软件测试方法与技术

组合测试

内容大纲

- 多参数的故障模型
- 利用正交表实现组合测试
- 组合测试的数学基础和定义
- 成对组合测试用例的生成策略
- 可变强度和具有约束的组合测试

1. 多参数的故障模型

- 除了单个因素导致的软件缺陷外，很多缺陷往往由多个因素交互而导致的缺陷。
- 组合测试（Combinatorial Test）是一种测试用例生成方法。它将被测应用抽象为一个受到多个因素影响的系统，其中每个因素的取值是离散且有限的。两因素（Pairwise）组合测试生成一组测试用例集，可以覆盖任意两个因素的所有取值组合。多因素（N-way，N>2）组合测试可以生成测试用例集，以覆盖任意 N 个因素的所有取值组合。
- 软件系统的行为受到很多因素的影响，例如输入参数、环境配置和状态变量。利用等价类划分或者边界值分析的方法确定不同因素可能的取值。

1. 多参数的故障模型

- 例如，Ubuntu 12.04 服务器版系统的 more 命令，其命令参数列表如图1所示。共有 11 个可选项，寻找能够满足测试需要的关系。在这些选项中，8 个选项是开关型的选项，比较容易确定参数的输入范围，其他几个选项先采用其他模式确定几个类型的输入类别。

```
clz@ubuntu:~$ more
Usage: more [options] file...
Options:
        -d              display help instead of ring bell
        -f              count logical, rather than screen lines
        -l              suppress pause after form feed
        -p              suppress scroll, clean screen and disblay text
        -c              suppress scroll, display text and clean line ends
        -u              suppress underlining
        -s              squeeze multiple blank lines into one
        -NUM            specify the number of lines per screenful
        +NUM            display file beginning from line number NUM
        +/STRING        display file beginning from search string match
        -V              output version information and exit
```

图1 Ubuntu中的more 命令参数

1. 多参数的故障模型

- 从简单的参数含义来看，不同的参数之间相互作用是不一样的，例如-V参数，仅仅显示版本信息，和显示效果没有直接关系，而其他对于显示的影响程度是不一样的，如表1所示。

表1 more命令的参数及其含义

参数	含义	影响显示行
-d	提示“Press space to continue，’q’ to quit”，禁用响铃功能	否
-f	计算行数时，以实际上的行数，而非自动换行过后的行数（有些单行字数太长的会被扩展为两行或两行以上）	是
-l	忽略Ctrl+l（换页）字符	是
-p	通过清除窗口而不是滚屏来对文件进行换页，与-c选项相似	否
-c	从顶部清屏，然后显示	否
-u	把文件内容中的下画线去掉	否
-s	把连续的多个空行显示为一行	是
-NUM	定义屏幕大小为NUM行	是
+NUM	从第NUM行开始显示	是
+/STRING	在每个档案显示前搜寻该字串（STRING），然后从该字串前两行之后开始显示	是
-V	输出版本信息	否

1. 多参数的故障模型

- 在实体的操作环境中也存在类似的问题。例如，在美国航天飞机驾驶舱中存在大量的开关和操作按钮，如图2所示。这些开关可能会存在相互作用，在测试时必须将所有的存在相互作用的开关或者按钮可能的组合都覆盖到。若系统具有 34 个开关，每个开关具有 2 种状态。根据全组合，那么需要的测试用例用例数量为：$2^{34}=1.7\times10^{10}$

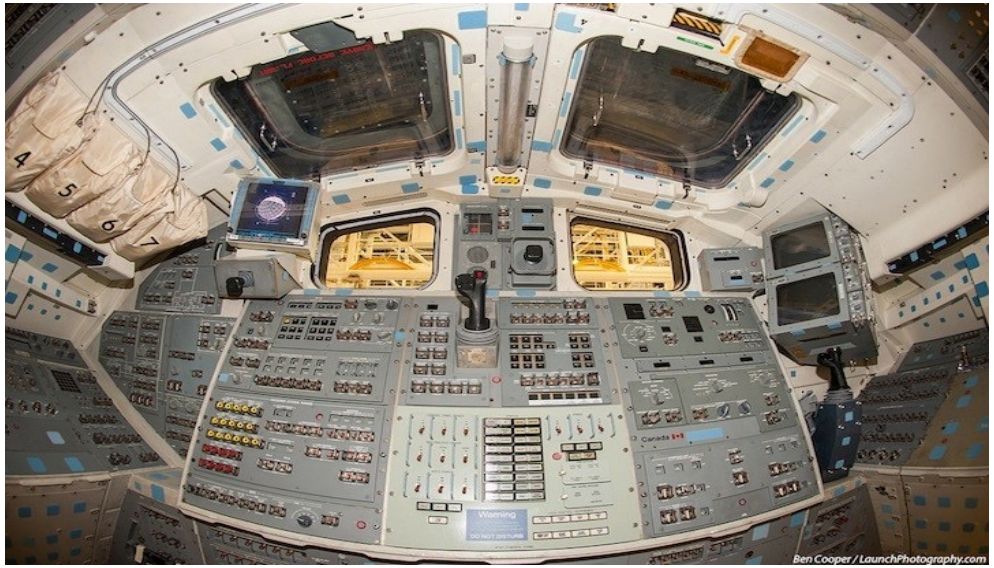


图2 美国航天飞机驾驶舱内景

1. 多参数的故障模型

- WPS的字体设置情况类似。在9.10 版的 WPS 文字中的字体设置中的效果设置，共有 11 个设置参数，包括删除线、阴影等，每个参数有选择和未选择两种状态，如 图3所示。这些设置之间存在较大的交互关系，也将产生巨大的测试用例集合。

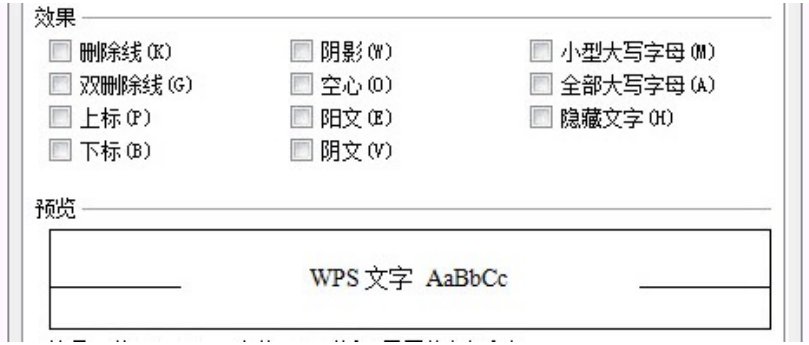


图3 WPS的字体设置中的效果设置

1. 多参数的故障模型

- 美国国家标准和技术研究院（NIST）研究浏览器、服务器软件、NASA 分布式数据库、医疗设备 4 类系统的错误检测率，如图4所示。以 NASA 分布式数据系统为例，67% 的错误是由单参数测试的，2 路参数组合测试可以发现 93% 的缺陷，3 路参数组合测试可以发现 98% 的缺陷。在其他系统中，缺陷发现曲线都是非常类似，在6参数组合，几乎可以达到了100%。如何降低在显著不降低质量的前提下降低测试数量是问题的出发点。美国国家标准和技术研究院（NIST）研究发现多种组合的测试效果随着的组合强度的增加快速增长，大量的测试结果表明在一般情况，4 路到 6 路组合已经满足大部分测试需求。

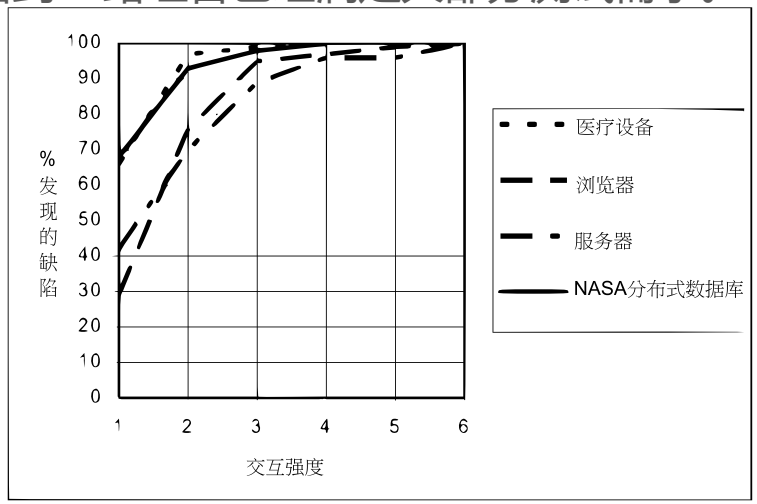


图4 发现缺陷和组合强度的关系

2. 利用正交表实现组合测试

- 2.1 拉丁方阵
  - 拉丁方阵的定义。用 n 个不同的拉丁字母（数字）排成一个 n 阶方阵，如果每行的 n 个字母均不相同，每列的 n 个字母（数字）均不相同，则称这种方阵为 n*n 拉丁方阵或 n 阶拉丁方阵。每个字母在任一行、任一列中只出现一次。即n阶方阵的每行、每列构成{1,2,3，…n}的两个全排列。

2. 利用正交表实现组合测试

- 例1 四个数字1、2、3、4构成的拉丁方阵：

$$A=\begin{matrix}1&2&3&4\\2&3&4&1\\3&\underline{4}&1&2\\4&1&2&3\end{matrix}$$

- 一个n阶拉丁方阵包含n个元素值，每一个元素同时包含了1个纵坐标和1个横坐标，元素值、横坐标、纵坐标三个取值范围均为1到n，将三者合成构成一个三元组（行号，列号，元素值），这个三元组恰好能够表示三个参数的组合。在例1中第3行，带有下划线的4，可以用三元组（3,2,4）。因此一个n阶拉丁方阵可以表示具有3个参数，每个参数具有n候选值的测试用例。在整个拉丁方阵表示的测试用例，可以用表2表示。

表2 4阶拉丁方阵表示的测试用例

用例编号	参数1	参数2	参数3
1	1	1	1
2	1	2	2
3	1	3	3
4	1	4	4
5	2	1	2
6	2	2	3
7	2	3	4
8	2	4	1
9	3	1	3
10	3	2	4
11	3	3	1
12	3	4	2
13	4	1	4
14	4	2	1
15	4	3	2
16	4	4	3

2. 利用正交表实现组合测试

- 若被测函数的输入参数大于3个，则需要应用正交拉丁方阵来表示。
- 正交拉丁方阵。设 $A=(a_{ij})$，$B=(b_{ij})$ 是两个n阶拉丁方阵，如果由A和B对应元素构成所有的有序对 (a_{ij},b_{ij}) 两两不同，则称A和B为一对正交的n阶拉丁方阵。
- 例2 下面拉丁方正A和B分别为3阶正交拉丁方阵。

$$A=\begin{matrix}1&2&3\\2&3&1\\3&1&2\end{matrix}\qquad B=\begin{matrix}1&2&3\\3&1&2\\2&3&1\end{matrix}$$

- 因为由A和B构成的组合方阵：

$$C=\begin{matrix}(1,1)&(2,2)&(3,3)\\(2,3)&(3,1)&(1,2)\\(3,2)&\underline{(1,3)}&(2,1)\end{matrix}$$

2. 利用正交表实现组合测试

- C中每一个元素都是由方阵A和B对应的元素构成有序对，并且互不相同。可以验证，由三个元素构成的正交拉丁方阵最多只有2个。
- 在一个由t个拉丁方阵组合而成的方阵中，每一个元素为由单个方阵元素构成的有序组合（含有t个元素），每一个元素也存在一个纵坐标和横坐标，它们可以进一步组合成t+2个元素的元组（行号、列,号、有序组合）。在例2中构成组合方阵C中，在第3行第2列带下划线的元组（1,3），其行号为3，列号为2，连同其坐标可以构成组合（3,2,1,3）。该组合可以表示由4个参数被测函数的测试用例。

2. 利用正交表实现组合测试

- 由组合方阵C表示的完整测试用例，可以测试具有4个参数的函数，每一个参数具有3个候选值的函数或者配置，表3所示。

表3 测试具有4个参数的拉丁方阵

编号	参数1	参数2	参数3	参数4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

- 若有t个n阶拉丁方阵，若他们两两正交，则他们组成一个n阶(t个)正交拉丁方阵组。若以N(n)表示n阶正交拉丁方阵组包含最多的拉丁方个数，已经证明N(n)≤n-1。对于n阶拉丁方阵，若存在n-1个正交拉丁方阵，则成其为n阶正交拉丁方阵完备组。

2. 利用正交表实现组合测试

- 对于任何的正整数都存在n阶拉丁方，但不是存在任意阶数的正交拉丁方。
- 例如，2阶拉丁方只有两个：
- $$A=\begin{matrix}1 & 2\\ 2 & 1\end{matrix} \qquad B=\begin{matrix}2 & 1\\ 1 & 2\end{matrix}$$
- 显然A和B不正交。
- 从上面的原理分析可以知道，应用正交拉丁方阵设计测试用例存在几个方面的限制：
  - （1）部分正交拉丁方阵不存在。若没有对应的拉丁方正，可以采取扩大拉大方阵阶数；
  - （2）一般要求参数的候选值的个数相同。如果候选值的个数不相同，只能选取候选个数最多的参数值个数作为统一的个数。
  - （3）参数的个数远远大于参数值个数。正交拉丁方阵的大小受参数值的个数限制，也可能不存在对应的正交拉丁方阵。

2. 利用正交表实现组合测试

- 2.2 正交表
  - 正交试验设计 (Orthogonal experimental design) 是研究多因素多水平的又一种设计方法，它是根据正交性从全面试验中挑选出部分有代表性的点进行试验，这些有代表性的点具备了“均匀分散，齐整可比”的特点是一种高效率、快速、经济的实验设计方法。
  - 日本著名的统计学家田口玄一将正交试验选择的水平组合列成表格，称为正交表。用正交表设计出来的试验方案之所以合理，是因为具有如下两个重要的特征：均衡搭配——正交性可以用较少的试验次数替代全部可能试验组合中好的、中等的、不好的搭配组合，使选出的较少的搭配组合具有均衡的代表性。综合可比——数据分析的依据可把复杂的多因素试验数据处理问题转化成单因素试验数据处理。通过试验数据的适当组合，可发现各组试验数据以及各因素影响之间的某种可比性。

2. 利用正交表实现组合测试

- 设有两组元素 a1, a2, ..., an 与 b1, b2, b3, ..., bk ，它们可构成如下的元素对：

$$\begin{matrix} (a_1,b_1) & (a_1,b_2) & \cdots & (a_1,b_k) \\ (a_2,b_1) & (a_2,b_2) & \cdots & (a_2,b_k) \\ \vdots & \vdots & \vdots & \vdots \\ (a_n,b_1) & (a_n,b_2) & \cdots\cdots & (a_n,b_k) \end{matrix}$$

称这些元素对是由元素 a1, a2, ..., an 与 b1, b2, b3, ..., bk的完全有序元素对，简称元素对。

例3 由数字（1,2,3,4）和（1,2,3）构成的完全有序数字对：

$$\begin{matrix} (1,1) & (1,2) & (1,3) \\ (2,1) & (2,2) & (2,3) \\ (3,1) & (3,2) & (3,3) \\ (4,1) & (4,2) & (4,3) \end{matrix}$$

2. 利用正交表实现组合测试

- 若在一个矩阵中的任意两列中，由两列中对应的数字是完全元素对并且每对出现的次数相等，则称两列是均衡搭配。
- 假设有矩阵A：

$$A=\begin{bmatrix}1&1&1\\1&1&2\\1&2&1\\1&2&2\\2&1&2\\2&1&2\\2&2&2\\2&2&2\end{bmatrix}$$

- 在矩阵A中，第1列和第 2 列中对应的元素构成 8 个数字对：(1, 1), (1, 1), (1, 2), (1, 2)(2, 1), (2, 1), (2, 2), (2, 2) 是完全数字对，每对各出现 2 次，因此称这两列为均衡搭配。

2. 利用正交表实现组合测试

- 因素 (Factors)。在一项试验中，凡欲考察的变量称为因素（变量）。在软件测试中，是指有多少个影响输出参数、变量或者配置。
- 水平 (Levels)。任何单个因素能够取得的值的个数。在软件测试中，是指一个参数、变量或者配置允许取值的个数。
- 强度（Strength）。变量间的相互关系。当强度为2时，只考虑变量两两之间的影响，如果强度为3，同考虑3个变量对结果的影响；当强度增加时，用例的个数会急剧增加。
- 次数（Runs）。实验次数的多少，在测试中就是指多少个用例。

2. 利用正交表实现组合测试

- 正交表。设A是一个 n×k 的矩阵（ n行k列)，其中j列元素的由元素 (1, 2, …, n) 构成，(j = 1, 2, …, k),若 A 的任意两列均衡搭配，则称A是一张正交表。
- 正交表一般表示成： $L_n(m_1 \times m_2 \times m_3 \times \cdots \times m_k)$
  - 式中：
  - L： 正交表的代号，是拉丁方正 Latin Square 的首字母；
  - k： 正交表的列数，每一列对应着一个实验因素；
  - n： 正交表的行数，表示实验的次数；
  - m_j ： 第 j 列中元素的个数，表示第 j 个因素的水平数。
- 正交 $L_8(4\times2\times2\times2\times2)$表示第1列的水平数为4，后面4列的水平数为2，实验次数为8。这个正交表可以表示一个函数的测试用例集，该函数共具有5个参数，第1个参数有4个不同的取值，而后4个参数各有2个不同的取值，共执行8次测试。

2. 利用正交表实现组合测试

- 在正交表中，若某些列的元素格式相同，则可以写成为指数形式：$L_n(l_1^{f_1} \times l_2^{f_2} \times \cdots \times l_i^{f_i})$
  - 式中：n表示次数，l_i 表示次数，f_i 表示因素数。例如前面的正交表可以表示成为：$L_8(4 \times 2^4)$。
- 各个水平数不完全相同的正交表称为混合水平正交表。$L_{16}(4^4 \times 2^3)$，$L_{16}(4 \times 2^{12})$ 等都是混合水平正交表。
- 在不显著降低检测能力的前提下，利用正交表可以显著减少测试用例数量。例如作一个三因素三水平的实验，按全面实验要求，须进行 $3^3 = 27$ 种组合的实验，且尚未考虑每一组合的重复数。若按 $L_9(3^3)$ 正交表按排实验，只需作9次实验，显然大大减少了工作量。因而正交实验设计在很多领域的研究中已经得到广泛应用。

2. 利用正交表实现组合测试

- 例4 有一个函数其输入有3个变量,每个变量分别有 2 种取值，有4个测试用例，即n=4。那么其对应的正交表可以表示为 $L_4(2^3)$，假设采用1,2 分别表示不同的变量取值，一个满足该条件的正交表可以表示如表4所示。

表4 一个3个变量的正交表示例

1	1	1
1	2	2
2	1	2
2	2	1

- 容易验证其满足正交性。

2. 利用正交表实现组合测试

- 例5 有一个函数其输入有5个变量, 前 4 个变量，每个变量分别有 2 种取值，第 5 个变量有4种取值。如果有8个测试用例，即n=8。那么其对应的正交表可以表示为：$L_8(2^4 \times 4)$。假设采用0,1,2,3 分别表示不同的变量取值，一个满足该条件的正交表可以表示如表5所示。

表5 一个5个变量的正交表示例

0	0	0	0	0
0	0	1	1	2
0	1	0	1	1
0	1	1	0	3
1	0	0	1	3
1	0	1	0	1
1	1	0	0	2
1	1	1	1	0

- 现以第4个因素和第5个因素为例，校验表格正交性。将下面左边的矩阵根据第1从0到1排序，然后将左边第2列根据0,1,2,3顺序排列，形成右边的矩阵。可以清晰看出其正交性。

$$\begin{bmatrix} 0 & 0 \\ 1 & 2 \\ 1 & 1 \\ 0 & 3 \\ 1 & 3 \\ 0 & 1 \\ 0 & 2 \\ 1 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 2 \\ 0 & 3 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}$$

2. 利用正交表实现组合测试

- 2.3 正交表性质
  - 正交表具有如下两个性质：
  - 性质1：任意列中各水平重复出现的次数相等。第 j 列个水平重复出现的次数： $t = n / m_j$
    - 性质1所表示的含义，在同一张正交表中，每个因素的每个水平出现的次数是完全相同的，所以由正交表所构成的测试用例，一个变量各个输入（或者配置）值出现次数相同。由于在试验中每个因素的每个水平与其它因素的每个水平参与试验的机率是完全相同的，这就保证在各个水平中最大程度的排除了其它因素水平的干扰。因而，能最有效地进行比较，找到好的试验条件。
  - 性质2：任意两列所构成的水平对是完全有序数字对，各个水平重复出现的次数相同（均衡搭配）。第i和j列所构成水平对重复的次数： $s = \frac{n}{m_i \times m_j}(i \neq j)$
    - 性质2表示，在同一张正交表中，任意两列（两个因素）的水平搭配（横向形成的数字对）是完全相同的，所以正交表所构成的测试用例，任意两个变量各个输入（或者配置）值对出现次数相同。保证了试验条件均衡地分散在因素水平的完全组合之中，因而具有很强的代表性，容易得到好的试验条件。

2. 利用正交表实现组合测试

- 用一个立方体图来说明正交表测试分布的均匀性。假设有个函数的测试需要3个输入变量（配置），每个变量3种可能的取值，全部覆盖需要做27次测试，如图5所示。这27次测试当于图中立方体各条线的交点，经正交试验设计后，正交试验的9个测试均匀分布在立方体的各个部位。在上、中、下、左、中、右，前、中、后的9个面上均衡整齐地分布着3个试验点，在27条线上，每条线上分布着一个点，非常均匀。因此，用这9个点进行测试基本上反映了27个测试的情况。

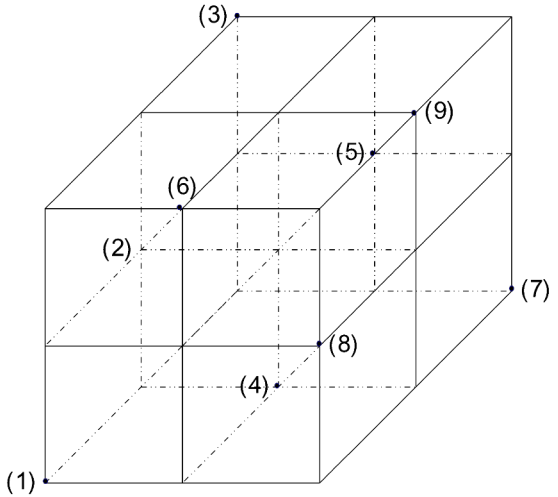


图5 三个因素三个水平的分布示意图

2. 利用正交表实现组合测试

- 根据正交表的性质，可以得到三种变换：
  - 列间置换： 正交表中任意两列可以相互交换。
  - 行间置换： 正交表中任意两行可以相互交换。
  - 水平置换： 正交表中任意一列中的水平数字可以相互交换。
- 如果一个正交表经过上述变换以后仍为正交表，则变换以后的正交表称为原正交表的等价表。对于具有等价表的正交表，在测试领域的含义：由正交表构成的测试用例，输入变量 (参数) 之间没有顺序关系。两个测试用例没有顺序关系。一个变量的参数取值没有顺序关系。

2. 利用正交表实现组合测试

- 2.4 正交表测试
  - 在应用正交表设计测试时，一般包括以下步骤：
    - （1）确定被测函数的输入参数个数，或者配置测试的因素个数，将其作为正交表的因素数。
    - （2）确定每一个被测函数输入参数或者配置因素的取值个数。这里的取值个数，通过其他的测试方法，例如边界值、等价类等方法确认的取值个数，是离散的、有限个取值，并非是变量值域范围的取值个数，因为在连续的值域范围内，其本身输入个数可能是无限的。
    - （3）依据步骤1和步骤2确定的因素数和水平数，选择合适的正交表。由于并非所有因素数和水平数的组合都存在正交表，当不存在对应的正交表时，应选择能够包含其因素和水平的正交表。可能存在三种情况：
      - 存在因素数和水平数刚好符合被测问题的正交表。
      - 因素数不相同。水平数相同，但是找不到相同因素数的正交表，可以选择因素数最接近，但略大的实际因素数的正交表。
      - 水平数不相同：因素的水平数不相同，但是可以选择水平数最接近但是略大的正交表。

2. 利用正交表实现组合测试

- （4）如果所选择的正交表中某个因素有多余的水平数，可以应用这个因素的可选值，进行填充，以增加发现缺陷的机会。填充的原则可以包括：
  - 随机填充，任意选择因素的一个值进行填充。
  - 依据因素不同可选值，可能发现问题的几率进行填充。例如边界值。
  - 依据该因素值和其他因素值的交互方式，进行填充。
  - 依据行业意见的经验进行填充。
- （5）在此基础上，可以根据行业要求或者测试经验增加若干个测试。

2. 利用正交表实现组合测试

- 例6： word 2010 中段落设置包含了换行选项和字符间距选项，文本对齐方式。由于换行选项和字符间距之间相互影响比较小，在此考虑换行设置之间的相互影响。换行设置共有三个选项，如图6所示。三个选项分别为：
  - （1）按中文习惯控制首尾字符
  - （2）允许西文在单词中间换行
  - （3）允许标点溢出边界
- 试采用正交法设计测试用例。
- 解答： 分析因素数目和水平数目，该设置共有三个选项：允许中文习惯控制首尾字符、允许西文在单词中间换行、允许标点溢出边界，因素的个数为3。
- 每个选项都包含了选择和不选择两种情况，每个因素的的水平数相同均为2。
- 选择正交表：
  - （1）表中的因数数目大于等于 3；
  - （2）表中至少有三个因数的水平数大于等于 2；
  - （3）行数取最少的一个。

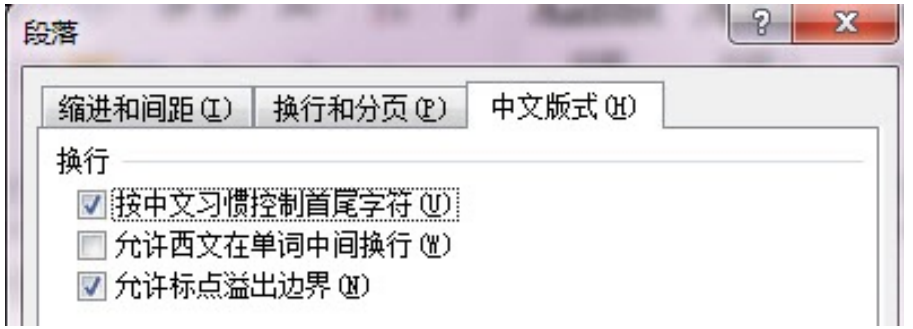


图6 Word 2010 中段落中换行选项

2. 利用正交表实现组合测试

- 依据上述原则，选择正交表，如表6所示。

表6 正交表 $L_4(2^3)$

	1	2	3
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

- 将变量取值和实际含义进行映射：
- 按中文习惯控制首尾字符： 0⇒ 不选择；1⇒选择。
- 允许西文在单词中间换行：0⇒ 不选择；1⇒选择。
- 允许标点溢出边界：0⇒ 不选择；1⇒选择。
- 可以得到最后的测试用例，如表7所示。

表7 WPS 字符设置的测试用例

	按中文习惯控制首尾字符	允许西文在单词中间换行	允许标点溢出边界
测试用例1	不选择	不选择	不选择
测试用例2	不选择	选择	选择
测试用例3	选择	不选择	选择
测试用例4	选择	选择	不选择

2. 利用正交表实现组合测试

- 例7：某手机相机软件，其设置如图7所示。
- 共有6个设置项：使用音量键作为、自拍、闪光灯、拍摄模式、效果、场景模式。各个选项候选值如表8所示。

表8 某照相机的设置参数以及取值

编号	选项名称（因素）	候选值（水平）	候选值个数
1	使用音量键作为	缩放键、摄像头键	2
2	自拍	开、关	2
3	闪光灯	开、关	2
4	拍摄模式	正常拍摄、面部优选、全景拍摄	3
5	效果	无效果、复古、黑白	3
6	场景模式	无、肖像、风景、运动、宴会/室内、海滩/雪景	6

图7 手机相机的设置界面

2. 利用正交表实现组合测试

- 在本软件的设置中，设置的选项为6个。使用音量键作为、自拍、闪光灯3个因素的水平数=2，拍摄模式和效果2个因素水平数为3个，场景模式这个因素的水平数为6。查阅相关的正交表，适合的有 $L_{49}(7^8)$、$L_{18}(3^6\times6^1)$等。选择行数较少的作为生成测试用例，如图8（a）所示。
- 在该正交表共有7因素，在实际系统中仅仅有6个因素。如果将照相机的第1个因素到第5个因素分别和正交表中的第1个因素至第5个因素对应，正交表第6列因素为多余因素，其取值对于相机系统的测试没有作用，可以忽略。
- 在前三个因素中，被测系统的选项仅有2个，在正交表提供了三个水平，若第1个选项和第2个选项，分别用0和1表示，在正交表中前三列中2在被测系统并没有任何含义。为了充分利用这些没有含义的2，可以将其转换成其有效的选项。在本例中，循环使用前两个选项作为测试用例值，以增加发现问题的能力。如图8（b）所示。

2. 利用正交表实现组合测试

- 依据正交表的值和系统实际选项进行映射，如表9所示。

表9 照相机设置参数及其取值

编号	选项名称（因素	候选值（水平）	水泡编号
1	使用音量键作为	缩放键	0
		摄像头键	1
2	自拍	关	0
		开	1
3	闪光灯	关	0
		开	1
4	拍摄模式	正常拍摄	0
		面部优选	1
		全景拍摄	2
5	效果	无效果	0
		复古	1
		黑白	2
6	场景模式	无	0
		肖像	1
		风景	2
		运动	3
		宴会/室内	4
		海滩/雪景	5

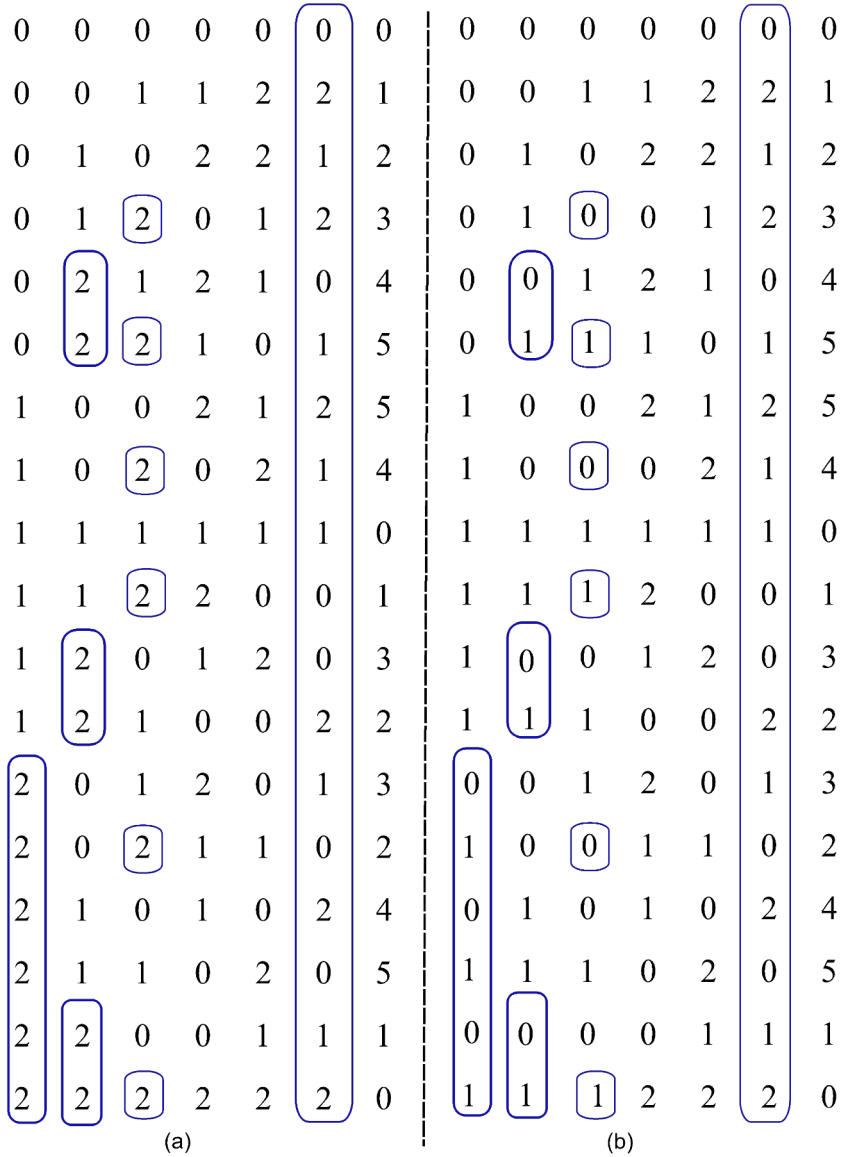


图8 手机照相机设置对应的正交表

2. 利用正交表实现组合测试

- 依据映射表和正交表，形成最终的测试用例，如表10所示。

表10 照相机设置的测试用例

用例编号	使用音量键作为	自拍	拍摄模式	闪光灯	效果	场景模式
1	缩放键	关	关	正常拍摄	无效果	无
2	缩放键	关	开	面部优选	黑白	肖像
3	缩放键	开	关	全景拍摄	复古	风景
…	…	…	…	…	…	…
17	缩放键	关	关	正常拍摄	复古	肖像
18	摄像头键	开	开	全景拍摄	复古	无

3. 组合测试的数学基础和定义

- 从n个不同元素中，任取m(m与n均为自然数，下同）个元素按照一定的顺序排成一列，叫做从n个不同元素中取出m个元素的一个排列；从n个不同元素中取出m个元素的所有排列的个数，称为从n个不同元素中取出m个元素的排列数。

$$P_n^m = \frac{n!}{(n-m)!}$$

- 假设有4个自然数，分别用1,2,3,4表示，先从中取3个数字并将其排列成1排，共有多少种排列结果。首先从4数中取1,2,3共三个数，其排列情况为（1,2,3）（1,3,2）(2,1,3),(2,3,1),(3,1,2),(3,2,1)，共6种排列。其他依次类推，可以分别取1,2,4和1,3,4以及2，3，4三组数据，每组分别有6中排列，共有24种情况，即 $P_4^3 = 4!/(4-3)! = 24$。

3. 组合测试的数学基础和定义

- 从n个不同元素中，任取m(m≤n)个元素并成一组，成为从n个不同元素中取出m个元素的一个组合。如果两个组合中的元素完全相同，不管元素的顺序如何，都是相同的组合；只有当两个组合中的元素不完全相同时，才是不同的组合。从n个不同元素中取出m(m≤n)个元素的所有组合的个数，成为从n个不同元素中取出m个元素的组合数。

$$C_n^m = \frac{n!}{m!(n-m)!}$$

- 为了对一个具有n个参数的程序的输入执行m路（维）组合测试，需要对于一个 m 路变量（输入参数）的所有取值组合进行测试，假设每一个变量有 r 种取值。覆盖的取值组合数为：$r^m C_n^m = r^m \frac{n!}{m!(n-m)!}$
- 每一个测试具有n个参数的一种取值组合，包含了 C_n^m 个m参数取值组合情况，这是组合测试的基本依据。

3. 组合测试的数学基础和定义

- 例8：若一个问题具有5个参数a,b,c,d,e，每一个变量具有2个取值，假设需要覆盖3路变量的组合，共需要覆盖的取值组合数为：$2^3 \times \frac{5!}{3!(5-3)!} = 80$
- 而在5个变量取3个变量，共有10中变量组合：

  abc,abd,abe,acd,ace,ade,bcd,bce,bde,cde
- 一个测试用例，例如(0,1,0,0,1)覆盖了$C_5^3 = 10$中变量的组合，如表11所示。

表11（0,1,0,0,1）覆盖的组合情况

1	2	3	4	5	6	7	8	9	10
abc	abd	abe	acd	ace	ade	bcd	bce	bde	cde
010	000	011	001	001	001	100	101	101	001

3. 组合测试的数学基础和定义

- 设待测软件 SUT(Software Under Testing) 的因素个数为 n，这些因素数形成有限集合 $P=\{p_1,p_2,...,p_n\}$ 其中每个因素 p_i 经过前期处理后共包含 a_i 个取值。为了讨论的方便，将 p_1 的可能取值记为V1 ，p_2 的可能取值集合记为V2，依次类推。
- 单一水平系统。若一个 SUT 中所有的参数（因素）的取值数量均相等 a1 = a2 = . . . = an，则称该 SUT 为单一水平系统。
- 混合水平系统。若一个 SUT 中的参数（因素）的取值数量不完全相等，即存在 ai≠ aj(i = j)则称该 SUT 为混合水平系统。

3. 组合测试的数学基础和定义

- 测试用例集：根据一定的覆盖准则设计的测试用例的集合。
- 依据测试用例集对于组合覆盖的强度，可以分为单一选择测试（Each choice）、基本选择测试（Base Choice）、2维组合测试（2-way）或者成对测试（Pairwise）、N维组合测试等。
- 依据覆盖的强度是否相等，可以分等强度组合测试和变强度测试。
- 依据是否附带约束条件，可以区分成不带约束的组合测试和带约束的组合测试。
- 单一选择测试（each choice）。若一个 SUT 中所有因素的所有可能取值至少被测试集中的一个测试用例覆盖，该覆盖准则由Ammann和Offutt提出。
- 在单一选择测试中，最少的测试用例个数由所有因素中取值最多的因素的取值个数所决定，即： $N_{\min} = \max(|Vi|)1 \le i \le n$

3. 组合测试的数学基础和定义

- 例9：一个系统有三个输入参数 P = {p1, p2, p3}, 三个参数对应的候选值分别为： V1= (a, b),V2= (1, 2, 3),V3= (x, y)。设计满足单一选择准则的测试用例。

p_1	p_2	p_3
a	1	x
b	2	y
–	3	–

p_1	p_2	p_3
a	1	x
b	2	y
a	3	x

- 首先在三个输入参数中的候选取值中，各取一个参数取值，例如（a，1，x）作为第1个测试用例。在此基础上要产生新的用例，对于p1和p3而言，取值值分别只剩下a和y，所以直接选取这两个值，而p2可以在2和3中选取一个值，暂且选择了2。然后确定第3个测试用例，p2只能选择值3，而p1和p3待定。这样第 1 列中前两行覆盖了参数 p1 中的两个参数 a, b。而第 2 列覆盖了参数 p2 中的两个参数 1，2，3 三个参数，第 3 列中前两行覆盖了参数p3 中的两个参数 x, y。如上图左边所示。p1的第 3 行选择 a 和 b 可以是任意，或者根据需求中的其他线索决定。类似地，p3的第 3 行选择 x 和 y 可以是任意，或者根据需求中的其他线索决定，若右边所示。

3. 组合测试的数学基础和定义

- 基本选择测试（base choice）。若一个 SUT 中的测试用例集每次仅由一个因素的可能选项变化而构成。
- 基本选择测试可以通过如下方法生成测试用例。第 1 个用例可以随机生成，或者根据其他信息（如参数的取值的重要性、或者取值出现的频率）而决定。第 2 个用例根据第 1 个参数的取值变化而其他参数值保持初始值不变。在第 1 个参数取完了所有的值以后，再根据第 2 个参数的取值变化而产生测试用例，以此类推。在上述所有的变化过程中，不涉及变化的参数均保持初始取值不变。

3. 组合测试的数学基础和定义

- 例10：一个系统有三个输入参数 P = {p1, p2, p3}, 三个参数对应的候选值分别为： V1= (a, b),V2= (1, 2, 3),V3= (x, y)。设计满足基本选择准则的测试用例。

p_1	p_2	p_3
a	1	x
b	1	x
a	2	x
a	3	x
a	1	y

- 在这个例子中，被测问题和前一个例子相同，但是覆盖的准则不同。首先在第 1 个用例，在各个参数分别选择了 a, 1, x，第 2 个用例根据参数 p1 的取值变化，取值b, 而其他两个参数值 p2,p3 保持不变。第 3,4 个用例，保持参数 p1 的初始值 1 不变，而将根据 p2的取值变化而产生的两个用例。最后 1 个用例仅仅变化 p3 的取值。
- 在基本选择组合过程中，初始用例的生成可以随机生成，或者根据其他信息（如参数的取值的重要性、或者取值出现的频率）而决定。但是初始用例的选择将影响不同参数取值出现的频率。除了初始取值以外，其他取值出现的可能性均为 1。

3. 组合测试的数学基础和定义

- 在例子中，初始用例的选择为 (a, 1, x)，除了初始选择以外的取值，例如参数p1中的 b, 参数 p2 中的 2, 3 以及 p3 中的 y 均只出现了 1 次。而参数在初始用例中的取值出现的次数是：

$$N_i = (|V_1| - 1) + (|V_2| - 1) + \cdots (|V_{i-1}| - 1) + (|V_{i+1}| - 1) \cdots |V_n| - 1) + 1$$

- 参数p1中A的取值出现(3-1)+(2-1)=4次。参数p2中a的取值出现了(2-1)+(2-1)+1=3次。在单一选择中，每次可以在一次用例中同时变化多个参数的选择，而基本选择组合测试中，每次只能变化一个参数的取值，显然满足基本选择的测试用例集，必然满足单一选择准则。

4. 成对组合测试用例的生成策略

- 4.1 CATS算法
  - 成对测试（Pairwise coverage）。对于一个软件任意两个参数，它们的任意一对有效取值至少被一个测试用所覆盖。
  - 例如一个系统有三个输入参数 P = (p1, p2, p3), 其中 p1的取值V1= {a, b}，p2的取值V2= {1, 2, 3},p3的取值V3= {x, y}。则下面的测试用例满足成对覆盖准则：

p_1	p_2	p_3
a	1	x
a	2	x
a	3	x
a	$--$	y
b	1	y
b	2	y
b	3	y
b	$--$	x

4. 成对组合测试用例的生成策略

- 在这个用例中的三个参素 p1, p2, p3, 对于每两个参数构成的任意一对有效取值，(p1, p2)、(p1, p3)、(p2, p3)构成的取值对，如：

p_1	p_2
a	1
a	2
a	3
b	1
b	2
b	3

p_1	p_3
a	x
a	y
b	x
b	y

p_2	p_3
1	x
1	y
2	x
2	y
3	x
3	y

- 对于其中任意两个因素pi和pj而言，构成的测试用例个数为|Vi|×|Vj|，例如p1和p2两个参数组合数为6种。但是对于超过两个因素，满足成对覆盖准则的测试用例个数是不确定的。人们一直采用各种策略实现满足成对测试需求，但是生成的测试用例个数尽可能地少，包括 CATS、AETG、DDA 等。

4. 成对组合测试用例的生成策略

- CASTS(Constrained Array Test System) 算法由 Sherwood 提出，采用一次生成一个测试用例，最终生成所需要的组合测试用例。在该方法中，先构造所有参数取值构成全组合并用集合Q表示，同时构造2个参数的组合放在UC集合中。从全组合集合Q中选取当前覆盖最多成对组合的的组合作为测试用例。

算法1 CATS 算法

（1）TS⇐Ø;
（2）UC ⇐ 因素数 P 的成对组合，表示未被覆盖的组合的集合;
（3）Q⇐所有参数取值构成全组合;
（4）while UC≠ Ø do
① 选择覆盖 UC 中当前最多成对组合的组合作为测试用例，如果存在多个，选② 择第一个遇到的组合作为测试用例;
③ 从 UC 中移去 TC 覆盖的组合;
④ 从 Q 中移去选择作为测试用例的组合;
⑤ TS ⇐ TS + TC;
（5）end while

4. 成对组合测试用例的生成策略

- 现举例说明CATS算法过程。假设一个系统有四个参数 p1,p2,p3,p4，其中第一个参数p1有 4 种取值即V1={a1,a2,a3,a4}，p2 和 p3 有 2 种取值，V2={b1,b2}，V3={c1，c2}，p4 有 3种取值，V4={d1,d2,d3}那么UC共有 4×2+4×2+4×3+2×2+2×3+2×3= 44 种取值。测试用例套集合TS=ø，Q 的初始值包含所有参数的组合，Q={(a1, b1,c1, d1), . . . , (a4,b2,c2,d3)}。UC包含了所有2个参数取值构成组合。

$$
\begin{aligned}
UC = \{&(a_1,b_1),(a_1,c_1),(a_1,d_1),(a_1,b_2),(a_1,c_2),(a_1,d_2),(a_1,d_3),\\
&(a_2,b_1),(a_2,c_1),(a_2,d_1),(a_2,b_2),(a_2,c_2),(a_2,d_2),(a_2,d_3),\\
&(a_3,b_1),(a_3,c_1),(a_3,d_1),(a_3,b_2),(a_3,c_2),(a_3,d_2),(a_3,d_3),\\
&(a_4,b_1),(a_4,c_1),(a_4,d_1),(a_4,b_2),(a_4,c_2),(a_4,d_2),(a_4,d_3),\\
&(b_1,c_1),(b_1,d_1),(b_1,c_2),(b_1,d_2),(b_1,d_3),\\
&(b_2,c_1),(b_2,d_1),(b_2,c_2),(b_2,d_2),(b_2,d_3),\\
&(c_1,d_1),(c_1,d_2),(c_1,d_3),\\
&(c_2,d_1),(c_2,d_2),(c_2,d_3)\}
\end{aligned}
$$

4. 成对组合测试用例的生成策略

- 开始时，一个测试用例最多可以覆盖个两参数值组合。第1个取值组合为TC1=(a1,b1,c1,d1)其覆盖了C={(a1,b1)、（a1,c1）、（a1,d1）、(b1,c1)、（b1,d1）、（c1,d1）}，共6个值组合，已经覆盖最多组合的测试用例。所以Q=Q-{(a1,b1,c1,d1)}，UC=UC-C。

$$
\begin{aligned}
UC = \{&(a_1,b_2),(a_1,c_2),(a_1,d_2),(a_1,d_3),\\
&(a_2,b_1),(a_2,c_1),(a_2,d_1),(a_2,b_2),(a_2,c_2),(a_2,d_2),(a_2,d_3),\\
&(a_3,b_1),(a_3,c_1),(a_3,d_1),(a_3,b_2),(a_3,c_2),(a_3,d_2),(a_3,d_3),\\
&(a_4,b_1),(a_4,c_1),(a_4,d_1),(a_4,b_2),(a_4,c_2),(a_4,d_2),(a_4,d_3),\\
&(b_1,d_1),(b_1,c_2),(b_1,d_2),(b_1,d_3),\\
&(b_2,c_1),(b_2,d_1),(b_2,c_2),(b_2,d_2),(b_2,d_3),\\
&(c_1,d_2),(c_1,d_3),\\
&(c_2,d_1),(c_2,d_2),(c_2,d_3)\}
\end{aligned}
$$

4. 成对组合测试用例的生成策略

- 第2次选择了测试用例TC2=(a1,b2,c2,d2)，第3次选择了测试用例TC3=(a2,b1,c2,d3)，TC4=(a3,b2,c1,d3)这4个测试用例其覆盖的两参数组合数均为6,。CASTS算法经过 4 次循环以后，产生了 4 个测试用例，测试用例套集合为：

p_1	p_2	p_3	p_4
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_2
a_2	b_1	c_2	d_3
a_3	b_2	c_1	d_3

- UC的成员成为：

$$
\begin{aligned}
UC = \{&(a_1, d_3),\\
&(a_2, c_1),(a_2, d_1),(a_2, b_2),(a_2, d_2),\\
&(a_3, b_1),(a_3, d_1),(a_3, c_2),(a_3, d_2),\\
&(a_4, b_1),(a_4, c_1),(a_4, d_1),(a_4, b_2),(a_4, c_2),(a_4, d_2),(a_4, d_3),\\
&(b_1, d_2),\\
&(b_2, d_1),\\
&(c_1, d_2),\\
&(c_2, d_1)\}
\end{aligned}
$$

4. 成对组合测试用例的生成策略

- 此时，覆盖UC中值对最多的测试用例分别为(a4,b1,c1,d2)和(a4,b2,c2,d1)，这两个用例覆盖的UC中元素数量都是5。选择TC5= (a4,b1,c1,d2)和TC6= (a4,b2,c2,d1)作为测试用例。测试用例套集合为：

p_1	p_2	p_3	p_4
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_2
a_2	b_1	c_2	d_3
a_3	b_2	c_1	d_3
a_4	b_1	c_1	d_2
a_4	b_2	c_2	d_1

UC的元素为：$UC = \{(a_1, d_3), (a_2, c_1), (a_2, d_1), (a_2, b_2), (a_2, d_2), (a_3, b_1), (a_3, d_1), (a_3, c_2), (a_3, d_2), (a_4, d_3)\}$

- 此时，覆盖UC中值对最多的测试用例分别为(a2,b2,c1,d1)和(a3,b1,c2,d1)，这两个用例覆盖的UC中元素数量都是3。选择TC5=(a2,b2,c1,d1)和TC6=(a3,b1,c2,d1)作为测试用例。测试用例套集合为：

p_1	p_2	p_3	p_4
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_2
a_2	b_1	c_2	d_3
a_3	b_2	c_1	d_3
a_4	b_1	c_1	d_2
a_4	b_2	c_2	d_1
a_2	b_2	c_1	d_1
a_3	b_1	c_2	d_1

4. 成对组合测试用例的生成策略

- 此时UC包含剩下的4个两参数组合：$UC=\{(a_1,d_3),(a_2,d_2),(a_3,d_2),(a_4,d_3)\}$
- 由于四个两参数组合都是有p1和p4的取值构成，不包含p2和p3的取值，并且取值之间没有任何的交集，所以增加4个用例，分别包含剩下的p1和p4的取值，如下面左边所示。p2和p3的取值可以任意，根据出现的先后次序，也可以根据其它的启发式策略分别加以填充。最后生成的测试用例TS如下面右边所示。

p_1	p_2	p_3	p_4
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_2
a_2	b_1	c_2	d_3
a_3	b_2	c_1	d_3
a_4	b_1	c_1	d_2
a_4	b_2	c_2	d_1
a_2	b_2	c_1	d_1
a_3	b_1	c_2	d_1
a_1	–	–	d_3
a_2	–	–	d_2
a_3	–	–	d_2
a_4	–	–	d_3

p_1	p_2	p_3	p_4
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_2
a_2	b_1	c_2	d_3
a_3	b_2	c_1	d_3
a_4	b_1	c_1	d_2
a_4	b_2	c_2	d_1
a_2	b_2	c_1	d_1
a_3	b_1	c_2	d_1
a_1	b_1	c_1	d_3
a_2	b_1	c_2	d_2
a_3	b_2	c_1	d_2
a_4	b_2	c_2	d_3

4. 成对组合测试用例的生成策略

- 4.2 AETG法
  - AETG(Automatic Efficient Test Generator) 法也属于一次生成一个测试用例的策略，在生成过程中引入随机技术的启发式组合，由 D.M.Cohen 提出。
  - 首先从UC中选择出现次数最多的参数取值作为该参数取值，然后将其他参数依据随机排列构成参数元组。从第2个参数开始，一次选择和前面参数构成覆盖最多UC元素作为该参数的取值，形成本轮测试用例的候选。这样的过程需要重复 M 遍。在这个M个候选中，依据其覆盖UC最多元素确定本轮的测试用例。
  - 当 1 个参数有多个取值覆盖同样多的参数对时，任意选择一个参数值，作为候选的测试用例，这样会产生不一样的候选测试用例。当 M 个候选的测试用例都已经产生以后，选择其中覆盖 UC 中最多的参数值对的候选测试用例作为下一个测试用例。

4. 成对组合测试用例的生成策略

- 在 AETG 算法中，最后生成的测试用例集规模的大小和 M 值的大小有关。一般地，M 值越大，产生的测试用例数越少。但是当 M 超过 50 以后，最后测试用例集规模的下降就不明显了。

算法2 AETG算法

（1）TS⇐Ø;

（2）UC ⇐ 因素数 P 的成对组合，表示未被覆盖的组合的集合。

（3）while UC≠ Ø do

① for i =0 to M do

i 在UC中找出出现次数最多的参数取值v_{ij} ，并将对应的参数p_i 设为第一个参数 p'_1。

ii 对剩下的参数进行随机排列，和第一个参数一起构成了参数序列 $(p'_1,p'_2,...p'_k)$

iii 从 p'_2 开始， p'_i 选取与前面参数 $(p'_1,p'_2,...p'_{i-1})$ 取值构成的组合覆盖UC中最多元素作为本次值。形成一个新的测试用例TC

② endfor

③ 从 UC 中移去 TC 覆盖的组合；

④ TS ⇐ TS + TC;

（4）end while

4. 成对组合测试用例的生成策略

- 现举例说明ATEG算法的过程。假设一个系统有四个参数 p1,p2,p3,p4，其中第一个因素p1有3种取值即V1={a1,a2,a3 }，p2 和 p3 有 2 种取值，V2={b1,b2}，V3={c1，c2}，p4有 3种取值，V4={d1,d2,d3}那么共有 36 种取值。测试用例套集合TS=ø，Q 的初始值包含所有参数的组合，Q={(a1, b1,c1, d1), . . . , (a3,b2,c2,d3)}。
- 假设经过3轮循环以后，产生了3个测试用例：

p_1	p_2	p_3	p_4
a_1	b_2	c_1	d_2
a_2	b_2	c_1	d_3
a_3	b_1	c_2	d_1

- UC所包含的集合为：

$$UC = \{(a_1,b_1),(a_1,d_1),(a_1,c_2),(a_1,d_3),$$
$$(a_2,b_1),(a_2,d_1),(a_2,c_2),(a_2,d_2),$$
$$(a_3,c_1),(a_3,b_2),(a_3,d_2),(a_3,d_3),$$
$$(b_1,c_1),(b_1,d_2),(b_1,d_3),$$
$$(b_2,d_1),(b_2,c_2),$$
$$(c_2,d_2),(c_2,d_3)\}$$

4. 成对组合测试用例的生成策略

- 在选择新的候选用例时，计算在 UC 中的出现次数最多的参数值。在上述的 UC 中，b1 和 c2 均出现5次。若选择参数值c2，那么将会产生用例模板: $(-,-,c_2,-)$
- 对于除了p3 参数以外的其他三个参数做随机排序，假设为p4、p1 和 p2。首先考虑参数 p4 的取值情况，(−, −, c2, d1) 没有覆盖任何 UC 中的任何参数对。而 (−, −, c2, d2) 和 (−, −, c2, d3)各覆盖了 1 个参数值对。假设算法选择 d2，那么产生了： $(-,-,c_2,d_2)$
- 接下来选择参数 p1 的取值情况，(a1, −, c2, d3) 和 (a3, −,c2, d3) 个覆盖了 UC 中的 1 个参数对，而(a2, −,c2, d3) 覆盖了 UC 中的 2 个参数对，因此参数p1选择值a2。形成

$$\vdots$$
$$(a_2,-,c_2,d_2)$$

- 最后考虑参数p2 的取值情况，(a2, b2, c2, d3) 覆盖了 1 个参数值对，而 (a2, b1, c2,d3) 覆盖了 UC 中的2 个参数值对。至此，产生了第 1 个候选的测试用例： (a_2,b_1,c_2,d_2)

4. 成对组合测试用例的生成策略

- 4.3 IPO方法
  - CAT、AETG 等算法均采用一次生成一个测试用例(one-test-at-a-time) 策略，而 IPO (In parameter order) 算法考虑复用已有的测试用例集。
  - IPO 由Y.Lei 等人提出。
  - 其基本思想是首先生成任意两个参数的所有组合，然后依次进行水平扩展（ horizontal growth ）和垂直扩展(vertical growth)。水平扩展即每次新加入一个参数，并确定其取值；在水平扩展之后，通过垂直扩展来补充尚未被覆盖的配对组合。它与AETG算法的根本区别是它不是同时考虑所有参数，而是每次渐进考虑一个参数。该方法依据前两个参数生成满足成对覆盖准则的测试用例集，然后通过扩展测试用例使之能够满足 3 个参数的成对组合覆盖，依此类推，直至所有的参数都包括到测试用例中,详细描述见算法3。

4. 成对组合测试用例的生成策略

算法3 IPO 算法

输入：系统S的参数及其对应的取值集合。

输出：满足成对测试的测试用例集。

（1）$TS=\{(v_i,v_j)\mid v_i\in V_1,v_j\in V_2\}$ %取第1个和第2个因素的组合

（2）n=|P| %n为参数个数

（3）if n==2 then 算法终止。

（4）for p_i (i=2,3,…n) do

水平扩展 IPO_H(TS, p_i)

for each test($v_1,v_2,\ldots,v_{i-1}$) in TS do

用$(v_1,v_2,\ldots,v_{i-1},v_i)$ 替换$(v_1,v_2,\ldots,v_{i-1})$ %根据不同的策略选择v_i

endfor

垂直扩展 IPO_V(π, TS);

% π 是由参数 $p_1,p_2,\ldots,p_i$ 取值构成的，未被TS覆盖的值对。

While TS 未覆盖 p_i 和 $p_1,p_2,\ldots,p_{i-1}$ 构成的值对 do

增加针对参数$p_1,p_2,\ldots,p_{i-1},p_i$ 一个新的用例到TS；

end while

（5）endfor

4. 成对组合测试用例的生成策略

- Y.Lei 等人分析IPO算法的优点在于：
  - （1）IPO策略允许在水平扩展和垂直扩展是采用不同的优化策略来生成测试用例。
  - （2）在软件更新，接口增加新参数时复用已有的测试用例。假设对于系统S，已经生成的测试用例为TS，新版本为S’。通过IPO策略，将通过扩展TS来产生新的测试用例TS’，可以节省产生测试用例的精力。
  - （3）在软件更新，若某一个参数增加新的取值时，复用已有的测试用例。假设对于系统S，已经生成的测试用例为TS，新版本为S’。在这种情况下，仅仅应用垂直扩展增加新的用例，即可生成新测用例集TS’。 TS’的测试用例集，可能比重新生成新的用例集的数量更多些，但节省了比较多的精力。
- 通过不同的策略可以有不同的水平扩展算法，事实上Y.Lei本人就已经提出了IPO_H_IV算法和IPO_H_EC算法。

4. 成对组合测试用例的生成策略

- IPO_H_IV水平扩展。首先比较当前参数pi的取值个数|Vi|和已经生成的测试用例个|TS|大小，将其最小值设为s。对于参数pi而言，前s个参数，不需要做任何的启发式动作，直接将其扩展到测试用例中；若已经生成的测试用例个数多余参数个数，则对于余下的测试用例的扩展，选择其覆盖参数值对最多的参数值作为pi的取值；若pi的取值个数多于已经生成的测试用例个数，则直接pi的前s个取值，pi多余的取值在垂直扩展中增加。其详细描述见算法4。

4. 成对组合测试用例的生成策略

算法4 IPO_H_IV 算法

输入：由参数 $p_1,p_2,\cdots,p_{i-1}$ 构成的测试集TS，水平扩展参数 p_i 。
输出：增加 p_i 取值水平扩展以后的测试集TS。
%初始化;
（1）π ={参数 p_i 和参数 $p_1,p_2,\cdots,p_{i-1}$ 之间构成的值对}
%s为p参数值个数和已经产生的测试用例个数的最小值。
（2）s=min(|V|,|TS|) %简单选择 p_i 的前s个值。
（3）for j=1 to s do
给参数 p_i 赋值 v_{ij} 扩展第j个测试用例;
从π中移除覆盖的值组合。
（4）end for
（5）if |TS|>|V| then
%已经产生测试用例个数大于参数 p_i 的取值个数|V|
a) for j=|V|+1 to |TS| do
选择参数 p_i 中覆盖π中组合最多的取值来扩展第j个测试用例;
从π中移去已覆盖的组合。
b) end for
（6）end if

算法5 IPO_V_IV 算法

输入：水平扩展后的测试用例集TS，尚未覆盖的组合集π。
输出：完成p取值垂直扩展的测试用例集。
（1）π ={参数p和参数p, p, …p之间构成的、尚未被覆TS覆盖的值对集合}
（2） TS' =Ø
（3）for (v_{iu}, v_{jw}) in π do
a) if TS'中包含测试用例，其 p_i 的取值为"-"， p_j 的取值为w then修改该测试用例，将"-"替换成u;
b) else产生新的测试用例tc，该用例中 p_i 的取值为u，p_j 的取值为w，其他参数的取值为"-"。 $TS' = TS' \cup \{tc\}$
c) endif
（4）end for
（5）TS=TS∪TS'

- 通过算法4水平扩展以后，如果还存在未被覆盖的值对，那么执行算法5进行扩展。

4. 成对组合测试用例的生成策略

- 举例说明IPO算法实现过程。系统S具有三几个参数p1,p2,p3,p1的取值集合为V1={a,b }，p2的其取值集合为V2={1,2}，p3的取值集合为V3={x,y,z}，依据IPO算法构建满足两两组合的测试用例集。
- 依据构建前两个参数p1，p2的取值两两覆盖准则。前两个参数均具有两个取值，共有4个元素：

p_1	p_2
a	1
a	2
b	1
b	2

- 接下来，考虑p3的水平扩展，由于参数p3有3个取值，而p1和p2已经产生的参数取值共有4个用例。依据水平扩展算法，前3个测试用例直接取p3的值，产生如下右边的测试用例。

p_1	p_2	p_3
a	1	x
a	2	y
b	1	z
b	2	–

4. 成对组合测试用例的生成策略

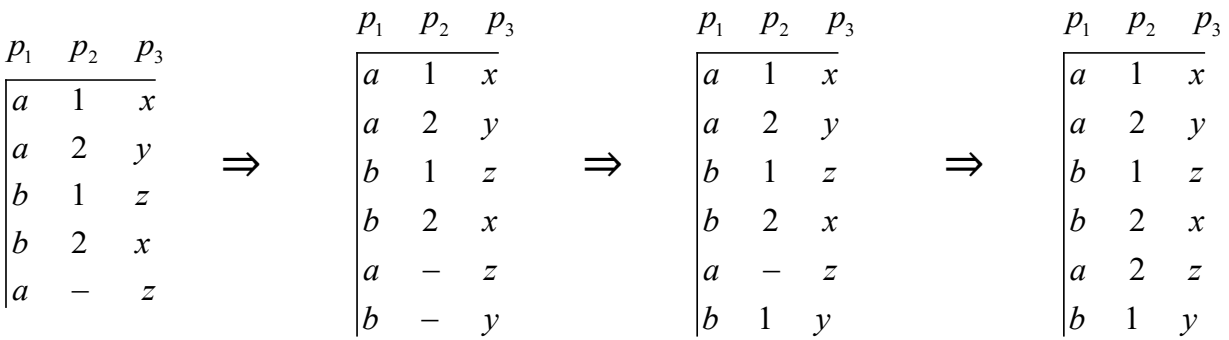
- 对于第4个测试用例中，p3值没有确定，可以选择x,y,z三个取值中的一个。扩展到第3个参数，构成未被测试用例覆盖的值对包括：π={(a,z), (b,x),(b,y), (1,y), (2,x), (2,z)}。如果选择x作为参数p3的值，那么构成了测试用例（b,2,x）其覆盖π中间的2个值对(b,x)和(2,x)。选择y作为参数p3的值，那么构成测试用例（b,2,y）其覆盖π中间1个值对(b,y)。如果选择z作为参数p3的值，那么构成了测试用例（b,2,z）其覆盖π中间的 1个值对(2,z)。因此，可以选择x作为参数p3的值，那么构成了测试用例（b,2,x），如下所示

p_1	p_2	p_3
a	1	x
a	2	y
b	1	z
b	2	x

4. 成对组合测试用例的生成策略

- 自此，完成p3参数的横向扩展，已经生成的测试用例集中共有4个测试用例，未被覆盖的值对包括：π={(a,z), (b,y), (1,y), (2,z)}。接下来执行垂直扩展来覆盖π中的值对。在开始时TS’=ø。对于π中的（a,z）的值对，（a,z）被测试用例(a,-,z)所覆盖，这里-表示在当前暂时不关心其取值。将其增加到TS’中，TS’={(a,-,z)}。接下来考虑第2个值对(b,y)，(b,y)中的b和y均为在TS’中任何测试用例中出现过，所以产生新的测试用例(b,-,y)，并将其添加到TS’中TS’={(a,-,z),(b,-,y)}。接下来考虑(1,y),由于y在TS’出现过，并且其p2对应的值为”-“，所以将TS’中的测试用例(b,-,y)修改为(b,1,y)。对于第4个值对(2,z)，由于在TS’中存在测试用例(a,-,z)，存在z，并且p2值为”-”，所以将其(a,-,z)修改为(a,2,z)，所以TS’={(a,2,z),(b,1,y)}，TS=TS∪TS’。

4. 成对组合测试用例的生成策略



- 重复上述过程对因素集合中剩余的因素进行水平扩充和垂直扩充可得到最终组合覆盖表，若表中仍有未填的值”-”，则从该因素可选值集合中随机选择产生一个值替换”-”。

4. 成对组合测试用例的生成策略

- 4.4 GA法
  - 构建所有参数覆盖组合 UC。然后从一个空的测试用例集开始，每次利用遗传算法增加一个测试用例到测试用例集中，该测试用例为当前覆盖 UC 种参数组合最多的测试用例，并删除该测试用例在 UC 覆盖的参数组合，一直到 UC 为空。
  - 遗传算法是一类模拟生物进化的智能优化算法，它是由 J.H.Holland 于六十年代提出 。
  - 通过模拟自然界并应用随机理论而形成的生命进化机制，其主要特征在于群体搜索策略和简单的遗传算子，群体搜索可使遗传算法实现整个解空间的分布式信息探索、采集和继承。遗传算法是从种群 (代表问题潜在解的集合) 开始的，按照适者生存和优胜劣汰的原理。逐代演化产生出越来越好的近似解。在每一代，根据问题域中个体的适应度大小挑选个体，并借助于自然遗传学的遗传算予进行组合交叉和变异，产生出代表新的解集的种群。新生代种群比前代更加适应于环境，末代种群中的最优个体经过解码，可以作为问题近似最优解。

4. 成对组合测试用例的生成策略

- 遗传算法包括三个基本操作：选择、交叉和变异。
- 算法的具体描述如下：
  - （1）初始化参数组合 UC 和测试用例集 TS。
  - （2）利用遗传算法计算当前覆盖 UC 最多的测试用例 TC。
  - （3）将 TC 添加到集合 TS 中 T S = T S ∪ T C ，删除 TC 覆盖的 UC 中参数组合。
  - （4）重复步骤 (2) 和 (3)，直到 UC 为空。
- 系统 S1 输入由 A、 B和C3 个参数组成, 参数取值个数依次为 2、2和3，p1的其取值集合V1={a,b }, p2的其取值集合V2={1,2}, p3的其取值集合V3={x,y,z},那么可以构造 UC，如表12所示。

表12 GA算法中UC集合示例

第1值对		第2值对		第3值对	
p	p	p	p	p	p
a	1	a	x	1	x
a	2	a	y	1	y
b	1	a	z	1	z
b	2	b	x	2	x
--	--	b	y	2	y
--	--	b	z	2	z

4. 成对组合测试用例的生成策略

- 在遗传算法产生组合测试用例，首先要执行测试用例的编码。所谓编码方式，就是将问题的潜在解使用适合遗传算法的基因编码表示。对于组合测试而言，就是将参数映射成二进制编码。
- 由于不同的参数的取值个数并不相同，一个参数在基因编码中的位数也不相同。一个参数 p_i 的可能取值个数为 L_i，如果 $2^{n-1} \le L_i \le 2^{n-1}$，那么该参数的编码长度为 n。
- 设有一个系统 S2 输入由 4 个参数组成 (p1, p2, p3, p4), 参数取值个数依次为4,2,2,3。由于组合测试用例的生成关注点在于测试用例的生成，用参数取值的序号来表示参数的实际值，这样可以避免参数类型的转换。例如测试用例 (3,0,1,2) 表示p1 取第 3个值 (第一个为 0)，p2 取第0个值，p3 取第1个值，p4 取第2个值。

4. 成对组合测试用例的生成策略

- 若 $L_i \le 2^n$, 那么在 L_i 和 2^n 之间随机填充所允许的参数值。系统 S2 可以采用长度为 6 的二进制编码表示，p1占用位b0b1，p2占用位b2，p3占用位b3，p4占用位b4 b5。对于p4而言，前 3 个编码 00,01,10，分别为v41,v42,v43，编码 11 可以随机v41、v42和v43中间的任意一个，可以使用某一启发式算法选取，以增加测试效果。最后的编码情况如表 13所示。

表13 参数取值以及编码

参数p_1		参数p_2	参数3_2	参数p_4	
b_0	b_1	b_2	b_3	b_4	b_5
00	v_{10}	0 v_{20}	0 v_{30}	00	v_{40}
01	v_{11}	1 v_{21}	1 v_{31}	01	v_{41}
10	v_{12}	--	--	10	v_{42}
11	v_{13}	--	--	11	-

4. 成对组合测试用例的生成策略

- 利用遗传算法生成的测试用例集，要求在满足成对测试准则的前提下，测试用例数尽可能地少，因此在每一代进化时选择尽可能多覆盖 UC 中参数组合的测试用例。设群体数量为 N，个体 (测试用例)TCi 覆盖的 UC中参数组合数为 Ci,1 ≤i≤N 。则个体 TCi 的适应度函数可以表示如下

$$f_i = \frac{c_i}{\sum_{i=1}^{N} c_i} \quad \text{其中} \quad \sum_{i=1}^{N} f_i = 1$$

- 计算当前群体的适应度值后，从当前群体中选择一些个体作为新一代群体的父辈。若个体的适应度高，则被选中的几率较大，且可能多次选中；反之，几率较小，甚至不会被选中。根据个体 (测试用例) 的适应度值 fi确定选择的概率 PSi = fi。选择的累积函数 Cj 定义：

$$C_j = \sum_{i=1}^{j} PS_i$$

4. 成对组合测试用例的生成策略

- 在选择新个体时，按照赌轮 (rouletter wheele) 方式来确定, 每个个体 TCj 处于选择区间 [Cj , Cj+1)。计算时每次产生一个选择随机数，那么处于对应选择区间的个体将被选中作为进化成员。
- 在选择以后，个体通过交叉和变异实现进化。交叉采用单点交叉的方法实现。两个长度为N的个体，分别表示如下：

$$\begin{array}{cccccccc} a_1 & a_2 & \cdots & a_{m-1} & a_m & \cdots & a_N \\ b1 & b2 & \cdots & b_{m-1} & b_m & \cdots & b_N \end{array}$$

- 在m出(1≤m≤N)出交叉，则产生两个新的个体：

$$\begin{array}{cccccccc} a_1 & a_2 & \cdots & a_{m-1} & b_m & \cdots & a_N \\ b1 & b2 & \cdots & b_{m-1} & a_m & \cdots & b_N \end{array}$$

- 对应于系统S2的2个测试用例及其编码为：

$$\begin{array}{l} TC_1 = (3,0,1,2) \\ TC_2 = (1,1,0,2) \end{array} \Rightarrow \begin{array}{l} CD_1 = 110110 \\ CD_2 = 011010 \end{array}$$

- 假设选择交叉点为4位，那么交叉以后编码和测试用例分别为：

$$\begin{array}{l} CD_1^{'} = 110110 \\ CD_2^{'} = 011010 \end{array} \Rightarrow \begin{array}{l} TC_1^{'} = (3,0,0,2) \\ TC_2^{'} = (1,1,1,2) \end{array}$$

4. 成对组合测试用例的生成策略

- 变异可以促进群体的多样化，防止群体进化过早的收敛，即防止群体进化停滞不前或冻结，若无变异，则新群体中的测试数据值即为局限于初始化的数值。变异对于个体选定一个变异位，在变异几率的控制下，将变异位用随机数替换该位，变异过程将产生单个新个体，添加到新一代群体中。在本算法中采用二进制编码，对于一个个体 (测试用例) ：只需将变异位取反就可以得到新的测试用例。以测试用例(3,0,1,2) 的编码表示为 110110 为例，选择变异位为第2位和第3位，那么变异以后的编码为 101110，对应的测试用例演变成为 (2,1,1,2)。

$$TC_1 = (3,0,1,2) \Rightarrow \mathrm{CD}_1 = 110110 \Rightarrow CD_1^{'} = 101110 \Rightarrow TC_2^{'} = (2,1,1,2)$$

5. 可变强度和具有约束的组合测试

- 5.1 混合强度的组合测试
  - 实际的软件系统的输入参数之间的关系有多种形式：
    - （1）有些输入参数之间不存在约束关系。
    - （2）有些输入参数之间可能会存在约束关系。
    - （3）参数之间存在交互关系，但是不同的参数之间的组合要求强度是不相同。
  - 组合方法所产生的测试用例集TS是一个m×n的矩阵。其中n为被测函数的参数（或者被测配置的因素），m为测试用例集TS所包含的测试用例数量。
  - 对于一个正整数k（1≤k≤n），如果TS中的任意k列，即第i1,i2,…ik列均要求满足k路组合，则k称为组合强度。其所产生的测试用例集为固定强度组合测试用例套（Fixed-strength），k可以称为组合强度。

5. 可变强度和具有约束的组合测试

- 固定强度组合测试可以表示为： $TS_{Inv}(P,m,k)$
  - 式中：
    - P：为参数或者因素的集合
    - m：测试用例的个数
    - k： 组合的强度
- 若组合强度为k的组合测试用例集TS存在t个子集TSi⊆TS(t=1,2,…,t)，其中TSi包括了ni个因素，因素之间构成了m×ni个元素的矩阵。其中TSi为强度为Ki的固定组合强度测试集，且Ki≠K，则称其为混合强度（Mixed-Strength）的组合测试。其具有ni因素子集及其强度ki，可以用如下表示：$C_i = \{p_{i1}, p_{i2}, \cdots, p_{in_i}\}@k_i$

5. 可变强度和具有约束的组合测试

- 混合强度的组合的基本含义，在全局的基础上，对于部分参数的组合提出了更高的要求。若有A、B、C、D、E五个参数，其取值均是0、1。在强度为2，其覆盖的组合如表14所示。

表14 两个参数之间的组合情况

	AC	AD	AE	BC	BD	BE	CD	CE	DE
00	00	00	00	00	00	00	00	00	00
01	01	01	01	01	01	01	01	01	01
10	10	10	10	10	10	10	10	10	10
11	11	11	11	11	11	11	11	11	11

- 若在此基础上，加入C={B,C,D}@3，那么覆盖的组合如表15所示。

表15 增加3参数强度以后的组合要求

AB	AC	AD	AE	BCD	BE	CE	DE
00	00	00	00	000	00	00	00
01	01	01	01	001	01	01	01
10	10	10	10	010	10	10	10
11	11	11	11	011	11	11	11
--	--	--	--	100	--	--	--
--	--	--	--	101	--	--	--
--	--	--	--	110	--	--	--
--	--	--	--	111	--	--	--

5. 可变强度和具有约束的组合测试

- 相比与强度为2的组合测试，其覆盖的2参数组合BC、BD、CD均包含在一个3参数组合中。
- 对于变强度的组合测试，其组合要求，可以在固定强度组合的基础上，添加约束表示，该约束用因素集合及其组合强度表示。 $C=\{C_1,C_2,\cdots,C_i\}$
- 变强度的组合表示为：$TS_v(P,m,k,C)$
  - 式中：
    - P：为参数或者因素的集合
    - m：测试用例的个数
    - k: 全局的组合强度
    - C：个性化的强度要求的集合

5. 可变强度和具有约束的组合测试

- 例如，一个函数（或者系统）包含了4个输入参数（配置因素），为了区分各个参数的取值分别用不同的代号表示，其本身没有特定的含义：
  - V1={1,2}
  - V2={a,b}
  - V3={I,II}
  - V3={x,y,z}
- 若采用一个成对组合测试，可以找到一个用例数为6的测试用例集 $TS_{Inv}(P,6,2)$ 。

p_1	p_2	p_3	p_4
1	a	I	x
1	a	II	y
1	b	I	z
2	a	II	z
2	b	I	y
2	b	II	x

- 若根据系统的特征分析，认为p1,p2和p3三个参数必须满足强度为3的组合测试要求。即增加了一个强度要求：$C_1 = \{p_1, p_2, p_3\}@3$
- 显然上述测试用例，并不能满足要求。一个满足该要求的测试用例表示如下：

p_1	p_2	p_3	p_4
1	a	I	x
1	a	II	y
1	b	I	z
1	b	II	x
2	a	I	y
2	a	II	z
2	b	I	x
2	b	II	y

5. 可变强度和具有约束的组合测试

- 5.2 参数值之间的约束
  - 参数值之间的约束关系一般用条件判断来实现，其一般的规则为：
    - IF Condition1 THEN Condition2 ELSE Condition3
    - 这里Condition1、Condition2、Condition3均为条件表达式或者关系表达式，一般不存在赋值或者其他语句。其含义是如果Condition1满足，那么Condition2 也必须满足，否则 Condition3必须满足。
    - 例如：
      - Fat格式的文件大小不能超过4096，FAT32格式的文件大小不能超过32000，约束关系可以表示为：
      - IF [File system] = "FAT" THEN [Size] <= 4096；
      - IF [File system] = "FAT32" THEN [Size] <= 32000；

5. 可变强度和具有约束的组合测试

- 条件表达式，一般由逻辑运算符连接关系表达式而成，逻辑运算符支持非、与、或三种运算：NOT, AND, and OR。其中NOT为逻辑非，是单目运算符，其结果和运算对象的逻辑结果相反。AND为逻辑与运算，是双目运算符，只有参加运算的对象同时为真，结果才为真。AND为逻辑与运算，是双目运算符，只要参加运算的对象有一个为真，结果就为真。
- 关系表达式，由关系运算符连接不同的变量或常量表达式而成：=, <>, >, >=, <, <=。在PICT中，定义了变量或者常量表达式的类型，从而支持关系表达式。PICT支持了字符串和数值两种类型。除了字符串以外，均认为数值类型，如表16所示。

表16 PICT中各种约束表达式

	设计运算符
条件表达式	IF Condition1 THEN Condition2 ELSE Condition3
逻辑运算符	NOT,AND, and OR
关系运算符	=, <>, >, >=, <, <=
通配符	表示任何字符,表示一个字符
集合关系	IN

5. 可变强度和具有约束的组合测试

- 在实际应用中，有时参数的表达要比简单一个表达更加复杂。例如，表17给出了一个约束的例子。

表17 复杂关系的约束示例

参数	取值
A:	0, 1
B:	0, 1
C	0, 1
约束：IF [A] = 0 THEN [B] = 0;IF [B] = 0 THEN [C] = 0;IF [C] = 0 THEN [A] = 1;	

- 在这个约束中，当A取值0时，B只能取值0，而B取值为0时，C只能取值0，而C取值0时，A取值为1，显然这和前面预定义A取值0矛盾。因此，包含A取值为0的测试用例均是无效的测试用例，而不能简单去考察A和B之间的约束值。同时，包含B取值为0，而C取值为1的测试用例也是无效的测试用例。

5. 可变强度和具有约束的组合测试

- 增加约束，并不一定减少测试用例。在一个a,b,c三个参数取值均为0,1。在没有约束的条件下，产生如下包含了4个测试用例的测试用例集。其模型文件为内容为：
  - a: 0, 1
  - b: 0, 1
  - c: 0, 1
- 在组合强度为2的情况下，PICT产生的测试用例为：

a	b	c
0	1	0
0	0	1
1	1	1
1	0	0

- 若加上约束，当a因素取值为0是，b因素取值也一定为0。则其模型文件内容为：
  - a: 0, 1
  - b: 0, 1
  - c: 0, 1
- IF [a] = 0 THEN [b] = 0;
- 在强度为2的情况下，PICT产生的测试用例集为：

a	b	c
1	1	1
0	0	1
1	1	0
0	0	0
1	0	0

5. 可变强度和具有约束的组合测试

- 5.3 种子组合和负面测试
  - 在实际应用中，由于有些参数值之间组合的重要性，要求在实际测试过程中，测试必须覆盖到该组合。这种必须测试到的参数组合称为种子组合(Seed)。
  - 一般而言，种子组合用于测试一些关键参数值组合。另一方面，通过种子组合，也可以使得以前产生的测试用例重新得到利用。
  - 在PICT中，种子组合是一个单独文件存在。第1行是用TAB分隔的参数名称，后继每一行表示一个种子组合，在种子组合中的值之间也是用TAB进行分隔。每一行的种子组合可以使完整的，每一个参数均指定了特定的值，也可以是忽略其中的部分参数的取值。

5. 可变强度和具有约束的组合测试

- 假若有如下的PICT模型文件model.txt
  - #
  - # Machine
  - #
  - OS: Win2K, WinXP
  - SKU: Pro, Server, Datacenter, WinPowered
  - LANG: EN, DE
  - ARCH：X86,IA64
- 一个可能的种子文件如下：

OS	SKU	LANG	ARCH
Win2K	Pro	EN	X86
Win2K		DE	X86
WinXP	Pro	EN	IA64

- 原则上，种子文件的参数、值及其约束关系应该和模型文件中参数一致。如果不一致，PICT将根据如下原则处理种子组合：
  - （1）忽略种子文件中包含了模型文件中不存在的参数。
  - （2）忽略种子文件中在模型文件中不存在的参数值。
  - （3）忽略种子文件中违反了在模型文件中的定义约束条件的行。

5. 可变强度和具有约束的组合测试

- 例子，若有一个系统的模型文件如下：
  - a: a1,a2,a3
  - b: b1,b2
  - c: c1,c2,c3,c4
  - d: d1,d2
- 在没有添加任何种子文件时，利用PICT所生成的测试用例集如表18所示。

表18 未带种子文件的测试用例

编号	a	b	c	d
1	a1	b1	c2	d2
2	a1	b2	c3	d1
3	a2	b2	c3	d2
4	a2	b1	c2	d1
5	a1	b2	c4	d2
6	a2	b1	c1	d1
7	a3	b1	c4	d1
8	a1	b2	c1	d2
9	a2	b1	c4	d1
10	a3	b2	c3	d2
11	a3	b1	c1	d1
12	a1	b1	c3	d2
13	a3	b2	c2	d2

5. 可变强度和具有约束的组合测试

- 依据系统特性，（a1,b1,c1,d1）增加到种子文件。即种子文件为内容为：
  - a b c d
  - a1 b1 c1 d1
- 在PICT所产生的测试用例中，其中第一行就是种子文件中所提供的测试组合。在没有增加任何种子，PICT所产生的测试用例集中包含了13个测试用例，而增加了一个组合以后，其测试用例集合反而降低为12个用例。由此可以看出增加种子组合，并不一定增加的测试用例，如表19所示。

表19 带有2个种子组合的测试用例

编号	a	b	c	d
1	a1	b1	c1	d1
2	a1	b2	c4	d2
3	a3	b2	c3	d1
4	a3	b1	c2	d2
5	a1	b2	c2	d1
6	a2	b1	c2	d2
7	a2	b1	c3	d1
8	a1	b1	c3	d2
9	a2	b1	c4	d1
10	a2	b2	c1	d2
11	a3	b1	c1	d2
12	a3	b1	c4	d2

5. 可变强度和具有约束的组合测试

- 依据系统特性，（a1,b1,c1,d1）和（a2,b2,c2,d2）增加到种子文件，即产生的种子文件内容为：
  - a b c d
  - a1 b1 c1 d1
  - a2 b2 c2 d2
- 在PICT所产生的测试用例中，其中前两行就是种子文件中所提供的测试组合，其所产生的测试用例集合为12个用例，如表20所示。

表20 带有3个种子组合的测试用例

编号	a	b	c	d
1	a1	b1	c1	d1
2	a2	b2	c2	d2
3	a3	b1	c4	d2
4	a2	b2	c4	d1
5	a2	b1	c3	d1
6	a3	b2	c1	d1
7	a3	b1	c2	d1
8	a1	b2	c2	d2
9	a1	b2	c3	d2
10	a3	b2	c3	d1
11	a1	b1	c4	d1
12	a2	b1	c1	d2

5. 可变强度和具有约束的组合测试

- 依据系统特性，（a1,b2,c3,d2）和（a2,b2,c4,d2）增加到种子文件，即产生的种子文件内容为：
  - a b c d
  - a1 b2 c1 d2
  - a1 b2 c2 d2
  - a1 b2 c3 d2
  - a1 b2 c4 d2
- 在PICT所产生的测试用例中，其中前两行就是种子文件中所提供的测试组合，其所产生的测试用例集合为13个用例，如表21所示。

表21 带有3个种子组合的测试用例

编号	a	b	c	d
1	a1	b2	c1	d2
2	a1	b2	c2	d2
3	a1	b2	c3	d2
4	a1	b2	c4	d2
5	a2	b1	c4	d1
6	a3	b2	c3	d1
7	a3	b1	c4	d2
8	a2	b2	c1	d2
9	a2	b1	c2	d1
10	a2	b1	c3	d1
11	a3	b1	c2	d2
12	a1	b1	c1	d1
13	a3	b2	c1	d2

5. 可变强度和具有约束的组合测试

- 在软件测试过程中，经常还需要使用一种称为“负面测试”的技术，其核心本质是输入不合法的数据，用于确认程序是否正确处理了错误的输入。在负面测试过程中，在一个测试用例中，仅允许出现一个参数取值为不合法的值。在一般的程序第1次遇到一个错误时，立即进入到异常处理，这样有可能屏蔽后面不合法数据的检测功能。例如，在程序1中，求变量a、变量b和变量量c的平方根的和。加入简单输入执行print sumsquareroot(-4,-9,-16)语句，-4和-9,-16均是不合法数据，但是在语句[1]中的已经抛了异常，实际上使得语句[2]和语句[3]并没有被测试到。

程序1 一个负面测试例子

```
import math
def sumsquareroot(a,b):
   if(a<0):   raise ;    #[1]
   if(b<0):   raise;     #[2]
   if(b<0):   raise;     #[3]
   return (math.sqrt(a)+math.sqrt(b)+ math.sqrt(c))
print sumsquareroot(-4,-9)
```

5. 可变强度和具有约束的组合测试

- 为了避免这种情况的出现，需要在产生组合测试用例之前，确定哪些是非法数据，哪些是合法数据。在最后产生测试用例中，避免在一个测试用例中同时出现多个非法数据值。在PICT中，默认采用在参数值之前增加~来表示。
- 例如，上述程序中，模型文件内容为：
  - a: ~-1，0, 1
  - b: ~-1，0, 1
  - c: ~-1，0, 1
- 其产生的测试用例如表22所示。

表22 带有负面测试的测试用例

a	b	c
0	1	0
0	0	1
1	1	1
1	0	0
0	1	~-1
1	~-1	0
~-1	0	0
1	~-1	1
1	0	~-1
0	~-1	0
~-1	1	1