组合测试用例集的动态生成算法

崔应霞，李龙澍，姚 晟

(安徽大学计算机科学与技术学院 合肥 230039)

【摘要】在逐因素扩展算法的基础上，提出了一种有效的组合测试用例生成算法IPO_T。该算法根据已被扩展的测试用例覆盖$t(t\geq 2)$参数值组合的情况确定水平扩展方法，并依据新的被覆盖的t参数值组合修改已被扩展的部分测试用例，从而达到对测试用例集的优化。设计实现了基于该方法的测试用例生成工具。通过和部分现有的支持t维组合测试工具比较，IPO_T在合理的时间内产生的测试用例集较小。

关 键 词 组合测试； 动态生成； 水平扩展； 测试用例生成

中图分类号 TP311.5 **文献标识码** A doi:10.3969/j.issn.1001-0548.2011.03.028

Dynamic Generation Algorithm of Combinatorial Test Suite

CUI Ying-xia, LI Long-shu, and YAO Sheng

(School of Computer Science and Technology, Anhui University Hefei 230039)

Abstract An effective combinatorial test case generation algorithm called in parameter order_T-way (IPO_T) is proposed based on in parameter order (IPO) algorithm. In order to achieve the optimization of test suite, IPO_T algorithm determines the method of horizontal growth according to t-way($t\geqslant 2$) combinations that are covered by the extended test cases, and revises extended test cases depending on the new covered t-way combinations. A test case generation tool is designed and implemented with this approach, and compared with some existing tools. Experimental results show that IPO_T outperforms them in terms of the number of generated test case within reasonable execution times.

Key words combinatorial test; dynamic generation; horizontal growth; test case generation

软件测试贯穿于软件定义与开发的整个周期，其核心任务是测试用例的生成，并占据整个周期工作量的50%以上。因为不同的测试用例具有不同的检错能力，加上时间和资源的限制以致不可能执行所有的测试用例，所以如何从大量的测试用例中选择具有代表性的测试用例，使得它们具有较高的检错能力是软件测试中的关键课题。组合测试充分考虑了系统中各种因素以及各种因素之间的相互作用可能对系统产生的影响，力求用尽可能少的测试用例覆盖尽可能多的影响因素，是一个在测试效果和时间上有着很好的折中的测试方法。根据覆盖程度不同，组合测试分为单因素组合测试、成对组合测试和t维($t\geq 2$)组合测试。组合测试的早期研究主要集中在成对组合测试用例生成方面，并证实它可以检测出待测系统的大部分错误。但是随着研究的深入，研究者发现该结论不能适用于所有的软件系统。如美国国家标准和技术协会(the national institute of standards and technology，NIST)一项调查显示，4维组合测试的检错率为95%，6维组合测试的检错率才将近100%[1]。故2维组合测试(成对组合测试)是不完全的，必须进行更高维的组合测试。

为一个待测系统构造一个最小t维组合测试用例集是NP问题[2]，目前关于组合测试用例集的生成方法都是近似求解。基于该情况，本文在逐因素扩展算法(in parmeter order，IPO)框架的基础上，提出了一种新的组合测试用例生成方法(in parameter order-T-wag，IPO_T)，该方法利用对测试用例集中已确定的测试用例进行修改这一动态特性，实现对测试用例集的进一步优化。本文还设计基于该方法的测试用例生成工具，通过和部分支持t维组合测试工具比较，实验结果表明，多数情况下，IPO_T在合理的时间内产生的测试用例集较小。

收稿日期：2009－12－01； 修回日期：2010－06－22
基金项目：安徽省自然科学基金(090412054)；安徽省科技攻关计划重大科技专项(08010201002)；安徽省教育厅重点项目(KJ2009A001Z)
作者简介：崔应霞(1984－)，女，博士，主要从事程序切片、软件测试方面的研究.

1 组合测试原理

设有影响待测系统S的输入参数(因素)有p_1, $p_2,\cdots,p_n$共n个，p_{iv}是p_i的一个合理取值，$|p_i|$是p_i的合理取值数。

定义 1 设t-parameter是t参数组合集，则t-parameter = $\{(p_{i1}, p_{i2},\cdots, p_{it})|1\leqslant ij\leqslant n\}$。

定义 2 设t-value是t元参数值组合集，则t-value=$\{(p_{i1.v1},p_{i2.v2},\cdots, p_{it\ vt})|1\leqslant ij\leqslant n\}$，$t$参数值的组合也称$t$元组。

定义 3 设$\boldsymbol{T}$是t维组合测试用例集，则$\boldsymbol{T}$是t维组合测试用例集的必要条件是任意t个输入参数的t-value中的每个元素至少被一个测试用例覆盖。$\boldsymbol{T}$可以看作是一个$n\times m$的矩阵，其中每列表示一个输入参数，每行表示一个测试用例。

为了阐明t维组合测试的概念，下面以一个具有3个输入参数p_1、p_2和p_3的系统S_1为例。p_1的取值域为{0,1,2}，p_2和p_3的取值域为{0,1}。S_1的2-parameter=$\{(p_1, p_2),(p_1, p_3),(p_2, p_3)\}$，有$C_3^2$个。$(p_1,p_2)$和$(p_1, p_3)$的2-value={(0,0),(0,1),(1,0),(1,1),(2,0), (2,1)}，有3×2个；(p_2, p_3)的2-value= {(0,0),(0,1),(1,0), (1,1)}，有2×2个。图1显示了待测系统S_1的成对组合测试用例集，其中三列依次为p_1、p_2和p_3，每行表示一个测试用例，共有6个测试用例。可以看出，该测试用例集覆盖了S_1的所有2维参数值组合。

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 2 & 0 & 0 \\ 2 & 1 & 1 \end{pmatrix}$$

图1　系统S_1的成对组合测试用例

2 逐因素扩展算法

文献[3]首次提出组合测试，经过20多年的研究，出现了许多不同的组合测试方法。其中，基于贪心算法的组合测试方法从空矩阵开始逐行或逐列扩展矩阵，直到所有的t维参数值的组合都被覆盖。按照扩展方式的不同分为一维扩展和二维扩展。一维扩展是逐行扩展，每次选择新的测试用例覆盖最多的未被覆盖的t参数值组合。该方法首先由文献[4]提出，并开发了工具AETG。二维扩展是逐列扩展，又称逐因素扩展IPO[5-6]，和AETG不同，每次迭代不是产生矩阵的一行，而是一列，它又分为水平扩展和垂直扩展。基于启发式搜索算法的组合测试方法(模拟退火(simulated annealing)[7]、禁忌搜索(tabu search)[8]和遗传算法(generic algorithm)[9])：利用一个已有的矩阵，通过合适的变换得到一个更优的覆盖矩阵。数学构造方法直接或递归地使用某些代数结构构造规模较小甚至最小的组合测试用例集[2]。但是，在大多数情况下，需要所有的输入参数取值数相等或者输入参数个数为素数，因此其通用性受到严重影响，难以进行扩展。

2.1 IPO算法

IPO算法最初由文献[5]提出，用于成对测试用例的生成，后来改进为t维组合测试用例的生成，证明IPO不仅在生成的测试用例集的大小上还是在时间的花费上和其他工具相比都有其优越性[10]。IPO算法的主要思想是：先把待测系统的所有输入参数按取值域大小非递增排列；然后对前t个参数进行全组合生成一个初始的测试用例集；最后进行$n–t+1$次迭代，依次把后$n–t+1$个输入参数加入测试用例集，并使其满足t覆盖；在对后$n–t+1$个参数进行迭代时，通过对测试用例集的水平扩展和垂直扩展完成。

水平扩展：在$\boldsymbol{T}$中增加一列(输入参数)，为这一列上的每一项赋值，尽可能覆盖更多没有被覆盖的t参数值的组合。

垂直扩展：经过水平扩展之后，如果有t参数值的组合没有被覆盖，则通过改变现有的测试用例或者增加新的测试用例来覆盖。水平扩展的结果直接决定着垂直扩展和最后的测试用例集的大小。

2.2 IPO_T算法

IPO_T算法框架和IPO算法框架都是水平扩展和垂直扩展，但它们有不同的扩展算法。IPO的水平扩展算法是添加新的参数值，尽可能覆盖更多的没有被覆盖的t参数值的组合，而IPO_T算法是依据正在扩展的测试用例与已被扩展的测试用例关于t维参数值的覆盖情况而确定。IPO_T算法和其他组合测试算法的不同之处不仅在于扩展算法上，更在于在不影响覆盖率的情况下，它会根据新扩展的测试用例和部分已被扩展的测试用例是否包含了某些已被扩展的测试用例的所有t维参数值组合，并对它们进行修改(动态特性)。而其他组合测试算法都是，一旦一个测试用例或参数值被选定，在以后的扩展中就不再对它进行修改(静态特性)。由于被选定的值都是在当时情况下作为最优值被选择的，所以随着更多的值被选择，它们整体上不能构成最优解，即它们是局部最优。IPO_T算法能对已被扩展的测试用例进行修改这一动态特性，可以在某种程度上克服局部最优，最终得到一个相对更优的结果。算法详

细描述如下。

算法 IPO_T

输入：输入参数集ps，组合维数t

输出：测试用例集$\boldsymbol{T}$

{

1．初始化$\boldsymbol{T}$为空

2．把输入参数集ps按参数取值数多少非递增排列，设排列后的顺序为：$P_1, P_2, \cdots, P_n$

3．为$P_1, P_2, \cdots, P_t$ 的每个t参数值组合生成一个测试用例TestNum=$|P_1|*|P_2|*|P_t|$

//TestNum为测试用例数

4．for (int $i = t+1$; $i \leqslant n$; i ++)

{

5．从前i–1个参数中选出t–1个参数与第i个参数一起组成的t参数值组合构成集合π

// 水平扩展P_i

6．for (int j =0; $j<|P_i|$; j++)

{

7．$\boldsymbol{T}[j][i]=p_{i.vj}$;

8．初始化每行t参数值组合重复覆盖集R_j为空

9．从π中移出所有被覆盖的t参数值组合

}

10．for(int $j= |P_i|$; $j \leqslant$ TestNum; j++)

{

11．选择一个值$P_{i.v}$，使得第j个测试用例与前j–1个测试用例中t参数值组合重复数最少

12．计算R_j={(conbination$_1$, line$_1$), (conbination$_2$, line$_2$),$\cdots$, (conbination$_A$, line$_B$)

//(conbination$_n$, line$_n$)表示第j个测试用例与第n个测试用例中有conbinationn这个t参数值组合相同

13．for(每个line$_n$)

{

14．if (line$_n$行(测试用例)的所有t参数值的组合已被其它测试用例全部覆盖)

{

15．$\boldsymbol{T}[\text{line}_n][i]=*$;

16．更新所有包含line$_n$的R_m;

}

}

17．从π中移出所有被覆盖的t参数值组合

}

// 垂直扩展P_i

18．for (π中每个($p_{a.vb}, p_{c.vd,} \cdots, p_{e.vm}$))

{

19．If ($\boldsymbol{T}$中存在行相应列上的值为这个t参数值组合中的值或者$*$)

20．{替换这行中的$*$，使其包含($p_{a.vb}, p_{c.vd}, \cdots, p_{e.vm}$)}

else

{

21．TestNum++

22．$\boldsymbol{T}$中添加一行，使$a,c,\cdots,n$列的值分别为($p_{a.vb}, p_{c.vd}, \cdots, p_{e.vm}$)，其余列赋值为$*$从$\pi$中移出($p_{a.vb}, p_{c.vd}, \cdots, p_{e.vm}$)

}

}

23．return ts;}

该算法框架和IPO算法一样，首先初始化测试用例集$\boldsymbol{T}$为空，把输入参数按照非递增排序，并且产生前t个参数的测试用例集；然后执行一个从t+1到n的循环，每次循环处理一个参数。算法生成的测试集$\boldsymbol{T}$中存在“$*$”，表示可以用对应列的取值域中任何值代替，而不会影响覆盖率。

下面用一个简单的例子演示IPO_T算法。待测系统S_2有3个输入参数p_1、p_2和p_3，每个参数有3个可选值：{0,1,2}，t=2，所以算法首先为p_1和p_2产生测试用例集，如图2中的a所示，它包含了所有的p_1和p_2的参数值组合，共3×3个。又因为3大于2，所以需要对$\boldsymbol{T}$进行水平扩展覆盖输入参数p_3。因为p_1和p_2的所有2维参数值组合都已被覆盖，所以想要覆盖p_3，只需覆盖(p_1,p_3)和(p_2,p_3)这两个2维参数组合的所有2维参数值组合即可。其中，每组含有9个参数值组合，所以集合π中含有18个元素：

π={($P_{1.0}$,$P_{3.0}$),($P_{1.0}$,$P_{3.1}$),($P_{1.0}$,$P_{3.2}$),($P_{1.1}$,$P_{3.0}$),($P_{1.1}$,$P_{3.1}$),($P_{1.1}$,$P_{3.2}$),($P_{1.2}$,$P_{3.0}$),($P_{1.2}$,$P_{3.1}$),($P_{1.2}$,$P_{3.2}$),($P_{2.0}$,$P_{3.0}$),($P_{2.0}$,$P_{3.1}$),($P_{2.0}$,$P_{3.2}$),($P_{2.1}$,$P_{3.0}$),($P_{2.1}$,$P_{3.1}$),($P_{2.1}$,$P_{3.2}$),($P_{2.2}$,$P_{3.0}$),($P_{2.2}$,$P_{3.1}$),($P_{2.2}$,$P_{3.2}$)}

p_3有$p_{3.0}$、$p_{3.1}$、$p_{3.2}$共3个取值，因此，将$\boldsymbol{T}$中前3个测试用例分别扩展，有$R_0=\phi$，$R_1=\phi$，$R_2=\phi$,如图2中的b所示。图2中的c的第7个测试用例选择$P_{3.2}$是因为：选择值$P_{3.2}$，则($P_{1.2}$,$P_{2.0}$,$P_{3.2}$)与($P_{1.0}$,$P_{2.2}$,$P_{3.2}$)和($P_{1.1}$,$P_{2.1}$,$P_{3.2}$)的2维参数值组合重复数为0；而选择$P_{3.0}$和$P_{3.1}$2维参数值组合重复数都为1。同理，在扩展第8个测试用例时，选择参数值$P_{3.0}$，此时R_8={(($p_{2.1}$, $p_{3.0}$),5)}，所以更新R_5={(($p_{1.1}$,$p_{3.0}$),6)} ∪ {(($p_{2.1}$, $p_{3.0}$),8)}={(($p_{1.1}$, $p_{3.0}$),6), (($p_{2.1}$,$p_{3.0}$),8)}，可以看出第5个测试用例的所有的2维参数值组合都被其他测试用例覆盖，所以置图2中的d的第5个测试用例的第3

个参数值为*。经过水平扩展，所有的2维参数值组合都被覆盖。

因为垂直扩展和IPO算法差别不大，不再赘述。待测系统S_2生成的测试用例如图2中的e所示。

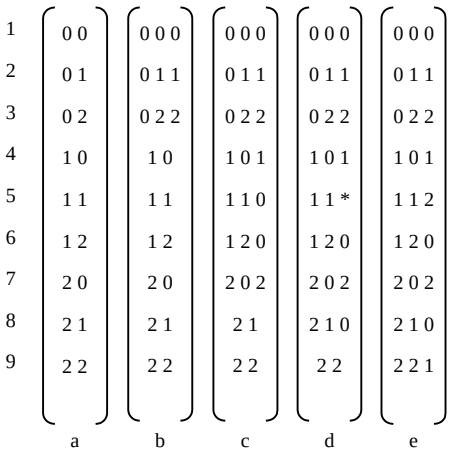

图2 运用IPO_T算法测试待测系统S_2的测试用例生成过程

3 实验结果

为了检测IPO_T生成测试用例的效果，本文将它的在生成测试用例数和时间花费与现有的部分t维组合测试算法(IPO、TCongfig和Jenny)进行比较。表1和表2分别列出了它们对TCAS(tracffic collision avoidance system)[10]系统和待测系统S_3进行不同维组合测试生成的测试用例数和时间花费。NA(not available)表示在适合的时间内运行相应算法没有获得结果。表3显示了各种算法对不同的待测系统进行不同维组合测试生成的测试用例数。所有待测系统如下：

S_3：8个输入参数，其中，2个参数有10个取值；1个参数有6个取值；1个参数有5个取值；2个参数有4个取值；2个参数有3个取值。

S_4：6个输入参数，每个参数都有10个取值。

S_5：8个输入参数，每个参数都有5个取值。

S_6：有15个输入参数，每个参数都有2个取值。

S_7：11个输入参数，1个参数有40个取值；1个参数有30个取值；1个参数有20个取值；1个参数有16个取值；2个参数有15个取值；4个参数有10个取值；1个参数有2个取值。

因为表1中除了IPO_T算法的实验数据是运行得的，其余数据都是从参考文献7中获得的，所以算法的时间不具可比性。从3个表中看出，在适合的时间内和多数情况下，IPO_T产生的测试用例相对较少。

表1　对TCAS各个算法生成的测试用例数和时间花费的比较

t	IPO_T		IPO		TConfig		Jenny	
	例数	时间/s	例数	时间/s	例数	时间/h	例数	时间/s
2	100	0.031	100	0.80	108	>1	108	0.001
3	401	0.624	400	0.36	472	>12	413	0.71
4	1 267	8.256	1 361	3.05	1 476	>21	1 536	3.54
5	3 809	38.453	4 219	18.41	NA	>24	4 580	43.54
6	9 560	149.234	10 919	65.03	NA	>24	11 625	470

表2　对S_3各个算法生成的测试用例数的比较

t	IPO_T	IPO	TConfig	Jenny
2	100	101	100	101
3	601	614	638	656
4	3 068	3 240	NA	3 533
5	12 831	13 606	NA	15 750

表3　各个算法生成的测试用例数的比较

系统(t)	IPO_T	IPO	TConfig	Jenny
S_4 (4)	16 846	18 761	NA	15 909
S_5 (3)	266	274	274	257
S_6 (6)	260	352	NA	272
S_7 (2)	1 200	1 224	1 208	1 214

4 结　论

IPO_T、IPO、TCongfig和Jenny算法对几个待测系统进行t维组合测试结果可知，多数情况下，IPO_T生成的测试用例较少，花费的时间也相对合理。

参 考 文 献

[1] KULN W D R. Failure modes in medical device softwre: an analysis of 15 years of recall data[J]. International Journal of Reliability, Quality and Safety Engineering, 2001, 8(4): 351-371.

[2] 王子元, 徐宝文, 聂长海. 组合测试用例生成技术[J]. 计算机科学与探索, 2008, 2(6): 571-588.
WANG Zi-yuan, XU Bao-wen, NIE Chang-hai. Survey of combinatiorial test generation[J]. Journal of Frontiers of Computer Science and Technology, 2008, 2(6): 571-588.

[3] MANDL R. Orthogonal latin squares: an application of experimental design to compiler testiing[J]. Communications of the ACM, 1985, 28(10): 1054-1058.

[4] COHEN D M, DALAL S R, FREDMAN M L, et al. The AETG system: an approach to testing based on combinatorial design[J]. IEEE Transactions on Software Engineering, 1997, 23(7): 437-444.

[5] LEI Y, TAI K C. In-parameter-order: a test generation strategy for pairwise testing[C]//Proc of the 3rd IEEE International High-Assurance Systems Engineering Symposium. Washington D C, USA: IEEE, 1998: 254-261.

(下转第619页)