Pairwise/All-Pairs Testing
Combinatorial Interaction Test Case Generation

Consider 3 different scenarios:

1

Web Combinations:

Browsers	8
Plug-ins	3
Client OS	6
Servers	3
Server OS	3

1296 combos:

8 × 3 × 6 × 3 × 3 =
1296

2

Bank Combinations:

Customer Types	4
Account Types	5
States	6

120 combos:

4 × 5 × 6 =
120

3

OO Combinations:

Senders	4
Parameters	5
Receivers	3

60 combos:

4 × 5 × 3 =
60

Each of these different scenarios has large combinations:

- That should be tested
- That may be risky if untested
- That we may not have the resources to construct and run all the tests (too many)

We must select a reasonably sized subset that we can test given our resource constraints. We must choose a specially selected, fairly small subset that finds a great many defects - more than you would expect from such a subset.

Note:
Random selection can be a very good approach to choosing a subset but most people have a difficult time choosing truly randomly.

We should not attempt to test all the combinations for all the values for all the variables but to test **all pairs** of variables. This significantly reduces the number of tests that must be created and run.

These examples exhibit the significant reduction in test effort:

- A system has 4 different input parameters, and each one can take on one of 3 different values.
  Number of combos: 3^4 = 81
  Cover all the pairwise input combos in only 9 tests.
- A system has 13 different input parameters, and each one can take on one of 3 different values.
  Number of combos: 3^13 = 1,594,323
  Cover all the pairwise input combos in only 15 tests.
- A system has 20 different input parameters, and each one can take on one of 10 different values.
  Number of combos: 10^{20}
  Cover all the pairwise input combos in only 180 tests.

A hypothesis behind why pairwise testing works is that most defects are either single-mode defects (the function under test simply does not work and any test of that function would find the defect) or they are double-mode defects (it is the pairing of this function/module that fails, even though all other pairings perform properly). Pairwise testing defines a minimal subset that guides us to test for all single-mode and double-mode defects.

https://ieeexplore.ieee.org/abstract/document/5676343
Combinatorial testing (also called interaction testing) is an effective specification-based test input generation technique. By now most of the research work in combinatorial testing aims to propose novel approaches trying to generate test suites with minimum size that still cover all the pairwise, triple, or n-way combinations of factors. Since the difficulty of solving this problem is demonstrated to be NP-hard, existing approaches have been designed to generate optimal or near optimal combinatorial test suites in polynomial time.

Comparison of efficiency

The basic measure of efficiency of a pairwise test generation tool is the number of tests a tool generates given some size of input. For example, when the input has four parameters with 3 values each, denoted 3^4 in the table below, tools create between 9 and 11 test cases. All of the test suites meet the pairwise testing criterion of covering each pair of values in at least one test case; however some tools can pack all these combinations into fewer tests. This may not matter for small problems, test suites for large test domains can exhibit bigger differences.

Test generation efficiency is one aspect of a tool that a user will want to consider when choosing a tool to use, but numbers are easier to compare than less tangible aspects like “usability”, so a standard set of benchmarks emerged over the years. The table below summarizes efficiencies for several tools that happened to publish their numbers.

The table summarizes efficiencies for several tools:

#	Model	3^4	3^{13}	$4^{15}\,3^{17}\,2^{29}$	$4^1\,3^{39}\,2^{35}$	2^{100}	10^{20}	Source
1	AETG	9	15	41	28	10	180	Y. Lei and K. C. Tai In-parameter-order: a test generation strategy for pairwise testing, p. 8.
2	IPO	9	17	34	26	15	212	K. C. Tai and Y. Lei A Test Generation Strategy for Pairwise Testing p. 2.
3	TConfig	9	15	40	30	14	231	A. W. Williams Determination of Test Configurations for Pair-wise Interaction Coverage, p. 15.
4	CTS	9	15	39	29	10	210	A. Hartman and L. Raskin Problems and Algorithms for Covering Arrays, p. 11.
5	Jenny	11	18	38	28	16	193	Supplied by Bob Jenkins.
6	TestCover	9	15	29	21	10	181	Supplied by George Sherwood.
7	DDA	?	18	35	27	15	201	C. J. Colbourn, M. B. Cohen, R. C. Turban A Deterministic Density Algorithm for Pairwise Interaction Coverage, p. 6.
8	AllPairs [McDowell]	9	17	34	26	14	197	Supplied by Bob Jenkins.
9	PICT	9	18	37	27	15	210	Supplied by Jacek Czerwonka.
10	EXACT	9	15	?	21	10	?	J. Yan, J. Zhang Backtracking Algorithms and Search Heuristics to Generate Test Suites for Combinatorial Testing, p. 8.
11	IPO-s	9	17	32	23	10	220	A. Calvagna, A. Gargantini IPO-s: Incremental Generation of Combinatorial Interaction Test Data Based on Symmetries of Covering Arrays, p. 17.
12	ecFeed	10	19	37	28	16	203	Supplied by Patryk Chamuczynski.
13	JCUnit	10	23	49	33	18	245	Supplied by Hiroshi Ukai link.
14	CoverTable	9	17	34	26	12	195	CoverTable's webpage.

Another more extensive list of pairwise testing tools: https://www.pairwise.org/tools.html.

Links:
https://www.pairwise.org/
http://neilsloane.com/oadir/
http://support.sas.com/techsup/technote/ts723b.pdf
http://support.sas.com/techsup/technote/ts723_Designs.txt
Testing methodology details: https://www.pairwise.org/
Microsoft PICT tool: https://github.com/microsoft/pict

- PICT Pairwise Test Generation tool

Orthogonal Arrays

Orthogonal array testing is a black box testing technique that is a systematic, statistical way of software testing. It is used when the number of inputs to the system is relatively small, but too large to allow for exhaustive testing of every possible input to the systems.It is particularly effective in finding errors associated with faulty logic within computer software systems. Orthogonal arrays can be applied in user interface testing, system testing, regression testing, configuration testing and performance testing. The permutations of factor levels comprising a single treatment are so chosen that their responses are uncorrelated and therefore each treatment gives a unique piece of information. The net effects of organizing the experiment in such treatments is that the same piece of information is gathered in the minimum number of experiments.

- The origin of orthogonal arrays can be traced back to mathematician Euler in the guide of Latin squares.
- Genichi Taguchi popularized their use in hardware testing.
- Madhav Phadke wrote an excellent book *QA Engineering Using Robust Design.*

Notes:
As a tester, you do not have to create orthogonal arrays, all you must do is locate one of proper size. Books, websites, and automated tools will help you do this.

Reference:
Neil J.A. Sloane maintains a comprehensive catalog of orthogonal arrays.

Tool:
The rdExpert tool from Phadke Associates implements the orthogonal array approach. See http://phadkeassociates.com/.