3-Way Software Testing with Budget Constraints

Soumen MAITY(a), Member

SUMMARY In most software development environments, time, computing and human resources needed to perform the testing of a component is strictly limited. In order to deal with such situations, this paper proposes a method of creating the best possible test suite (covering the maximum number of 3-tuples) within a fixed number of test cases.

key words: covering arrays, test case generation, finite fields, linear group

1. Introduction

Given the different input parameters with multiple possible values for each parameter, performing exhaustive testing which tests all possible input combinations is practically impossible. A test case is an input combination. The most common term for a collection of test cases is a test suite. New approaches are required to generate test suites that are substantially smaller than exhaustive test suite but highly effective at detecting faults. Pair-wise or 2-way interaction testing is known for its effectiveness in different types of software testing [4], [9]–[14]. However, software failures may be caused by interactions of more than two parameters. A recent NIST study indicates that failures can be triggered by interactions up to 6 parameters [5]. Increased coverage leads to a higher level of confidence but the number of test cases may increase rapidly as the degree of interactions increases. A practical limitation in the realm of testing is budget. To model this situation, we consider the problem of creating the best possible test suite (covering the maximum number of 3-tuples) within a fixed number of test cases.

The coverage measure \( \mu \) of a (covering) array \( CA(n, k, s) \) is defined by the ratio between the number of distinct 3-tuples covered in the array and the total number of distinct 3-tuples \( \binom{s}{3} \). For example, the array with parameters \( n = 9, k = 5 \) and \( s = 2 \) shown below

\[
\begin{pmatrix}
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0
\end{pmatrix}
\]

has coverage measure \( \frac{2 \times 8 \times 6 \times 7}{\binom{5}{3} \times 2} = \frac{74}{80} \). Because here we are looking to cover 80 3-way interactions but the total interactions covered is 74. These four choices of rows (1, 2, 3), (1, 2, 4), (1, 3, 4), (2, 3, 4) contain all eight possible values of the three parameters and remaining six choices contain only seven possible values of the three parameters. Given fixed values of \( n, k, \) and \( s \), the covering arrays with budget constraints problem is to construct a covering array \( CA(n, k, s) \) having large coverage measure \( \mu \).

2. The Group Construction

Let \( F \) be a Galois field \( GF(s) \) where \( s = p^n \) and \( p \) be a

LETTER Special Section on Software Reliability Engineering

Soumen MAITY(a), Member

SUMMARY In most software development environments, time, computing and human resources needed to perform the testing of a component is strictly limited. In order to deal with such situations, this paper proposes a method of creating the best possible test suite (covering the maximum number of 3-tuples) within a fixed number of test cases.

key words: covering arrays, test case generation, finite fields, linear group

1. Introduction

Given the different input parameters with multiple possible values for each parameter, performing exhaustive testing which tests all possible input combinations is practically impossible. A test case is an input combination. The most common term for a collection of test cases is a test suite. New approaches are required to generate test suites that are substantially smaller than exhaustive test suite but highly effective at detecting faults. Pair-wise or 2-way interaction testing is known for its effectiveness in different types of software testing [4], [9]–[14]. However, software failures may be caused by interactions of more than two parameters. A recent NIST study indicates that failures can be triggered by interactions up to 6 parameters [5]. Increased coverage leads to a higher level of confidence but the number of test cases may increase rapidly as the degree of interactions increases. A practical limitation in the realm of testing is budget. To model this situation, we consider the problem of creating the best possible test suite (covering the maximum number of 3-tuples) within a fixed number of test cases.

The coverage measure \( \mu \) of a (covering) array \( CA(n, k, s) \) is defined by the ratio between the number of distinct 3-tuples covered in the array and the total number of distinct 3-tuples \( \binom{s}{3} \). For example, the array with parameters \( n = 9, k = 5 \) and \( s = 2 \) shown below

\[
\begin{pmatrix}
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0
\end{pmatrix}
\]

has coverage measure \( \frac{2 \times 8 \times 6 \times 7}{\binom{5}{3} \times 2} = \frac{74}{80} \). Because here we are looking to cover 80 3-way interactions but the total interactions covered is 74. These four choices of rows (1, 2, 3), (1, 2, 4), (1, 3, 4), (2, 3, 4) contain all eight possible values of the three parameters and remaining six choices contain only seven possible values of the three parameters. Given fixed values of \( n, k, \) and \( s \), the covering arrays with budget constraints problem is to construct a covering array \( CA(n, k, s) \) having large coverage measure \( \mu \).

2. The Group Construction

Let \( F \) be a Galois field \( GF(s) \) where \( s = p^n \) and \( p \) be a
prime. Let \( H(s) \) be the set of all functions of the form \( x \mapsto ax + b \), \( a \neq 0 \), where \( a, b \) belong to \( GF(s) \). It is easy to verify that \( H(s) \) is a group with respect to functional composition and its action on \( GF(s) \) is sharply 2-transitive. \( H(s) \) is called linear group and the order of \( H(s) \) is clearly \( s(s - 1) \). For the undefined terms and more details we refer the reader to Robinson [6]; Chapter 7.

2.1 The Group Construction of Covering Arrays with Coverage Measure One

Given fixed values of \( n, k \), and prime power \( s \), we are to construct a covering array \( 3-CA(n, k, s) \) having coverage measure \( \mu = 1 \). The elements of \( GF(s) \) are the symbols of covering array. Group construction involves selecting a group \( G \) and finding a vector \( v \in GF(s)^k \), called a starter vector. We use this vector to form a circulant matrix \( M \). We take \( G = H(s) \). The group \( H(s) \) acting on the matrix \( M \) produces \( |H(s)| = s(s - 1) \) matrices which are concatenated horizontally to form a covering array. Often we need to add a small array \( C \) to complete the covering conditions. If \( M \) has the property that every \( 3 \times k \) subarray contains at least one representative from each non-constant orbit of \( H(s) \) acting on \( 3 \)-tuples from \( GF(s) \), then the associated vector is called starter vector with respect to \( H(s) \). If \( v \) is a starter vector and \( k[H(s)] + s \leq n \), then we get a covering array \( 3-CA(n, k, s) \) with coverage measure \( \mu = 1 \). This group construction follows the technique used in [7]. We show an example to explain the method.

Example 1: Let \( n = 33 \), \( k = 5 \) and \( s = 3 \). Let \( G = H(3) \), \( F = GF(3) \) and a starter vector \( v = (1 2 1 0 0) \in F^5 \). Build the following circulant matrix from \( v \).

\[
M = \begin{pmatrix}
1 & 0 & 0 & 1 & 2 \\
2 & 1 & 0 & 0 & 1 \\
1 & 2 & 1 & 0 & 0 \\
0 & 1 & 2 & 1 & 0 \\
0 & 0 & 1 & 2 & 1
\end{pmatrix}
\]

The elements of \( H(3) = \{x \mapsto ax + b \mid a, b \in GF(3), a \neq 0 \} \) acting on \( M \) produce the matrices \( M_{(x)} \) by concatenating the matrices produced by group action of \( H(3) \) on \( M \).

\[
M_{(x)} = \begin{pmatrix}
0 & 2 & 2 & 0 & 1 \\
1 & 0 & 2 & 2 & 0 \\
0 & 1 & 0 & 2 & 2 \\
2 & 0 & 1 & 0 & 2 \\
2 & 2 & 0 & 1 & 0
\end{pmatrix},
M_{(x+1)} = \begin{pmatrix}
2 & 1 & 1 & 2 & 0 \\
0 & 2 & 1 & 1 & 2 \\
2 & 0 & 2 & 1 & 1 \\
1 & 2 & 0 & 2 & 1 \\
1 & 1 & 2 & 0 & 2
\end{pmatrix}
\]

where \( M_{(ax+b)} \) represents the group action of \( x \mapsto ax + b \) on \( M \). We need to add the following matrix

\[
C = \begin{pmatrix}
0 & 1 & 2 \\
0 & 1 & 2 \\
0 & 1 & 2 \\
0 & 1 & 2
\end{pmatrix}
\]

to ensure the coverage of identical 3-tuples. By horizontally concatenating the matrices \( M_{(x)}, M_{(x+1)}, M_{(x+2)}, M_{(x+3)}, M_{(x+4)}, M_{(x+5)}, M_{(x+6)}, \) and \( C \), we build a \( 3-CA(33, 5, 3) \) with coverage measure \( \mu = 1 \).

2.2 Choice of Starter Vectors

For the group action on the matrix to produce a covering array, any three rows in the matrix, \( M \), must have at least one element from every orbit of the group action of \( H(s) \) on \( GF(s) \times GF(s) \times GF(s) \). To determine which vectors will work as starters, consider the sets \( d[x, y] = \{(v_i, v_{i+x}, v_{i+y}) \mid i = 0, 1, \ldots, k - 1 \} \) for all \( x, y \) such that \( x \leq y \) and \( x + y \leq k - 2 \), where the superscripts are taken modulo \( k \). For \( v \) to be a starter vector, each set \( d[x, y] \) must contain a representative from each orbit of the group action of \( H(s) \) on \( GF(s) \times GF(s) \times GF(s) \).

Example 2: Let \( s = 3 \). The action of \( H(3) \) on 3-tuples from \( GF(3) \) has 5 orbits. These five orbits are determined by the pattern of the entries in their 3-tuples:

1. \( \{(a, a, a) \mid a \in GF(3) \} \)
2. \( \{(a, a, b) \mid a, b \in GF(3), a \neq b \} \)
3. \( \{(a, b, a) \mid a, b \in GF(3), a \neq b \} \)
4. \( \{(b, a, a) \mid a, b \in GF(3), a \neq b \} \)
5. \( \{(a, b, c) \mid a, b, c \in GF(3), a \neq b \neq c \} \)

There are total 27 ordered 3-tuples from 3 symbols. Each of the orbits 2-5 has six 3-tuples as \( |H(3)| = 6 \) and orbit 1 has three 3-tuples with all equal entries. For \( v \) to be a starter vector, each set \( d[x, y] \) must contain a representative from each of the orbits 2-5. We use this starter vector \( v \) to form a circulant matrix \( M \). In other words, \( v \) is a starter vector if and only if every \( 3 \times k \) submatrix of the associated circulant matrix \( M \) contains at least one representative from each of the orbits 2-5.

For example, the vector \( v = (1 2 1 0 0) \) given in Example 1 is a starter vector because every \( 3 \times 5 \) submatrix of \( M \) contains at least one representative from each of the orbits 2-5. Finally a covering array is obtained by horizontally concatenating the matrices produced by group action of \( H(3) \) on \( M \), along with \( C \). To see this, consider any three rows \( x_1, x_2, x_3 \). It must
Table 1  
Starter vectors and a comparison of the number of test cases produced by three different algorithmic approaches: Group Construction, AETG, and IPOG-NIST. For \( s = 4 \), the elements of \( GF(4) \) are represented as 0, 1, 2, and 3. Here, 2 stands for \( x + 1 \) and 3 stands for \( x \), and the primitive polynomial used is \( p(x) = x^2 + x + 1 \). For \( s = 8 \), the elements of \( GF(8) \) are represented as 0, 1, ..., 7. Here 2 stands for \( x \), 3 stands for \( x + 1 \), 4 stands for \( x^2 \), 5 stands for \( x^2 + 1 \), 6 stands for \( x^2 + x \), and 7 stands for \( x^2 + x + 1 \), and the primitive polynomial used is \( p(x) = x^3 + x + 1 \).

| Systems Starter Vector / Vector with Good Coverage | Group Construction | AETG | IPOG-NIST |
|-----------------------------------------------|--------------------|-------|-----------|
| \((k,s)\) | \((k,s)\) | \((k,s)\) | \((k,s)\) | \((k,s)\) | \((k,s)\) |
| \((5,3)\) | 12100 | 33 (1.0) | 28 (0.85) | 40 | 42 |
| \((5,4)\) | 12100 | 64 (1.0) | 59 (0.92) | 93 | 94 |
| \((6,3)\) | 201000 | 39 (0.94) | – | 47 | 49 |
| \((6,4)\) | 201000 | 76 (0.85) | – | 105 | 106 |
| \((7,3)\) | 2110100 | 45 (1.0) | 38 (0.85) | 50 | 52 |
| \((7,4)\) | 2312100 | 112 (0.98) | – | 141 | 146 |
| \((8,3)\) | 122101000 | 57 (1.0) | 48 (0.85) | 60 | 62 |
| \((8,4)\) | 2320100 | 100 (0.97) | – | 131 | 135 |
| \((9,3)\) | 211210000 | 124 (1.0) | 114 (0.92) | 150 | 155 |
| \((9,4)\) | 231120000 | 136 (1.0) | 125 (0.92) | 157 | 163 |
| \((10,3)\) | 122101000 | 148 (1.0) | 136 (0.92) | 166 | 170 |
| \((10,4)\) | 122101000 | 160 (1.0) | 147 (0.92) | 176 | 177 |
| \((11,3)\) | 122101000 | 172 (1.0) | 158 (0.92) | 181 | 183 |
| \((11,4)\) | 122101000 | 184 (1.0) | 169 (0.92) | 189 | 188 |
| \((12,3)\) | 122101000 | 145 (0.91) | – | 229 | 236 |
| \((12,4)\) | 122101000 | 165 (0.93) | – | 249 | 257 |
| \((13,3)\) | 122101000 | 185 (0.98) | – | 272 | 276 |
| \((13,4)\) | 122101000 | 205 (0.98) | – | 287 | 299 |
| \((14,3)\) | 122101000 | 225 (1.0) | 214 (0.95) | 308 | 312 |
| \((14,4)\) | 122101000 | 245 (1.0) | 233 (0.95) | 319 | 329 |
| \((15,3)\) | 122101000 | 265 (1.0) | 252 (0.95) | 337 | 344 |
| \((15,4)\) | 122101000 | 285 (1.0) | 271 (0.95) | 353 | 355 |
| \((16,3)\) | 122101000 | 305 (1.0) | 290 (0.95) | 365 | 367 |
| \((16,4)\) | 122101000 | 325 (1.0) | 309 (0.95) | 376 | 380 |
| \((17,3)\) | 122101000 | 345 (1.0) | 328 (0.95) | 391 | 393 |
| \((17,4)\) | 122101000 | 365 (1.0) | 347 (0.95) | 401 | 401 |
| \((18,3)\) | 122101000 | 385 (0.90) | – | 713 | NA |
| \((18,4)\) | 122101000 | 427 (0.89) | – | 766 | NA |
| \((19,3)\) | 122101000 | 469 (0.94) | – | 818 | NA |
| \((19,4)\) | 122101000 | 511 (0.95) | – | 860 | NA |
| \((20,3)\) | 122101000 | 553 (0.98) | – | 899 | NA |
| \((20,4)\) | 122101000 | 595 (0.99) | – | 941 | NA |
| \((21,3)\) | 122101000 | 637 (0.98) | – | 977 | NA |
| \((21,4)\) | 122101000 | 679 (0.97) | – | 1014 | NA |
| \((22,3)\) | 122101000 | 721 (0.97) | – | 1051 | NA |
| \((22,4)\) | 122101000 | 763 (0.96) | – | 1081 | NA |
| \((23,3)\) | 122101000 | 805 (0.97) | – | 1112 | NA |
| \((23,4)\) | 122101000 | 847 (0.96) | – | 1143 | NA |
under $H_3$-tuples. These 6 orbits are:

- Orbit 1 has four 3-tuples with all equal entries, each of the patterns $[a, a, a]^T$, $[a, a, b]^T$, $[a, b, a]^T$, and $[b, a, a]^T$ occur in these rows. The patterns with all equal entries occur on rows $x_1$, $x_2$, $x_3$ since they occur in $C$. Patterns with two equal entries and one different and patterns with three different entries appear on rows $x_1$, $x_2$, $x_3$ of the array since $H(3)$ is 3-transitive on $GF(3)$, and every $3 \times k$ submatrix of $M$ contains at least one representative from each of the orbits 2-5.

**Example 3:** Let $s = 4$. There are 64 ordered 3-tuples from 4 symbols. Orbit 1 has four 3-tuples with all equal entries, and each of the orbits 2-4 has twelve 3-tuples. Since there are $4 \times 3 \times 2 = 24$ patterns with three distinct entries and $H(4)$ is of order 12, there are two orbits of such patterns under $H(4)$. Thus, under $H(4)$ there are precisely 6 orbits of 3-tuples. These 6 orbits are:

1. $[[a, a, a]^T : a \in GF(4)]$
2. $[[a, a, b]^T : a, b \in GF(4), a \neq b]$
3. $[[a, b, a]^T : a, b \in GF(4), a \neq b]$
4. $[[b, a, a]^T : a, b \in GF(4), a \neq b]$
5. $[[a, b, c]^T : a, b, c \in GF(4), a \neq b \neq c]$
6. $[[a', b', c']^T : a', b', c' \in GF(4), a \neq b \neq c']$.

Orbits 5 and 6 are clearly partition of 24 patterns with three distinct entries. For $v$ to be a starter vector, each set $d[x, y]$ must contain a representative from each of the orbits 2-6.

**Example 4:** (General Case) Let $s$ be a prime power. The action of $H(s)$ on 3-tuples from $GF(s)$ has $s + 2$ orbits:

(a) One orbit of patterns with three equal entries
(b) Three orbits of patterns with two equal entries and one different
(c) $s - 2$ orbits of patterns with three distinct entries. The reason is this. There are $s(s - 1)(s - 2)$ patterns with three distinct entries and each orbit contains atmost $s(s - 1)$ patterns as $H(s)$ is of order $s(s - 1)$.

### 2.3 Covering Arrays with Coverage Measure Less Than One

Given $n, k$, and $s$, covering arrays with coverage measure strictly less than one are obtained in the following two cases:

(i) Starter vector exists but $k|H(s)| + s > n$; in this case we consider a subset $S \subseteq H(s)$ of maximum possible cardinality such that $|S| + s \leq n$ and the subset $S$ acting on $M$ produces $|S|$ matrices which are horizontally concatenated, along with $C$, to form a (covering) array of size less than or equal to $n$. The coverage measure of this array...
\[
\mu \geq \frac{\binom{k}{3}((s + 1)|S| + s)}{\binom{k}{3}s^3}.
\]

Note that \(\mu = 1\) when \(S = H(s)\).

(ii) Starter vector is not found. In this case we look for a vector that produces (covering) array with maximum possible coverage measure. Such vector is called vector with good coverage.

**Example 5:** Suppose we are given that \(n = 28, k = 5\) and \(s = 3\). A starter vector exists, but \(33 = k|G| + s > n = 28\). We consider a subset \(S \subseteq H(3)\) of cardinality 5 and then by concatenating the matrices \(M(a), a \in S\), and \(C\), we build a \(CA(28, 5, 3)\) with coverage measure \(\mu \geq \frac{\binom{5}{3}3^2}{\binom{5}{3}3^3} = 0.85\).

2.4 Results

We use a simple computer search to find starter vectors or vectors with good coverage. Table 1 shows starter vectors/ vectors with good coverage, the sizes of test suites generated by three different techniques, and coverage measure for \(s = 3, 4, 5, 7, 8\) and for \(k\) such that \(s + 1 \leq k \leq 3(s + 1)\). The starter vectors \(v\) are found by an exhaustive search. The data from Table 1 is shown graphically in Fig. 1. A comparison of group construction with similar tools like AETG and IPOG in Fig. 1 shows that group construction produces significantly smaller test suites as \(s\) increases.

3. Conclusions

In this paper, we have proposed group construction of strength three covering arrays with budget constraints. In order to test a software component with 14 parameters each having 7 values, the group construction can generate a test suit within 595 test cases with coverage measure 0.99. These 595 test cases ensure with probability 0.99 that failure cannot be caused due to interactions of two or three parameters whereas AETG requires 941 test cases for full coverage. The results show that the proposed method could reduce the number of test cases. Also, for full coverage, the results show that the proposed method required fewer test cases than the methods like AETG and IPOG. An interesting problem is to investigate general strategies which are capable of handling more than 3-way interactions.

References

[1] A. Hartman and L. Raskin, “Problems and algorithms for covering arrays,” Discrete Mathematics, vol.284, pp.149–156, 2004.
[2] A. Hartman, “Software and hardware testing using combinatorial covering suites,” in Graph Theory, Combinatorics and Algorithms: Interdisciplinary Applications, vol.34, pp.237–266, Kluwer Academic Publishers, 2006.
[3] C. Colbourn, “Combinatorial aspects of covering arrays,” Le Matematiche (Catania), vol.58, pp.121–167, 2004.
[4] D.M. Cohen, S.R. Dalal, M.L. Fredman, and G.C. Patton, “The AETG system: An approach to testing based on combinatorial design,” IEEE Trans. Softw. Eng., vol.23, no.7, pp.437–443, 1997.
[5] D.R. Kuhn, D.R. Wallace, and A.M. Gallo, “Software fault interactions and implications for software testing,” IEEE Trans. Softw. Eng., vol.30, no.6, pp.418–421, 2004.
[6] D.J.S. Robinson, A course in the theory of groups, Second ed., Springer, 1995.
[7] K. Meagher and B. Stevens, “Group construction of covering arrays.” J. Combinatorial Designs, vol.13, no.1, pp.70–77, 2005.
[8] M.A. Chateneuf, C.J. Colbourn, and D.L. Kreher, “Covering arrays of strength three,” Designs, Codes and Cryptography, vol.16, pp.235–242, 1999.
[9] M.B. Cohen, P.B. Gibbons, W.B. Mugridge, and C.J. Colbourn, “Constructing test suites for interaction testing,” 25th International Conference on Software Engineering, p.38, 2003.
[10] M. Forbes, J. Lawrence, Y. Lei, R.N. Kacker, and D.R. Kuhn, “Refining the in-parameter-order strategy for constructing covering arrays,” J. Research of the National Institute of Standards and Technology, vol.113, no.5, pp.287–297, 2008.
[11] R. Mandl, “Orthogonal latin squares: An application of experimental design to compiler testing,” Commun. ACM, vol.28, no.10, pp.1054–1058, 1985.
[12] S. Maity and A. Nayak, “Improved test generation algorithms for pair-wise testing,” Proc. 16th IEEE International Symposium on Software Reliability Engineering, pp.235–244, 2005.
[13] S.R. Dalal and C.L. Mallows, “Factor-covering designs for testing software,” Technometrics, vol.40, no.3, pp.234–243, 1998.
[14] Y. Lei and K.C. Tai, “In-parameter-order: A test generating strategy for pairwise testing,” IEEE Trans. Softw. Eng., vol.28, no.1, pp.1–3, 2002.