般而言，突变在GA中的作用很小。
- 在我们的示例中，适当的突变操作是将生成的后代字符串中的位值从“ 0”随机更改为“ 1”，或从“ 1”更改为“ 0”。
- 突变会给种群带来一定程度的多样性，防止过早收敛，并有助于对搜索空间的未探索区域进行采样。
- 突变操作的概率p_m很小，例如。 0.001、0.01。一条经验法则：
  ∽每个新解决方案一个位。
- 通常，随着世代数的增加，突变概率逐渐降低。在早期阶段，有必要对整个搜索空间进行采样 需要更高的突变。在后面的阶段中，这只是一个很好的搜索- 需要进行少量修改才能不干扰高质量的解决方案。

SGA公司：实施变异

- 总体中有4个字符串，每个字符串的长度为5位 4 * 5，或者在生成的后代中总共为20位。
- 根据突变概率（以下示例中为0.1）随机选择用于突变的位位置。
- 我们可以生成0到1之间的20个随机数。如果生成的值小于或等于突变概率，则可以翻转该位置处解决方案字符串中的位值。
- 例：让x叠加后的后代为：01100、11001、11011、10000。
- 随机数：0.26,0.34,0.46,0.59,**0.05**,0.27,0.87,0.91,0.79, 0.61,**0.09**,0.34,0.4,0.81,0.65,0.86,0.11,0.43,0.62,0.69.
- 突变后代：0110**1**, _ 11001, **0**1011, 10000

替代方案和启发式决策

- 通常，每个步骤都有几种替代方法，例如。复制，种群替换，解决方案表示/编码，交叉，将目标映射到适应度值等。
- 根据试探法或经验或经验法则或反复试验来选择最大代数，人口规模，突变/异化几率等。
- 在接下来的几周中，我们将学习其中一些替代方法。最好的学习方法是自己进行计算机仿真！
- 通常，研究解决特定问题的传统方法是一个好主意，以便在基于GA的搜索中寻找良好选择时有一个好的开始。有时，混合方法（传统方法与GA方法的混合物）效果很好。

SGA继续…示例2：f（x）= x^2

- 我们考虑最大化的问题 $f(x)= x^2$.
- 假设x是0到31之间的整数。
- 接下来，我们将详细介绍SGA，并重点介绍6个主要步骤。

SGA-步骤1：对决策变量进行编码

- 在这个简单的问题中，我们只有一个整数变量，限制在0到31之间。
- 显然，编码将使用5位，如前面的黑盒示例所示。

SGA-步骤2：初始人口

- 如前所述，初始种群是随机产生的。让我们再次使用相同的初始人口。

SGA步骤3：计算健身比例

- 将字符串转换为整数，并评估f（x）。
- 表1.2中显示了（初始）总体，解码后的决策变量x，适应度f（x），适应度比例。

SGA-步骤4：复制

- 我们可以使用轮盘赌获得交配池（如表1.2的第7列所示）

SGA-步骤5：交叉

- 第一　步骤是选择n / 2对父字符串进行交叉。随机选择的对显示在“配合”下的第8列中。
- 然后，在1和4之间随机选择一个点的交叉点。在第9栏中显示为“ C”交叉点。
- 交叉操作的结果在第10栏中给出，称为“ New Popn”。

1	2	3	4	5	6	7	8	9	10	11	12
String No.	Initial Popn.	x	$f(x)$	% of Total	No. Sel.	Mating Pool	Mate	C'over Site	New Popn.	x	$f(x)$
1	01101	13	169	14.4	1	01101	2	4	01100	12	144
2	11000	24	576	49.2	2	11000	1	4	11001	25	625
3	01000	8	64	5.5	0	11000	4	2	11011	27	729
4	10011	19	361	30.9	1	10011	3	2	10000	16	256
Sum			1170	100.0	4						1754
Average			293								439

Table 1.2 Simulation of a simple GA

SGA步骤6：变异

- 在这里，我们假设突变概率为0.001。总数为20位。因此，0.02位应发生突变。
- 因此，没有位突变，最终结果（下一代种群）显示在第10列中。

第一代后的适应度值

- 解码字符串后，我们可以计算新总体的适应度值。
- 解码后的值和适应性值在表1.2的第11和12栏中显示。
- 我们可以观察到，由于初始种群中最好的字符串在交配池中出现了两次，因此平均适应度已从293提高到439。
- 此示例说明了GA的过程。

SGA的演变与终止

- 通常，步骤1和步骤2在SGA开始时仅执行一次。
- 重复步骤3、4、5和6，直到满足终止条件为止。
- 可能的终止条件可能是
  - 给定的世代数（或许多适应性评估）已完成
  - 最佳解决方案或集合的适用性（即优化问题的目标值）已达到可接受的水平。
  - 在几代人中，适应性的顺序增加可以忽略不计。
  - 以上条件的组合。

相似性模板和架构

- 接下来，我们将看到Google Analytics（分析）如何有效执行搜索。
- 从表1.1可以看出，第1位的“ 1”位与较高的适应性值相关。
- 这表明第一 “ 1”位具有有用的信息，GA可以方便地使用这些信息。
- 我们利用相似性模板或模式（复数形式为模式或模式）来解释GA的搜索过程以及其如何利用此类信息。

相似性模板和架构

- 模式是相似性模板，描述了在某些字符串位置具有相似性的字符串子集。
- 我们考虑带有通配符*的二进制字母{0, 1}。
- 现在，我们使用字母{0，1，*}创建字符串。
- 例如，模式* 101 *表示子集{01010、01011、11010、11011}。
- 使用模式，我们可以表示各种字符串之间的相似性。

相似度模板

- 如果二进制字符串的长度为5，则有3个^{5}-1不同的相似性模板。这是因为我们可以在5个位置中的每个位置使用0或1或*（不包括仅*的字符串）。
- 如果字母表中有k个符号（而不是0和1），则存在（k + 1）^{l}-1个模式。
- 显然，模式太多而不是实际字符串的总数。架构仍然提供了一种有用的方式来分析GA的工作原理。

相似度模板

- 长度为5的二进制字符串（例如10111）是2^5模式的成员，因为任何特定位置都可以用*代替。
- 长度为1的二进制字符串包含2^1方案。
- 因此，取决于种群的多样性，大小为n且长度为1的二进制字符串的种群可能具有介于2^1和$n.2^1$方案之间的某个位置。
- GA可以有效地处理其中多少种模式，以提高总体总体适应水平？
- 为了回答这个问题，我们必须研究各种遗传算子（繁殖，杂交，突变）如何影响重要图式的生长和衰退。

相似度模板

- 复制的效果很简单：高度适合的架构会多次复制到配对池中。
- 要查看交叉效果，请考虑以下两种模式：1 *** 0和** 11 *。
- 显然，第一模式比第二模式更容易被单点交叉干扰。
- 据说1stschema具有较大的定义长度。
- 突变率通常很低，并且不会以很大的方式破坏模式。

相似度模板

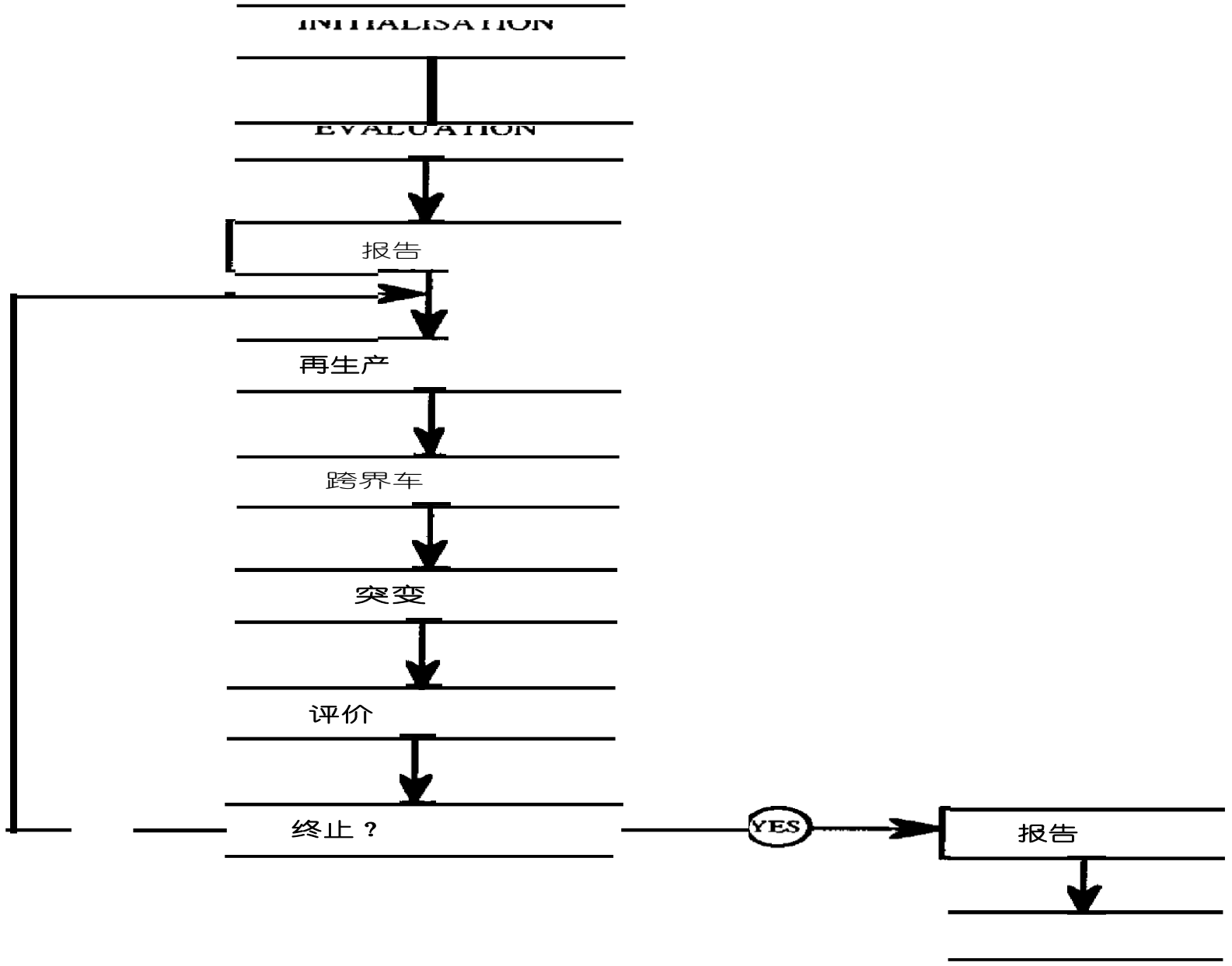
- 观察发现，定义长度短的高度适合的方案将以越来越多的数量传播。
- 尽管GA并未明确考虑架构，但此过程是在GA处理总体时进行的。
- 这些定义长度短的高度适合的架构称为构建块。
- 在每一代中有用处理的模式数量大约为n^3，其中n是总体大小。
- *这称为隐式并行。*

第二章计算机实施

- 我们使用Matlab代码实现SGA。
- 我们考虑两个简单的问题：
  - 1个无符号整数变量编码为无符号二进制的简单函数的优化。
  - 限制在指定值之间的2个变量的函数的优化。

下一张幻灯片显示了我们的总体结构

辛普勒GA



停

数字 2.1 一般 简单的结构 佐治亚州

初始化

- 为了初始化总体，我们需要
  - 解决方案候选中的位数称为“磅”
  - 候选解决方案的总数，即人口规模（称为“ popnsize”）
- 通常使用随机初始化（如果有先验知识，还有其他几种选择）
- 如果需要离散化，并且变量是实数值。在这种情况下，可以根据所需的精度来计算“磅数”。
- “ popnsize”的选择会影响计算成本和总体多样性

初始化

函数popn = initialise (popnsize, lbits)

%产生初始种群

Popn=rand (波森大小, lbits) < 0.5;

% 结束初始化

评价

- 下一步是评估总体中每个候选解决方案的适用性。
- 候选解决方案必须转换回无符号整数才能评估适用性。
- 我们可能想知道最大适应度，平均适应度，最佳当前解决方案候选者等。
- 这些统计信息可以保存在文件中，以供日后查看或作图，以检查GA仿真运行的特性和收敛性能。

评价

功能　　[xpopn, fitness, xopt, maxf, meanf] =评估 (popn, lbits)

%求值函数返回当前总体的适应度。

%1st此函数调用Decode () 和Objfun () 函数。

xpopn =解码 (popn) ; Fitness =

Objfun (xpopn, lbits) ; meanf =

sum (适合度) / size (popn, 1) ;

[maxf, imax] = max (适合度) ; %imax是最合适的指标。

xopt=xpopn(imax)　　xopt%是当前这一代的最佳价值

%评估功能结束。

%稍后我们将介绍“解码”和“ Objfun”功能。

再生产

- 复制使用轮盘赌选择方法。
- 选定的字符串保留在“选择”中–交配池。
- 交配池是随机重新排序的。
- 在交叉阶段，第一个字符串与第二个字符串匹配，依此类推。

再生产

功能[配对，选择]=复制（弹出，适应性）

%重现功能使用轮盘选择

% 产生交配配对。“选择”包含

% popn中每个字符串的数量已经

% 添加到交配池。 normfit =适合度/总和

（适合度）；partum = 0; randnums =

rand（size（fitness））；

计数（1）=0; 配合池=;

```
对于I = 1: 长度（适合）partum = partsum +
normfit (i) ; count (i + 1) = length (find
 (randnums <partsum) ) ; select (i, 1) = count
 (i + 1) -count (i) ; matepool = [matepool;
ones (select (i, 1) , 1) * popn (i, :)];结束

%接下来，对字符串重新排序，以使字符串1匹配
%与字符串2，依此类推。

[垃圾，交配] = sort (rand (size (matepool, 1) ,
1) ) ; matingpairs = matepool (mating, :);
%复制结束功能。
```

交叉

- 如前所述，我们使用简单的单点交叉。
- 交叉应用于由“复制”功能生成的有序配对池。
- 交叉站点是随机选择的。
- 交叉站点的数量是人口总数的一半。
- 发生交叉的概率为p_c。

交叉

```
函数offspring = Crossover (popn, pc) lbits =
size (popn, 2) ; sites = ceil (rand (size (popn,
1) / 2,1) * (lbits-1) ) ;
site = sites * (rand (size (sites) ) <pc) ;交叉概率百分比
对于j = 1: length (sites) ,
后代 (2 * j-1, : ) = [popn (2 * j-1,1: sites (j) )
popn (2 * j, sites (j) +1: lbits) ];
后代 (2 * j, : ) = [popn (2 * j, 1: sites (j) )
popn (2 * j-1, sites (j) +1: lbits) ];
结束
%交叉功能结束。
```

突变

* 仅一位被突变。
* 随机进行变异的概率为 p_m.

```
函数newpopn = Mutation (后代, pm) mutate =查找 (rand
   (size (后代) ) <pm) ;
```

突变百分比具有待突变基因的位置。

```
newpopn =后代;
newpopn (mutate) = 1-后代 (mutate) ;

%突变结束功能。
```

报告

- 此例程可用于监视GA的进度，并获得人口演变的统计数据。
- 可以修改此例程以适合不同应用程序的需求。

报告功能 (gen, popn, xpopn, Fitness, Mean, fhistory, maxfhistory, xopthistory)

```
disp(' '); disp('    '); disp(' ');
disp([sprintf('Generation %0.5f', gen)]);
disp('   ');
```

```
disp(['        串                x     健身']);
popnstring =字符 (48+ popn); %为i = 1: size (popn, 1)
压缩矩阵,                                    %当前输出popn统计信息。 disp
([popnstring (i, :) sprintf (%16.8g%16.8g', xpopn (i),
fitness (i) )]); 结束
disp(' ');
disp('     健身历史统计数据");
disp ('Generation Mean Fitness Max Fitness Optimal x);
history = [[0: gen]'的均值历史maxfhistory xopthistory];
disp ([sprintf ('% 5.0f % 16.6g % 16.6g \ n',
history')]);

%功能结束报告。
```

通用航空发展的终止

- 在指定的世代之后，GA可能会终止
- 另一种可能性是几代人的平均或最大适应度值饱和。

其余模块

- 我们将必须编写“解码”功能和“ Objfun”。
- 两者都是特定于问题的。
- 在我们的示例中，我们将单个变量编码为无符号整数。

解码功能

函数xpopn =解码（popn）

%解码功能可转换人口

从二进制到整数的字符串的百分比（弹出）

%假设变量为非负数。

```
lbits = size (popn, 2) ; twopowers = 2。^
 (磅1: -1: 0) ; xpopn = popn * twopowers″;
```

%解码结束功能。

目标函数-Objfun

- 我们考虑以下目标函数：
  $f(x)=(x/c)^{10}$.
- c是缩放常数，用于避免获得太大的函数值。
- 如果lbits = 30&c = 2 ^ 30-1;最佳f (x) = 1时
  $x=2^{30}-1$.

对象功能

```
函数适应性= Objfun (xpopn, lbits)
%  Objfun评估目标/适合性
%  解码种群的值
%  此示例为f (x) = (x / c) ^ 10, 其中
%  c=2=lbits-1;

c=2=lbits-1;
适合度= (xpopn / c) 。^ 10;

%Objfun结束
```

主程序

```
函数[xpopn, fitness, meanf, maxf, xopt] = onevarSimpleGA
    (popnsize, lbits, pc, pm, numgens)
%onevarSimpleGA优化简单功能
1个变量的百分比，编码为无符号二进制
    %整数meanfhistory
= []; maxfhistory = [];
xopthistory = [];
%产生初始种群
gen=0;
popn=初始化 (popnsize, lbits) ;
```

%获得初始人群的健身统计数据，并 %保存历史记录参数以在报告中使用

[xpopn, fitness, xopt, maxf, meanf] =评估 (popn , lbits) ;xopt 历史 [xopt 历史;xopt];最大史=最大史;最大历史;最大史;最大史平均历史[平均史;平均史];

%要求报告初始人口

%和初始人口统计

报告 (gen, popn, xpopn, fitness, meanfhistory, ma xfhistory, xopthistory) ;

对于gen = 1: numgens　　% 主世代循环

繁殖，杂交和变异

matingpairs = 复制（popn，fitness）；后代 = crossover（matingpairs，pc）；popn =突变（后代，pm）；

%获得当前人口的统计数据

%并保存报告的历史记录参数。

[xpopn，fitness，xopt，maxf，meanf]=评估（p opn，磅）；

xopthistory=[xopthistory;xopt];

maxfhistory=[maxfhistory;maxf];

meanfhistory=[meanfhistory;meanf];

结束　　% 主世代循环

主程序

```
%致电报告以打印出最终结果
人口百分比和
所有年龄段的人口统计百分比
gen=数量;
报告 (gen, popn, xpopn, fitness, meanfhist ory,
   maxfhistory, xopthistory) ;
xopt = xopthistory; maxf =
maxfhistory; meanf =平均历
史;
%varSimpleGA结尾
```

运行测试问题

- 最*大化f（x）=（x／c）^10，c=2^30-1*
- *x*编码为无符号二进制整数。
- 必须指定一些参数值：
- *popnsize=30, lbits=30, pc=0.6, pm=0.0333.*
- 当lbits = 30时，解空间中有2^30个点。因此，即使对于这个简单的问题，穷举搜索也几乎是不可能的。

结果观察

- 随机初始化给出了一个适合度为0.980787的好候选者。
- 在10代中，最佳解决方案出现在第7代中，适应性为0.994064。
- 最好的解决方案随后在下一代中被交叉，突变等破坏。
- 每次运行都可能给出不同的最佳解决方案。
- 现代的GA / EA保留了几代人以来最好的解决方案。

代 0

串	x	适合度
01111110101101010111011000111O	5.3145537e=0B	0.0008B240779
10011111100100100010011000001	6.6928883e 0B	0.00B8539506
01000101011100101011111100l010	2.9128699e+OB	2.15B7986e-06
10011100010011010110101000l110	6.5557979e 0B	0.00719B7464
01010111001010011000101l101001	3.65S8513e=0B	2.0935653e-05
01000110000110001010110010011l	2.9400554e 0B	2.36B9532e-06
10110000110011111100110100011l1	7 4159886e 0B	0.02469971
10110110110001100100011010010O	7.6661188e+08	0.03441571
01111100000001101001001011110O	5.202014e+操作系统	0.00071238785
10001001000011101100111010111l	5.7486226e+08	0.001934B141
00001011101001110001100001001O	48875026	3.B183112e-14
01110110100111111000101001001O	4.97S41？塞奥布	0.00045634873
00001111100100011011000011111l	65301567	6.922091Se-13
01010110011000001000001001010O	3.622913硒+ 0B	1.9124097e-05
10110000010011000000101001011l	7 .3944335e+08	0.023991112
10011110000101010110010001100l	6.6305052e=0B	0.0080624589
·00000 10 1000 10 1110 11100 110000 11	21355715	9.6859555e-1 ***B***
01101011101101000001110011001l	4.5174149e+08	0.00017374052
11001011000111011110000011100O1	8.S193125e 0B	0.098865961
01000001010110011000001100010O	27 409632e=0B	1 _174988e-06
11010111110010111001100001110l	9.0511107e+OB	0.18114238
10011110001011011100110000001O	6.6345037e=0B	0.0081112112
01000111010111000101110100000l	2.9930886e=0B	2.8326773e-06
10000110011110010110111010100O	5.6402628e+OB	0.001 59953B5
11111111100000001111101101000O	1.071660Be=09	0.9B078668
10001001001110110010010011010l	575S8B66e+0B型	**0.001**9594023
11100011000011100111111111101	9.52344秒？e+0B公司	0.30126066
00110011101000110010101101001l	2.1658287e+08	1.1149164e-07
10001110000011001001001100010l	5.957971 9e+08	0.002766B512
10010001111100110010011000011l	6.1215783e+OB	0.0036277337

健身历史统计平均值健身

第0代	最大健身	最佳X
0.056385	0.980787	1.07166e+09

健身功能

- SGA要求适应度值为正，以便能够执行轮盘赌的选择或复制。
- 如果目标函数是-ve，则将目标函数映射到+ ve适应度函数。
- 为了最大化，如果目标函数u（x）可以取 - ve值，则将适应度f（x）定义为：

$$f(x) = \begin{cases} u(x) + C_{\text{分钟}} & \text{如果u（x） } C_{\text{分}} > 0 \\ 0 & \text{除此以外} \end{cases}$$

- *Cmin公司*可以在初始或评估之后分配一个值，例如，最小\u（x）的绝对值（即–ve）。

健身功能

- 对于最小化问题，给定g（x）是目标函数，适应度可以定义如下：

$$f(x) = \begin{cases} C_{\text{最高}} - g(x) & \text{如果} C_{\text{最高}} - g(x) > 0 \\ 0 & \text{除此以外} \end{cases}$$

- 在此，可以为Cmax初始分配一个值，或将其分配给当前总体中g（x）的最大值。
- 需要上述映射，才能将较小的目标值转换为较大的适应度值，并对轮盘进行编程。
- 轮盘赌轮有很多替代方法，不需要上述映射。

健身缩放

- 简单的GA可能会表现出不良的行为，因为在第1代的第一代中，个体非常适合主导选择。
- 这可能导致过早收敛。
- 经过几代人之后，人口的最大适应度和平均适应度将相似，从而导致本质上是随机的选择和搜索。
- 我们可以采用适应度缩放来解决这些问题。

线性适应度缩放

- 给定非负适应度f（x），缩放函数为：

$$g=af+b$$

- 使用试探法获得a和b的值
  - 缩放前后保持平均适应度值

    $g_{avg}=f_{avg}$.
  - $g_{max} = Cf_{avg}$，通过设置以下值来限制最合适的候选人在交配池中最多拥有2个样本

$C=2.$

$$a = \frac{f_{平均}(C \quad -1)}{f_{平均}} \qquad b = f_{平均}(1 \quad a)$$

$f_{最大}$

  - 2个方程式和2个未知数…

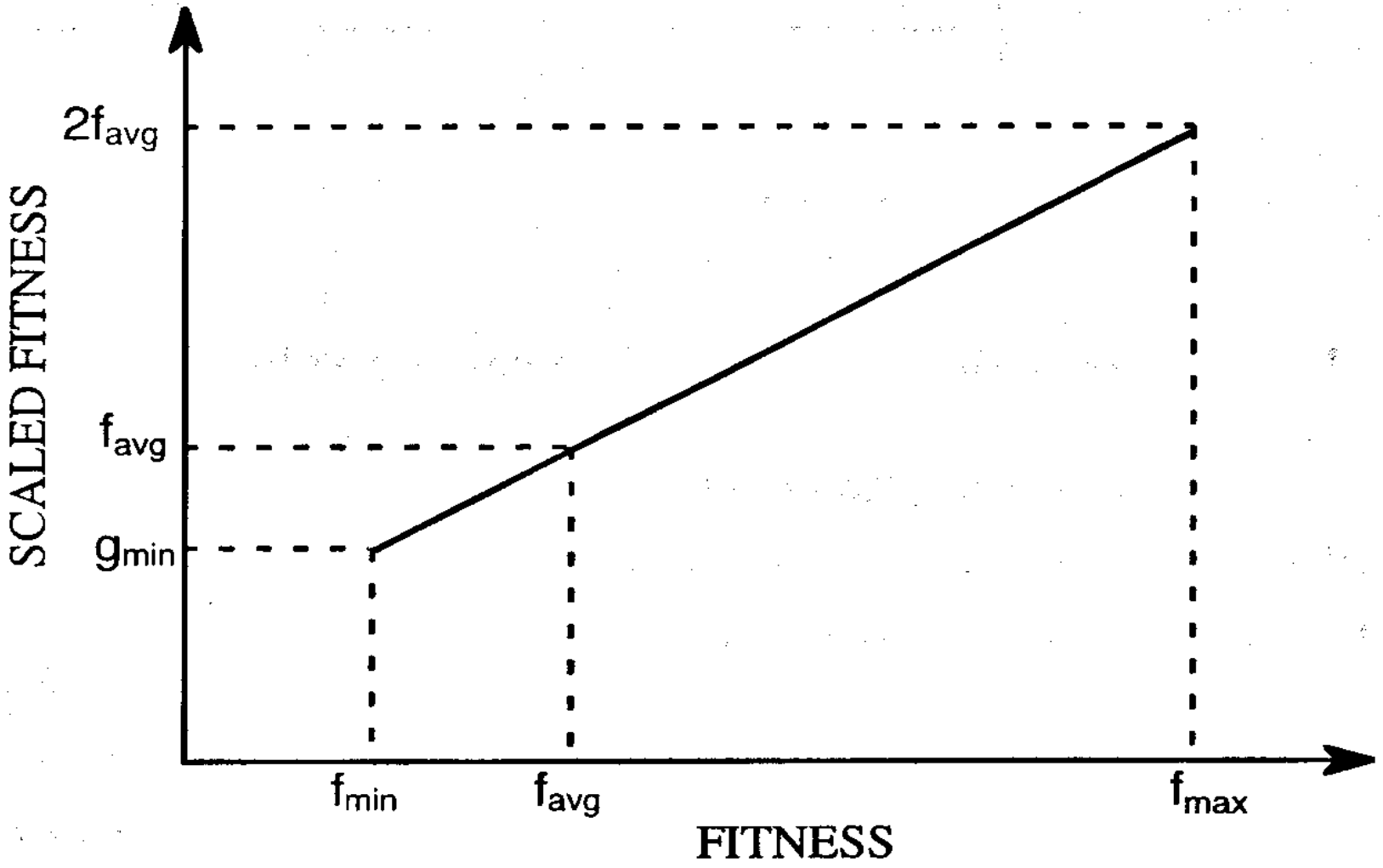


Figure 2.2 Linear scaling under normal conditions

健身缩放

- 当许多人具有与最佳人相似的适应度值时，缩放可能会在以后的世代产生–ve适应度。
- 一种可能的解决方案是在缩放前后将平均值保持不变，但将最小适应度映射为0，而不是将最大适应度按C缩放。

$$a = \frac{f_{平} \; b = f}{f_{平均} - f_{min}} \qquad (1 - a)$$

均　　平均

- 适应性缩放在“复制”操作中完成，仅用于选择父母。

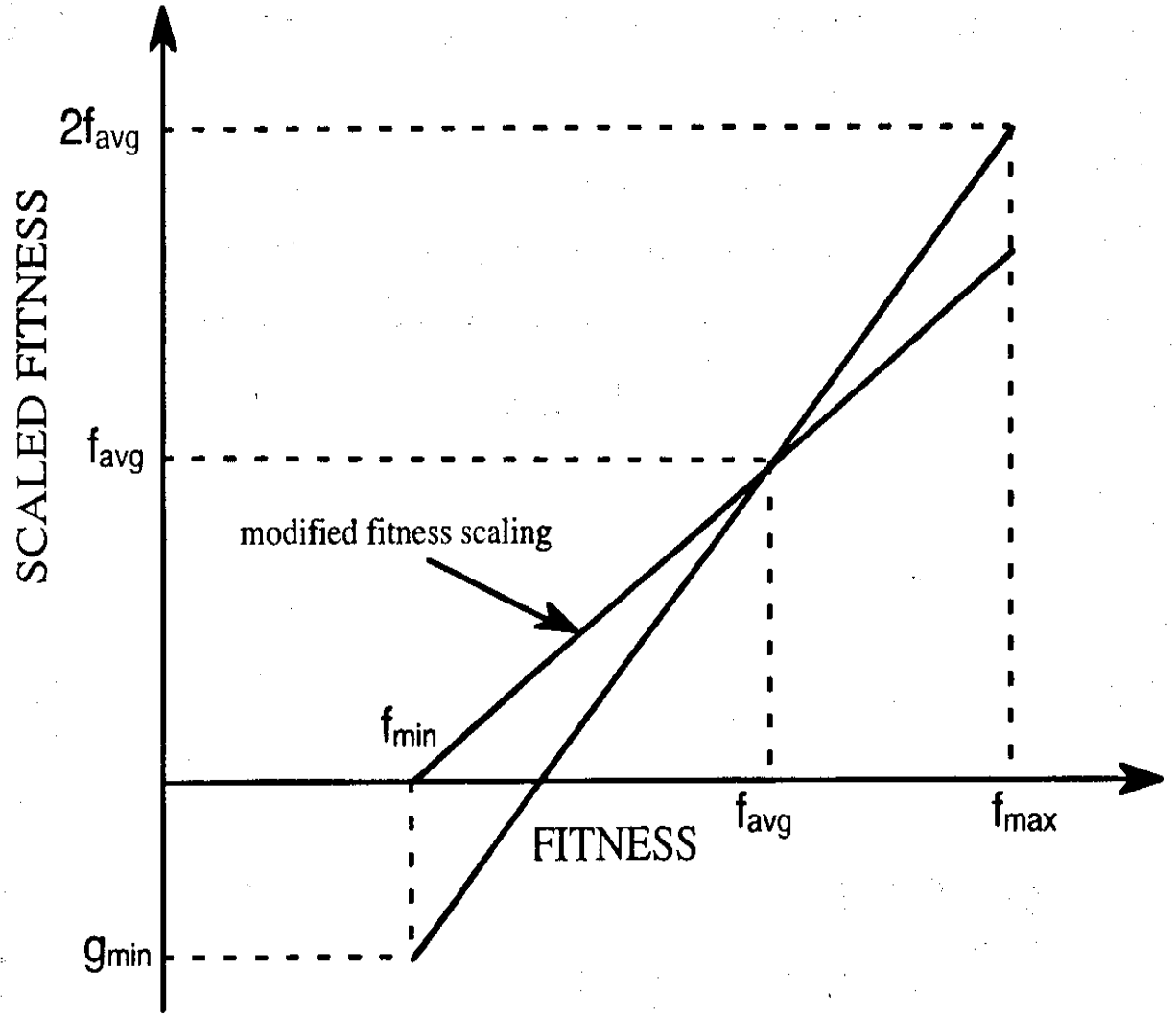


Figure 2.3 Linear scaling with negative scaling values and the modified function

```
函数scfitness = Scalefitness (fitness, fmultiple)
%线性缩放适应度。返回结果
% 在理智上。首先计算参数a和b favg = sum
 (fitness) / length (fitness) ; [fmax, i] =
max (fitness) ; [fmin, j] = min (fitness) ; a
= favg * (fmultiple-1) / (fmax-favg) ; b =
favg * (1-a) ;
如果a * fmin + b <0 a =
 favg / (favg-fmin) ; b
 = favg * (1-a) ;
结束scfitness=a*适合度+b*个 (大小 (适合度)) ;
Scalefitness功能的结束。
```

Sigma截断

- 在应用线性缩放之前可以使用此方法
- 利用适应度值的方差和平均值，如下所示： $f'=f-(f_{平均}-c\sigma)$

  其中$f_{大道}$是适应度平均值，σ是方差，f'是sigma截断后的适应度。 c是通常设置在2-3左右的比例因子。
- 任何-ve f'设置为零。
- 线性缩放可以应用于f'。

幂律定标

- 缩放后的适应度f'是+ ve原始适应度f的幂，例如：

$$f' = g(f) = f^{\alpha}$$

.

- alpha应该适当地选择：
  - 如果0⩽f ＜1，并且希望增加间隔，则0 ＜α＜1。
  - 如果1 ＜f，并且您希望增加适应度值之间的间隔，则1 ＜α。
- 在早期，可能有必要减少适应度值的变化。
- 更接近终止，可能需要增加适应度值的变化。
- 现代的GA / EA尝试避免使用原始适用性值。取而代之的是，大多数情况下使用排名代替轮盘赌轮选择方法。

编码注意事项

- 有些问题可能具有自然编码。
- 通常，有许多可能的编码方案。
- 选择合适的编码：
  - 优先考虑与问题相关的简短的低阶模式
  - 使用最小数目的字母（二进制为0和1）。
- 第一个目标通常很难实现。我们可能会尝试对编码字符串重新排序。
- 第二点自然导致二进制编码–最小的字母。
- 在GA中，如果决策变量是0-1二进制，则上述准则有效。
- 如果决策变量是浮点数或整数，则为二进制GA可能不是使用的最佳算法...

编码多变量问题

- 考虑编码决策变量

.

$x = (x_1, x_2, ..., x_m)$ 其中 $\leq x_i \leq b_i$

- 每个组件将被编码为长度为1_i的二进制字符串。
- 长度取决于所需的精度。
- 如果我们要求变量的小数点后5位精度 x_i，则选择1_i作为最小整数满意的 $10^5(b_i - a_i) \quad 2^{里} - 1$

多变量问题

- 使用此编码，解码为：

$$x_i = a_i + \text{十进制}(\text{字符串}_i)\frac{b_i - a_i}{2^{l_i} - 1}$$

- 简单的GA必须修改：
  - 输入lbit现在是一个向量，该向量具有分量l_i，i = 1，…，m
  - 解码功能必须更改。
  - main函数应该能够具有上限和下限形式的其他输入，如下所示：

函数[xpopn, fitness, meanf, maxf, xopt] = multivarSimpleGA
(popnsize, lbits, pc, pm, num gens, vlb, vub)

```
函数xpopn =解码 (popn, lbits, vlb,
 vub)
%从中转换字符串的填充量 (popn)                    二进制到真实。
%总字符串长度为sum (lbits)      与m = length (lbits) 子字符串
% 每个为   x_1, ..., x_m变量。 1解码每个子串
%无符号十进制              整数: 辛特

指数1=1;        index2 = 0;
对于我= 1: 长度 (磅)
 index2 = index2 + lbits (i) ; twopowers = 2。^ (磅 (i)
 -1: -1: 0) ; xint (: , i) = popn (: , index1: index2) *
 twopowers'; index1 = index1 + lbits (i) ;
结束
```

解码功能

现在计算x值factor = (vub-vlb) ./ (2. ^ lbits-1) ;

xpopn = ones (size (popn, 1) , 1) * vlb + xint * dia

g (因子) ;

%解码结束功能

%多个变量

❖对于多变量情况，可能还必须更改某些其他功能。

例

- 考虑以下问题：

最大化 $f(x_1, x_2) = 21.5 + x_1 \sin(2 x_1) + x_2 \sin(10 x_2)$

服从 $-3.0 \le x_1 \le 12.1$ 和 $4.1 \le x_2 \le 5.8$

- x_1和x_2是优化问题的决策变量。
- 假定所需的4个小数点的精度为11 = 18和12 = 15。
- *popn=10， pc=0.5， pm=0.01个数=10* （ GA的参数）

- 通过将numgens增加到100左右可以获得更好的结果。

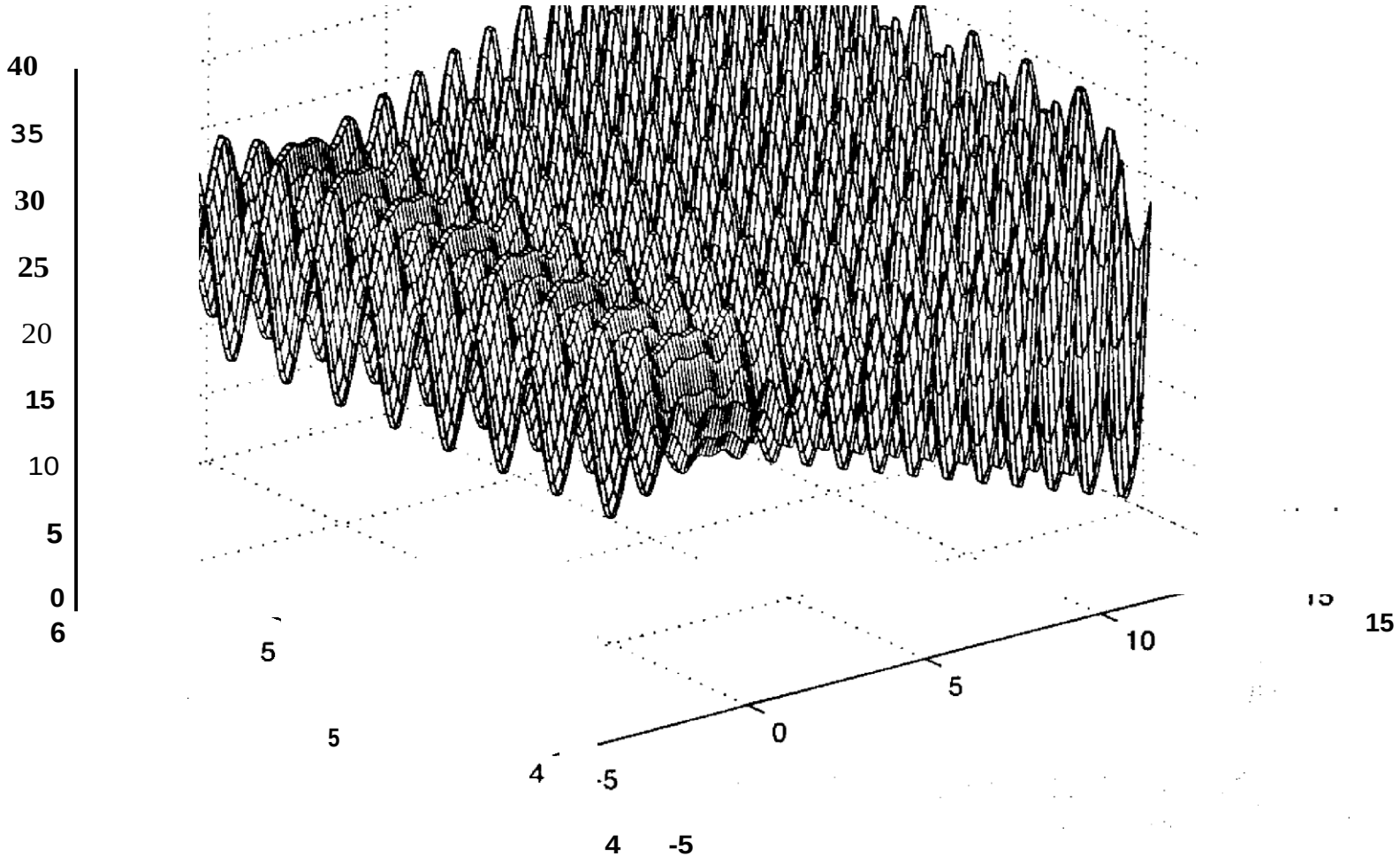


图2.4　　的网格图　　函数f（xi，x2）

进一步调查

- 研究突变率，交叉，比率等对所得溶液质量的影响。
- 需要使所有变量保持恒定，除了被调查的变量之外，在某些范围内逐渐变化。
- 需要评估和获取有关每个特定参数集的几种不同随机初始化的统计信息。
- 对于不同的参数集，请保持相同的初始化。

功能优化

- 对“简单GA”的改进
- 功能优化

选择过程的改进

- 如果人口很小，选择轮盘赌可能不会产生预期的数字-　　这是相当普遍的。
- 健壮的个体有可能以更高的比例支配　导致过早趋同。
- 或者，可以使用随机通用选择（SUS）。
- 在SUS中，使用n个标记在轮子上旋转轮子一次。这n个标记将给出n个字符串。给好

　　内近似　　±1.

- SUS仍然需要正适应度值，每个适应度值的实际大小会影响选择。

随机通用选择

- 通过替换该方法可以轻松实现

  randnums = rand (size

  (fitness)) ;由rr = rand;

  间距= 1 /长度 (适合度) ;

  randnums=排序 (mod (rr: 间距: 1+rr-

  *0.5 *间距, 1)) ;*

排名选择

- 排名选择是另一种选择，其中根据适应度值对总体进行排序，并为每个字符串分配后代数。
- 如果我们首先对总体进行排序，则与k日排序的字符串的选择比例为：

$$p_k = q_{\text{最大}} - \left(\frac{k-1}{n}\right)(q_{\text{最高}} - q_{\text{分}})$$

$q_{最高}$和$q_{分}$分别是最佳字符串和最差字符串的分配。 $q_{最高}$和$q_{分}$满足以下关系，因为比例之和应为1： $q_{最高}+q_{分}=2/n$

- 将这些比例转换为概率后，可以按这些比例使用SUS。
- 精确的适应度值（+ ve或 -ve） 不使用/不需要。

比赛选择

- 此方法有多种变体。
- 在一个版本中，随机选择2个（或更多）字符串，比较其适应性值，最好的一个包含在交配池中，并将所有字符串返回给总体。
- 重复此过程，直到选择了必要数量的字符串。
- 排名和比赛选择比轮盘赌选择方法更好。

分频器的改进

- 交叉算子是GA的主要区别特征。
- 它的主要功能是组合各种构件。
- 简单GA中使用的单点交叉有一个严重的局限性–它不能组合所有可能的模式。
- 例如，1点交叉不能将11 ****** 1和**** 11 ***组合为11 ** 11 ** 1。

1点和2点交叉

- 1点交叉能够组合简短的低阶模式。但是我们可能事先不知道什么位顺序将功能相关的位组合在一起。
- 一种可能的解决方案是使用两点交叉。
- 两点交叉不太可能破坏长模式。

两点交叉

- 2点交叉可以比1点交叉组合更多模式。
- 例如，将2点交叉应用于以下两个模式，其中2个随机选择的交叉站点分别位于4和8：

  11 ** | **** | * 1和

  **** | 1 * 1 * | **给 11 ** 1 * 1 ** 1和

  **********作为后代模式。
- 但是，有些架构甚至无法通过两点交叉进行组合。

均匀交叉

- 统一的交叉具有组合几乎所有方案的能力。
- 没有交叉站点，相反，对于子代中的每个位位置，我们随机决定贡献哪个父代：

父母1　　1001011

父母2　　*0101 01*

模板：　　1101001

后代1　　1001101

后代2　　*0101011*

随机生成的“模板”，具有相等的概率为0和1。

- 遵循2中的下划线（父母1）和斜体（父母2）。后代与模板中的1和0有关。

均匀交叉

- 对于均匀的交叉，字符串中相关位的位置（模式定义长度）并不重要。
- 1点和2点交叉可能会保留紧凑编码的良好属性。
- 很难说哪个交叉是最好的。这取决于问题和编码。
- 通常，对于人口较少的人群，均匀交叉比较合适。
- 如果人口很大，那么两点交叉可能就足够了。
- 也可以在早期使用Uniform交叉，然后在最后阶段转移到2点交叉。

人口替代的改善

- 在简单的GA中，一整代人都被后代所取代。
- 有时称为非重叠世代，
  即，父母与子女之间没有重叠（与自然更相似）。
- 某些先前的结果表明，有时候后代的表现可能会比其父母差。
- 主要缺点是，父母中的一些良好构件可能会丢失，并且将来不会对进化过程有所帮助。
- 在实践中很少使用非重叠方法。

重叠的世代与精英

- 提出了几种人口替代方法，主要是为了使新的和旧的人口（即后代和父母）重叠。
- 有时，当前这一代的最佳适应度值可能会比过去几代减小。
- 精英策略通过将最好的或最好的几个父母复制到下一代中来随机替换最坏的孩子或某些孩子来解决此问题。
- 尽管精英策略可能会导致一两个适合的人占主导地位，但在大多数情况下，它会改善简单GA并在人口规模为大。

稳态复制

- 即使采用了精英主义策略，许多最佳个体也可能无法繁殖，而且基因可能会丢失。
- 因此，在每一代中只替换少数种群成员-称为稳态繁殖。
- 产生了少量后代，它们取代了人口中弱小的父母。
- 稳态再现通常在不断发展的基于规则的系统中使用，其中增量学习非常重要。

选拔

- 在这个计划中，父母和后代争夺下一代的生存权。
- 后代和 父母被选为父母下一代。
- 如果 是 很小，那就是精英策略。
- 另一种特殊情况是 和 .

一个优点是我们可以具有较高的突变概率，而缺点是该方法无法在变化/动态的环境中很好地执行。

德荣与功能优化

- 德容在功能最小化问题领域做了一些开拓性的工作。
- 函数F1在原点x = 0处具有最小值。
- F2（Rosenbrock的香蕉函数）的全局最小值为（1,1）。
- F3是3D步骤，最小值为 -15。
- F4具有接近原点的全局最优值x = 0，其中 $N(0,1)$ 是0表示1方差白噪声。
- F5有25个深度相似的非常尖锐的槽。 $a_{ij公司}$在[-65.536，65.536]之间随机选择。

Number	Function	Limits
F1	$f_1(\boldsymbol{x}) = \sum_{i=1}^{3} x_i^2,$	$-5.12 \leq x_i \leq 5.12$
F2	$f_2(\boldsymbol{x}) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2,$	$-2.048 \leq x_i \leq 2.048$
F3	$f_3(\boldsymbol{x}) = \sum_{i=1}^{3} \text{integer}(x_i),$	$-5.12 \leq x_i \leq 5.12$
F4	$f_4(\boldsymbol{x}) = \sum_{i=1}^{30} i x_i^4 + N(0,1),$	$-1.28 \leq x_i \leq 1.28$
F5	$f_5(\boldsymbol{x}) = 0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6},$	$-65.536 \leq x_i \leq 65.536$

Table 4.1 Test functions for minimisation

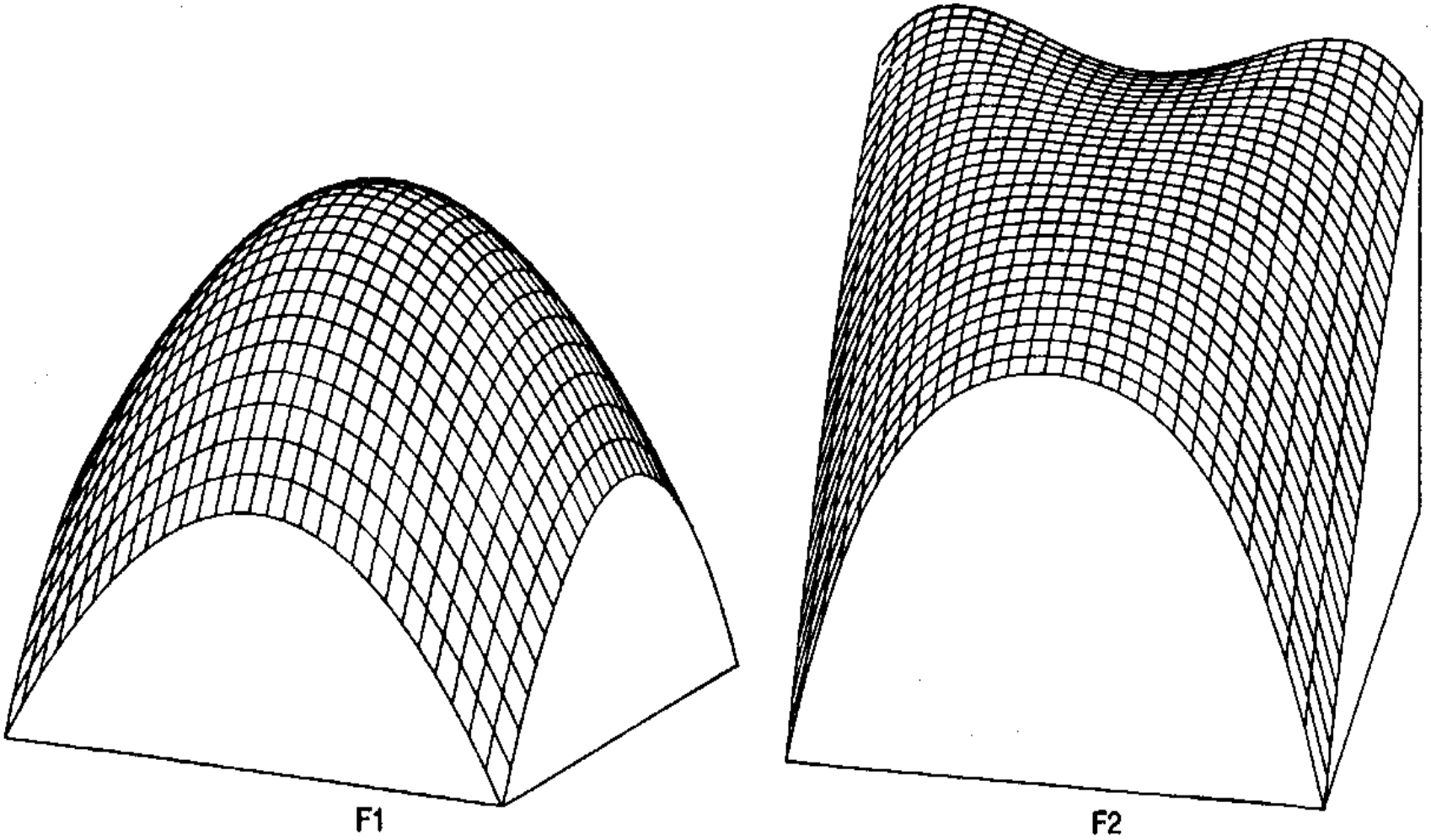
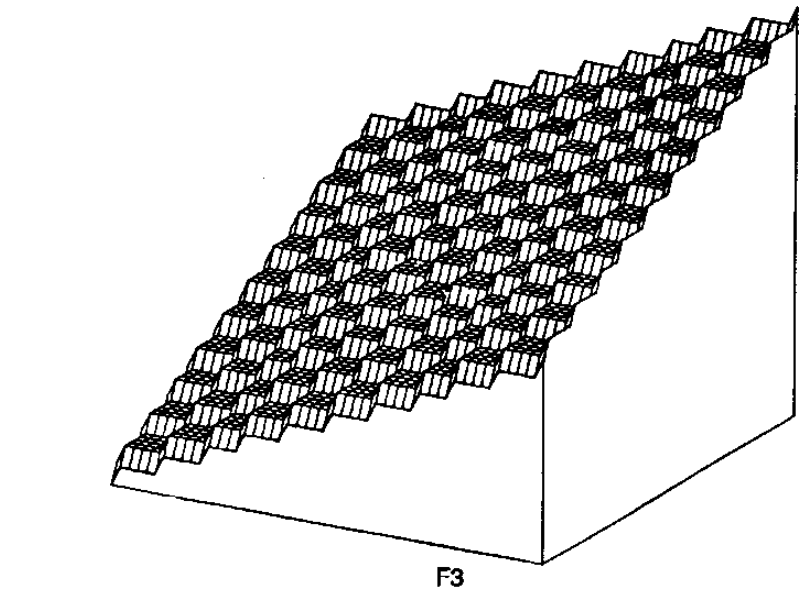


FIGURE 4.9 Inverted, two-dimensional versions of De Jong's (1975) test functions *F*1 and *F*2. Reprinted by permission.



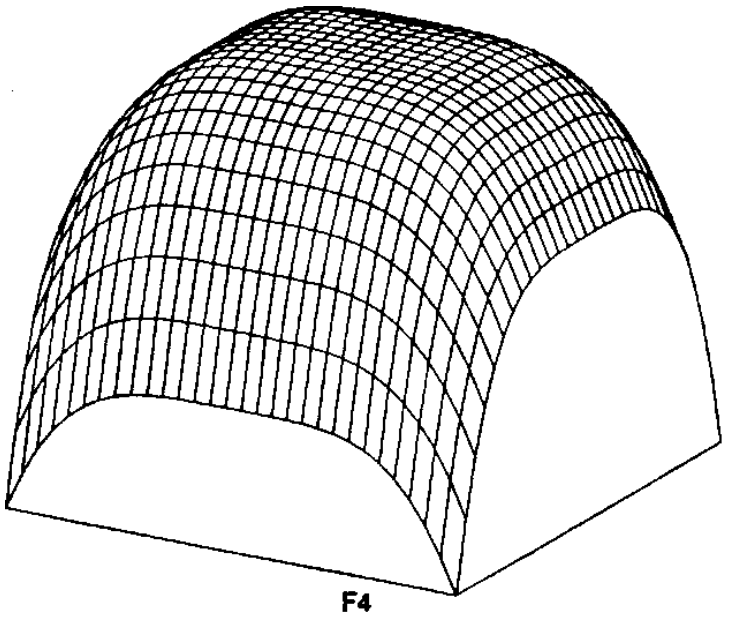


FIGURE 4.10 Inverted, two-dimensional versions of De Jong's (1975) test functions *F*3 and *F*4. Reprinted by permission.

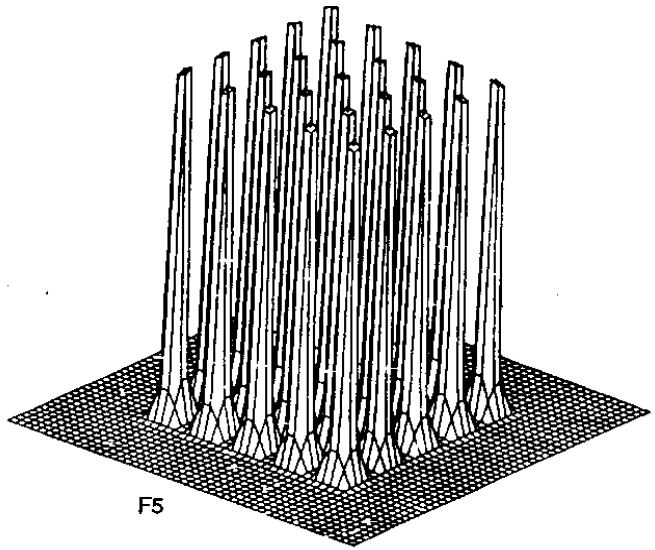


FIGURE 4.11 Inverted version of De Jong's (1975) test function *F5*. Reprinted by permission

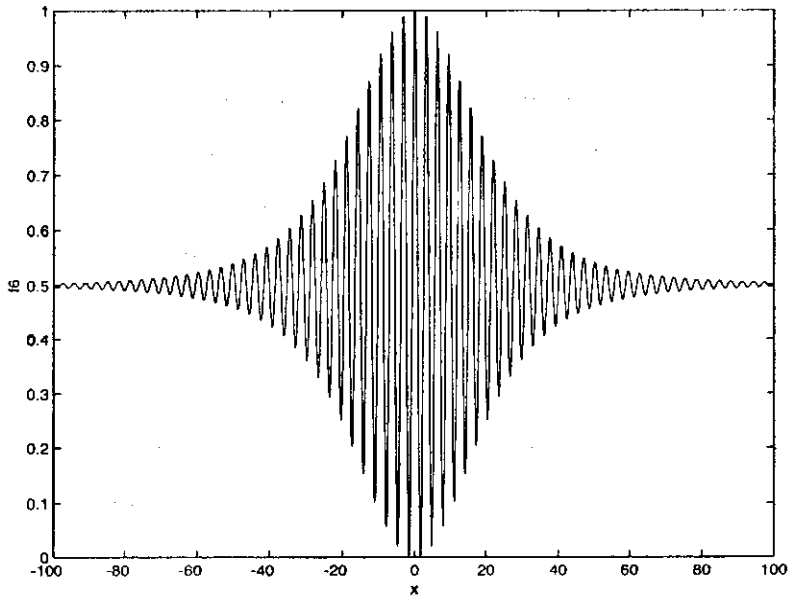


Figure 4.1 The test function $f6$ when y is constant $(y = 0)$

德荣与功能优化

- 所有功能都是积极的。
- 我们应该否定将最小化转化为最大化问题的函数。
- 我们应该添加一个偏移量以使适应度+ ve。
- 德钟考虑了2个绩效指标：

$$J_{在线} = \frac{1}{T}\sum_{t=0}^{T} \bar{f}(t)$$ [$\bar{f}$（t）主题适应性 在t一代]是

$$J_{离线} = \frac{1}{T}\sum_{t=0}^{T} f_{最大}(t)$$ [$f_{最高}$（t）最大健身 在t一代]是

现在不常用“在线”和“离线”这两个名称。

实验结果

- 结果在扩展注释中给出（第6页）。
- 首先测试了简单GA。然后，改进交叉，选择，人口置换等以获得更好的结果。
- 与几种传统的非线性优化方法相比，遗传算法的效果很好　　多峰函数的算法。
- 在单峰函数上不太好。 GA不应成为单峰目标函数的选择。
- 您可以运行Matlab程序并进行更多探索！

戴维斯的功能优化

- 戴维斯的职能：

$$f^6(x,y) = 0.5 - \frac{(\sqrt{x^2+y^2})^2 - 0.5}{(1+0.001(x^2+y^2))^2}$$

- 它有几个山丘，最大值为（0,0），最大值仅占总面积的一小部分。
- 给出了几种情况的结果：1.简单GA（表4.2）； 2.具有适应性缩放的SGA（T。4.3）； 3.具有线性缩放和均匀Xover的SGA（T 4.4）； 4.具有SGA的SGA。
  精英替换者（T 4.5），带有5. SGA ＋ (T 4.6 &4.7）。表编号是指扩展注释。

Fitness History Statistics

Generation	Mean Fitness	Max Fitness	Optimal x	
0	0.503697	0.867592	4.64513	-11.7565
1	0.507463	0.867592	4.64513	-11.7565
2	0.514698	0.820131	-15.0977	-4.47166
3	0.498547	0.820131	-15.0977	-4.47166
4	0.501326	0.821075	-15.0977	-4.40943
5	0.502437	0.821075	-15.0977	-4.40943
6	0.500878	0.921811	-8.65238	-3.71592
7	0.513566	0.921811	-8.65238	-3.71592
8	0.52791	0.921811	-8.65238	-3.71592
9	0.545438	0.921811	-8.65238	-3.71592
10	0.54032	0.921811	-8.65238	-3.71592
11	0.578093	0.921811	-8.65238	-3.71626
12	0.555383	0.921811	-8.65238	-3.71626
13	0.595485	0.921811	-8.65238	-3.71626
14	0.63703	0.921811	-8.65238	-3.71626
15	0.649057	0.921811	-8.65238	-3.71511
16	0.6869	0.960978	-2.63069	-5.74782
17	0.707684	0.960978	-2.63069	-5.74782
18	0.742898	0.921811	-8.65238	-3.71459
19	0.745931	0.921811	-8.65238	-3.71473
20	0.740784	0.921811	-8.65238	-3.71473
21	0.765929	0.921811	-8.65238	-3.71473
22	0.789503	0.921811	-8.65276	-3.71363
23	0.828096	0.921811	-8.65276	-3.71363
24	0.813929	0.921811	-8.65238	-3.71473
25	0.830831	0.921811	-8.65238	-3.71473
26	0.848045	0.921811	-8.65286	-3.71363
27	0.858572	0.921811	-8.65286	-3.71363
28	0.853368	0.921811	-8.65286	-3.71363
29	0.861227	0.921811	-8.65286	-3.71363
30	0.855864	0.921811	-8.65238	-3.71473
31	0.851469	0.921811	-8.65238	-3.71502
32	0.858286	0.921811	-8.65238	-3.71463
33	0.866796	0.921811	-8.65238	-3.71363
34	0.869285	0.921811	-8.65391	-3.71153
35	0.87443	0.921811	-8.65391	-3.71153
36	0.868587	0.921811	-8.65391	-3.71153
37	0.903035	0.921811	-8.65247	-3.71363
38	0.898683	0.921811	-8.65276	-3.71363
39	0.878667	0.921811	-8.65247	-3.71363
40	0.853164	0.921811	-8.65247	-3.71363

Table 4.2 Fitness statistics for 40 generations of the simple GA

简单GA函数

f6 22位x y

p_c= 0.65的

22位;

P_m=0.004

40代人口= 100

第16代最佳

焊锡

Fitness History Statistics

Generation	Mean Fitness	Max Fitness	Optimal x	
0	0.503697	0.867592	4.64513	-11.7565
1	0.508326	0.867592	4.64513	-11.7565
2	0.509802	0.867592	4.64513	-11.7565
3	0.53039	0.868933	4.64513	-11.7443
4	0.561234	0.921625	5.82707	7.37765
5	0.59649	0.921625	5.82707	7.37765
6	0.672782	0.950617	5.21481	-3.69651
7	0.752719	0.921625	5.82707	7.37765
8	0.78568	0.921633	5.82707	7.37803
9	0.821143	0.921633	5.82707	7.37803
10	0.785656	0.921811	5.82707	7.39653
11	0.763477	0.950584	5.215	-3.69651
12	0.780457	0.950549	5.215	-3.6968
13	0.801794	0.950617	5.21481	-3.69651
14	0.852029	0.950617	5.21481	-3.69651
15	0.854064	0.953766	5.21481	-3.66867
16	0.822871	0.962424	5.21481	-3.52895
17	0.841983	0.962424	5.21481	-3.52895
18	0.818125	0.962424	5.21481	-3.52895
19	0.819909	0.962424	5.21481	-3.52895
20	0.831519	0.962427	5.21472	-3.52895
21	0.862256	0.962627	5.21481	-3.51675
22	0.904606	0.962775	5.1904	-3.52895
23	0.873061	0.962775	5.1904	-3.52895
24	0.86957	0.962775	5.1904	-3.52895
25	0.883483	0.962775	5.1904	-3.52895
26	0.899208	0.962776	5.19059	-3.52895
27	0.864422	0.962776	5.19059	-3.52895
28	0.85123	0.962776	5.19059	-3.52895
29	0.878127	0.962776	5.1904	-3.53048
30	0.867648	0.962776	5.1904	-3.53048
31	0.899257	0.962776	5.1904	-3.53048
32	0.892281	0.962776	5.1904	-3.53048
33	0.905438	0.962776	5.1904	-3.53048
34	0.880897	0.962776	5.1904	-3.53048
35	0.834948	0.962776	5.1904	-3.53048
36	0.901784	0.962776	5.1904	-3.53029
37	0.914267	0.962776	5.19097	-3.52933
38	0.907366	0.962776	5.19097	-3.52933
39	0.887017	0.962776	5.19097	-3.52933
40	0.885118	0.962776	5.19097	-3.52933

Table 4.6 Fitness statistics for 40 generations choosing the best offspring and parents to form the next generation - first run

Pc=0. 75，

下午

=0. 01

Fitness History Statistics

Generation	Mean Fitness	Max Fitness	Optimal x	
0	0.498652	0.890776	3.13227	−5.76427
1	0.49634	0.887869	3.14448	−5.76427
2	0.515782	0.890055	3.13532	−5.76427
3	0.542739	0.93341	3.14448	−5.63891
4	0.568121	0.960416	3.14448	−5.37436
5	0.583212	0.955109	3.14753	−1.0844
6	0.646628	0.960549	3.14448	−5.37603
7	0.744575	0.95005	3.14448	−5.5682
8	0.78392	0.956143	3.14457	−1.08421
9	0.842531	0.989737	3.14753	−0.303149
10	0.842104	0.990168	3.14753	−0.107837
11	0.839353	0.98977	3.14753	−0.29552
12	0.891083	0.990168	3.14753	−0.107074
13	0.867687	0.990168	3.14753	−0.107074
14	0.891209	0.990169	3.14753	−0.106311
15	0.858183	0.990265	3.13227	−0.107837
16	0.878565	0.99027	3.13227	−0.123954
17	0.927549	0.99027	3.13227	−0.123954
18	0.889455	0.990284	3.13456	−0.151134
19	0.938775	0.990284	3.13685	−0.107837
20	0.939114	0.990284	3.1369	−0.107837
21	0.935044	0.990284	3.13456	−0.156665
22	0.943409	0.990284	3.13685	−0.107837
23	0.930018	0.990284	3.1369	−0.100732
24	0.971233	0.990284	3.13685	−0.0987768
25	0.936272	0.999751	0.0120401	−0.0101805
26	0.92873	0.999751	0.0120401	−0.0101805
27	0.935169	0.990284	3.13685	−0.10097
28	0.935309	0.998658	0.0120401	−0.0345945
29	0.919122	0.999845	0.0120401	−0.0030756
30	0.938004	0.999751	0.0120401	−0.0101805
31	0.937849	0.99974	0.0120401	−0.010705
32	0.909272	0.999844	0.0120401	−0.00326634
33	0.928413	0.999861	0.00593662	−0.0101805
34	0.938213	0.999849	0.0116587	−0.00388622
35	0.934315	0.999845	0.0120401	−0.0030756
36	0.943323	0.999844	0.00841618	−0.0092268
37	0.943836	0.999849	0.0116587	−0.00388622
38	0.929207	0.999849	0.0116587	−0.00388622
39	0.946492	0.999907	0.00879765	−0.00388622
40	0.954651	0.999914	0.00841618	−0.00388622

Table 4.7 Fitness statistics for 40 generations choosing the best offspring and parents to form the next generation - second run

第四章SGA的数学分析

- 相似性模板和架构
- 基本定理
- 两臂k匪问题
- 隐式并行性和构建块
- 最小的欺骗性问题

架构或相似性模板

- 模式或相似性模板表示在某些字符串位置具有相似性的子集字符串。
- 使用0和1的二进制字母，我们引入通配符’*’。
- 现在可以使用{0, 1，*}创建字符串。
- 架构* 101 *代表以下字符串：{01010, 11010, 01011, 11011}.
- 模式是表示字符串之间相似性的一种紧凑方式。

模式

- 如果字符串长度l为5，则存在3 ^ 5个不同的相似性模板，大于字符串的总数：2 ^ 5。
- 对于具有k个符号的字母，存在（k + 1）^ l个方案。
- 长度为1的任何特定二进制字符串都是2 ^ l模式的成员。字符串10111在给出2 ^ 5模式的5个位置中的每个位置可以取其实际值或“ *”。
- 具有n个成员且长度为l的总体可能具有2 ^ l到n.2 ^ l之间的模式，具体取决于总体多样性。

模式的生存与传播

- 由于复制，高度适合的架构可能会以越来越多的数量复制。
- 1点交叉可能会破坏模式1 **** 1，而不是*** 11 *。
- 因此，也非常适合的较短的模式在越来越多的一代之间传播。
- 这些简短的高度适合的架构被称为构建块。

模式顺序

- 如果H是取自3个字母的字母的模式{0,1，*}，则H的阶由o（H）表示
  这是字符串中固定位置的数量。
- 模式的顺序0 1 1 * 1 1 * 为5，架构0的顺序为* * * * * * * * *为1。
- 一个明显的观察结果是，低阶模式更可能在交叉和变异中幸存。

定义H的长度

- 定义模式H的长度是第一个和最后一个固定字符串位置之间的距离，表示为 (H)。

- 例如。 011 * 11 **的定义长度为 (H)=6-1=5
  定义长度1 *******为 (H)=1-1=0.
- 令m (H, t) 表示在时间t总体A (t) 中特定模式H的示例数。
- 选择，交叉和突变下模式的生存情况在详细注释中得出。

基本定理

- 考虑在一组字符串中，复制，交叉和突变对模式的单独影响和组合影响。
- 组*合后的结果称为遗传算法的模式定理或基本定理。*
- 具有更高总体平均适应性的短和低阶方案将在随后的世代中呈指数增长。

两臂强盗问题

- 假设一个赌徒有N个硬币来玩2臂老虎机。
- 这些部门的平均收益和方差如下：

1，　2，　1，　2

- 在给定赌徒的情况下，如何最大程度地提高总收益还不知道这些参数的值。
- 这是统计决策理论中的一个重要问题。

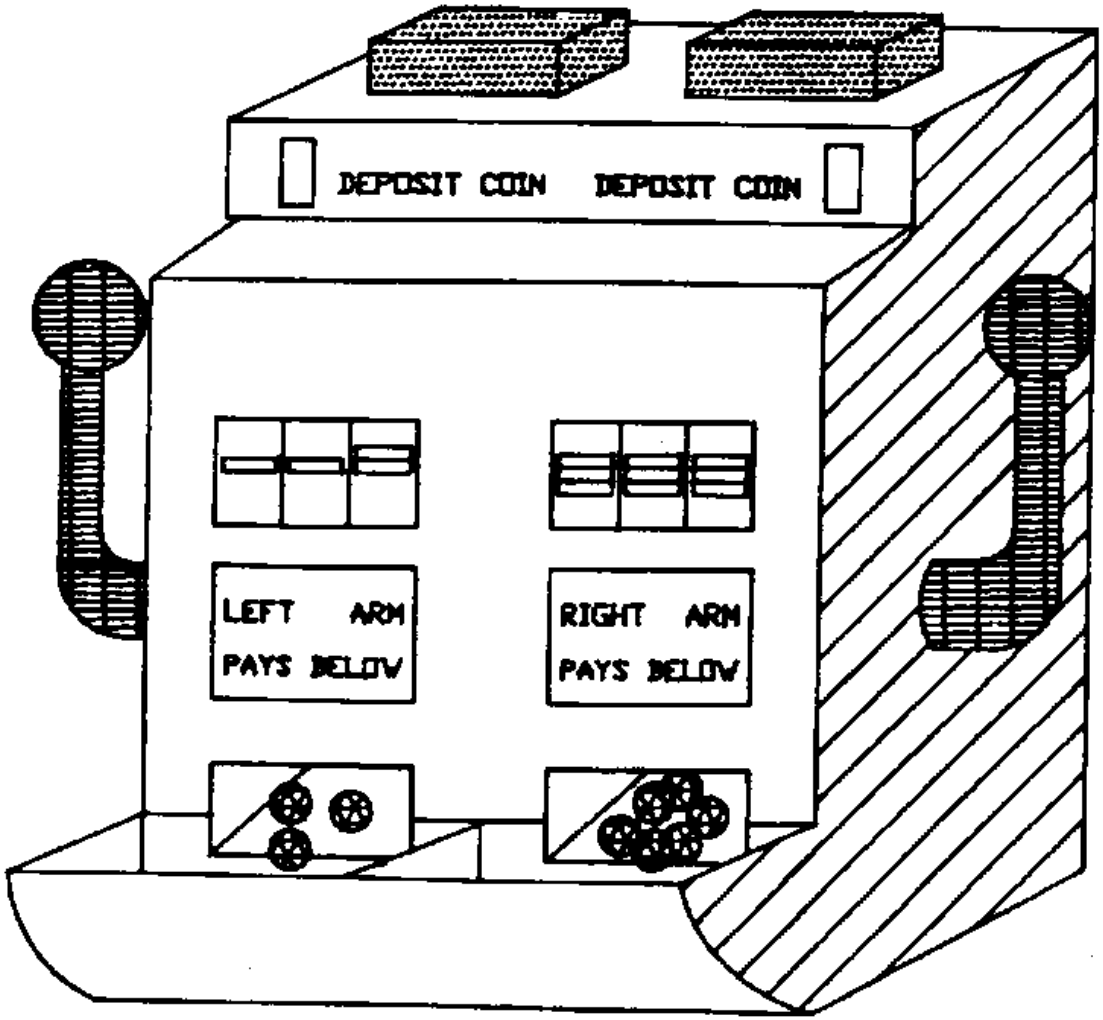


FIGURE 2.1 The two-armed bandit problem poses a dilemma: how do we search for the right answer (exploration) at the same time we use that information (exploitation)?

两臂强盗问题

- 让我们将n个硬币分配给每个手臂，以确定最佳手臂。
- 将剩余的N-2n个硬币分配给最佳手臂。
- 但是，有一个概率q（n）表明我们对最佳分支的识别是不正确的。
- 经过N次试验后与将所有N分配给最佳部门有关的预期损失：

$$L(N,n) = \quad n|\quad_1 - \quad_2\ | + q(n)(N\quad -2n)|\quad_1 - \quad\ _2|$$

两臂强盗问题

- 上面的表达式包含两个部分：（a）n次审判已明确分配给最差的一个部门。（b）将其余的N-2n次试验分配给最佳部门。 q（n）是我们对“最佳臂”的选择不正确的概率。
- 确定n以最小\化L（N，n）。差异：

$$\frac{dL\text{公司}}{\text{公称通径}} = -2q(n) + (N - 2n)\left[\frac{dq(n)}{\text{公称通径}}\right] \Big| - {}_2 \Big| = 0$$

$$\text{公称通径}_1$$

- 为了解决这个问题，我们需要知道q（n）。

两臂强盗问题

- 这有点复杂。因此，我们追求最终结果。
- 称为n *的最优n近似为：

$$n^* \approx \frac{8}{c^2} \ln\left(\frac{N^2}{\ln(N^2)}\right)$$ 哪里 $$c = \left|\frac{\square_1}{\left(\frac{}{1} - \frac{}{2}\right)}\right|^2$$

- 从上面的表达式中，我们获得：

$$n^* \sqrt{}$$

$$N \approx e^{2c} \frac{}{} 8 \; c^2 \ln (N (^2)$$

两臂强盗问题

- 给予最佳评价的次数：

$$N - n^* \approx e^{\frac{n^*}{2c}} \sqrt{8 \quad c^2 \ln \quad (\ N \quad (^2)}$$

- *解释是，最佳的试验组将成倍地增加试验次数。*
- k臂匪问题的结果与2臂匪问题的结果相似。
- 在通用航空中，存在几个多武装匪徒问题
  同时解决。

多臂强盗问题和遗传算法

- 考虑以下8种模式：

  * 0 0 * 0 * *

  * 0 0 * 1 * *

  * 0 1 * 0 * *

  * 0 1 * 1 * *

  * 1 0 * 0 * *

  * 1 0 * 1 * *

  * 1 1 * 0 * *

  * 1 1 * 1 * *

  可以将其与8臂强盗问题进行比较。

多臂强盗问题和遗传算法

- 这些2 ^ 3 = 8个方案相互竞争，以在再现阶段进行选择。
- 就像在k武装匪徒问题中一样，我们将指数递增的数字分配给最佳模式。
- 在Google Analytics（分析）中，一些问题并行进行。
- 例如，对于字符串长度7和模式顺序3，有 $\binom{7}{3}$ = 35个不同的8臂问题。
- 在一般 l，f，或模式顺序 j和字符串长度 l中，是 $\binom{l}{j}$ 不同的 2^j 武装匪徒问题。

隐式并行

- 每个字符串长度为l的个人都有2 ^ l个模式。
- 总体可能具有2 ^ l和n.2 ^ l之间的模式。
- 因此，在每一代中，最多可并行处理n.2 ^ l个模式 - 这称为隐式并行性。
- 尽管GA在每个世代都处理总体中的n个字符串，但估计在每个世代中都处理了大约n ^ 3个模式。
- GA的计算能力来自这种隐式并行性。

构件假设

- 具有较长定义长度的架构很可能会被交叉破坏。
- 短，低阶和高度适合的架构以成倍的速率采样。
- 这些架构被称为构建块。
- 遗传算法通过重新组合构建基块以形成越来越好的字符串来执行计算-这称为构建基块假设。
- 存在一些可能违反此假设的问题。

GA欺骗性问题

- 如果好的点/解决方案被坏的点/解决方案包围，那么GA很难实现最优。
- 这些功能和编码被称为GA欺骗性的。
- 特定问题可能具有GA欺骗性编码和非欺骗性编码。
- 接下来，我们将尝试构造一个违反构件假设的问题。简而言之，低阶高度适合的模式导致 错误的较长的高阶模式。

最小欺骗性问题

- 没有订单-1个问题会欺骗GA
- 可能的最小GA欺骗性问题是一个阶数-2问题-称为最小欺骗性问题（MDP）。
- 考虑一组4个订单 - 2个模式，如下所示：

$***0******0*$ f_{00}是架构平均适应度值。

$***0******1*$ f_{01}是架构平均适应度值。

$***1******0*$ f_{10}是架构的平均适应度值。

$***1******1*$ f_{11}是架构平均适应度值。

- 定义长度为 (H).

最小欺骗性问题

- 现在，我们的目标是试图欺骗GA给出错误的答案。
- 假设f11是与全局最优解关联的模式，则订单的适应度f0 * –一个模式0 *大于该订单的适应度f1 * –一个模式1 *，即

$$\frac{f_{00} + f_{01}}{2} > \frac{f_{10} + f_{11}}{2}$$

最小欺骗性问题

- 给定f11对应于全局最优解，我们可以得出以下结论：

$$f_{10} < f_{01} \quad \text{和} \quad f_{10} < f_{00}$$

- f01和f00之间可能存在两种关系：

类型 I： $f_{01} > f_{00}$

类型 II： $f_{01} \leq f_{00}$

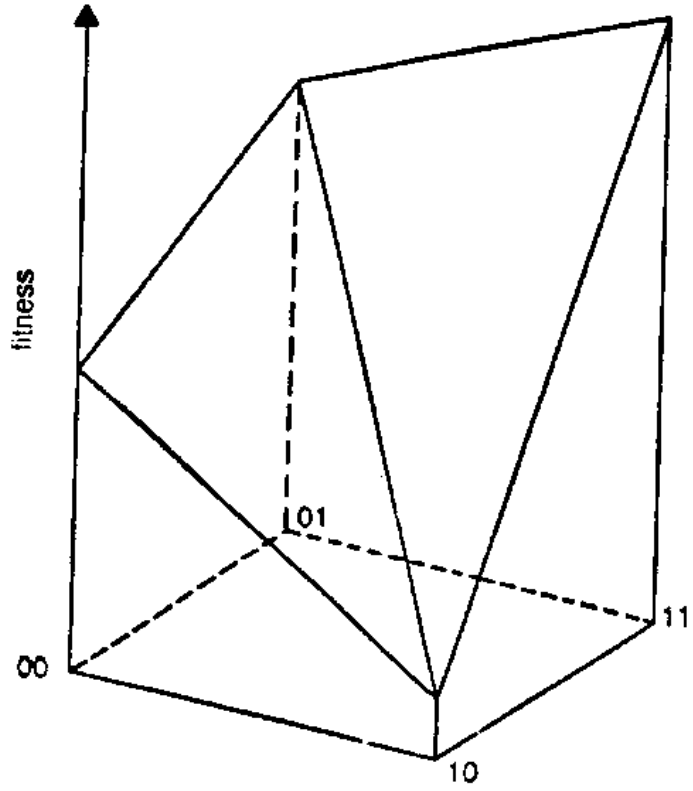


FIGURE 2.8 Sketch of Type I. minimal deceptive problem (MDP) $f_{01} > f_{00}$.

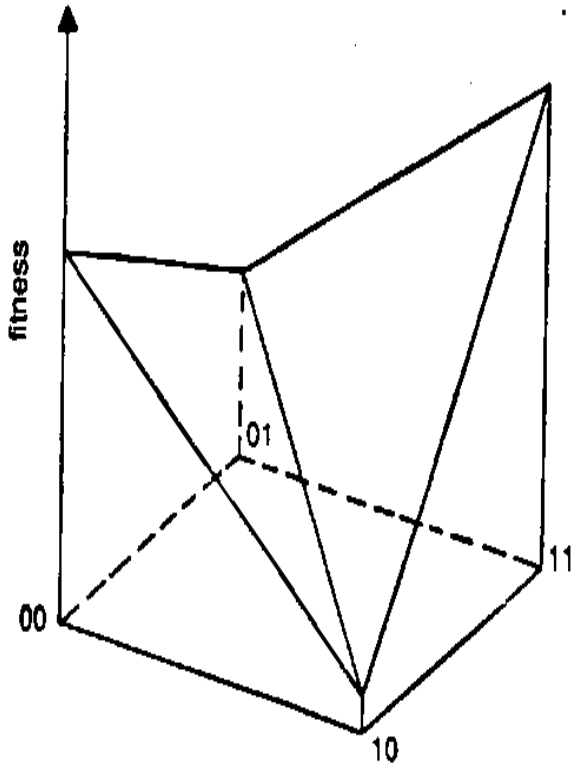


FIGURE 2.9 Sketch of Type II, minimal deceptive problem (MDP) $f_{00} > f_{01}$.

最小欺骗性问题

- 接下来，我们将以0.0的突变概率为每个模式的预期部分构造差分方程关系。
- 架构定理（在扩展的讲义中）仅假设所有交叉都可能破坏架构而给出下界。
- 在这里，我们可以通过考虑所有交叉对来计算出更好的估算值，如
  表2向前滑动。

表：1点交叉的结果

- “ S”表示两种模式在1点交叉条件下均有效，而与交叉点无关。
- 仅当（a）两种模式相互补充并且（b）交叉位点在固定位置之间时，两个模式才能被破坏。
- 如上沿对角线的表格所示，创建的新模式适用于上述情况（b）。

对角线（从左下到右上）显示了xover站点在固定位置之间时1点交叉的结果。

S－1点交叉获得的相同架构，与交叉站点无关。

	00	01	10	11
00	S	S	S	01 10
01	S	S	00 11	S
10	S	00 11	S	S
11	01 10	S	S	S

最小欺骗性问题

- 模式在时间t的比例表示为：

  P_{00}^{t}，P_{01}^{t}，P_{10}^{t}，P_{11}^{t}

- 交叉站点掉落的概率

  两个固定位之间的位置是：

$$p_c \quad l-1$$

- 如果我们假设交叉概率p_c为1，

  然后：

$$p_c \frac{(H)}{l-1} = 0.6$$

在时间t + 1推算人口比例

- 考虑模式“ 00”的比例。
- 除了（a）与固定位置之间的交叉点与模式’11’配对时，此’00’模式被破坏，以及（b）当模式’10’和’01’与交叉点匹配时，该比例始终存在。在固定位置之间，将创建此“ 00”模式。给出：

$$\Pi^{t+1}_{00} = f_{00} P^t_{00} - 0.6 f_{0*} f_{*1} + 0.6 f_{01} f_{10} P^t P^t$$

$$f \qquad f \quad f \qquad f^2$$

在时间t + 1推算人口比例

- 另外，请考虑所有成对的架构，并确定X重叠后架构“ 00”的比例：

00 00 $f^2 \frac{f_{00}}{\bar{f}} P^t_{00} \quad \frac{f_{00}}{\bar{f}} P^t_{00}$

00 01 $\frac{f_{00}}{\bar{f}} P^t_{00} \quad \frac{f_{01}}{\bar{f}} P^t_{01}$

00 10 $\frac{f_{00}}{\bar{f}} P^t_{00} \quad \frac{f_{10}}{\bar{f}} P^t_{10}$

01 10 $0.6 \frac{f_{01}}{\bar{f}} P^t_{01} \quad \frac{f_{10}}{\bar{f}} P^t_{10}$

00 11 $\frac{f_{00}}{\bar{f}} P^t_{00} \quad \frac{f_{11}}{\bar{f}} P^t_{11} - 0.6 \frac{f_{00}}{\bar{f}} P^t_{00} \quad \frac{f_{11}}{\bar{f}_{150}} P^t_{11}$

继续 .....

在时间t + 1推算人口比例

- 在时间t + 1时考虑计算模式“ 00”的比例的其他对为： (01 00), (10 00), (10 01), (11 00)
- 还请记住，对于n种群，交叉运算仅执行n / 2次。
- 因此，当计算比例时，因子2将被取消。

最小欺骗性问题

- t + 1代中的4个模式的比例为：

$$P_{00}^{t+1} = \frac{f_{00}}{\bar{f}} P_{00}^{t}(1 - 0.6\frac{f_{11}}{\bar{f}} P_{11}^{t}) + 0.6\frac{f_{01} f_{10}}{\bar{f}^2} P_{01}^{t} P_{10}^{t}$$

$$P_{01}^{t+1} = \frac{f_{01}}{\bar{f}} P_{01}^{t}(1 - 0.6\frac{f_{10}}{\bar{f}} P_{10}^{t}) + 0.6\frac{f_{00} f_{11}}{\bar{f}^2} P_{00}^{t} P_{11}^{t}$$

$$P_{10}^{t+1} = \frac{f_{10}}{\bar{f}} P_{10}^{t}(1 - 0.6\frac{f_{01}}{\bar{f}} P_{01}^{t}) + 0.6\frac{f_{00} f_{11}}{\bar{f}^2} P_{00}^{t} P_{11}^{t}$$

$$P_{11}^{t+1} = \frac{f_{11}}{\bar{f}} P_{11}^{t}(1 - 0.6\frac{f_{00}}{\bar{f}} P_{00}^{t}) + 0.6\frac{f_{01} f_{10}}{\bar{f}^2} P_{01}^{t} P_{10}^{t}$$

最小欺骗性问题

- $\bar{f}$ 是t代中的平均人口适应度由：

$$\bar{f} = f_{00}P_{00}^{t} + f_{01}P_{01}^{t} + f_{10}P_{10}^{t} + f_{11}P_{11}^{t}$$

- 模拟上一张幻灯片中的表达式，以了解历代中模式编号的变化。
- 关于f00规范架构适应性。例如，当我们指定f11 = 1.1时，意味着f11 = 1.1f00。

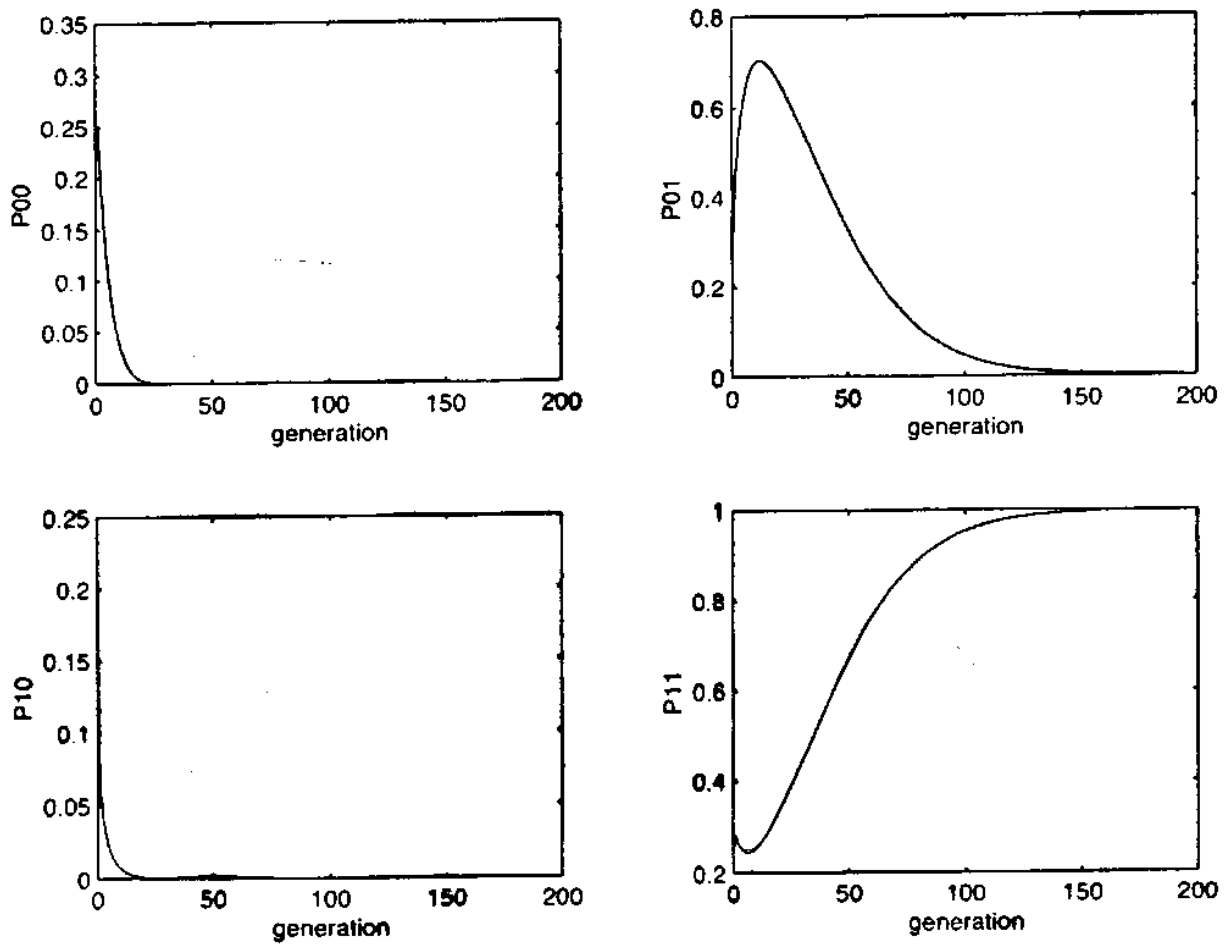


Figure 3.1 Type 1: $f_{01} = 1.05$, $f_{10} = 0$, $f_1 = 1.1$ and equal initial proportions.

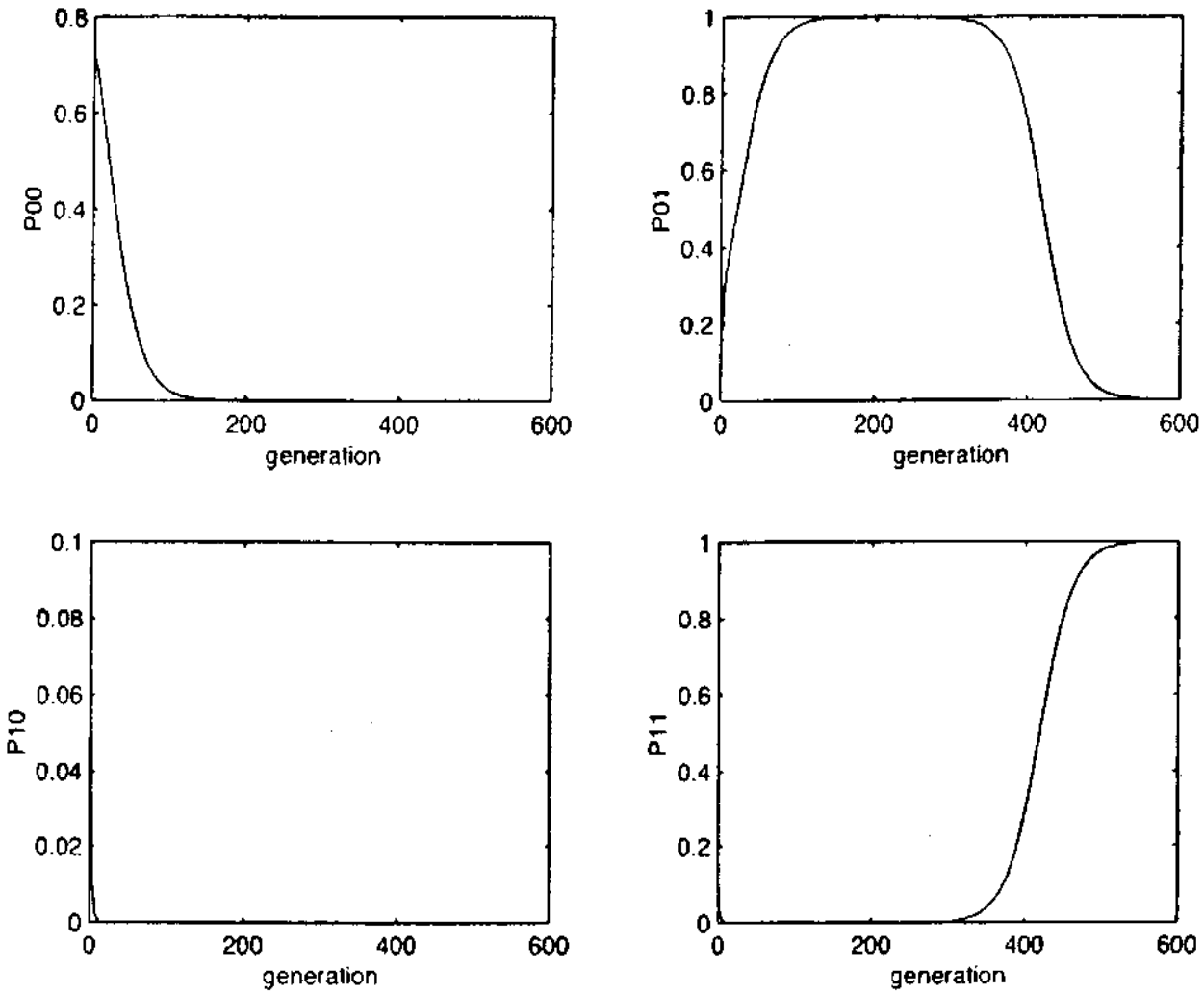


Figure 3.2 Type 1: $f_{01} = 1.05$, $f_{10} = 0$, $f_1 = 1.1$ and unequal initial proportions.

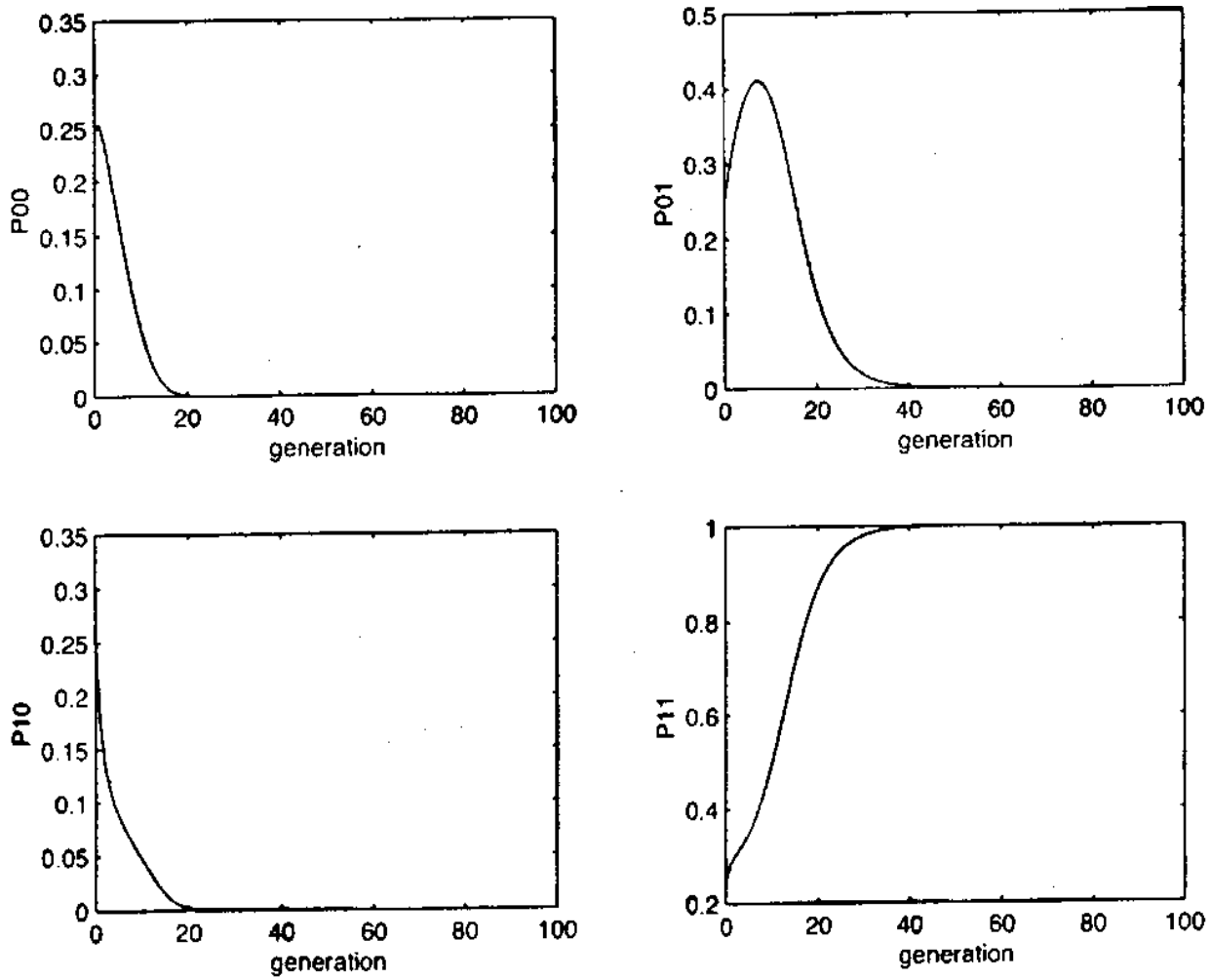


Figure 3.3 Type 2: $f_{01} = 0.9$, $f_{10} = 0.5$, $f_1 = 1.1$ and equal initial proportions.

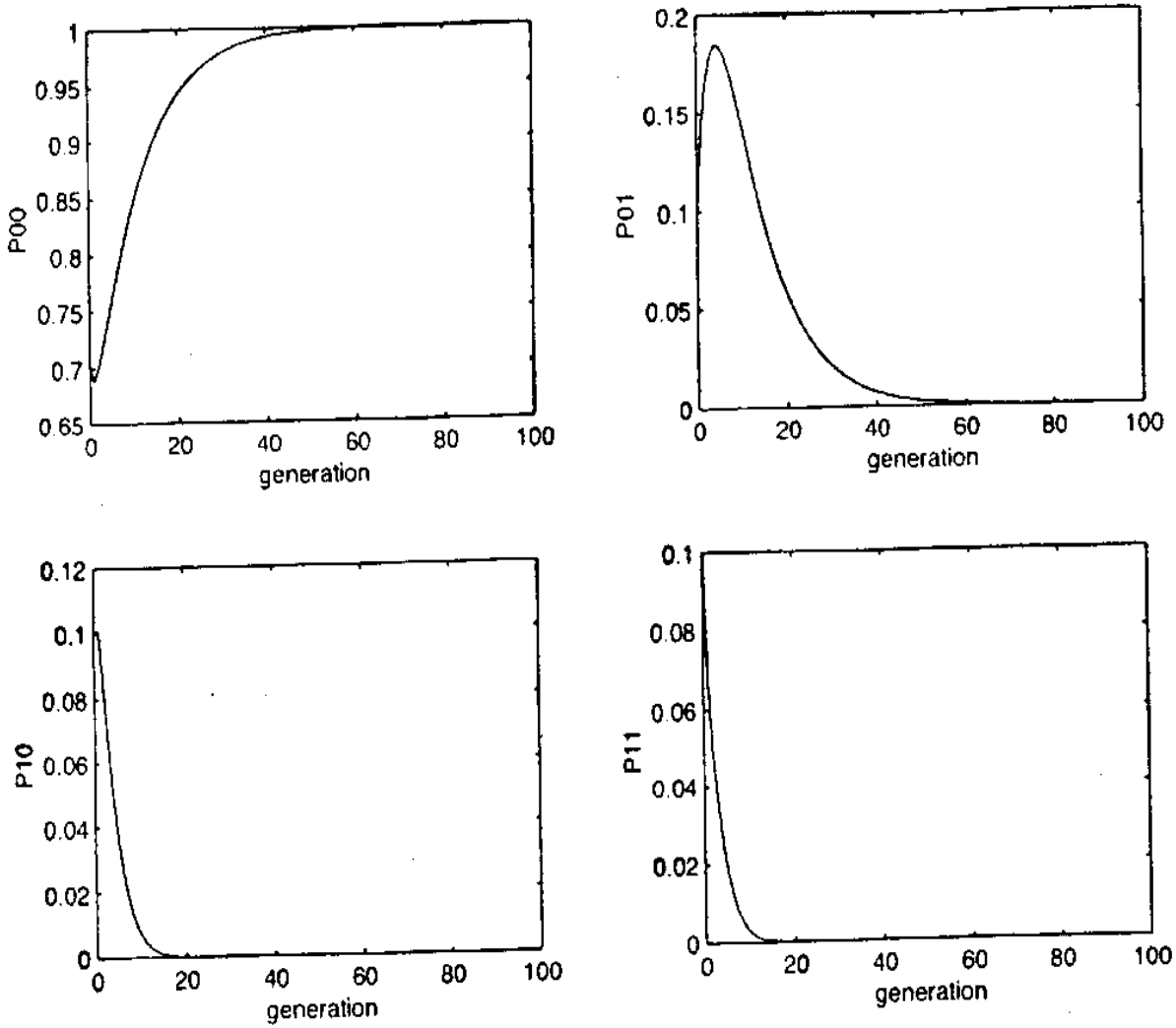


Figure 3.4 Type 2: $f_{01} = 0.9$, $f_{10} = 0.5$, $f_1 = 1.1$ and unequal initial proportions.