相邻因素组合测试用例集的最优生成方法

王子元 聂长海 徐宝文 史 亮
（东南大学计算机科学与工程学院 南京 210096）
（江苏省软件质量研究所 南京 210096）

摘 要 软件系统是一个复杂的逻辑系统，有很多因素可能影响系统的正常运行，组合测试可以对这些因素及其相互作用可能对系统产生的影响进行检测. 针对一类只在相邻因素间存在相互作用的系统，文中提出了相邻因素组合测试的概念，分别给出了相邻因素两两（二维）组合覆盖表、相邻因素 $N(N\geq 2)$ 维组合覆盖表和多重维数相邻因素组合覆盖表的生成算法，并证明了 3 个算法均可以生成数量最优的相邻因素组合测试用例集. 最后通过实际应用场景，分析了相邻因素组合测试的应用价值.

关键词 软件测试；相邻因素；组合测试；测试用例生成
中图法分类号 TP311

Optimal Test Suite Generation Methods for Neighbor Factors Combinatorial Testing

WANG Zi-Yuan NIE Chang-Hai XU Bao-Wen SHI Liang
(*School of Computer Science and Engineering*, *Southeast University*, *Nanjing* 210096)
(*Jiangsu Institute of Software Quality*, *Nanjing* 210096)

Abstract As a complex logic system, software may be affected by many factors. Combinatorial testing can detect the faults triggered by these factors and their interactions. For one kind of software in which the interactions only exist between neighbor factors, this paper represents the concept of neighbor factors combinatorial testing, and describes the covering array generation algorithms for neighbor factors pair-wise ($N=2$) coverage, neighbor factors N-way ($N\geq 2$) coverage and variable strength neighbor factors coverage, then proves that the test suites generated by these three algorithms are optimal. Finally the authors analyze a practical application scene, which shows that neighbor factors combinatorial testing is very practical.

Keywords software testing; neighbor factors; combinatorial testing; test suite generation

1 引 言

软件测试是软件开发过程中的一个重要的环节，同时也是一个成本高昂的环节. 近年来，软件测试越来越受到人们的普遍重视和广泛研究. 一方面，对于具体的待测软件系统，人们研究应该从哪些方面对其进行科学系统的测试；另一方面，研究如何生成数量少、质量高的测试用例集来对系统进行充分有效的测试，以提高效率，降低成本.

作为一个复杂的逻辑系统，有很多因素可能影响软件系统的正常运行. 这些因素可能是系统的配

收稿日期：2005-12-26；修改稿收到日期：2006-09-14. 本课题得到国家杰出青年科学基金（60425206）、国家自然科学基金重点项目（60633010）、国家自然科学基金（60373066，60403016）、江苏省自然科学基金（BK2005060）、江苏省高技术研究项目基金（BG2005032）以及东南大学优秀青年教师基金资助. **王子元**，男，1982 年生，博士研究生，主要从事软件测试领域的研究. E-mail：wangziyuan@seu.edu.cn. **聂长海**，男，1971 年生，博士，副教授，主要从事软件工程和软件测试技术、模糊信息处理、神经网络等方面的教学与科研工作. **徐宝文**，男，1961 年生，博士，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. E-mail：bwxu@seu.edu.cn. **史 亮**，男，1979 年生，博士，主要从事软件分析与测试领域的研究.

置、内部事件、外部输入等. 人们已经开始研究如何有效地检测系统中各个因素以及它们之间的相互作用对系统产生的影响. 研究工作主要集中在组合测试中测试数据的生成以及测试结果的分析等方面. 但是在实际的软件系统中，一般很少出现任意两个因素之间都存在相互作用的情况，更多的情况是有些因素间存在着较强的相互作用，而有些因素间则没有任何相互作用. 在对组合测试进行大量研究的基础上，本文针对一类只在相邻因素间存在相互作用的软件系统，提出了相邻因素组合测试的概念，给出了相邻因素组合测试用例集的最优生成方法，分别讨论了二维、$N(N>2)$维和多重维数相邻因素组合覆盖表的生成，完整地解决了相邻因素组合测试用例的生成问题. 在此基础上，我们还开发了一套组合测试用例集的自动生成工具，实现了本文给出的所有算法.

本文第 2 节介绍了相关的研究工作并分析了存在的问题；第 3 节介绍了基本模型和相关定义；第 4 节首先给出相邻因素两两(二维)组合覆盖表的生成算法，然后将情况推广到 $N(N\geq 2)$维的情况，给出了相邻因素 N 维组合覆盖表的生成算法，并证明了该算法可以生成数量最优的测试用例集；第 5 节在上述算法的基础上给出了最优多重维数相邻因素组合覆盖表的生成算法；第 6 节分析了相邻因素组合覆盖表对任意因素间组合的覆盖率并对算法做了进一步改进；第 7 节通过一个应用场景说明相邻因素组合测试的应用价值；最后对全文进行总结并指出未来的研究方向.

2　相关工作及存在的问题

1985 年 Mandl 最早提出参数两两组合覆盖的概念，将此概念应用于 Ada 编译器的测试，并利用正交拉丁方来产生测试用例集[1].

1992 年，Brownlie 报告了使用正交表对 PMX/StarMAIL 系统进行测试的实例研究，他们利用 OATS 工具产生正交表，使用较少的测试用例检测出很多以前无法发现的错误[2].

20 世纪 90 年代后期，贝尔实验室将组合测试方法的研究推向了一个新的高潮，他们提出一种基于贪心策略的组合测试用例生成算法，并开发了 AETG 系统用于产生组合测试用例集[3-5].

2001 年，Schroeder 在其博士论文中提出了利用软件附加信息对组合测试的测试用例集进行约简和优化的方法[6]. 同年，Salem 在其博士论文中系统研究了如何将试验设计方法应用于软件测试[7]. Lei 和 Tai 则提出了一种基于参数顺序扩充的两两组合测试数据生成策略，并开发了 PairTest 系统用于产生两两组合的测试用例[8-9].

2002 年，Kuhn 和 Reilly 研究了组合测试的可用性，发现大约 70％的故障是由两个以下参数的相互作用引起的，由此说明了两两组合测试的应用价值[10]. 同年，Kobayashi 和 Tsuchiya 提出了一种生成两两组合测试数据的代数方法[11]. William 在其博士论文[12]中不仅研究了一套用于生成两两组合覆盖表的代数方法，同时还深入研究并提出了组合测试的 8 个应用场景.

2003 年，Cohen M. B. 等人将模拟退火算法应用于多重维数组合测试用例的生成，该方法综合考虑了运算效率和运算结果之间的关系，可以根据时间和效率上不同的要求来生成尽可能优的结果[13-15].

2004 年，Kuhn 和 Wallace 通过实验进一步研究了大规模分布式系统中组合测试可用性，发现此类系统中的故障一般最多由 4～6 个参数的相互作用引发[16]. 同年，Shiba 和 Tsuchiya 等人研究了遗传算法和蚁群算法在组合测试数据自动生成中的应用[17]. Colbourn 和 Cohen M. B. 等人提出一种两两组合测试数据生成的确定性密度算法[18]. Schroeder 等人通过实验比较了 N 维组合测试与相同规模随机测试的错误检测能力[19].

2005 年，Colbourn 和 Sherwood 等人研究了两种组合覆盖表的生成方法，分别应用于具有不同参数取值的两两组合覆盖表和高维组合覆盖表的生成[20-21].

但是在实际的软件系统中，一般很少出现任意两个因素之间都存在相互作用的情况，更多的情况是有些因素间存在着较强的相互作用，而有些因素间则没有任何相互作用. 其中一种比较特殊的情况是这种相互作用只存在于相邻因素之间，例如：

(1) 一个具有多个输入和输出参数的软件，其中每个输出参数仅被部分输入参数所影响，而这些

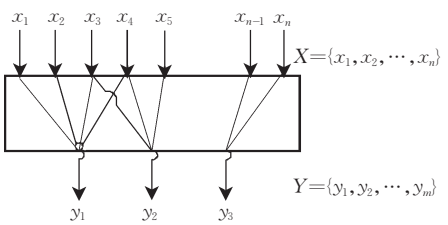

图 1　程序的输入输出关系

能够影响某一个输出的输入参数可能是相邻的，如图 1 所示；

(2) 一个面向对象软件中存在多个类体系结构(class clusters)，不同类体系结构的子类之间存在交互. 为了降低软件的复杂度，可以在设计时令这种交互只存在于相邻的类体系结构之间，如图 2 所示.

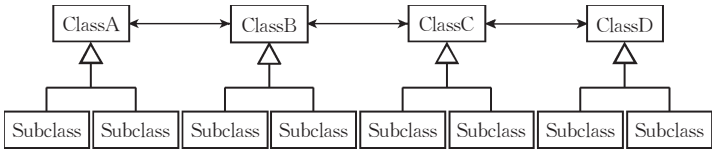

图 2 面向对象软件的类体系结构及其交互

针对上述情况，本文提出的相邻因素组合测试方法对任一给定维数 $N(N\geq 2)$ 都可以生成相邻因素 N 维组合测试用例集，检测相邻 N 个因素间的相互作用. 在某些情况下，系统中的一些模块对安全性有更高的要求，有必要对影响该模块的因素进行更高强度的测试、相邻因素组合测试可以生成多重维数相邻因素组合测试用例集来满足这样的测试需求.

虽然一般的组合测试也可以实现对相邻因素间相互作用的检测，但是基于以下原因，我们需要研究相邻因素组合测试用例集的生成方法：(1) $N(N\geq 2)$ 维组合覆盖表的构造问题是 NP-C 问题[8]，目前的方法均无法产生最优解；(2) 如果只考虑相邻因素间的组合覆盖，可以生成最优(即数量最少)的测试用例集；(3) 在某些情况下，即使约简数量很少的测试用例，对成本的节约和效率的提高也具有不可忽视的作用.

近几年来，我们在组合测试用例的生成算法、测试用例自动生成工具、组合测试的应用以及基于组合测试的故障诊断等方面也进行了大量研究[22-27]. 本文所研究的相邻因素组合测试是组合测试的一种特例，是以往对组合测试所作研究的进一步延伸.

3 基本模型和相关定义

假设影响待测软件 SUT 的因素共有 n 个，形成有限集 $F=\{f_1,f_2,\cdots,f_n\}$，其中因素 f_i 有 $a[i]$ 个可能的取值，从而形成有限集 $V_i(1\leq i\leq n)$，不妨设 $V_i=\{1,2,\cdots,a[i]\}$.

定义 1. 称 n 元组 $(v_1,v_2,\cdots,v_n)(v_1\in V_1, v_2\in V_2,\cdots,v_n\in V_n)$ 为 SUT 的一个测试用例.

定义 2. 设 $\boldsymbol{A}=(a_{i,j})_{m\times n}$ 为一个 $m\times n$ 的矩阵，其第 j 列表示 SUT 的参数 f_j，第 j 列的所有元素均取自集合 $V_j(j=1,2,\cdots,n)$，即 $a_{i,j}\in V_j$. 给定正整数 $N(N\geq 2)$，如果 A 中任意 N 列(不妨设为第 $i_1,i_2,\cdots,i_N$ 列)均满足：$V_{i1},V_{i2},\cdots,V_{iN}$ 中符号的所有的 N 维组合均在这 N 列所形成的 N 元有序组中等频率出现，则称 A 为正交表，记为 $OA(m;N,F)$；如果所有的组合均至少出现一次但不要求等频率，则称 A 为 N 维组合覆盖表，记为 $CA(m;N,F)$；如果 A 只保证任意相邻的 N 列(不妨设为第 $i,\cdots,i+N-1$ 列)均满足：$V_i,\cdots,V_{i+N-1}$ 中符号的所有的 N 维组合均在这 N 列所形成的 N 元有序组中至少出现一次，则称 A 为相邻因素 N 维组合覆盖表，记为 $NCA(m;N,F)$. A 中的每一行均表示一条测试用例，m 为测试用例集中测试用例的数量. 对于某一个(相邻因素) N 维组合覆盖表 A，如果 m 是能够保证上述条件成立的最小正整数，则称 A 是最优(相邻因素) N 维组合覆盖表.

定义 3. 对于相邻因素 $N(N\geq 2)$ 维组合覆盖表 A，如果 F 中存在 $t(t\geq 1)$ 个不同的子集 $F_i\subseteq F$，其中 $F_i(i=1,2,\cdots,t)$ 包含 n_i 个相邻参数 $f_{i,1}, f_{i,2},\cdots,f_{i,n_i}$，这些参数对应的列所组成的 $m\times n_i$ 的子矩阵 $\boldsymbol{A}_i$ 满足相邻因素 $N_i(n_i\geq N_i>N)$ 维组合覆盖表的条件，则称 A 是多重维数相邻因素组合覆盖表，记为 $VNCA(m;N,F,NC)$，其中 NC 包含 t 个相邻因素 N_i 维组合覆盖表 $NCA(m_i;N_i,F_i)$. 对于某一个多重维数相邻因素组合覆盖表 A，如果 m 是能够保证上述条件成立的最小的正整数，则称 A 是最优多重维数相邻因素组合覆盖表.

定义 4. 利用正交表生成测试用例集的方法称为正交试验设计法；利用组合覆盖表生成测试用例集的方法称为组合测试方法；利用相邻因素组合覆盖表生成测试用例集的方法称为相邻因素组合测试方法.

从定义 2 可知，正交表的条件强于 N 维组合覆盖表，而相邻因素 N 维组合覆盖表的条件最弱. 如果把用正交实验设计法生成的测试用例集记为 T_{OA}，把用组合测试方法生成的测试用例集记为 T_{CA}，把用相邻因素组合测试方法生成的测试用例集记为 T_{NCA}，则 3 个集合中的测试用例数量一般有以下关系：$|T_{NCA}|\leq|T_{CA}|\leq|T_{OA}|$.

定义 5.　对于某一个 SUT，其中某 t 个参数 $f_{i1}, f_{i2}, \cdots, f_{it}(1 \le t \le n)$ 的所有组合的数量为 $Ct = a[i_1] \times a[i_2] \times \cdots \times a[i_t]$. 设 T_{NCA} 为该 SUT 的相邻因素组合测试用例集，被 T_{NCA} 覆盖的这 t 个参数的组合数为 CtA. 则 CtA 与 Ct 的比值称为 T_{NCA} 对这 t 个参数的组合覆盖率，简称覆盖率，记为 $p = \frac{CtA}{Ct}$.

引理 1.　设系统 SUT 的相邻因素 N 维组合测试用例集为 T_{NCA}，则 T_{NCA} 中至少需要有 $\max\limits_{1 \le i \le n-N+1}\left(\prod\limits_{k=i}^{i+N-1} a[k]\right)$ 条测试用例，即

$$|T_{NCA}| \ge \max_{1 \le i \le n-N+1}\left(\prod_{k=i}^{i+N-1} a[k]\right).$$

证明.　SUT 的任意 N 个相邻因素 $f_i, f_{i+1}, \cdots, f_{i+N-1}(1 \le i \le n-N+1, 1 \le N \le n)$ 之间的 N 维组合共有 $\prod\limits_{k=i}^{i+N-1} a[k]$ 个，而每条测试用例最多只能覆盖其中的一个组合，因此，T_{NCA} 中至少需要 $\max\limits_{1 \le i \le n-N+1}\left(\prod\limits_{k=i}^{i+N-1} a[k]\right)$ 条测试用例.　　证毕.

定理 1.　设系统 SUT 的相邻因素 N 维组合测试用例集为 T_{NCA}，则 T_{NCA} 对任意 $t(t \ge N)$ 个相邻参数 $f_i, f_{i+1}, \cdots, f_{i+t-1}(1 \le i \le n-t+1)$ 的组合覆盖率 $p \ge \max\limits_{i \le j \le i+t-N}\left(\prod\limits_{k=j}^{j+N-1} a[k]\right) \Big/ \prod\limits_{k=i}^{i+t-1} a[k]$.

证明.　由相邻因素 N 维组合覆盖表的定义可知，相邻因素 N 维组合测试用例集 T_{NCA} 覆盖了所有的相邻因素 N 维组合，因此当 $t=N$ 时，有 $p=1$. 结论正确.

当 $t>N$ 时，对任意 t 个相邻参数 $f_i, f_{i+1}, \cdots, f_{i+t-1}$，在 T_{NCA} 中至少有 $\max\limits_{i \le j \le i+t-N}\left(\prod\limits_{k=j}^{j+N-1} a[k]\right)$ 个不同的组合，而这 t 个参数所有组合的数量是 $\prod\limits_{k=i}^{i+t-1} a[k]$，所以 T_{NCA} 对这 t 个参数的覆盖率满足：

$$p \ge \max_{i \le j \le i+t-N}\left(\prod_{k=j}^{j+N-1} a[k]\right) \Big/ \prod_{k=i}^{i+t-1} a[k].$$　证毕.

4　相邻因素组合覆盖表

首先给出相邻因素两两(二维)组合覆盖表的生成算法. 通过对该算法进行改进，进而给出相邻因素 $N(N \ge 2)$ 维组合覆盖表的生成算法. 我们还将证明使用该算法可以得到数量最优的测试用例集.

4.1　相邻因素两两组合覆盖表生成算法

下面给出相邻因素两两组合覆盖表的生成算法. 首先找出 SUT 中相邻两个因素间的最大组合数 $m = \max\limits_{1 \le i \le n-1}(a[i] \times a[i+1])$. 我们使用一个 $m \times n$ 的二维数组来存放相邻因素两两组合覆盖表. 将参数 f_1 和 f_2 的 $a[1] \times a[2]$ 个组合依次填入数组相应位置，此时仍有 $m - a[1] \times a[2]$ 行中参数 f_1 和 f_2 的值没有确定，如表 1(a)所示.

在此基础上从左向右依次扩展参数. 此时参数 f_2 的每一个值均出现 $a[1]$ 次，如果 $a[1] < a[3]$，则需要对 f_2 进行补充，使得 f_2 的每一个值的出现次数达到 $a[3]$(可以在 m 行之内实现，这是由于 $a[3] \times a[2] \le m$，否则矛盾)，如表 1(b)所示. 在参数 f_2 的每个值的前 $a[3]$ 次出现之后依次将参数 f_3 的 $a[3]$ 个值分别填入数组的第 3 列中相应的位置，此时仍有 $m - a[2] \times a[3]$ 行中参数 f_3 的值尚未确定，如表 1(c)所示.

表 1　相邻因素两两组合覆盖表的构造过程($F=[2 \times 3^2 \times 2]$)

(a)				(b)				(c)				(d)			
a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d
1	1	-	-	1	1	-	-	1	1	1	-	1	1	1	1
1	2	-	-	1	2	-	-	1	2	1	-	1	2	1	2
1	3	-	-	1	3	-	-	1	3	1	-	1	3	1	-
2	1	-	-	2	1	-	-	2	1	2	-	2	1	2	1
2	2	-	-	2	2	-	-	2	2	2	-	2	2	2	2
2	3	-	-	2	3	-	-	2	3	2	-	2	3	2	-
-	-	-	-	-	1	-	-	-	1	3	-	-	1	3	1
-	-	-	-	-	2	-	-	-	2	3	-	-	2	3	2
-	-	-	-	-	3	-	-	-	3	3	-	-	3	3	-

依此类推，依次扩展参数直至 f_n，详见算法 1. 此时，生成覆盖表中可能存在一些位置尚未被赋值，如表 1(d)所示. 这些位置的存在对相邻因素组合的覆盖率不产生影响，因此称这些位置上的值为“无关值”，任意填入 $x \in V_i$ 即可.

算法 1 的 2～3 行确定最大的相邻因素两两组合数 m；第 4 行初始化一个 $m \times n$ 的数组 T 作为空表，并用 T 来存放覆盖表；5～7 行将相邻的参数 f_1 和参数 f_2 的所有两两组合依次填入数组 T 的第 1 列和第 2 列；8～21 行依次扩展参数 $f_i(3 \le i \le n)$. 最后产生的数组 T 就是所求的相邻因素两两组合覆盖表. 整个算法的最坏时间复杂性是 $O(m^2 n)$.

算法 1.　相邻因素两两组合覆盖表生成算法.

输入：参数集 F

输出：相邻因素两两组合覆盖表 $T[m,n]$

步骤：

1. $m=0$；$k=1$；

```
2. for i=1 to n-1    //计算测试用例数量
3.   {if m<a[i]×a[i+1] then m=a[i]×a[i+1];}
4. Initialize T[m,n]; //初始化矩阵 T
5. for i=1 to a[1]    //处理参数 f1和 f2
6.   for j=1 to a[2]
7.     {T[k,1]=i; T[k,2]=j; k++;}
8. for i=3 to n{       //向右依次扩展参数
9.   if a[i-2]<a[i] then{ //对参数 fi-1进行补充
10.     d=1;
11.     for j=1 to a[i]-a[i-2]
12.       for k=1 to a[i-1]{
13.         if T[d,i-1]=0 then
14.           T[d,i-1]=k;
15.         d++;}}
16.   for j=1 to a[i-1] //初始化 fi-1值的出现次数
17.     {count[j]=0;}
18.   for j=1 to m{        //为参数 fi赋值
19.     count[T[j,i-1]]++;
20.     if count[T[j,i-1]]≤a[i] then
21.       T[j,i]=count[T[j,i-1]];}}
```

由算法 1 的描述可以看出，该算法生成的测试用例集规模为 m. 而由引理可知，相邻因素两两组合覆盖表中至少有 m 条测试用例. 如果该算法生成的测试用例集确实覆盖了所有的相邻因素两两组合，则说明该算法生成的测试用例集是最优的. 我们将在以下的部分中证明该问题.

4.2 相邻因素 N 维组合覆盖表生成算法

与相邻因素两两组合覆盖表的生成方法类似，首先找出 SUT 中相邻 N 个因素间的最大组合数 $m=\max\limits_{1\le i\le n-N+1}\left(\prod\limits_{k=i}^{i+N-1}a[k]\right)$. 我们使用一个 $m\times n$ 的二维数组来存放相邻因素 N 维组合覆盖表. 将相邻参数 $f_1,f_2,\cdots,f_N$ 的 $\prod\limits_{k=1}^{N}a[k]$ 个组合填入数组中相应的位置，此时仍有 $m-\prod\limits_{k=1}^{N}a[k]$ 行中参数 $f_1,f_2,\cdots,f_N$ 的值没有确定.

在此基础上从左向右依次扩展参数. 对于参数 f_{N+1} 来说，由于参数 $f_2,f_3,\cdots,f_N$ 的每一个 $N-1$ 维组合均出现 $a[1]$ 次，如果 $a[1]<a[N+1]$，则需要对这些 $N-1$ 维组合进行补充，使每一个 $N-1$ 维组合的出现次数达到 $a[N+1]$（可以在现有的 m 行之内实现，证明见定理 2）. 具体方法是，针对每一个组合，首先在上述 $m-\prod\limits_{k=1}^{N}a[k]$ 行中寻找一行，该行中参数 $f_2,f_3,\cdots,f_N$ 对应的位置与该 $N-1$ 维组合的匹配程度最高，且不匹配的位置上均未赋值，填充该行中未赋值的位置，使得该行在参数 $f_2,f_3,\cdots,f_N$ 对应的位置上出现此 $N-1$ 维组合，依此类推，直至参数 $f_2,f_3,\cdots,f_N$ 对应的列中该 $N-1$ 维组合总的出现次数达到 $a[N+1]$. 然后在每一个 $N-1$ 维组合的前 $a[N+1]$ 次出现之后依次将参数 f_{N+1} 的 $a[N+1]$ 个值分别填入第 $N+1$ 列中相应的位置，此时共有 $m-\prod\limits_{k=2}^{N+1}a[k]$ 行中参数 f_{N+1} 的值没有确定.

依此类推，依次扩展参数直至 f_n，详见算法 2. 未赋值位置的处理同算法 1.

算法 2. 相邻因素 N 维组合覆盖表生成算法.

输入：参数集 F，维数 N

输出：相邻因素 N 维组合覆盖表 $T[m,n]$

步骤：

```
1. m=0; e=1;
2. for i=1 to n-N+1 //计算测试用例数量
3.   {m[i]=a[i]×a[i+1]×…×a[i+N-1];}
4. m=max{m[i]};
5. Initialize T[m,n]; //初始化矩阵 T
6. for i=1 to N{       //处理参数 f1,f2,…,fN
7.   d=1;
8.   for j=1 to e
9.     for k=1 to a[i]
10.       for p=1 to m[1]/(e*a[i]){T[d++,i]=j;}
11.   e=e*a[i];}
12. for i=N+1 to n{  //向右依次扩展参数
13.   if a[i-N]<a[i] then{
14.     for 参数 fi-N+1,…,fi-1的所有 N-1 维组合{
15.       找出与该组合不匹配但最相似的行，按此组合
          补足该行中的值，不断重复直至该 N-1 维组
          合的出现次数达到 a[i];}}
16.   for j=1 to m{
17.     记录第 j 行出现的参数 fi-N+1,…,fi-1的N-1
        维组合的当前出现次数 count;
18.     if count≤a[i] then T[j,i]=count;}}
```

算法 2 的 2～4 行确定最大的相邻因素 N 维组合数 m；第 5 行初始化一个 $m\times n$ 的数组 T 作为空表，并用 T 来存放覆盖表；6～11 行将相邻的参数 $f_1,f_2,\cdots,f_N$ 的所有 N 维组合依次填入数组 T 的第 $1,2,\cdots,N$ 列；12～18 行依次扩展参数 f_i（$N+1\le i\le n$）. 最后产生的数组 T 就是所求的相邻因素 N 维组合覆盖表.

下面我们分析算法 2 的时间复杂性. 算法 2 处理前 N 个参数的时间复杂度为 $O(mN)$. 在扩展参

数的过程中，共有 $n-N$ 个参数需要扩展. 在扩展某个参数 f_i 时，如果 $a[i-N]<a[i]$，则最多需要考虑 m 个 $N-1$ 维组合，在寻找最匹配行的过程中需要比较所有 m 行，每行需比较 $N-1$ 个参数值. 所以算法 2 的最坏时间复杂度是 $O(mN+(n-N)m^2N)$.

4.3　算法的最优性质

定理 2.　针对系统 SUT，算法 2 生成的测试用例集 T_{NCA} 的规模为 m.

证明.　通过算法 2 的描述可以看出，如果该算法能够按照预期执行，则生成的测试用例的数量为 m. 其中的关键在于：假设参数 f_{i+N-1} 已被处理，扩展参数 $f_{i+N}(i\geq 1)$ 时，若 $a[i]<a[i+N]$，对参数 $f_{i+1},\cdots,f_{i+N-1}$ 的每一个 $N-1$ 维组合进行补充，使其出现次数达到 $a[i+N]$ 的操作是否可以在现有的 m 个测试用例之内实现. 因此只需证明在现有的 m 个测试用例之内可以实现这一操作即可.

已知相邻参数 $f_{i+1},\cdots,f_{i+N-1}$ 的每一个 $N-1$ 维组合的出现次数均为 $a[i]$，在剩余的 $m-\prod_{k=i}^{N+i-1}a[k]$ 行中不会出现参数 $f_{i+1},\cdots,f_{i+N-1}$ 的完整的 $N-1$ 维组合，有以下两种情况：① $N-1$ 个参数均未赋值；②从 f_{i+1} 开始的连续 $u(0<u<N-1)$ 个参数已赋值，其余参数未赋值.

当所有剩余行都属于情况①时，由于 $m=\max_{1\leq i\leq n-N+1}\left(\prod_{k=i}^{i+N-1}a[k]\right)\geq\prod_{k=i+1}^{N+i}a[k]$，现有的 m 个测试用例可以保证参数 $f_{i+1},\cdots,f_{i+N-1}$ 的每一个 $N-1$ 维组合均出现 $a[i+N]$ 次.

当剩余的行中存在属于情况②的用例时，由于其中出现的已赋值位置是在对该参数进行扩展时加入的，所以这些参数的任何一个组合在这些剩余行中的出现都是均匀的，因此可以将这些组合扩充为所需的 $N-1$ 维组合，使这些 $N-1$ 维组合的出现次数保持均匀. 与情况①同理，可以保证参数 $f_{i+1},\cdots,f_{i+N-1}$ 的每一个 $N-1$ 维组合均出现 $a[i+N]$ 次. 定理得证.　　证毕.

定理 3.　针对系统 SUT，算法 2 生成的测试用例集 T_{NCA} 可以覆盖所有相邻因素 N 维组合.

证明.　首先将参数 $f_1,f_2,\cdots,f_N$ 的 $\prod_{k=1}^{N}a[k]$ 个组合填入数组中相应的位置，生成 $\prod_{k=1}^{N}a[k]$ 条测试用例，其中每一条测试用例覆盖关于参数 $f_1,f_2,\cdots,f_N$ 的一个 N 维组合，其它参数的取值待定. 此时这些测试用例已经将参数 $f_1,f_2,\cdots,f_N$ 所有的 N 维组合全部覆盖.

然后从左向右依次扩展参数. 使用数学归纳法，假设参数 f_{i+N-1} 已经被处理，相邻参数 $f_i,f_{i+1},\cdots,f_{i+N-1}$ 的所有 N 维组合均被覆盖，考虑参数 f_{i+N} $(1\leq i\leq n-N)$，按照算法 2 所描述的方法使参数 $f_{i+1},\cdots,f_{i+N-1}$ 的每一个 $N-1$ 维组合均出现 $a[i+N]$ 次后，对参数 f_{i+N} 进行扩展，在每一个 $N-1$ 维组合的前 $a[i+N]$ 次出现之后依次将参数 f_{i+N} 的 $a[i+N]$ 个值分别填入第 $i+N$ 列中相应的位置，这样参数 $f_{i+1},\cdots,f_{i+N}$ 的所有 N 维组合均被覆盖. 依此类推，故可得证.　　证毕.

由定理 2 和定理 3 可知，算法 2 可以生成最优的相邻因素 N 维组合覆盖表. 同理，算法 1 也可以生成最优的相邻因素两两组合覆盖表.

5　多重维数相邻因素组合覆盖表

在相邻因素 $N(N\geq 2)$ 维组合覆盖表最优生成算法的基础上，本文进一步给出多重维数相邻因素组合覆盖表的生成算法. 参数集合 F 的子集 $F_i(i=1,2,\cdots,t)$ 之间可能存在的关系有独立、包含、相交三种情况，我们就这三种情况分别进行讨论.

5.1　子集 F_i 相互独立的情况

对于一个多重维数相邻因素组合覆盖表 $VNCA(m;N,F,NC)$，其中 NC 包含 t 个相邻因素组合覆盖表 $NCA(m_i;N_i,F_i)(i=1,2,\cdots,t)$，$F$ 的 t 个子集 $F_i\subseteq F(i=1,2,\cdots,t)$ 相互独立，依 i 的值升序排列. 首先对每一个相邻因素组合覆盖表 $NCA(m_i;N_i,F_i)$ 找出 F_i 中相邻 N_i 个因素的最大 N_i 维组合数 m_i 以及相邻因素 N 维组合覆盖表 $NCA(m_0;N,F)$ 的规模 m_0，令 $m=\max_{0\leq i\leq t}\{m_i\}$. 我们使用一个 $m\times n$ 的二维数组来存放多重维数相邻因素组合覆盖表.

首先，如果参数 $f_1\in F_1$，则按照生成相邻因素 N_1 维组合覆盖表的方法生成 $NCA(m_1;N_1,F_1)$，之后按照生成相邻因素 N 维组合覆盖表的方法依次扩展其后的参数直至 F_2 中的第 N_2-1 个参数 f_{2,N_2-1}；否则按照生成相邻因素 N 维组合覆盖表的方法依次处理参数直至 F_1 的第 N_1-1 个参数 f_{1,N_1-1}，之后按照生成相邻因素 N_1 维组合覆盖表的方法扩展参数生成 $NCA(m_1;N_1,F_1)$，再按照生成相邻因素 N 维组合覆盖表的方法依次扩展其后的参数直至 F_2 中的第 N_2-1 个参数 f_{2,N_2-1}. 然后按照生成相邻因素 N_2 维组合覆盖表的方法扩展参数

生成 $NCA(m_2;N_2,F_2)$，之后按照生成相邻因素 N 维组合覆盖表的方法来依次扩展其后的参数直至 F_3 中的第 N_3-1 个参数 f_{3,N_3-1}. 依次类推，详见算法 3. 未赋值位置的处理同算法 1.

算法 3 的第 1～4 行确定多重维数相邻因素组合覆盖表的规模 m；第 5 行初始化一个 $m\times n$ 的数组 T 作为空表，并用 T 来存放多重维数相邻因素组合覆盖表；若参数 $f_1\in F_1$，第 6 行生成 $NCA(m_1;N_1,F_1)$，否则第 7～8 行先按照相邻因素 N 维组合的方式处理参数，然后再生成 $NCA(m_1;N_1,F_1)$；第 9 行继续按照 N 维组合的方式扩展参数至 f_{2,N_2-1}；第 10～14 行按照类似的方法依次处理 $F_i(i=1,2,\cdots,t)$，直至所有参数均被处理. 最后产生的数组 T 就是所求的多重维数相邻因素组合覆盖表. 设 $N_{\max}=\max\limits_{0\le i\le t}\{N_i\}$，根据算法 1 和算法 2 的时间复杂度，可以得到算法 3 的最坏时间复杂度为 $O(mN_{\max}+(n-N_{\max})m^2N_{\max})$.

算法 3. 多重维数相邻因素组合覆盖表生成算法.

输入：参数集 F，维数 N，参数子集 F_i，对应维数 N_i $(1\le i\le t)$

输出：多重维数相邻因素组合覆盖表 $T[m,n]$

步骤：

1. $m=|NCA(m_0;N,F)|$；
2. for $i=1$ to t $\{m[i]=|NCA(m_i;N_i,F_i)|;\}$
3. for $i=1$ to t {
4. if $m<m[i]$ then $m=m[i]$;}
5. Initialize $T[m,n]$；
6. if $f_1\in F_1$ then 生成 $NCA(m_1;N_1,F_1)$；
7. else {对参数 $1,2,\cdots,f_{1,N_1-1}$ 生成 N 维 NCA；
8. 扩展参数生成 $NCA(m_1;N_1,F_1)$；}
9. 按照 N 维的方式扩展参数至 f_{2,N_2-1}；
10. for $i=2$ to t{
11. 扩展参数生成 $NCA(m_i;N_i,F_i)$；
12. if $i!=t$ then
13. 按照 N 维的方式扩展参数至 $f_{i+1,N_{i+1}-1}$；
14. else 按照 N 维的方式扩展至 f_n；}

5.2 子集 F_i 存在相交和包含的情况

下面讨论当 F 的 t 个子集 $F_i\subseteq F(i=1,2,\cdots,t)$ 存在相交和包含时的两种情况：

① 存在 $F_i\subseteq F_j(i\ne j)$. 在处理子集 F_j 时，再次调用算法 3 生成一个关于参数子集 F_j 的多重维数相邻因素组合覆盖表 $VNCA(m_j;N_j,F_j,NC)$，其中 NC 包含 $NCA(m_i;N_i,F_i)$ 即可.

② 存在 $F_i\cap F_j\ne\varnothing$ 且 $F_i\ne F_j(i\ne j)$. 不妨设 $j=i+1$，令 $F_{ij}=F_i\cap F_j$，$F_i'=F_i-F_{ij}$，$F_j'=F_j-F_{ij}$，$N_{ij}=\max\{N_i,N_j\}$，则 F 所包含的子集由 $F_1,\cdots,F_i,F_j,\cdots,F_t$ 调整为 $F_1,\cdots,F_i',F_{ij},F_j',\cdots,F_t$. 使用算法 3 按照新的参数子集生成多重维数相邻因素组合覆盖表 $VNCA(m;N,F,NC)$ 即可，其中在处理 F_{ij} 和 F_j' 时略做修改，具体处理为 F_{ij} 的前 $N_{ij}-1$ 个参数时按照 N_i 维的方式进行扩展，处理 F_j' 的前 N_j-1 个参数时按照 N_j 维的方式进行扩展.

5.3 算法的最优性质

定理 4. 针对系统 SUT，算法 3 生成的多重维数相邻因素组合测试用例集 T_{VNCA} 是满足覆盖要求的最优测试用例集.

证明. 首先应证明该算法生成的测试用例的数量为 m，由定理 2 可知，对每一个参数的处理均可以在现有的 m 个测试用例内完成，因此算法可行，生成规模为 m 的测试用例集. 又由 m 的定义可知，至少需要 m 条测试用例才能保证覆盖所有需要的组合.

其次，定理 3 已表明每一步参数扩展都能保证覆盖所有该参数与前 $N-1$ 个参数之间的 N 维组合，而算法 3 中对参数的处理方法与算法 2 相同，因此算法 3 生成的测试用例集同样保证覆盖所有需要的组合. 可见该测试用例集是满足覆盖要求的最优测试用例集. 证毕.

6 覆盖率分析及算法的改进

相邻因素 N 维组合覆盖表可以保证覆盖所有的相邻因素 N 维组合，即覆盖率 $p=1$. 但是由于上述算法生成的覆盖表中存在大量冗余组合，在对 SUT 的测试发现故障后进行故障定位时，所提供的信息量太少. 因此需要对算法进行改进，使得相邻因素组合覆盖表能在不扩大覆盖表规模的前提下尽量提高对任意因素间组合的覆盖率.

例如，当 $n=4$，$a[1]=a[2]=a[3]=a[4]=3$ 时，表 2 中的相邻因素两两组合覆盖表 $NCA(9;2,3^4)$ 对任意因素两两组合的覆盖率仅为 $p=42/54=7/9$. 如果在测试时，参数 $a=2$ 导致第 4，5，6 号测试用例 (2,1,2,1)，(2,2,2,2)，(2,3,2,3) 出错，由于参数 a 和 c 的取值完全相同，我们无法判断故障的诱发因素是 a 还是 c. 而根据正交实验设计的理论，此时可以生成表 3 所示的正交表 $OA(9;2,3^4)$，在规模相同的情况下不仅覆盖了所有的相邻因素两两组合，对任意因素两两组合的覆盖率也达到了 $p=1$，

对于上述的故障则可以将故障诱因定位到 $a=2$. 通过观察表 2 与表 3 间的区别，可以看出产生冗余组合的一个重要原因是扩展参数时没有考虑与非相邻参数的组合情况. 如 $NCA(9;2,3^4)$ 在扩展参数 c 时只考虑了与参数 b 的组合，对于参数 a 与 c 则只覆盖了 9 个可能组合中的 3 个，分别为(1,1)，(2,2)，(3,3). 而相同规模的正交表则在覆盖 b 与 c 之间所有组合的同时也覆盖了 a 与 c 之间所有的组合. 一般的，当正交表存在时，使用正交实验设计法生成测试用例即可.

表 2　$NCA(9;2,3^4)$

a	b	c	d
1	1	1	1
1	2	1	2
1	3	1	3
2	1	2	1
2	2	2	2
2	3	2	3
3	1	3	1
3	2	3	2
3	3	3	3

表 3　$OA(9;2,3^4)$

a	b	c	d
1	1	1	1
1	2	2	2
1	3	3	3
2	1	2	3
2	2	3	1
2	3	1	2
3	1	3	2
3	2	1	3
3	3	2	1

下面考虑对算法 2 进行改进，以提高任意因素间组合的覆盖率. 方法如下：在扩展参数 f_i 之前，将参数 $f_{i-N+1},\cdots,f_{i-1}$ 的 $\prod_{k=i-N+1}^{i-1} a[k]$ 个 $N-1$ 维组合依次编号为 $1,2,\cdots,\prod_{k=i-N+1}^{i-1} a[k]$，若 $N=2$，则参数 f_{i-1} 的 $a[i-1]$ 个值的编号就是该值本身. 扩展参数时，在每一个 $N-1$ 维组合的前 $a[i]$ 次出现后填入 $v=1+(count+number-2) \bmod a[i]$，其中 $count$ 为该组合当前的出现次数，$number$ 为该组合的编号. 改进后的算法称为算法 2′.

定理 5.　当 $N=2$，且 $a[1]=a[2]=\cdots=a[n]=p$，$n\leq p+1$（p 为素数）时，算法 2′ 生成的相邻因素两两组合覆盖表为正交表 $OA(p^2;2,p^n)$.

证明.　由正交实验设计的理论可知，当 $a[1]=a[2]=\cdots=a[n]=p$，$n\leq p+1$（p 为素数）时，存在 $p-1$ 个 p 阶的正交拉丁方 $C_k(k=1,2,\cdots,p-1)$ 组成正交拉丁方完全组，其中的加法以 p 为模. 令 $C_{k,i}$ $(i=1,2,\cdots,p)$ 表示 C_k 的第 i 列组成的子阵，则该正交拉丁方完全组与基本列组成正交表 $OA(p^2;2,p^n)$，如图 3 所示.

$$C_k=\begin{pmatrix} 1 & 2 & \cdots & p \\ 1+k & 2+k & \cdots & p+k \\ 1+2k & 2+2k & \cdots & p+2k \\ \vdots & \vdots & \vdots & \vdots \\ 1+(p-1)k & 2+(p-1)k & \cdots & p+(p-1)k \end{pmatrix}$$

$$k=1,2,\cdots,p-1$$

$$\begin{pmatrix} 1 & 1 & & & & \\ 1 & 2 & C_{1,1} & C_{2,1} & \cdots & C_{p-1,1} \\ \vdots & \vdots & & & & \\ 1 & p & & & & \\ 2 & 1 & & & & \\ 2 & 2 & C_{1,2} & C_{2,2} & \cdots & C_{p-1,2} \\ \vdots & \vdots & & & & \\ 2 & p & & & & \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ p & 1 & & & & \\ p & 2 & C_{1,p} & C_{2,p} & \cdots & C_{p-1,p} \\ \vdots & \vdots & & & & \\ p & p & & & & \end{pmatrix}$$

图 3　正交拉丁方与正交表

当 $N=2$ 时，算法 2′ 在处理第 1 列和第 2 列时形成两个基本列. 扩展参数 f_3 时形成的列为 $(1,2,\cdots,p-1,p;2,3,\cdots,p,1;\cdots;p-1,p,\cdots,p-3,p-2;p,1,\cdots,p-2,p-1)^{\mathrm{T}}$，与正交拉丁方 C_1 按列展开后形成的列 $(C_{1,1}{}^{\mathrm{T}},C_{1,2}{}^{\mathrm{T}},\cdots,C_{1,p}{}^{\mathrm{T}})^{\mathrm{T}}$ 相同. 又由于两个相邻的正交拉丁方中相同位置的元素间存在关系：$C_{k+1}[i,j]=(C_k[i,j]+(i-1)) \bmod p(i=1,2,\cdots,p-1;\ j=1,2,\cdots,p)$，如果用 p 代替 0 则有：$C_{k+1}[i,j]=1+(C_k[i,j]+i-2)) \bmod p(i=1,2,\cdots,p-1;\ j=1,2,\cdots,p)$，其中 i 正好是 $C_k[i,j]$ 在正交拉丁方 C_k 中当前的出现次数. 由此可见，算法 2 在扩展参数 $f_i(i=3,4,\cdots,n)$ 时形成的列正是正交拉丁方 C_{i-2} 按列展开形成的列，即算法 2′ 形成正交表 $OA(p^2;\ 2,p^n)$.　　证毕.

使用算法 2′ 可以生成表 4 所示的相邻因素两两组合覆盖表 $NCA'(9;2,3^4)$，该表与正交表 $OA(9;2,3^4)$ 完全相同. 我们还在表 7 中比较了算法 2 与算法 2′ 生成的相邻因素两两组合覆盖表（存在未赋值

位置)对任意因素两两组合的覆盖率 $p(2)$ 和 $p'(2)$ 以及对任意因素三维组合的覆盖率 $p(3)$ 和 $p'(3)$,可以看出使用算法 2′后覆盖率得到明显提高.

表 4 $NCA'(9;2,3^4)$

a	b	c	d
1	1	1	1
1	2	2	2
1	3	3	3
2	1	2	3
2	2	3	1
2	3	1	2
3	1	3	2
3	2	1	3
3	3	2	1

表 5 $NCA(9;2,F)$

a	b	c	d
1	1	1	1
1	2	2	2
1	3	3	1
2	1	2	1
2	2	3	2
2	3	1	2
-	1	3	-
-	2	1	-
-	3	2	-

表 6 $NCA'(9;2,F)$

a	b	c	d
1	1	1	1
1	2	2	2
1	3	3	1
2	1	2	1
2	2	3	2
2	3	1	3
1	1	3	2
1	2	1	1
2	3	2	1

表 7 算法 2 与算法 2′生成的相邻因素两两组合覆盖表的覆盖率比较

F	$p(2)$	$p(3)$	$p'(2)$	$p'(3)$
3^4	0.77778	0.33333	1.00000	0.33333
3^{13}	0.69231	0.29060	0.84615	0.32401
4^{10}	0.66667	0.21875	0.78889	0.23490
2^{100}	0.75253	0.43939	0.75748	0.44121
10^{20}	0.57368	0.08105	0.69210	0.08986
$2\times3^2\times2$	0.78378	0.33333	0.94595	0.40000
$4^{15}\times3^{17}\times2^{29}$	0.53736	0.19792	0.62840	0.21742
$3^3\times4\times5\times3\times2^3$	0.59063	0.19808	0.71875	0.21866

提高任意因素间组合覆盖率的另一种可行方法是利用可能存在的未赋值位置,通过贪心算法选择适当的值填入这样的位置,使得相邻因素组合覆盖表对任意因素间组合的覆盖率尽可能高.例如当 $F=[2\times3^2\times2]$ 时,利用算法 2′生成表 5 所示的相邻因素两两组合覆盖表 $NCA(9;2,F)$,其中"-"表示未赋值位置,该表对任意因素两两组合的覆盖率为 $p=35/37$.填充表中的未赋值位置后得到表 6 所示的 $NCA'(9;2,F)$,对任意因素两两组合的覆盖率提高到 $p'=1$,而对任意因素三维组合的覆盖率也由 $p=24/60$ 提高到 $p'=34/60$.

7 应用实例分析

在基于输入输出的黑盒测试中,需要检测软件中所有输入对输出的影响,以判断系统是否满足需求.用 X 表示所有输入变量的集合,Y 表示所有输出变量的集合;用 $D(x_i)$ 表示输入变量 x_i 的取值的集合,T 表示输入变量的所有组合:$T=D(x_1)\times D(x_2)\times\cdots\times D(x_n)$.在一个程序中,一般不可能每个输出都受到所有输入变量的影响,而只是受到部分输入变量的影响.用 $X(y)$ 表示与输出变量 y 有关的输入变量的集合,用 $T(y)$ 表示只与输出变量 y 有关的输入变量的值组合,在这个集合中包含 $X(y)$ 中所有变量的值组合,而除 $X(y)$ 以外的变量的值任意取定.$T(Y)$ 表示与所有输出变量 y 有关的值组合的并集:$T(Y)=\bigcup_{y\in Y}T(y)$,显然,$T(Y)\subseteq T$.

例如,中国铁路一般采用自动闭塞信号系统(ABSS)来保证行车安全,具体的方法是在车站之间的线路上每隔一定距离设置一个自动闭塞信号机,两个信号机之间的线路称为一个闭塞分区,每个闭塞分区最多只允许被一组列车占用,四显示自动闭塞信号机根据前方 3 个相邻分区的当前状态(分区内线路是否正常,分区是否被列车占用)显示 4 种信号中的 1 种,司机根据信号机的显示在安全的范围内控制列车速度[28].在对一个控制四显示自动闭塞信号系统的软件系统进行基于功能的测试时,可以认为每一个信号机的显示状态(输出)只与该信号机前方 3 个相邻闭塞分区的 4 种可能状态(3 个相邻的输入参数,每个参数取值个数为 4)有关.

假设一段线路由 $n(n>3)$ 个闭塞分区组成,则对该段线路信号系统的输入进行全覆盖所需要的测试用例数量为 $N_T=D(x_1)\times D(x_2)\times\cdots\times D(x_n)=4^n$,数量过大且没有必要.由于对于大多数 n 的取值不存在相应的 N 维正交表,故使用正交实验设计法是不可取的.如果使用一般的组合测试方法,则所需测试用例的数量依赖于线路中闭塞分区的数量 n,表 8 给出了使用文献[22]中提出的 GA-N 方法生成的 3 维组合测试用例集的规模.而实际上,我们只

需从 T 中选取一组测试用例 $T(Y)$ 保证对所有 $T(y)$ 进行覆盖即可，即只需对影响各输出变量的输入变量进行组合覆盖[6]. 该系统的特点是每个输出 y_i 只受 3 个相邻的输入变量 x_j，x_{j+1} 和 x_{j+2} 影响，即 $T(y_i)=\{x_j,x_{j+1},x_{j+2}\}(j=1,2,\cdots,n-2)$. 因此采用相邻因素组合测试的方法就可以在不降低检测能力的基础上，仅用 64 条测试用例对系统进行有效的测试(如图 4 所示).

表 8　3 维组合测试生成的测试用例的数量

n	测试用例的数量
5	64
6	64
7	68
8	87
9	114
10	138
11	157
12	175
13	192

分区 1	分区 2	分区 3	分区 4	分区 5	分区 6	…
线路正常 & 占用	线路异常 & 占用	线路正常 & 空闲	线路异常 & 空闲	线路正常 & 空闲	线路正常 & 占用	…
线路正常 & 占用	线路正常 & 空闲	线路异常 & 空闲	线路异常 & 占用	线路正常 & 占用	线路异常 & 占用	…
⋮	⋮	⋮	⋮	⋮	⋮	⋱

图 4　ABSS 系统的测试用例

下面我们分析相邻因素组合测试的错误检测能力. 假设上述系统有 10 个输入参数，利用算法 2 生成规模为 64 的相邻因素 3 维组合测试用例集，对相邻因素 3 维组合的覆盖率 $p=1$，即该测试用例集可以检测所有相邻 3 因素交互导致的故障. 如果使用随机算法生成 10 组规模相同的随机测试用例集，它们对相邻因素 3 维组合的平均覆盖率 $p_{\text{average}}=0.646652$，如表 9 所示. 设事件 E：系统中存在一个相邻 3 因素错误；事件 H：一个测试用例集可以检测出该错误. 则相邻因素 3 维组合测试用例集的条件概率 $P(H|E)=1$，而这 10 组随机测试用例集的平均条件概率 $P_{\text{average}}(H|E)=0.646652<1$. 可见对于上述应用场景，相邻因素组合测试用例集的检测能力明显高于相同规模的随机测试用例集.

表 9　随机测试用例集对相邻因素 3 维组合的覆盖率

	p
1	0.65848
2	0.62053
3	0.64732
4	0.63616
5	0.62277
6	0.66295
7	0.67411
8	0.64286
9	0.64509
10	0.65625

为了提高测试效率，为未来的研究工作提供更多便利，我们开发了一套组合测试用例的自动生成工具，它实现了本文给出的算法 2、算法 2′和算法 3，相关工作中所提及的大部分组合测试用例集生成算法也作为其中的一个重要组成部分而加以实现. 使用该工具可以很方便地生成各种应用所需的(相邻因素)组合测试用例集，进而对其进行分析和应用.

8　结论及进一步工作

本文系统地研究了相邻因素组合测试，给出了相邻因素 $N(N\geq 2)$ 维组合覆盖表和多重维数相邻因素组合覆盖表的生成算法，证明了这些算法可以生成最优的相邻因素组合测试用例集，并分析了相邻因素组合测试的应用价值. 作为软件测试的一种重要方法，组合测试在软件测试的实践中有着重要的应用价值. 对于只在相邻因素间存在相互作用的软件系统，相邻因素组合测试作为组合测试的一种特例，可以保证生成数量最优的测试用例集而不降低错误检测能力. 但对于不具备这样特征的系统，相邻因素组合测试用例集无法适用，仍然需要使用传统的组合测试方法来进行测试.

关于组合测试的研究尽管已经取得了丰富的成果，但还存在很多问题：首先由于 $N(N\geq 2)$ 维组合覆盖表以及多重维数组合覆盖表的构造问题是一个 NP-C 问题，现有算法均为近似算法，算法的性能往往无法满足需求；其次，现有算法多针对两两组合覆盖，对 $N(N>2)$ 维组合覆盖以及多重维数组合覆盖的研究相对较少，而在很多情况下为了提高软件测试的可靠性，有必要采取 $N(N>2)$ 维和多重维数的组合测试；此外，基于组合测试的软件故障诊断和软件调试技术还需要进一步地研究.

针对以上问题，我们将在以后的研究中从以下几个方面继续深入：首先是研究更高性能的组合测试用例生成算法，针对各种不同的需求尽量减小组合覆盖表的规模；其次是对基于组合测试的故障诊

断和软件调试技术进行深入研究;并通过实验对组合测试的适用性以及可靠性进行分析;最后是对组合测试自动工具的研制,来实现测试用例生成、测试执行、测试度量、测试结果分析以及故障定位等一系列过程的自动化.

参 考 文 献

[1] Mandl R. Orthogonal latin squares: An application of experimental design to compiler testing. Communications of the ACM, 1985, 28(10): 1054-1058

[2] Brownlie R, Prowse J, Phadke M. Robust testing of AT&T PMX/StarMail using OATS. AT&T Technical Journal, 1992, 3(71): 41-47

[3] Cohen D M, Dalal S R, Fredman M L et al. The AETG system: An approach to testing based on combinatorial design. IEEE Transactions on Software Engineering, 1997, 23(7): 437-444

[4] Cohen D M, Dalal S R, Parelius J et al. The combinatorial design approach to automatic test generation. IEEE Software, 1996, 13(5): 83-87

[5] Cohen D M, Dalal S R, Kajla A et al. The automatic efficient tests generator//Proceedings of the IEEE 5th International Symposium on Software Reliability Engineering. Monterey, California, 1994: 303-309

[6] Schoeder P J. Black-box test reduction using input-output analysis[Ph. D. dissertation]. Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA, 2001

[7] Salem A M. A software testing model: Using design of experiments(DOE) and logistic regression [Ph. D. dissertation]. Florida Institute of Technology, Melbourne, Florida, USA, 2001

[8] Lei Y, Tai K C. In_Parameter_Oder: A test generation strategy for pairwise testing. Department of Computer Science, North Carolina State University, Raleigh, North Carolina, USA: Technical Report: TR-2001-03, 2001

[9] Tai K C, Lei Y. A test generation strategy for pairwise testing. IEEE Transactions on Software Engineering, 2002, 28(1): 109-111

[10] Kuhn D R, Reilly M J. An investigation of the applicability of design of experiments to software testing//Proceedings of the 27th NASA/IEEE Software Engineering Workshop. NASA Goddard Space Flight Center, 2002: 91-95

[11] Kobayashi N, Tsuchiya T, Kikuno T. A new method for constructing pair-wise covering designs for software testing. Information Processing Letters, 2002, 81(2): 85-91

[12] Williams A W. Software component interaction testing: Coverage measurement and generation of configurations[Ph. D. dissertation]. Ottawa-Carleton Institute for Computer Science, School of Information Technology and Engineering, University of Ottawa, Canada, 2002

[13] Cohen M B, Colbouns C J, Collofello J S et al. Variable strength interaction testing of components//Proceedings of the 27th International Computer Software and Applications Conference(COMPSAC2003). Dallas TX, 2003: 413-418

[14] Cohen M B, Colbouns C J, Ling A C H. Augmenting simulated annealing to build interaction test suites//Proceedings of the 14th International Symposium on Software Reliability Engineering (ISSRE 2003). Denver Colorado, 2003: 394-405

[15] Cohen M B, Colbouns C J, Gibbons P B et al. Constructing test suites for interaction testing//Proceedings of the International Conference on Software Engineering (ICSE2003). Portland OR, 2003: 38-48

[16] Kuhn D R, Wallace D R. Software fault interaction and implication for software testing. IEEE Transactions on Software Engineering, 2004, 30(6): 1-4

[17] Shiba T, Tsuchiya T, Kikuno T. Using artificial life techniques to generate test cases for combinatorial testing//Proceedings of the 28th International Computer Software and Applications Conference (COMPSAC2004). Hong Kong, China, 2004: 72-78

[18] Colbourn C J, Cohen M B, Turban R C. A deterministic density algorithm for pairwise interaction coverage//Proceedings of the IASTED International Conference on Software Engineering (SE2004). Innsbruck, Austria, 2004: 345-352

[19] Schroeder P J, Bolaki P, Gopu V. Comparing the fault detection effectiveness of n-way and random test suite//Proceedings of the 2004 International Symposium on Empirical Software Engineering (ISESE2004). Redondo Beach, California, 2004: 49-59

[20] Colbourn C J, Martirosyan S S, Mullen G L et al. Products of mixed covering arrays of strength two. Journal of Combinatorial Designs, 2005, 14(2): 124-138

[21] Sherwood G B, Martirosyan S S, Colbourn C J. Covering arrays of higher strength from permutation vectors. Journal of Combinatorial Designs, 2005, 14(3): 202-213

[22] Nie Chang-Hai, Xu Bao-Wen, Shi Liang, Dong Guo-Wei. Automatic test generation for n-way combinatorial testing//Proceedings of the 2nd International Workshop on Software Quality(SOQUA2005). Fair and Convention Center, Erfurt, Germany, 2005: 203-211

[23] Nie Chang-Hai, Xu Bao-Wen. A minimal test suite generation method. Chinese Journal of Computers, 2003, 26(12): 1690-1695(in Chinese)
(聂长海,徐宝文.一种最小测试用例集生成方法.计算机学报, 2003, 26(12): 1690-1695)

[24] Nie Chang-Hai, Xu Bao-Wen. An algorithm for automatically generating black-box test cases based interface parameters. Chinese Journal of Computers, 2004, 27(3): 382-388(in Chinese)
(聂长海,徐宝文.基于接口参数的黑箱测试用例自动生成算法.计算机学报, 2004, 27(3): 382-388)

[25] Xu Bao-Wen, Nie Chang-Hai, Shi Liang et al. A software failure debugging method based on combinatorial design approach for testing. Chinese Journal of Computers, 2006, 29(1): 132-138(in Chinese)
(徐宝文,聂长海,史　亮等.一种基于组合测试的软件故障调试方法.计算机学报,2006,29(1):132-138)

[26] Nie Chang-Hai, Xu Bao-Wen, Shi Liang. A new pairwise covering test data generation algorithm for the system with many 2-level factors. Chinese Journal of Computers, 2006, 29(6): 841-848(in Chinese)
(聂长海,徐宝文,史　亮.一种新的二水平多因素系统两两组合覆盖测试数据生成算法.计算机学报,2006,29(6):841-848)

[27] Shi Liang, Nie Chang-Hai, Xu Bao-Wen. Pairwise test data generation based on solution space tree. Chinese Journal of Computers, 2006, 29(6): 849-857(in Chinese)
(史　亮,聂长海,徐宝文.基于解空间树的组合测试数据生成.计算机学报,2006,29(6):849-857)

[28] China Ministry of Railway. Railway Technical Management Regulation(10th edition). Beijing: Chinese Railway Publishing House, 2006(in Chinese)
(中华人民共和国铁道部.铁路技术管理规程(第10版).北京:中国铁道出版社,2006)

WANG Zi-Yuan, born in 1982, Ph. D. candidate. His research interest is software testing.

NIE Chang-Hai, born in 1971, Ph. D., associate professor. His research interests include software engineering and software testing, fussy information processing and neutral network etc.

XU Bao-Wen, born in 1961, Ph. D., professor, Ph. D. supervisor. His research interests include teaching and research on programming language, software engineering, parallel and network software, acquisition technique on knowledge and information etc.

SHI Liang, born in 1979, Ph. D.. His research interests include software analysis and testing.

Background

This paper is supported by the National Natural Science Foundation of China (No. 60373066, 60403016), which title is the "Research on Software Testing Technology Based on Combinatorial Cover". The project mainly research on the following fields: Combinatorial coverage method and its effectiveness for software testing, the existence and the generation algorithm for several minimal combinatorial covering table, some techniques for software fault diagnosis and debugging based on combinatorial testing, the application of the combinatorial testing in web testing and software configuration testing. The authors have made some works on the test case generation algorithm and fault debugging method. This paper focuses on neighbor factors combinatorial testing, which is a special case of combinatorial testing.