ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.6, June 2009, pp.1393−1405
doi: 10.3724/SP.J.1001.2009.03497

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

组合测试:原理与方法*

严 俊+, 张 健

(中国科学院 软件研究所 计算机科学国家重点实验室,北京 100190)

Combinatorial Testing: Principles and Methods

YAN Jun+, ZHANG Jian

(State Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

+ Corresponding author: E-mail: yanjun@ios.ac.cn

Yan J, Zhang J. Combinatorial testing: Principles and methods. *Journal of Software*, 2009,20(6):1393−1405.
http://www.jos.org.cn/1000-9825/3497.htm

Abstract: Combinatorial testing can use a small number of test cases to test systems while preserving fault detection ability. However, the complexity of test case generation problem for combinatorial testing is NP-complete. The efficiency and complexity of this testing method have attracted many researchers from the area of combinatorics and software engineering. This paper summarizes the research works on this topic in recent years. They include: various combinatorial test criteria, the relations between the test generation problem and other NP-complete problems, the mathematical methods for constructing test cases, the computer search techniques for test generation and fault localization techniques based on combinatorial testing.

Key words: combinatorial testing; covering array; test case generation

摘 要: 组合测试能够在保证错误检出率的前提下采用较少的测试用例测试系统.但是,组合测试用例集的构造问题的复杂度是NP完全的.组合测试方法的有效性和复杂性吸引了组合数学领域和软件工程领域的学者们对其进行深入的研究.总结了近年来在组合测试方面的研究进展,主要内容包括:组合测试准则的研究、组合测试生成问题与其他NP完全问题的联系、组合测试用例的数学构造方法、采用计算机搜索的组合测试生成方法以及基于组合测试的错误定位技术.

关键词: 组合测试;覆盖数组;测试用例生成
中图法分类号: TP301 **文献标识码**: A

在软件的功能测试中,可以通过检查系统参数的所有取值组合来进行充分的测试.例如:对一个具有k个参数的待测系统(software under test,简称SUT),这些参数分别有$v_1,v_2,\ldots,v_k$个可能取值,完全测试这个系统需要$\prod_{i=1}^{k}v_i$个测试用例.对于一般的被测系统而言,这个组合数是一个很庞大的数字.如何从中选择一个规模较小的子集作为测试用例集是测试用例生成(test case generation)中一个很重要的问题.在测试性能和代价上的一个折衷就是组合测试(combinatorial testing),因为根据观察,对于很多应用程序来说,很多程序错误都是由少数几个参

* Supported by the National Natural Science Foundation of China under Grant Nos.60673044, 60633010 (国家自然科学基金)
Received 2008-05-16; Revised 2008-08-07; Accepted 2008-10-07

数的相互作用导致的.例如:Kuhn和Reilly分析了Mozilla浏览器的错误报告记录,发现超过 70%的错误是由某两个参数的相互作用触发的,超过 90%的错误是由 3 个以内的参数互相作用而引发的[1].这样,我们可以选择测试用例,使得对于任意t(t是一个小的正整数,一般是 2 或者 3)个参数,这t个参数的所有可能取值的组合至少被一个测试用例覆盖.我们称这种测试准则(test criterion)为t组合测试.

组合测试方法在系统测试中是非常有效的,因为对于有k个参数的系统来说,完成t组合测试所需要的最小测试用例数目是按照k的对数级增长的[2].美国国家标准和技术协会(NIST)的Kuhn等人采用 4 个软件系统研究了组合测试的错误检出率.其结果表明,3 组合测试的错误检出率已经超过 80%[3].Grindal的研究指出,组合测试方法具有模型简单、对测试人员要求低、能够有效处理较大规模的测试需求的特点,是一种可行的实用测试方案[4].

下面用一个简单的例子来说明组合测试方法.表 1 描述了一个电子商务系统,这个系统有 4 个参数,每个参数有 3 个可选值,完全测试该系统需要 3^4=81 个测试用例.

采用 2 组合测试准则,测试时仅需要表 2 中的 9 个测试用例,即可覆盖任意两个参数的所有取值组合.

Table 1 A system with four parameters

表 1 一个四参数系统

Client	Web Server	Payment	Database
Firefox	WebSphere	MasterCard	DB/2
IE	Apache	Visa	Orade
Opera	.NET	UnionPay	Access

Table 2 Test suite covering all pairs

表 2 2 组合测试用例

Test No.	Client	Web Server	Payment	Database
1	Firefox	WebSphere	MasterCard	DB/2
2	Firefox	.NET	UnionPay	Orade
3	Firefox	Apache	Visa	Access
4	IE	WebSphere	UnionPay	Access
5	IE	Apache	MasterCard	Orade
6	IE	.NET	Visa	DB/2
7	Opera	WebSphere	Visa	Orade
8	Opera	.NET	MasterCard	Access
9	Opera	Apache	UnionPay	DB/2

本文总结了近年来学术界和工业界在组合测试上的研究成果.第 1 节介绍组合测试的基础概念.第 2 节介绍部分数学构造方法.第 3 节介绍利用其他 NP 完全问题来研究组合测试生成问题.第 4 节分类介绍采用计算机搜索方法自动生成测试用例集合的方法.第 5 节介绍如何利用组合测试的结果定位错误的位置.第 6 节是总结和展望.

1 基本概念

大部分的组合测试方法,也就是传统的组合测试方法,都是基于覆盖数组的.另外,针对一部分特殊的软件测试,学术界提出了一些变种的组合测试方法.

1.1 覆盖数组

从表2可以看出,组合测试用例集合可以用一个矩阵来表示,矩阵的每一行表示一个测试用例,每一列代表系统的一个参数,每一项(entry)代表测试用例的相应参数的取值.Cohen等人给出了如下覆盖数组CA(covering array)和混合覆盖数组MCA(mixed level covering array)的定义[5],用以描述测试用例集.

定义1(覆盖数组). 覆盖数组CA(N;t,k,v)是一个值域大小为v的$N\times k$矩阵,任意的$N\times t$子矩阵包含了在v值域上所有大小为t的排列.这里,t被称为强度(strength,国内某些文献将其翻译为水平,见文献[6]).k被称为阶数(degree),v称为序(order).一个覆盖数组如果具有最小的行数,则被称为最优的.这个最小的行数称为覆盖数(covering array number),记为CAN(t,k,v).

一般地,强度为t的覆盖数组被称为t覆盖数组(t-covering array).特别地,t=2 的覆盖数组称为成对覆盖数组(pairwise covering array).覆盖数组要求矩阵的每一列具有相同大小的值域,表 2 中的测试集合即是一个覆盖数组CA(9;2,4,3).由于任意两个参数之间的组合至少有3^2=9 种,故这个覆盖数组是最优的.目前已知的一些覆盖数组的最好结果可以参见文献[7].

对于真实的程序,并不一定都满足每一个参数具有相同的值域这个限制,于是,我们可以进一步扩展覆盖数组的概念.

定义 2(混合覆盖数组). 混合覆盖数组MCA($N;t,k,v_1v_2\ldots v_k$)是一个由v个符号组成的$N\times t$矩阵,其中$v=\sum_{i=1}^{k}v_i$,并具有以下性质:

i) 第i列的所有符号是一个大小为v_i的集合S_i的元素.

ii) 任意的 $N\times t$ 子矩阵包含了在相应值域上的所有 t 元组.

类似地,我们可以定义混合覆盖数组的强度t以及覆盖数MCAN($t,k,v_1v_2\ldots v_k$).

在记录CA或者MCA时,可以把一些具有相同值域的项合并,并省略参数的个数k.例如,如果矩阵有3列,其值域大小都是2,则记为2^3.在这种情况下,一个MCA($N;t,k,v_1v_2\ldots v_k$)可以表示为 MCA($N;t,s_1^{p_1}s_2^{p_2}\ldots s_r^{p_r}$),其中$k=\sum_{i=1}^{r}p_i$.下面用一个对真实程序进行测试的例子来说明混合覆盖数组.

例 1(ldd 的测试):Linux 程序 ldd 的主要功能是打印目标文件 FILE(包括编译好的可执行程序或者动态链接库)依赖的动态链接库,它的用法是

ldd [OPTION] FILE

假定我们要对 ldd进行功能测试,根据 ldd 的用法,影响程序的参数包括所有选项(OPTION)以及输入文件(FILE).在实际测试时,显然无法穷举所有的文件,于是将 FILE 通过等价类划分(equivalence partitioning)抽象成 3 个等价类:无输入文件(unused)、无效文件(invalid file)以及有效文件(valid file).这样,根据 ldd(version 2.6.1)的使用说明,我们可以列出 ldd 的所有控制参数,见表 3.

Table 3 Parameters of ldd

表 3 ldd 的参数

Parameter	Meaning	Levels
FILE	Object program or shared library.	Unused, Invalid File, Valid File
--version	Print the version number of ldd.	Unused, Used
-v	Print all information.	Unused, Used
-u	Print unused direct dependencies.	Unused, Used
-d	Perform relocations and report any missing objects.	Unused, Used
-r	Perform relocations for both data objects and functions, and report any missing objects or functions.	Unused, Used
--help	Usage information.	Unused, Used

从表 3 可以看出,ldd有 6 个二值参数,1 个三值参数,那么,对于这样一个简单程序的所有可能的输入有$3^1\times2^6$=192 种.假定我们需要测试任意 3 个参数之间的所有取值组合,可以用表 4 的 18 个测试用例来对ldd进行测试,显然这一组测试用例构成了一个MCA(18;3,3^12^6).可以证明,对于这个实例至少需要 18 个测试用例(证明参见第 2 节),即这个测试集合是最优测试集.

区别CA和MCA的原因主要在于,在构造这些矩阵时,某些数学构造方法只适用于CA.而采用其他构造方法,尤其是在利用计算机搜索的自动化方法时,CA可以看成MCA的一个特例,在处理时并没有区别.下文中如无特别说明,本文所指的覆盖数组包含CA以及MCA.

Seroussi和Bshouty的工作[8]表明,构造最优t覆盖数组这个问题是NP完全的.同时,Lei等人[9]也证明了这个问题的判定问题,即判断一个被测系统是否有一个大小为N的成对覆盖集也是NP完全的.所以,我们不大可能找到一种对所有实例都很高效的构造算法.这两个问题,即寻找最优覆盖数组和其判定问题,目前已都有较多的研究[5].

Table 4 Test cases for ldd

表 4 ldd 的测试用例

Test No.	FILE	--version	-v	-u	-d	-r	--help
1	Unused	Unused	Unused	Unused	Unused	Unused	Unused
2	Unused	Unused	Used	Unused	Unused	Unused	Used
3	Unused	Used	Unused	Unused	Used	Used	Unused
4	Unused	Used	Used	Used	Unused	Used	Unused
5	Invalid file	Unused	Unused	Unused	Unused	Unused	Unused
6	Invalid file	Unused	Used	Unused	Unused	Unused	Used
7	Invalid file	Used	Unused	Unused	Used	Used	Unused
8	Invalid file	Used	Used	Used	Unused	Used	Unused
9	Valid file	Unused	Unused	Unused	Used	Used	Used
10	Valid file	Unused	Used	Used	Used	Unused	Unused
11	Valid file	Used	Unused	Used	Unused	Unused	Used
12	Valid file	Used	Used	Unused	Unused	Used	Used
13	Unused	Unused	Unused	Used	Used	Used	Used
14	Unused	Used	Used	Used	Used	Unused	Used
15	Invalid file	Unused	Used	Used	Used	Used	Used
16	Invalid file	Used	Unused	Used	Used	Unused	Used
17	Valid file	Unused	Unused	Used	Unused	Used	Unused
18	Valid file	Used	Used	Unused	Used	Unused	Unused

1.2 非经典组合测试

采用覆盖数组的组合测试要求任意 t 个参数的所有参数组合都必须被覆盖,而没有考虑具体参数的特性.近年来,一些学者将组合测试的方法进行了扩展,提出了一些非经典的组合测试方法.另外,对于某些测试实例而言,即使采用组合测试方法,测试用例数目还是太多.因此,学术界针对某些具体问题提出了一些非经典的组合测试方法,这些测试方法的主要出发点一般是对测试用例集合中的参数取值组合加入一些限制.这些限制主要包括以下几个方面.

1.2.1 种子组合

在实际测试中,用户可能会要求某些取值组合必须被测试,这种取值组合被称为种子(seed)[10].目前,很多贪心算法支持这些参数限制(参见第4.1节).例如,本文第4.1.1节介绍的AETG(automatic efficient test generator)等工具的输入规范AETGSpec[10]支持如下描述:

```
seed {
    a b c d
    2 4 8 3
}
```

这一段约束表示(a,b,c,d)=(2,4,8,3)这种组合形式在测试用例集中至少出现 1 次.

1.2.2 参数之间的限制

某些参数组合事实上是不允许出现或者是没有意义的.例如,简单的计算器,在非十进制的时候是不支持浮点数运算的,那么,“十六进制+浮点数”的组合是无效组合,测试中应该禁止出现.这种情况称为参数之间的冲突(conflict)[4].AETG[11]等工具支持参数之间具有这样的约束关系.例如,其规范支持形如

If b<9 then c>=8 and d<=3

的表达式,用以描述对参数b,c,d之间的限制.另外微软的免费测试工具PICT(pairwise independent combinatorial testing)也支持较为复杂的约束,包括数值和字符串的比较以及布尔逻辑约束[12].更多关于冲突处理的介绍见文献[4].

1.2.3 变强度组合测试

Cohen等人提出了变强度的组合测试方法,允许不同参数之间的覆盖强度不同[13].他们将覆盖数组扩展到了变强度覆盖数组.例如, $VCA(27;2,3^9 2^2,\{CA(27;3,3^3)^3\})$ 表示一个强度为 2 的混合覆盖数组,但是其中包含 3 个子数组,每个数组包含 3 个值域大小为 3 的参数,并且在每个子数组内部需要所有的参数组合满足 3 覆盖.这种方法适合于重点测试系统中的某些关键参数.

1.2.4 特殊的测试场景

某些被测程序可能不仅有多个输入,也可能有多个输出.如果每个输出O_j是由少数几个参数$\{I_{j_1},I_{j_2},...,I_{j_s}\}$所影响,那么测试集合需要覆盖影响任意一个输出的所有输入参数的取值组合[14].如何自动生成满足这种覆盖的测试用例集是一个很复杂的问题.Cheng等人指出,可以利用图染色方法将这个测试生成问题约简为变强度组合测试[15].

王子元等人针对一种特殊的测试场景提出了另一种组合测试方法[16].在这种场景中,仅有相邻的参数之间可能有关联.例如,在中国铁路的信号系统中,火车前一个信号灯的状态取决于之前3个区间的状态,这样可以采用强度为 3 的相邻因素组合测试方法来测试该系统.该文同时提出了一种自动测试生成方法,可以在多项式时间内产生最优测试用例集合.

相对于经典的组合测试方法,这些非经典的组合测试方法的研究不是很成熟.王子元等人借鉴了Cheng的集合覆盖问题思想(参见第 3.2 节),提出了一个较为通用的非经典组合测试框架,试图将这些非经典组合测试统一用参数的交叉关系(interaction relationship)来描述.所有的交叉关系组成一个集合(称为interaction coverage requirement),那么,非经典的组合测试生成问题可以统一为寻找覆盖这个关系集合的测试用例集合[17].

下文中的内容,若无特殊声明,针对的都是标准的组合测试.

2 数学构造方法

最早的构造方法是利用正交数组(orthogonal array,简称OA).OA的定义与CA类似,二者的区别是,对于任意的$N\times t$子矩阵,所有的t元组出现的次数(大于0)均相同.OA是被组合数学界研究很久的问题,$N\le 100$的OA$(N;t,k,v)$已经被完全构造[18],也可以用计算机自动搜索的方法(如文献[19,20])构造小规模的OA.在某些特殊情形下,OA就是最优的CA.例如OA$(v^t;t,t+1,v)$就是一个最优的覆盖数组;又如,如果$v\ge t-1$是一个素数的幂(prime power),那么OA$(v^t;t,v+1,v)$也是一个最优覆盖数组[21].

一般情况下,我们可以通过如下两步的坍塌(collapse)方法由一个OA$(N;t,k,v)$构造出一个MCA$(M;t,k,v_1v_2...v_k)$,其中$M\le N$并且$v_i\le v(0<i\le k)$[22].

i) 将第$i(0<i\le k)$列上在值域v_i以外的符号用值域内任意一个符号替换;

ii) 删除某些冗余的列(这些列包含的所有 t 元组已经被其他列覆盖).

另一种坍塌方法是利用大的覆盖数组构造小的覆盖数组.例如,我们可以分别利用MCA$(N;t,k,(v_1+1)v_2...v_k)$和MCA$(M;t,k,v_1v_2...v_k)$构造MCA$(N';t,k,v_1v_2...v_k)$和MCA$(M';t,k-1,v_2...v_k)$,其中$N'\le N,M'\le M$.坍塌方法的缺陷在于,最终得到的覆盖数组往往比最优解要大,尤其是对MCA[23].

另一种构造方法是用小的矩阵递归构造大的覆盖矩阵.例如,Stevens和Mendelsohn证明了,如果存在一个CA$(N;2,k,v)$实例$A=(a_{ij})$以及一个CA$(M;2,\ell,v)$实例$B=(b_{ij})$,那么我们可以构造一个CA$(N+M;2,k\ell,v)$的实例,如图1所示[24].

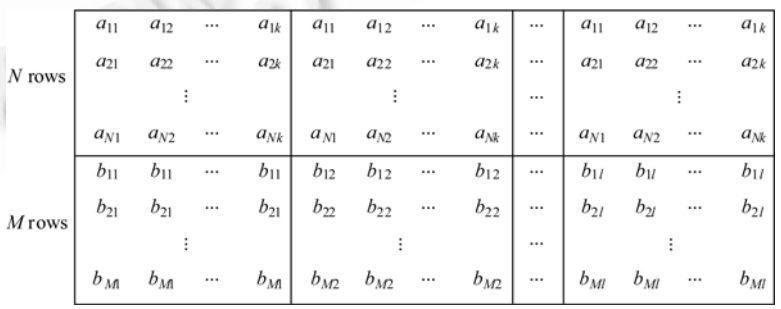

Fig.1 An instance of CA$(N+M;2,k\ell,v)$

图1 一个CA$(N+M;2,k\ell,v)$实例

递归构造方法在组合数学中是一种常见的研究方法.更多关于覆盖数组的递归构造方法,可以参见文献

[21,25–27].递归构造方法对于一些特殊的实例能够给出优美的结论.但是,对于一般的组合测试问题,这些方法并不具备通用性.

构造方法还有一些副产品,即给出了部分实例的覆盖数的范围.对于t=v=2 的情况,Katona[28],Kleitman和Spencer[29]分别于 1973 年证明了对于$\binom{N-2}{\lceil (N-1)/2\rceil}<k\le\binom{N-1}{\lceil N/2\rceil}$的CA,其覆盖数为$N$.那么对于所有的$t$=$v$=2的最优覆盖问题已经在理论上完全解决.对于这个问题,聂长海等人利用覆盖数组任意两列不相同、不互补等特性给出了一种构造算法,实验结果接近理论值[6].

对于其他形式的覆盖数组,目前的理论研究还无法精确地给出所有覆盖数的值.不少数学结论是关于一些不同覆盖数组实例的覆盖数之间的关系.例如,易知下列结论成立[26]:

$$\mathrm{MCAN}(t,k,v_1v_2...v_k)\ge v_1\cdot\mathrm{MCAN}(t-1,k-1,v_2...v_k),$$

又由以前结论可知, $\mathrm{CAN}(2,2^6)=6$ (t=v=2 的最优覆盖问题).那么,

$$\mathrm{MCAN}(3,3^1 2^6)\ge 3\cdot\mathrm{CAN}(2,2^6)=18,$$

这就证明了表 4 的测试用例集合是最优的.利用这些相互关系可以更一般地给出覆盖数的上、下界.例如Chateauneuf等人证明[30]了

$$\mathrm{CAN}(3,k,3)\le 6.1209(\log k)^2,$$

更多的结论可见文献[25],这里不再赘述.

目前已有一些采用数学方法构造覆盖数组的工具.Bell实验室的工具CATS[31]以及在线服务Testcover[32]采用递归构造和坍塌的方法构造覆盖数组.Williams等人曾经采用坍塌和递归构造的方法测试分布式系统[22],并开发了一个工具Tconfig[33].该工具的运行速度比较快,但是得到的覆盖数组往往比较大.IBM Haifa研究院的WHITCH项目(以前称为CTS项目)[34]中采用了一些递归构造方法,在部分实例上取得了很好的结果[35].

3 转化成其他问题

本文在第1节中指出,覆盖数组的构造问题的复杂度是NP完全的,目前已有一些NP完全问题的研究比较深入,如可满足问题、整数规划问题以及图上的一些问题.一些学者通过将覆盖数组的构造问题转化为其他类型的问题,然后借鉴这些问题的理论和方法来研究覆盖数组.

3.1 转化为可满足问题

可满足问题(SAT)[36]是指,给定一组布尔逻辑公式,是否存在一组对所有变量的赋值,使得这一组公式是可满足的.SAT是第一个被证明为NP完全的问题,近年来在国际上是一个比较热门的研究方向.理论上说,所有NP完全问题都可以转化为SAT问题求解.目前,SAT问题的求解算法主要包括两类,基于DPLL的穷举搜索算法和基于局部搜索的随机算法.其中前者是完备的,即一个问题是可满足的,算法就一定能够找到一组解;而后者对某些可满足的实例有较快的求解速度.关于SAT的求解算法和工具可以在相关网站[37]上找到.

Hnich等人[38]尝试将组合测试的判定问题的约束转化为SAT的子句(CNF形式)集,并采用基于局部搜索的SAT工具walksat来处理这个问题.其实验结果表明,walksat能够在数分钟内处理一些小规模的实例.由于walksat不是完备算法,采用这种方法生成的覆盖数组可能不是最优的.

严俊等人[39]采用与Hnich等人类似的SAT编码方法,并且尝试使用另一个SAT工具zChaff(这是最高效的完备求解工具之一)来处理这个问题.一般来说,直接编码得到的SAT子句集合zChaff处理起来比较困难.为了减小搜索空间,该方法使用了工具Shatter来打破SAT子句中的对称性,然后调用zChaff处理.但实验结果却令人失望.即使对一些较小的实例, zChaff也要花费大量的时间.

SAT方法的主要问题在于,即使是针对一个小规模覆盖数组问题转化得来的SAT子句集合规模也可能是非常庞大的.另外,在翻译过程中,一些覆盖数组的组合特性(如对称性)也丢失了,简单地调用SAT工具的求解效率并不高.改进SAT翻译方法以及SAT工具调用的方式(如文献[20]采用的将SAT与回溯算法结合的方式)是较为可

行的研究方向.

3.2 转化为整数规划问题

Cheng等人指出,一个一般的组合测试问题(包括非经典的组合测试问题)可以形式化为一个集合覆盖问题(set-covering problem)[15].一个集合覆盖问题包括一个有限集合 $X=\{x_1,x_2,...,x_m\}$(这里指的是所有可能的参数值组合)以及由X的子集组成的集合 $F=\{f_1,f_2,...,f_n\}$(这里指的是所有测试用例集合,每个测试用例对应着若干参数值组合).集合覆盖问题的目标是寻找F的一个最小子集C,使得C覆盖X中的所有元素.

Williams等人采用0-1整数规划的方法求解这个问题[40].令二值变量y_i=1表示$f_i \in C$.假定F中的元素 $f_{j_1},f_{j_2},...,f_{j_s}$ 包含x_j,则寻找最小集合C的问题可以描述为最优化问题:

$$\begin{aligned} &\min \quad y_1+y_2+...+y_m \\ &\text{s.t.} \quad \text{for all } j,\ y_{j_1}+y_{j_2}+...+y_{j_s}\geq 1. \end{aligned}$$

与SAT方法类似,由于集合X的规模很大,采用0-1规划虽然能够得到最优解,但是能够处理问题的规模却很小.

3.3 转化为图问题

前面我们在第1.2.4节中提到,可以用图染色的方法来约简基于I/O关系的组合测试.此外,加拿大Ottawa大学的研究小组将成对覆盖数组(包括非经典的覆盖数组)转化为图,然后利用图的理论研究覆盖数组的性质.他们的研究方法是,首先定义属性无关(qualitatively independent,简称QI)的概念.所谓属性无关,指的是两个N维向量,两个向量的值域分别为 $A\in\mathbb{Z}_u^N, B\in\mathbb{Z}_v^N$,则对于任意 $(\alpha,\beta)\in\mathbb{Z}_u\times\mathbb{Z}_v$,总存在一个整数$i$,使得$A$,$B$的第$i$维的数对 $(a_i,b_i)=(\alpha,\beta)$.接下来,我们定义图上的覆盖数组(covering array on graph)CA(N,G,v)是一个具有k个顶点的图,每个顶点代表一个N维向量(即覆盖数组中的一列).两个顶点之间有边当且仅当对应覆盖矩阵的两列是QI的.显然,标准的成对覆盖数组CA(N;2,k,v)对应的图是一个k完全图(complete graph),而王子元等人的相邻因素覆盖数组[16]对应的图是一个长为k的链.最后定义图同态(graph homomorphism)这一概念.如果从图G到H存在一个映射φ,使得对于任意相邻的顶点$x,y\in V(G)$,$\varphi(x)$和$\varphi(y)$在H上也是相邻的,那么图G和H是同态的.利用图同态的性质,我们可以给出覆盖数的范围.Meagher采用这种方法研究了成对覆盖数组[41]和混合覆盖数组[42]的一些性质,Zekaoui研究了一些非经典覆盖数组的性质[43].

4 直接搜索方法

组合测试的很大一部分研究是利用传统的约束求解或者最优化方法来直接搜索覆盖数组.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.另外,有少量的研究采用全局搜索算法,能够对一定规模的问题得到最优解.直接搜索方法中有不少算法已经形成了实用的工具,关于这些工具的介绍可以参见文献[44].

4.1 贪心算法

贪心算法的思想是从空矩阵开始,逐行或者逐列扩展矩阵,直到所有的t组合都被覆盖.按照扩展方式的不同,可以分成一维扩展和二维扩展两类,另外还有一些方法将其他算法与贪心算法结合起来使用.

4.1.1 一维扩展

我们主要介绍逐行扩展的一维扩展方法,所谓的one-test-at-a-time方法,即在构造覆盖数组时,按照贪心策略依次增加一行,使得这一行覆盖一些未覆盖的t元组,直到所有的t元组都被覆盖.最直接的贪心策略就是,每次选择的新测试用例覆盖最多的未覆盖t元组.Cohen等人证明,对于成对测试CA(N;2,k,v),采用这种一维扩张策略产生的贪心测试集大小与SUT参数的个数呈对数关系($O(v^2\log k)$)[11].但是,枚举所有可能的测试用例是不现实的(复杂度随着参数个数的增加呈指数增长),因而,大部分的贪心策略都是从一个较小的测试用例集合(称为候选测试用例,candidates)中选择下一个测试用例.一维扩张这个算法框架最初由商业工具AETG[11]提出来.AETG的贪心策略如下:

i) 首先随机选择一些(50个左右)候选测试用例.候选用例的选择方法是,随机指定一个参数(也就是矩阵的列)的次序,然后按照这个次序依次给每个参数赋值.赋值的策略是,新的赋值和用例中已有的赋值能够覆盖最多的t元组.

ii) 然后从这些测试用例中选择一个覆盖了最多的未覆盖t元组作为数组的下一行.

AETG的贪心策略是非确定的,所以多次运行AETG的结果可能不同.TCG[45]也是从若干候选中选择最好的一行.与AETG不同,TCG在对参数排序的时候是按照参数的值域从大到小的次序使 $v_1 \geq v_2 \geq ... \geq v_k$,然后依次给第1个参数赋值为1,2,…,v_1,最后按照与AETG相同的方式产生v_1个候选用例.因而,TCG的贪心策略是确定的.

微软开发的工具PICT采用类似AETG的方法选择候选测试用例.与AETG的不同之处在于,PICT不产生固定大小的候选测试用例集(或者说只选择大小为 1 的候选测试用例),并且总是采用固定的随机种子,所以PICT的贪心策略实际上是确定性的.PICT目前仅能用于成对测试中[12].

AETG框架存在一个不足之处是,在给某一项赋值时,采用的贪心策略是新的赋值和当前测试用例(行)中"已有的赋值"能覆盖最多的t元组,而没有考虑未赋值的项.鉴于此,Bryce等人提出了DDA[23]算法.该算法采用了所谓的tie-breaking的确定型贪心策略,给每个项以及可能的赋值定义了一个密度(density)值.通过计算密度选择最佳赋值.目前DDA算法也只能应用在成对测试中.

在处理成对测试这一类特殊问题时,史亮等人提出了一个解空间树(solution space tree)的模型,这样,测试集合中的每个元素对应解空间树的一条从根到叶节点的路径.那么,根据 2 组合测试的性质,可以选择的贪心策略是,新加入的用例与测试集已有元素的重叠数不超过 2(这里,重叠数的定义为两个测试用例中取值相同的参数个数.例如,(1,2,3)与(1,2,1)在前两个参数的取值相同,所以重叠数为 2).史亮等人的实验结果表明,该方法对于部分实例能够给出最优的结果[46].

4.1.2 二维扩展

Lei等人提出的IPO(in-parameter-order)[9]是另外一种类型的贪心算法.该算法主要针对成对测试.首先构造前 2 个参数的所有组合,形成一个较小的矩阵,然后重复如下两步的二维扩展:

i) 水平扩展(horizontal growth):在矩阵中增加一个参数(列),给这一列上的每一项赋值,尽可能覆盖更多的t元组;

ii) 垂直扩展(vertical growth):在水平扩展的基础上,如果有t元组没有覆盖,则增加新的测试用例(行).

图2是IPO算法处理实例 MCA(6;2,$2^2$3^1) 的过程.首先构造前两个参数A和B的所有取值组合,共4个.然后水平扩展到第3列.依次选择第3列的值为C1,C2,C3,C1能够覆盖尽可能多的新2元组.这时,算法发现还有(A1,-,C3),(A2,-,C2),(-,B1,C2),(-,B2,C3)这些2元组未覆盖.为了使用最少的垂直扩展,IPO算法将(A2,-,C2)与(-,B1,C2),以及(A1,-,C3)与(-,B2,C3)分别合并,于是通过垂直扩展两行得到了最后的结果.

A	B
A1	B1
A1	B2
A2	B1
A2	B2

A	B	C
A1	B1	C1
A1	B2	C2
A2	B1	C3
A2	B2	C1

Horizontal growth

A	B	C
A1	B1	C1
A1	B2	C2
A2	B1	C3
A2	B2	C1
A2	B1	C2
A1	B2	C3

Vertical growth

Fig.2 Horizontal and vertical growth of IPO algorithm

图2 IPO算法的水平扩展与垂直扩展

最初的IPO算法只能产生成对覆盖数组.后来,Lei等人将这种算法扩展,得到了一种能产生任意t组合测试用例的算法IPOG,并开发了一个工具FireEye[47].IPOG算法虽然是贪心算法,但它采用了指数级复杂度的方法来选择最少行进行垂直扩展,故在处理大规模实例时仍然需要花费大量时间.

4.1.3 其他贪心算法

对大规模的难解问题,随机型的算法是不错的选择.Kuhn提出的Paintball[48]采用随机策略生成大量的测试用例,然后采用贪心算法补足未覆盖的t元组,最后合并一些冗余的行.这种算法在处理大规模问题时相对于

IPOG体现出了其速度快的优势.

相对于其他搜索方法,贪心算法的处理速度是最快的.并且由于贪心算法的简单、有效性,很多贪心算法,如AETG和DDA,都能支持部分非经典的组合测试方法.贪心算法的不足之处在于,贪心算法本身是一种近似算法,无法保证解的最优性.

4.2 启发式搜索算法

贪心算法的基本思想是从小到大构造矩阵,直到所有的覆盖条件都得到满足.另外一种寻找最优的(或者较优的)覆盖数组的方法是,利用一个已有的数组,通过合适的变换得到一个更优的覆盖矩阵.这样,通过逐次变换得到较优的矩阵.为了避免陷入局部最优,算法采用了一些启发式的策略,通过多样化或变异的方法跳出局部最优点.这里说的一些启发式算法,主要是指近年来流行的模拟自然界行为的智能优化算法.目前已经应用到组合测试中的算法主要包括下面几种:

模拟退火(simulated annealing)算法的主要原理是模拟统计物理中固体物质的结晶过程.Cohen等人[5]给出的模拟退火算法需要预先给出覆盖矩阵的大小,然后随机产生一些矩阵.每个矩阵S的评价函数$c(S)$取决于矩阵中未覆盖的t元组数,矩阵变换的方法随机改变矩阵中的某些位.在退火的过程中以概率$e^{-(c(S')-c(S))/KT}$接受不好的解.其中,$c(S')$和$c(S)$分别表示变换后和变换前矩阵的评价,K是常数.T表示控制温度,按照下一时刻温度$T'=\alpha T$模拟降温过程,这里,系数α略小于1,用于控制降温的速度.Shiba等人的实验指出,模拟退火算法在启发式算法中能够得到最优的结果[49],但同时花费的时间也比较长.

禁忌搜索(tabu search)的原理是模拟人的思维,通过禁忌表记忆最近搜索过程的若干步,以避免走回头路,达到跳出局部最优解的目的.Nurmela[50]采用的禁忌搜索算法与模拟退火算法类似,需要预先给定覆盖矩阵的大小N,然后随机选择初始矩阵.每个矩阵的评价函数就是未覆盖的t元组数目.矩阵变换方法就是改变矩阵中的某一项,使得评价函数变小.搜索的每一步就是选择不在禁忌表中的使评价函数变得最小的一个变换.Nurmela采用的禁忌表大小T满足$1\le T\le 10$.

遗传算法(generic algorithm)是根据生物演化,模拟演化过程中基因染色体的选择、交叉和变异得到的算法.在进化过程中,较好的个体有较大的生存几率.蚁群算法(ant colony algorithm)的主要原理是模拟蚁群的集体行为,寻找最优的路线.Shiba等人[49]给出了这两种算法求解覆盖数组问题的方法.与模拟退火和禁忌搜索方法不同,他们的算法使用one-test-at-a-time策略,即每次通过搜索得到一条较好的测试用例,直至所有组合均被覆盖.此外,还可以通过一种压缩算法(compaction algorithm)来对覆盖数组加以改进.具体而言,就是覆盖矩阵中有些项可以随便改写而不影响整个数组对t元组的覆盖,记作DC(don't care)位.如果两行除了DC位,其他位都相同,那么两行可以合并.例如,如果两行为(3,2,1,X)和(3,X,X,1)(其中X代表DC位),那么它们可以合并为(3,2,1,1),即覆盖矩阵可以减少一行.Shiba等人的实验结果表明,遗传算法和蚁群算法的实验结果优于AETG和IPO这两种贪心算法.

相对于贪心算法,启发式搜索算法能够给出较优的近似结果.但是,大部分启发式算法需要同时变换多个矩阵实例,进行多次变换,因而运行时间一般比较长.目前,启发式算法主要停留在研究阶段,很少有实用的工具采用启发式搜索算法[44].

4.3 完备搜索算法

对于覆盖数组的判定问题,即某个实例,是否存在大小为N的覆盖数组,我们可以把它看成是一个变量数为$N\times k$的约束满足问题(constraint satisfaction problem,简称CSP),从而利用约束求解的方法来搜索覆盖矩阵.

Hnich等人首先建立了覆盖数组的约束模型,并在商用的约束逻辑程序设计(constraint programming)平台ILOG上求解这个问题[38].这种方法采用了部分对称打破(symmetry-breaking)技术.所谓对称打破,指的是如果解空间中有两个对称的子空间**A**和**B**(也就是说,**A**或者**B**可以通过一个同构变换得到另外一个空间),那么算法只需搜索其中一个就可以了.例如在搜索覆盖矩阵时,考虑任意两行l_1和l_2,令空间**A**表示$l_1\le l_2$的解空间,空间**B**表示$l_1\ge l_2$的解空间(这里,两个行向量按照字典序进行比较).**A**和**B**是对称的,因为对于**A**中的任意一个覆盖数组,交换

l_1和l_2这两行就成为**B**中的一个解,反之亦然.于是在搜索过程中,搜索完**A**就不必再搜索**B**,从而减少了搜索空间.Hnich等人的实验结果表明,这种方法在部分小规模问题上能够得到最优的覆盖数组,但是程序的性能随着问题规模的增加急剧下降.

严俊等人采用完备回溯搜索的方法[39]来求解覆盖数组的判定问题.这种方法采用了更多的对称打破技术,并引入了约束求解中的其他技术,如剪枝(pruning)、约束传播(constraint propagation)等.采用这些技术实现的工具EXACT[19]在很多小规模问题上能够在可接受的时间范围内得到最优解.

完备搜索的最大问题在于问题的复杂性太大,搜索 MCA($M;t,k,v_1v_2...v_k$) 的解空间大小为 $\prod_{i=1}^{k}v_i^N$.压缩搜索空间的策略对算法的性能影响很大.总体来说,这一类方法处理的问题规模都不大,而若处理大规模实例,则需要与其他方法,如递归构造方法结合起来使用.

5 基于组合测试的错误定位技术

当错误被检出后,如何利用运行失败的测试用例进行错误定位是软件测试过程中必不可少的一个环节.Yilmaz等人采用分类树(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)[51].该项工作主要分析了与系统配置相关的错误(option-related failures),即在测试执行过程中出现超过3%的错误.Yilmaz等人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.

但是,相对于程序所有参数取值的t组合,组合测试用例的数量要小很多.这样,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)很难被精确地确定.例如,假定用户采用本文表2中的测试用例集来测试表1描述的系统,如果测试用例1执行时出现错误,而其他测试用例的执行均正常.我们假定程序的错误是有两个参数相互作用产生的.那么,程序的所有可能的错误模式有:(Firefox,WebSphere,-,-),(Firefox,-,MasterCard,-),(Firefox,-,-,DB/2),(-,WebSphere,MasterCard,-),(-,WebSphere,-,DB/2)以及(-,-,MasterCard,DB/2),共6种.采用组合测试无法进一步确定错误发生的具体原因.徐宝文等人首先假定“若某个模式是错误模式,则任意包含该模式的其他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[52].

6 总结与展望

组合测试是近年来比较活跃的研究方向,学术界对组合测试技术的研究取得了一系列的成果.同时,组合测试也在工业界取得了广泛的应用,如本文中指出, IBM、微软、Bell实验室等国际知名企业都开发了组合测试的工具.组合测试已被软件业公认为一种行之有效的测试方法.但是另一方面,也有学者对组合测试提出了不同的看法.例如,Schroeder等人利用工具AETG产生测试用例来研究组合测试的检错能力.其实验结果表明,组合测试的检错能力与随机测试的检错能力相当[53](从该文的实验数据中可以看出,作为对比的随机测试用例集的组合覆盖率基本上也都接近100%,这可能是产生这个结论的原因).Bach等人进一步指出,组合测试被过分高估了(“the technique to be over promoted and poorly understood”),并不是任何系统都适合采用组合测试方法,或者说,简单地采用组合测试并不是放之四海而皆准的测试方法[54].

未来的研究包括以下几个方面:1) 针对被测系统的特性或者具体的测试场景,设计针对性的组合覆盖准则.2) 拓展组合测试的应用范围.例如最初的组合测试方法主要应用于黑盒测试,但是,近年来也有一些工作将组合测试应用到其他方面,如文献[17]指出,可以用非经典组合测试方法测试面向对象系统中的类.未来的组合测试方法可能被扩展到面向代码的程序测试中.3) 在测试生成算法方面,工业界的工具普遍采用速度较快的贪心算法,从已经公开发表的实验结果来看,贪心算法产生的许多测试集的大小与最优值之间还有较大的差距.而学术界提出的众多算法在性能上尚不尽如人意,处理较大规模的实例花费时间较多.实用的算法需要在性能和

精确度上作进一步的改进.4) 其他与组合测试相关的问题.例如,组合测试具有较高的错误检出率,但是这些结论是从实验中得出的.那么,如何建立可靠性模型定量地描述组合测试对软件可靠性的影响,是一个有意义的研究方向.

References:

[1] Kuhn DR, Reilly MJ. An investigation of the applicability of design of experiments to software testing. In: Caulfield M, ed. Proc. of the Annual NASA/IEEE Software Engineering Workshop (SEW). Los Alamitos: IEEE Press, 2002. 91−95.

[2] Godbole AP, Skipper DE, Sunley RA. *t*-Covering arrays: Upper bounds and Poisson approximations. Combinatorics, Probability and Computing, 1996,5:105−118.

[3] Kuhn R, Lei Y, Kacker R. Practical combinatorial testing: Beyond pairwise. IT Professional, 2008,10(3):19−23.

[4] Grindal M. Handling combinatorial explosion in software testing [Ph.D. Thesis]. Linköpings Universitet, 2007.

[5] Cohen MB, Gibbons PB, Mugridge WB, Colbourn CJ. Constructing test suites for interaction testing. In: Dillon L, Tichy W, eds. Proc. of the Int'l Conf. on Software Engineering (ICSE). Loa Alamitos: IEEE Press, 2003. 38−48.

[6] Nie CH, Xu BW, Shi L. 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 with English abstract).

[7] Colbourn CJ. CA tables for *t*= 2,3,4,5,6. 2009. http://www.public.asu.edu/~ccolbou/src/tabby/catable.html

[8] Seroussi G, Bshouty NH. Vector sets for exhaustive testing of logical circuits. IEEE Trans. on Information Theory, 1988,34(3): 513−522.

[9] Lei Y, Tai KC. In-Parameter-Order: A test generation strategy for pairwise testing. In: Tsai J, Keefe T, Stewart D, eds. Proc. of the IEEE Int'l. Symp. on High Assurance Systems Engineering. Los Alamitos: IEEE Press, 1998. 254−261.

[10] Dalal SR, Jain A, Karunanithi N, Leaton JM, Lott CM, Patton GC, Horowitz BM. Model-Based testing in practice. In: Boehm B, Garlan D, Kramer J, eds. Proc. of the Int'l Conf. on Software Engineering (ICSE). New York: ACM Press, 1999. 285−294.

[11] Cohen DM, Dalal SR, Fredman ML, Patton G. The AETG system: An approach to testing based on combinatorial design. IEEE Trans. on Software Engineering, 1997,23(7):437−443.

[12] Czerwonka J. Pairwise testing in real world: Practical extensions to test case generators. In: Butt D, Gens C, eds. Proc. of the 24th Pacific Northwest Software Quality Conf. 2006. http://www.pnsqc.org/proceedings/pnsqc2006.pdf

[13] Cohen MB, Gibbons PB, Mugridge WB, Colbourn CJ. Variable strength interaction testing of components. In: Bae D, Voas J, eds. Proc. of the IEEE Annual Int'l Computer Software and Applications Conf. (COMPSAC). Los Alamitos: IEEE Press, 2003. 413−418.

[14] Schroeder PJ, Korel B. Black-Box test reduction using input-output analysis. In: Harold MJ, ed. Proc. of the Int'l Symp. on Software Testing and Analysis (ISSTA). New York: ACM Press, 2000. 173−177.

[15] Cheng C, Dumitrescu A, Schroeder PJ. Generating small combinatorial test suites to cover input-output relationships. In: Lin H, Ehrich HD, eds. Proc. of the 3rd Int'l Conf. on Quality Software (QSIC). Los Alamitos: IEEE Press, 2003. 76−82.

[16] Wang ZY, Nie CH, Xu BW, Shi L. Optimal test suite generation methods for neighbor factors combinatorial testing. Chinese Journal of Computers, 2007,30(2):200−211 (in Chinese with English abstract).

[17] Wang ZY, Nie CH, Xu BW. Generating combinatorial test suite for interaction relationship. In: Pezzè M, ed. Proc. of the 4th Int'l Workshop on Software Quality Assurance (SOQUA 2007). New York: ACM Press, 2007. 55−61.

[18] Sloane NJA. A library of orthogonal arrays. http://www.research.att.com/~njas/oadir/index.html

[19] Yan J, Zhang J. A backtracking search tool for constructing combinatorial test suites. The Journal of Systems and Software, 2008, 81(10):1681−1693.

[20] Ma F, Zhang J. Finding orthogonal arrays using satisfiability checkers and symmetry breaking constraints. In: Ho TB, Zhou ZH, eds. Proc. of the 10th Pacific Rim Int'l Conf. on Artificial Intelligence (PRICAI 2008). LNAI 5351, Berlin/Heidelberg: Springer-Verlag, 2008. 247−259.

[21] Chateauneuf M, Kreher D. On the state of strength-three covering arrays. Journal of Combinatorial Designs, 2002,10(4):217−238.

[22] Williams AW, Probert RL. A practical strategy for testing pair-wise coverage of network interfaces. In: Lyu MR, ed. Proc. of the 7th Int'l Symp. on Software Reliability Engineering (ISSRE). Los Alamitos: IEEE Press, 1996. 246−254.

[23] Bryce RC, Colbourn CJ. The density algorithm for pairwise interaction testing. Software Testing, Verification and Reliability, 2007. 17(3):159−182.

[24] Stevens B, Mendelsohn E. New recursive methods for transversal covers. Journal of Combinatorial Designs, 1999,7(3):185−203.

[25] Colbourn CJ, Dinitz JH. Handbook of combinatorial designs. 2nd ed., Discrete Mathematics and Its Applications Series, Boca Raton: Chapman & Hall/CRC, 2006.

[26] Colbourn CJ, Martirosyan SS, Mullen GL, Shasha D, Sherwood GB, Yucas JL. Products of mixed covering arrays of strength two. Journal of Combinatorial Designs, 2006,14(2):124−138.

[27] Martirosyan S, Trung TV. On *t*-covering arrays. Designs, Codes and Cryptography, 2004,32(1):323−339.

[28] Katona GOH. Two applications (for search theory and truth functions) of Sperner type theorems. Periodica Mathematica. Hungarica, 1973,3(1-2):19−26.

[29] Kleitman DJ, Spencer J. Families of *k*-independent sets. Discrete Mathematics, 1973,6:255−262.

[30] Chateauneuf MA, Colbourn CJ, Kreher DL. Covering arrays of strength three, Designs, Codes Cryptography, 1999,16(3):235−242.

[31] Sherwood G. On the construction of orthogonal arrays and covering arrays using permutation groups. http://home.att.net/~gsherwood/cover.htm

[32] http://testcover.com/.2009

[33] Williams AW. TConfig: Test configuration generator. 2009. http://www.site.uottawa.ca/~awilliam/TConfig.jar

[34] IBM Intelligent Test Case Handler. 2009. http://alphaworks.ibm.com/tech/whitch

[35] Hartman A, Raskin L. Problems and algorithms for covering arrays. Discrete Mathematics, 2004,284(1-3):149−156.

[36] Zhang J. Deciding Satisfiability of Logic Formulae: Methods, Tools and Applications. Beijing: Science Press, 2000 (in Chinese).

[37] SAT Live! 2009, http://www.satlive.org/

[38] Hnich B, Prestwich S, Selensky E. Constraint-Based approaches to the covering test problem. In: Faltings B, Petcu A, Fages F, Rossi F, eds. Proc. of the Joint Annual Workshop of ERCIM/CoLogNet on Constraint Solving and Constraint Logic Programming (CSCLP). Berlin/Heidelberg: Springer-Verlag, 2004. 172−186.

[39] Yan J, Zhang J. Backtracking algorithms and search heuristics to generate test suites for combinatorial testing. In: Wong J, ed. Proc. of the IEEE Annual Int'l Computer Software and Applications Conf. (COMPSAC). Los Alamitos: IEEE Press, 2006. 385−394.

[40] Williams AW, Probert RL. Formulation of the interaction test coverage problem as an integer program. In: Schieferdecker I, König H, Wolisz A, eds. Proc. of the 14th Int'l Conf. on the Testing of Communicating Systems (TestCom). Berlin: Kluwer, 2002. 283−298.

[41] Meagher K, Stevens B. Covering arrays on graphs. Journal of Combinatorial Theory (Series B), 2005,95(1):134−151.

[42] Meagher K, Moura L, Zekaoui L. Mixed covering arrays on graphs. Journal of Combinatorial Designs, 2007,15(5):393−404.

[43] Zekaoui L. Mixed covering arrays on graphs and tabu search algorithms [MS. Thesis]. Ottawa-Carleton Institute for Computer Science, University of Ottawa, 2006.

[44] Czerwonka J. Pairwise Testing. 2009. http://www.pairwise.org/

[45] Tung YW, Aldiwan WS. Automating test case generation for the new generation mission software system. In: Proc. of the IEEE Aerospace Conf. 2000. 431−437.

[46] Shi L, Nie CH, Xu BW. Pairwise test data generation based on solution space tree. Chinese Journal of Computers, 2006,29(6): 849−857 (in Chinese with English abstract).

[47] Lei Y, Kacker R, Kuhn DR, Okun V, Lawrence J. IPOG: A general strategy for *t*-way software testing. In: Leaney J, O'Neill T, Peng J, eds. Proc. of the Annual IEEE Int'l Conf. and Workshops on the Engineering of Computer-Based Systems (ECBS). Los Alamitos: IEEE Press, 2007. 549−556.

[48] Kuhn R. An algorithm for generating very large covering arrays. Technical Report, NISTIR7308, 2006.

[49] Shiba T, Tsuchiya T, Kikuno T. Using artificial life techniques to generate test cases for combinatorial testing. In: Wong WE, Kanoun K, eds. Proc. of the IEEE Annual Int'l Computer Software and Applications Conf. (COMPSAC). Alamitos: IEEE Press, 2004. 72−77.

[50] Nurmela KJ. Upper bounds for covering arrays by tabu search. Discrete Applied Mathematics, 2004,138(9):143−152.

[51] Yilmaz C, Cohen MB, Porter MB. Covering arrays for efficient fault characterization in complex configuration spaces. IEEE Trans. on Software Engineering, 2006,32(1):20−34.

[52] Xu BW, Nie CH, Shi L, Chen HW. A software failure debugging method based on combinatorial design approach for testing. Chinese Journal of Computers, 2006,29(1):132−138 (in Chinese with English abstract).

[53] Schroeder PJ, Bolaki P, Gopu V. Comparing the fault detection effectiveness of *n*-way and random test suites. In: Juristo N, Shull F, eds. Proc. of the Int'l Symp. on Empirical Software Engineering (ISESE). Alamitos: IEEE Press, 2004. 49−59.

[54] Bach J, Shroeder P. Pairwise testing—A best practice that isn't. In: Butt D, Derby E, eds. Proc. of the 22nd Pacific Northwest Software Quality Conf. 2004. 180−196. http://www.pnsqc.org/proceedings/pnsqc2004.pdf

附中文参考文献:

[6] 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.

[16] 王子元,聂长海,徐宝文,史亮.相邻因素组合测试用例集的最优生成方法.计算机学报,2007,30(2):200−211.

[36] 张健.逻辑公式的可满足性判定——方法、工具及应用.北京:科学出版社,2000.

[46] 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.

[52] 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.

严俊(1980−),男,湖北石首人,博士,助理研究员,主要研究领域为程序分析,软件测试.

张健(1969−),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为自动推理,约束求解,形式化方法,软件测试.