DOI: http://dx.doi.org/10.21123/bsj.2020.17.2.0575

DEO: A Dynamic Event Order Strategy for t-way Sequence Covering Array Test Data Generation

Mohammed Issam Younis

Received 14/6/2019, Accepted 5/10/2019, Published 1/6/2020

Abstract:

Sequence covering array (SCA) generation is an active research area in recent years. Unlike the sequence-less covering arrays (CA), the order of sequence varies in the test case generation process. This paper reviews the state-of-the-art of the SCA strategies, earlier works reported that finding a minimal size of a test suite is considered as an NP-Hard problem. In addition, most of the existing strategies for SCA generation have a high order of complexity due to the generation of all combinatorial interactions by adopting one-test-at-a-time fashion. Reducing the complexity by adopting one-parameter- at-a-time for SCA generation is a challenging process. In addition, this reduction facilitates the supporting for a higher strength of coverage. Motivated by such challenge, this paper proposes a novel SCA strategy called Dynamic Event Order (DEO), in which the test case generation is done using one-parameter-at-a-time fashion. The details of the DEO are presented with a step-by-step example to demonstrate the behavior and show the correctness of the proposed strategy. In addition, this paper makes a comparison with existing computational strategies. The practical results demonstrate that the proposed DEO strategy outperforms the existing strategies in term of minimal test size in most cases. Moreover, the significance of the DEO increases as the number of sequences increases and/ or the strength of coverage increases. Furthermore, the proposed DEO strategy succeeds to generate SCAs up to t=7. Finally, the DEO strategy succeeds to find new upper bounds for SCA. In fact, the proposed strategy can act as a research vehicle for variants future implementation.

Key words: Combinatorial testing, Event testing, Multi-way testing, Sequence testing, T-way testing.

Introduction:

T-way test case generation (also, termed combinatorial testing) strategies have been adopted as effective black-box testing strategies for many systems under test (SUT). In which, a subset of exhaustive test cases is generated to cover the interaction between parameters for certain strength (termed t-way strength) at least one. Combinatorial test case generation can be either sequence-less or sequence based process, the resulting test suites are called either Covering Array (CA) or Sequence (Event) Covering Array (SCA), respectively (1).

Earlier works in combinatorial testing have been reported that the generation of test cases is considered both NP-Complete (i.e., there is no unique solution for covering tuples) and NP-Hard problems (there is no unique solution that always generates minimal test size) (2,3).

Thus, many strategies exist in the literature. Some of these strategies are based on algebraic algorithms in which the test size is near minimal (2, 3). Nevertheless, search based computational strategies are also considered as competitive strategies and can produce minimal test suite for some configurations (1, 3).

The test suite generation of computational strategies involves two approaches: one-test-at-a-time and one-parameter-at-a-time. In the first approach, the generation of all t-way interaction elements (tuples) is followed by a search algorithm that generates a test case that maximizes the covered tuples. The generated test case is added to the test suite. This process is repeated for each test case until all tuples are covered. This approach is adopted for both CAs and SCAs generation. In contrast, in one-parameter-at-a-time approach, the interaction elements are subdivided in the generation for each input parameter. The test suite starts from t parameters covered. Then through horizontal extension, a new parameter is considered to be added at the pre-generated test case. Next, a

Department of Computer Engineering, College of Engineering, University of Baghdad, Baghdad, Iraq.
E-mail: younismi@coeng.uobaghdad.edu.iq
*ORCID ID: 0000-0003-4884-3747

vertical expansion is followed to cover the remaining tuples like the one-test-at-a-time approach. Thus, one-parameter- at-a-time achieves a lower order of complexity than that of one-test-at-a-time (3-6). For this reason, this paper proposes a strategy based on one-parameter-at-a-time method. The remaining sections of this paper give a literature review on combinatorial test data generation strategies and focus on SCA notations and applications. Followed by the proposed strategy with a step-by-step example. Next, an evaluation, comparison with the existing works, and critical analysis of the results are discussed. The final section states the conclusion and future directions.

Related Works:

The CAs are well studied in the literature (7), examples for one-test-at-a-time approach involve: Automatic Efficient Test Generator (AETG) (8), IBM's Intelligent Test Case Handler (ITCH) (9), Jenny (10), simulated annealing (SA) (8), genetic algorithm (GA) (9), ant colony algorithm (ACA) (10), Hill climbing (HC) (11), GTWAY (12), particle swarm optimization (PSO) (13), harmony search (HS) (14), cuckoo search (CS) (15), high level hyper-heuristic (HHH) (16), adaptive teaching learning-based optimization (ALTBO) (17), flower pollination algorithm (FPA) (18), elitist flower pollination algorithm (eFPA) (1). Examples for one-parameter-at-a-time involve: Input Parameter Order-Generalized (IPOG) (4, 5), the modified IPOG (MIPOG), IPOD, Forbes' IPOG (IPOG-F) (6), and OPAT-HS (19). Another classification of t-way test data generation strategies is whether it is deterministic or non-deterministic.

Deterministic means running the algorithm several times yield the same test suite. Although deterministic nature is more preferred for a tester, Non-deterministic some-times generate different test size; thus it may generate a minimal test suite (1, 6, 7).

SCAs unlike CAs in history; CAs research started in the last 20 years for t=2 (pairwise testing) then some algorithms are generalized for t-way testing in the last decade. Whilst, SCAs is first identified as a combinatorial explosion problem by National Institute of Standards and Technology (NIST) research group in 2012 (20).

SCA (N, t, n) as an N x n matrix where entries are from a finite Sequence Set SS of n events, such that every t-length permutation of symbols from SS occurs in at least one row (21). The test size is N.

To understand the effectiveness of SCA combinatorial test case generation, for a system with 10 events, the exhaustive testing generates 10! =3,628,800 test cases. By relaxing the strength of coverage, 7-way SCA generates 12946 test cases, 6-way SCA generates 1987 test cases, 5-way SCA generates 324 test cases, 4-way SCA generates 60 test cases, 3-way SCA generates 12 test cases, and 2-way SCA generates merely 2 test cases.

In 2012, Kuhn et al. presented a strategy called t-way Sequence (TSEQ). In TSEQ a greedy algorithm is adopted to facilitate the selection among candidate test cases to select the highest score test case that covers the maximum number of t-way sequence combination until all combinations are covered in a deterministic one-test-at-a-time fashion (21). In addition, Kuhn et al. proved that an optimal solution for t=2 is quite simple by considering the sequence in order and reverse the sequence as a second test case which always generates a lower bound SCA of size 2 (i.e., SCA (2, 2, n), where n ≥2).

In 2012, Zabil et al. proposed a non-deterministic strategy based on the population Bee Algorithm (BA) (22). In BA, the t-way tuples present the food and the bees search the best source in one-test-at-a- time fashion.

In 2013, Chee et al. presented two deterministic algorithms namely: upper bound algorithm (U) and its reversal (Ur) (2). The U algorithm is based on a greedy method that selects one permutation set from t-way sets covering the most uncovered tuples. The Ur algorithm like U algorithm but a reverse order of the generated test case is also appended at the end of the test set (2).

In 2014, Rahman et al. adopted the SA algorithm for generation non-deterministic SCA called EDIST-SA (23). However, EDIST-SA is a conceptual strategy that reported a lower bound SCA for t=3 and merely four input sequence to generate a SCA of size 6 (i.e., SCA (6, 3, 4)).

In 2016, Ahmad et al. proposed sequence covering array testing generator (SCAT). SCAT selects one of the best candidates from candidates' pool deterministically in greedy one-test-at-a-time fashion (24).

In 2017, Rabbi proposed a swarm intelligence sequence generator (SISEQ). The SISEQ selects the particles, which in turn, identify the search domain. Next, the SISEQ determines the global velocity to maximize the tuples covered in a greedy one-test-at-a- time manner (25). Later, Rabbi proposed a modification to SISEQ (mSISEQ) to tackle the complexity of the search space for t=4. In some results, mSISEQ outperforms the original SISEQ as generates minimal test size.

In 2018, Nasser et al. proposed an enhanced exploration capability via elitism (eFPA) (1) for its counterpart FPA. Unlike all existing works, both strategies can generate CAs and SCAs in the same implementation. Moreover, these non-deterministic strategies are reported SCAs up to t=6.

SCAs are used so far for testing graphical user interfaces (26-28), testing a factory automation system (21, 29), and combined with CAs to handle multi-values sequence covering array (30).

DEO Strategy:

DEO strategy adopts the one-parameter-at-a-time approach. Unlike the one-parameter-at-a-time family that takes the parameter-values in order for CAs generation, the proposed DEO search the order dynamically for both horizontal extension and vertical expansion. The DEO strategy is illustrated in Fig. 1. In order to demonstrate the proposed DEO strategy and demonstrate the correctness. an illustrative example is to be considered for sequence set= (0,1,2,3,4), t=3. The example uses the symbols defined in Fig. 1. The DEO starts with empty π and ts, t=3, n=5. The DEO denotes E1E2E3= [0,1,2], then generates 3! combinations, appends them to ts, as tabulated in Table 1. Next, i=4, and the Π set consists of 3! combinations of sequences E1E2E3= [1,2,3], [0,2,3] and [0,1,3]. Here, it should be mentioned that there is no need to generate the previously generated sequences (i.e., for E1E2E3= [0,1,2]) since these combinations are already covered in previous steps. Thus, the Π set for 3-way combinations is listed in Fig. 2.

```
        Algorithm DEO Test Case Generation (int t, SequenceSet ss) let ts be the test set;
1.    {
2.      initialize ts as an empty set;
3.      let n be the size of ss;
4.      let E1, E2, ..., and En be the events in ss in an arbitrary order;
5.      generate t! permutation (t-way combination of events) of the first t events;
6.      append the generated tuples in step 4 to ts;
7.      for (index i = t+1 .. n, step 1)
8.       {
9.         let π be t-way permutations of sequences set involving the event Ei
                                   and the t−1 events among the previous i−1 events;
10.        // horizontal process for event Ei
11.         for (each test case (τ) in ts)
12.          {
13.            starts from the end to the first position;
14.            select a sequence position of Ei;
15.            append Ei to τ (τ') such that τ' maximizes the number of tuples covered in π;
16.            remove the t-way tuples included in τ' from π;
17.            }// horizontal process
18.          // vertical process for event Ei
19.         while (π contains tuples)
20.           {
21.             choose the first tuple from π and do the exhaustive search for missing events
    such that the combined test case (τ) covers the maximum number of tuples
                                                                   in π set;
22.             remove the t-way tuples included in τ from π;
23.             append τ to ts;
24.             } // vertical expansion process
25.         } // for
26.     return ts;
27.   } //DEO
```

Figure 1. The proposed DEO strategy.

Table 1. The SCA (6, 3, 3) generated by DEO strategy.

Test Case Number	Test Case	Tuples Covered
1	0, 1, 2	[0, 1, 2]
2	0, 2, 1	[0, 2, 1]
3	1, 0, 2	[1, 0, 2]
4	1, 2, 0	[1, 2, 0]
5	2, 0, 1	[2, 0, 1]
6	2, 1, 0	[2, 1, 0]

```
Π= [
 [[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]],
 [[0, 2, 3], [0, 3, 2], [2, 0, 3], [2, 3, 0], [3, 0, 2], [3, 2, 0]],
 [[0, 1, 3], [0, 3, 1], [1, 0, 3], [1, 3, 0], [3, 0, 1], [3, 1, 0]]
     ]
```

Figure 2. The Π set for 3-way combinations when i=4.

Now, the horizontal extension starts. For the first test case, the candidate positions for event

(3) are 3,2,1, and 0 respectively (i.e., τ = 0, 1, 2, 3, τ = 0, 1, 3, 2, τ = 0, 3,1, 2; and τ =3, 0, 1, 2 respectively). Since, the test case τ =0, 1, 2, 3 covers three tuples (i.e., [1,2,3], [0,2,3] and [0,1,3]), this implies that τ'=0, 1, 2, 3. The remaining candidate test cases also cover three tuples, thus, the extension for the first test case is done due to fact that the proposed DEO takes the first solution. Next, the tuples covered by this test case are deleted from the Π list. Similarly, for the second test case, the candidate test cases are: τ =0, 2, 1, 3, τ =0, 2, 3, 1, τ=0, 3, 2, 1, and τ =3, 0, 2, 1 respectively. The test case "0, 2, 1, 3" covers only one tuple (i.e., [2, 0, 3]) in the remaining Π set. The test case "0, 2, 3, 1" covers two tuples (i.e., [0,3,1] and [2,3,1]). The test case "0, 3, 2, 1" covers three tuples (i.e., [0, 3, 1], [0, 3, 2], and [3, 2,1]). Finally, the test case "3, 0, 2, 1" also covers three tuples (i.e., [3, 0, 1], [3, 0, 2], and [3, 2, 1]). Thus, τ'=0, 3, 2, 1. Next, the tuples covered by this test case are deleted from the Π list. In similar way, the horizontal extension is continued for the remaining test cases in the test set list (ts) and the resulted SCA is tabulated in Table 2.

Table 2. The SCA (6, 3, 4) generated by DEO strategy.

Test Case Number	Test Case	Tuples Covered
1	0, 1, 2, 3	[0 ,1 ,3], [0, 2, 3], [1, 2, 3]
2	0, 3, 2, 1	[0, 3, 1], [0, 3, 2], [3, 2, 1]
3	1, 3, 0, 2	[1, 3, 0], [1, 3, 2], [3, 0,2]
4	3, 1, 2, 0	[3, 1, 2], [3, 1, 0], [3, 2, 0]
5	2, 3, 0, 1	[2, 3, 0], [2, 3, 1], [3, 0, 1]
6	2, 1, 0, 3	[2, 1, 3], [2, 0, 3], [1, 0, 3]

After the horizontal extension, the Π set is empty which implies that there is no vertical expansion. Finally, for the last iteration (i.e., i=5), and the Π set consists of 3! combinations of sequences E1E2E3= [2, 3, 4], [1, 3, 4], [1, 2, 4], [0, 3, 4], [0, 2, 4], and [0,1,4]. The Π set for 3-way combinations is listed in Fig. 3.

```
Π=[
[[2, 3, 4], [2, 4, 3], [3, 2, 4], [3, 4, 2], [4, 2, 3], [4, 3, 2]],
[[1, 3, 4], [1, 4, 3], [3, 1, 4], [3, 4, 1], [4, 1, 3], [4, 3, 1]],
[[1, 2, 4], [1, 4, 2], [2, 1, 4], [2, 4, 1], [4, 1, 2], [4, 2, 1]],
[[0, 3, 4], [0, 4, 3], [3, 0, 4], [3, 4, 0], [4, 0, 3], [4, 3, 0]],
[[0, 2, 4], [0, 4, 2], [2, 0, 4], [2, 4, 0], [4, 0, 2], [4, 2, 0]],
[[0, 1, 4], [0, 4, 1], [1, 0, 4], [1, 4, 0], [4, 0, 1], [4, 1, 0]]
 ]
```

Figure 3. The Π set for 3-way combinations when i=5.

The horizontal extension for the first six test cases is tabulated in Table 3, in the same manner, described previously. When the horizontal extension has been finished, there are four uncovered tuples in the Π set, namely: [[[1, 4, 3], [3, 4, 1]], [[2, 1, 4], [4, 1, 2]]].

The vertical expansion starts with tuple [1, 4, 3] and does an exhaustive search for this missing sequences (i.e., 0 and 2 in this case). This yields a test case "0, 2, 1, 4, 3" which covers two tuples (i.e., [1, 4, 3], and [2, 1, 4]). Next, this test case is appended to the test case list (i.e., test case number 7), and the tuples covered are deleted from the Π set. By the same way, the final test case "3, 4, 1, 0, 2" which covers the remaining two tuples are appended to the test case list (i.e., test case number 8), and the Π set is empty. Thus, during the vertical expansion, the test cases are expanded to cover the uncovered tuples in horizontal extension. The SCA (8, 3, 5) is tabulated in Table 3.

Table 3. The SCA (8, 3, 5) generated by DEO strategy.

Test Case Number	Test Case	Tuples Covered
1	01234	[2, 3, 4], [1, 3, 4], [1, 2, 4],[0, 3, 4], [0, 2, 4], [0, 1, 4]
2	04321	[4, 3, 2], [4, 3, 1], [4, 2, 1],[0, 4, 3], [0, 4, 2], [0, 4, 1]
3	13402	[3, 4, 2], [1, 4, 2], [3, 4, 0],[4, 0, 2], [1, 4, 0]
4	31204	[3, 2, 4], [3, 1, 4], [3, 0, 4],[2, 0, 4], [1, 0, 4]
5	24301	[2, 4, 3], [2, 4, 1], [4, 3, 0],[2, 4, 0], [4, 0, 1]
6	42103	[4, 2, 3], [4, 1, 3], [4, 0, 3],[4, 2, 0], [4, 1, 0]
7	02143	[1, 4, 3], [2, 1, 4]
8	34102	[3, 4, 1], [4, 1, 2]

Results and Discussions:

In order to evaluate the performance of DEO and make a comparison with existing works, it is desired to adopt evaluation experiments. These experiments are done partially in (22, 24, 25) and extended in (1) to include the results of PA and ePA strategies. This paper also extends the experiments to make an evaluation of DEO and to facilitate the comparison against existing works. As such, five experiments are conducted to generate SCA (N, t, n). In the first experiment, t=3 and n vary from 4 to 20. In the second experiment, t=4 and n vary from 5 to 20. In the third experiments, t=5 and n vary from 6 to 10. In the fourth experiment, t=6 and n vary from 7 to 10. In the fifth experiment, t=7 and n vary from 8 to 10. All these results are tabulated in

Tables 4 till 8, respectively. In these tables for each strategy, a shaded cell means the best size. The unavailable results in the literature are marked as NA. For non-deterministic strategies, the best (i.e., minimal size) is taken from the literature. As shown in tabulated results, the size of SCA in DEO can be started by t! and expanded logarithmically as the number of events increases. In fact, when DEO generates t! test cases it is considered absolute minimal (i.e., lower bound). Moreover, the test suites size for DEO is less than the half of the U and Ur algorithms. There is no mathematical expression to determine the upper bound due to NP-hardness problem. Thus, finding an upper bound is done by comparing existing strategies on a one-by-one basis. In order to tackle the significant of DEO as a minimization test case generation, we introduce the delta (Δ) as the difference in size between the DEO's result (NDEO) and the best-known result (NBest) as given in Eq. 1. Thus, when Δ=0 this means that DEO generates the same upper bound of well-known results. When Δ has a positive value, it means that there is a diverging from a minimal solution. Similarly, when Δ has a negative value, it means that DEO contributes to finding a new upper bound for a minimal solution, the significant increased as the value increased.

$$\Delta = \text{NDEO} - \text{NBest} \qquad \textbf{Equation (1)}$$

Referring to Table 4 when t=3, BA, FPA, eFPA, and DEO starts with lower boundary when n=4; which is identical to the best solution of EDIST-SA. Both FPA and eFPA outperforms existing strategies when n=5. eFPA outperforms existing strategies when n=8. Nevertheless, DEO outperforms the other strategies for higher number of events and produces new upper bounds for SCA when n= 13, 14, 15, 16, 18, 19, and 20.

Table 4. Comparison between the state-of-the-art strategies for SCA (N, 3, 4≤n≤20).

SUT	U	Ur	BA	SCAT	TSEQ	SISEQ	mSISEQ	FPA	eFPA	DEO	Δ
	(2)	(2)	(22)	(24)	(21)	(25)	(25)	(1)	(1)		
SCA (N, 3, 4)	12	12	6	8	NA	NA	NA	6	6	6	0
SCA (N, 3, 5)	17	16	8	10	8	8	NA	7	7	8	+1
SCA (N, 3, 6)	20	18	9	12	10	10	NA	9	8	8	0
SCA (N, 3, 7)	23	22	10	12	12	10	NA	10	10	10	0
SCA (N, 3, 8)	26	24	11	12	12	12	NA	11	10	11	+1
SCA (N, 3, 9)	28	26	13	14	14	12	NA	12	11	11	0
SCA (N, 3, 10)	30	28	14	16	14	NA	NA	13	12	12	0
SCA (N, 3, 11)	32	30	NA	16	14	NA	NA	13	13	13	0
SCA (N, 3, 12)	33	30	NA	16	16	NA	NA	14	13	13	0
SCA (N, 3, 13)	35	32	NA	18	16	NA	NA	15	15	14	-1
SCA (N, 3, 14)	36	34	NA	18	16	NA	NA	16	16	15	-1
SCA (N, 3, 15)	37	34	NA	20	18	NA	NA	16	16	15	-1
SCA (N, 3, 16)	39	36	NA	18	18	NA	NA	17	17	16	-1
SCA (N, 3, 17)	40	36	NA	20	20	NA	NA	18	17	17	0
SCA (N, 3, 18)	41	38	NA	20	20	NA	NA	18	18	17	-1
SCA (N, 3, 19)	42	38	NA	20	22	NA	NA	19	19	18	-1
SCA (N, 3, 20)	42	38	NA	22	22	NA	NA	20	20	19	-1

Referring to Table 5 when t=4, SCAT, SISEQ, and DEO start with lower bound size when n=5. BA, SCAT, TSEQ, FPA, and eFPA outperforms DEO when n=6. mSISEQ outperforms existing strategies when n=7and n=9. SISEQ, mSISEQ, and eFPA outperform existing strategies when n=8. Whilst, DEO outperforms the existing strategies when 10≤n≤20. It should be mentioned that the significant of DEO increases as the number of events increased. Referring to Table 6 when t=5, eFPA outperforms DEO when n=7. Whilst, DEO outperforms the existing strategies and generates new upper bound for SCA when n=6, 8, 9, and 10.

Referring to Table 7 when t=6, DEO outperforms the existing strategies when 7≤n≤10 significantly and produces new upper bound for SCA. There is a little attention for providing higher degree of interaction for SCA in the literature (i.e., t>=7). For this reason, only DEO generates SCAs as tabulated in Table 8. Thus, Δ is kept empty when t=7.

Unlike the previous works on CA, the proposed DEO strategy makes the position of the incoming sequence dynamically whilst the one-parameter-at-time makes the position static. Regarding the previous works on SCA, the DEO strategy is the first implementation that adopts the one-parameter-at-time approach. The adopting of DEO for SCA generation not just reduced the complexity but also contribute to find new upper bounds of SCAs, namely: SCA (14, 3, 13), SCA (15, 3, 14), SCA (15, 3, 15), SCA (15, 3, 16), SCA (17,3, 18), SCA (18, 3, 19), SCA (19, 3, 20), SCA (60, 4, 10), SCA (66, 4, 11), SCA (72, 4, 12), SCA

(78, 4, 13), SCA (83, 4, 14), SCA (86, 4, 15), SCA (91, 4, 16), SCA (95, 4,17), SCA (99, 4, 18), SCA (103, 4, 19), SCA (106, 4, 20), SCA (138, 5, 6), SCA (234, 5, 8), SCA (280, 5, 9), SCA (324, 5, 10), SCA (903, 6, 7), SCA (1265, 6, 8), SCA (1555, 6, 9), SCA (1987, 6, 10), SCA (6936, 7, 8), SCA(9965, 7, 9), and SCA (12946, 7, 10).

Table 5. Comparison between the state-of-the-art strategies for SCA (N, 4, 5≤n≤20).

SUT	U	Ur	BA	SCAT	TSEQ	SISEQ	mSISEQ	FPA	eFPA	DEO	Δ
	(2)	(2)	(22)	(24)	(21)	(25)	(25)	(1)	(1)		
SCA (N, 4, 5)	54	54	28	24	26	24	39	29	28	24	0
SCA (N, 4, 6)	79	78	36	36	36	37	38	36	36	37	+1
SCA (N, 4, 7)	98	96	45	46	46	42	41	43	42	44	+3
SCA (N, 4, 8)	114	112	55	54	50	48	48	50	48	50	+2
SCA (N, 4, 9)	128	126	62	62	58	53	52	57	54	56	+4
SCA (N, 4, 10)	140	138	71	64	66	NA	NA	63	61	60	-1
SCA (N, 4, 11)	151	148	NA	72	70	NA	NA	67	68	66	-1
SCA (N, 4, 12)	160	158	NA	82	78	NA	NA	74	74	72	-2
SCA (N, 4, 13)	169	166	NA	86	86	NA	NA	79	79	78	-1
SCA (N, 4, 14)	177	174	NA	90	90	NA	NA	84	84	83	-1
SCA (N, 4, 15)	184	180	NA	90	96	NA	NA	90	89	86	-3
SCA (N, 4, 16)	191	188	NA	96	100	NA	NA	97	97	91	-6
SCA (N, 4, 17)	197	194	NA	104	108	NA	NA	101	103	95	-6
SCA (N, 4, 18)	203	200	NA	106	112	NA	NA	106	105	99	-7
SCA (N, 4, 19)	209	204	NA	114	114	NA	NA	109	110	103	-6
SCA (N, 4, 20)	214	210	NA	112	120	NA	NA	114	115	106	-8

Table 6. Comparison between the state-of-the-art strategies for SCA (N, 5, 6 ≤n≤10).

SUT	U	Ur	BA	SCAT	TSEQ	SISEQ	mSISEQ	FPA	eFPA	DEO	Δ
	(2)	(2)	(22)	(24)	(21)	(25)	(25)	(1)	(1)		
SCA (N, 5, 6)	294	294	159	154	NA	NA	NA	148	152	138	-10
SCA (N, 5, 7)	437	436	212	212	NA	NA	NA	199	194	196	+2
SCA (N, 5, 8)	552	550	271	264	NA	NA	NA	247	240	234	-6
SCA (N, 5, 9)	648	646	329	324	NA	NA	NA	295	283	280	-3
SCA (N, 5, 10)	731	728	383	368	NA	NA	NA	344	344	324	-20

Table 7. Comparison between the state-of-the-art strategies for SCA (N, 6, 7 ≤n≤ 10).

SUT	U	Ur	BA	SCAT	TSEQ	SISEQ	mSISEQ	FPA	eFPA	DEO	Δ
	(2)	(2)	(22)	(24)	(21)	(25)	(25)	(1)	(1)		
SCA (N, 6, 7)	NA	NA	NA	NA	NA	NA	NA	980	960	903	-57
SCA (N, 6, 8)	NA	NA	NA	NA	NA	NA	NA	1301	1274	1265	-9
SCA (N, 6, 9)	NA	NA	NA	NA	NA	NA	NA	1636	1628	1555	-73
SCA (N, 6, 10)	NA	NA	NA	NA	NA	NA	NA	1998	2161	1987	-174

Table 8. Comparison between the state-of-the-art strategies for SCA (N, 7, 8≤n≤10).

SUT	U	Ur	BA	SCAT	TSEQ	SISEQ	mSISEQ	FPA	eFPA	DEO	Δ
	(2)	(2)	(22)	(24)	(21)	(25)	(25)	(1)	(1)		
SCA (N, 7, 8)	NA	NA	NA	NA	NA	NA	NA	NA	NA	6936	-
SCA (N, 7, 9)	NA	NA	NA	NA	NA	NA	NA	NA	NA	9965	-
SCA (N, 7, 10)	NA	NA	NA	NA	NA	NA	NA	NA	NA	12946	-

Conclusions:

This paper proposed a strategy called DEO based on one-event-at-a-time for t-way SCA test data generation. The proposed strategy outperforms most of the existing works as far as minimal test size is concerned. In addition, unlike the existing works, DEO can generate SCAs up to t=7.Moreover, the significant of DEO increases as the number of sequences (n) increases as well as the strength of coverage (t) increases. Furthermore, new upper bounds of SCAs are reported. Apart from future works, we are currently studying some variants for both horizontal extension and vertical expansion which may lead to further minimization to the test suite. Another direction for future work is to make a hybrid strategy based on pre-generated SCA and MVSCA.

Author's declaration:

- Conflicts of Interest: None.
- I hereby confirm that all the Figures and Tables in the manuscript are mine. Besides, the Figures and images, which are not mine, have been given the permission for re-publication attached with the manuscript.
- Ethical Clearance: The project was approved by the local ethical committee in University of Baghdad.

References:

1. Nasser AB, Zamli KZ, Alsewari AA, Ahmed BS. An elitist-flower pollination-based strategy for constructing sequence and sequence-less t-way test suite. IJBIC, 2018, 12(2):115–127. DOI: 10.1504/IJBIC.2018.094223.
2. Chee YM, Colburn CJ, Horsley D, Zhou J. Sequence covering arrays. SIDMA, 2013, 27(4): 1844–1861, DOI: 10.1137/120894099.
3. Anand S, Burke EK, Chen TY, Clark J, Cohen MB, Grieskamp W, et.al. An orchestrated survey of methodologies for automated software test case generation. J Syst Softw, 2013, 86(8): 1978-2001. DOI: 10.1016/j.jss.2013.02.061.
4. Tai KC, Lei Y. A test generation strategy for pairwise testing. IEEE T Softw Eng, 2002, 28(1):109-111. DOI: 10.1109/32.979992.
5. Lei Y, Kacker R, Kuhn DR, Okun V, Lawrence J. IPOG/IPOG-D: efficient test generation for multi-way combinatorial testing. STVR, 2008, 18: 125-148. DOI=http://dx.doi.org/10.1002/stvr.v18:3.
6. Younis MI, Zamli KZ. MIPOG-an efficient t-way minimization strategy for combinatorial testing. IJCTE, 2011, 3(3): 388-397.
7. Zamli KZ, Othman RR, Younis MI, Zabil MHM. Practical adoptions of t-way strategies for interaction testing. ICSECS, 2011,181:1-14. Springer, Berlin, Heidelberg. DOI: https://doi.org/10.1007/978-3-642-22203-0_1.
8. Cohen DM, Dalal SR, Fredman ML, Patton GC. The AETG system: an approach to testing based on combinatorial design. IEEE T Softw Eng, 1997, 23(7): 437–443. DOI: 10.1109/32.605761.
9. Hartman A, Raskin L. Problems and algorithms for covering arrays. Discrete Math, 2004, 284: 149-156. DOI: https://doi.org/10.1016/j.disc.2003.11.029.
10. Jenkins B. Jenny test tool. Available at: http://www.burtleburtle.net./bob/math/jenny.html (accessed on 13 June 2019).
11. Cohen MB. Designing test suites for software interaction testing. Doctoral dissertation, University of Auckland ,2004, Available at: https://cse.unl.edu (accessed on 1 June 2019).
12. Zamli KZ, Klaib MFJ, Younis MI, Isa NAM, Abdullah R. Design and implementation of a t-way test data generation strategy with automated execution tool support. Inform Sciences, 2011,181(9):1741-1758. DOI: https://doi.org/10.1016/j.ins.2011.01.002.
13. Ahmed BS, Zamli KZ, Lim CP. Application of particle swarm optimization to uniform and variable strength covering array construction. Appl Soft Comput, 2012, 12(4): 1330–1347.
14. Alsewari ARA, Zamli KZ. Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support. Inform Softw Tech, 2012, 54(6):553–568.
15. Ahmed BS, Abdulsamad TS, Potrus MY. Achievement of minimized combinatorial test suite for configuration-aware software functional testing using the cuckoo search algorithm. Inform Softw Tech, 2015, 66(C):13-29. DOI: 10.1016/ j.infsof.2015.05.005.
16. Zamli KZ, Alkazemi BY, Kendall G. A tabu search hyper-heuristic strategy for t-way test suite generation. Appl Soft Comput, 2016, 44: 57–74. DOI: https://doi.org/10.1016/j.asoc.2016.03.021.
17. Zamli KZ, Din F, Baharom S, Ahmed BS. Fuzzy adaptive teaching learning-based optimization strategy for the problem of generating mixed strength t-way test suites. Eng Appl Artif Intel, 2017, 59: 35–50. DOI: https://doi.org/10.1016/j.engappai.2016.12.014.
18. Yang XS, Deb S, Fong S. Metaheuristic algorithms: optimal balance of intensification and diversification. Appl Math Inform Sci, 2014, 8(3): 977-983.
19. Alsewari AA, Muaza AA, Rassem TH, Tairan NM, Shah H, Zamli KZ. One-parameter-at-a-time combinatorial testing strategy based on harmony search algorithm OPAT-HS. Adv Sci Lett, 2018, 24 (10): 7273-7277, DOI: https://doi.org/10.1166/asl.2018.12927.
20. NIST WEB Site: Available at https://csrc.nist.gov/Projects/Automated-Combinatorial-Testing-for-Software/Event-Sequence-Testing, (accessed on 1 June 2019).
21. Kuhn DR, Higdon JM, Lawrence JF, Kacker RN, Lei Y. Combinatorial methods for event sequence testing. ICST, 2012, pp. 601-609. DOI: 10.1109/ICST.2012.1.
22. Hazli MZM, Zamli KZ, Othman RR. Sequence-based interaction testing implementation using bees algorithm. ISCI, 2012, pp. 81-85.DOI: 10.1109/ISCI.2012.6222671.
23. Rahman M, Othman RR, Ahmad RB, Rahman MM. Event driven input sequence t-way test strategy using simulated annealing. ISMS, 2014, pp. 663-667. DOI: 10.1109/ISMS.2014.119.
24. Ahmad MZZ, Othman RR, Ali MSAR. Sequence covering array generator (scat) for sequence based combinatorial testing. IJAER, 2016, 11(8): 5984–5991.
25. Rabbi KF. Combinatorial testing strategies based on swarm intelligence. Ph.D. thesis, School of Computing and Mathematics, Charles Sturt University, August 2017.
26. Yuan X, Cohen MB, Memon AM. GUI interaction testing: incorporating event context. IEEE T Softw Eng, 2011, 37(4): 559–574.
27. Yilmaz C, Fouche S, Cohen MB, Porter A, Demiroz G, Koc U. Moving forward with combinatorial

interaction testing. Comp, 2014, 47 (2): 37-45. DOI: 10.1109/MC.2013.408.

28. Rattliff ZB. Black box testing mobile applications using sequence coverying arrays. BSc. Thesis, Texas A&M University, May 2018, Available at: https://oaktrust.library.tamu.edu/bitstream/handle/1969.1/166475/RATLIFF-FINALTHESIS-2018.pdf?sequence=1&isAllowed=y (accessed on 1 June 2019).

29. Musa J, Romli R, Yusoff N. An analysis on the applicability of meta-heuristic searching techniques for automated test data generation in automatic programming assessment. Baghdad Sci J, 2019, 16: 515-533. DOI: 10.21123/bsj.2019.16.2(SI).0515.

30. Younis MI. MVSCA: multi-valued sequence covering array, J Eng, 2019, 25 (11), pp. 82-91.DOI: 10.31026/j.eng.2019.11.0 7.

استراتيجية ترتيب الأحداث الديناميكية لتوليد بيانات فحص مصفوفة التغطية المتسلسلة

محمد عصام يونس

قسم هندسة الحاسبات، كلية الهندسة، جامعة بغداد، بغداد، العراق

الخلاصة:

تعد مصفوفة التغطية المتسلسلة (SCA) من مجالات البحث النشطة في السنوات الأخيرة. بخلاف مصفوفة التغطية الاعتيادية (CA)، يختلف ترتيب تسلسل العوامل في عملية إنشاء حالة الاختبار. تقوم هذه الورقة بمراجعة أحدث الاستراتيجيات في الأعمال السابقة، حيث أن العثور على الحد الأدنى لحجم مجموعة الاختبار يعتبر مشكلة NP-Hard. بالإضافة إلى ذلك ، تتمتع معظم الاستراتيجيات الحالية الخاصة بتوليد SCA بترتيب عالٍ من التعقيد نظرًا لتوليد جميع التفاعلات التوافقية من خلال تبني أسلوب اختبار واحد في كل مرة. يعد الحد من التعقيد من خلال تبني عامل واحد في وقت واحد لتوليد SCA عملية صعبة. بالإضافة إلى ذلك، يوفر هذا الحد من التعقيد دعما للحصول على قوة تغطية أعلى. وبمواجهة هذا التحدي، تقترح هذه الورقة استراتيجية SCA جديدة تسمى ترتيب الاحداث الديناميكية Dynamic Event Order (DEO)، والتي يتم فيها إنشاء حالة الاختبار باستخدام عامل واحد في وقت واحد. يقدم هذا البحث تفاصيل DEO مع مثال خطوة بخطوة لإظهار السلوك وإظهار صحة الاستراتيجية المقترحة. بالإضافة إلى ذلك، تقوم هذه الورقة بإجراء مقارنة مع الاستراتيجيات الحسابية الحالية. توضح النتائج العملية أن استراتيجية DEO المقترحة تتفوق على الاستراتيجيات الحالية من حيث الحد الأدنى لحجم الاختبار في معظم الحالات. علاوة على ذلك، تزداد أهمية DEO مع زيادة عدد التتابعات و / أو زيادة قوة التغطية. حيث نجحت استراتيجية DEO المقترحة في إنشاء SCAs حتى t = 7. أخيرًا، نجحت إستراتيجية DEO في إيجاد حدود عليا جديدة ل SCA. في الواقع، أن الإستراتيجية المقترحة تعد قاعدة بحثية لتنفيذ وتطوير خوارزميات مستقبلية باعتماد فكرة العامل واحد في الوقت الواحد المنفذة.

الكلمات المفتاحية: الفحص الاندماجي ، اختبار الحدث ، اختبار متعدد طرق الترتيب ، اختبار التسلسل ،فحص ترابط_الترتيب .