Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 33(11): 6610–6630.
DOI: 10.3934/era.2025292
Received: 22 August 2025
Revised: 18 October 2025
Accepted: 24 October 2025
Published: 11 November 2025

Research article

Mixed-level detecting arrays on graphs

Quanrui Zhang[1], **Ce Shi**[2] **and Yonggang Yi**[3,*]

[1] School of Information Science and Technology, Yunnan Normal University, Kunming 650500, China

[2] School of Statistics and Mathematics, Shanghai Lixin University of Accounting and Finance, Shanghai 201209, China

[3] Shanghai Normal University Tianhua College, Shanghai 201815, China

* **Correspondence:** Email: yonggangyi@163.com.

Abstract: The purpose of this paper was to develop a unified combinatorial framework for fault localization in heterogeneous software systems, where parameters may have different numbers of levels. Specifically, we investigated mixed-level detecting arrays (MDAs) on graphs, extending the classical detecting array model to accommodate non-uniform factor structures. In this paper, we established an optimality criterion that minimizes the number of required test cases and analyzes the structural and combinatorial properties of optimal MDAs on graphs. Furthermore, several constructive methods were proposed to generate optimal arrays, and existence results were derived that achieve the theoretical lower bounds. The findings enhance the theoretical understanding of detecting arrays in graph-based settings and provide practical guidelines for designing cost-efficient and fault-sensitive test suites in complex, heterogeneous software systems.

Keywords: combinatorial testing; detecting arrays; mixed-level factors; graph-based testing; optimality criterion

1. Introduction

In the era of large-scale, component-based software systems, ensuring reliability and robustness has become a central challenge in software engineering. As such systems grow increasingly complex, their configuration spaces expand exponentially, rendering exhaustive testing infeasible. This motivates the demand for efficient testing strategies that can detect potential faults prior to deployment. Combinatorial interaction testing (CIT) has emerged as a powerful methodology that significantly reduces test suite sizes while retaining strong fault detection capability. Instead of enumerating all possible parameter combinations, CIT focuses on covering all t-way interactions

among parameters, guided by empirical evidence that most software failures arise from the interaction of only a small subset of parameters [1,2]. By targeting these critical subsets, CIT provides a practical balance between test coverage and efficiency.

A fundamental combinatorial object in CIT is the covering array (CA), which guarantees that every t-tuple of values across any t factors appears in at least one test. Covering arrays have been extensively investigated in both combinatorics and software engineering, with applications ranging from network protocols and embedded systems to user interface testing. Representative constructions and applications can be found in [3–6], while algorithmic generation methods were surveyed in [7–9]. Constructions using linear feedback shift register (LFSR) sequences were studied in [10–12]. In practice, software systems often consist of heterogeneous components, whose parameters admit different domain sizes. To address this situation, mixed covering arrays (MCAs) were introduced [13, 14], generalizing standard covering arrays by allowing each factor (column) to take values from a distinct set. Formally, a *mixed covering array* of type $(v_1, v_2, \ldots, v_k)$, size N, strength t, and index λ, denoted by $\mathrm{MCA}_\lambda(N; t, k, (v_1, v_2, \ldots, v_k))$, is an $N \times k$ array A in which every t-tuple across any t columns appears at least λ times, with column i taking values from a set V_i of size v_i. Without loss of generality, we assume $v_1 \leq v_2 \leq \cdots \leq v_k$, and V_i is often taken as the cyclic group $\mathbb{Z}_{v_i}$. When all v_i are equal, the array reduces to a standard $\mathrm{CA}_\lambda(N; t, k, v)$, and if $N = \lambda v^t$, it becomes an orthogonal array $\mathrm{OA}_\lambda(t, k, v)$. For $t = 2$, the notation may be simplified as $\mathrm{MCA}_\lambda(N; k, (v_1, \ldots, v_k))$, and repeated levels are written compactly using exponents (e.g., 2^2 denotes two factors with two levels each).

In many real-world systems, not all parameter interactions are relevant because of structural, physical, or logical constraints. To model this, covering arrays on graphs were introduced [15], where parameters are represented as vertices and only interactions along edges must be covered, thereby reducing the required number of tests. This was further generalized to mixed covering arrays on graphs [16], accommodating mixed-level factors subject to graph-based interaction constraints. Formally, an interaction graph G specifies feasible parameter interactions.

Definition 1.1. (Mixed Covering Arrays on Graphs [16]) Let G be an interaction graph with k vertices $V_1, V_2, \cdots V_k$, each associated with a weight $v_1, v_2, \ldots, v_k$. A mixed covering array on G, denoted by $\mathrm{MCA}_\lambda(N, G, (v_1, v_2, \ldots, v_k))$, is an $N \times k$ array A satisfying the following conditions:

1) The entries in column i are drawn from $\mathbb{Z}_{v_i}$;
2) Column i corresponds to vertex $V_i \in V(G)$ with $w_G(V_i) = v_i$;
3) For every edge $\{V_i, V_j\} \in E(G)$, all possible 2-tuples of values appear at least λ times in columns i and j.

In other words, in an MCA on an interaction graph G, each column corresponds to a vertex, and only column pairs associated with edges of G must cover all 2-way combinations. Without loss of generality, we assume $v_1 \leq v_2 \leq \cdots \leq v_k$, and column i corresponds to vertex V_i with weight v_i. This correspondence will be assumed implicitly in what follows.

As a special case, if G is the complete graph K_k, the MCA on G reduces to the classical MCA of strength 2. More generally, a covering array on a graph is denoted by $\mathrm{CA}_\lambda(N, G, v)$, where all vertices have the same level v. The concept related to covering arrays on graphs is the covering arrays avoiding forbidden edges (CAFEs), in which certain pairwise interactions are prohibited while all other interactions must be covered [17]. Graph-based MCAs can significantly reduce test suite size

while maintaining essential coverage. For example, consider a configuration system from a network game software (NGS) application adapted from [2], which will serve as a running example throughout the paper.

Example 1.1. Table 1 lists four configuration parameters, each with different levels. Exhaustive testing requires $2 \cdot 3 \cdot 4^2 = 96$ test cases. Using an MCA$(16; 2, 2^1 3^1 4^2)$, as shown in [14], can achieve complete 2-way coverage with only 16 tests. If further system knowledge reveals that, for instance, parameters F_3 (Browser) and F_4 (Access) do not directly interact, the edge $\{V_{F_3}, V_{F_4}\}$ can be removed from the complete graph on four vertices. The resulting mixed covering array on the interaction graph G then requires only 12 tests (see Table 2).

Table 1. Configuration parameters for a NGS.

	Parameter	Values			
F_1	Audio	Digital (0)	Creative (1)		
F_2	OS	Windows (0)	Linux (1)	Mac (2)	
F_3	Browser	Chrome (0)	Edge (1)	Firefox (2)	Opera (3)
F_4	Access	VPN (0)	ISDL (1)	Modem (2)	ZTNA (3)

Despite their efficiency, (mixed) covering arrays (on graphs) have an intrinsic limitation: they can detect the presence of interaction faults but cannot identify the specific faulty interactions. For instance, in Table 2, if all tests pass except for the first one, it is impossible to determine which of the four 2-way interactions $(Windows, Chrome)$, $(Windows, ISDL)$, $(Digital, Windows)$, and $(Digital, ISDL)$ is responsible for the fault. To overcome this diagnostic limitation, *detecting arrays* (DAs) were introduced by Colbourn and McClary [18]. DAs strengthen covering arrays by ensuring that distinct faults lead to distinct failing tests, enabling fault localization. More recently, detecting arrays on graphs have been studied [19], combining the diagnostic capability of DAs with graph-based constraints. As a special case, consecutive DAs of strength 2 are equivalent to detecting arrays on paths, capable of localizing faults in adjacent interactions [20]. Constructions using M-sequences have also been proposed [21]. However, most existing studies are restricted to uniform-level assumptions, where all factors are treated as having the same number of levels. Such a simplification greatly limits the applicability and realism of detecting array models, since many practical software and system testing scenarios involve heterogeneous parameters with varying domain sizes. To address this limitation, we introduce the concept of mixed-level detecting arrays (MDAs) on graphs, which are capable of simultaneously capturing mixed-level factor structures and graph-based interaction constraints that arise in real systems. This new framework not only generalizes the classical detecting array model but also integrates the ideas of mixed covering arrays on graphs, thereby providing a unified and more flexible combinatorial foundation for fault localization in complex heterogeneous environments.

The remainder of the paper is organized as follows. Section 2 formally defines MDAs on graphs and illustrates their utility with practical examples. Section 3 presents an optimality criterion and explores the combinatorial structure of optimal MDAs on graphs. Section 4 develops construction methods and existence results for MDAs on paths, cycles, and binary trees. Section 5 concludes with future research directions.

Table 2. An MCA$(12; G, (2, 3, 4, 4))$.

Test case	F_1	F_2	F_3	F_4
1	Digital	Windows	Chrome	ISDL
2	Creative	Windows	Edge	Modem
3	Creative	Windows	Firebox	ZTNA
4	Creative	Windows	Opera	VPN
5	Creative	Linux	Chrome	ISDL
6	Digital	Linux	Edge	Modem
7	Digital	Linux	Firebox	ZTNA
8	Digital	Linux	Opera	VPN
9	Digital	Mac	Chrome	Modem
10	Creative	Mac	Edge	ZTNA
11	Digital	Mac	Firebox	VPN
12	Creative	Mac	Opera	ISDL

2. Preliminaries

In this section, we present the basic concepts, notations, and structures that form the foundation for the study of MDAs on graphs. These preliminaries provide the combinatorial and graph-theoretic background necessary for the constructions and existence results established in subsequent sections.

2.1. Definitions and terminology

Let $I_n = \{1, 2, \ldots, n\}$ denote the set of the first n positive integers. We begin by formalizing the underlying testing framework. Consider a system with k distinct factors (also called parameters or components), where each factor $j \in I_k$ assumes values from the finite set $\mathbb{Z}_{v_j} = \{0, 1, \ldots, v_j - 1\}$, the residue class ring modulo v_j. Without loss of generality, we assume that $v_1 \leq v_2 \leq \cdots \leq v_k$. A test case (or simply a test) is a k-tuple $(x_1, x_2, \ldots, x_k) \in \mathbb{Z}_{v_1} \times \mathbb{Z}_{v_2} \times \cdots \times \mathbb{Z}_{v_k}$, representing a concrete assignment of levels to all factors. Each test yields a binary outcome: *pass* or *fail*. A t-way interaction (or interaction of strength t) is a set of factor-value pairs of the form $T = \{(j_i, \sigma_{j_i}) | 1 \leq i \leq t\}$, where $\{j_1, j_2, \ldots, j_t\} \subseteq I_k$ with $j_1 < j_2 < \ldots < j_t$ and $\sigma_{j_i} \in \mathbb{Z}_{v_{j_i}}$. A test case $R = (x_1, x_2, \ldots, x_k)$ is said to cover an interaction T if $x_{j_i} = \sigma_{j_i}$ for all $i = 1, \ldots, t$. Each test therefore simultaneously covers $\binom{k}{t}$ different t-way interactions.

A test suite is a finite collection of such tests, with their collective outcomes used to detect faults. A test fails if and only if it covers at least one faulty interaction. Therefore, a test suite can be represented as an $N \times k$ array $A = (a_{ij})$, where each row corresponds to a test and $a_{ij} \in \mathbb{Z}_{v_j}$. For a given interaction T, define

$$\rho(A, T) = \{i : a_{ij_r} = x_r, 1 \leq r \leq t\},$$

the set of row indices in which T is covered. For a set of interactions $\mathcal{T}$, let

$$\rho(A, \mathcal{T}) = \bigcup_{T \in \mathcal{T}} \rho(A, T).$$

An array A is called a mixed covering array of strength t, denoted by $\mathrm{MCA}_\lambda(N; t, k, (v_1, v_2, \ldots, v_k))$, if

$|\rho(A,T)| \geq \lambda$ for every t-way interaction T of A. Such coverage ensures that all interactions of strength t can potentially be observed in the test outcomes.

Following Colbourn and McClary [18], a mixed covering array A is called a (d,t)-detecting array, denoted by (d,t)-DA$(N;k,(v_1,v_2,\ldots,v_k))$, if for every t-way interaction T and any subset $\mathcal{T} \subseteq \mathcal{I}_t$ with $|\mathcal{T}| = d$, the following condition holds:

$$\rho(A,T) \subseteq \rho(A,\mathcal{T}) \Leftrightarrow T \in \mathcal{T}, \tag{2.1}$$

where $\mathcal{I}_t$ denotes the set of all t-way interactions. This property guarantees the unique identification of any set of up to d interaction faults.

We now extend this notion to a graph-based setting. Let $G = (V,E)$ be a simple weighted interaction graph with k vertices, each corresponding to a factor of level size v_i, ordered as $v_1 \leq v_2 \leq \cdots \leq v_k$. We assume that G contains no isolated vertices and exclude cases where two degree-one vertices are adjacent, as such factor pairs can be independently tested. Unless otherwise specified, we assume throughout the remainder of this paper that the interaction graph G possesses the properties described above.

Let $A = (a_{ij})_{N\times k}$ be an MCA$_\lambda(N,G,(v_1,v_2,\cdots,v_k))$. A 2-way interaction $T = \{(i,x_i),(j,x_j)\}$ is called *valid* if $\{V_i,V_j\} \in G$, with $x_i \in \mathbb{Z}_{v_i}$ and $x_j \in \mathbb{Z}_{v_j}$. Let $V\mathcal{I}_2$ denote the set of all such valid 2-way interactions, whose total number is

$$|V\mathcal{I}_2| = \sum_{(V_i,V_j)\in E} v_i v_j.$$

Coverage of all valid interactions suffices in practical systems, leading to the following formal definition.

Definition 2.1. *(Mixed-Level Detecting Arrays on Graphs) Let G be a simple weighted graph with vertex set $V_1,\ldots,V_k$ and weights $v_1 \leq v_2 \leq \ldots, \leq v_k$. An $N\times k$ array $A = (a_{ij})$ with $a_{ij} \in \mathbb{Z}_{v_j}$ is called a mixed-level detecting array on G, denoted by MDA$(N;d,G,(v_1,v_2,\cdots,v_k))$, if for any subset $\mathcal{T} \subseteq V\mathcal{I}_2$ with $|\mathcal{T}| = d$ and any $T \in V\mathcal{I}_2$, $\rho(A,T) \subseteq \rho(A,\mathcal{T}) \Leftrightarrow T \in \mathcal{T}$.*

The expression $\rho(A,T) \subseteq \rho(A,\mathcal{T}) \Leftrightarrow T \in \mathcal{T}$ means that an interaction T belongs to the fault set $\mathcal{T}$ if and only if every test case that covers T also covers at least one interaction in $\mathcal{T}$. This ensures that distinct fault sets produce distinct sets of failing test cases, thereby enabling precise fault localization. It is easy to observe that if $T \in \mathcal{T}$, then $\rho(A,T) \subseteq \rho(A,\mathcal{T})$ must hold. As a direct consequence, $T \notin \mathcal{T} \Rightarrow \rho(A,T) \nsubseteq \rho(A,\mathcal{T})$, which will be used implicitly in later discussions. It is evident that the array A also qualifies as an MDA$(N;s,G,(v_1,v_2,\cdots,v_k))$, for any integer s such that $0 \leq s \leq d-1$. When G is the complete graph, this definition reduces to the classical $(d,2)$-detecting array. This indicates that MDAs on graphs can be seen as a natural extension of traditional detecting arrays with strength 2. If all factors have the same number of levels, i.e., $v_1 = \cdots = v_k = v$, the definition recovers the uniform-level detecting arrays on graphs as in [19]. This observation highlights that mixed-level detecting arrays on graphs serve as a natural generalization of their uniform level counterparts.

By setting $d = 0$ in the definition, we have $\rho(A,\mathcal{T}) = \emptyset$, which implies that $\rho(A,T) \neq \emptyset$ for every $T \in \mathcal{VI}_2$. In other words, an MDA$(N;0,G,(v_1,v_2,\cdots,v_k))$ is an array in which each valid 2-way interaction is covered in at least one row. This coincides with the concept of mixed covering arrays on graphs, as mentioned earlier. Therefore, an MCA$(N;G,(v_1,v_2,\cdots,v_k))$ is equivalent to an

MDA$(N; 0, G, (v_1, v_2, \cdots, v_k))$. As with traditional detecting arrays (DAs), certain parameter constraints must be met for an MDA on a graph G. From a practical standpoint, it is crucial that the parameter d satisfies $d \geq 1$. Additionally, since d is intrinsically linked to the number of levels of the factors, it cannot be arbitrarily large. In fact, a necessary condition on d is given by the following lemma.

Lemma 2.1. *Let $P_2(G)$ and $C_3(G)$ denote the path and cycle $u - v - w$, respectively. Let A be an MDA$(N; d, G, (v_1, v_2, \cdots, v_k))$, where G is a simple weighted graph. Then $d < \min\{m_1, m_2\}$, where*

$$m_1 = \min_{(u,v,w)\in P_2(G)} \{w_G(u), w_G(w)\}, \quad m_2 = \min_{\{u,v,w\}\in C_3(G)} \{w_G(u), w_G(v), w_G(w)\}.$$

Proof. From the assumption on G, we know that G must contain a path from u to w, i.e., a length-2 path of the form $u - v - w$. Let the corresponding factors be denoted by F_u, F_v, and F_w, respectively. Without loss of generality, assume that $w_G(u) \leq w_G(w)$. Suppose, for contradiction, that $d = w_G(u)$. Since A is an MDA$(N; d, G, (v_1, v_2, \ldots, v_k))$, consider the following set of valid 2-way interactions:

$$\mathcal{T} = \{(F_v, a), (F_u, i) \mid i \in \mathbb{Z}_{w_G(u)}\}.$$

Let $T = \{(F_v, a), (F_w, b)\}$ be a valid 2-way interaction involving F_v and F_w. Then, by the definition of MDAs on graphs, we have

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

which contradicts the assumption that A is a mixed level detecting array on G. Therefore, we conclude that $d < w_G(u) \leq \min\{w_G(u), w_G(w)\}$. Since this argument applies to every such path in $P_2(G)$, we obtain $d < m_1$. Similarly, if G contains a 3-cycle (i.e., $\{u, v, w\} \in C_3(G)$), then an analogous argument shows that $d < m_2$.

When the graph G is complete, the bound reduces to $d < v_1$, which aligns with Theorem 7.11 in [18] for the special case $t = 2$. This demonstrates that the parameter d serves as a natural generalization of the classical bound. The following example illustrates the validity of the lemma in practice.

Example 2.1. *The array obtained by transposing the following matrix is an MDA$(18; 2, G, (3, 2, 3))$, where G is a path graph with vertex weights 3, 2, and 3, corresponding to the factors F_1, F_2, and F_3, respectively.*

F_1	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2
F_2	0	0	1	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1
F_3	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2

Here, $d = 2$ meets the condition $d < min\{3, 3\} = 3$.

According to the definition, an MDA$(N; d, G, (v_1, v_2, \ldots, v_k))$ is essentially a special class of mixed-level covering arrays defined on graphs. The key advantage of using such detecting arrays (DAs) is their ability to precisely identify any set of up to d valid 2-way interaction faults based on the observed test outcomes. Furthermore, if the actual number of such faults exceeds d, this condition can also be detected. For practical applications of detecting arrays in fault localization and software testing, readers may refer to [18].

According to the definition, an MDA$(N; d, G, (v_1, v_2, \ldots, v_k))$ is essentially a special class of mixed level covering arrays defined on graphs. The key advantage of using such detecting arrays (DAs) lies

in their capability to precisely identify any set of up to d valid 2-way interaction faults based on the outcome of the tests. Moreover, if the actual number of such faults exceeds d, this condition can also be detected. For practical applications of detecting arrays in fault localization and software testing, the reader is referred to [18]. Since each row in an MDA on graphs represents a single test, minimizing the number of rows N while keeping other parameters fixed is particularly important in practical settings. The smallest possible value of N for which an MDA$(N; d, G, (v_1, v_2, \ldots, v_k))$ exists is referred to as the mixed-level detecting array number on graphs, denoted by MDAN$(d, G, (v_1, v_2, \ldots, v_k))$. An MDA on graphs that achieves this minimum size is called *optimal*. In the following section, we derive a lower bound for the function MDAN$(d, G, (v_1, v_2, \ldots, v_k))$ for G, and present a combinatorial property that characterizes the existence of optimal MDAs on graphs.

2.2. An application example

One important application of MDAs on graphs lies in the testing of information systems. As an illustrative example, consider the networked game software system (NGS) introduced earlier in the Introduction. Table 3 presents a test suite consisting of 24 selected tests out of all possible combinations. This test suite corresponds to an MDA$(24; 1, G, (2, 3, 4, 4))$, where the interaction graph is defined as $G = (V, E)$ with vertex set $V = \{F_1, F_2, F_3, F_4\}$ and edge set $E = \{\{F_1, F_2\}, \{F_1, F_3\}, \{F_1, F_4\}, \{F_2, F_3\}, \{F_2, F_4\}\}$.

To demonstrate the practical applicability of MDAs on graphs, we present a simulated fault localization scenario based on the test suite in Table 3. In this scenario, only Test 2 and 13 produce failures, while all other tests pass. By analyzing the interactions covered by these two failing tests, we observe that the interaction $(F_2, F_3) = (\text{Windows}, \text{ISDL})$ is the only valid 2-way interaction common to both tests. Let $\mathcal{T} = \{T\}$ denote the (hypothetical) set of faulty interactions. The set of rows covering $\mathcal{T}$ is then $\rho(A, \mathcal{T}) = \{2, 13\}$. According to the definition of MDAs on graphs, there exists no other interaction set $\mathcal{T}' \subseteq \mathcal{VI}_2$ with $|\mathcal{T}'| = 1$ such that $\rho(A, \mathcal{T}') = \{2, 13\}$. Therefore, the faulty interaction T is uniquely identifiable from the observed test outcomes.

This example clearly illustrates the fault localization capability of the proposed MDA$(24; 1, G, (2, 3, 4, 4))$. Unlike mixed covering arrays on graphs, which may only guarantee fault detection, our graph-based MDA enables precise identification of faulty interactions. Moreover, by incorporating interaction constraints through the graph G, the size of the test suite can be significantly reduced without sacrificing diagnostic power. In the given example, a full factorial design would require $2 \times 3 \times 4 \times 4 = 96$ tests. In contrast, the graph-based detecting array achieves the same level of fault localization with only 24 tests. This empirical result validates the efficiency and effectiveness of the proposed framework. It is particularly advantageous in practical testing scenarios such as software systems with heterogeneous configurations where reducing testing cost while maintaining high diagnostic accuracy is critically important.

Table 3. An MDA(24; 1, G, (2, 3, 4, 4)) for the NGS.

Test case	F_1: Audio	F_2: OS	F_3: Browser	F_4: Access
1	Digital	Windows	Chrome	VPN
2	Digital	Windows	Edge	ISDL
3	Digital	Windows	Firefox	Modem
4	Digital	Windows	Opera	ZTNA
5	Digital	Linux	Chrome	ISDL
6	Digital	Linux	Edge	Modem
7	Digital	Linux	Firefox	ZTNA
8	Digital	Linux	Opera	VPN
9	Digital	Mac	Chrome	Modem
10	Digital	Mac	Edge	ZTNA
11	Digital	Mac	Firefox	VPN
12	Digital	Mac	Opera	ISDL
13	Creative	Windows	Chrome	ISDL
14	Creative	Windows	Edge	Modem
15	Creative	Windows	Firefox	ZTNA
16	Creative	Windows	Opera	VPN
17	Creative	Linux	Chrome	Modem
18	Creative	Linux	Edge	ZTNA
19	Creative	Linux	Firefox	VPN
20	Creative	Linux	Opera	ISDL
21	Creative	Mac	Chrome	VPN
22	Creative	Mac	Edge	ISDL
23	Creative	Mac	Firefox	Modem
24	Creative	Mac	Opera	ZTNA

3. Optimality criterion and combinatorial characterization of MDAs on graphs

The primary goal of this section is twofold. First, we establish a lower bound for the function MDAN($d, G, (v_1, v_2, \ldots, v_k)$). Second, we examine the combinatorial properties of optimal mixed-level detecting arrays (MDAs) on graphs that attain this bound. To this end, we begin by introducing a benchmark for evaluating the optimality of an MDA($N; d, G, (v_1, v_2, \ldots, v_k)$). The following result parallels Lemma 2.1 in [22], implying that $|\rho(A, T)| \geq 2$ must hold for any valid 2-way interaction T in A. Since an MDA($N; d, G, (v_1, \ldots, v_k)$) is also an MDA($N; s, G, (v_1, \ldots, v_k)$) for any $0 \leq s \leq d-1$, this observation can be generalized as follows. Similar results for classical (non-graph-based) detecting arrays appear in [23].

Lemma 3.1. *Let A be an MDA($N; d, G, (v_1, v_2, \ldots, v_k)$). Then, for any valid 2-way interaction T in A, we have $|\rho(A, T)| \geq d + 1$.*

Applying Lemma 3.1, we immediately obtain the following lower bound, which serves as a benchmark for assessing the optimality of MDAs on graphs.

Theorem 3.2. *Let $G = (V, E)$ be a simple weighted graph, and define the pairwise weight $PW(G) = max\{v_iv_j : \{V_i, V_j\} \in E\}$. Then,*

$$MDAN(d, G, (v_1, v_2, \ldots, v_k)) \geq (d + 1)PW(G). \tag{3.1}$$

Proof. Let A be an MDA$(N; d, G, (v_1, \ldots, v_k))$. By the definition of PW(G), there exists an edge $\{V_i, V_j\}$ such that $v_iv_j = \text{PW}(G)$. By Lemma 3.1, each of the v_iv_j possible interactions on $\{F_i, F_j\}$ must appear at least $d + 1$ times. Consequently, we have $N \geq (d + 1)PW(G)$.

Due to the structural constraints imposed by G, the two factors with the highest levels may not participate in any valid 2-way interaction. Hence, $PW(G)$ may be strictly smaller than the product $v_{k-1}v_k$, even though these are the largest levels among all factors. Our objective is to construct an MDA$(N; d, G, (v_1, \ldots, v_k))$ with $N = (d + 1)PW(G)$, which we regard as *optimal*. Optimal MDAs on graphs are of practical significance in software testing, as they minimize the number of required test cases. To precisely characterize the properties and conditions of optimal MDAs on graphs, we introduce several key concepts below.

Let A be an MDA$(N; d, G, (v_1, v_2, \ldots, v_k))$, where G is a simple weighted graph. Consider a valid 2-way interaction $T_2 = \{((j_1, a_{j_1}), (j_2, q_{j_2})) : a_{j_1} \in \mathbb{Z}_{v_{j_1}}, a_{j_2} \in \mathbb{Z}_{v_{j_2}}\}$ corresponding to an edge $e = \{V_{j_1}, V_{j_2}\} \in E(G)$. For another edge $e' = \{V_i, V_j\} \in E(G)$, we define an extension of T_2, denoted by T, as follows:

1) If e and e' are vertex-disjoint ($|V(e) \cap V(e')| = 0$), then $T = \{((j_1, a_{j_1}), (j_2, a_{j_2}), (i, a_i), (j, a_j)) : a_i \in \mathbb{Z}_{v_i}, a_j \in \mathbb{Z}_{v_j}\}$;
2) If e and e' share exactly one vertex V_i ($|V(e) \cap V(e')| = 1$), add the non-common vertex of e' to T_2 to form $T = \{((j_1, a_{j_1}), (j_2, a_{j_2}), (j, a_j)) : a_j \in \mathbb{Z}_{v_j}\}$.

Definition 3.1. An array A is said to be *extendable* if, for every valid 2-way interaction T_2 and any collection of d extensions $T^{(i)}$ ($1 \leq i \leq d$) of T_2, the set $\rho(A, T_2) \setminus \bigcup_{i=1}^{d} \rho(A, T^{(i)})$ is non-empty. In other words, there exists at least one row covering T_2 but none of its d extensions simultaneously.

We are now ready to present a key combinatorial property of MDAs defined on graphs.

Theorem 3.3. Let A be an $N \times k$ array of type $(v_1, v_2, \ldots, v_k)$. Then A is an MDA$(N; d, G, (v_1, v_2, \ldots, v_k))$ if and only if it is an extendable MCA$_{d+1}(N; G, (v_1, v_2, \ldots, v_k))$.

Proof. "$\Leftarrow$" Suppose that A is not an MDA$(N; d, G, (v_1, v_2, \ldots, v_k))$. Then there exist a valid 2-way interaction $T = \{((j_1, a_{j_1}), (j_2, a_{j_2})) : a_{j_1} \in \mathbb{Z}_{v_{j_1}}, a_{j_2} \in \mathbb{Z}_{v_{j_2}}\}$ and a collection of d valid 2-way interactions $\mathcal{T} = \{T_1, T_2, \cdots, T_d\}$ such that $T \notin \mathcal{T}$ and $\rho(A, T) \subseteq \rho(A, \mathcal{T}) = \bigcup_{i=1}^{d} \rho(A, T_i)$. Construct an extension $T^{(i)}$ of T corresponding to each T_i as follows:

1) If the first coordinates in T_i coincide with T, let $T^{(i)}$ be any extension of T.
2) If T_i contains an element (f_i, a_i) where f_i does not appear as a first coordinate in T, then define $T^{(i)}$ as the interaction obtained by adding (f_i, a_i) to T.
3) If T_i contains two elements (f_i, a_i) and (f_j, a_j), with neither f_i nor f_j appearing in T, then define $T^{(i)}$ by adding both of these pairs to T.

Now consider any row $R \in \rho(A, T) \subseteq \rho(A, \mathcal{T})$. Then there exists some $T_i \in \mathcal{T}$ such that $R \in \rho(A, T_i)$. By the construction above, it follows that $R \in \rho(A, T^{(i)})$. Therefore, we conclude that $\rho(A, T) \setminus \bigcup_{i=1}^{d} \rho(A, T^{(i)}) = \emptyset$, which contradicts the assumption that A is extendable.

"⇒" Conversely, assume that there exists a valid 2-way interaction $T = \{((j_1, a_{j_1}), (j_2, a_{j_2})) : a_{j_1} \in \mathbb{Z}_{v_{j_1}}, a_{j_2} \in \mathbb{Z}_{v_{j_2}}\}$ and a collection of d extensions $T^{(i)}$ for $1 \le i \le d$ such that $\rho(A, T) \setminus \bigcup_{i=1}^{d} \rho(A, T^{(i)}) = \emptyset$. By definition, each $T^{(i)}$ is an extension of T, which implies $\rho(A, T^{(i)}) \subseteq \rho(A, T)$ for all i, and therefore $\rho(A, T) = \bigcup_{i=1}^{d} \rho(A, T^{(i)})$. Based on the structure of these extensions, we can associate each $T^{(i)}$ with a valid 2-way interaction T_i in the following manner:

1) If $T^{(i)} = \{((j_1, a_{j_1}), (j_2, a_{j_2}), (i, a_i), (j, a_j))\}$ with $a_i \in \mathbb{Z}_{v_i}$, $a_j \in \mathbb{Z}_{v_j}$, then the corresponding valid 2-way interaction T_i is defined as $T_i = \{(i, a_i), (j, a_j)\}$;
2) If $T^{(i)} = \{((j_1, a_{j_1}), (j_2, a_{j_2}), (j, a_j)\}$ with $a_j \in \mathbb{Z}_{v_j}$, then the corresponding interaction T_i is defined as either $T_i = \{(j_1, a_{j_1}), (j, a_j)\}$ or $T_i = \{(j_2, a_{j_2}), (j, a_j)\}$, depending on which vertex is shared.

Thus, we have $\rho(A, T) = \bigcup_{i=1}^{d} \rho(A, T^{(i)}) \subseteq \bigcup_{i=1}^{d} \rho(A, T_i)$ which contradicts the definition of a mixed-level detecting array (MDA) on graphs.

As an immediate consequence of Theorem 3.3, we obtain the following result.

Theorem 3.4. An optimal MDA$(N; d, G, (v_1, v_2, \ldots, v_k))$ that attains the lower bound in (3.1) is equivalent to an extendable MCA$_{d+1}(N; G, (v_1, v_2, \ldots, v_k))$ of optimal size $N = (d + 1)PW(G)$.

Theorem 3.4 establishes a structural equivalence between two fundamental combinatorial constructs under graph-based interaction constraints: optimal mixed-level detecting arrays (MDAs) and extendable mixed-level covering arrays (MCAs) defined on the same interaction graph G. In particular, any MDA$(N; d, G, (v_1, v_2, \ldots, v_k))$ that attains the theoretical lower bound in (3.1) is necessarily an extendable MCA$_{d+1}(N; G, (v_1, v_2, \ldots, v_k))$ of size $N = (d + 1)\mathrm{PW}(G)$. From a practical perspective, Theorem 3.4 provides a constructive framework for designing optimal MDAs when factor interactions are constrained by an underlying graph structure. Specifically, the construction of a minimal-size extendable MCA on G directly yields an MDA that is not only size-optimal but also facilitates efficient fault localization. This property is particularly advantageous in application domains such as software testing and system configuration, where factor interactions are often limited by architectural constraints. In these contexts, the ability to simultaneously achieve comprehensive interaction coverage and precise fault detection with minimal testing overhead is of critical importance.

4. Optimal MDAs on paths, cycles, and binary trees

In this section, we will construct a large number of optimal MDAs on paths, cycles, and binary trees by leveraging the notion of extendability and the equivalent characterization established in Theorem 3.4. To facilitate the combinatorial construction of optimal MDAs on graphs, we introduce the concept of simple mixed covering arrays defined over graphs.

Definition 4.1. An MCA$_\lambda(N; G, (v_1, v_2, \ldots, v_k))$ is said to be *simple* if it satisfies the following conditions:

1) For any two edges in G that share a common vertex, the corresponding $N \times 3$ subarray formed by the factors associated with the vertices of these two edges contains each possible combination of levels at most once.
2) For any two disjoint edges in G, the corresponding $N \times 4$ subarray formed by the factors at the endpoints of the two edges contains each possible combination of levels at most once.

Table 3 provides an example of a simple $\mathrm{MCA}_2(24; G, (2, 3, 4, 4))$, where G denotes a complete factor interaction graph except that the edge representing the interaction between the two factors of level 4 has been removed. When G is the complete graph on k vertices, this definition coincides with the standard super-simple MCA of strength 2 introduced in [24], in which the two conditions reduce to requiring that every $N \times 3$ subarray contains each possible 3-tuple of values from the selected columns at most once. Simple MCAs on graphs provide a sufficient condition for the existence of optimal MDAs on graphs, as formalized in the following result.

Theorem 4.1. If there exists a simple $\mathrm{MCA}_{d+1}(N; G, (v_1, v_2, \cdots, v_k))$ of optimal size $N = (d+1)PW(G)$, then there exists an optimal $\mathrm{MDA}(N; d, G, (v_1, v_2, \cdots, v_k))$ that achieves the lower bound given in (3.1).

Proof. Let B be a simple $\mathrm{MCA}_{d+1}(N; G, (v_1, v_2, \cdots, v_k))$ with $N = (d+1)PW(G)$. We aim to show that B is extendable. Suppose, for the sake of contradiction, that there exists a valid 2-way interaction T and d of its extensions $T^{(i)}$ $(i = 1, 2, \ldots, d)$ such that $\rho(B, T) \setminus \bigcup_{i=1}^{d} \rho(B, T^{(i)}) = \emptyset$. By Lemma 3.1, the interaction T must occur at least $d+1$ times in B. By the pigeonhole principle, at least one extension $T^{(i)}$ must appear in at least two rows, i.e., $|\rho(B, T^{(i)})| \geq 2$, which implies that a specific 3-tuple or 4-tuple of symbols occurs more than once in the same set of three or four columns of B, violating the simplicity condition. Hence, B must be extendable, and by Theorem 3.3, an optimal MDA on G can be constructed from it.

It is worth noting that any super-simple $\mathrm{MCA}_{d+1}(N; k, (v_1, v_2, \ldots, v_k))$ is also a simple $\mathrm{MCA}_{d+1}(N; G, (v_1, v_2, \ldots, v_k))$ for some weighted graphs G, whereas the converse is not necessarily true; the class of super-simple MCAs is strictly contained within the broader class of simple MCAs on graphs. Motivated by Theorem 4.1, which provides a sufficient condition for the existence of optimal MDAs on graphs via simple MCAs on graphs, we further explore combinatorial constructions of optimal MDAs on graphs using super-simple MCAs as foundational building blocks.

4.1. Constructions of (optimal) MDAs on paths, cycles, and binary trees

In this subsection, we present constructions of (optimal) MDAs on paths, cycles, and binary trees, derived from super-simple mixed covering arrays.

Construction 4.2. *Let $k \geq 3$ be an odd integer. Let $a_1 = 1, a_2, \ldots, a_{\frac{\varphi(k)}{2}}$ denote the multiplicative inverses in $\mathbb{Z}_k$, listed in increasing order. Suppose there exists a super-simple $MCA_{d+1}(N; k, (v_1, v_2, \cdots, v_k))$ of optimal size $N = (d+1)v_{k-1}v_k$. Then there exists a simple $MCA_{d+1}(N; G, (v_1, v_2, \cdots, v_k, v_1, v_{a_2+1}, \cdots, v_{a_2(k-1) \pmod k+1}, \cdots, v_1, v_{a_i+1}, \cdots, v_{a_i(k-1) \pmod k+1}, \cdots, v_1, v_{\frac{\varphi(k)}{2}+1}, \cdots, v_{a_{\frac{\varphi(k)}{2}}(k-1) \pmod k+1}, v_1)$, where G is a path whose length ranges from k to $k\frac{\varphi(k)}{2} + 1$.*

Proof. Let $A = (A_0, A_1, \cdots, A_{k-1})$ denote a super-simple $\mathrm{MCA}_{d+1}(N; k, (v_1, v_2, \cdots, v_k))$ of optimal size $N = (d+1)v_{k-1}v_k$. By number-theoretic properties, the elements $a_1 = 1, a_2, \cdots, a_{\frac{\varphi(k)}{2}} \in \mathbb{Z}_k$ satisfy $a_i + a_j \not\equiv 0 \pmod k$ for all $1 \leq i \neq j \leq \frac{\varphi(k)}{2}$. We construct a new array A' by concatenating sequences as

$$A' = (A_0, A_{a_1}, \cdots, A_{a_1 \cdot (k-2)}, A_{a_1 \cdot (k-1)}, \cdots, A_0, A_{a_i}, \cdots, A_{a_i \cdot (k-2)}, A_{a_i \cdot (k-1)}, \cdots, A_0),$$

with subscripts that are taken as modulo k, and $i = 1, 2, \ldots, \frac{\varphi(k)}{2}$. It is straightforward to verify that A' is a simple $\mathrm{MCA}_{d+1}(N; G, (v_1, v_2, \cdots, v_k, \cdots, v_1))$, where G forms a path beginning and ending at vertex v_1, with total length $k\frac{\varphi(k)}{2} + 1$. Since removing vertices from a path does not affect the simplicity or extendability, the conclusion holds.

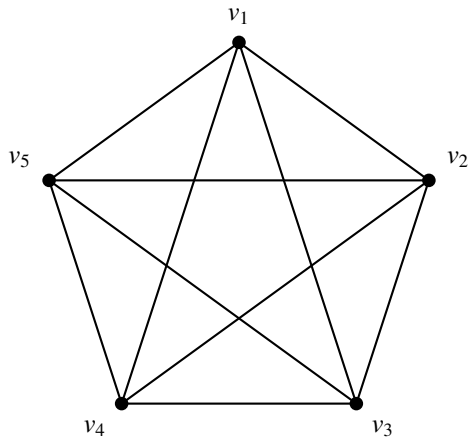

Figure 1. A complete graph on 5 vertices.

To illustrate this construction, we consider an example. A super-simple $\mathrm{MCA}_2(32; 2, 5, 3^1 4^4)$ was previously constructed in [25], which can equivalently be viewed as a super-simple $\mathrm{MCA}_2(32; K_5, 3^1 4^4)$. In Figure 1, the complete interaction graph K_5 is depicted, with vertex weights $3, 4, 4, 4, 4$ assigned to V_1, V_2, V_3, V_4, V_5, respectively. By applying Construction 4.2, we obtain a simple $\mathrm{MCA}_2(32; G, (3, 4, 4, 4, 4, 3, 4, 4, 4, 4, 3))$, where the corresponding path graph G is illustrated in Figure 2.

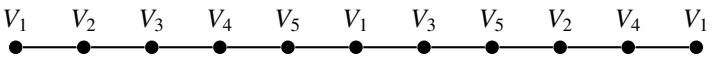

Figure 2. A path with length 11.

For even values of k, additional vertices can be incorporated into the construction without affecting the overall structure of the resulting array.

Construction 4.3. *Let $k \geq 6$ be an even integer, and let $a_1 = 1, a_2, \ldots, a_{\frac{\varphi(k)}{2}}$ denote the multiplicative inverses in $\mathbb{Z}k$, listed in increasing order. Suppose that there exists a super-simple $MCA_{d+1}(N; k, (v_1, v_2, \cdots, v_k))$ of optimal size $N = (d+1)v_{k-1}v_k$. Then, there exists a simple $MCA_{d+1}(N; G, (v_1, v_2, \cdots, v_k, v_1, v_{a_2+1}, \cdots, v_{a_2(k-1) \pmod k+1}, \cdots, v_1, v_{a_i+1}, \cdots, v_{a_i(k-1) \pmod k+1}, \cdots, v_1, v_{\frac{\varphi(k)}{2}+1}, \cdots, v_{a_{\frac{\varphi(k)}{2}}(k-1) \pmod k+1}, v_1, v_3, \cdots, v_{k-1}, v_1)$, where G is a path whose length ranges from k to $k\frac{\varphi(k)}{2} + \frac{k}{2} + 1$.*

Proof. The proof follows analogously to Construction 4.2, with the additional step of concatenating the segment $(A_0, A_2, \ldots, A_{k-2}, A_0)$ at the end of the array.

To construct MDAs on cycles, we modify Constructions 4.2 and 4.3 by removing the final column, thereby ensuring that the resulting graph forms a cycle. This leads to the following constructions.

Construction 4.4. *Let $k \geq 3$ be an odd integer, and let $a_1 = 1, a_2, \cdots, a_{\frac{\varphi(k)}{2}}$ be the multiplicative inverses in $\mathbb{Z}_k$, listed in increasing order. Suppose there exists a super-simple $MCA_{d+1}(N; k, (v_1, v_2, \cdots, v_k))$ of optimal size $N = (d+1)v_{k-1}v_k$. Then there exists a simple*

$MCA_{d+1}(N; G, (v_1, v_2, \cdots, v_k, v_1, v_{a_2+1}, \cdots, v_{a_2(k-1) \pmod k+1}, \cdots, v_1, v_{a_i+1}, \cdots, v_{a_i(k-1) \pmod k+1}, \cdots, v_1, v_{\frac{\varphi(k)}{2}+1}, \cdots, v_{a_{\frac{\varphi(k)}{2}}(k-1) \pmod k+1})$, *where* G *is a cycle of length* ki *for* $1 \le i \le \frac{\varphi(k)}{2}$.

Construction 4.5. *Let* $k \ge 6$ *be an even integer, and let* $a_1 = 1, a_2, \ldots, a_{\frac{\varphi(k)}{2}}$ *denote the multiplicative inverses in* $\mathbb{Z}_k$*, arranged in increasing order. Suppose there exists a super-simple* $MCA_{d+1}(N; k, (v_1, v_2, \cdots, v_k))$ *of optimal size* $N = (d+1)v_{k-1}v_k$*. Then there exists a simple* $MCA_{d+1}(N; G, (v_1, v_2, \cdots, v_k, v_1, v_{a_2+1}, \cdots, v_{a_2(k-1) \pmod k+1}, \cdots, v_1, v_{a_i+1}, \cdots, v_{a_i(k-1) \pmod k+1}, \cdots, v_1, v_{\frac{\varphi(k)}{2}+1}, \cdots, v_{a_{\frac{\varphi(k)}{2}}(k-1) \pmod k+1}, v_1, v_3, \cdots, v_{k-1})$, *where* G *is a cycle of length is* $ki + \frac{k}{2}$ *for* $1 \le i \le \frac{\varphi(k)}{2}$.

We now present an algorithm for constructing (optimal) binary trees derived from super-simple MCAs. A binary tree is a rooted tree in which each internal node has at most two children, commonly referred to as the left and right child. The detailed construction procedure is outlined in Algorithm 1. The first step of the algorithm is to select one of the last three factors from the given super-simple MCA as the root of the binary tree, and assign the remaining two factors as its left and right children, respectively. Subsequently, the algorithm recursively constructs the left and right subtrees. When constructing the subtree for a given node, the current node and all its neighbors (i.e., nodes connected to it in the interaction graph) are removed from the set of available factors. If the remaining set is non-empty, one factor is selected as the left child, and the procedure recurses. If the set is empty, the branch terminates. The right child is constructed in a similar, symmetric manner. This recursive process is applied to all newly added nodes, and the algorithm terminates when no further valid children can be assigned to any expandable node. The resulting binary tree is thus derived from the super-simple MCA and conforms to the interaction structure of the factors.

The time complexity of Algorithm 1 is dominated by the recursive traversal of the factor set and the repeated adjacency checks in the interaction graph G. Assuming that adjacency queries in G are implemented using an adjacency matrix or hash-based structure (so that each query takes $O(1)$ time), the algorithm examines at most k nodes, and for each node, it may scan up to k remaining factors to determine valid children. Hence, the overall worst-case time complexity is $O(k^2)$. The space complexity is $O(k)$, primarily due to the recursion depth (at most k) and the storage of intermediate factor subsets. In practice, the algorithm is highly efficient for moderate-sized systems. For typical combinatorial testing applications where the number of factors rarely exceeds 50, the recursive construction completes almost instantaneously on modern hardware. This makes Algorithm 1 well-suited for generating hierarchical interaction structures in real-world software configuration and integration testing scenarios.

To ensure deterministic behavior and reproducibility, the function `chooseOneFrom`(L) is formally defined to select the factor in L with the largest index (or, equivalently, the one assigned to the highest hierarchical level, if such a level-based ordering is used as a tie-breaking rule). Similarly, `theOtherOne`(R, x) is defined as the unique element in the set $R \setminus x$, which is well-defined under the condition $|R| = 2$, guaranteed by the structure of our recursive decomposition. A node v is considered expandable if, after removing v and all its adjacent factors in the interaction graph G from the current factor set, at least one factor remains available for assignment as a child. This criterion ensures that the recursion proceeds meaningfully beyond trivial leaf cases. From an efficiency perspective, the time complexity of Algorithm 1 is dominated by the recursive traversal of the factor set and repeated adjacency checks in G. Assuming adjacency queries are supported by an adjacency matrix or a hash-based data structure—both allowing $O(1)$ edge lookups—the algorithm visits at most k nodes,

and for each node, it may scan up to k remaining factors to identify valid children. Consequently, the worst-case time complexity is $O(k^2)$. The space complexity is $O(k)$, primarily attributed to the recursion depth (bounded by k) and the temporary storage of intermediate factor subsets. In practice, the algorithm performs efficiently. For typical combinatorial testing scenarios—where the number of factors rarely exceeds 50—the construction process completes almost instantaneously on modern hardware. These properties make Algorithm 1 particularly suitable for generating hierarchical interaction structures in real-world software configuration and integration testing pipelines.

Algorithm 1 Constructing a Binary Tree

Require: A set of factors from given super-simple $\mathrm{MCA}_{d+1}(N; k, (v_1, v_2, \ldots, v_k))$
Ensure: A binary tree representing a hierarchical structure of the factors
if the input factor set contains only one factor **then**
    Create a leaf node labeled with that factor
    return the leaf node
end if
Select a root factor r by calling `chooseOneFrom(factorSet)`
Let $L \leftarrow \text{factorSet} \setminus \{r\}$
if r has no neighbor in L (i.e., no $f \in L$ satisfies $\{r, f\} \in E$) **then**
    Create a leaf node labeled r
    return this node
end if
Choose a left child factor $f_{\text{left}} \in L$ such that $\{r, f_{\text{left}}\} \in E$
Let $f_{\text{right}} \leftarrow$ `theOtherOne`(L, f_{left})
Recursively construct the left subtree using the factors associated with f_{left}
Recursively construct the right subtree using the factors associated with f_{right}
Create an internal node labeled r, with the left and right subtrees as its children
return this internal node

We now provide an example to illustrate the algorithm. Let A be a super-simple $\mathrm{MCA}_{d+1}(N; 5, (v_1, v_2, v_3, v_4, v_5))$, where $v_1 \leq v_2 \leq \cdots \leq v_5$. Following the algorithm, we construct a binary tree using the vertices V_3, V_4, and V_5. Suppose V_5 is chosen as the root, with V_3 as its left child and V_4 as its right child. For the left child V_3, whose neighbor in the tree is V_5, the remaining available vertices are V_1, V_2, and V_4. We may assign V_1 and V_2 as the left and right children of V_3, respectively. Similarly, for the right child V_4, whose neighbor is also V_5, the available vertices are V_1, V_2, and V_3, from which V_1 and V_2 are assigned as the left and right children of V_4, respectively. Next, consider the left child V_1 of V_3. Its neighbors in the tree are V_3 and V_4, leaving V_2 and V_5 as available vertices, which are assigned as the left and right children of V_1. In contrast, the right child V_2 of V_3 has no remaining vertices for further expansion. For the right child V_2 of V_3, we may assign V_5 as its left child, leaving the right child unassigned. A similar situation occurs for the children of V_4, where neither the left child V_1 nor the right child V_2 can be further extended. Figure 3 illustrates the complete process of constructing the binary tree according to this procedure.

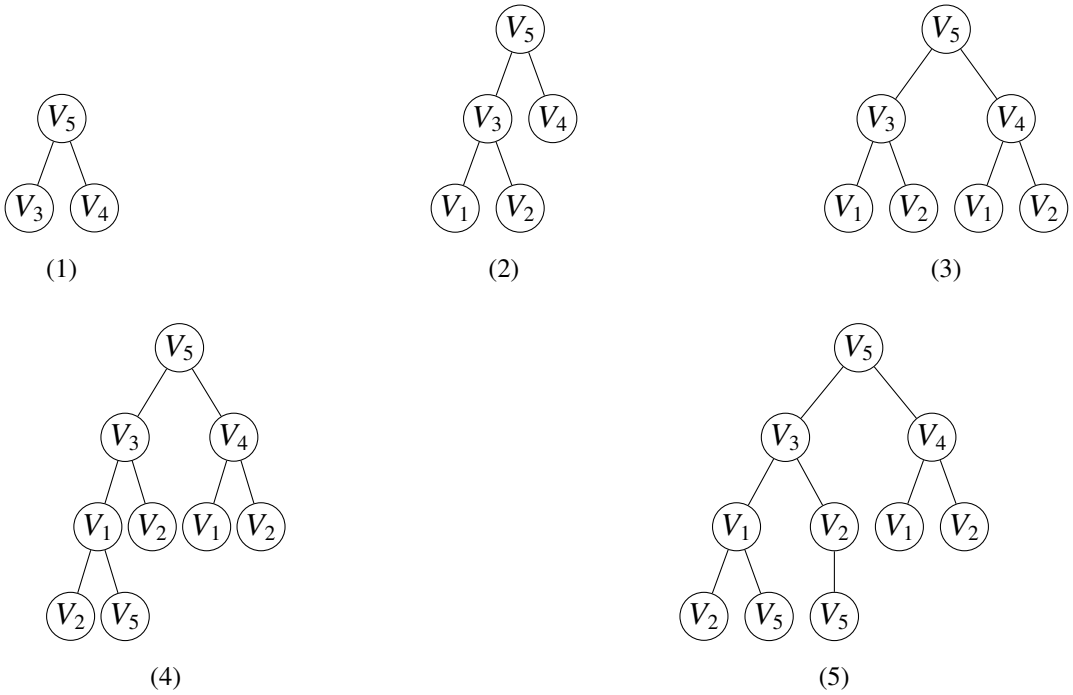

Figure 3. The process of constructing a binary tree.

4.2. Existence of optimal MDAs on paths, cycles, and binary trees

In this subsection, we construct several families of optimal MDAs over paths, cycles, and binary trees, based on the construction framework presented in Subsection 4.1. As a preliminary step, we first review several known results on super-simple MCAs, which serve as the fundamental building blocks for the constructions that follow. We begin with the existence result of optimal super-simple MCAs of strength 2 with $k = 3$, as established in [24].

Lemma 4.6 ([24])**.** *Let $2 \le v_1 \le v_2 \le v_3$. Then an optimal super-simple $MCA_\lambda(\lambda v_2 v_3; 2, 3, (v_1, v_2, v_3))$ exists if and only if $\lambda \le v_1$.*

The next result extends the above construction to $k = 4$, providing several important configurations [25].

Lemma 4.7. *Let v, λ, m, and u be positive integers with $\lambda \ge 2$. Then the following optimal super-simple MCAs exist:*

(i) *$MCA_\lambda(\lambda v^2; 2, 4, \lambda^1 v^3)$ for any $\lambda \le v$;*
(ii) *$MCA_\lambda(\lambda(vm)^2; 2, 4, v(vm)^3)$ for any positive integer $m \ge 1$;*
(iii) *$MCA_2(2v^2; 2, 4, u^1 v^3)$ for any $2 \le u \le v$.*

Lemma 4.8. *Let $v \ge 4$ be an integer. If $v \not\equiv 2 \pmod 4$, then an optimal super-simple $MCA_\lambda(\lambda v^2; 2, 5, u^1 v^4)$ exists for $\lambda \le u \le \left\lfloor \frac{v}{\lambda} \right\rfloor \cdot \lambda$.*

The following lemma provides optimal super-simple MCAs where the number of factors is a prime power or one more than a prime power, leveraging finite field properties.

Lemma 4.9 ([25]). *Let $q > 3$ be a prime power. Then an optimal super-simple $MCA_\lambda(\lambda q^4; 2, q, q^1(q^2)^{q-1})$ exists for any positive integer $\lambda \le q$. Moreover, if $q \ge 4$ is a power of 2, then an optimal super-simple $MCA_\lambda(\lambda q^4; 2, q+1, q^1(q^2)^q)$ also exists.*

We now generalize the above results to arbitrary v, provided v is the product of distinct prime powers.

Lemma 4.10 ([25]). *Let $v = q_1 q_2 \cdots q_s$, where each q_i is a distinct prime power and $k = \min\{q_i : 1 \le i \le s\}$. If each $q_i > 3$ and $\lambda \le v$, then an optimal super-simple $MCA_\lambda(\lambda v^4; 2, k, v^1(v^2)^{k-1})$ exists.*

These foundational results provide a rich set of super-simple MCAs with varying parameters. In the following, we leverage them to construct optimal MDAs on paths, cycles, and binary trees.

4.2.1. Optimal MDAs on paths

We first focus on constructing optimal MDAs on paths, one of the most fundamental graph structures. A path consists of a linear sequence of vertices connected by consecutive edges, whose simplicity and regularity make it particularly suitable for sequential interaction testing in software systems. By exploiting the properties of paths and the construction techniques derived from super-simple MCAs, we present a series of optimal MDAs.

Theorem 4.11. *Let $2 \le v_1 \le v_2 \le v_3$. Then for any $\lambda \le v_1$, there exists an optimal $MDA(\lambda v_2 v_3; \lambda - 1, G, (v_1, v_2, v_3, v_1))$, where G is a path with vertex sequence $V_1 - V_2 - V_3 - V_1$.*

Proof. By Lemma 4.6, Construction 4.2, and Theorem 3.3, the existence of the desired MDAs on paths follows directly.

We next extend to MDAs involving four main factors over paths of length six, allowing for more complex configurations.

Theorem 4.12. *Let λ, v, u, and m be positive integers satisfying one of the following:*

(i) $\lambda \le v$, $u = \lambda$, $m = 1$;
(ii) $2 \le \lambda \le v$, $u = v$, $m \ge 1$;
(iii) $\lambda = 2$, $2 \le u \le v$, $m = 1$.

Then there exists an optimal $MDA(N; \lambda - 1, G, (u, vm, vm, vm, u, vm))$ with $N = \lambda(vm)^2$, where G is a path of length 6.

Proof. In each case, the corresponding super-simple MCA is obtained by Lemma 4.7. Denote the array as $A = (A_0, A_1, A_2, A_3)$ and construct a new array $A' = (A_0, A_1, A_2, A_3, A_0, A_2)$. This sequence aligns with the path structure $V_1 - V_2 - V_3 - V_4 - V_1 - V_3$, ensuring optimal coverage.

For paths with eleven vertices, we have the following result.

Theorem 4.13. *Let $v \ge 4$ be an integer such that $v \not\equiv 2 \pmod 4$. Then for any $\lambda \le u \le \left\lfloor \frac{v}{\lambda} \right\rfloor \cdot \lambda$, there exists an optimal $MDA(\lambda v^2; \lambda - 1, G, (u, v, v, v, v, u, v, v, v, v, u))$, where G is a path of length 11.*

Proof. This follows directly from Lemma 4.8 and Construction 4.2.

Using Lemma 4.10, we can also construct MDAs for a prime power number of levels.

Theorem 4.14. *Let $q > 3$ be a prime power. Then:*

1) *An optimal MDA*$(\lambda q^4; \lambda-1, G, (q, q^2, \cdots, q^2, q, q^2, \ldots, q^2, \cdots, q, q^2, \cdots, q^2, q))$ *exists for any* $\lambda \leq q$, *where* G *is a path of length* $\frac{q^2-q+2}{2}$.
2) *Moreover, if* $q \geq 4$ *is a power of* 2, *then an optimal MDA*$(\lambda q^4; \lambda-1, G, (q, q^2, \cdots, q^2, q, q^2, \cdots, q^2, \cdots, q, q^2, \cdots, q^2, q))$ *also exists, where* G *is a path of length* $(q+1) \cdot \frac{\varphi(q+1)}{2} + 1$.

Finally, when v is a product of several prime powers, the smallest prime power factor governs the size and structure of the resulting MDA.

Theorem 4.15. *Let* $v = q_1 q_2 \cdots q_s$ *with each* $q_i > 3$ *a distinct prime power, and* $k = min\{q_i : 1 \leq i \leq s\}$. *Then an optimal MDA*$(\lambda v^4; \lambda-1, G, (v, v^2, \cdots, v^2, v, \cdots, v^2, \cdots, v, v^2, \cdots, v^2, v))$ *exists where:*

1) *If* $k \geq 3$ *is odd, then* G *is a path of length* $k \cdot \frac{\varphi(k)}{2} + 1$;
2) *If* $k \geq 6$ *is even, then* G *is also a path of length* $k \cdot \frac{\varphi(k)}{2} + \frac{k}{2} + 1$.

Proof. The result follows directly from Lemma 4.10 and Constructions 4.2 and 4.3, where the smallest prime power factor k determines the path length.

4.2.2. Optimal MDAs on cycles

We now turn to optimal MDAs over cycles. Unlike paths, cycles impose stricter structural constraints due to their closed-loop topology. In particular, for a small number of interacting factors (e.g., $k = 3, 4$), super-simple MCAs cannot, in general, be extended to additional factors without sacrificing optimality.

Theorem 4.16. *Let* $2 \leq v_1 \leq v_2 \leq v_3$, *and let* G *be a cycle with vertex sequence* $V_1 - V_2 - V_3 - V_1$. *Then an optimal MDA*$(\lambda v_2 v_3; \lambda-1, G, (v_1, v_2, v_3))$ *exists for any* $\lambda \leq v_1$.

The following theorem incorporates four factors arranged in a 4-cycle.

Theorem 4.17. *Let* λ, v, u, *and* m *be positive integers with* $2 \leq \lambda \leq v$, *and let* G *be a* 4*-vertex cycle. Then the following MDAs exist:*

1) *An optimal MDA*$(\lambda v^2; \lambda-1, G, (\lambda, v, v, v))$ *exists for* $\lambda \leq v$;
2) *An optimal MDA*$(\lambda (vm)^2; \lambda-1, G, (v, vm, vm, vm))$ *exists;*
3) *An optimal MDA*$(2v^2; 1, G, (u, v, v, v))$ *exists, where* $2 \leq u \leq v$.

We then generalize to larger cycles by replicating basic construction blocks. The following result provides explicit constructions for cycles of lengths 5 and 10.

Theorem 4.18. *Let* $v \geq 4$ *be an integer such that* $v \not\equiv 2 \pmod 4$. *Then for any* $\lambda \leq u \leq \left\lfloor \frac{v}{\lambda} \right\rfloor \cdot \lambda$, *there exist:*

1) *An optimal MDA*$(\lambda v^2; \lambda-1, G, (u, v, v, v, v))$, *where* G *is a* 5*-cycle;*
2) *An optimal MDA*$(\lambda v^2; \lambda-1, G, (u, v, v, v, v, u, v, v, v, v))$, *where* G *is a* 10*-cycle.*

Proof. These results follow from Lemma 4.8 together with Construction 4.4.

Theorem 4.19. *Let* $q > 3$ *be a prime power. Then:*

1) *An optimal MDA*$(\lambda q^4; \lambda-1, G, (q, q^2, \ldots, q^2, q, q^2, \ldots, q^2, \cdots))$ *exists for any* $\lambda \leq q$, *where* G *is a cycle of length* $q \cdot i$ *for* $1 \leq i \leq \frac{q-1}{2}$.

2) *Moreover, if* $q \geq 4$ *is a power of* 2*, then an optimal* $MDA(\lambda q^4; \lambda - 1, G, (q, q^2, \ldots, q^2, q, q^2, \ldots, q^2, \cdots))$ *also exists, where* G *is a path of length* $(q+1) \cdot i$ *for* $1 \leq i \leq \frac{\varphi(q+1)}{2}$.

Proof. The results are based on Lemma 4.10 using Constructions 4.4.

Finally, we generalize to the setting where the number of levels v is a product of distinct prime powers. In this context, the smallest prime power factor plays a pivotal role in determining the structure and size of the optimal MDAs over cycles.

Theorem 4.20. *Let* $v = q_1 q_2 \cdots q_s$ *be the standard factorization of* v *into distinct prime powers, and let* $k = min\{q_i : 1 \leq i \leq s\}$. *Suppose that each* $q_i > 3$ *and* $\lambda \leq v$. *Then there exists an optimal* $MDA(\lambda v^4; \lambda - 1, G, (v, v^2, \ldots, v^2, v, \cdots, v^2, \cdots,))$, *where the cycle structure* G *is determined as follows:*

1) *If* $k \geq 3$ *is odd, then* G *is a cycle of length* $k \cdot i$ *for* $1 \leq i \leq \frac{\varphi(k)}{2}$;
2) *If* $k \geq 6$ *is even, then* G *is also a cycle of length* $k \cdot i + \frac{k}{2}$ *for* $1 \leq i \leq \frac{\varphi(k)}{2}$.

Proof. The result follows directly from Lemma 4.10 and Constructions 4.4 and 4.5, where the smallest prime power factor k determines the cycle length.

For binary trees, numerous results can be derived by applying Algorithm 1 in conjunction with Lemmas 4.6–4.10. Details are omitted for brevity.

5. Concluding remarks

In this paper, we introduced the concept of MDAs on graphs, extending classical detecting array theory to accommodate heterogeneous factors and the graph-structured interaction constraints. This new model is particularly relevant for testing component-based systems where parameter interactions are governed by a dependency graph—reflecting real-world scenarios such as network configurations, hierarchical components, and modular software architectures. We established theoretical bounds on the size of MDAs on graphs and provided combinatorial characterizations for their existence. Moreover, we presented explicit constructions of optimal MDAs on paths, cycles, and binary trees by leveraging super-simple mixed covering arrays (MCAs). These results demonstrate that well-structured MCAs can serve as efficient and practical building blocks for generating cost-effective, fault-sensitive test suites under graph-based interaction constraints. Overall, this study provides both a rigorous theoretical foundation and constructive methodologies, highlighting the novelty and practical relevance of MDAs on graphs in the design of systematic testing strategies for heterogeneous and graph-constrained systems. Looking ahead, a particularly promising direction for future research is to develop efficient algorithmic techniques for constructing mixed-level detecting arrays (MDAs) on specific classes of graphs or for super-simple mixed covering arrays, especially when additional structural constraints or optimality requirements are imposed. Such algorithmic advancements would not only improve the practical feasibility of generating MDAs but also enable their application to large-scale and complex systems where computational efficiency is critical. Another important avenue for exploration is to generalize the current framework to encompass a broader spectrum of graph topologies, including grids, scale-free networks, unbalanced networks, tree-symmetric networks [26, 27] , and orthogonal graph squares for the disjoint union of paths [28]. This extension would allow researchers to systematically investigate how the underlying graph

structure affects both the existence conditions and the efficiency of MDAs, thereby providing deeper insights into the interplay between combinatorial design and network topology in heterogeneous testing scenarios.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported by NSFC No. 11301342 and Natural Science Foundation of Shanghai No. 17ZR1419900.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. D. R. Kuhn, R. N. Kacker, Y. Lei, *Introduction to Combinatorial Testing*, Chapman and Hall/CRC, Boca Raton, FL, USA, 2013.
2. C. Nie, H. Leung, A survey of combinatorial testing, *ACM Comput. Surv.*, **43** (2011), 1–29. https://doi.org/10.1145/1883612.1883618
3. C. J. Colbourn, Strength two covering arrays: Existence tables and projection, *Discrete Math.*, **308** (2008), 772–786. https://doi.org/10.1016/j.disc.2007.07.050
4. J. T. Jimenez, I. I. Marquez, Covering arrays of strength three from extended permutation vectors, *Des. Codes Cryptogr.*, **86** (2018), 2629–2643. https://doi.org/10.1007/s10623-018-0465-6
5. J. T. Jimenez, I. I. Marquez, Improved covering arrays using covering perfect hash families with groups of restricted entries, *Appl. Math. Comput.*, **369** (2020), 124826. https://doi.org/10.1016/j.amc.2019.124826
6. L. Ji, J. Yin, Constructions of new orthogonal arrays and covering arrays of strength three, *J. Comb. Theory Ser. A*, **117** (2010), 236–247. https://doi.org/10.1016/j.jcta.2009.06.002
7. L. G. Hernandez, New bounds for mixed covering arrays in t-way testing with uniform strength, *Inf. Softw. Technol.*, **59** (2024), 17–32. https://doi.org/10.1016/j.infsof.2014.10.009
8. I. I. Marquez, J. T. Jimenez, B. A. Juárez, H. A. George, A greedy-metaheuristic 3-stage approach to construct covering arrays, *Inf. Sci.*, **460-461** (2018), 172–189. https://doi.org/10.1016/j.ins.2018.05.047
9. K. Sarkar, C. J. Colbourn, Two-stage algorithms for covering array construction, *J. Comb. Des.*, **27** (2019), 475–505. https://doi.org/10.1002/jcd.21657
10. K. Shokri, L. Moura, New families of strength 3 covering arrays using linear feedback shift register sequences, *J. Comb. Des.*, **33** (2025), 156–171. https://doi.org/10.1002/jcd.21963

11. S. Raaphorst, L. Moura, B. Stevens, A construction for strength-3 covering arrays from linear feedback shift register sequences, *Des. Codes Cryptogr.*, **73** (2014), 949–968. https://doi.org/10.1007/s10623-013-9835-2
12. G. Tzanakis, L. Moura, D. Panario, B. Stevens, Covering arrays from m-sequences and character sums, *Des. Codes Cryptogr.*, **85** (2017), 437–456. https://doi.org/10.1007/s10623-016-0316-2
13. C. J. Colbourn, C. Shi, C. M. Wang, J. Yan, Mixed covering arrays of strength three with few factors, *J. Stat. Plann. Inference*, **141** (2011), 3640–3647. https://doi.org/10.1016/j.jspi.2011.05.018
14. L. Moura, J. Stardom, B. Stevens, A. Williams, Covering arrays with mixed alphabet sizes, *J. Combin. Des.*, **11** (2003), 413–432. https://doi.org/10.1002/jcd.10059
15. K. Meagher, B. Stevens, Covering arrays on graphs, *J. Comb. Theory Ser. B*, **95** (2005), 134–151. https://doi.org/10.1016/j.jctb.2005.03.005
16. K. Meagher, L. Moura, L. Zekaoui, Mixed covering arrays on graphs, *J. Comb. Des.*, **15** (2007), 393–404. https://doi.org/10.1002/jcd.20149
17. P. Danziger, E. Mendelsohn, L. Moura, B. Stevens, Covering arrays avoiding forbidden edges, *Theor. Comput. Sci.*, **410** (2009), 5403–5414. https://doi.org/10.1016/j.tcs.2009.07.057
18. C. J. Colbourn, D. W. McClary, Locating and detecting arrays for interaction faults, *J. Comb. Optim.*, **15** (2008), 17–48. https://doi.org/10.1007/s10878-007-9082-4
19. C. M. Wang, C. Shi, T. Tsuchiya, Q. R. Zhang, Detecting arrays on graphs, *Electron. Res. Arch.*, **33** (2025), 3328–3347. https://doi.org/10.3934/era.2025147
20. C. Shi, L. Jiang, A. Y. Tao, Consecutive detecting arrays for interaction faults, *Graphs Combin.*, **36** (2020), 1203–1218. https://doi.org/10.1007/s00373-020-02176-7
21. C. Shi, A. Tao, Consecutive detecting arrays from m-sequence, *IAENG Int. J. Appl. Math.*, **50** (2020), 80–86.
22. Y. Tang, J. X. Yin, Detecting arrays and their optimality, *Acta Math. Sin. Engl. Ser.*, **27** (2011), 2309–2318. https://doi.org/10.1007/s10114-011-0184-7
23. C. Shi, Y. Tang, J. X. Yin, The equivalence between optimal detecting arrays and super-simple OAs, *Des. Codes Cryptogr.*, **62** (2012), 131–142. https://doi.org/10.1007/s10623-011-9498-9
24. C. Shi, Y. Tang, J. X. Yin, Optimum mixed level detecting arrays, *Ann. Stat.*, **42** (2014), 1546–1563. https://doi.org/10.1214/14-AOS1228
25. C. Shi, Optimum Super-simple mixed covering arrays of type a^1b^{k-1}, *Acta Math. Sin. Engl. Ser.*, **33** (2017), 153–164. https://doi.org/10.1002/jcd.21657
26. J. B. Liu, X. Wang, L. Hua, J. D. Cao, L. P. Chen, The coherence and robustness analysis for a family of unbalanced networks, *IEEE Trans. Signal Inf. Process. Netw.*, **11** (2025), 378–387. https://doi.org/10.1109/tsipn.2025.3555164
27. J. B. Liu, L. Guan, J. D. Cao, Property analysis and coherence dynamics for tree-symmetric networks with noise disturbance, *J. Complex Netw.*, **12** (2024), 475–505. https://doi.org/10.1093/comnet/cnae029

28. R. El-Shanawany, A. El-Mesady, S. M. Shaaban, Mutually orthogonal graph squares for disjoint union of paths, *Appl. Math. Sci.*, **12** (2018), 303–310. https://doi.org/10.1155/2022/9308708