Journal of Uncertain Systems
Vol.8, No.3, pp.233-238, 2014
Online at: www.jus.org.uk

How to Faster Test a Device for Different Combinations of Parameters

Francisco Zapata[1], Luis Gutierrez[2], Vladik Kreinovich[2,*]
[1] *Department of Industrial and Systems Engineering, University of Texas at El Paso El Paso, TX 79968, USA*
[2] *Department of Computer Science, University of Texas at El Paso, El Paso, TX 79968, USA*

Received 27 May 2013; Revised 1 August 2013

Abstract

A device has to function properly under all possible conditions: e.g., for all temperatures within a given range, for all possible humidity values within a given range, etc. Ideally, it would be nice to be able to test a device for all possible combinations of these parameters, but the number of such combinations is often so huge that such an exhaustive testing is not possible. Instead, it is reasonable to check the device for all possible values of each parameter, for each possible pairs of values of two parameters, and, in general, for all possible combinations of values of k parameters for some k. For n parameters, a straightforward testing design with this property contains $O(n^k)\cdot N^k$ experiments, where N is the number of tested values of each parameter. We show that, by using a more sophisticated testing design, we can decrease the number of experiments to a much smaller number $O(\log^{k-1}(n))\cdot N^k$.

Keywords: system testing, pairwise testing, triple testing

1 Formulation of the Problem

It is important to test a device for different combinations of parameters. Many devices have to function correctly under many different values of the corresponding parameters: e.g., for temperatures within the given range, for pressure within the given range, for humidity within the given range, etc.

It is not possible to test all possible combinations of parameters. Ideally, we should test the device for all possible combinations of the corresponding parameters. However, often, such a testing is not realistic. For example, if we have 20 possible parameters, and we consider 10 possible values of each of these parameters, then testing all possible combinations would require an unrealistic amount of 10^{20} tests. Even in the idealized situation when each test takes 1 second, then, with $3\cdot 10^7$ seconds in a year, this testing would require $3\cdot 10^{12}$ years – longer than the lifetime of the Universe.

Solution: test for all pairs, or all triples, etc. Since we cannot test for all possible combinations of all the parameters, we need to test at least for all possible values of each parameter separately. In other words, we need to test the device for all possible values of outside temperature, then test this device for all possible values of humidity, etc.

In this testing, we may overlook possible joint effect of two or more different parameters. To take such an effect into account, it makes sense to arrange the tests in such a way that for every two parameters, we test all possible combinations of values. Similarly, we may want to test in such a way that for every three parameters, we test all possible combinations of values, etc.; see, e.g., [1, 2, 3, 4].

[*]Corresponding author.
Emails: fazg74@gmail.com (F. Zapata), lcgutierrez@miners.utep.edu (L. Gutierrez), vladik@utep.edu (V. Kreinovich).

How to arrange such a test: first simple idea. For each parameter x_i, we have a range $[\underline{x}_i, \overline{x}_i]$ of possible values. Let us assume that for each parameter, we test for N different values $x_{i1} < x_{i2} < \ldots < x_{iN}$. In this case, we need N experiments to test the device's behavior for all N values of each parameter.

If we simply want to test for all possible values of each parameter, then a straightforward idea is to first test all possible values of the first parameter x_1, then test all possible values of the second parameter x_2, etc., until we have tested all the parameters. If we denote by n the number of parameters, then this scheme requires $n \cdot N$ experiments.

If we want to test all possible pairs of parameters, then, for each of $\binom{n}{2}$ pairs of parameters, we test all possible N^2 pairs of values. This requires $\binom{n}{2} \cdot N^2$ experiments.

Similarly, if we want to test all possible triples of parameters, then for each of $\binom{n}{3}$ triples of parameters, we test all possible N^3 triples of values. This requires $\binom{n}{3} \cdot N^3$ experiments. In general, if we fix an integer k, and we want to test all possible combinations of values of each k parameters, then for each of $\binom{n}{k}$ k-tuples of parameters, we test all possible N^k combinations of values. This requires $\binom{n}{k} \cdot N^k$ experiments.

We can test faster than that. It is easy to see that the simple straightforward approach uses too many combinations of parameters, we can often use much fewer experiments.

For example, if we want to test all possible values of each parameter, then in the above straightforward approach, we perform $n \cdot N$ experiments. In reality, it is sufficient to perform only N experiments. Namely, in each experiment $j = 1, \ldots, N$, we take each parameter x_i to be equal to x_{ij}:

- in the first experiment, we select the first value of each of n parameters, i.e., use parameters $(x_{11}, \ldots, x_{n1})$;
- in the second experiment, we select the second value of each of n parameters, i.e., use parameters $(x_{12}, \ldots, x_{n2})$;
- ...
- in the j-th experiment, we select the j-th value of each of n parameters, i.e., use parameters $(x_{1j}, \ldots, x_{nj})$;
- ...
- finally, in the last (N-th) experiment, we select the N-th value of each of n parameters, i.e., use parameters $(x_{1N}, \ldots, x_{nN})$.

When we have many parameters $n \gg 1$, we then have $n \cdot N \gg N$, so this idea drastically decreases the number of necessary experiments – and thus, the testing time.

Comment. In some cases, we have different number of values N_i for different parameters x_i. In this case, in the straightforward approach, we need $\prod\limits_{i=1}^{n} N_i$ combinations, but instead, we can simply use $N = \max(N_i)$ combinations: namely, we set $x_{ij} = x_{iN_i}$ when $j > N_i$. Thus, when all the values N_i are of the same order, we still get a drastic decrease in the number of experiments.

What we do in this paper. In this paper, we show that similar faster testing is possible when we test all possible pairs of parameters, all possible triples, etc.

2 New Testing Design: Main Idea and Step-by-Step Description

Let us formulate the problem in precise terms. The above description leads to the following definition.

Definition. *Let $n > 0$, $N > 0$, and $k > 0$ be positive natural numbers. The number n will be called the* number of parameters, *and the number N will be called the* number of values.

- *By an* experiment, *we mean a tuple of n integers $j_1, \ldots, j_n$, where $1 \le j_i \le N$ for all i. We say that in this experiment, we use the j_i-th value of the i-th parameter. An experiment will also be denoted by $(x_{1i_1}, \ldots, x_{nj_n})$.*
- *By a* testing design, *we mean a finite set of experiments.*
- *We say that a testing design* tests each combination of k parameters *if for every k-tuple $1 \le i_1 < \ldots < i_k \le N$ and for all k-tuples of integers $v_1, \ldots, v_k$, with $1 \le v_\ell \le N$, this testing design contains an experiment in which, for all ℓ from 1 to k, we use the v_ℓ-th value of the i_ℓ-th parameter.*

Main objective. Our main objective is minimize the required number of experiments.

The straightforward ideas leads to a design that tests each combination of k parameters and that consists of $\binom{n}{k} \cdot N^k$ experiments. As a function of the number n of parameters, this number of experiments is $O(n^k) \cdot N^k$.

For $n = k$, we need to test all N^k possible combinations of parameters, so we cannot have fewer than N^k tests anyway. However, as the above case of $k = 1$ shows, we can try to minimize the factor depending on n.

Main Result. *For each k, there exists a testing design that tests each combination of k parameters and that consists of $O(\log^{k-1}(n)) \cdot N^k$ experiments.*

Discussion.

- For $k = 1$, we get the known fact that we need $O(N)$ experiments.
- For testing all possible pairs ($k = 2$), we need $O(\log(n)) \cdot N^2$ experiments. This is much smaller than $O(n^2) \cdot N^2$ experiments needed in the straightforward approach.
- For testing all possible triples ($k = 3$), we need $O(\log^2(n)) \cdot N^3$ experiments. This is much smaller than $O(n^3) \cdot N^3$ experiments needed in the straightforward approach.

Description of the new testing design: case of $k = 2$. Let $B = \lceil \log_2(n) \rceil \sim \log(n)$ be the number of bits needed to describe all the natural numbers from 0 to $n - 1$. Let us enumerate the bit from lowest to the highest. Let us denote the b-th bit in the binary expansion of an integer i by $\text{bit}_b(i)$. For example, for the binary number $i = 1011_2 = 11_{10}$:

- the first (lowest) bit is 1: $\text{bit}_1(i) = 1$;
- the second bit is 1: $\text{bit}_2(i) = 1$;
- the third bit is 0: $\text{bit}_3(i) = 0$;
- the fourth bit is 1: $\text{bit}_4(i) = 1$; and
- all the other bits are 0s: $\text{bit}_b(i) = 0$ for all $b > 4$.

Our new testing design consists of B groups of experiments. Each of these groups consists of N^2 experiments, so that total number of experiments is indeed $O(\log(n)) \cdot N^2$. In the b-th group of experiments, for each pair of integers (f, s), $1 \le f \le N$ and $1 \le s \le n$, we set:

- $j_i = f$ if $\text{bit}_b(i - 1) = 0$, and
- $j_i = s$ if $\text{bit}_b(i - 1) = 1$.

If we have two different integers $i_1 < i_2$, then $i_1 - 1 \ne i_2 - 1$, so at least one bit b in the binary expansions of $i_1 - 1$ and $i_2 - 1$ is different. Thus, for this bit b, the corresponding group of experiments tests all possible pairs (f, s).

To make the testing design clear, let us illustrate it on three examples: $n = 2$, $n = 4$, and $n = 8$.

First example: $n = 2$. For $n = 2$, we need $B = 1$ bit to represent integers 0 and 1. Thus, in this case, it is sufficient to have a single group of experiments, in which, for each pair (s, f), we set $x_1 = f$ and $x_2 = s$. In other words, each experiment has the form (s, f).

Second example: $n = 4$. For $n = 4$, we need $B = 2$ bits to represent integers 0, 1, 2, and 3. Here, $0_{10} = 00_2$, $1_{10} = 01_2$, $2_{10} = 10_2$, and $3_{10} = 11_2$. Thus, in this case, we have two groups of N^2 experiments each:

- In the first group of experiments, we assign s to all the values i for which $\text{bit}_1(i-1) = 0$, and f to all the values i for which $\text{bit}_1(i-1) = 1$. Thus, each experiment has the form (f, s, f, s).
- In the second group of experiments, we assign s to all the values i for which $\text{bit}_2(i-1) = 0$, and f to all the values i for which $\text{bit}_2(i-1) = 1$. Thus, each experiment has the form (f, f, s, s).

If $i_1 < i_2$ are both odd or both even, then the second group of experiments tests all possible combinations of the values of the corresponding parameters. If one of the values i_1 and i_2 is odd and another value is even, then the first group of experiments tests all possible combinations of values.

Third example: $n = 8$. For $n = 8$, we need $B = 3$ bits to represent integers from 0 to 7. Thus, in this case, we have three groups of N^2 experiments each:

- In the first group of experiments, each experiment has the form (f, s, f, s, f, s, f, s).
- In the second group of experiments, each experiment has the form (f, f, s, s, f, f, s, s).
- In the third group of experiments, each experiment has the form (f, f, f, f, s, s, s, s).

Description of the new testing design: case of $k > 2$. To describe the testing design for $k > 2$, we use the following recursive algorithm that reduces a testing design for given k and n to a testing designs for smaller k and n.

For $n = k$, we just have to test all N^k possible combinations of values of all k parameters.

For $n > k$, we divide the set of n parameters into two halves of size $n/2$. Then:

- To cover situations when all k parameters are in the first half and situations when all k parameters are in the second half, we use the testing design for $n/2$ and k; each experiment in this design is copied for the second half, so, e.g., a design fs becomes $fsfs$ (see example below).
- To cover situations in which $k-1$ parameters are in the first half and 1 parameter is in the second half, we combine each experiment from testing plan for $n/2$ and $k-1$ with each experiment from the testing plan for $n/2$ and 1.
- To cover situations in which $k-2$ parameters are in the first half and 2 parameters are in the second half, we combine each experiment from testing plan for $n/2$ and $k-2$ with each experiment from the testing plan for $n/2$ and 2.
- ...
- To cover situations in which $k-i$ parameters are in the first half and i parameters are in the second half, we combine each experiment from testing plan for $n/2$ and $k-i$ with each experiment from the testing plan for $n/2$ and i.
- ...
- Finally, to cover situations in which 1 parameter are in the first half and $k-1$ parameters are in the second half, we combine each experiment from testing plan for $n/2$ and $k-1$ with each experiment from the testing plan for $n/2$ and 1.

Proof that this algorithm requires $O(\log_2^k(n)) \cdot N^k$ experiments. Let us prove, by induction over k, that this algorithm indeed requires $O(\log_2^k(n)) \cdot N^k$ experiments. We already know that this is true for $k = 1$ and $k = 2$. Let us prove assume that this property holds for $1, 2, \ldots, k-1$. Then, according to the algorithm, the total number of experiments $E_k(n)$ for n and k consists of:

- the total number of experiments $E_k(n/2)$ for $n/2$ and k;
- the total number of experiments $E_{k-1}(n/2)$ for $n/2$ and $k-1$ multiplied by the total number of experiments $E_1(n/2)$ for $n/2$ and 1;
- $\ldots$
- the total number of experiments $E_{k-i}(n/2)$ for $n/2$ and $k-i$ multiplied by the total number of experiments $E_i(n/2)$ for $n/2$ and i;
- $\ldots$
- the total number of experiments $E_1(n/2)$ for $n/2$ and 1 multiplied by the total number of experiments $E_{k-1}(n/2)$ for $n/2$ and $k-1$.

In other words,

$$E_k(n) = E_k(n/2) + \sum_{i=1}^{k-1} E_{k-i}(n/2) \cdot E_i(n/2). \tag{1}$$

By induction, we know that for some constant C_{k-1}, for all $i \le k-1$, we have $E_i(n/2) \le C_{k-1} \cdot \log_2^{i-1}(n/2) \cdot N^i$. Since $\log_2(n/2) \le \log_2(n)$, we get $E_i(n/2) \le C_{k-1} \cdot \log^{i-1}(n) \cdot N^i$. Thus,

$$E_{k-i}(n/2) \cdot E_i(n/2) \le C_{k-1}^2 \cdot \log_2^{k-i-1}(n) \cdot N^{k-i} \cdot \log_2^{i-1}(n) \cdot N^i = C_{k-1}^2 \cdot \log_2^{k-2}(n) \cdot N^k. \tag{2}$$

Thus, the sum of $k-1$ such products is bounded by $(k-1) \cdot C_{k-1}^2 \cdot \log_2^{k-2}(n) \cdot N^k$. So, from (1), we conclude that

$$E_k(n) \le E_k(n/2) + (k-1) \cdot C_{k-1}^2 \cdot \log_2^{k-2}(n) \cdot N^k. \tag{3}$$

Similarly, we get

$$E_k(n/2) \le E_k(n/4) + (k-1) \cdot C_{k-1}^2 \cdot \log_2^{k-2}(n) \cdot N^k,$$
$$E_k(n/4) \le E_k(n/8) + (k-1) \cdot C_{k-1}^2 \cdot \log_2^{k-2}(n) \cdot N^k,$$

etc. Every time we decrease n in half, we add a term $(k-1) \cdot C_{k-1}^2 \cdot \log_2^{k-2}(n) \cdot N^k$. In $\le \log_2(n)$ steps, we get from n to k, with $E_k(k) = N^k$. Thus, $E_k(n)$ can be bounded by adding N^k and $\le \log_2(n)$ terms of the type $(k-1) \cdot C_{k-1}^2 \cdot \log_2^{k-2}(n) \cdot N^k$:

$$E_k(n) \le [1 + \log_2(n) \cdot (k-1) \cdot C_{k-1}^2 \cdot \log_2^{k-2}(n)] \cdot N^k \le C \cdot \log_2^{k-1}(n) \cdot N^k.$$

The statement is proven.

To make the algorithm clearer, let us illustrate it on two examples: $n = 4$ and $k = 3$ and $n = 8$ and $k = 3$.

First example: $n = 4$ and $k = 3$. Let us describe all the experiments corresponding to $n = 4$ and $k = 3$. The corresponding testing design consists of the following experiments:

- First, according to our algorithm, we should list all the experiments corresponding to $n/2 = 2$ and $k = 3$; since in this case, $k < n/2$, there are no such experiments: it is not possible to have two parameters and test all possible values of three of them.
- Then, we combine each experiment with $n = 2$ and $k = 2$ with each experiment with $n = 2$ and $k = 1$. As we have mentioned earlier, there is one group of experiments with $n = 2$ and $k = 2$: sf, with s and f going from 1 to N; and one group corresponding to $n = 2$ and $k = 1$: tt, with t from 1 to N. Thus, by combining them, we get experiments of the type $sftt$.
- Finally, we combine each experiment with $n = 2$ and $k = 1$ with each experiment with $n = 2$ and $k = 2$. We have already described all constituent experiments, so, by combining them, we get experiments of the type $ttsf$.

As a result, we get two groups of experiments: $sftt$ and $ttsf$.

Second example: $n = 8$ and $k = 3$. According to the algorithm, the corresponding testing design consists of the following experiments:

- First, according to our algorithm, we should list all the experiments corresponding to $n/2 = 4$ and $k = 3$, and repeat each for the second half as well. Thus, from the experiments $sftt$ and $ttsf$, we get
$$sfttsftt \text{ and } ttsfttsf.$$
- Then, we combine each experiment with $n = 4$ and $k = 2$ with each experiment with $n = 4$ and $k = 1$. As we have mentioned earlier, there are two groups of experiments with $n = 4$ and $k = 2$: $sfsf$ and $ssff$, and one group corresponding to $n = 4$ and $k = 1$: $tttt$. Thus, by combining them, we get experiments of the type
$$sfsftttt \text{ and } ssfftttt.$$
- Finally, we combine each experiment with $n = 4$ and $k = 1$ with each experiment with $n = 4$ and $k = 2$. We have already described all constituent experiments, so, by combining them, we get experiments of the types
$$ttttsfsf \text{ and } ttttssff.$$

Totally, we have 6 groups of N^3 experiments.

Acknowledgments

This work was supported in part by the Nati+onal Science Foundation grants HRD-0734825 and HRD-1242122 (Cyber-ShARE Center of Excellence) and DUE-0926721, by Grants 1 T36 GM078000-01 and 1R43TR000173-01 from the National Institutes of Health, and by a grant N62909-12-1-7039 from the Office of Naval Research.

The authors are greatly thankful to Dr. Ricardo Pineda for valuable suggestions.

References

[1] Black, R., *Managing the Testing Process*, Wiley, New York, 2009.

[2] Kuhn, D.R., Lei, Y., and R. Kacker, Practical combinatorial testing: beyond pairwise, *IEEE IT Professional*, vol.10, no.3, pp.19–23, 2008.

[3] Pries, K.H., and J.M. Quigley, *Testing Complex and Embedded Systems*, CRC Press, Boca Raton, Florida, 2010.

[4] Reorda, M.S., Peng, Z., and M. Violante (eds.), *System-Level Test and Validation of Hardware/Software Systems*, Springer Verlag, London, 2005.