Consecutive Detecting Arrays from m-Sequences

Ce Shi, *Member, IAENG,* and Aiyuan Tao

***Abstract*—Linear feedback shift register (LFSR) sequences are sequences of elements of a finite field that satisfy some linear recurrence relations. LFSR sequences that attain maximum possible periods are called maximal (period) sequences, often abbreviated as m-sequences. Arrays constructed from cyclic shifts of m-sequences possess rich combinatorial properties and have previously been used in the construction of (ordered) orthogonal and covering arrays. We propose the notion of consecutive detecting arrays for generating test suites for locating and detecting interaction faults between neighboring factors in combinatorial interaction testing. In this paper, we explain the construction of a class of consecutive detecting arrays from m-sequences and illustrate their use through concrete examples. Furthermore, we show some consecutive detecting arrays with minimum size for practical use.**

***Index Terms*—Neighbor combinatorial testing, consecutive detecting arrays, LFSR sequences, m-sequence, construction.**

I. Introduction

We denote the set of the first n positive integers by I_n. A *covering array* (CA) of size N, strength t, degree k and order v, denoted by CA$(N;t,k,v)$, is an $N \times k$ array with entries from a set V of v symbols such that in any t columns of the array all t-tuples in V^t occur as rows at least once.

The study of CAs is primarily motivated by their applications in generating test suites for *combinatorial interaction testing (CIT)* [1], [2]. Combinatorial interaction testing is a testing strategy for achieving high fault detection capability with a small number of tests. Specifically, a test suite represented as a t-way CA ensures every combination of values of any t parameters is tested at least once. In recent years, as noted in [3], [4], testing has played an important role in component-based systems. Considering the complexity of systems, the number of possible tests can be exponentially large. Interactions among components are complex and numerous. Thus, components are prone to unexpected interaction faults. Ideally, one would test all possible interactions (called exhaustive testing or complete testing), but the testing burden would be extremely heavy, even when the system is of moderate complexity. Thus, exhaustive testing is often not feasible. Consequently, generating test suites that provide coverage of the most prevalent interactions provides a more feasible path. A primary combinatorial object that satisfies the coverage criteria is covering arrays. In fact, problems faced in software interaction testing are not limited to software; rather, they are present in similar forms in other disciplines such as agriculture, manufacturing, and medicine. For more details, readers are referred to [3], [5], [6] and the references therein. CAs usually take an input of integer t to ensure that all interactions on t parameter values (i.e., t-way interactions), instead of all combinations of system parameter values, can be tested at least once. This property of CAs significantly reduces the testing cost when compared to exhaustive testing.

Manuscript received June 18, 2019; revised September 26, 2019. This work was supported in part by the National Natural Science Foundation of China(11301342), the Natural Science Foundation of Shanghai(17ZR1419900) and Ministry of Education in China Project of Humanities and Social Sciences for Youth Scholars(19YJCZH031)

Ce Shi is with the School of Statistics and Mathematics, Shanghai Lixin University of Accounting and Finance, Shanghai 201209, China. E-mail: shice060@lixin.edu.cn.
Aiyuan Tao is with the School of International Economics and Trade, Shanghai Lixin University of Accounting and Finance, Shanghai 201209, China. E-mail: taoaiyuan@126.com.
Corresponding author: Aiyuan Tao. (taoaiyuan@126.com).

TABLE I
Configuration parameters for NGS

Parameter		Values		
F_1	Browser	Netscape	IE	Firefox
F_2	OS	Win	Linux	Mac
F_3	Access	ISDL	Modem	VPN
F_4	Audio	Creative	Digital	Maya

Table I shows an example of *system under testing* (SUT) models from [1]. The Network Game Software (NGS) has four components: Browser, OS, Access, and Audio, each of which can take one of three different values. A complete test would require $3^4 = 81$ tests. Table II gives a 2-way CA for the SUT. The CA represents a test suite comprising of ten tests. Every combination of two components appears at least once in the array. The advantage of CIT is that it can detect failures triggered by the interactions among parameters in SUT.

Covering arrays have been widely studied in recent decades. The terminology "covering array" was proposed by Sloane as a generalization of orthogonal arrays [8]. There are also various equivalent formulations, for example, transversal covers [9]. The main techniques for finding a covering array include direct constructions employing mathematical and information-theoretic structures, recursive constructions for creating large covering arrays from smaller ones, probabilistic methods and stochastic algorithms, and computational methods based on (heuristic) search algorithms. The algorithms include stimulation annealing (SA), genetic algorithms (GAs), and particle swarm optimization (PSO) [10], [11], [12], [13]. GAs and PSO have many applications in engineering and science, such as predicting product [14], route optimization of non-holonomic leader–follower control [15].

When using a CA$(N;t,k,v)$ to generate test suites, the columns of the CA represent factors affecting response and the entries within the columns denote settings or values of the factor. The rows represent tests to be run for the values of each factor specified. It is a valuable step in screening a system for interaction faults prior to its release. However, testing with a CA provides little information to assist in the correction of interaction faults. For example, suppose that all the tests in Table II were executed and that all tests

TABLE II
A $\mathrm{CA}(10; 2, 4, 3)$ FOR THE SUT IN TABLE I

Test Case	F_1	F_2	F_3	F_4
1	Netscape	Win	ISDL	Creative
2	Netscape	Linux	Modem	Digital
3	Netscape	Mac	VPN	Maya
4	IE	Win	Modem	Maya
5	IE	Linux	VPN	Creative
6	IE	Mac	ISDL	Digital
7	Firefox	Win	VPN	Digital
8	Firefox	Linux	ISDL	Maya
9	Firefox	Mac	Modem	Creative
10	IE	Win	Modem	Creative

TABLE III
A $(1, 2)$-DETECTING ARRAY FOR THE SUT IN TABLE I

Test Case	Browser	OS	Access	Audio
1	Netscape	Win	ISDL	Creative
2	IE	Linux	Modem	Digital
3	Firefox	Mac	VPN	Maya
4	Netscape	Win	Modem	Maya
5	IE	Linux	VPN	Creative
6	Firefox	Mac	ISDL	Digital
7	Netscape	Linux	VPN	Maya
8	IE	Mac	ISDL	Creative
9	Firefox	Win	Modem	Digital
10	Netscape	Linux	ISDL	Digital
11	IE	Mac	Modem	Maya
12	Firefox	Win	VPN	Creative
13	Netscape	Mac	Modem	Creative
14	IE	Win	VPN	Digital
15	Firefox	Linux	ISDL	Maya
16	Netscape	Mac	VPN	Digital
17	IE	Win	ISDL	Maya
18	Firefox	Linux	Modem	Creative

were passed except the fourth test. The failed test contains six 2-way interactions, namely, (IE, Win), (IE, Modem), (IE, Maya), (Win, Modem), (Win, Maya), and (Modem, Maya). Three of the six interactions, namely, (IE, Win), (IE, Modem), and (Win, Modem), can be safely excluded from the candidates because they occur in the 10th passed test. However, it is impossible to determine which of the remaining three interactions triggered the failure.

Practically, tests to reveal the location of interaction faults are of interest. Colbourn and McClary [7] formalized the problem of a nonadaptive location of interaction faults under the hypothesis that a system contains (maximum) some d number of faults, each involving (maximum) some t number of interacting factors. They proposed the notion of detecting arrays (DAs) to solve this problem. A DA that has parameters d and t is denoted by (d, t)-DA. Roughly speaking, when a SUT model is given, if there exist d or fewer faulty t-way interactions, testing with (d, t)-DA can determine their existence and locate which interactions are faulty; if there exist more than d faulty t-way interactions, (d, t)-DA can return specific results to indicate that the number of faulty interactions is more than d.

Table III presents a $(1, 2)$-DA for the SUT in Table I. From the test outcome, one can determine any faulty interaction when there is exactly one faulty interaction and $t = 2$. If there is more than one faulty interaction and $t = 2$, these interactions can be also detected. For example,

- If test cases #1 and #4 fail while the others pass, then we can locate ((Browser, Netscape), (OS, Win)) as the faulty interactions because they are the only 2-way interactions that appear in both failed test cases.
- If test cases #1, #4, #7, and #17 fail while the others pass, then we cannot locate ((Browser, Netscape), (OS, Win)) as the faulty interactions because ((OS, Win),(Access, ISDL)) may also give rise to faults. Thus, we know that the faulty interaction is more than 1. In fact, only the faulty interaction ((Browser, Netscape), (Audio, Maya)) can be determined from the outcome of the tests.

In real-world systems, there usually exist dependent relations between system parameters and their values, i.e., system constraints. System constraints usually originate from the requirements of the system, physical restrictions for systems, etc. Godbole *et al.* [16] introduced the consecutive covering array (CCA), where the columns capture some linear progression data (for example, data across a series of consecutive dates) or data organized by consecutive proximity (for example, consecutive switches in a circuit). CCAs can be regarded as a variant of CAs. For example, NGS may be influenced by the neighbor configuration parameters, i.e., the failures may be triggered by interactions among neighboring parameters in SUT. This is a special case of software systems with some combinations that need not be tested [17].

Analogous to CAs, CCAs can generate test suites for combinatorial testing of neighboring factors to indicate the presence or absence of faulty interactions. However, they cannot identify or determine the faulty interactions from the outcomes of the tests. Although DAs can be used to locate and detect interaction faults between neighboring factors, they are not well adapted for this kind of software testing. For example, some DAs with minimum sizes do not exist for fixed t, k, v [18], [19], but the optimum arrays for locating and detecting interaction faults between neighboring factors may exist. Moreover, combinatorial testing of neighboring factors only considers consecutive interactions rather than arbitrary interactions. To solve this problem, Shi *et al.* [20] proposed a similar family of DAs, called consecutive detecting arrays (CDAs), which consider the interactions between factors as ordered. A general criterion for measuring the optimality of CDAs in terms of their size was established. Based on this optimality criterion, the equivalence between optimum CDAs and consecutive orthogonal arrays with prescribed properties was explored. Following this equivalence, a large number of optimum CDAs was presented. In particular, the existence of optimum CDAs with a few factors was almost completely determined.

In this paper, a class of CDAs constructed from m-sequences is presented, and some optimum CDAs are obtained. The optimum CDAs obtained from the m-sequences have large numbers of factors, making them readily available for real-world applications.

The remainder of this paper is organized as follows. In the next section, we review basic definitions of CCAs, COAs, DAs, and CDAs. The equivalence between CDAs and COAs is also discussed in this section. Section 3 presents some basic notions and properties of the m-sequences used to construct CDAs. In Section 4, we present a technique to design optimal CDAs from m-sequences. Some examples are given to illustrate this approach and consequently to deduce some optimum CDAs. Finally, in Section 5, we provide some conclusions.

II. Preliminaries

In this section, preliminaries for the field of CIT are discussed. First, some basic definitions of SUT, i.e., test cases and interactions, etc., are introduced. By defining the interactions, the definitions of CCAs, COAs, DAs, and CDAs are given. Finally, the lower bound and combinatorial characteristics for CDAs are presented.

A. SUT, test cases and (consecutive) t-way interactions

In the field of CIT, the objective system can be described as a SUT model. A SUT with k parameters (variables) can be represented by (F, V), where $F = (F_1, F_2, \cdots, F_k)$ is the set of parameters in this system and $V = (V_1, V_2, \cdots, V_k)$ consists of the set V_i of associated values for $F_i (i = 1, 2, \cdots, k)$. A *test case* is a k-tuple $(v_1, v_2, \cdots, v_k)$ such that $v_i \in V_i$ for $1 \le i \le k$. A collection of test cases is referred to as a *test suite*.

Let $A = (a_{ij})(i \in I_N, j \in I_k)$ be an $N \times k$ array with entries from an alphabet V of size v. Each t-set of columns is called a t-way interaction, denoted by $T = \{(j_r, x_r) : x_r \in V, 1 \le r \le t\}$. The t-way testing strategy requires testing every t-way interaction unless it is not testable. We write $\rho(A, T)$ for the set of indices of rows of A that cover T, i.e., $\rho(A, T) = \{i : a_{ij_r} = x_r, 1 \le r \le t\}$. For an arbitrary set $\mathcal{T}$ of interactions, define $\rho(A, \mathcal{T}) = \cup_{T \in \mathcal{T}} \rho(A, T)$. Denote the set of all t-way interactions of A by $\mathcal{I}_t$.

The CA defined above can be described as follows. An array A is a CA of strength t, written t-CA, if and only if the following condition holds:

$$\forall\ T \in \mathcal{I}_t : \rho(A, T) \neq \emptyset$$

A test, when executed, can pass or fail. Thus, a test on k parameters contains (covers) C_k^t interactions of strength t. The outcomes are the corresponding set of pass/fail results. A fault is evidenced by a failure outcome for a test. A fault is rarely due to a complete k-way interaction; rather, it is the result of one or more faulty interactions of strength smaller than k covered in the test. Tests considered here are executed concurrently, so that the testing is nonadaptive or predetermined.

A consecutive t-way interaction is denoted as $T = \{((i, x_i), (i+1, x_{i+1}), \cdots, (i+t-1, x_{i+t-1}))\}$, where $1 \le i \le k-t+1$, $x_r \in V$ for $r = i, i+1, \cdots, i+t-1$. Clearly, there is a total of $(k-t+1)v^t$ consecutive t-way interactions for k neighboring factors, each of which can take v values. The set of all consecutive t-way interactions is denoted by $C\mathcal{I}_t$.

B. CCAs and COAs

CCAs [resp. consecutive orthogonal arrays (COAs)], written as CCA$(N; t, k, v)$ [resp. COA$_\lambda(t, k, v)$], are $N \times k$ arrays with entries from a set V of v symbols where each set of t consecutive columns contains each t-tuple at least once (resp. exactly λ times) within its rows. In other words, an array A is a CCA (resp.COA) of strength t, denoted by t-CCA (t-COA), if and only if the following respective conditions hold:

$$\forall\ T \in C\mathcal{I}_t : \rho(A, T) \neq \emptyset,$$

TABLE IV
A COA$(9; 2, 8, 3)$

Test Case	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
1	0	0	0	0	0	0	0	0
2	0	1	0	1	0	1	0	1
3	0	2	0	2	0	2	0	2
4	1	0	1	0	1	0	1	0
5	1	1	1	1	1	1	1	1
6	1	2	1	2	1	2	1	2
7	2	0	2	0	2	0	2	0
8	2	1	2	1	2	1	2	1
9	2	2	2	2	2	2	2	2

TABLE V
Test outcomes

Test Case	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	Outcome
1	0	0	0	0	0	0	0	0	pass
2	0	1	0	1	0	1	0	1	pass
3	0	2	0	2	0	2	0	2	pass
4	1	0	1	0	1	0	1	0	fail
5	1	1	1	1	1	1	1	1	pass
6	1	2	1	2	1	2	1	2	fail
7	2	0	2	0	2	0	2	0	pass
8	2	1	2	1	2	1	2	1	pass
9	2	2	2	2	2	2	2	2	pass

$$\forall\ T \in C\mathcal{I}_t : |\rho(A, T)| = \lambda$$

Table IV is a COA$(9; 2, 8, 3)$ over Z_3. CCAs present a family similar to CAs. The analogy between CAs and CCAs is evident. Like CAs, CCAs can generate test suites for combinatorial testing of neighboring factors to indicate the presence or absence of faulty interactions. They cannot identify or determine these interactions from the outcome of the tests. For example, Table V depicts the outcome of tests. Any consecutive 2-way interaction from the failed test case may produce the faults. Thus, we do not determine the faulty interaction. To locate and detect interaction faults between neighboring factors, it is only necessary to identify the consecutive interaction faults from the outcomes of the tests.

C. DAs and CDAs

Suppose $A = (a_{ij})(i \in I_N, j \in I_k)$ is a CA$(N; t, k, v)$ over V. For any $\mathcal{T} \subseteq \mathcal{I}_t$ with $|\mathcal{T}| = d$ and any $T \in \mathcal{I}_t$, if

$$\rho(A, T) \subseteq \rho(A, \mathcal{T}) \Leftrightarrow T \in \mathcal{T},$$

then the array A is called a (d, t)-*detecting array*, denoted by (d, t)-DA$(N; k, v)$.

Suppose $A = (a_{ij})$ $(i \in I_N, j \in I_k)$ is a CCA$(N; t, k, v)$ over V. For any $\mathcal{T} \subseteq C\mathcal{I}_t$ with $|\mathcal{T}| = d$ and any $T \in C\mathcal{I}_t$, if

$$\rho(A, T) \subseteq \rho(A, \mathcal{T}) \Leftrightarrow T \in \mathcal{T},$$

then the array A is called a (d, t)-*consecutive detecting array*, denoted by (d, t)-CDA$(N; k, v)$.

Clearly, a (d, t)-DA$(N; k, v)$ must be a (d, t)-CDA$(N; k, v)$, but the converse is not always true. It is straightforward that $T \in \mathcal{T}$ implies $\rho(A, T) \subseteq \rho(A, \mathcal{T})$. Hence, the condition $\rho(A, T) \subseteq \rho(A, \mathcal{T}) \Leftrightarrow T \in \mathcal{T}$ is satisfied if $T \notin \mathcal{T} \Rightarrow \rho(A, T) \nsubseteq \rho(A, \mathcal{T})$. We will make extensive use of this simple fact in the following sections. As well as DAs, there are some admissible parameters for the existence of CDAs. We give the following lists:

Lemma 1: (Shi et al. [20]) If a (d,t)-CDA$(N;k,v)$ exists, then $d<v$.

Lemma 2: (Shi et al. [20]) Let A be a (d,t)-CDA$(N;k,v)$. Then, A is also a (s,t)-CDA$(N;k,v)$, where $1\le s\le d-1$.

D. Lower bound for CDAs and combinatorial characteristics

By definition, a (d,t)-CDA is a special CCA of strength t. One significance of using the CDA to generate test suites is that any set of d consecutive t-way interaction faults can be determined from the outcomes. Further, it can detect if there are more than d consecutive t-way interactions causing the faults. Because the rows of a CDA correspond to the number of tests, a CDA of minimum size when other parameters are fixed is of considerable interest. The minimum N for which a (d,t)-CDA$(N;k,v)$ exists is referred to as the *consecutive detecting arrays number* (CDAN), denoted by (d,t)-CDAN(k,v).

Theorem 1: (Shi et al. [20]) Let t,k, and v be positive integers with $t<k$. Then, (d,t)-CDAN $(k,v)\ge(d+1)v^t$.

We call a (d,t)-CDA$(N;k,v)$ with $N=(d+1)v^t$ *optimum.* It is worth mentioning that optimum CDAs have useful applications in software testing because they contain minimum rows. To explore the combinatorial features of optimum CDAs, we need to introduce the notion of simple COAs. A COA$_\lambda(t,k,v)$ is *simple* if any $N\times(2t-i)$ subarray consisting of two consecutive t columns with i columns in common contains each $(2t-i)$-tuple at most once, where $0\le i\le t-1$. From this definition, it is obvious that a simple COA$_\lambda(t,k,v)$ can only exist if $\lambda\le v$.

Theorem 2: (Shi et al. [20]) Suppose t and k are two positive integers and $t<k$. Then, a simple COA$_{d+1}(t,k,v)$ is equivalent to an optimum (d,t)-CDA$((d+1)v^t;k,v)$.

III. LFSR SEQUENCE AND m-SEQUENCE

In this section, we review the definition of a linear feedback shift register (LFSR) sequence and an m-sequence and recall some relevant properties for later use.

Let $(s_i)_{i\ge0}$ be a sequence of elements from a finite field F_q. If there exist integers n such that $s_{n+i}=s_i$ for all $i\ge0$, then the sequence is said to be *periodic*. The smallest n with that property is the least period of the sequence. For example, the period of the sequence $(000121200000121200\cdots)$ is 9.

Let $f=c_0+c_1x+c_2x^2+\cdots+c_{t-1}x^{t-1}+x^t\in F_q[x]$ and $I=(b_0,b_1,\cdots,b_{t-1})\in F_q^t$. A *linear feedback shift register (LFSR)* sequence with characteristic polynomial f, denoted by $S(f,I)$, is defined as $S(f,I)=(a_0,a_1,\cdots,)$, where $a_i=b_i$, $0\le i<t$; $a_i=-c_{t-1}a_{i-1}-c_{t-2}a_{i-2}-\cdots-c_1a_{i-(t-1)}-c_oa_{i-t}$, and $i\ge t$.

Lemma 3: (Lidl and Niederreter, Theorem 8.33 [21]) If f is a primitive polynomial over F_q with $degf=t$ and initial values $I=(b_0,b_1,\cdots,b_{t-1})\in F_q^t\setminus(0,0,\cdots,0)$, then $S(f,I)$ has least period q^t-1.

Definition 1: (Maximal period sequence) An LFSR sequence over F_q generated by a primitive polynomial is a maximal (period) sequence. A maximal sequence is often referred to as an m-sequence.

If $S=(a_0,a_1,\cdots,)$ is a sequence, for a positive integer n, $C_i^n(S)=(a_i,a_{i+1},\cdots,a_{i+n-1})$ is called the subinterval of length n beginning in position i. Let $\delta\in F_q$, the subinterval $C_i^n(S)$ is a run of δ values of length n if $a_{i+j}=\delta$ for $0\le j\le n-1$ and $a_{i-1}\ne\delta, a_{i+n}\ne\delta$.

Next, we present some properties of m-sequences. Properties (1)–(4) below characterize the run property, also known as Golomb's second randomness postulate. Property (5) below is Golomb's fourth randomness postulate

Lemma 4: (Golomb and Gong, Section 5.2 [22]) If f is a primitive polynomial over F_q with $degf=t$ and nonzero initial values, $I=(b_0,b_1,\cdots,b_{t-1})\in F_q^t$. In a period of m-sequence $S(f,T)$, the following properties hold:

1) For $1\le l\le t-2$, the runs of $\delta\in F_q$ of length l occur $(q-1)^2q^{t-l-2}$ times.
2) The runs of $\delta\in F_q^*$ of length $t-1$ occur $q-2$ times.
3) The runs of 0 of length $t-1$ occur $q-1$ times.
4) The runs of $\delta\in F_q^*$ of length t occur once, and there is no run of zeros of length t.
5) Each nonzero-tuple in F_q^t appears exactly once as a consecutive element.

Arrays constructed from cyclic shifts of maximal sequences possess strong combinatorial properties and have been previously used to construct (ordered) orthogonal and covering arrays [23], [24]. The following lemma presents the positions of zeros in any two subintervals of $k=\frac{q^t-1}{q-1}$, beginning in positions that differ by a multiple of k.

Lemma 5: ([23]) Let $k=\frac{q^t-1}{q-1}$. If f is a primitive polynomial over F_q with $degf=t$ and nonzero initial values $I=(b_0,b_1,\cdots,b_{t-1})\in F_q^t$, then $S(f,I)$ has the following properties:

1. For any $i\ge0$, $C_i^k(S)$ contains exactly $\frac{q^{t-1}-1}{q-1}$ zeros.
2. For any $i\ge0,j\ge0$, the positions of zeros in $C_i^k(S)$ and $C_{i+jk}^k(S)$ are identical.

IV. A CLASS OF CDAS FROM m-SEQUENCE

In this section, we will construct a class of CDAs with minimum size from m-sequences. By Theorem 2, we only need to construct simple COAs. First, let $f=c_0+c_1x+c_2x^2+\cdots+c_{t-1}x^{t-1}+x^t\in F_q[x]$ be a primitive polynomial and $I=(a_0,a_1,\cdots,a_{t-1})\ne(0,0,\cdots,0)$. $S(f,I)$ is an m-sequence. Let $k=\frac{q^t-1}{q-1}$. Consider the following $q^t\times k$:

$$M=M(f,I)=\begin{bmatrix} C_0^k(S(f,I)) \\ C_1^k(S(f,I)) \\ \vdots \\ C_{q^t-2}^k(S(f,I)) \\ 0,0,\cdots,0 \end{bmatrix}$$

$$=\begin{bmatrix} a_0 & a_1 & \cdots & a_{k-1} \\ a_1 & a_2 & \cdots & a_k \\ \vdots & \vdots & & \vdots \\ a_{q^t-2} & a_{q^t-1} & \cdots & a_{q^t-2+k-1} \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

If A is an $N\times k$ array over a set V, then for a set of columns $B=\{b_1,b_2,\cdots,b_t\}$, B is called *covered* if the $N\times t$ subarray over the columns of B contains each t-tuple over V as a row at least once. A characterization of which sets of columns of M are covered was presented in [23].

Lemma 6: (Raaphorst et. al [23]) Let f be a primitive polynomial over F_q with $degf=t$ and nonzero initial values

$I = (b_0, b_1, \cdots, b_{t-1}) \in F_q^t$. Let $k = \frac{q^t-1}{q-1}$ and M be the $q^t \times k$ subinterval arrays of f. The following are equivalent:

(1) A set of t columns $i_1, i_2, \cdots, i_t$ is covered in M.

(2) There is no row r other than the all-zero row of M such that $M_{ri_1} = \cdots = M_{ri_t} = 0$, where $0 \le r \le q^t - 2$.

Lemma 7: Let $k = \frac{q^t-1}{q-1}$ and $C_i^k(S)$ be defined as above. Then, the run of zeros of length $t-1$ occurs at most once in $C_i^k(S)$, where $0 \le i \le q^t - 2$.

Proof. This conclusion follows from (3) and (4) of Lemma 4 and (2) of Lemma 5. $\square$

Lemma 8: Any subarray of M formed by t consecutive columns of M is covered.

Proof. It is an immediate consequence of Lemma 4 item (5) and the definition of M. $\square$

Now, we are ready to construct simple COAs from the subinterval subarray generated by the given primitive polynomial and nonzero initial value.

Theorem 3: If f is a primitive polynomial over F_q with $degf = t \ge 3$ and nonzero initial values $I = (b_0, b_1, \cdots, b_{t-1}) \in F_q^t$, then $M(f, I)$ is a simple COA$_q(t-1, \frac{q^t-1}{q-1}, q)$ over F_q.

Proof. The proof is given in the Appendix. $\square$

Example 1: Let $f = x^4 + 0x^2 + 2x + 2$ be a primitive polynomial over F_3 and initial values $I = (1, 0, 0, 0)$. A period of $S(f, T)$ is

$$\begin{array}{c}1000100110121100210201221010111122201121\\2000200220212200120102112020222211102212\end{array}$$

Then,

$$C_0^{40}(S) = 1000100110121100210201221010111122201121$$

and

$$C_{40}^{40}(S) = 2000200220212200120102112020222211102212.$$

Clearly, the runs of zeros of length 3 occur exactly once and there is no run of zeros of length 4. The positions of zeros in $C_{40}^{40}(S)$ and $C_0^{40}(S)$ are identical. M formed as above is a simple COA$_3(3, 40, 3)$.

For $t = 3$ in Theorem 3, we have another proof from combinatorial theory. For this, we introduce the notion of balanced incomplete block design (BIBD). Given a finite set X of elements, called points and a family B of k-element subsets of X, called blocks. (X, B) is a (v, k, λ)-BIBD, if $|X| = v$ and any pair of distinct points x and y in X is contained in exacltly λ blocks. A (v, k, λ)-difference set over an abelian group G is a set $d_i, i = 1, 2, \cdots, k$ of G such that each element in $G \setminus \{0\}$ occurs exactly λ times in the difference list $\{d_i - d_j \pmod v\}$, where $1 \le i \ne j \le k$.

Theorem 4: If f is a primitive polynomial over F_q with $degf = 3$ and nonzero initial values $I = (b_0, b_1, b_2) \in F_q^3$, then $M(f, I)$ is a simple COA$_q(2, q^2 + q + 1, q)$.

Proof. By the proof of Theorem 3, M is a COA$_q(2, q^2 + q + 1, q)$. We only need to prove that M is simple. When any two consecutive columns i_1, i_2 and j_1, j_2 have one column in common, without loss of generality, we suppose the columns are $\{i_1, i_2, j_2\}$. Clearly, each 3-tuple occurs exactly once. If any two consecutive columns i_1, i_2 and j_1, j_2 are disjoint, then we only need to consider that there is a row r other than the all-zero row of M such that $M_{ri_1} = M_{ri_2} = M_{rj_1} = M_{rj_2} = 0$. By Lemma 5, each $C_i^k(S)$ contains exactly $q+1$ zeros, where $0 \le i \le q^3 - 2$. Write $B = \{\{a_1, a_2, \cdots, a_{q+1}\} : M_{ia_1} = M_{ia_2} = \cdots = M_{ia_{q+1}} = 0$ for some $0 \le i \le q^2 + q\}$. Then, B is a $(q^2+q+1, q+1, 1)$-BIBD by Theorem 3 in [23]. The block set can be obtained by the translation of a $(q^2 + q + 1, q + 1, 1)$-difference set, which is the set $H = \{0 \le i < q^2 + q + 1 : a_i = 0\}$. If there is a row r other than the all-zero row of M such that $M_{ri_1} = M_{ri_2} = M_{rj_1} = M_{rj_2} = 0$, then the block set must contain the block $\{i, i+1, j, j+1\}$, where $i \ne j$. Because the difference 1 in the set $\{i, i+1, j, j+1\}$ occurs at least twice, this contradicts the fact that it can be obtained by the translation of a $(q^2 + q + 1, q + 1, 1)$-difference set. $\square$

Example 2: Let $f = x^3 + 0x^2 + 2x + 1$ be a primitive polynomial over F_3 and initial values $I = (0, 1, 2)$. Let $(012112011100202122102220010121120111002021221022$ $2001 \cdots)$ be an m-sequence generated by f and T. Form an 27×13 array as follows: $M = M(f, I) =$

$$\begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&2&1&1&2&0&1&1&1&0&0&2\\ 1&2&1&1&2&0&1&1&1&0&0&2&0\\ .&.&.&.&.&.&.&.&.&.&.&.&.\\ 0&2&0&2&1&2&2&1&0&2&2&2&0\\ 2&0&2&1&2&2&1&0&2&2&2&0&0\\ 0&2&1&2&2&1&0&2&2&2&0&0&1\\ 2&1&2&2&1&0&2&2&2&0&0&1&0\\ .&.&.&.&.&.&.&.&.&.&.&.&.\\ 0&1&0&1&2&1&1&2&0&1&1&1&0\\ 1&0&1&2&1&1&2&0&1&1&1&0&0 \end{bmatrix}$$

For any two consecutive columns, each 2-tuple occurs exactly $q = 3$ times. Thus, M is a COA$_3(2, 13, 3)$. For any three consecutive columns, each 3-tuple occurs exactly once by the property of the m-sequence. It remains to check that each 4-tuple occurs at most once for any two disjoint consecutive columns. The set $B = \{\{a_1, a_2, a_3, a_4\} : M_{ia_1} = \cdots = M_{ia_4}$ for some $0 \le i \le 12\}$ forms the set of blocks of a $(13, 4, 1)$-BIBD as follows:

$$\begin{array}{cccc} \{0,6,10,11\} & \{1,7,11,12\} & \{0,2,8,12\} & \{0,1,3,9\} \\ \{1,2,4,10\} & \{2,3,5,11\} & \{3,4,6,12\} & \{0,4,5,7\} \\ \{1,5,6,8\} & \{2,6,7,9\} & \{3,7,8,10\} & \{4,8,9,11\} \\ \{5,9,10,12\} & & & \end{array}$$

The block set can be obtained by the translation of the $(13, 4, 1)$-difference set $\{0, 6, 10, 11\}$. It can be concluded that there is no row r other than the all-zero row of M such that $M_{ri_1} = M_{ri_2} = M_{rj_1} = M_{rj_2} = 0$.

Theorem 5: If f is a primitive polynomial over F_q with $degf = t \ge 3$ and nonzero initial values $I = (b_0, b_1, \cdots, b_{t-1}) \in F_q^t$, then $M(f, I)$ is a $(q-1, t-1)$-CDA$(\frac{q^t-1}{q-1}, q)$.

Proof. The conclusion follows from Theorem 2 and 3. $\square$

Table VI presents some optimal CDAs over F_p from Theorem 5 with $p = 2, 3, 5, 7$. The first column lists the finite field F_p. The second and third columns present degree and primitive polynomial, respectively. The strength t and the number of factors k of optimal CDAs are given in the last two columns.

TABLE VI
OPTIMAL CDAS OVER F_p WITH $p = 2, 3, 5, 7$

F_p	degree n	$f(x)$	t	k
	3	x^3+x+1	2	7
	4	x^4+x+1	3	15
	5	x^5+x^2+1	4	31
$p=2$	6	x^6+x+1	5	63
	7	x^7+x+1	6	127
	8	$x^8+x^4+x^3+x^2+1$	7	255
	9	x^9+x^4+1	8	511
	3	x^3+2x+1	2	13
	4	x^4+x+2	3	40
	5	x^5+2x+1	4	121
$p=3$	6	x^6+x+2	5	364
	7	x^7+2x^2+1	6	1093
	8	x^8+x^3+2	7	3280
	9	x^9+2x^4+1	8	9841
	3	x^3+3x+2	2	31
	4	x^4+x^2+2x+2	3	156
	5	x^5+4x+2	4	781
$p=5$	6	x^6+x+2	5	3906
	7	x^7+3x+2	6	19531
	8	x^8+x^2+2x+3	7	97656
	9	x^9+2x^4+3	8	488281
	3	x^3+3x+2	2	57
	4	x^4+x^2+3x+5	3	400
	5	x^5+x+4	4	2801
$p=7$	6	x^6+3x^2+x+5	5	19608
	7	x^7+6x+2	6	137257
	8	x^8+x+3	7	960800
	9	x^9+3x^2+4	8	6725601

V. CONCLUSION

Combinatorial testing can detect failures triggered by interactions of parameters in the Software Under Test. The test suite used for combinatorial testing can often be represented as a mathematical object called covering arrays. Testing with a CA can indicate the presence or absence of interaction faults. However, testing with a CA provides little information to assist in the correction of interaction faults. Following practical situations, tests to reveal the location of interaction faults are of interest. Colbourn and McClary formalized the problem of the nonadaptive location of interaction faults under the hypothesis that a system contains some number d of faults, each involving some number t of interacting factors. They proposed the notion of detecting arrays to solve this problem. Using a (d,t)-DA as a test suite allows us to determine and identify them if the number of faulty interactions is d or less. If there exist more than d interaction faults, a (d,t)-DA can return specific results to indicate whether the number of faulty interactions is more than d.

Similar to CAs, CCAs can be used to generate test suites for combinatorial testing of neighboring factors to indicate the presence or absence of faulty interactions. However, even when a failure occurs, it is not always possible to locate which interaction causes the failure. Hence, the notion of consecutive detecting arrays was proposed. Consecutive detecting arrays have properties similar to those of detecting arrays, but they can be used to generate test suites for neighbor combinatorial testing. In this study, we demonstrated the construction of a class of consecutive detecting arrays from m-sequences, LFSR sequences that attain a maximum possible period. Consequently, a number of optimum CDAs were then obtained based on the existence of an m-sequence. Finding the primitive trinomial for constructing m-sequences is more complicated, but it is worth further research.

APPENDIX A
PROOF OF THEOREM 3

Proof. Clearly, M is a $q^t \times k$ array with entries from a finite field F_q. First, we prove that M is a $\mathrm{COA}_q(t-1,k,q)$, where $k = \frac{q^t-1}{q-1}$, i.e., for any consecutive $t-1$ column, each $(t-1)$-tuple occurs exactly q times as rows in the subarray consisting of $(t-1)$ columns. By Lemma 8, any consecutive t columns are covered. Each t-tuple occurs exactly once as a row in the subarray consisting of t columns because M has q^t rows. Thus, each $(t-1)$-tuple occurs exactly q times and M is a $\mathrm{COA}_q(t-1,k,q)$. It remains to prove that M is simple. This task is divided into two cases.

Case 1 : Two consecutive $t-1$ columns $\{i_1, i_2, \cdots, i_{t-1}\}$ and $\{j_1, j_2, \cdots, j_{t-1}\}$ have no common column.

In this case, we only need to consider that there is a row r other than the all-zero row of M such that $M_{ri_1} = M_{ri_2} = \cdots = M_{ri_{t-1}} = M_{rj_1} = M_{rj_2} = \cdots = M_{rj_{t-1}} = 0$, where $0 \le r \le q^t - 2$. Otherwise, there is no row r other than the all-zero row of M such that $M_{rl_1} = M_{rl_2} = \cdots = M_{rl_t} = 0$, where $\{l_1, l_2, \cdots, l_t\} \subset \{i_1, i_2, \cdots, i_{t-1}, j_1, j_2, \cdots, j_{t-1}\}$. By Lemma 6, $\{l_1, l_2, \cdots, l_t\}$ is covered in M. Each t-tuple from the t columns occurs exactly once because M has q^t rows. Thus, each $(2t-2)$-tuple occurs at most once. By Lemma 7, the runs of zeros of length $t-1$ occur at most once in $C_i^k(S)$, where $0 \le i \le q^t - 2$. Consequently, there is no row r other than the all-zero row of M such that $M_{ri_1} = M_{ri_2} = \cdots = M_{ri_{t-1}} = M_{rj_1} = M_{rj_2} = \cdots = M_{rj_{t-1}} = 0$, where $0 \le r \le q^t - 2$.

Case 2 : Two consecutive $t-1$ columns $\{i_1, i_2, \cdots, i_{t-1}\}$ and $\{j_1, j_2, \cdots, j_{t-1}\}$ have i common columns, where $1 \le i \le t-2$

In this case, there is no run of zeros of length t by Lemma 4. Consequently, there is no row r other than the all-zero row of M such that $M_{r,l} = M_{r,l+1} = \cdots = M_{r,l+t-1} = 0$, where $\{l, l+1, l+t-1\} \subset \{i_1, i_2, \cdots, i_{t-1}\} \cup (\{j_1, j_2, \cdots, j_{t-1}\} \setminus (\{i_1, i_2, \cdots, i_{t-1}\} \cap \{j_1, j_2, \cdots, j_{t-1}\}))$. By Lemma 6, $\{l, l+1, \cdots, l+t-1\}$ is covered in M. Similar to the proof of Case 1, we can prove that M is a simple $\mathrm{COA}_q(t-1,k,q)$. $\square$

REFERENCES

[1] C. Nie and H. Leung, "A survey of combinatorial testing," *ACM Computing Surveys,* vol. 43, no. 2, pp. 1-29, 2011.

[2] D. R. Kuhn, R. N. Kacker and Y. Lei, *Introduction to combinatorial testing*, Boca Raton, FL: CRC Press, 2013.

[3] C. J. Colbourn, S. S. Martirosyan, G. L. Mullen, D. E. Shasha, G. B. Sherwood and J. L. Yucas, "Products of mixed covering arrays of strength two," *Journal of Combinatorial Designs,* vol. 14, pp. 124-138, 2006.

[4] L. J. Lun, X. Chi and H. Xu "Testing Approach of Component Interaction for Software Architecture," *IAENG International Journal of Computer Science,* vol. 45, no. 2, pp. 353-363, 2018.

[5] C. J. Colbourn, "Combinatorial aspects of covering arrays," *Le Matematiche (Catania),* vol. 58, pp. 121-167, 2004.

[6] C. J. Colbourn, "Strength two covering arrays: Existence tables and projection," *Discrete Mathematics,* vol. 308, pp. 772-786, 2008.

[7] C. J. Colbourn and D. W. McClary, "Locating and detecting arrays for interaction faults," *Journal of Combinatorial Optimization,* vol. 15, pp. 17-48, 2008.

[8] N. J. Sloane, "Covering arrays and intersecting codes," *Journal of Combinatorial Designs,* vol. 1, pp. 51-63, 1993.

[9] B. Stevens, L. Moura and E. Mendelsohn, "Lower bounds for transversal covers," *Designs, Codes and Cryptography,* vol. 15, no. 3, 279-299, 1998.

[10] J. Torres-Jimenez and E. Rodriguez-Tello, "New bounds for binary covering arrays using simulated annealing," *Information Sciences,* vol. 185, PP. 137-152, 2012.

[11] S. Sabharwal, P. Bansal and N. Mittal, "Construction of t-way covering arrays using genetic algorithm," *International Journal of System Assurance Engineering & Management,* vol. 8, no. 2, pp. 264-274, 2017.

[12] H. Y. Wu, C. H. Nie, F. C. Kuo, H. Leung and C. J. Colbourn, "A discrete Particle Swarm Optimization for covering array generation," *IEEE Transaction on Evolutionary Computation,* vol. 19, no. 4, pp. 575-591, 2015.

[13] B S. Ahmeda, K. Z. Zamli and C. P. Lim, "Application of Particle Swarm Optimization to uniform and variable strength covering array construction," *Applied Soft Computing,* vol. 12, pp. 1330-1347, 2012.

[14] S. J. Saputra, B. Subartini, J. H. F. Purba, S. Supian and Y. Hidayat, " An Application of Genetic Algorithm Approach and Cobb-Douglas Model for Predicting the Gross Regional Domestic Product by Expenditure-Based in Indonesia ," *Engineering Letters,* vol. 27, no. 3, pp. 411-420, 2019.

[15] B. Tutuko, S. Nurmaini, Saparudin and P. Sahayu "Route Optimization of Non-holonomic Leader-follower Control Using Dynamic Particle Swarm Optimization," *IAENG International Journal of Computer Science,* vol. 46, no. 1, pp. 1-11, 2019.

[16] A. P. Godbole, M. V. Koutras and F. S. Milienos, "Binary consecutive covering arrays," *Annals of the Institute of Statistical Mathematics,* vol. 63, no. 3, pp. 559-584, 2011.

[17] K. Meagher and B. Stevens, "Covering arrays on graphs," *Journal of Combinatorial Theory, Series B,* vol. 95, pp. 134-151, 2005.

[18] C. Shi, Y. Tang and J. X. Yin, "The equivalence between optimal detecting arrays and super-simple OAs," *Designs, Codes and Cryptography,* vol, 62, pp. 131-142, 2012.

[19] C. Shi and J. X. Yin, "Existence of super-simple $\mathrm{OA}_\lambda(3,5,v)'$s," *Designs, Codes and Cryptography,* vol, 72, pp. 369-380, 2014.

[20] C. Shi, L. Jiang and A. Y. Tao, "Consecutive Detecting Arrays for Interaction Faults," *arXiv:1905.10914 [math.CO]*, 2019.

[21] R. Lidl and H. Niederreiter, *Finite fields*, Cambridge, England: Cambridge University Press, 1997.

[22] S. W. Golomb and G. Gong, *Signal Design for Good Correlation for Wireless Communication, Cryptography, Radar*, Cambridge, U.K.: Cambridge Univ. Press, 2005.

[23] S. Raaphorst, L. Moura and B. Stevens, "A construction for strength-3 covering arrays from linear feedback shift register sequences," *Designs, Codes and Cryptography*, vol. 73, no. 3, pp. 949-968, 2014.

[24] A. G. Castoldi, L. Moura, D. Panario and B. Stevens, "Ordered Orthogonal Array Construction Using LFSR Sequences," *IEEE Transaction on Information Theory,* vol. 63, no. 2, pp. 1336-1346, 2017.

Ce Shi was born in Bengbu, Anhui, China in 1983. He received a BSc in Mathematics a nd Applied Mathematics from Anhui Polytechnic University (2006), M.S. (2009) and Ph.D. (2012) in Applied Mathematics from Soochow University. He has worked in the School of Statistics and Mathematics at Shanghai Lixin University of Accounting and Finance since 2012. He became a reviewer of Mathematical Reviews in 2013. His research interests include combinatorics, software testing and coding theory.

Aiyuan Tao was born in Wuwei, Anhui, China in 1970. He received a BSc in Mathematics from Anqing Normal University (1992), M.S. (2004) and Ph.D. (2010) in Economics from Shanghai University of Finance and Economics. He has worked in the School of International Economics and Trade at Shanghai Lixin University of Accounting and Finance since 2005. He was a reviewer of Journal of Applied Mathematics in 2013. His research interests include game theory, model simulation and data analysis.