6. Generation from Combinatorial Design

Andrea Polini

Advanced Topics on Software Engineering – Software Testing
MSc in Computer Science
University of Camerino

Combinatorial Design

- Configuration space: all possible settings of the environment variable under which P could be used
- Input space: all possible values that can be taken by input variables

Combination of hardwares, OSs, platforms etc. is generally referred to as compatibility testing

Combinatorial Design

Example

Consider a program P that takes two positive integers x, y as input, and that is meant to be executed on the OSs Windows, Mac Os, and Linux through Mozilla, Explorer or Chrome browsers. Which are the Configuration and input spaces?

- factors: parameters possibly influencing program behaviour
- levels: values that can be assumed by a factor

- Factor combination leads to exponential growth
- test configuration is a static selection while test values (parameters) are input provided to a running IOT
- it is in general not meaningful to combine input parameters and the configuration space

Combinatorial Design

Example

Consider a program P that takes two positive integers x, y as input, and that is meant to be executed on the OSs Windows, Mac Os, and Linux through Mozilla, Explorer or Chrome browsers. Which are the Configuration and input spaces?

- factors: parameters possibly influencing program behaviour
- levels: values that can be assumed by a factor

- Factor combination leads to exponential growth
- test configuration is a static selection while test values (parameters) are input provided to a running IOT
- it is in general not meaningful to combine input parameters and the configuration space

Combinatorial Design

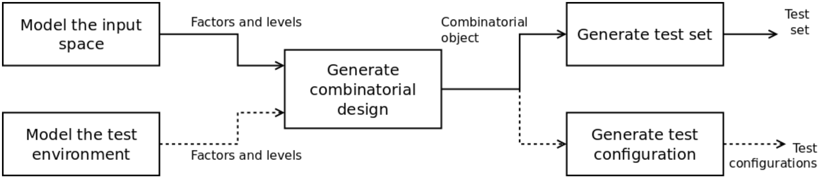
Example

Consider a program P that takes two positive integers x, y as input, and that is meant to be executed on the OSs Windows, Mac Os, and Linux through Mozilla, Explorer or Chrome browsers. Which are the Configuration and input spaces?

- factors: parameters possibly influencing program behaviour
- levels: values that can be assumed by a factor

- Factor combination leads to exponential growth
- test configuration is a static selection while test values (parameters) are input provided to a running IOT
- it is in general not meaningful to combine input parameters and the configuration space

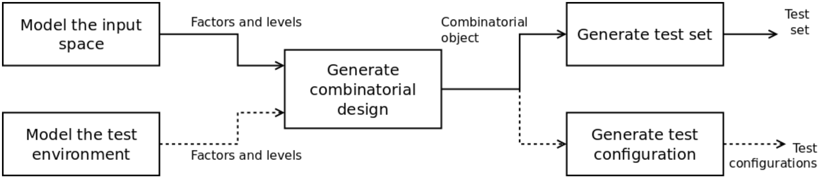
Combinatorial test-design process



Each factor combination may lead to one or more test cases where each test case consists of values of input variables and the expected output. Nevertheless, as usual the generation of all combinations is generally not feasible

k factors with n level each lead to n^k possible combinations

Combinatorial test-design process



Each factor combination may lead to one or more test cases where each test case consists of values of input variables and the expected output. Nevertheless, as usual the generation of all combinations is generally not feasible

k factors with n level each lead to n^k possible combinations

Fault model

The approach we are going to discuss targets interaction faults

- interaction faults are triggered when a certain combination of $t \geq 1$ parameter values causes the program containing the fault to enter an invalid state
- faults triggered by some value of input variables regardless of the values of other inputs variables are known as simple faults. When $t = 2$ they are known as pairwise interaction faults. For arbitrary value of t we refer to t-way interaction faults.

Example - 1

Imagine a program that should return the value calculated by different combinations of a couple of functions. In particular when `x=x1` and `y=y1` the returned value should be `f(x,y,z)+g(x,y)` and `f(x,y,z)-g(x,y)` when `x=x2` and `y=y2`. Now consider the program:

```
begin
  int x,y,z;
  input(x,y,z);
  if (x==x1 and y==y2)
    output(f(x,y,z));
  else
    if (x==x2 and y==y1)
      output(g(x,y));
    else
      output(f(x,y,z)+g(x,y));
end
```

Example - 2

Let $x, y \in \{-1, 0, 1\}$ and $z \in \{0, 1\}$. Are there interaction faults that can be discovered in the following code snippet?

```
begin
  int x,y,z,p;
  input(x,y,z);
  p = (x+y)*z; // instead should be (x-y)*z
  if (p >= 0)
    output(f(x,y,z));
  else
    output(g(x,y));
end
```

Fault vectors and Latin squares

- A fault vector is a k-uple of values for the factors of a program able to trigger a fault. The vector is considered a t-fault vector if any $t \leq k$ elements in V are needed to trigger the fault in P.
- A Latin Square of order n is an $n \times n$ matrix such that no symbol appears more than once in a row and a column where the alphabet set Σ as cardinality n.
  e.g. $\Sigma = \{A, B\}$ and $\Sigma = \{1, 2, 3\}$
  - Latin squares are a useful tool to derive factor combinations in a smaller number with respect to brute force strategies

Latin squares properties

Given a Latin square described by matrix $\mathscr{M}$ a large number of same order matrices can be obtained through row and column interchange and symbol-renaming operations
A latin square obtained by the mentioned operations is said to be isomorphic to the starting latin square

A latin square can be easily derived using modulo arithmetic
$M(i,j) = (i+j) \ mod \ k$ – where k is the order of the square

Latin squares properties

Given a Latin square described by matrix $\mathscr{M}$ a large number of same order matrices can be obtained through row and column interchange and symbol-renaming operations
A latin square obtained by the mentioned operations is said to be isomorphic to the starting latin square

A latin square can be easily derived using modulo arithmetic
$M(i,j) = (i + j)\ mod\ k$ – where k is the order of the square

Mutually orthogonal latin squares (MOLS)

MOLS

MOLS are a useful tool to generate $t - wise$ vectors from latin squares. Two latin squares are mutually orthogonal if their combination in a matrix of the same order does not generate duplicates.
Let's consider the case of two latin squares of order 3

$MOLS(n)$ indicates a set of MOLS of order n. If n is prime MOLS(n) contains $n-1$ MOLS and it is referred as a complete set. MOLS exists for each $n > 2 \wedge n \neq 6$

Let's build the MOLS(5) set

Pairwise design - binary factors

Let's consider three factors X, Y, Z each one with two levels, and let's generate a pairwise design.

A set of combinations is balanced when each value occurs exactly the same number of times

Pairwise design - binary factors

Let's consider three factors X, Y, Z each one with two levels, and let's generate a pairwise design.

A set of combinations is balanced when each value occurs exactly the same number of times

Pairwise design - binary factors

Generalizing the problem on n factors each one having two levels.

- we need to define $\mathscr{S}_{2k-1}$ to be the set of strings of lenght $2k-1$ such that each string has exactly k 1s. e.g. $k=3$

	1	2	3	4	5
1	0	0	1	1	1
2	0	1	1	1	0
3	1	1	1	0	0
4	1	0	1	1	0
5	0	1	1	0	1
6	1	1	0	1	0
7	1	0	1	0	1
8	0	1	0	1	1
9	1	1	0	0	1
10	1	0	0	1	1

The SAMNA procedure

Input: n - number of two-valued input variables (factors) Output: A set of factor combinations such that all pairs of input values are covered

1. Compute the smallest integer k such that $n \leq |\mathscr{S}_{2k-1}|$
2. Select any subset of n strings from $\mathscr{S}_{2k-1}$. Arrange these to form an $n \times (2k-1)$ matrix with one string in each row, while the columns contain different bits each string
3. Append a columns of 0s to the end of each string selected
4. Each one of the $2k$ columns contain a bit pattern from which we generate a combination is of the kind $(X_1^*, X_2^*, \ldots, X_n^*)$ where the value of each variable is selected depending on whether the bit in column i, $i \leq i \leq n$ is a 0 or a 1

Example

Consider a simple Java applet named `ChemFun` that allows a user to create an in-memory database of chemical elements and search for an element.

Factor	Name	Levels	Comments
1	Operation	{Create,Show}	Two buttons
2	Name	{Empty,Nonempty}	Data Field, String
3	Symbol	{Empty,Nonempty}	Data Field, String
4	Atomic Number	{Invalid, Valid}	Data Field, data > 0
5	Properties	{Empty,Nonempty}	Data Field, String

Testing for all combinations would require a total of 2^5 tests, but if we are interested for testing for pairwise interactions we can reduce the number of tests to 6.

Pairwise design for multivalued factors

In most practical cases factors can assume more than just two levels

- SAMNA cannot be applied
- MOLS(n) can be used to derive test set to satisfy the pairwise criterion

PDMOLS algorithm

Input: n - number of factors
Output: a test set satisfying the pairwise criterion

1. Label the factors as $F_1, F_2, \ldots, F_n$ such that the following ordering constraint is satisfied: $|F_1| \geq |F_2| \geq \ldots \geq |F_{n-1}| \geq |F_n|$. Let $b = |F_1|$ and $k = |F_2|$.
2. Prepare a table containing n columns and $b \times k$ rows divided into b blocks. Label the columns as $F_1, F_2, \ldots, F_n$. Each block contains k rows.
3. Fill column F_1 with 1s in block 1, 2s in block 2 and so on. Fill block 1 of columns F_2 with the sequence 1,2,. . . ,k.
4. Find $s = n(k)$ MOLS of order k. Denote them as $M_1, M_2, \ldots, M_s$. Note that $s < k$ for $k > 1$.
5. Fill block 1 of column F_3 with entries from column 1 of M_1, block 2 with entries from column 2 of M_1, and so on. If the number of blocks $b = b_1 > k$ then reuse columns of M_1 to fill rows in the remaining $b_1 - k$ blocks. Repeat the procedure for the remaining columns. If $s < (n-2)$ then fill columns by randomly selecting the values.
6. Generate the test set from the rows of the resulting filled table.

PDMOLS and combination constraints

In most real cases it is not meaningful/possible to use all the possible tests generated according to PDMOLS.

- If the factor X assumes level *x* than factor Y cannot assume level *y*

The AGTCS system

Factor	Levels			
F'_1:Hardware (H)	PC	Mac		
F'_2:OS (O)	Win2000	Win XP	OS9	OS10
F'_3:Browser(B)	Explorer	Netscape 4.x	Firefox	Chrome
F'_4:PI(P)	New	Existing		

How to handle constraints

- The “PC” level is incompatible with “OSx” families.
- The “Mac” level is incompatible with “Win OS” families.
- there are invalid levels

PDMOLS and combination constraints

In most real cases it is not meaningful/possible to use all the possible tests generated according to PDMOLS.

- If the factor X assumes level *x* than factor Y cannot assume level *y*

The AGTCS system

Factor	Levels			
F_1':Hardware (H)	PC	Mac		
F_2':OS (O)	Win2000	Win XP	OS9	OS10
F_3':Browser(B)	Explorer	Netscape 4.x	Firefox	Chrome
F_4':PI(P)	New	Existing		

How to handle constraints

- The “PC” level is incompatible with “OSx” families.
- The “Mac” level is incompatible with “Win OS” families.
- there are invalid levels

Consider a system that needs to be tested according to possible configurations given by the combination of 6 different factors each one constituted by the following levels:

- $A = \{a_1, a_2, a_3, a_4\}$
- $B = \{b_1, b_2, b_3\}$
- $C = \{c_1, c_2, c_3, c_4\}$
- $D = \{d_1, d_2, d_3, d_4\}$
- $E = \{e_1, e_2, e_3\}$
- $F = \{f_1, f_2, f_3\}$

Derive a test set according to the pairwise design using the most suitable approach among the ones presented in the course. In the generation consider that there are some constraints that have to be respected:

- factors D, E, F are strongly interrelated factors and among all the possible configurations that are theoretically possible only the following 3 should be considered as real (d_1, e_1, f_2), (d_2, e_2, f_1), (d_3, e_3, f_2).
- for factors A and B the levels a_4 and b_3 cannot be assumed together

MOLS shortcomings

- A sufficient number of MOLS might not exist for the problem at hand
- MOLS assist with the generation of balanced design but the number of configuration could be larger than necessary

To address such issues other approaches have been proposed:

- Orthogonal Arrays (and variants)
- Covering Arrays

Orthogonal Arrays

Definition

An Orthogonal Array is an $N \times k$ matrix in which the entries are from a finite set S of s symbols such that any $N \times t$ subarray contains each t-uple exactly the same number of times. Such an orthogonal array is denoted by $OA(N, k, s, t)$. The index of an orthogonal array, denoted by λ, is equal to N/s^t.

Example

Run	F_1	F_2	F_3
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1

Orthogonal arrays assume that each factor assumes values from the same set of s values. This is not generally the case and Mixed Level Orthogonal Arrays can be used in such contexts.

Orthogonal Arrays

Definition

An Orthogonal Array is an $N \times k$ matrix in which the entries are from a finite set S of s symbols such that any $N \times t$ subarray contains each t-uple exactly the same number of times. Such an orthogonal array is denoted by $OA(N, k, s, t)$. The index of an orthogonal array, denoted by λ, is equal to N/s^t.

Example

Run	F_1	F_2	F_3
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1

Orthogonal arrays assume that each factor assumes values from the same set of s values. This is not generally the case and Mixed Level Orthogonal Arrays can be used in such contexts.

Covering Arrays

A Covering Array, denoted as $CA(N, k, s, t)$ is an $N \times k$ matrix in which entries are from a finite set S of s symbols such that each $N \times t$ subarray contains each possible t-uple at least λ times. In this case we have an unbalanced design.

Run	F_1	F_2	F_3	F_4	F_5
1	1	1	1	1	1
2	2	1	1	2	2
3	1	2	1	2	1
4	1	1	2	1	2
5	2	2	1	1	2
6	2	1	2	2	1
7	1	2	2	2	2
8	2	2	2	1	1

Run	F_1	F_2	F_3	F_4	F_5
1	1	1	1	1	1
2	2	2	1	2	1
3	1	2	2	1	2
4	2	1	2	2	2
5	2	2	1	1	2
6	1	1	1	2	2

Generation of Covering Arrays

IPO Procedure permits the derivation of mixed-level covering arrays for pairwise designs.

Summary from test generation strategies

- Generation from requirements
- Generation from formal models
- Generation using combinatorial design