

# 基于接口参数的黑箱测试用例自动生成算法

聂长海 徐宝文

(东南大学计算机科学与工程系 南京 210096)

(江苏省软件质量研究所 南京 210096)

**摘要** 测试用例的选择与生成技术是软件测试尤其是黑箱测试的一个重要研究领域,测试用例的质量将直接决定软件测试的科学性和有效性。该文在一般的测试用例选择方法的基础上,提出了一种基于对接口参数进行组合覆盖的黑箱测试用例自动生成算法模型,据此可以生成一个对所有接口参数进行两两组合覆盖的测试用例表。并证明了该方法产生的测试用例具有数量少、能实现对接口参数最大限度组合覆盖的特点,从而可以在提高软件测试质量的同时,降低成本,提高效率。最后介绍了该算法在作者研究开发的测试数据生成工具中的实际效果。

**关键词** 软件测试; 黑箱测试; 测试用例; 算法; 软件工程

中图法分类号 TP311

## An Algorithm for Automatically Generating Black-Box Test Cases Based Interface Parameters

NIE Chang-Hai XU Bao-Wen

(Department of Computer Science and Engineering, Southeast University, Nanjing 210096)

(Jiangsu Institute of Software Quality, Nanjing 210096)

**Abstract** This paper presents a network graph model for test case automatic generation, which is based on the combinatorial coverage of all the interface parameters for black box testing. In this model, a path from left to right represents a test case. The algorithm chooses a test case every time to make the corresponding path in the network graph cover all the uncovered vertices to the greatest degree by some rules. It can generate a test case table to cover all the pair-wise combination of all the interface parameters. Contrasted with the existing work, authors prove that the generated test cases are able to cover all the combinations of parameters to the greatest degree with the smallest scale of test suite, thus it can improve the quality of software testing by decreasing software testing cost and improving its efficiency.

**Keywords** software testing; black-box testing; test case; algorithm; software engineering

## 1 引言

测试用例的选择与自动生成技术是软件测试的

一个重要研究领域。测试用例的选择无论是对黑箱测试还是对白箱测试都起着关键的作用,决定着软件测试的质量和效果。所谓测试用例选择就是指从所有的可用测试用例中选出少量典型的测试用例,

收稿日期: 2002-01-08; 修改稿收到日期: 2003-07-07. 本课题得到国家自然科学基金(60373066)、国家“九七三”重点基础研究发展规划项目基金(2002CB312000)、江苏省自然科学基金(BK2001004)、教育部高等学校骨干教师基金、江苏省科技攻关项目基金(BE2001025)、教育部跨世纪优秀人才培养计划基金、高等学校博士学科点专项科研基金、江苏省三三三人才基金、武汉大学软件工程国家重点实验室基金资助。聂长海,男,1971年生,讲师,博士研究生,主要从事模糊信息处理、软件工程和软件测试技术、神经网络等方面的教学与科研工作。E-mail: changhainie@seu.edu.cn。徐宝文,男,1961年生,教授,博士生导师,主要从事程序设计语言、软件工程、并行与网络软件、知识与信息获取技术等方向的教学与科研工作。

以达到对测试域的最大限度覆盖。多年来,许多研究者对之进行了广泛而深入的研究,并取得了许多研究成果<sup>[1~8]</sup>。常用的基于接口参数的黑箱测试用例选择方法是对系统每个接口参数采用边际值分析法和等价类划分法等选取一组典型的值,然后在这些取值组合中随机选取一组测试用例,或者使用一些启发式方法从中进行筛选<sup>[9,10]</sup>。但这些方法的缺点是带有主观倾向性,不具有普遍性。

本文给出了一种按照对系统各个接口参数两两组合全面覆盖的原则来选择测试用例的方法。这种方法充分考虑了待测试软件的所有外部接口参数的各种取值和各种取值组合可能对系统产生的影响。本文充分讨论了按照这种覆盖原则来选择测试用例方法的优点和特点,在此基础上详细给出了生成这种测试用例表的算法模型并证明了有关性质,最后列举了在我们研究开发的测试数据生成工具中该自动生成算法的一些实际效果。

## 2 黑箱测试用例选择方法

设一个系统对外接口由  $n$  个参数  $c_1, c_2, \dots, c_n$  组成。每个参数都有其指定的取值范围(约束): $c_1 \in D_1, c_2 \in D_2, \dots, c_n \in D_n$ , 其中  $D_i$  为参数  $c_i$  的合理取值范围。现在从这组接口出发,决定选取测试用例来进行测试。遍历是不可能也是不必要的,而随机选取又带有盲目性。根据前人的研究,一般先根据边际分析法选择  $c_i$  可以取到的边际值,因为边际值往往具有较强的暴露错误的能力,再根据等价划分法补充一些取值点,使得  $c_i$  的取值更具有代表性。经过这样处理之后,参数  $c_1, c_2, \dots, c_n$  都取一组离散值  $T_1, T_2, \dots, T_n$ 。其中  $T_i$  表示参数  $c_i$  可取的有限离散点集,并假设符号  $t_i$  表示第  $i$  个参数  $c_i$  可取值的个数,即  $T_i$  中元素的个数: $t_i = |T_i|$ 。

经过以上处理,虽然可以大大降低测试用例选择的难度,但仍需要  $m = t_1 \times t_2 \times \dots \times t_n$  个测试用例,数量还是相当大,需要进一步精简。由于此时导致系统发生问题有这样一些因素:某一个参数的某一种取值,某两个参数的某种取值组合, …, 某  $n-1$  个参数的某种取值组合,直到所有这  $n$  个参数的某种取值组合。因此我们考虑挑选的测试用例能够以最少的数量最大限度地对上面各种可能性因素进行覆盖。从上面的讨论可以知道,如果对每个参数的各个取值进行覆盖,即在测试用例中,保证每个参数的

各个取值都至少出现一次,需要  $m = \max_{1 \leq i \leq n} t_i$  个测试用例;如果考虑对任意两个参数的所有取值组合进行覆盖,保证它们每一个都在测试用例中出现,则至少需要  $m = \max_{1 \leq i \neq j \leq n} t_i \times t_j$  个测试用例, …, 依此类推,随着参数组合覆盖要求的递增,测试用例的数量也呈指数上升。注意到如果某组测试用例能实现对  $m(1 \leq m \leq n)$  个参数的组合覆盖,则也能实现对  $m-1, m-2, \dots, 2, 1$  个参数的所有取值组合进行覆盖,所以要综合考虑测试用例的规模、覆盖能力以及测试费用。我们现在先考虑对任意两个参数组合覆盖的测试用例表的生成,下面先给出这种测试用例表的形式定义。

**定义 1.** 设  $\mathbf{A}$  是一个  $n \times m$  矩阵,  $\mathbf{A} = (a_{ij})_{n \times m}$ , 其第  $j$  列表示第  $j$  个参数,它的元素取自有穷符号集合  $T_j(j = 1, 2, \dots, m)$ , 即  $a_{ij} \in T_j, i = 1, 2, \dots, n$ , 且  $t_j = |T_j|$  表示集合  $T_j$  中符号元素个数或者第  $j$  个参数的取值个数为  $t_j$ , 如果  $\mathbf{A}$  的任两列第  $i$  列和第  $j$  列都满足:  $T_i$  的符号和  $T_j$  符号的全部组合都在第  $i$  列和第  $j$  列形成的二元有序对中等概率出现,那么称  $\mathbf{A}$  是正交表;如果只要求出现而不要求等概率出现,那么称  $\mathbf{A}$  是一个测试用例表,其中  $n$  是测试用例的个数,如果它是能保证上述条件成立的最小正整数,那么  $\mathbf{A}$  可以称为最小两两组合覆盖表。 $\mathbf{A}$  的每一行就是一测试用例。

**引理 1.** 如果  $t$  是一个素数或者素数方幂,则存在一个有  $t^2$  行、 $t+1$  列且每列有  $t$  个取值的正交表,记为  $L_t(t^{t+1})$ <sup>[12]</sup>。

据引理 1 可知,当  $t$  是一素数或者素数方幂时,正交表  $L_t(t^{t+1})$  同时也是最小组合覆盖表。

测试用例表是最小两两组合覆盖表的近似,因为最小组合覆盖表的生成方法是一个很困难的数学问题,目前关于这种表的研究还未见到,人们的研究主要集中在测试用例表的生成方法上<sup>[13,14]</sup>。测试用例表的行数即测试用例的数量有一个上界和下界,具体表现为下面的有界性定理。

**定理 1(有界性定理).** 设系统有  $n$  个参数,每个参数的取值个数为  $t_i, i = 1, 2, \dots, n$ , 则存在为该系统生成的测试用例表,其行数  $row$  的范围是

$$\max_{1 \leq i \neq j \leq n} (t_i \times t_j) \leq row \leq (\text{pri}(\max_{1 \leq i \leq n} t_i, n-1)))^2,$$

其中,  $\text{pri}(x)$  表示不小于  $x$  的最小素数或素数方幂。

**证明.** 由于第  $i$  个参数的  $t_i$  个取值与第  $j$  个参数的  $t_j$  个取值完全组合有  $t_i \times t_j$  个,所以测试用例表

要满足对任意两个参数的所有取值组合的完全覆盖,至少要  $\max_{1 \leq i, j \leq n, i \neq j} (t_i \times t_j)$  行, 即  $row \geq \max_{1 \leq i \neq j \leq n} (t_i \times t_j)$ .

设  $t = (\text{pri}(\max(\max_{1 \leq i \leq n} t_i, n - 1)))^2$ , 则  $\sqrt{t}$  为素数

或素数方幂. 把  $n$  个参数中每个参数扩展为  $\sqrt{t}$  个取值, 即  $t_i = \sqrt{t}, i = 1, 2, \dots, n$  时, 根据正交表的构造理论可以生成正交表  $L_t(\sqrt{t}^n)$ , 对这个正交表扩展的部分用可以取到的值进行随机取代, 即可得到实际系统的一张最坏情况下的测试用例表. 所以  $row \leq (\text{pri}(\max(\max_{1 \leq i \leq n} t_i, n - 1)))^2$ . 证毕.

这个结论保证了测试用例表的存在性和测试用例规模的范围, 如果用  $N$  表示每个取值个数与参数个数的最大值, 则测试数据的数据复杂程度是  $O(N^2)$ . 例如, 对于一个具有 10 个接口参数、每个参数有 10 个不同取值的系统, 如果进行全面组合覆盖, 则需要  $10^{10}$  个测试用例, 而用测试用例表则最多只需要  $11^2 = 121$  个测试用例, 这不仅大大减少了测试的次数, 降低了软件测试的成本, 而且这样的测试用例表还具有以下优良特性.

**定义 2.** 如果某个系统有  $n$  个接口参数, 每个参数  $c_i$  的取值个数为  $t_i, i = 1, 2, \dots, n$ , 其中某  $m$  个参数  $c_{i_j} (1 \leq j \leq m)$  的各种取值组合数为  $C_m = t_{i_1} \times t_{i_2} \times \dots \times t_{i_m}$ , 若  $\mathbf{A}$  为该系统的一个测试用例表, 被  $\mathbf{A}$  覆盖的这  $m$  个参数的组合数为  $C_{mA}$ , 称测试用例表  $\mathbf{A}$  覆盖的这  $m$  个参数的组合数  $C_{mA}$  与该  $m$  个参数的所有组合数  $C_m$  的比值为该系统对应测试用例表  $\mathbf{A}$  对该  $m (1 \leq m \leq n)$  个参数的组合覆盖率, 记为  $p = \frac{C_{mA}}{C_m}$ .

**定理 2.** 设系统有  $n$  个接口参数, 每个参数  $c_i$  的取值个数为  $t_i, i = 1, 2, \dots, n$ , 且  $t_1 \geq t_2 \geq \dots \geq t_n$ , 则该系统对应测试用例表  $\mathbf{A}$  对任意  $m (1 \leq m \leq n)$  个参数的组合覆盖率为

$$p \geq \frac{t_1 \times t_2}{t_1 \times t_2 \times \dots \times t_m}.$$

**证明.** 根据定理 1, 该系统存在一个行数  $row \geq t_1 \times t_2$  的测试用例表  $\mathbf{A}$ , 该测试用例表以最少的行数实现了对系统的任意两个参数  $c_i$  和  $c_j$  的全部组合的覆盖, 即任意两个参数的组合覆盖率  $p \geq 1$ , 且  $\mathbf{A}$  的任意两行不会相同. 对某个参数  $c_i (2 \leq i \leq n)$  各种取值的覆盖率, 可以从参数  $c_1$  与参数  $c_i$  的  $t_1 \times t_i$  个组合都被覆盖, 知道参数  $c_i$  的每个取值的被覆盖率  $p \geq t_1 \geq t_2$ ; 参数  $c_1$  的各个取值的覆盖率由参数  $c_1$  和参数  $c_2$  的  $t_1 \times t_2$  个组合都被覆盖, 可知其覆盖

率  $p \geq t_2$ , 因此  $m=1$  时结论是正确的.  $m=2$  时结论显然成立. 以下我们证明  $3 \leq m \leq n$  时, 结论也是正确的.

对任意  $m$  个参数  $c_{i_1}, c_{i_2}, \dots, c_{i_m}$ , 不妨设它们所对应的取值个数有  $t_{i_1} \geq t_{i_2} \geq \dots \geq t_{i_m}$ , 根据条件,  $\mathbf{A}$  中至少有  $t_{i_1} \times t_{i_2}$  个不同的关于这  $m$  个参数的不同组合, 而这  $m$  个参数的所有组合数是  $t_{i_1} \times t_{i_2} \times \dots \times t_{i_m}$ , 所以测试用例表  $\mathbf{A}$  对这  $m$  个参数的组合覆盖率  $p$  是

$$\begin{aligned} p &\geq \frac{t_{i_1} \times t_{i_2}}{t_{i_1} \times t_{i_2} \times \dots \times t_{i_m}} = \frac{1}{t_{i_3} \times t_{i_4} \times \dots \times t_{i_m}} \\ &\geq \frac{1}{t_3 \times t_4 \times \dots \times t_m} = \frac{t_1 \times t_2}{t_1 \times t_2 \times \dots \times t_m}, \end{aligned}$$

故得证. 证毕.

从上面的讨论可知, 测试用例表是以最少的测试用例实现对各个接口参数的各种组合进行最大限度覆盖的测试用例集. 与现有的一些方法相比, 用它所进行的测试用例的选择不仅更科学, 而且所产生的测试用例相对比较少.

### 3 测试用例表生成算法

基于上面的讨论, 下面我们给出测试用例表的生成算法. 该算法由主体用例生成、覆盖率检查与追加用例生成 3 个步骤组成, 其中, 第 1 步主体用例生成用于初步生成一组测试用例, 这一组测试用例是否满足要求将由第 2 步进行检测, 如果检测的结果满足要求, 那么输出结果, 否则将由第 3 步追加用例生成补充一些测试用例后一起将符合要求的结果输出.

设某个系统其接口参数有  $n$  个, 第 1 个参数有  $t_i$  个取值 ( $i = 1, 2, \dots, n$ , 对参数按取值个数的大小顺序排列, 即  $t_1 \geq t_2 \geq \dots \geq t_n$ ). 下面按这 3 个步骤顺序来说明如何产生测试用例表, 使得用最少的测试用例实现对任意两个接口参数两两组合的全面覆盖.

#### 3.1 主体用例生成

主体用例生成部分基于如图 1 所示的模型(在该图中假定  $n=4, t_1=t_2=t_3=t_4=3$ ), 其基本思想是, 构造一个如图 1 的网络, 网络的第 1 层 3 个节点分别表示第 1 个参数的 3 个取值, 余下以此类推, 从第 1 层某个节点出发沿着节点间的路径向前推进, 直到这个网络的终点(最右边的节点层), 形成一条路径, 路径上的节点号序列就代表一个测试用例. 例

如 1—2—3—1,就表示第 1 个参数取第 1 个值,第 2 个参数取第 2 个值,第 3 个参数取第 3 个值,第 4 个参数取第 1 个值的一个测试用例. 假定每个节点具有一定的计算和处理能力,我们对网络赋予以下计算功能:

(1) 计算顺序:按网络层次从左到右,每一层是从上到下的顺序逐个节点执行.

(2) 第 1 层每个节点沿着路径按下一层节点号递增的次序给每个节点发送自己的节点号.

(3) 第 2 层每个节点每接受到一个节点号,就把自己装配在这个节点号的后面,形成一个节点号串,并把它按照接收的顺序保存.

(4) 第 1 层每个节点发送完毕,从第 2 层的每个节点开始按照下面的算法向下一局每个节点发送该节点内形成的号码串. 第  $i$  层 ( $i \geq 2$ ) 的第  $j$  号节点 ( $1 \leq j \leq t_i$ ) 把该节点内第 1 个组装好的号码串,与第  $i+1$  层节点按照  $j, j+1, \dots, t_{i+1}, 1, 2, \dots, j-1$  的顺序与每个节点内已经收到的来自其它第  $i$  层节点发来的每个符号串进行比较,前面  $i$  个位置的相同位置上每有一个相同符号,则相应计数器加 1(计数器每次初始化为 0). 最后选择第 1 个遇到的计数器的值最小的节点  $k$  将这个符号串发过去. 然后再从第  $j$  号节点取第 2 个组装好的号码串,按照  $k+1, k+2, \dots, t_{i+1}, 1, 2, \dots, k$  顺序从第  $k+1$  号节点开始,与该节点内每个符号串进行比较,前面  $i$  个位置的相同位置上每有一个相同符号,则相应计数器加 1(计数器每次初始化为 0). 最后选择第 1 个遇到的计数器的值最小的节点  $k$  将这个符号串发过去,依此类推,直到第  $j$  号节点内的符号串全部发完.

(5) 最后一层节点为输出层,每个节点内形成一组节点号串,这些节点号串全体就是产生的测试用例主体.

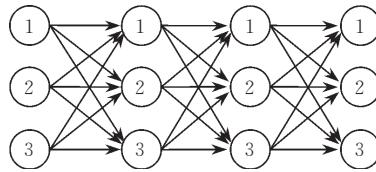


图 1 算法的网络结构模型

### 3.2 覆盖性检查

由 3.1 节产生的输出不一定是满足要求的测试用例表,所以我们还要对该输出进行检测,检查任意两个参数之间的所有组合是否被全部覆盖,如果覆盖,则输出所有符号串即为生成的测试用例表,计算

终止;否则,则输出所有遗漏的组合,形成一张遗漏项表(见表 1),等待下一步处理. 例如表 1 中第 3 列有遗漏项 13 与 22,就说明是参数 1 与参数 4 的这两个组合没有被覆盖.

表 1 组合遗漏项表

遗漏项 编号	参数组合						
	参数 1 参数 2	参数 1 参数 3	参数 1 参数 4	参数 2 参数 3	参数 2 参数 4	参数 3 参数 4	
	13	11	13	22	33	12	
1							
2	22	21	22	11	31	33	
3	12						

### 3.3 追加用例生成

这一步的目的是根据检测出来的遗漏项,设计最少数量的用例追加到前面生成的用例中去,形成一个完整的测试用例表. 算法如下:

1. 从遗漏项表中找出一个遗漏项,产生一个用例,使其在相应的参数位置填上相应参数的取值号码,其它参数位置上标记为空缺,从遗漏项表中去掉这个遗漏项;
2. 分别以上面产生的这个用例中确定参数值为中心,查找遗漏表,找出相关遗漏项,并把其参数号码填在相应空缺位置. 同时把表中相应的遗漏项删掉;
3. 以新填入的参数值为中心,执行步骤 2 相同的搜索,直到这个用例的各个参数位置没有空缺标记或者从表中找不到相关遗漏项为止;
4. 输出该条测试用例;
5. 如果表中还有遗漏项,转步骤 1.

## 4 测试用例表生成算法的性质

下面我们来分析上面算法的特性. 该算法具有以下性质.

**性质 1.** 按照算法步骤(3.1 节)得到的所有测试用例能保证任意两个相邻参数的各种组合都能全部被覆盖. 而不相邻的任意两个参数的各种组合不一定能全部被覆盖. 如果也能被全部覆盖,那么,这个算法产生的输出就是测试用例表,此时测试用例的数量是  $t_1 \times t_2$  个.

**性质 2.** 从算法步骤(3.1 节)可以看到,要做到对任意两个参数的所有组合的全面覆盖,至少需要  $t_1 \times t_2$  个测试用例.

**性质 3.** 对于有  $m$  个参数,每个参数有  $t$  个取值的系统,最好的设计是对任意两个参数的各种组合只覆盖一次的测试用例表,反映到该模型上是  $t^2$  条以第 1 层节点为起点,最后一层节点为终点的路径,这些路径经过模型上所有相邻节点间的连线,而且只经过一次.

**定理 3.** 当  $t$  为素数且  $t_1 = t_2 = \dots = t_n = t, n \leq$

$t+1$  时, 该算法产生的测试用例表能够对任意两个参数的所有组合进行一次等概率覆盖.

证明. 为证明这个结论, 我们先根据正交表构造理论, 证明在该定理条件下存在一张满足要求的正交表, 然后证明依据测试用例表生成算法正好能生成这样一张正交表作为测试用例表.

不失一般性, 考虑  $n=t+1$ , 即  $n$  取最大值的情况. 当  $t$  为素数时, 则存在正交拉丁方完全组

1	1	1	1	...	1	1
1	2	2	2	...	2	2
1	3	3	3	...	3	3
:	:	:	:	:	:	:
1	$t-1$	$t-1$	$t-1$	...	$t-1$	$t-1$
1	$t$	$t$	$t$	...	$t$	$t$
2	1	2	3	...	$i$	$i+1$
2	2	3	4	...	$i+1$	$i+2$
2	3	4	5	...	$i+2$	$i+3$
:	:	:	:	:	:	:
2	$t-1$	$t$	1	...	$i-2$	$i-1$
2	$t$	1	2	...	$i-1$	$i$
:	:	:	:	:	:	:
$t$	1	$t$	$t-1$	...	$1+(t-1)(i-1)$	$1+(t-1)i$
$t$	2	1	$t$	...	$2+(t-1)(i-1)$	$2+(t-1)i$
$t$	3	2	1	...	$3+(t-1)(i-1)$	$3+(t-1)i$
:	:	:	:	:	:	:
$t$	$t-1$	$t-2$	$t-3$	...	$t-1+(t-1)(i-1)$	$t-1+(t-1)i$
$t$	$t$	$t-1$	$t-2$	...	$t+(t-1)(i-1)$	$t+(t-1)i$

根据测试用例表生成算法所依据的算法模型, 定理中条件所形成的网络为  $t+1$  层, 每层有  $t$  个节点. 每一层节点依次执行完之后, 每个节点内形成的表的后三列按顺序具有特征:  $1ii, 2(i-1)i, 3(i-2)i, \dots, t(i-t+1)i$ . 下面我们来分析通过该算法正好能生成这张正交表.

(1) 第 1 层节点按照前面测试用例生成算法模型中赋予网络的一般计算功能执行之后, 在第 2 层的第 1 个节点中有  $t$  个顺序排列的两位符号串, 分别是  $11, 21, \dots, t1$ . 对应上表前两列中从上到下以 1 结尾的两位位串. 同样, 第 2 个节点当中也有  $t$  个顺序排列的符号串, 分别是  $12, 22, 32, \dots, t2$ , 对应上表中前两列中以 2 结尾的两位位串, ..., 第  $t$  个节点内也有  $t$  个顺序排列的符号串, 分别是  $1t, 2t, 3t, \dots, tt$ , 对应上表中前两列中从上到下以  $t$  结尾的两位位串.

(2) 从第 2 层的每个节点开始, 第 1 个节点把位串  $11, 21, \dots, t1$ , 分别发给第 3 层的 1 号, 2 号, ...,  $t$  号节点, 第 2 个节点把  $12, 22, 32, \dots, t2$  分别发给第

$$C_i = \begin{pmatrix} 1 & 2 & \cdots & t \\ 1+i & 2+i & \cdots & t+i \\ 1+2i & 2+2i & \cdots & t+2i \\ \vdots & \vdots & \vdots & \vdots \\ 1+(t-1)i & 2+(t-1)i & \cdots & t+(t-1)i \end{pmatrix}, \quad i=1, 2, \dots, t-1$$

其中加法以  $t$  为模. 由正交表构造理论, 该正交拉丁方完全组与基本列一起构成正交表, 如下所示

1	...	1	1
2	...	2	2
3	...	3	3
:	...	:	:
$t-1$	...	$t-1$	$t-1$
$t$	...	$t$	$t$
$i+2$	...	$t-1$	$t$
$i+3$	...	$t$	1
$i+4$	...	1	2
:	...	:	:
$i$	...	$t-3$	$t-2$
$i+1$	...	$t-2$	$t-1$
:	...	:	:
$1+(t-1)(i-1)$	$1+(t-1)i$	$1+(t-1)(i+1)$	$1+(t-1)(t-2)$
$2+(t-1)(i-1)$	$2+(t-1)i$	$2+(t-1)(i+1)$	$2+(t-1)(t-2)$
$3+(t-1)(i-1)$	$3+(t-1)i$	$3+(t-1)(i+1)$	$3+(t-1)(t-2)$
:	...	:	:
$t-1+(t-1)(i-1)$	$t-1+(t-1)i$	$t-1+(t-1)(i+1)$	$(t-1)^2$
$t+(t-1)(i-1)$	$t+(t-1)i$	$t+(t-1)(i+1)$	$t+(t-1)(t-2)$

3 层的 2 号, 3 号, ...,  $t$  号, 1 号节点, ..., 第  $t$  个节点把  $1t, 2t, 3t, \dots, tt$  分别发给  $t$  号, 1 号, 2 号, ...,  $t-1$  号节点. 这样第 2 层每个节点执行完毕之后, 在第 3 层每个节点中形成  $t$  个 3 位的符号串; 第 3 层中第  $i$  ( $1 \leq i \leq t$ ) 号节点中按照顺序排列着下面  $t$  个 3 位符号串:  $1ii, 2(i-1)i, 3(i-2)i, \dots, t(i-t+1)i$ . 其中加减运算的结果以  $t$  取模. 这些位串分别对应上表中从上到下前 3 列中以  $i$  结尾的位串.

(3) 依此类推, 第  $t+1$  层每个节点内的符号串输出则构成上面的这张正交表. 证毕.

性质 1 说明了该算法第 1 步(3.1 节)在有些条件下可直接产生满足要求的测试用例表, 这是算法的最佳效果, 此时测试用例数量规模为  $t_1 \times t_2$ , 性质 2 说明这是满足要求的测试用例数量的最小值. 定理 3 给出了这样一种情况, 即当系统每个接口参数都取某素数个不同值, 且参数个数不超过一定值时, 算法的第 1 步就能完成测试用例表的生成. 性质 3 针对该算法模型说明了算法的实际意义是寻求最少路线实现对网络中不同层上的任意两个结点之间的

路径覆盖(图 1).

## 5 测试用例自动生成工具及其效果

为了提高软件的测试效率,降低软件测试成本,我们研究开发了一套测试数据自动生成工具,并且把本算法作为测试数据生成算法的一个重要组成部分加以实现.该算法运行的结果具有很好的效果,下面我们就列举了一组测试用例生成需求,用该生成工具产生了一组相应的测试用例表,测试用例的数量即测试用例表的行数,见表 2.从表 2 第 1 行中可以看到,当系统有 4 个参数,每个参数有 3 个取值时,我们的生成算法生成 9 个用例,而如果对这 4 个参数的各种组合进行全面覆盖,则需要 81 个测试用例.精简测试用例高达 75%.实际上在多数情况下,我们的生成工具产生的测试用例数量远远小于全面覆盖测试要求的测试用例数,关于这一点,从表 2 中的其它几个例子可以看出.如该表中第 2 行表示一个系统有 10 个参数,其中有 3 个参数是每个参数有 5 个取值,3 个参数是每个参数有 4 个取值,1 个参数有 3 个取值,3 个参数是每个参数有 2 个取值时,达到对任意两个参数两两组合的全面覆盖需要的测试用例是 30 个,即测试用例表有 30 行,而对这 10 个参数的各种组合的全面覆盖要 192000 个测试用例,精简了 99.8%.对于定理 3 中论证的结果也在我们的算法中得到验证,如表 2 中的第 1,6 行.

表 2 生成工具效果

测试需求	测试用例数
$3^4$	9
$5^3 \times 4^3 \times 3 \times 2^3$	30
$3^{13}$	19
$4 \times 3 \times 2$	12
$2^{10}$	10
$17^{18}$	289
$2^{126}$	86
$3^{12}4^5$	28

## 6 结束语

本文提出了一种基于接口参数的黑箱测试用例自动生成算法,证明了该算法的有关性质,列举了算法实现的实际效果.说明了该算法产生的测验用例具有数量少、全面覆盖各个参数两两组合等优点.如果我们考虑保证任意两个参数之间各种组合的等概率覆盖,这样也能够精简测试用例,这种方法实际上

就是正交实验设计.它是一种比较有效的测试用例选择方法,然而这种方法依赖于正交表,因为正交表的构造还存在很多未解决的难题,特别是对于混合型的正交表,目前还没有比较好的构造方法,所以这给正交试验设计在黑箱测试用例选择中的应用带来较大的局限<sup>[11,12]</sup>.而在软件测试中,测试用例只要达到对各个参数两两组合的全面覆盖就可以了,并不要求等概率.而且使用正交试验设计产生的测试用例数量在多数情况下要远远多于我们生成的测试用例表中测试用例的数量.例如表 2 中第 2 行 10 个参数的例子如果用正交试验设计的方法至少需要 300 个测试用例,而且生成方法比较复杂.所以测试用例表更适合软件的黑箱测试.

关于测试用例表生成的理论研究的论文还不是很多,Cohen 等曾提出了一种基于组合设计的测试数据启发式生成方法,所产生的测试数据可以根据测试要求实现对系统参数的两两组合覆盖,或者多个参数的组合覆盖<sup>[13,14]</sup>.但这种方法无法保证所产生的测试用例最优.我们关于黑箱测试用例的自动生成算法充分考虑了保证生成结果的最优性,下一步我们还将就一般情况下如何生成最优的测试用例做更进一步的研究.

**致 谢** 史群峰等同学为本工具的开发和实现做了大量的工作.周毓明、陈振强、卢虹等同学参与了有关讨论,华为公司为本研究提供了很好的应用背景,在此一并致谢.

## 参 考 文 献

- 1 Schach S. R.. Software Engineering with Java. Boston: McGraw-Hill, 1999
- 2 DeMillo R. *et al.*. Constraint-based automatic test data generation. IEEE Transactions on Software Engineering, 1991, 17 (9): 900~910
- 3 Grabowski J. *et al.*. On the design of the new testing language TTCN-3. In: Ural H. *et al.* eds.. Testing of Communicating Systems. Kluwer: Academic Publishers, 2000,13: 161~176
- 4 Schieferdecker I. *et al.*. Conformance testing with TTCN. In: Telektronikk, 2000,96(4): 85~95
- 5 DeMillo R. *et al.*. Experimental results from an automatic test case generator. ACM Transactions on Software Engineering Methodology, 1993, 2(2): 109~175
- 6 Offutt J.. An integrated automatic test data generation system. Journal of Systems Integration, 1991, 1(3): 391~409
- 7 Rohermel G. *et al.*. Regression test selection for C++ soft-

- ware. Journal of Software Testing, Verification and Reliability, 2000, 10(2): 77~109
- 8 Offutt J. *et al.*. The dynamic domain reduction approach to test data generation. Software Practice and Experience, 1999, 29(2): 167~193
- 9 Chen T. Y. *et al.*. A new heuristic for test suite reduction. Information and Software Technology, 1998, 40(5~6): 347~354
- 10 Walter T. *et al.*. A framework for the specification of test cases for real-time distributed systems. Information and Software Technology, 1999, 41(11~12): 781~798
- 11 De Cock D. *et al.*. On finding mixed orthogonal arrays of strength 2 with many 2-level factors. Statistics & Probability Letters, 2000, 50(4): 383~388
- 12 Hedayat A. S. *et al.*. Orthogonal Arrays: Theory and Applications. New York: Springer-Verlag, 1999
- 13 Cohen D. M. *et al.*. The AETG system: An approach to testing based on combinatorial design. IEEE Transactions on Software Engineering, 1997, 23(7): 437~444
- 14 Cohen D. M. *et al.*. The combinatorial design approach to automatic test generation. IEEE Software, 1996, 13(5): 83~88



**NIE Chang-Hai**, born in 1971, Ph. D. candidate, lecturer. His research and teaching are in software engineering, software testing, fuzzy information processing, neural network and etc.

## Background

This research is supported by the National Science Foundation of China under Grant No. 60373066, and the title is “Research on Techniques of Software Testing Based on Combinatorial Coverage”. The task is to research on the combinatorial coverage method for software testing, its scientific effectiveness, and related problems. The research

**XU Bao-Wen**, born in 1961, Ph. D., professor, Ph. D. supervisor. His main research and teaching are in programming language, software engineering, parallel and network software, knowledge and information capture technology.

group has made a lot of work about the techniques of software testing based on combinatorial coverage, such as the design and implement of several algorithms and tools for test data generation, the methods for debugging based on combinatorial testing and so on. This paper is a part of test data generation algorithms for different coverage requirements.