基于部分覆盖表的错误交互定位方法

周吴杰[1),2)] 张德平[4)] 徐宝文[2),3)]

[1)](东南大学计算机科学与工程学院 南京 210096)

[2)](南京大学软件新技术国家重点实验室 南京 210093)

[3)](南京大学计算机科学与技术系 南京 210093)

[4)](南京航空航天大学信息科学与技术学院 南京 210016)

摘 要 在组合测试定位模型的基础上提出了部分覆盖表的错误交互定位方法,该方法在错误交互个数已知的条件下,通过生成部分覆盖表,利用测试用例运行结果提供的信息来对软件错误交互定位.从理论上证明了部分覆盖表等价于一类特殊的错误定位表,进而研究了部分覆盖表行数的上界,提出生成部分覆盖表的贪心算法,从而给出了定位引发软件故障的错误交互的非自适应算法,并通过实验验证了该方法的有效性.

关键词 组合测试;覆盖表;部分覆盖表;错误定位表

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2011.01126

Locating Error Interactions Based on Partial Covering Array

ZHOU Wu-Jie[1),2)] ZHANG De-Ping[4)] XU Bao-Wen[2),3)]

[1)](*Department of Computer Science & Engineering, Southeast University, Nanjing* 210096)

[2)](*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210093)

[3)](*Department of Computer Science and Technology, Nanjing University, Nanjing* 210093)

[4)](*College of Information Science and Technology, Nanjing University of Aeronautics & Astronautics, Nanjing* 210016)

Abstract Combinatorial testing is practical and effective method to detect the faults triggered by the interactions among parameters or components in the system. It is a key problem that how to locate the faulty interactions after some faults were detected in the test stages. Error locating arrays (ELAs) were defined by Martínez C et al for detecting and locating the faulty interactions in a system. In this paper, the authors prove that a special partial covering array (PCA) is also a special error locating array and study the upper bound of PCA's size. It is obtained that the number of tests given by PCAs is polynomial in log$k$ and d, where k is the number of parameters and d is the number of faulty interactions in the system. Furthermore the authors propose the greedy algorithms to generate the special PCAs, and these algorithms are also a non-adaptive algorithm of producing ELAs. The experimental results show the above approach for detecting and locating the faulty interactions using PCAs is effective.

Keywords combinatorial testing; covering array; partial covering array; error locating array

---

收稿日期:2010-11-16;最终修改稿收到日期:2011-05-17.本课题得到国家自然科学基金(90818027,91018005)、国家"八六三"高技术研究专题项目与发展计划项目基金(2009AA01Z147)、国家"九七三"重点基础研究发展规划项目基金(2009CB320703)资助.**周吴杰**,男,1973 年生,博士研究生,讲师,主要从事软件测试技术、软件可靠性等方面的科研工作.E-mail: zhouwujie@seu.edu.cn. **张德平**,男,1973 年生,博士,讲师,主要从事软件测试技术、软件可靠性、数理统计等方面的教学、科研工作. **徐宝文**(通信作者),男,1961 年生,博士,教授,博士生导师,中国计算机学会高级会员,主要研究领域为程序设计语言、软件工程、并行与网络软件.

1 引　言

随着计算机技术的飞速发展，软件系统变得越来越庞大，软件测试成为保证这些系统质量的一个关键手段.然而影响系统运行的因素有很多，如何检测各个因素之间的相互作用对系统产生的影响，通常可通过组合测试方法来检测.组合测试作为一种重要的软件测试方法，因能以较少的测试用例来实现对被测试系统进行科学有效的测试而得到广泛的研究和应用.对于那些故障来源于软件系统中一些参数和参数之间的相互作用的系统，这种方法的效果尤其明显.一般地，对于小的软件系统，其因素数很少时，可采取穷尽测试来检测各个因素交互作用，但对于大的软件系统，穷尽测试是不可能的，也是没有必要的.如一个系统有 10 个因素，每个有 4 个不同的取值，则穷尽测试需要 $4^{10}=1048576$ 个测试用例.所以我们可以选择测试用例来测试所有因素的两两交互，或者 t 维交互，这样需运行的测试用例会大大减少.如上例，若采用两两交互测试只需至多 25 个测试用例.Kuhn 等人的研究表明测试两两组合覆盖能发现大约 70%的错误，三维组合覆盖能发现 90%的错误[1].

在组合测试中，由于测试的失败预示系统组件中存在错误，这就需要测试人员找出触发系统故障的错误交互.目前现存的基于组合测试的错误定位方法主要包括分类树法[2]、故障调试法[3]和错误定位表[4-6]等.利用分类树法，系统的错误模式(导致系统故障的是某些参数的特定取值或是这些取值的组合)一般很难被精确地确定[7].采用故障调试定位法通过增加附加测试用例数能够对一些特殊的系统错误得到较为精确的定位，但对于一些大型软件系统，由于系统的交互因素数目众多，一个测试用例包含的模式数目呈指数增长，从而导致触发软件故障的可能模式数目呈指数增长，使得错误定位变得不适用[3].在某些特殊假设下，错误定位表虽然能够对所有软件交互错误都能精确定位，但错误定位表的行数随系统中错误交互数增加呈指数增长，并且对于较为大型软件系统其生成也是一项相当复杂且费时的任务，在实际组合测试中进行错误定位显得低效.因此，当利用组合测试策略来对系统进行测试，发现某些因素的组合导致系统发生故障时，如何定位和侦测哪些因素组合会导致系统故障，一直是组合测试中一个亟待解决的重要课题.

本文在研究 Martínez 等人[5-6]提出的组合测试错误定位表的基础上，提出了一类特殊的错误定位表——部分覆盖表，在错误交互个数与安全值向量已知的条件下，通过生成部分覆盖表，利用测试用例运行结果提供的信息来对软件错误定位.并从理论上证明了部分覆盖表等价于这类特殊的错误定位表，进而研究了部分覆盖表行数的上界，并提出生成部分覆盖表的贪心算法，给出了定位引发软件故障的错误交互的非自适应算法.本文第 2 节介绍背景及相关研究工作；给出了基本的组合测试模型、故障定位模型以及相关结论；第 3 节提出基于部分覆盖表的错误定位模型，证明了在某种情形下，错误定位表等价于某种类型的部分覆盖表，并用概率方法证明了部分覆盖表的行数的上界；第 4 节给出基于逐条生成测试用例的部分覆盖表的贪心算法，估计出利用贪心算法来生成此特殊的部分覆盖表时覆盖表的规模，并进一步给出了局部模拟退火算法；第 5 节通过实例分析对部分覆盖表生成方法的有效性进行验证；最后是全文总结并讨论未来可进行研究的方向.

2 背景及相关研究

2.1 相关研究

目前，人们关于组合测试的研究主要集中在组合覆盖测试用例集的生成技术，即如何在满足给定组合覆盖标准的前提下，生成规模尽可能小的测试用例集以节约测试成本.组合测试用例集生成一般是利用传统的代数方法[8-9]求解或者最优化方法来直接搜索或构造覆盖数组.由于这个问题的复杂性是 NP 完全的，大部分的方法都是局部搜索算法，这些方法不能保证得到最优解，但是处理时间相对较少.这些方法主要包括贪心算法和启发式搜索方法[10-15].另外，有少量的研究采用全局搜索算法，如一些元启发式搜索方法[16-19]，能够对一定规模的问题得到较优解.对于生成测试用例规模问题的确定，即对于具体的应用场景，至少生成多少条测试用例才能满足组合覆盖需求，这个问题只有在某些特殊的情况下才能确定，对于一般情形这仍然是个公开问题；Godbole 等人[20]采用概率方法来研究生成测试用例规模的上界，Carey 等人[21]提出部分覆盖表的概念并研究其行数(测试用例数)的上界.

当软件缺陷被检测出后，如何利用运行测试用例所得的信息来进行错误定位是软件测试过程中必不可少的一个环节.目前，对组合测试的结果进行调试和分析的研究还很少.Yilmaz 等人[2]采用分类树

(classification tree)的方法分析程序错误的特征(fault characterization),触发被测系统错误的参数取值.该项工作主要分析了与系统配置相关的错误(option-related failures),他们采用一个开源中间件ACE+TAO分析了采用覆盖数组的组合测试的错误定位能力.其实验结果表明,如果采用分类树算法对测试结果进行分析,那么,即使是强度较小的组合测试方法,其错误定位能力也与完备测试相当,但采用组合测试无法进一步确定错误发生的具体原因[7].徐宝文等人[3]首先假定"若某个模式是错误模式,则任意包含该模式的其它模式也将引发相同的错误".在此前提条件下,他们通过分析执行失败的测试用例,生成一批与之类似的附加测试用例,并通过执行附加测试用例,逐步缩小故障模式集合M的范围,直到错误被精确地定位.Colbourn和McClary提出了(d,t)错误定位表及错误侦测表的概念,用这些表来对组合交互错误进行定位[4].随后Martínez等人[5-6]提出了一般的错误定位表,并在此模型下提出了自适应算法来定位错误交互.

2.2 错误定位表及相关定义

为了更好介绍基于组合测试的软件故障定位与侦测技术,这里先给出一些记号和形式化定义.

假设影响待测系统(Software Under Test,SUT)的因素共有k个,因素i有$v_i(1\leq i\leq k)$个可能的取值,用$0,1,\cdots,v_i-1$表示.我们用记号$[0,v_i-1]$表示集合$\{0,1,\cdots,v_i-1\}$.假设这些参数的取值是相互独立的,即某个参数的具体取值不会影响其它参数的取值或存在性.

定义 1. 设k维向量$\boldsymbol{T}=(T_1,T_2,\cdots,T_k)$,其中$T_i\in[0,v_i-1]$,$i=1,2,\cdots,k$,则称这个$k$维向量$\boldsymbol{T}$为第$i$个因素取值为$T_i$的测试用例.

为了定位交互,引入如下定义.

定义 2. 任取t个因素,设集合$I=\{(i_1,a_{i_1}),(i_2,a_{i_2}),\cdots,(i_t,a_{i_t})\}$,其中因素$i_j$互不相等,$a_{i_j}\in\{0,1,\cdots,v_{i_j}-1\}(i=1,2,\cdots,t)$,则称这个集合$I$为一个$t$维交互.称集合$f=\{i_1,i_2,\cdots,i_t\}$为交互$I$对应的因素集,$(i_1,a_{i_1})$称为顶点.

定义 3. 设一条测试用例$\boldsymbol{T}$的第i_j个因素取值是$a_{i_j}(j=1,2,\cdots,t)$,即$\boldsymbol{T}(i_j)=a_{i_j}$,则称这条测试用例覆盖了$t$维交互$I=\{(i_1,a_{i_1}),(i_2,a_{i_2}),\cdots,(i_t,a_{i_t})\}$.

由定义3可知,一条测试用例覆盖了$\binom{k}{t}$个t维交互.为了产生交互测试用例,定义覆盖表如下.

定义 4. 如果A是一个$n\times k$表,表中第i列元素都取自$[0,v_i-1]$,且满足每个可能的t维交互都被表中某一行所对应的测试用例所覆盖,即对任意的$\{(i_1,a_{i_1}),(i_2,a_{i_2}),\cdots,(i_t,a_{i_t})\}$,至少存在一行$r$,使得$A[r,i_j]=a_{i_j}$,$j=1,2,\cdots,t$,则称表$A$是一个$t$维混合覆盖表,记此表为$MCA(n;t,(v_1,v_2,\cdots,v_k))$.$t$称为覆盖表的强度.给定$t$和$v_1,v_2,\cdots,v_k$,称使得$MCA(n;t,(v_1,v_2,\cdots,v_k))$存在的最小的整数$n$为混合覆盖表数,记为$MCAN(t,(v_1,v_2,\cdots,v_k))$,当定义中的$v_1=v_2=\cdots=v_k=v$时,我们简记为$CA(n;t,k,v)$和$CAN(t,k,v)$.

在不引起混淆的情况下覆盖表或混合覆盖表统称为覆盖表.

一般地,人们用覆盖表或混合覆盖表来产生测试用例,表中每一列对应一个因素,k表示因素数目,v_i表示每个因素可能取的取值个数,每一行表示一个测试用例,t维覆盖表产生的测试用例集能覆盖到k个因素中任意t个因素所有可能的取值组合,2维覆盖表产生的测试用例集称为两两组合测试用例集,人们希望在不降低测试标准情况下来产生尽可能少的测试用例.

例如一个打印系统有两种类型打印机P_1,P_2,分别用0,1表示;打印的文件格式有3种JPEG,PDF,PS,分别用0,1,2表示,颜色及文件大小如表1所示.

表 1 打印机系统

打印机	文件格式	颜色	文件大小
$0=P_1$	0=JPEG	0=black & white	0=(≤50)
$1=P_2$	1=PDF	1=colour	1=(>50 and <500)
	2=PS		2=(≥500)

若对因素的所有可能取值组合进行测试,则需要$2\times3\times2\times3=36$个测试用例,若只考虑任意两个因素之间的交互作用,则只需要9条测试用例即可,如表2.

表 2 打印系统的两两组合测试用例集

测试用例	打印机	文件格式	颜色	文件大小	输出结果
1	0	0	0	0	pass
2	0	0	1	1	fail
3	1	0	1	2	fail
4	1	1	1	0	pass
5	1	1	0	1	pass
6	0	1	0	2	fail
7	0	2	0	0	fail
8	0	2	1	1	pass
9	1	2	1	2	pass

假设每条测试用例在运行时只有两个可能结果:通过或失败.假定一个运行失败的测试用例至少

包含一个交互错误，否则就运行通过. 进一步，假定若一个交互是错误的，则所有包含此错误交互的测试用例运行都是失败的. 运行表 2 中的测试用例，得到输出结果列在表的最后一列.

对于那些失败的测试用例，如何才能精确定位是哪个取值组合出了问题？因此，必须对覆盖表进行更深入的研究，最好在测试完之后就能知道哪些交互是错误交互，这在没有额外的假设下很难做到. 基于此 Colbourn 和 McClary[4] 提出了 (d,t) 错误定位表及错误侦测表的概念，随后 Martínez 等人[5-6] 提出了一般的错误定位表概念. 他们提出的错误定位模型是把待测系统(SUT)中的错误交互看成超图的边，从而构成一个错误交互超图来进行定位. 为了简化起见，这里采用交互集合的语言来介绍错误定位表模型.

假设某个 s 维交互导致错误，则所有包含这个 s 维交互的其它交互也会导致错误，为此只记录极小的错误交互，即其任何的真子集都不再是错误交互. 设某个待测系统(SUT)有 d 个极小错误交互，用 Π 表示所有的极小错误交互构成的集合，则 Π 中错误交互都不可能互相包含. 如果 Π 中每个错误交互都有 t 个元素，记为 Π_t，如果每个错误交互至多有 t 个元素，记为 Π_t. 以后若不特别说明，错误交互集都是指极小的错误交互集. 如果一条测试用例没有覆盖 Π 中任何错误交互，则称这条测试用例避开了 Π.

定义 5.　给定 t 维交互 I 与错误交互集 Π_t，如果存在覆盖这个交互 I 的一条测试用例 T 避开了 $\Pi_t\backslash\{I\}$，则称这个 t 维交互 I 关于 Π_t 是可定位的，并称此测试用例 T 定位了交互 I. 如果每个 t 维交互关于 Π_t 都是可定位的，则称 Π_t 是可定位的.

类似地，可以定义：

定义 6.　给定 $s(s\leq t)$ 维交互 I 与错误交互集 Π_t，设 I 包含的 Π_t 中的极小错误交互集合为 Γ_I，如果存在覆盖这个交互 I 的一条测试用例 T 避开了 $\Pi_t\backslash\Gamma_I$，则称这个 s 维交互 I 关于 Π_t 是可定位的，并称此测试用例 T 定位了交互 I. 如果每个 $s(s\leq t)$ 维交互关于 Π_t 都是可定位的，则称 Π_t 是可定位的.

由定义 5 与定义 6 知，我们的组合测试方法只能定位可定位的错误交互集 Π，如一个待测系统(SUT)中有 3 个因素，每个因素取值都是二元的，如图 1 所示，类型 1 中交互集{{(2,0), (3,1)}{(2,1}, (3,1)}}就是不可定位的，因为交互{(1,0), (3,1)}不可定位；类型 2 中交互集{{(2,0),(3,1)}{(2,1}, (3,1)}}也是不可定位的，因为交互{(1,0), (3,1)}不可定位.

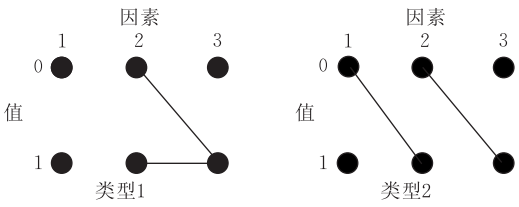

图 1　两种不同类型的错误交互

对于错误交互集 Π 是否可定位，判断起来比较复杂，如果对于任意一个因素 i，都至少存在一个顶点 (i,s_i)，错误交互集 Π 没有覆盖这个顶点，则错误交互集 Π 是可定位的.

定义 7[6].　如果一个待测系统(SUT)中的错误交互集为 Π，且对 $\forall i\in[1,k]$，都存在顶点 (i,s_i)，使得 (i,s_i) 没有被包含在任何交互 $I\in\Pi$ 中，则称这个待测系统(SUT)具有安全值，并称 $(s_1,s_2,\cdots,s_k)$ 为这个待测系统的安全值向量.

如果一个待测系统(SUT)具有安全值，则其错误交互集 Π_t 是可定位的[5].

对于可定位的错误交互集 Π，可定义错误定位表.

定义 8[5] (错误定位表 $ELA(n;t,(v_1,v_2,\cdots,v_k))$).　设待测系统(SUT)中，其错误交互集 Π_t 是可定位的，一个 $n\times k$ 表 A，其第 i 列元素取自 $[0,v_i-1](i=1,2,\cdots,k)$，且满足：每个 t 维交互 I 都能被 A 的某一行对应的测试用例所定位. 则称 A 是强度为 t 的错误定位表，记为 $ELA(n;t,(v_1,v_2,\cdots,v_k))$.

定义 9[5] (错误定位表 $ELA(n;\bar{t},(v_1,v_2,\cdots,v_k))$).　设待测系统(SUT)中，其错误交互集 Π_t 是可定位的，一个 $n\times k$ 表 A，其第 i 列元素取自 $[0,v_i-1](i=1,2,\cdots,k)$，且满足：每个不包含 Π_t 中任何 $s(s\leq t)$ 维交互 I 都能被 A 的某一行所对应的测试用例所定位，则称 A 是强度至多为 t 的错误定位表，记为 $ELA(n;\bar{t},(v_1,v_2,\cdots,v_k))$.

显然，每个错误定位表 $ELA(n;t,(v_1,v_2,\cdots,v_k))$ ($ELA(n;\bar{t},(v_1,v_2,\cdots,v_k))$) 都是 $MCA(n;t,k,(v_1,v_2,\cdots,v_k))$.

Martínez 等人证明了当 Π 中错误交互数是 k 的高阶无穷小时，绝大多数待测系统(SUT)都有安全值[6]. 利用错误定位表 $ELA(n;\bar{t},(v_1,v_2,\cdots,v_k))$，我们很容易确定错误交互集 Π_t. 运行错误定位表所形成的测试用例集，对所有通过的测试用例其包含的所有交互都是正确的，只需从所有的 $s(s\leq t)$ 维交

互形成的集合中划去包含在某个通过的测试用例中的那些交互,剩下的就是所寻找的错误交互集,再对这个集合中划去所有的非极小错误交互,剩下就是所求的 Π_t,具体过程由算法 1 给出.

算法 1. 由错误定位表得到错误交互集.

输入:$ELA(n;\bar{t},(v_1,v_2,\cdots,v_k))$

输出:Π_t

```
begin
  Π_t = ∅   //初始化 Π_t 为空集
  运行 ELA(n;t̄,(v_1,v_2,…,v_k)) 中每一行所构成的测试用例;
  for s=1 to t do
     Γ={(i_1,a_{i_1}),(i_2,a_{i_2}),…,(i_s,a_{i_s}) | i_j∈[1,k],
        a_{i_j}∈[0,v_{i_j}-1] }
                          //初始化 Γ 为 s 维交互的全体
     for i=1 to n do
        if ELA(n;t̄,(v_1,v_2,…,v_k)) 的第 i 行测试
           用例是通过的测试用例
           Γ=Γ\{被 i 行覆盖的交互}
        endif
     endfor
     Π_t = Π_t ∪ Γ
  endfor
     Π_t = Π_t \{非极小错误交互}
     return Π_t
end
```

对于可定位的 Π,错误定位表大小规模有多大?即最多有多少行? Martínez 等人用更高强度的覆盖表来构造错误定位表. 假设待测系统(SUT)具有安全值,其错误交互集 Π_t 中错误交互数最多为 d,则当 $t+d\leq k$ 时,每个 $MCA(n;t+d,(v_1,v_2,\cdots,v_k))$ 都是 $ELA(n;\bar{t},(v_1,v_2,\cdots,v_k))$[6].

对于覆盖表,可以估计其行数的上界,若待测系统(SUT)具有安全值,其错误交互集 Π_t 中错误交互数最多为 d,则其 $ELA(n;\bar{t},(v_1,v_2,\cdots,v_k))$ 的行数满足 $n=O(d(v)^d\log k)$[6]. 所以用 $MCA(n;t+d,(v_1,v_2,\cdots,v_k))$ 来构造错误定位表,其行数虽然关于因素数是对数阶的,但是关于错误数 d 则是指数阶的,当 d 比较大时,其行数会爆炸增长,并且由于需覆盖的交互数呈指数增长,所以构造混合覆盖表 $MCA(n;t+d,(v_1,v_2,\cdots,v_k))$ 的很多算法也变得不适用,下面我们在安全值已知的条件下寻找另外的组合结构来构造错误定位表.

3 基于部分覆盖表的错误定位模型

我们设待测系统(SUT)具有安全值,且安全值已知,设安全值向量 $\boldsymbol{s}=(s_1,s_2,\cdots,s_k)$,可以对每个因素的取值进行置换,所以不失一般性,我们设每个因素的安全值为 0,即安全值向量为 $\boldsymbol{s}=(0,0,\cdots,0)$. 另外,为了研究此类错误定位表构造,我们把 Carey 等人[21]提出部分覆盖表的概念一般化如下.

定义 10. 设 Γ 是一个 t 维交互集合,一个 $n\times k$ 表 A,如果表中第 i 列元素都取自 $[0,v_i-1]$,且满足:每个 Γ 中的 t 维交互都被表中某一行所对应的测试用例所覆盖,即对任意的 $\{(i_1,a_{i_1}),(i_2,a_{i_2}),\cdots,(i_t,a_{i_t})\}\in\Gamma$,至少存在一行 r,使得 $A[r,i_j]=a_{i_j}$,$j=1,2,\cdots,t$,则称表 A 是一个 t 维的部分混合覆盖表,记为 $PMCA(n;t,(v_1,v_2,\cdots,v_k),\Gamma)$. 给定 t 和 $v_1,v_2,\cdots,v_k,\Gamma$,称使得 $PMCA(n;t,(v_1,v_2,\cdots,v_k),\Gamma)$ 存在的最小的正整数 n 为部分混合覆盖表数,记为 $PMCAN(t,(v_1,v_2,\cdots,v_k),\Gamma)$,当定义中的 $v_1=v_2=\cdots=v_k=v$,我们简记为 $PCA(n;t,k,v,\Gamma)$ 和 $PCAN(t,k,v,\Gamma)$.

有了部分覆盖表的概念,我们得到下面的定理.

定理 1. 设待测系统(SUT)具有安全值,其安全值向量为 $\boldsymbol{s}=(0,0,\cdots,0)$,其错误交互集 Π_t 中错误交互数最多为 d,$t+d\leq k$,记交互集

$$\Gamma=\{\{(i_1,a_{i_1}),(i_2,a_{i_2}),\cdots,(i_t,a_{i_t}),(i_{t+1},0),(i_{t+2},0),\cdots,(i_{t+d},0)\}\mid a_{i_1}\in[1,v_{i_1}-1],a_{i_2}\in[1,v_{i_2}-1],\cdots,a_{i_t}\in[1,v_{i_t}-1]\},$$

则部分覆盖表 $PMCA(n;t+d,(v_1,v_2,\cdots,v_k),\Gamma)$ 就是错误定位表 $ELA(n;\bar{t},(v_1,v_2,\cdots,v_k))$.

证明. 设部分覆盖表 $PMCA(n;t+d,(v_1,v_2,\cdots,v_k),\Gamma)$ 为 A,则对任意的交互 $I=\{(i_1,a_{i_1}),(i_2,a_{i_2}),\cdots,(i_u,a_{i_u})\}(u\leq t)$,若 I 没有包含 $\Pi_t\backslash\{I\}$ 中的任何错误交互,对任意的交互 $I'\in\Pi_t\backslash\{I\}$,若 I' 对应的因素集是 I 对应的因素集的子集,则任何覆盖交互 I 的测试用例必定不覆盖 I'. 设 $\Pi_t\backslash\{I\}$ 中剩下的交互数为 $d'\leq d$,即设 $\{I_1,I_2,\cdots,I_{d'}\}=\Pi_t\backslash\{I'\mid I'$ 对应的因素集是 I 对应的因素集的子集$\}$,则存在 $j_1,j_2,\cdots,j_{d'}\in[1,k]\backslash\{i_1,i_2,\cdots,i_t\}$,使得 j_i 是交互 I_i 包含的某个顶点对应的因素,即存在 $a_{j_i}\in[1,v_{j_i}-1]$,$i=1,2,\cdots,d'$ 使得 $(j_i,a_{j_i})\in I_i$,不妨设 $j_1,j_2,\cdots,j_{d'}$ 互不相同. 由于 A 为 $PMCA(n;t+d,(v_1,v_2,\cdots,v_k),\Gamma)$,所以存在某一行所对应的测试用例覆盖交互 $\{(i_1,a_{i_1}),(i_2,a_{i_2}),\cdots,(i_u,a_{i_u}),(i_{u+1},a_{i_{u+1}}),\cdots,(i_t,a_{i_t}),(j_1,0),(j_2,0),\cdots,(j_{d'},0),(i_{t+d'+1},0),\cdots,(i_{t+d},0)\}$,其中 $i_{u+1},\cdots,i_t,i_{t+d'+1},\cdots,i_{t+d}$ 互不相同且取与 $i_1,i_2,\cdots,i_u,j_1,j_2,\cdots,j_{d'}$ 不相同的任意因素,$a_{u+1},\cdots,a_t$ 任取非零

的合法值，由于这一行不可能覆盖 $\Pi_t\backslash\{I\}$ 中的错误交互，所以这一行定位了交互 I. 所以 Π_t 是可定位的，所以表 A 是一个错误定位表 $ELA(n;\bar{t},(v_1,v_2,\cdots,v_k))$.　　证毕.

定理 1 表明，对极小错误交互数最多为 d 的待测系统(SUT)，具有安全值向量 $\boldsymbol{s}=(0,0,\cdots,0)$，只需构造部分覆盖表 $PMCA(n;t+d,(v_1,v_2,\cdots,v_k),\Gamma)$ 作为它的错误定位表即可，根据 Γ 的定义知，$PMCA(n;t+d,(v_1,v_2,\cdots,v_k),\Gamma)$ 表示一个 $n\times k$ 表，任取 $t+d$ 列，任何形如 $(0,\cdots,0,v_{i_1},0,\cdots,0,v_{i_2},0,\cdots,0,v_{i_t})$ 的 $t+d$ 维向量都能被某一行所覆盖，其中 $v_{i_j}\neq 0$ 为对应列所对应因素的合法取值. 那么 $PMCA(n;t+d,(v_1,v_2,\cdots,v_k),\Gamma)$ 的行数的上界是多少？我们先来研究这个问题，首先我们给出下面定理.

定理 2.　记交互集

$$\Gamma=\{\{(i_1,a_{i_1}),(i_2,a_{i_2}),\cdots,(i_t,a_{i_t}),(i_{t+1},0),(i_{t+2},0),\cdots,(i_{t+d},0)\}\mid a_{i_1}\in[1,v_{i_1}-1],a_{i_2}\in[1,v_{i_2}-1],\cdots,a_{i_t}\in[1,v_{i_t}-1]\},$$

则部分覆盖表数

$$PMCAN(t+d,(v_1,v_2,\cdots,v_k),\Gamma)\leq\sum_{1\leq i_1<i_2<\cdots<i_t\leq k}(v_{i_1}-1)(v_{i_2}-1)\cdots(v_{i_t}-1)\leq\binom{k}{t}(v-1)^t,$$

其中 $v=\max\{v_1,v_2,\cdots,v_k\}$.

证明.　构造 PMCA 如下：对每个 t 维交互 $\{(i_1,a_{i_1}),(i_2,a_{i_2}),\cdots,(i_t,a_{i_t})\}$，$a_{i_1}\in[1,v_{i_1}-1]$，$a_{i_2}\in[1,v_{i_2}-1]$，$\cdots$，$a_{i_t}\in[1,v_{i_t}-1]$ 构造一条测试用例，使得这条测试用例覆盖这个交互且其余因素取值为 0，把这条测试用例作为 $PMCA$ 的一行，所有这些行构成了满足要求的 $PMCA$. 该 $PMCA$ 的行数为

$$\sum_{1\leq i_1<i_2<\cdots<i_t\leq k}(v_{i_1}-1)(v_{i_2}-1)\cdots(v_{i_t}-1),$$

从而

$$PMCAN(t+d,(v_1,v_2,\cdots,v_k),\Gamma)\leq\sum_{1\leq i_1<i_2<\cdots<i_t\leq k}(v_{i_1}-1)(v_{i_2}-1)\cdots(v_{i_t}-1)\leq\binom{k}{t}(v-1)^t,$$

其中 $v=\max\{v_1,v_2,\cdots,v_k\}$.　　证毕.

定理 2 中估计的部分覆盖表数是关于 k 多项式增长的，下面我们用 Lovász local lemma 来进一步估计部分覆盖表行数的上界.

引理 1[21] (Lovász local lemma).　设 $A_1,A_2,\cdots,A_n$ 是某概率空间的事件，假设事件 A_i 与余下的除了至多 r 个事件以外的其它事件互相独立，且 $P(A_i)\leq p,i=1,2,\cdots,n$. 如果 $ep(r+1)\leq 1$，则 $P(\bigcap_{i=1}^{n}A_i^C)>0$.

有了这个引理，我们就采取如文献[21]中方法来估计 $PMCA(n;t+d,(v_1,v_2,\cdots,v_k),\Gamma)$ 行数的上界. 考虑当 $v_1=v_2=\cdots=v_k=v$ 情形，我们有下面定理.

定理 3.　当 $t+d\leq k$ 时，下列不等式成立：

$$PCAN(t+d,k,v,\Gamma)\leq\left(\frac{(v-1)e}{t}\right)^t(t+d)^t(t+d-1)\log k(1+o(1)),k\to\infty,$$

$$PCAN(t+d,k,v,\Gamma)=O(d^{t+1}\log k),\ d\to\infty.$$

证明.　用概率的方法来构造 $n\times k$ 表，设表中元素以概率 α 取值为 0，以概率 $\frac{1-\alpha}{v-1}$ 取值为 $i\in[1,v-1]$，设 A_i 表示第 i 个 $t+d$ 列不满足覆盖条件的事件，即某个 $t+d$ 维向量 $(0,\cdots,0,v_{i_1},0,\cdots,0,v_{i_2},0,\cdots,0,v_{i_t})$ 没有被覆盖. 则

$$\begin{aligned}P(A_i)&=P(\bigcup(0,\cdots,0,v_{i_1},0,\cdots,0,v_{i_2},0,\cdots,0,v_{i_t})\text{ 没有被覆盖})\\&\leq(v-1)^t\binom{t+d}{t}P((\overbrace{1,\cdots,1}^{t\text{个}1},0,\cdots,0)\text{没有被覆盖})\\&=(v-1)^t\binom{t+d}{t}\left(1-\alpha^d\left(\frac{1-\alpha}{v-1}\right)^t\right)^n,\end{aligned}$$

而每个 $t+d$ 列取值与其余的至多 r 个 $t+d$ 列取值互相有影响，所以 A_i 与最多 r 个事件不互相独立，则

$$r+1\leq(t+d)\binom{k-1}{t+d-1}\leq\frac{t+d}{(t+d-1)!}k^{t+d-1},$$

从而，当

$$e(v-1)^t\binom{t+d}{t}\left(1-\alpha^d\left(\frac{1-\alpha}{v-1}\right)^t\right)^n\frac{t+d}{(t+d-1)!}k^{t+d-1}\leq 1\qquad(*)$$

时，$P(\bigcap_{i=1}^{\binom{k}{t+d}}A_i^C)>0$，即构造的表是一个 $PMCA(n;t+d,k,v,\Gamma)$.

设 $1-\alpha^d\left(\frac{1-\alpha}{v-1}\right)^t=q$，则解式(*)得

$$n\geq-\frac{1}{\log q}(2\log(t+d)+\log e+t\log(v-1)+(t+d-1)\log k)-\log(t!)-\log(d!)),$$

而 $q=1-\alpha^d\left(\frac{1-\alpha}{v-1}\right)^t=f(\alpha)$，在 $\alpha=\frac{d}{t+d}$ 时取到最小值，此时 $q=1-\frac{t^td^d}{(t+d)^{t+d}(v-1)^t}$，所以 $PMCA(n;t+d,k,v,\Gamma)$ 的行数为

$$n\geq-\frac{1}{\log q}(t+d-1)\log k(1+o(1)),\ k\to\infty,$$

即

$$PCAN(t+d,k,v,\Gamma)\leq -\frac{1}{\log q}(t+d-1)\log k(1+o(1)),\ k\to\infty.$$

其中，$q=1-\frac{t^t d^d}{(t+d)^{t+d}(v-1)^t}$. 由

$$q=1-\frac{t^t d^d}{(t+d)^{t+d}(v-1)^t},$$

$$\left(1+\frac{t}{d}\right)^d=\left(\left(1+\frac{t}{d}\right)^{\frac{d}{t}}\right)^t\leq e^t,$$

可得

$$-\frac{1}{\log q}\leq\frac{(t+d)^{t+d}(v-1)^t}{t^t d^d}\leq\left(\frac{v-1}{t}\right)^t(t+d)^t\left(1+\frac{t}{d}\right)^d\leq\left(\frac{(v-1)e}{t}\right)^t(t+d)^t,$$

从而有

$$PCAN(t+d,k,v,\Gamma)\leq\left(\frac{(v-1)e}{t}\right)^t(t+d)^t(t+d-1)\log k(1+o(1)),\ k\to\infty,$$

$$PCAN(t+d,k,v,\Gamma)=O(d^{t+1}\log k),\ d\to\infty,k\to\infty.$$

证毕.

由定理 1 及定理 3 知，对极小错误交互数最多为 d 的待测系统(SUT)，具有安全值向量 $\boldsymbol{s}=(0,0,\cdots,0)$，用部分覆盖表来构造错误定位表其行数关于 k 是对数增长的，关于 d 是多项式增长的.

4 生成部分覆盖表的贪心算法

对于生成部分覆盖表，我们可以用生成一般的覆盖表方法来生成部分覆盖表，如贪心算法、智能搜索等，然而用一次只生成一条测试用例的贪心算法来生成部分覆盖表，其生成覆盖表的行数关于 d 是多项式增长的吗？由定理 3 中证明知道，设表中元素以概率$\frac{d}{t+d}$取值为 0，以概率$\frac{t}{t+d}$取值非 0，得到的表是部分覆盖表的概率大于零时表的行数最少，所以我们对于一次只生成一条测试用例的贪心算法，如果每次选择的测试用例都是有$\left\lceil k\frac{d}{t+d}\right\rceil$个 0，$k-\left\lceil k\frac{d}{t+d}\right\rceil$个非 0，且覆盖了尽可能多的未被覆盖的 $t+d$ 维交互，则有可能最后生成的部分覆盖表行数较少，所以我们得到算法 2.

算法 2. 一次只生成一条测试用例的贪心算法.

输入：$t,d,(v_1,v_2,\cdots,v_k),\Gamma$

输出：部分覆盖表 $PMCA(n;t+d,(v_1,v_2,\cdots,v_k),\Gamma)A$

```
begin
  if t+d>k
    A=空表
  else
    Uncover=Γ      //未被覆盖的 t+d 维交互集
    n=0
    while (Uncover≠∅)
      n=n+1
      在包含⌈k d/(t+d)⌉个 0，k-⌈k d/(t+d)⌉个非 0 的
      测试用例集中，选择一条覆盖了最多尚未被覆
      盖的 t+d 维交互的测试用例. 将这条测试用例
      作为覆盖表 A 的 n 行.
      Uncover=Uncover\{覆盖表 A 的第 n 行所覆
      盖的 t+d 维交互}  //更新 Uncover
    endwhile
  endif
  return A
end
```

由算法 2，可以推出如下定理.

定理 4. 设 $v_1=v_2=\cdots=v_k=v$，当 $t+d\leq k$，$k\frac{d}{t+d}$为整数时，对于一次只生成一条测试用例的贪心算法 2，生成的部分覆盖表的行数 $n=O(d^{t+1}\log k)$，$d\to\infty$，$k\to\infty$.

证明. 类似于文献[10]中证明覆盖表行数是因素数的对数阶增长的方法，设在选取第 m 个测试用例后，集合 $Uncover$ 中未被覆盖的 $t+d$ 维交互个数为 D_m，则

$$D_0=|\Gamma|=\binom{k}{t+d}\binom{t+d}{t}(v-1)^t.$$

设 Φ 为包含 $k\frac{d}{t+d}$个 0，$k\frac{t}{t+d}$个非 0 的测试用例集，在选取第 m 个测试用例后，集合 $U_m=\{(T,I)\mid I\in Uncover,T\in\Phi,T\text{ 覆盖交互 }I\}$.

我们用两种方法计算 U_m 中元素个数 $|U_m|$.

对每个交互 $I\in Uncover$，由于在 Φ 中覆盖交互 I 的测试用例个数为$\binom{k-t-d}{k\frac{d}{t+d}-d}(v-1)^{k\frac{t}{t+d}-t}$，可知 $|U_m|=D_m\cdot\binom{k-t-d}{k\frac{d}{t+d}-d}(v-1)^{k\frac{t}{t+d}-t}$，而 Φ 中测试用例的条数$|\Phi|=\binom{k}{k\frac{d}{t+d}}(v-1)^{k\frac{t}{t+d}}$，所以平均每条测试用例覆盖未被覆盖的 $t+d$ 维交互数为

$$\frac{|U_m|}{|\Phi|}=\frac{D_m\cdot\begin{pmatrix}k-t-d\\ k\dfrac{d}{t+d}-d\end{pmatrix}(v-1)^{k\frac{t}{t+d}-t}}{\begin{pmatrix}k\\ k\dfrac{d}{t+d}\end{pmatrix}(v-1)^{k\frac{t}{t+d}}}$$

$$=\frac{D_m k\,\frac{d}{t+d}\left(k\,\frac{d}{t+d}-1\right)\cdots\left(k\,\frac{d}{t+d}-d+1\right)k\,\frac{t}{t+d}\left(k\,\frac{t}{t+d}-1\right)\cdots\left(k\,\frac{d}{t+d}-t+1\right)}{k(k-1)\cdots(k-t-d+1)(v-1)^t}$$

$$>\frac{D_m t^t d^d}{(t+d)^{t+d}(v-1)^t},$$

从而

$$D_{m+1}\leq D_m\left(1-\frac{t^t d^d}{(t+d)^{t+d}(v-1)^t}\right),$$

设生成的部分覆盖表的行数为 n，则 $D_{n+1}<1$，故

$$D_n<D_0\left(1-\frac{t^t d^d}{(t+d)^{t+d}(v-1)^t}\right)^n\leq 1,$$

所以

$$n=\frac{-\log D_0}{\log\left(1-\frac{t^t d^d}{(t+d)^{t+d}(v-1)^t}\right)}.$$

再由

$$\log\left(1-\frac{t^t d^d}{(t+d)^{t+d}(v-1)^t}\right)\leq-\frac{t^t d^d}{(t+d)^{t+d}(v-1)^t}$$

可得

$$n\leq\frac{(t+d)^{t+d}(v-1)^t\log D_0}{t^t d^d}$$

$$=\frac{(t+d)^{t+d}(v-1)^t\log\left(\begin{pmatrix}k\\ t+d\end{pmatrix}\begin{pmatrix}t+d\\ t\end{pmatrix}(v-1)^t\right)}{t^t d^d},$$

从而有 $n=O(d^{t+1}\log k)$，$d\to\infty$，$k\to\infty$.　　证毕.

定理 4 表明算法 2 生成的部分覆盖表行数是关于 d 多项式增长，关于 k 对数增长，这与文献[6]中结论相比是一个比较大的改进.

我们的贪心算法在选择下一条测试用例时，要求在包含$\left[k\,\frac{d}{t+d}\right]$个 0，$k-\left[k\,\frac{d}{t+d}\right]$个非 0 的测试用例集中，选择一条覆盖了最多尚未被覆盖的 $t+d$ 维交互的测试用例，这在实践中每次能选到最优的测试用例是不可能的，所以在实际操作中我们用一些启发式方法来选择下一条测试用例，使其尽可能多地覆盖尚未被覆盖的 $t+d$ 维交互(算法 3).

算法 3.　局部模拟退火算法.

输入：$t,d,(v_1,v_2,\cdots,v_k),\Gamma$

输出：部分覆盖表$PMCA(n;t+d,(v_1,v_2,\cdots,v_k),\Gamma)$ A

```
begin
  if t+d>k
    return 空表
  endif
  Uncover=Γ      //未被覆盖的 t+d 维交互集
  n=0
  while (Uncover≠∅)
    n=n+1
    随机的选取⌊k d/(t+d)⌋个因素为 0，k−⌊k d/(t+d)⌋个
    因素非 0 的测试用例 T，Temperature 初始化
    δ=测试用例 T 覆盖的未被覆盖的交互个数
    until(δ 稳定的条件) do
      T′=T
      随机选择因素 i，
      if T_i=0 then
        随机选择 v∈[1,v_i−1]，T′_i=v
        随机选择使得 T_j≠0 的 j，T′_j=0
      else
        随机选择 v∈[1,v_i−1]\{T_i}，T′_i=v
      endif
      δ′=测试用例 T′覆盖的未被覆盖的交互个数
      if δ′−δ>0 then
        T=T′
      else
        以概率 e^{(δ′−δ)/Temperature} do
          T=T′
      endif
      Temperature=cool(Temperature)
    enduntil
    将这条测试用例 T 作为覆盖表 A 的 n 行.
    Uncover=Uncover\{覆盖表 A 的第 n 行所覆盖的
            t+d 维交互}  //更新 Uncover
  endwhile
  return A
end
```

部分覆盖表作为错误定位表与用 $MCA(n;t+d,(v_1,v_2,\cdots,v_k))$来构造错误定位表相比，虽然其行数在理论上是一个大的改进，但是在生成部分覆盖表时，随着 d 的增大，同样会遭遇需覆盖的交互数呈指

数爆炸增长的困难，如何克服这个困难来生成 d,k 较大时的部分覆盖表？我们准备在未来的工作中采用组群测试中的池设计办法来生成部分覆盖表[22-23].

5 实例分析

假设一个待测系统有 5 个因素，第 2,4 因素有 3 个取值，其余因素有 2 个取值，系统中有 2 个两两错误交互{(1,1),(2,2)}{(2,2),(4,1)}，具有安全值向量(0,0,0,0,0)，错误交互如图 2 所示. 构造部分覆盖表时需覆盖的部分交互集 Γ 为

Γ={{(1,1),(2,1),(3,0),(4,0)}, {(1,1),(2,2),(3,0),(4,0)}, {(1,1),(2,0),(3,1),(4,0)}, {(1,1),(2,0),(3,0),(4,1)}, {(1,1),(2,0),(3,0),(4,2)}, {(1,0),(2,1),(3,1),(4,0)}, {(1,0),(2,2),(3,1),(4,0)}, {(1,0),(2,1),(3,0),(4,1)}, {(1,0),(2,1),(3,0),(4,2)}, {(1,0),(2,2),(3,0),(4,1)}, {(1,0),(2,2),(3,0),(4,2)}, {(1,0),(2,0),(3,1),(4,1)}, {(1,0),(2,0),(3,1),(4,2)}, {(1,1),(2,1),(3,0),(5,0)}, {(1,1),(2,2),(3,0),(5,0)}, {(1,1),(2,0),(3,1),(5,0)}, {(1,1),(2,0),(3,0),(5,1)}, {(1,0),(2,1),(3,1),(5,0)}, {(1,0),(2,2),(3,1),(5,0)}, {(1,0),(2,1),(3,0),(5,1)}, {(1,0),(2,2),(3,0),(5,1)}, {(1,0),(2,0),(3,1),(5,1)}, {(1,1),(2,1),(4,0),(5,0)}, {(1,1),(2,2),(4,0),(5,0)}, {(1,1),(2,0),(4,1),(5,0)}, {(1,1),(2,0),(4,2),(5,0)}, {(1,1),(2,0),(4,0),(5,1)}, {(1,0),(2,1),(4,1),(5,2)}, {(1,0),(2,1),(4,2),(5,0)}, {(1,0),(2,2),(4,1),(5,0)}, {(1,0),(2,2),(4,2),(5,0)}, {(1,0),(2,1),(4,0),(5,1)}, {(1,0),(2,2),(4,0),(5,1)}, {(1,0),(2,0),(4,1),(5,1)}, {(1,0),(2,0),(4,2),(5,1)}, {(1,1),(3,1),(4,0),(5,0)}, {(1,1),(3,0),(4,1),(5,0)}, {(1,1),(3,0),(4,2),(5,0)}, {(1,1),(3,0),(4,0),(5,1)}, {(1,0),(3,1),(4,1),(5,0)}, {(1,0),(3,1),(4,2),(5,0)}, {(1,0),(3,1),(4,0),(5,1)}, {(1,0),(3,0),(4,1),(5,1)}, {(1,0),(3,0),(4,2),(5,1)}, {(2,1),(3,1),(4,0),(5,0)}, {(2,2),(3,1),(4,0),(5,0)}, {(2,1),(3,0),(4,1),(5,0)}, {(2,1),(3,0),(4,2),(5,0)}, {(2,2),(3,0),(4,1),(5,0)}, {(2,2),(3,0),(4,2),(5,0)}, {(2,1),(3,0),(4,0),(5,1)}, {(2,2),(3,0),(4,0),(5,1)}, {(2,1),(3,0),(4,1),(5,1)}, {(2,1),(3,0),(4,2),(5,1)}, {(2,2),(3,0),(4,1),(5,1)}, {(2,2),(3,0),(4,2),(5,1)}}.

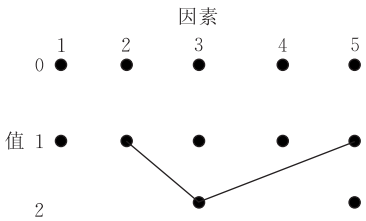

图 2 5 因素错误交互图

而生成的部分覆盖表为

0 0 1 0 1	0 1 1 0 0	1 0 0 1 0	0 1 0 0 1
0 0 1 1 0	0 2 0 2 0	1 0 1 0 0	0 1 0 1 0
0 2 0 0 1	0 2 1 0 0	0 0 1 2 0	1 2 0 0 0
1 0 0 2 0	0 0 0 1 1	0 2 0 1 0	0 1 0 2 0
0 0 0 2 1	1 0 0 0 1	1 1 0 0 0	

这个部分覆盖表也就是在已知安全值向量 s=(0,0,0,0,0)且已知错误交互数至多为 2 时的错误定位表 $ELA(19;\overline{2},(2,3,2,3,2))$，运行每一行所对应的测试用例，再根据运行的结果，利用算法 1 就能定位出错误交互集{(1,1),(2,2)}{(2,2),(4,1)}. 如果根据定理 2，用混合覆盖表 $MCA(n;2+2,(2,3,2,3,2))$来生成 $ELA(n;\overline{2},(2,3,2,3,2))$，则行数 $n\geq 36$.

对于一般情形，我们运行算法 3，生成的部分混合覆盖表的规模列表如表 3 所示.

表 3 部分混合覆盖表与覆盖表行数比较

类型	PCA 行数	CA 的行数	类型	PCA 行数	CA 的行数
$(2,2,2^6)^*$	16/15	21	$(2,4,3^{20})^*$	763/760	4486
$(2,2,3^{10})$	119/180	159	$(2,4,4^{15})^*$	1235/945	16384
$(2,2,4^{13})$	361/702	508	$(2,4,5^{15})^*$	2207/1680	82139
$(2,2,5^{15})$	743/1680	1245	$(3,2,2^{10})$	47/120	56
$(2,2,6^{10})$	751/1125	3237	$(3,2,3^{14})$	696/2912	922
$(2,3,2^{20})$	107/190	119	$(3,2,4^{15})$	2592/12285	3064
$(2,3,3^{20})$	460/760	1266	$(3,3,2^{10})$	107/120	116
$(2,3,4^{20})$	1058/1710	5516	$(3,3,3^{15})$	1686/3640	3234
$(2,3,5^{15})$	1417/1680	12704	$(3,3,4^{12})$	4093/5940	14284

表中$(2,2,2^6)$表示 $t=2,d=2$ 有 6 个因素，每个因素有 2 个取值的待测系统，其它类似，PCA 对应的强度为 $t+d$ 的部分覆盖表，PCA 的行数栏对应两个数值，线“/”左边表示运行 5 次算法 3 取的最小的行数，线“/”右边表示用定理 2 得到的上界. CA 表示对应的强度为 $t+d$ 的覆盖表. 其中 CA 的行数

是引自由 Charlie Colbourn 维护的覆盖表网址①，上面是已知的覆盖表数的最好上界.

表中带“*”栏中表明用定理 2 得到行数比算法 3 得到的行数要小，主要原因是因素数比较少错误交互数比较多时，算法 3 复杂度较高，其构造 PCA 行数反而比较多，随着因素数越来越大时，算法 3 优势才能发挥出来，PCA 行数关于因素数 k 及错误交互数 d 的渐近性质才能得到显示. 从表中可看出用部分覆盖表代替覆盖表来生成错误定位表，测试用例规模大大约简了.

6　结论及进一步工作

本文中我们在Colbourn和McClary[4]及Martínez等[5-6]基础上研究了在组合测试中怎样定位至多 t 维的错误交互的非自适应算法，即在安全值已知待测系统中，我们用新的组合结构部分混合覆盖表来生成错误定位表. 所以我们未来的研究工作主要集中在：(1) 定理 3 中的理论证明了表中元素以概率 $\frac{d}{t+d}$ 取值为 0，以概率 $\frac{t}{t+d}$ 取值非 0，得到的表是部分覆盖表的概率大于零时表的行数最少，然而实践上能否支持这一结论，我们准备做一些实验，研究对不同的取值为 0 的参数 α，比较部分覆盖表大小. (2) 研究生成部分混合覆盖表的高效算法. 因为当 d、k 比较大时，用生成一般的混合覆盖表的算法移植来生成部分覆盖表时效率很低，因为需被覆盖的交互集 Γ 的规模呈爆炸式增长，所以我们研究当 d、k 比较大时部分混合覆盖表生成，以及研究这些算法与组群测试(group testing)里的池设计(pooling design)之间的关系[22-23]. (3) 研究对具有安全值但安全值未知的待测系统构造多阶段的非自适应算法，因为本文研究用部分覆盖表来生成错误定位表的前提是已知安全值向量. (4) 对于非自适应算法生成错误定位表，然后用算法 1 就可自动定位触发系统故障的错误交互，当生成的错误定位表比较大，冗余的测试用例较多，尤其在 d 比较大时，生成错误定位表比较困难，所以我们研究定位错误交互的高效自适应算法，即根据前面的测试用例的运行结果来生成下一条测试用例的算法.

参　考　文　献

[1] 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

[2] Yilmaz C, Cohen M B, Porter M B. Covering arrays for efficient fault characterization in complex configuration spaces. IEEE Transactions on Software Engineering, 2006, 32(1): 20-34

[3] Xu Bao-Wen, Nie Chang-Hai, Shi Liang, Chen Huo-Wang. 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，陈火旺. 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)

[4] Colbourn C J, McClary D W. Locating and detecting arrays for interaction faults. Journal of Combinatorial Optimization, 2008, 15(1): 17-48

[5] Martínez C, Moura L, Panario D, Stevens B. Algorithms to locate errors using covering arrays//Proceedings of the 2008 8th Latin American Theoretical Informatics. Lecture Notes in Computer Science 4957. Buzios, Brazil, 2008: 504-519

[6] Martínez C, Moura L, Panario D, Stevens B. Locating errors using ELAs, covering arrays and adaptive testing algorithms. SIAM Journal on Discrete Mathematics, 2009, 23(4): 1776-1799

[7] Yan Jun, Zhang Jian. Combinatorial testing: Principles and methods. Journal of Software, 2009, 20(6): 1393-1405(in Chinese)
(严俊，张健. 组合测试：原理和方法. 软件学报，2009，20(6)：1393-1405)

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

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

[10] 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

[11] Tung Y W, Aldiwan W S. Automating test case generation for the new generation mission software system//Proceedings of the IEEE Arospace Conference. Big Sky, MT, USA, 2000: 431-437

[12] Bryce R C, Colbourn C J. The density algorithm for pairwise interaction testing. Software Testing, Verification and Reliability, 2007, 17(3): 159-182

[13] Bryce R C, Colbourn C J. A density-based greedy algorithm for higher strength covering arrays. Software Testing, Verification and Reliability, 2009, 19(1): 37-53

① Colbourn C J. Covering array tables. http://www.public.asu.edu/ccolbou/src/tabby

[14] 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: Technical Report TR-2001-03, 2001

[15] Lei Y, Kacker R, Kuhn D R, Okun V, Lawrence J. IPOG/IPOG-D: Efficient test generation for multi-way combinatorial testing. Software Testing, Verification and Reliability, 2008, 18(3): 125-148

[16] 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

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

[18] 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

[19] Shiba Toshiaki, Tsuchiya Tatsuhiro, Kikuno Tohru. Using artificial life techniques to generate test cases for combinatorial testing//Proceedings of the COMPSAC04. Hong Kong, China, 2004: 72-78

[20] Godbole A, Skipper D, and Sunley R. t-covering arrays: Upper bounds and Poisson approximations. Combinatorics Probability and Computing, 1996, 5: 105-118

[21] Carey P, Godbole A. Partial covering arrays and a generalized Erdös-Ko-Rado property. Journal of Combinatorial Designs, 2010, 18: 155-166

[22] Du D Z, Hwang F K. Pooling Designs and Nonadaptive Group Testing: Important Tools for DNA Sequencing. New York: World Scientific Publishing Company, 2006

[23] Cheng Y X, Du D Z. New constructions of one- and two-Stage pooling designs, Journal of Computational Biology, 2008, 15(2): 195-205

ZHOU Wu-Jie, born in 1973, Ph. D. candidate, lecturer. His research interests include software testing and statistical testing etc.

ZHANG De-Ping, born in 1973, Ph. D., lecturer. His research interests include teaching and research on software testing, statistical testing and mathematical statistics.

XU Bao-Wen, born in 1961, Ph. D., professor, Ph. D. supervisor. His current research interests include programming language, software engineering (software methodology, software analysis, software metrics and software testing), and Web technology.

Background

This paper is supported by the National Natural Science Foundation of China under Grant Nos. (90818027, 91018005), the National High-Tech Research Subjects and Development Plan of China under Grant No. 2009AA01Z147, and the National Grand Fundamental Research 973 Program of China under Grant No. 2009CB320703. These projects mainly research on the follows 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 the applications. This paper focuses on the research of software fault diagnosis and debugging based on combinatorial testing.