Optimal Combinatorial Testing with Constraints: The Balancing Act

Thiago Serra[1], Changkun Guan[2], Hunter Gehman[3], Sumit Dhar[3], Mikey Ferguson[4], John Hooker[5], and Marcel Schoppers

[1] University of Iowa, Iowa City IA, United States
thiago-serra@uiowa.edu
[2] Georgia Institute of Technology, Atlanta GA, United States
[3] Bucknell University, Lewisburg PA, United States
[4] University of Tulsa, Tulsa OK, United States
[5] Carnegie Mellon University, Pittsburgh PA, United States

Abstract. Imagine that you are in front of a cockpit with several on–off buttons. If you were to thoroughly test it, you would need to try a prohibitive number of configurations. But since most bugs in practice can be isolated to interactions among few components, having tests that cover every possible pairwise configuration is a good start. However, this is a problem that goes from easy to NP–hard as soon as some pairwise configurations are forbidden. In this paper, we revisit unconstrained combinatorial testing with pairwise coverage on binary parameters and contrast it with the constrained case, showing and conjecturing properties that either are upheld or change from one to the other. In particular, we discuss the extent to which it remains a good idea—and sometimes indeed optimal—to have every button almost as many times on as off to minimize testing. We propose the first exact algorithm based on integer programming and a faster heuristic that often produces optimal solutions, both outperforming or competitive with their baselines.

Keywords: Combinatorial testing · Graphs · Integer programming.

1 Introduction

Many safety tests in software and other types of systems involve the identification of faults due to the interaction among components. In the simplest setting, we may regard each component i as an on–off switch, say as a *parameter* X_i such that $X_i = 1$ when that component is on and $X_i = 0$ otherwise. For every test of the system, we need to assign a value for each of the system parameters. We would like to know if something unexpected, which is what we denote as a *fault*, would occur due to the combination of values for such parameters. For two distinct components i and j, we would like to have tests in which both components are on ($X_i = 1$ and $X_j = 1$), only the first component is on ($X_i = 1$ and $X_j = 0$), only the second component is on ($X_i = 0$ and $X_j = 1$), or both components are off ($X_i = 0$ and $X_j = 0$); provided that each of those cases is feasible. We would like to minimize the number of tests to cover such cases.

While testing every configuration of the entire system would be impractical as the number of components grow, having enough tests to cover every configuration on subsets of limited size does not grow as fast. In fact, most faults can be attributed to interactions among of small number of components [47,30,29,31,28]. If a test leads to a fault, then additional tests can help us isolate which specific subset interaction caused that fault [54]. This form of analysis, *combinatorial interaction testing*, has been applied to software testing [30], circuit verification [45], memory correction [16], gene regulation [44], materials development [6], aircraft development [22], and machine learning verification [38]. We refer the reader to references [12,40,22,38] for more information on applications.

In this work, we focus on the *covering array problem*, which concerns with obtaining sufficient tests to cover every configuration on subsets of limited size, hence ensuring that any fault caused by a subset of such size could be detected. Although this topic has extensive literature discussed in multiple surveys [12,49,20,42], comparatively few references focused on obtaining provably optimal solutions [27,24,14,39,36,25,51] and the only approach to the constrained case is by enumeration [14]. The presence of pairwise constraints makes combinatorial testing NP-hard [39], but constraints preventing specific configurations are often dealt with in an ad-hoc manner by altering the solutions produced for the unconstrained case [50]. The typical approach is by popular families of heuristic algorithms such as IPO [35,33,34,15,53,55], AETG [8,7,11], orthogonal arrays [48], PICT[13], DDA [4,3], as well as maximum satisfiability [1] and metaheuristics [9,23,10,19,37]. Meanwhile, Integer Programming (IP) has only been applied to the unconstrained case [36,25] or as part of heuristics [17,18].

To the best of our knowledge, no prior work has described the properties of optimal solutions for the covering array problem beyond the unconstrained case. Even then, we believe that a new perspective on the unconstrained case helps understanding what happens to the structure of the optimal solutions when constraints are imposed. Hence, our contributions are the following:

(i) we revisit the unconstrained binary pairwise covering array problem by focusing on the incidence of *balanced columns* in optimal solutions, each implying that a parameter is tested as often with zero and one (Section 3);
(ii) we establish how the structure of optimal solutions change, or not, when constraints prevent assignments to pairs of parameters, which define what we denote as *constrained pairwise covering problems* (Section 4); and
(iii) we present and evaluate IP models leveraging our findings in exact as well as heuristic approaches to the constrained case (Sections 5 and 6).

2 Background and Notation

Consider a system with parameters $X_1, X_2, \ldots, X_k$ and that each parameter is binary. We assign values to these parameters to verify the correct functioning of the system. Each such assignment is denoted as a *test*. A collection of tests which satisfies a desired verification property is denoted as a *covering array*. In terms of what would be desired from such tests, trying every possible test

(2^k in total) would be impossible for sufficiently large k. However, it is technically feasible, and in fact effective to try all configurations on small subsets of parameters. In the most common setting, *pairwise covering*, those tests should contain every possible assignment for each of the $\binom{k}{2}$ pairs of parameters. For two parameters X_i and X_j, for example, we should have each pairwise assignment among $(X_i, X_j) = (0,0), (0,1), (1,0)$, and $(1,1)$ in at least one of the tests. The matrix $\mathbf{M}^{(1)}$ below, with columns denoting parameters and rows denoting tests, represents an optimal pairwise covering array for $k = 10$ unconstrained binary parameters, in the sense that it has the minimum number $N = 6$ of tests:

$$\mathbf{M}^{(1)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

In other words, there are 2^{10} distinct assignments on 10 binary parameters, but with 6 of those assignments we covered every pairwise interaction. For example, the test $(X_1, X_2, \ldots, X_{10}) = (0, 0, \ldots, 0)$ is not part of $\mathbf{M}^{(1)}$, but there are rows of zeros in every submatrix on two columns, i.e., each pairwise assignment $(X_1, X_2) = (0,0)$, $(X_1, X_3) = (0,0), \ldots, (X_9, X_{10}) = (0,0)$ occurs at least once. Similarly, each such submatrix also covers $(X_i, X_j) = (0,1), (1,0)$, and $(1,1)$.

In the general decision version of the covering array problem, we want to determine if there are N tests on k parameters, each parameter having v values, that define a covering array of *strength* t. If so, then every possible assignment for each tuple of t out of the k parameters occurs in some test. Following the conventions in the literature, let $\mathrm{CA}(N; t, k, v)$ be the set of $N \times k$ matrices, or covering arrays, in which every submatrix on t columns includes all $t-$tuples over $\{0, \ldots, v-1\}^t$ at least once; and let $\mathrm{CAN}(t, k, v)$ be the minimum N such that $\mathrm{CA}(N; t, k, v) \neq \emptyset$. In this paper, we also introduce some additional notation: we use $\mathbf{M}_i$ for the i-th column of $\mathbf{M} \in \mathrm{CA}(N; t, k, v)$, i.e., the values assigned to X_i; and we define $\mathrm{CA}^*(t, k, v)$ as the set $\mathrm{CA}(N; t, k, v)$ in which $N = \mathrm{CAN}(t, k, v)$.

The most commonly studied case consists of $v = 2$ (binary parameters) and $t = 2$ (pairwise covering), for which the optimal number of tests is known: $\mathrm{CAN}(2, k, 2) = N$, where N is the smallest integer such that $k \leq \binom{N-1}{\lceil N/2 \rceil}$ [43,26,27]. Hence, $\mathrm{k}^{\max}(N) := \binom{N-1}{\lceil N/2 \rceil}$ is the largest number of binary parameters for which N tests may ensure pairwise covering. Bounds on the number of tests N for other families of cases involving t, k, and v can be found in [12] and [32].

For $N = 6$, $\mathrm{k}^{\max}(N) = 10$ as in matrix $\mathbf{M}^{(1)}$. In fact, we have obtained $\mathbf{M}^{(1)}$ with a commonly used construction technique by Kleitman and Spencer [27], which consists of having a row in which all values are one and the remaining rows are such that each column is distinct and has $\lfloor \frac{N}{2} \rfloor - 1$ other elements having value one. More generally, we discuss along the paper that the presence of a uniform row, or the equivalent of such a row, as well as having columns with a similar

number of zeros and ones are important for characterizing optimal solutions. Those are still relevant when pairwise constraints apply to the system.

In this paper, we consider pairwise covering problems in which some assignments are forbidden. For example, we may have $(X_i, X_j) \neq (0, 0)$ for a specific pair of parameters, in which case we must cover $(X_i, X_j) = (0, 1), (1, 0)$, and $(1, 1)$ across the tests, but $(X_i, X_j) = (0, 0)$ must never occur. We note that having more than one forbidden assignment for the same pair of parameters actually simplifies the covering array problem in one way or another. When there are two forbidden assignments, we have one of the following cases:

$$(X_i, X_j) \neq (\alpha, \beta) \wedge \begin{cases} (X_i, X_j) \neq (\alpha, 1-\beta) & \rightarrow \quad X_i = 1 - \alpha. \\ (X_i, X_j) \neq (1-\alpha, \beta) & \rightarrow \quad X_j = 1 - \beta. \\ (X_i, X_j) \neq (1-\alpha, 1-\beta) & \rightarrow \begin{cases} X_i = X_j & \text{if } \alpha \neq \beta. \\ X_i = 1 - X_j & \text{if } \alpha = \beta. \end{cases} \end{cases}$$

Therefore, having two forbidden assignments for a given pair of parameters implies that either one of the parameters is fixed (first two deductions), or that the value for one parameter is dependent on the other (last two deductions); and in both cases we can reduce the problem by removing one of those parameters. With three forbidden assignments, say all but $(X_i, X_j) = (\alpha, \beta)$ forbidden, then $X_i = \alpha$ and $X_j = \beta$. Finally, and obviously, the covering array problem is infeasible with four forbidden assignments on the same pair of parameters. Hence, without loss of generality, we assume the following simplifying conditions:

Assumption 1 *No parameter has a fixed value.*

Assumption 2 *There is at most one constraint on every pair of parameters.*

Assumption 3 *There are no parameters with an implicitly fixed value, nor a pair of parameters with an implicit constraint other than those known in advance.*

In practice, we can remove parameters with fixed or implied values in preprocessing and then reintroduce them with the corresponding values in postprocessing. Likewise, we can generate all implied constraints with the deductive steps as outlined in [52,53]. For example, $(X_1, X_2) \neq (1, 0) \wedge (X_2, X_3) \neq (1, 0) \rightarrow (X_1, X_3) \neq (1, 0)$. Hence, Assumptions 1 to 3 are also not restricting the scope of our work.

We will avoid an implicit dependence on the assumptions above in our theoretical results in Section 4 by using the following definition:

Definition 1 (Reduced Constrained Pairwise Covering Problem). *Let a constrained pairwise covering problem be* reduced *when it has (i) no fixed parameter values, and (ii) at most one pairwise constraint per pair of parameters; be those explicitly defined or implied by the other constraints.*

Effectively, Definition 1 characterizes a constrained pairwise covering problem to which the preprocessing previously described has already been applied.

Although there are always constraints in the pairwise covering array problem, since every pair of assignments should be either covered by a test or forbidden in all tests, we refer to this problem as being *unconstrained* if no specific interactions are forbidden. Our study starts by revisiting the unconstrained case.

3 Revisiting the Unconstrained Case

There are some properties that characterize many—and, in some cases, all—optimal solutions for the unconstrained covering array problem. By first understanding and reframing those in Section 3, we will later generalize such properties to circumvent forbidden interactions in a systematic way in Sections 4 and 5.

We continue using k and N to denote the number of parameters and tests.

3.1 Four Simple Facts About Unconstrained Covering Arrays

The propositions below are possibly already known in one form or another, but they are certainly useful. We start with necessary conditions for a matrix to be a covering array. Namely, all the columns should be unique and not the complement of one another. We formalize those conditions as follows:

Proposition 1 (Uniqueness of Columns). *Let $k, N \in \mathbb{Z}_+$, $N \geq \mathrm{CAN}(2, k, 2)$. Any two columns $\mathbf{M}_i, \mathbf{M}_j \in \{0,1\}^N$, $i \neq j$, of a matrix $\mathbf{M} \in \mathrm{CA}(N; 2, k, 2)$ are such that $\mathbf{M}_i \neq \mathbf{M}_j$.*

Proof. If $\mathbf{M}_i = \mathbf{M}_j$, then $\mathbf{M}$ would not cover the assignments $(X_i, X_j) = (0, 1)$ and $(X_i, X_j) = (1, 0)$. Hence, no two columns are the same. □

Proposition 2 (Exclusion of Complement). *Let $k, N \in \mathbb{Z}_+$, $N \geq \mathrm{CAN}(2, k, 2)$. Any two columns $\mathbf{M}_i, \mathbf{M}_j \in \{0,1\}^N$, $i \neq j$, of a matrix $\mathbf{M} \in \mathrm{CA}(N; 2, k, 2)$ are such that $\mathbf{M}_i \neq \mathbf{1} - \mathbf{M}_j$, $\mathbf{1} = \{1\}^N$.*

Proof. If $\mathbf{M}_i = \mathbf{1} - \mathbf{M}_j$, then $\mathbf{M}$ would not cover the assignments $(X_i, X_j) = (0, 0)$ and $(X_i, X_j) = (1, 1)$. Hence, no two columns are complements. □

Moreover, we can obtain alternate covering arrays by taking the complement of columns or swapping them. These properties will be useful later:

Proposition 3 (Covering by Column Complement). *Let $k, N \in \mathbb{Z}_+$, $N \geq \mathrm{CAN}(2, k, 2)$. For any $\mathbf{M} \in \mathrm{CA}(N; 2, k, 2)$ and column $\mathbf{M}_i$, let matrix $\mathbf{M}'$ be such that $\mathbf{M}'_i = \mathbf{1} - \mathbf{M}_i$, $\mathbf{1} = \{1\}^N$, and $\mathbf{M}'_j = \mathbf{M}_j$ $\forall j \neq i$. Then $\mathbf{M}' \in \mathrm{CA}(N; 2, k, 2)$.*

Proof. Since $\mathbf{M}$ and $\mathbf{M}'$ are identical except for column i, then all pairwise assignments not involving parameter X_i are covered by $\mathbf{M}'$ if they are covered by $\mathbf{M}$. For any row of $\mathbf{M}$ covering $(X_i, X_j) = (\alpha, \beta)$, the same row in $\mathbf{M}'$ covers $(X_i, X_j) = (1 - \alpha, \beta)$. Thus, if there is a row in $\mathbf{M}$ covering each assignment of (X_i, X_j), then there is another row in $\mathbf{M}'$ covering each of those assignments. Hence, taking the complement of a column yields another covering array. □

For an application of Proposition 3, consider the matrices $\mathbf{M}^{(2)}$ and $\mathbf{M}^{(3)}$:

$$\mathbf{M}^{(2)} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{M}^{(3)} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

Both of those matrices represent optimal pairwise covering arrays for $k = 5$. Similarly to $\mathbf{M}^{(1)}$, the matrix $\mathbf{M}^{(2)}$ is obtained with the construction technique by Kleitman and Spencer [27]. In turn, the matrix $\mathbf{M}^{(3)}$ is obtained based on Proposition 3 by taking the complement of the last column of $\mathbf{M}^{(2)}$.

Proposition 4 (Covering by Column Swapping). *Let $k, N \in \mathbb{Z}_+$, $N \geq \mathrm{CAN}(2, k, 2)$. For any $\mathbf{M} \in \mathrm{CA}(N; 2, k, 2)$ and columns $\mathbf{M}_{i_1}$ and $\mathbf{M}_{i_2}$, let matrix $\mathbf{M}'$ be such that $\mathbf{M}'_{i_1} = \mathbf{M}_{i_2}$, $\mathbf{M}'_{i_2} = \mathbf{M}_{i_1}$, and $\mathbf{M}'_j = \mathbf{M}_j$ $\forall j \notin i_1, i_2$. Then $\mathbf{M}' \in \mathrm{CA}(N; 2, k, 2)$.*

Proof. Since $\mathbf{M}$ and $\mathbf{M}'$ are identical except for columns i_1 and i_2, then all pairwise assignments not involving either parameter X_{i_1} nor X_{i_2} are covered by $\mathbf{M}'$ if they are covered by $\mathbf{M}$. Every assignment $(X_{i_1}, X_{i_2}) = (\alpha, \beta)$ is covered in $\mathbf{M}'$ due to the corresponding assignment $(X_{i_1}, X_{i_2}) = (\beta, \alpha)$ being covered in $\mathbf{M}$. Finally, for $i_A \in \{i_1, i_2\}$ (i.e., one of the indices among i_1 and i_2), $i_B \in \{i_1, i_2\} \setminus \{i\}$ (i.e., the other index), and $j \notin \{i_1, i_2\}$, every assignment $(X_i, X_j) = (\alpha, \beta)$ is covered in $\mathbf{M}'$ due to assignment $(X_{\bar{i}}, X_j) = (\alpha, \beta)$ being covered in $\mathbf{M}$. Hence, swapping any two columns yields another covering array. □

With Proposition 4, the columns of an unconstrained covering array are interchangeable. Along with Proposition 3, that implies that every covering array on k parameters belongs to an equivalence class of $2^k k!$ covering arrays upon column complement and column swapping. To the extent that we can retain part of this flexibility even after considering pairwise constraints breaking such symmetry, we can simplify the problem of finding an optimal covering array. That will be particularly helpful in Section 5.

3.2 Even Covering Arrays: The Special Case

When we have an even number of rows, and thus each column can have as many zeros as ones, we often observe optimal covering arrays that reflect that. For example, matrix $\mathbf{M}^{(1)}$ has 3 zeros and 3 ones in every column. In fact, we can rely on the following sufficiency condition when the number of rows N is even:

Lemma 1 (Balanced Covers). *Let $k, N \in \mathbb{Z}_+$ with N even. Any matrix $\mathbf{M} \in \{0, 1\}^{N \times k}$ in which each column $\mathbf{M}_i$ is (a) unique ($\mathbf{M}_i \neq \mathbf{M}_j$ $\forall j \neq i$); (b) not the complement of another column ($\mathbf{M}_i \neq \mathbf{1} - \mathbf{M}_j, \mathbf{1} = \{1\}^N$ $\forall j \neq i$); and (c) balanced (has $\frac{N}{2}$ zeros and $\frac{N}{2}$ ones) is such that $\mathbf{M} \in \mathrm{CA}(N; 2, k, 2)$.*

Proof. Let us suppose, for contradiction, that there is a matrix $\mathbf{M}$ satisfying conditions (a)–(c), but such that $\mathbf{M} \notin \mathrm{CA}(N; 2, k, 2)$. Hence, there should be columns $\mathbf{M}_i$ and $\mathbf{M}_j$, $i \neq j$, and a pair of values $(\alpha, \beta) \in \{0, 1\}^2$ such that there is no row in $\mathbf{M}$ covering $(X_i, X_j) = (\alpha, \beta)$. We show that this is not possible.

If that were the case for $\alpha \neq \beta$, then column $\mathbf{M}_i$ having α in a row implies that column $\mathbf{M}_j$ has α in that row. Since columns $\mathbf{M}_i$ and $\mathbf{M}_j$ have the same number of zeros and ones due to (c), then $\mathbf{M}_i$ and $\mathbf{M}_j$ would have α in the same rows and consequently β in the same rows. Thus, $\mathbf{M}_i = \mathbf{M}_j$, contradicting (a).

If that were the case for $\alpha = \beta$, then column $\mathbf{M}_i$ having α in a row implies that column $\mathbf{M}_j$ has $1 - \alpha$ in that row. Since $\mathbf{M}_i$ and $\mathbf{M}_j$ have as many zeros as ones due to (c), then column $\mathbf{M}_i$ would have α in the same rows that column $\mathbf{M}_j$ would have $1-\alpha$ and vice versa. Thus, $\mathbf{M}_i = \mathbf{1} - \mathbf{M}_j$, contradicting (b). □

These results outline a simple algorithm that does not require the conventional use of a row of ones for constructing covering arrays for a given choice of k parameters using an even number of N tests, which is based on three rules:

1. Generate columns without repetition (Proposition 1).
2. Do not generate the complement of a generated column (Propositon 2).
3. Only generate columns with equal number of zeros and ones (Lemma 1).

In fact, this algorithm produces an optimal solution for the smallest possible N, if such an N is even, given that $\mathrm{k}^{\max}(N) = \binom{N-1}{\lceil N/2 \rceil}$ (see Section 2):

Corollary 1 (Even Balanced Optimality). *Let $k \in \mathbb{Z}_+$. If $N := \mathrm{CAN}(2, k, 2)$ is even, we obtain a matrix in $\mathrm{CA}^*(2, k, 2)$ by following rules 1, 2, and 3.*

Proof. The number of binary columns with $N/2$ ones is $\binom{N}{N/2}$. If we deduct the complement of each column, we have $\frac{1}{2}\binom{N}{N/2} = \frac{1}{2}\frac{N!}{N/2!N/2!} = \frac{1}{2}\frac{N(N-1)!}{N/2(N/2-1)!N/2!} = \frac{(N-1)!}{(N/2-1)!N/2!} = \binom{N-1}{N/2} = \mathrm{k}^{\max}(N)$ columns. □

3.3 Odd Covering Arrays: The General Case

In the case of N odd, working with the most balanced columns would amount to considering columns with $\lfloor N/2 \rfloor$ ones and $\lceil N/2 \rceil$ zeros and vice versa. For example, consider the two previous matrices: $\mathbf{M}^{(2)}$ has 2 ones and 3 zeros in every column; whereas $\mathbf{M}^{(3)}$ has the last column with 3 ones and 2 zeros.

Since to every column with $\lfloor N/2 \rfloor$ ones and $\lceil N/2 \rceil$ zeros there is a complement with $\lceil N/2 \rceil$ ones and $\lfloor N/2 \rfloor$ zeros, then by Proposition 3 we can use each of such pairs of columns interchangeably. Without loss of generality, we make the following assumption while discussing the upcoming Propositions 5 to 7 to make it easier to understand the structure of covering arrays with N odd:

Assumption 4 *For N odd,* balanced columns *those with $\lfloor N/2 \rfloor$ ones and $\lceil N/2 \rceil$ zeros.*

We start by proving that any matrix with distinct columns of $\lfloor N/2 \rfloor$ ones and $\lceil N/2 \rceil$ zeros covers $(X_i, X_j) = (0, 0)$, $(X_i, X_j) = (0, 1)$, and $(X_i, X_j) = (1, 0)$ with the following results. Please note that they also hold for N even, which is a convenient generalization for reusing these results in the constrained case:

Proposition 5 (Generalized Zero-Zero Cover). *Let $k, N \in \mathbb{Z}_+$. Any matrix $\mathbf{M} \in \{0, 1\}^{N \times k}$ in which each column (a) is unique; (b) is not the complement of another column; and (c) has at most $\lfloor N/2 \rfloor$ ones (resp. at least $\lceil N/2 \rceil$ zeros) is such that $(X_i, X_j) = (0, 0)$ is covered in least one row for any $(i, j) \in \{1, \ldots, k\}^2$.*

Proof. Each column having at most $\lfloor N/2 \rfloor$ ones implies that there are at most $2\lfloor N/2 \rfloor$ rows with ones in either $\mathbf{M}_i$ or $\mathbf{M}_j$. Since $2\lfloor N/2 \rfloor = N$ only for N even and with both columns balanced, then the conditions of Lemma 1 apply in such case. Otherwise, if $2\lfloor N/2 \rfloor < N$, then at least one row has a pair of zeros. □

Proposition 6 (Generalized Zero-One Cover). *Let $k, N \in \mathbb{Z}_+$. A matrix $\mathbf{M} \in \{0,1\}^{N \times k}$ covers $(X_i, X_j) = (0,1)$ in at least one row for any $(i,j) \in \{1,\ldots,k\}^2, i \neq j$ if, and only if, column $\mathbf{M}_j$ is not a subset of $\mathbf{M}_i$ ($\mathbf{M}_j \nsubseteq \mathbf{M}_i$).*

Proof. If $\mathbf{M}_j \nsubseteq \mathbf{M}_i$, then is at least one row in which the value assigned to X_j is greater than the value assigned to X_i, which thus covers $(X_i, X_j) = (0,1)$.

Conversely, if $\mathbf{M}$ covers $(X_i, X_j) = (0,1)$, then the rows with ones in column $\mathbf{M}_j$ are not a subset of the rows with ones in $\mathbf{M}_i$, and thus $\mathbf{M}_j \nsubseteq \mathbf{M}_i$. □

In other words, by assuming that balanced columns may have more zeros than ones, we have made zeros slightly more abundant. Conveniently, that entails that all pairwise assignments involving the value zero are covered. Hence, we are left to discuss how to cover pairs of ones, given that ones are less abundant.

On the one hand, if we adapt the third rule in Section 3.2 to handle N odd by assuming as balanced columns those with $\lfloor N/2 \rfloor$ ones, we may not always find an optimal solution for the smallest N. For example, consider again matrix $\mathbf{M}^{(2)}$, repeated below for convenience, along with another matrix $\mathbf{M}^{(4)}$:

$$\mathbf{M}^{(2)} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \; \mathbf{M}^{(4)} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

We cover 4 parameters with the leftmost matrix $\mathbf{M}^{(2)} \in \mathrm{CA}(5; 2, 4, 2)$, and thus we cover 3 parameters with any submatrix of $\mathbf{M}^{(2)}$ on three columns. We also cover 3 parameters with the rightmost matrix $\mathbf{M}^{(4)} \in \mathrm{CA}(5; 2, 3, 2)$. However, there is no column that could be added to $\mathbf{M}^{(4)}$ to cover 4 parameters. Therefore, depending on which columns we choose first, we may not be able to extend a covering array on k parameters and N tests all the way to $\mathrm{k}^{\max}(N)$ parameters.

On the other hand, based on Propositions 5 and 6, our only concern is about covering $(X_i, X_j) = (1,1)$. This can be achieved by having a row of ones, as in the construction used for $\mathbf{M}^{(1)}$ and $\mathbf{M}^{(2)}$, implying a simple sufficient result:

Proposition 7 (One-Row Sufficiency). *Let $k, N \in \mathbb{Z}_+$. Any matrix $\mathbf{M} \in \{0,1\}^{N \times k}$ in which each column is (a) unique and (b) has $\lfloor N/2 \rfloor$ ones and $\lceil N/2 \rceil$ zeros; and (c) one row has k ones is such that $\mathbf{M} \in \mathrm{CA}(N; 2, k, 2)$.*

Proof. Condition (c) implies covering all assignments of form $(X_i, X_j) = (1,1)$, and (a)–(b) imply covering other assignments due to Propositions 5 and 6. □

Now we take a step back to drop[6] Assumption 4 and reframe the result above to consider the general case of mixing columns that have $\lfloor N/2 \rfloor$ ones and $\lceil N/2 \rceil$ zeros with columns that have $\lceil N/2 \rceil$ ones and $\lfloor N/2 \rfloor$ zeros:

Corollary 2 (Least-Row Sufficiency). *Let $k, N \in \mathbb{Z}_+$. Any matrix $\mathbf{M} \in \{0,1\}^{N \times k}$ in which each column (a) is unique; (b) is not the complement of another column; (c) has at least $\lfloor N/2 \rfloor$ ones and $\lfloor N/2 \rfloor$ zeros; and (d) has a row with the least frequent element of each column is such that $\mathbf{M} \in \mathrm{CA}(N; 2, k, 2)$.*

Proof. By taking the complement of each column with more ones than zeros, we obtain a matrix $\mathbf{M}' \in \{0,1\}^{N \times k}$ satisfying conditions (a), (b), and (c) of Proposition 7. Due to Proposition 3, it follows that $\mathbf{M} \in \mathrm{CA}(N; 2, k, 2)$.

In other words, we ensure that we have a covering array by having a row with the element that is slightly less frequent in each of the columns.

Because the sufficiency condition from Proposition 7 taps back to the construction by Kleitman and Spencer [27], the optimality is straightforward when $N = \mathrm{CAN}(2, k, 2)$ is odd: after deducting the row of ones, we are left with $N-1$ rows of which $\lceil N/2 \rceil$ are zeros, allowing for $\binom{N-1}{\lceil N/2 \rceil} = \mathrm{k}^{\max}(N)$ distinct columns. Nevertheless, the necessary conditions for covering $(X_i, X_j) = (\alpha, \beta)$ when $\alpha \neq \beta$ (Proposition 6 applied both ways) and the sufficient conditions for when $\alpha = \beta$ (Propositions 5 and 7) are helpful to tackle the constrained case.

3.4 Optimal Covering Arrays with Unbalanced Columns

We have seen that there is always an optimal covering array in which all columns are balanced. However, as one would expect, there might be other optimal covering arrays; at least in some cases. For example, consider matrix $\mathbf{M}^{(5)}$:

$$\mathbf{M}^{(5)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & \mathbf{1} \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 1 & 0 & 1 & 1 & \mathbf{0} \\ 0 & 1 & 0 & 1 & 0 & 1 & \mathbf{0} \\ 1 & 0 & 0 & 1 & 1 & 0 & \mathbf{0} \\ 1 & 1 & 1 & 0 & 0 & 0 & \mathbf{0} \end{pmatrix}.$$

Matrix $\mathbf{M}^{(5)}$ is a covering array for 7 parameters, a case that requires at least 6 tests, and thus $\mathbf{M}^{(5)}$ is optimal. You may notice that the last column of $\mathbf{M}^{(5)}$, in bold, is not balanced. However, it is not always the case that we can find covering arrays with unbalanced columns for any number of parameters.

We will use this example to provide a brief but more general intuition for the role of balanced columns in optimal covering arrays. First, notice that matrix $\mathbf{M}^{(5)}$ can be obtained from $\mathbf{M}^{(1)}$. We begin by changing the third element of the

[6] For the sake of generality, we never explicitly referred to columns with $\lfloor N/2 \rfloor$ ones and $\lceil N/2 \rceil$ zeros as balanced in Propositions 5 to 7. However, we primed the reader to think of those as balanced columns in advance to help communicate the results.

last column of $\mathbf{M}^{(1)}$, hence replacing column $[1\ 1\ 1\ 0\ 0\ 0]^T$ with $[1\ 1\ \mathbf{0}\ 0\ 0\ 0]^T$. Because only the first two elements in the last column are one, we must necessarily have both a zero and a one in the first two rows of all other columns to have a covering array. Hence, we must drop columns 4, 7, and 9. Starting from $\mathbf{M}^{(1)}$, we show below the changed element in the last column in bold and the removed columns in gray, so that the columns in black correspond to $\mathbf{M}^{(5)}$:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & \mathbf{0} \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Second, consider how the last column prevents us from adding any other column to $\mathbf{M}^{(5)}$ and still obtain a valid covering array. As we have seen above, a new column would necessarily need at least a zero and a one in the first two rows. Without loss of generality, we may assume that the value one is in the first row and the value zero in the second row, since we can otherwise take the complement by Preposition 3. Hence, any new column would necessarily have one in the first row and zero in the second row. However, a new column could not have two ones in the remaining rows since otherwise it would be identical to one of the existing columns, contradicting either Proposition 1 or that we have a valid covering array. Likewise, a new column with more or less than two ones would imply a containment ($\leq$) relationship with one of the existing columns, contradicting either Proposition 6 or that we have a valid covering array.

More generally, using an unbalanced column with fewer (or more) ones in a covering array prevents us from using other columns that are supersets (or subsets) of that column. When it comes to balanced columns, that means trading many balanced columns for an unbalanced one. For example, by using the unbalanced column $[1\ 1\ 0\ 0\ 0\ 0]^T$, we could not use columns $[1\ 1\ 0\ 0\ 0\ 1]^T$, $[1\ 1\ 0\ 0\ 1\ 0]^T$, $[1\ 1\ 0\ 1\ 0\ 0]^T$, $[1\ 1\ 1\ 0\ 0\ 0]^T$, and their complements. Hence, while it is possible that some optimal covering arrays with N tests have unbalanced columns, that possibility vanishes as the number of parameters approaches $\mathrm{k}^{\max}(N)$.

4 Optimization with Pairwise Constraints

So far we have assumed that all pairs of assignments must be covered. But if there is a forbidden assignment that must be avoided, say $(X_i, X_j) \neq (\alpha, \beta)$, then we need to adapt the constructions discussed in Section 3 accordingly. In this case, we must cover all but the pairs of assignments which are forbidden.

For example, what if we have $k = 5$ parameters subject to $(X_4, X_5) \neq (1, 0)$? With $N = 6$ rows as in the unconstrained case, we should not use balanced columns for both X_4 and X_5, since otherwise we would automatically cover $(X_4, X_5) = (1, 0)$. The solution changes even more if we add more constraints,

such as $(X_3, X_5) \neq (1, 0)$ and $(X_2, X_5) \neq (1, 0)$. Consider matrices $\mathbf{M}^{(6)}$ to $\mathbf{M}^{(9)}$:

$$\mathbf{M}^{(6)}=\begin{pmatrix}1&1&1&1&1\\1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&1\\0&0&0&1&1\\1&1&1&1&0\end{pmatrix},\mathbf{M}^{(7)}=\begin{pmatrix}1&1&1&1&1\\1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&1\\0&0&0&1&1\\1&1&1&\mathbf{0}&0\end{pmatrix},\mathbf{M}^{(8)}=\begin{pmatrix}1&1&1&1&1\\1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&1\\0&0&0&1&1\end{pmatrix},\mathbf{M}^{(9)}=\begin{pmatrix}1&1&1&1&1\\1&0&0&0&0\\0&1&0&\mathbf{1}&\mathbf{1}\\0&0&1&\mathbf{1}&1\\0&0&0&\mathbf{0}&\mathbf{0}\\1&1&1&\mathbf{0}&\mathbf{1}\end{pmatrix}.$$

Matrix $\mathbf{M}^{(6)}$ is an optimal unconstrained covering array with $k = 5$. In turn, matrix $\mathbf{M}^{(7)}$ is an optimal[7] covering array subject to $(X_4, X_5) \neq (1, 0)$. By also adding $(X_3, X_5) \neq (1, 0)$, matrix $\mathbf{M}^{(8)}$ is an optimal covering array with surprisingly fewer tests than before. By further adding $(X_2, X_5) \neq (1, 0)$, matrix $\mathbf{M}^{(9)}$ is now an optimal covering array and the number of tests goes back to six. We put in bold the elements in matrices $\mathbf{M}^{(7)}$ to $\mathbf{M}^{(9)}$ that differ from $\mathbf{M}^{(6)}$.

Those examples highlight important distinctions from the unconstrained case, which we enumerate below. They form the basis for the remainder of the paper:

(i) The columns are not—and cannot—be all balanced: the fourth column of $\mathbf{M}^{(5)}$ has more zeros than ones, and the fifth column of both $\mathbf{M}^{(6)}$ and $\mathbf{M}^{(7)}$ have more ones than zeros. We present a same underlying principle for both cases in Section 4.1.

(ii) The minimum number of tests may increase (as from $\mathbf{M}^{(8)}$ to $\mathbf{M}^{(9)}$) or decrease (as from $\mathbf{M}^{(7)}$ to $\mathbf{M}^{(8)}$) as we add more pairwise constraints. (But how is that even possible?) We discuss the factors involved in Section 4.2.

(iii) The optimal covering arrays still seem to have as many balanced—or closer to balanced—columns as possible, save some notable exceptions. We discuss circumstances which may exclude some or all columns from being balanced in optimal covering arrays if their parameters are subject to pairwise constraints in Section 4.3. In contrast, we state a conjecture based on the observation that it is apparently possible to find valid covering arrays having balanced columns for unconstrained parameters in Section 4.4.

(iv) The column chosen for one parameter may further affect the choice for another parameter if either needs an unbalanced column. For example, the fourth column of $\mathbf{M}^{(9)}$ is a balanced column, but it is not the same balanced column used in $\mathbf{M}^{(6)}$ because that column is not compatible with the fifth column of $\mathbf{M}^{(9)}$. We return to this discussion in Section 5.

4.1 Necessary Conditions

We present the following result as a unifying principle for the number of zeros and ones differing slightly across columns of matrices $\mathbf{M}^{(6)}$ to $\mathbf{M}^{(9)}$:

[7] The claim that the solutions presented for the constrained case are optimal have been validated with either the theory or the algorithms discussed later in the paper.

Lemma 2 (1–0 Constraint). *If a reduced constrained pairwise covering problem*[8] *is subject to* $(X_i, X_j) \neq (1, 0)$*, then in any valid covering array* $\mathbf{M} \in \{0, 1\}^{N \times k}$ *the column* $\mathbf{M}_i$ *has more zeros—and less ones—than* $\mathbf{M}_j$*.*

Proof. If $(X_i, X_j) = (1, 0)$ should not be covered, then $X_i = 1 \rightarrow X_j = 1$ in all tests. But if $(X_i, X_j) = (0, 1)$ should be covered in some test, then $X_j = 1$ occurs more often than $X_i = 1$; and, conversely, $X_i = 0$ more often than $X_j = 0$. Hence, there are relatively more ones in $\mathbf{M}_j$ and therefore more zeros in $\mathbf{M}_i$. □

In other words, if X_i and X_j are subject only to $(X_i, X_j) \neq (1, 0)$, any covering array $\mathbf{M}$ has more zeros in $\mathbf{M}_i$ (e.g., fourth column of $\mathbf{M}^{(7)}$) or more ones in $\mathbf{M}_j$ (e.g., fifth column of both $\mathbf{M}^{(8)}$ and $\mathbf{M}^{(9)}$).

Now we generalize Lemma 2 to lay out necessary conditions for pairwise covering subject to pairwise constraints with any pair of values:

Theorem 1 (α–β Constraints). *For* w_i *and* w_j *as the number of ones of parameters* X_i *and* X_j *in a reduced constrained pairwise covering problem, then:*

$$(X_i, X_j) \neq (0, 0) \rightarrow \quad w_i + w_j \geq N + 1. \tag{1}$$
$$(X_i, X_j) \neq (0, 1) \rightarrow \quad w_i - w_j \geq 1. \tag{2}$$
$$(X_i, X_j) \neq (1, 0) \rightarrow \quad w_j - w_i \geq 1. \tag{3}$$
$$(X_i, X_j) \neq (1, 1) \rightarrow \quad w_i + w_j \leq N - 1. \tag{4}$$

Proof. Let z_i be the number of zeros of X_i ($z_i = N - w_i$), and $\overline{X}_i$ the complement of X_i with z_i ones and w_i zeros. We proceed case by case as follows.

In the case of (1), $(X_i, X_j) \neq (0, 0) \rightarrow (\overline{X}_i, X_j) \neq (1, 0)$; and, by Lemma 2, $w_j - z_i \geq 1 \rightarrow w_j - (N - w_i) \geq 1 \rightarrow w_i + w_j \geq N + 1$.

In the case of (2), we apply Lemma 2 after flipping the places of X_i and X_j.

In the case of (3), we apply Lemma 2 directly.

In the case of (4), similar to the first one, $(X_i, X_j) \neq (1, 1) \rightarrow (X_i, \overline{X}_j) \neq (1, 0)$; and, by Lemma 2, $z_j - w_i \geq 1 \rightarrow (N - w_j) - w_i \geq 1 \rightarrow w_i + w_j \leq N - 1$. □

4.2 Optimality Conditions

On the one hand, forbidding a pair of assignments prevents the combination of certain columns in the solution, such as using pairs of balanced columns. Because balanced columns can typically be combined in larger number (see discussion in Section 3.4), we may expect the number of required tests to grow.

On the other hand, we do not need to cover a pair of assignment that has been forbidden, which implicitly eliminates covering constraints that we have taken for granted throughout the paper. Sometimes this benefit outweighs the inconvenience of having more pairwise constraints. To the best of our knowledge, a reduction on the number of tests such as from $\mathbf{M}^{(7)}$ to $\mathbf{M}^{(8)}$ have only been previously discussed in the context of nonbinary parameters [4].

[8] We remind the reader of Definition 1 in Section 2 for our assumptions in this work.

To prove optimality given that the number of tests may decrease, we may rely on a smaller coverage problem on parameters not constraining one another:

Lemma 3 (Lower Bound). *Let $X_1, X_2, \ldots, X_{k'}$ be binary parameters of a reduced constrained pairwise covering problem with $k \geq k'$ parameters, for which there are no forbidden pairwise assignments between the first k' parameters. Then any valid covering array on k parameters has at least* $\mathrm{CAN}(2, k', 2)$ *tests.*

Proof. If there are no forbidden assignments between the first k' parameters, then every pair of assignments among those should be covered. Hence, at least as many tests as in the unconstrained case on k' parameters are needed. □

For example, matrix $\mathbf{M}^{(8)}$ denotes a covering problem on $k = 5$ variables subject to $(X_3, X_5) \neq (1, 0)$ and $(X_4, X_5) \neq (1, 0)$. Since there is no forbidden pair of assignments between the first four parameters, then at least $\mathrm{CAN}(2, 4, 2) = 5$ rows are needed under those two constraints, and therefore $\mathbf{M}^{(8)}$ is optimal.

4.3 Going Off Balance

We have seen a few plot twists, but there is more yet to come. In this section, we will focus on the parameters that are subject to pairwise constraints. In some cases, those parameters may have some—or even all—columns being unbalanced in optimal covering arrays. In full disclosure, the cases discussed in this section emerged by analyzing preliminary computational experiments. They are likely not the only unusual cases to consider, but they are possibly among the most common up to the number of parameters and pairwise constraints used in our experiments in Section 6, given that we came across them a few times.

To begin our discussion, consider matrices $\mathbf{M}^{(10)}$ and $\mathbf{M}^{(11)}$, each of which having submatrices of ones in the secondary diagonal defined by the lines below:

$$\mathbf{M}^{(10)} = \left(\begin{array}{cc|cc} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{array}\right), \mathbf{M}^{(11)} = \left(\begin{array}{cc|cccc} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right).$$

Matrix $\mathbf{M}^{(10)}$ is an optimal covering array on $k = 4$ parameters subject to $(X_1, X_3) \neq (0, 0)$, $(X_1, X_4) \neq (0, 0)$, $(X_2, X_3) \neq (0, 0)$, and $(X_2, X_4) \neq (0, 0)$, with $N = 6$ rows and all columns having 4 ones and 2 zeros. In turn, matrix $\mathbf{M}^{(11)}$ is an optimal covering array on $k = 6$ parameters subject to $(X_1, X_3) \neq (0, 0)$, $(X_1, X_4) \neq (0, 0)$, $(X_1, X_5) \neq (0, 0)$, $(X_1, X_6) \neq (0, 0)$, $(X_2, X_3) \neq (0, 0)$, $(X_2, X_4) \neq (0, 0)$, $(X_2, X_5) \neq (0, 0)$, and $(X_2, X_6) \neq (0, 0)$, with $N = 7$ rows and the first two columns having 5 ones and 2 zeros. In both cases, we need one more test in comparison to the unconstrained case.

If we model a graph with the parameters as vertices and edges between parameters in 0–0 pairwise constraints, the cases just described would correspond to complete bipartite graphs. In the first case, the graph would be isomorphic to $K_{2,2}$, with one partition corresponding to $A := \{X_1, X_2\}$ and the other to $B := \{X_3, X_4\}$. In the second case, the graph would be isomorphic to $K_{2,4}$, with one partition corresponding to $A := \{X_1, X_2\}$ and the other to $B := \{X_3, X_4, X_5, X_6\}$. That makes it easier to observe the following: if the value of a parameter in either partition is set to zero, then the 0–0 pairwise constraints imply that the value of all the parameters in the other partition should be one. For the submatrices of $\mathbf{M}^{(10)}$ and $\mathbf{M}^{(11)}$ defined by the vertical and horizontal lines, the main diagonal having zeros for the parameters in A (top left) and in B (bottom right) imply that the secondary diagonal has submatrices of ones.

To proceed, we formalize the use of graphs to represent pairwise constraints:

Definition 2 (Induced Constraint Graph). *For a subset of parameters P of a constrained pairwise covering problem, let $G[P] = (P, E)$ be the induced constraint graph on P with vertices in P corresponding to the parameters and edges in E corresponding to all pairs of parameters in P having a pairwise constraint, be that constraint explicitly defined or implied by the other constraints.*

The circumstances above lead to nontrivial lower bounds on the number of rows of covering arrays and on the number of ones in some of their columns:

Lemma 4 (0–0 $K_{m,n}$ Bound). *If a subset of parameters P of a reduced constrained pairwise covering problem has an induced constraint graph $G[P]$ that is isomorphic to the complete bipartite graph $K_{m,n}$ with partitions A and B, $m := |A| \geq 2$ and $n := |B| \geq 2$, and such that $(X_i, X_j) \neq (0, 0)\ \forall X_i \in A, X_j \in B$, then*

$$N \geq \mathrm{CAN}(2, m, 2) + \mathrm{CAN}(2, n, 2) - 2 \tag{5}$$

and

$$w_i \geq \mathrm{CAN}(2, n, 2) \qquad \forall X_i \in A, \tag{6}$$

$$w_i \geq \mathrm{CAN}(2, m, 2) \qquad \forall X_i \in B, \tag{7}$$

where w_i is the number of ones in the column associated with parameter X_i.

Proof. To break the symmetry in the proof, let $P_1 \in \{A, B\}$ (i.e., one of the partitions among A and B), $P_2 \in \{A, B\} \setminus \{P_1\}$ (i.e., the other partition), so that any claim proven for P_1 applies to P_2 and vice versa.

Due to the pairwise constraints $(X_i, X_j) \neq (0, 0)$ for $X_i \in P_1$ and $X_j \in P_2$, it follows that $X_i \in P_1 \wedge X_i = 0 \rightarrow X_j = 1\ \forall X_j \in P_2$. In other words, any row with a zero in the columns corresponding to parameters in P_1 has a value of one in all columns corresponding to parameters in P_2.

Since there are no pairwise constraints within P_1, a valid covering array must cover all pairwise assignments $(X_i, X_j) = (\alpha, \beta)$ for $X_i \in P_1$ and $X_j \in P_1$. But with the exception of $(\alpha, \beta) = (1, 1)$, all other pairwise assignments within P_1 are covered by rows in which all parameters in P_2 are set to 1.

Considering the covering array problem on the parameters of P_1 alone, it takes a covering array with at least $\mathrm{CAN}(2, |P_1|, 2)$ rows to cover all pairwise assignments. Moreover, no more than one of the rows of such a covering array has only the value one, or it would be a duplicate. Hence, it follows that a minimum of $\mathrm{CAN}(2, |P_1|, 2) - 1$ rows of any valid covering array on all parameters of P must have at least one value zero in columns corresponding to parameters in P_1.

Because the rows with a value zero in columns corresponding to parameters in P_1 and to parameters in P_2 are disjoint, it follows that $N \geq (\mathrm{CAN}(2, |P_1|, 2) - 1) + (\mathrm{CAN}(2, |P_2|, 2) - 1) = \mathrm{CAN}(2, m, 2) + \mathrm{CAN}(2, n, 2) - 2$, proving (5).

Finally, the column associated with each parameter in P_1 in any valid covering array has at least as many ones as the number of rows in which the columns associated with the parameters in P_2 have at least one zero ($\mathrm{CAN}(2, |P_2|, 2) - 1$), since for all of those the value of all parameters in P_1 is one. If there are other parameters in P_1, there should be at least another row with the value one for each parameter $X_i \in P_1$ to cover the pairwise assignment for $(X_i, X_j) = (1, 0)$ for another $X_j \in P_1$. Hence, $w_i \geq \mathrm{CAN}(2, |P_2|, 2)\ \forall X_i \in P_1$, proving (6)–(7). □

The proof above entails the construction used for obtaining matrices $\mathbf{M}^{(10)}$ and $\mathbf{M}^{(11)}$. Let $A := \{X_1, \ldots, X_m\}$ and $B := \{X_{m+1}, \ldots, X_{m+n}\}$. Without loss of generality, let $\overline{\mathbf{M}}^A \in \mathrm{CA}^*(2, m, 2)$ and $\overline{\mathbf{M}}^B \in \mathrm{CA}^*(2, n, 2)$ be valid covering arrays with rows of ones, since any other valid covering array can be turned to a valid covering array with a row of ones upon complementing all the columns having a value of zero in a chosen row. Moreover, let $\mathbf{M}^A$ and $\mathbf{M}^B$ be the submatrices of $\overline{\mathbf{M}}^A$ and $\overline{\mathbf{M}}^B$ without the row of ones. Finally, let

$$\mathbf{M}^{(12)} := \begin{bmatrix} \mathbf{M}^A & \mathbf{W}_{\mathrm{CAN}(2,m,2)-1,n} \\ \mathbf{W}_{\mathrm{CAN}(2,n,2)-1,m} & \mathbf{M}^B \end{bmatrix}$$

be a matrix containing $\mathbf{W}_{i,j}$ as submatrices of ones with i rows and j columns. Hence, matrix $\mathbf{M}^{(12)}$ generalizes the construction of both $\mathbf{M}^{(10)}$ and $\mathbf{M}^{(11)}$.

This is the point in which we observe that *optimal covering arrays may never have balanced columns for certain parameters*. If $|B| > |A|$ and there are possibly more rows in $\overline{\mathbf{M}}^B$ than in $\overline{\mathbf{M}}^A$, as is the case with matrix $\mathbf{M}^{(11)}$, then the columns for the parameters in A can have significantly more ones than zeros.

More generally, any valid covering array on a set of parameters P having a set of pairwise constraints that meet the conditions of Lemma 4 would have that same structure upon permutation of rows and columns, with the exception that the covering arrays $\overline{\mathbf{M}}^A$ and $\overline{\mathbf{M}}^B$ on the subset of parameters A and B are not optimal in all covering arrays.

Note, however, that there is nothing special about constraining $(0, 0)$ pairs. Hence, we generalize Lemma 4 to the cases that can be obtained by replacing some of the parameters with their complements if we adjust the pairwise constraints accordingly. We first characterize such parameter partitionings:

Definition 3 (Coherent Partition of Induced Constraint Graph). *An induced constraint graph $G[P] = (P, E)$ has a* coherent partition *of parameters*

(P_0, P_1) if, for every pairwise constraint $(X_i, X_j) \neq (\alpha, \beta)$, it follows that $X_i \in P_\alpha$ and $X_j \in P_\beta$.

In other words, a partitioning of parameters is coherent is the value of every parameter in the forbidden assignments associated with it is always the same (in this case, the value 0 for parameters in P_0 and 1 for parameters in P_1).

Theorem 2 (Coherent $K_{m,n}$ Bound). *If a subset of parameters P of a reduced constrained pairwise covering problem has (i) a coherent partition of parameters (P_0, P_1); and (ii) an induced constraint graph $G[P]$ that is isomorphic to the complete bipartite graph $K_{m,n}$ with partitions A and B, $m := |A| \geq 2$ and $n := |B| \geq 2$, then*

$$N \geq \text{CAN}(2, m, 2) + \text{CAN}(2, n, 2) - 2 \tag{8}$$

and

$$w_i \geq \text{CAN}(2, n, 2) \qquad \forall X_i \in A \cap P_0, \tag{9}$$
$$w_i \geq \text{CAN}(2, m, 2) \qquad \forall X_i \in B \cap P_0, \tag{10}$$
$$w_i \leq N - \text{CAN}(2, n, 2) \qquad \forall X_i \in A \cap P_1, \tag{11}$$
$$w_i \leq N - \text{CAN}(2, m, 2) \qquad \forall X_i \in B \cap P_1, \tag{12}$$

where w_i is the number of ones in the column associated with parameter X_i.

Proof. By taking the complement of all parameters in P_1 in the forbidden assignments, we obtain a new constrained pairwise covering problem to which Lemma 4 applies. With a similar logic to Proposition 3, except that considering the constrained case, any valid covering array to this new covering problem and to the covering problem in the statement that we are proving are interchangeable upon complement of the parameters that are in P_1 for the latter.

Due to the interchangeability of valid covering arrays between the two covering problems, the bound (5) from Lemma 4 implies the claimed bound (8).

For the parameters in P_0, for which we do *not* take complements between the valid covering arrays of the two covering problems, the bounds (6) and (7) from Lemma 4 directly imply the claimed bounds (9) and (10).

For the parameters in P_1, for which we *do* take complements between the valid covering arrays of the two covering problems, the bounds (6) and (7) from Lemma 4 actually imply different bounds on columns of the covering problem:

$$z_i \geq N - \text{CAN}(2, n, 2) \qquad \forall X_i \in A \cap P_1, \tag{13}$$
$$z_i \geq N - \text{CAN}(2, m, 2) \qquad \forall X_i \in B \cap P_1, \tag{14}$$

where z_i is the number of ones in the column associated with parameter X_i. With $z_i = N - w_i$, those new bounds (13) and (14) imply the claimed bounds (11) and (12), respectively. □

The use of graphs for modeling pairwise constraints and their complements has been first proposed in Danziger et al. [14], where an algorithm of exponential complexity produces covering arrays of minimum size. Such graphical models have also been used in complexity results [14,39] and applied to study special lower bounds for this problem [14,51]. In fact, the next results are also inspired by the use of a clique of pairwise constraints in Danziger et al.'s algorithm.

To illustrate this next case, let us first consider matrices $\mathbf{M}^{(13)}$ and $\mathbf{M}^{(14)}$:

$$\mathbf{M}^{(13)}=\begin{pmatrix}0&1&1\\1&0&1\\1&1&0\end{pmatrix},\ \mathbf{M}^{(14)}=\left(\begin{array}{ccc|c}0&1&1&1\\0&1&1&0\\\hline1&0&1&1\\1&0&1&0\\\hline1&1&0&1\\1&1&0&0\end{array}\right).$$

Both of those are optimal covering arrays subject to the same pairwise constraints: $(X_1, X_2) \neq (0, 0)$, $(X_1, X_3) \neq (0, 0)$, and $(X_2, X_3) \neq (0, 0)$. In other words, we have a clique of constraints among the first three parameters while all other parameters are unconstrained. When one of those first three parameters has the value zero, the other two parameters must have the value one.

What changes from one case to another is the existence and an unconstrained parameter. We separate the constrained parameters from the unconstrained parameter with a vertical line. In the first case with $k = 3$ parameters, given that there are no unconstrained parameters, we actually have *one less test* than in the optimal unconstrained case. Once we add one unconstrained parameter and $k = 4$, however, the need to cover both $(X_i, X_4) = (0, 0)$ and $(X_i, X_4) = (0, 1)$ for every constrained parameter X_i implies that we need two rows in which each constrained parameter has the value zero. Given the clique of pairwise constraints, that effectively doubles the number of rows in comparison to $k = 3$. Now that we have rows that are identical with respect to the constrained parameters alone, we separate rows that are distinct on the constrained parameters with horizontal lines. With $k = 4$ parameters, we have *one more test* than in the optimal unconstrained case.

This is point in which we observe that *all parameters may have unbalanced columns in optimal covering arrays.* If we consider a covering array on $k \geq 3$ parameters with constraints of the form $(X_i, X_j) \neq (0, 0)\ \forall i \in \{1, \ldots, k\}, j \in \{1, \ldots, k\} \setminus \{i\}$, hence forming a fully connected graph on k parameters and generalizing the case with matrix $\mathbf{M}^{(13)}$, then each column has $k - 1$ ones and one zero. However, a fully connected graph is a rather specific and unlikely case.

Hence, our object of interest now is not exactly a clique, but a combination of a clique and isolated vertices. That has been denoted as a *sparse split graph* in the literature [5]. We use the following characterization for convenience:

Definition 4 (Sparse Split Induced Constraint Graph). *An induced constraint graph $G[P] = (P, E)$ is* sparse split *with a partition of parameters (P_C, P_U) if the induced constraint graph $G[P_C]$ is isomorphic to the clique K_n, where*

$n := |P_C|$, and there are no pairwise constraints between the parameters in P_U and the parameters in P.

However, matrices $\mathbf{M}^{(13)}$ and $\mathbf{M}^{(14)}$ are not sufficient to capture all the nuance of what happens when the pairwise constraints define a sparse split graph. To continue the discussion, let us consider matrices $\mathbf{M}^{(15)}$ and $\mathbf{M}^{(16)}$:

$$\mathbf{M}^{(15)}=\left(\begin{array}{ccc|cccc} 0&1&1&1&1&1&1\\ 0&1&1&0&0&0&0\\ \hline 1&0&1&1&1&0&0\\ 1&0&1&0&0&1&1\\ \hline 1&1&0&1&0&1&0\\ 1&1&0&0&1&0&1 \end{array}\right),\ \mathbf{M}^{(16)}=\left(\begin{array}{ccc|cccc|cc} 0&1&1&1&1&1&1&1&1\\ 0&1&1&0&0&0&0&0&0\\ 1&0&1&1&1&0&0&1&0\\ 1&0&1&0&0&1&1&0&1\\ 1&1&0&1&0&1&0&0&0\\ 1&1&0&0&1&0&1&0&0\\ \hline\hline 1&1&0&0&0&0&0&1&1 \end{array}\right).$$

Both of those are optimal covering arrays subject to the same pairwise constraints mentioned before. In the first case with $k = 7$ parameters, we keep a vertical line separating constrained parameters from unconstrained parameters, as well as horizontal lines separating distinct rows on the constrained parameters. When we compare the cases of $k = 4$ (matrix $\mathbf{M}^{(14)}$) and $k = 7$ (matrix $\mathbf{M}^{(15)}$), we see that adding more unconstrained parameters did not impact the optimal number of tests required (in fact, the number of tests for $k = 7$ is the *same* as in the optimal unconstrained case). In the second case with $k = 9$ parameters, however, there is one more test than before, and therefore *one more test* than in the optimal unconstrained case. We use the double vertical and horizontal lines to highlight that matrix $\mathbf{M}^{(15)}$ is the submatrix of $\mathbf{M}^{(16)}$ on the first 6 rows and 7 columns. We establish with the next result the need for at least 2 rows per parameter in the clique, and possibly more rows in cases such as with $\mathbf{M}^{(16)}$:

Lemma 5 (0–0 K_n Bound[9]). *If a subset of parameters P of a reduced constrained pairwise covering problem induces a sparse split constraint graph $G[P]$ with partitions (P_C, P_U), $n := |P_C| \geq 3$ and $m := |P_U| \geq 1$, then*

$$N \geq 2n + \alpha, \tag{15}$$

where $2n + \alpha$, $\alpha \geq 0$, is the smallest number of rows such that (a) there are m columns defining a valid covering array on the parameters in P_U; and (b) we can partition that covering array in n subsets of tests, each of which having $X_i = 0$ and $X_i = 1$ for each parameter $X_i \in P_U$. Moreover,

$$w_i \geq 2(n-1) \quad \forall i \in P_C, \tag{16}$$

$$\sum_{i \in P_C} w_i \geq (2n + \alpha)(n-1), \tag{17}$$

where w_i is the number of ones in the column of $X_i \in P_C$ in any solution.

[9] We will keep referring to this family of bounds as K_n because the clique is central to the result and cliques are more commonly known than sparse split graphs.

Proof. Let us assume that $P_C := \{X_1, \ldots, X_n\}$. Hence, let $M := [M_C \mid M_U]$ be any valid covering array for the constrained pairwise covering problem on P, in which M_C has n columns and M_U has m columns. Without loss of generality, we may assume that the first rows of M_C have the value zero in X_1, then the next rows of M_C have the value zero in X_2, and so on, since each test has at most one value zero in P_C, and thus we can rearrange the rows of any covering array M in such a way. Since there are no pairwise constraints between parameters in P_C and P_U, there should be at least two rows with $X_i = 0$ for each $X_i \in P_C$ to cover the pairwise assignments $(X_i, X_j) = (0, 0)$ and $(X_i, X_j) = (0, 1)$ for each $X_j \in P_U$. With n of such parameters, it follows that $N \geq 2n$, proving (15).

Since there are no pairwise constraints within the parameters in P_U, it follows that $M_U \in \mathrm{CA}(N; 2, m, 2)$, hence proving (a). Moreover, M_U is a valid covering array on P_U in which $X_j = 0$ and $X_j = 1$ for every $X_j \in P_U$ in the subset of tests for which $X_i = 0$ for each $X_i \in P_C$, since otherwise we would not cover $(X_i, X_j) = (0, 0)$ or $(X_i, X_j) = (0, 1)$. By assumption, those subsets of tests correspond to contiguous subsets of rows of M, the first with $X_1 = 0$, the second with $X_2 = 0$, and so on, with a total of n subsets and therefore proving (b).

Since each row has no more than one zero in the first n columns, and thus at least $n - 1$ ones, then $\sum_{i \in P_C} w_i \geq (2n + \alpha)(n - 1)$, proving (16). Since each of the first n columns has $n - 1$ sets with at least 2 rows each having value one in the column, then $w_i \geq 2(n - 1)\ \forall X_i \in P$, proving (17). □

Similarly to what we did before with Lemma 4, we now use the concept of a coherent partition of parameters to generalize the result from Lemma 5:

Theorem 3 (Coherent K_n Bound). *If a subset of parameters P of a reduced constrained pairwise covering problem (i) has a coherent partition of parameters (P_0, P_1); and (ii) induces a sparse split constraint graph $G[P]$ with partitions (P_C, P_U), $n := |P_C| \geq 3$ and $m := |P_U| \geq 1$, then*

$$N \geq 2n + \alpha, \tag{18}$$

where $2n + \alpha$, $\alpha \geq 0$, is the smallest number of rows such that (a) there are m columns defining a valid covering array on the parameters in P_U; and (b) we can partition that covering array in n subsets of tests, each of which having $X_i = 0$ and $X_i = 1$ for each parameter $X_i \in P_U$. Moreover,

$$w_i \geq 2(n - 1) \quad \forall i \in P_C \cap P_0, \tag{19}$$

$$w_i \leq N - 2(n - 1) \quad \forall i \in P_C \cap P_1, \tag{20}$$

$$\sum_{i \in P_C \cap P_0} w_i + \sum_{i \in P_C \cap P_1} (N - w_i) \geq (2n + \alpha)(n - 1), \tag{21}$$

where w_i is the number of ones in the column of $X_i \in P_C$ in any solution.

Proof. This proof is very similar in structure to the proof of Theorem 2.

By taking the complement of all parameters in P_1 in the forbidden assignments, we obtain a new constrained pairwise covering problem to which Lemma 5

applies. With a similar logic to Proposition 3, except that considering the constrained case, any valid covering array to this new covering problem and to the covering problem in the statement that we are proving are interchangeable upon complement of the parameters that are in P_1 for the latter.

Due to the interchangeability of valid covering arrays between the two covering problems, the bound (15) from Lemma 5 implies the claimed bound (18). Similarly, the validity of conditions (a) and (b) from Lemma 5 imply the validity of the claimed conditions (a) and (b) here.

For the parameters in P_0, for which we do *not* take complements between the valid covering arrays of the two covering problems, the bound (16) from Lemma 5 directly implies the claimed bound (19).

For the parameters in P_1, for which we *do* take complements between the valid covering arrays of the two covering problems, the bounds (16) and (17) from Lemma 5 actually imply different bounds on columns of the covering problem:

$$z_i \leq N - 2(n-1) \quad \forall i \in P_C \cap P_1, \tag{22}$$

$$\sum_{i \in P_C \cap P_0} w_i + \sum_{i \in P_C \cap P_1} z_i \geq (2n + \alpha)(n-1), \tag{23}$$

where z_i is the number of ones in the column associated with parameter X_i. With $z_i = N - w_i$, those new bounds (22) and (23) imply the claimed bounds (20) and (21), respectively. □

To finalize this part, we characterize a relevant case in which $\alpha \neq 0$. This is motivated by the construction of matrix $\mathbf{M}^{(15)}$, which we repeat for convenience:

$$\mathbf{M}^{(15)} = \left(\begin{array}{ccc|cccc} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ \hline 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}\right).$$

For the unconstrained parameters (to the right of the vertical line), note how there is a zero and a one for each column within every group defined by the horizontal lines. Essentially, our only degree of freedom is in deciding if the zero or the one goes first in each column within each group. That naturally produces balanced columns for the unconstrained parameters if each group has two rows. By restricting the columns within the first group to always have the value one first, we prevent having columns that are complements. By changing the order of the rows having each value in the other two groups, we can only obtain the four distinct columns that we observe to the right of the vertical line. Hence, having one more parameter would entail having another row in any valid covering array. We generalize the relationship between the number of constrained and unconstrained parameters and the need for an additional test as follows:

Corollary 3 (Extra Test Threshold for Coherent K_n Bound). *When the conditions of Theorem 3 apply, with n as the number of constrained parameters*

inducing a clique and m as the number of unconstrained parameters, then

$$m > 2(n-1) \rightarrow \alpha \geq 1.$$

Proof. Without loss of generality, let us assume that $P_1 = \emptyset$; since we can define an equivalent covering problem by taking the complement of all parameters in P_1 in the forbidden assignments, and then obtain a valid covering array for the original covering problem by taking the complement of the columns associated with P_1 in a valid covering array for the new problem, as in previous proofs.

Now let us suppose, for contradiction, that $\alpha = 0$ yields a tight bound on N. In other words, there is a valid covering array with $2n$ tests. In such an array, there are two rows in which $X_i = 0$ for each $X_i \in P_C$, which are disjoint from the rows in which $X_j = 0$ for a different parameter $X_j \in P_C$.

Since this array should cover the pairwise assignments $(X_i, X_j) = (0, 0)$ and $(X_i, X_j) = (0, 1)$ between each $X_i \in P_C$ and each unconstrained parameter $X_j \in P_U$, then there is one row covering each such assignment. If two columns on the unconstrained parameters $X_j \in P_U$ cover all of those assignments with the same rows, then those columns are identical, and that would contradict Proposition 1 with respect to the columns on the unconstrained parameters in P_U defining a valid covering array. For each pair of columns on P_U, the rows containing the assignments $X_j = 0$ and $X_j = 1$ should thus be different within the group in which $X_i = 0$ for at least one parameter $X_i \in P_C$. There are $2n$ columns that can be defined by alternating the order of the assignments to X_j within those groups. But since half of those would be complements since there are only two rows per group, then no more than $2(n-1)$ of such columns can define a valid covering array on the unconstrained parameters in P_U due to Proposition 2. However, that would contradict that $m > 2(n-1)$, hence implying that the initial assumption is false and thus that $\alpha \geq 1$ in this case.

4.4 Regaining Balance

Let us have a moment of hope now. In this section, we will focus on the parameters that are not subject to pairwise constraints. Despite the lengthy discussion about optimal covering arrays necessarily having unbalanced columns, that has so far applied only to the parameters that are subject to pairwise constraints. Hence, we have not yet seen a contradiction to the use of balanced columns for all other parameters. For that reason, we posit the following:

Conjecture 1 (Balanced Columns for Unconstrained Parameters) *In a reduced pairwise constrained covering problem, there is an optimal covering array in which parameters without pairwise constraints have balanced columns.*

In fact, this argument revisits the discussion from Section 3.4 about using unbalanced columns in optimal unconstrained covering arrays. While it is possible to use unbalanced columns in some cases, their use may further limit the number of columns that can be put together to produce a valid covering array.

Moreover, if two parameters are unconstrained, then their columns remain interchangeable. Hence, we can significantly reduce the symmetry when formulating the problem of finding optimal covering arrays if we assume that we simply need a certain number of balanced columns to be assigned to the unconstrained parameters. Since any association of those columns to parameters is possible, then the exact association is irrelevant. That is how Conjecture 1 will help in the next section.

5 Integer Programming Models

Following the developments from the last two sections, we are now able to propose an algorithm and an heuristic for producing optimal, or at least valid and hopefully good, covering arrays. Our general approach is outlined in Algorithm 1. It is based on solving a series of subproblems with the support of IP models, each considering covering arrays with a fixed number of tests.

The elements below are common throughout our approach and IP models:

- the set of parameter indices $\mathcal{P} := \{1, \ldots, k\}$;
- the set of pairs of parameter indices that are subject to a pairwise constraint $\mathcal{F} \subseteq \{(i_1, i_2) : i_1 \in \mathcal{P}, i_2 \in \mathcal{P} \setminus \{i_1\}\}$; and
- the forbidden assignment for each such pair $\mathscr{F} : \mathcal{F} \to \{0,1\} \times \{0,1\}$.

Starting with as many tests N as we would need in the unconstrained case, Algorithm 1 either (i) increments N as far as needed while trying to find a first

Algorithm 1 General approach for solving reduced pairwise constrained covering array problems, starting with the optimal number of unconstrained tests

```
 1: function Solve(k, P, F, 𝓕)
 2:     N ← CAN(2, k, 2)                ▷ Start with number of tests from unconstrained case
 3:     M ← FindCoveringArray(k, P, F, 𝓕, N)      ▷ Try to find a covering array
 4:     if M = ∅ then                   ▷ We assume M = ∅ if we cannot find a covering array
 5:         repeat                      ▷ If we do not find a covering array at first...
 6:             N ← N + 1               ▷ we look for one with more tests
 7:             M′ ← FindCoveringArray(k, P, F, 𝓕, N)
 8:         until M′ ≠ ∅                ▷ Stop after finding the first covering array
 9:         M ← M′                      ▷ Use this first covering array
10:     else
11:         repeat                      ▷ If we do find a covering array at first...
12:             N ← N − 1               ▷ we look for others with fewer tests
13:             M′ ← FindCoveringArray(k, P, F, 𝓕, N)
14:             if M′ ≠ ∅ then          ▷ If a smaller covering array is found...
15:                 M ← M′              ▷ we use that covering array instead
16:             end if
17:         until M′ = ∅                ▷ Stop when we cannot find covering arrays anymore
18:     end if
19:     return M
20: end function
```

valid covering array; or (ii) decrements N until we can no longer find a valid covering array. In addition, we may stop trying smaller number of tests when we reach a proven lower bound, such as in Lemma 3 and Theorems 2 and 3. It may seem unusual to increase or decrease the number of tests by 1 between steps rather than, for example, perform a binary search. However, we typically only need to try a few different number of tests before concluding the search.

Hence, the only part that changes between our exact and heuristic approaches is the implementation of function FindCoveringArray. This function must find a valid covering array for a given number of tests N. While our ultimate goal is to minimize the number of tests, allowing the number of tests to vary in the IP models would make them larger and more difficult to solve. Hence, we use the objective function of the IP models to heuristically direct the search in consistency with the traits that we have observed and conjectured to be common in most of the optimal covering arrays.

Now we consider what the function FindCoveringArray does in each case. We first present a simplified model coupled with backtracking for fast but heuristic results (Section 5.1). Then we present an exact model based on our theoretical results to determine if a valid covering array on N tests exists (Section 5.2) and discuss how it changes if we assume Conjecture 1 (Section 5.3). We also present a baseline model that does not depend on our theoretical results (Section 5.4).

5.1 Heuristic Approach H

We will first describe the heuristic approach, since it represents a simplified version of exact model presented afterwards.

$$\min\ kN\Delta + \sum_{i\in\mathcal{P}} \delta_i \tag{24a}$$

$$\text{s.t.}\ \ \delta_i \geq w_i - \lfloor N/2 \rfloor \qquad \forall i \in \mathcal{P} \tag{24b}$$

$$\delta_i \geq \lfloor N/2 \rfloor - w_i \qquad \forall i \in \mathcal{P} \tag{24c}$$

$$\Delta \geq \delta_i \qquad \forall i \in \mathcal{P} \tag{24d}$$

$$w_{i_1} + w_{i_2} \geq N+1 \qquad \forall (i_1,i_2) \in \mathcal{F} : \mathscr{F}(i_1,i_2) = (0,0) \tag{24e}$$

$$w_{i_1} - w_{i_2} \geq 1 \qquad \forall (i_1,i_2) \in \mathcal{F} : \mathscr{F}(i_1,i_2) = (0,1) \tag{24f}$$

$$w_{i_2} - w_{i_1} \geq 1 \qquad \forall (i_1,i_2) \in \mathcal{F} : \mathscr{F}(i_1,i_2) = (1,0) \tag{24g}$$

$$w_{i_1} + w_{i_2} \leq N-1 \qquad \forall (i_1,i_2) \in \mathcal{F} : \mathscr{F}(i_1,i_2) = (1,1) \tag{24h}$$

$$w_i \leq N-1 \qquad \forall i \in \mathcal{P} \tag{24i}$$

$$w_i \in \mathbb{Z}^+ \qquad \forall i \in \mathcal{P} \tag{24j}$$

$$\delta_i \in \mathbb{R}^+ \qquad \forall i \in \mathcal{P} \tag{24k}$$

$$\Delta \in \mathbb{R}^+ \tag{24l}$$

In the IP model (24a)–(24l), we only decide the number of ones w_i in the column used for each parameter $X_i, i \in \mathcal{P}$, leaving the actual choice of the column to

postprocessing. We capture with δ_i how far each parameter X_i is from having a balanced column in (24b–24c), and with Δ the largest of such values in (24d). We use those constraints to define lower bounds on δ_i and Δ, and then we use both in the objective function (24a). The objective function in this case is intended to make it easier to find a feasible solution when assigning columns to each parameter afterwards. First, we prioritize minimizing Δ first by effectively using kN as a sufficiently large constant. Second, we minimize the sum of the δ_i variables. We also ensure that the main decision variables satisfy Theorem 1 in (24e–24h). Moreover, we bound the number of ones to $N-1$ due to Assumption 1 in (24i). Finally, we may also derive other inequalities on variables w_i directly from Theorems 2 and 3 where applicable, but we omit those from the formulation for clarity.

Based on an optimal solution of the model above, we first tried to find columns with the prescribed number of ones by implementing a backtracking algorithm following best practices [46]. By assigning columns to the parameters with fewer options first, we were able to quickly obtain solutions to some instances—many of which backtracking-free and matching the optimal value from the exact approach. However, there were also instances in which the exact methods were successful while this approach timed out. In those, the solutions obtained by the exact methods had a different number of ones in the columns.

Hence, we engineered a more nuanced heuristic to exploit the structural properties discussed in the paper. Our goal was twofold: (i) for the instances that the exact methods could solve to optimality, we wanted to quickly produce solutions of similar quality; and (ii) for the other instances, we wanted to be competitive with the most popular heuristics in use. We briefly outline this approach below.

First, we only use the IP model (24a)–(24l) to determine if a covering array of size N exists. If not, we increment the value of N and repeat.

Second, we generate a few different solutions on the w variables, which we denote as *profiles*. Each profile defines a smaller search space to explore in order to find a feasible assignment of columns to the parameters, where the number of ones on the column associated with each parameter is fixed. Before generating the profiles, we use Theorems 1, 2, and 3 for tightening the domains of the w variables. We generate the first profile by trying to assign each variable in w to be as closed to balanced as possible, and then looping over the constraints entailed by Theorem 1 to correct violations by adjusting in one unit the value of the variable with largest domain. To obtain the first profile, this loop is repeated until there are no violations left. This gives us the first profile $w = \bar{w}$.

Then we derive additional profiles by adjusting the number of ones associated with the parameters with the most constraints on w entailed by Theorem 1. Always starting from the first profile $\bar{w}$, we generate separate profiles by adjusting both up and down the value in w for each of the C_1 most constrained parameters, as well as with all combinations of adjusting up and down the value in w for each pair among the C_2 most constrained parameters. In each of those $2C_1 + 4C_2$ new profiles, adjusting a value in w may entail changes in the value of w for other parameters due to Theorem 1 constraints, as well as to subsequent

corrections similar to those used for generating the first profile. We keep the top P resulting profiles after ranking them first by the largest slack with respect to the constraints defined by Theorem 1 (looser is better); second by the number of such constraints with zero slack (less is better); third by the sum of all slacks (larger is better); and then by how closed to balanced the values in w are.

Third, we search for a covering array of size N. We use the profiles above to search for an assignment of columns to the parameters by backtracking. We first search over the columns having w_i ones for each parameter i that is entailed by the top ranked profile w, and then we search over the columns with a number of ones in $\{w_i - 1, w_i, w_i + 1\}$ for each parameter i in each profile w that was kept. We try assigning columns to the constrained parameter with fewer options left first, and among the columns available we prioritize those that would keep more balanced columns available after propagation. While searching for such assignments, we ensure that there are enough balanced columns left for the unconstrained parameters. If we exhaust the profiles, we search over the entire space. If we do not succeed in T_N seconds, we increment N and repeat.

Finally, in case we find a covering array $\mathbf{M}$, say of size N, we may use it to find a covering array of size $N-1$ in two ways. The first is by generating profiles corresponding to removing each of the rows of $\mathbf{M}$, ranking them from most to least balanced, and then searching by backtracking on the first R_1 of them as above. The second is by removing each of the rows in order until obtaining R_2 distinct profiles, and then doing a backtrack-based search in which the column associated with each parameter cannot change by more than two elements. We limit the duration of each individual search to T_F seconds.

5.2 Exact Approach $\mathbf{M_0}$

We will now describe the exact approach, which modifies and extends the IP model previously described to decide on its own if a valid covering array can be found or not, hence without nontrivial postprocessing in comparison to before.

To make the next model easier to understand and maintain consistency with prior statements and models, we reserve the use of the following indices and single-letter variables to specific purposes, to the extent that it is possible:

- α, β for the values zero and one associated with parameters;
- c for columns;
- i, i_1, i_2 for parameter indices;
- j, j_1, j_2 for the number of ones in columns;
- k for the number of parameters;
- ℓ for other occasional summations; and
- N for number of rows.

The additional elements below are used in this IP model:

- the set of unconstrained parameters $\tilde{\mathcal{P}} \subseteq \mathcal{P}$;
- the set of pairs of parameters without a pairwise constraint $\overline{\mathcal{F}} \subseteq \mathcal{P} \times \mathcal{P}$;

- the set of allowable number of ones[10] $\mathcal{O}_i \subseteq \{1, \ldots, N-1\}$ in a column that can be assigned to parameter $X_i, i \in \mathcal{P}$;
- the set of all binary columns $\mathcal{C} := \{0,1\}^k$, and some of its subsets:
  - the columns with j ones $\mathcal{C}^j := \{c \in \mathcal{C} : \sum_{\ell=1}^{k} c_\ell = j\}$,
  - the canonical balanced columns with the value one in the first row $\mathcal{B} := \{c \in \mathcal{C}^{\lfloor N/2 \rfloor} : c_1 = 1\}$, and
  - the allowable columns[11] $\mathcal{C}_i := \{c : c \in \mathcal{C}^j, j \in \mathcal{O}_i\}$ for parameter $X_i, i \in \mathcal{P}$;
- the indices of parameters[12] $\mathcal{P}_c \subset \mathcal{P}$ that are compatible with column $c \in \mathcal{C}$;
- the set of incompatible pairs of columns $\mathcal{I} \subset \mathcal{C}^2$ because at least one pairwise assignment is not covered; and
- the set of incompatible pairs of columns $\mathcal{I}^{(\alpha,\beta)} \subset \mathcal{C}^2$ because at least one pairwise assignment other than (α, β) is not covered.

In the IP model (25a)–(25p), our main decision is what column $c \in \mathcal{C}$ is uniquely assigned to each parameter $X_i, i \in \mathcal{P}$ through the binary variable V_{ic}. Each parameter is assigned a column by (25b), and each column is assigned to at most one parameter by (25c).

Instead of a single variable w_i for the number of ones in the column associated with X_i as before, we use the binary variable W_{ij} to more easily capture each possible number of ones j for the column assigned to parameter X_i. Each parameter has a number of ones for the column assigned to it by (25d), and the columns can only be assigned if they have that corresponding number of ones due to (25e). Moreover, those variables make it easier to capture the inequalities from Theorem 1 for each possible number of ones with (25f–25i).

For each pair of parameters, we define inequalities avoiding the assignment of incompatible columns when there is a pairwise constraint between them with (25j) and when there is not with (25k), since the columns that would cover all and only the allowable pairwise assignments in each case would be different.

To exploit the fact that canonical balanced columns are all mutually compatible, we use an additional decision variable U_c denoting if a balanced column is used for an unconstrained parameter X_i. That way, we can subject $V_{ic} = 1$ to when $U_c = 1$ by (25l); and then limit the use of such a column, regardless of which parameter uses it, if a conflicting column is assigned to a constrained parameter with (25m). Please note that this is not yet exploiting Conjecture 1, since an unbalanced column may still be assigned to an unconstrained parameter. However, that makes that case easier to discuss in Section 5.3.

Finally, the objective function (25a) is intended to guide the solver toward a feasible solution: we aim to maximize the number of canonical balanced columns used, since they are automatically all compatible with one another, by penalizing

[10] This preprocessing is not necessary, but can reduce domains considerably. We find those sets based on which parameters must have more or less ones in their columns than other parameters, leading to stricter bounds on their number of ones given N.

[11] Also obtained by preprocessing to reduce domains.

[12] This is the converse of the previous preprocessing producing $\mathcal{C}_i$.

any other choice of column for a parameter and even more so if the assigned column has a greater absolute difference between the number of zeros and ones.

$$\min \sum_{i\in\mathcal{P}} \sum_{j\in\mathcal{O}_i} \sum_{c\in\mathcal{C}^j\setminus\mathcal{B}} (|j - \lfloor N/2 \rfloor| + 1)\, V_{ic} \tag{25a}$$

$$\text{s.t.} \sum_{c\in\mathcal{C}_i} V_{ic} = 1 \qquad \forall i \in \mathcal{P} \tag{25b}$$

$$\sum_{i\in\mathcal{P}_c} V_{ic} \leq 1 \qquad \forall c \in \mathcal{C} \tag{25c}$$

$$\sum_{j\in\mathcal{O}_i} W_{ij} = 1 \qquad \forall i \in \mathcal{P} \tag{25d}$$

$$V_{ic} \leq W_{ij} \qquad \forall i \in \mathcal{P}, j \in \mathcal{O}_i, c \in \mathcal{C}^j \tag{25e}$$

$$W_{i_1 j_1} + \sum_{j_2\in\mathcal{O}_{i_2}: j_2 \leq N - j_1} W_{i_2 j_2} \leq 1 \qquad \forall (i_1, i_2) \in \mathcal{F} : \mathscr{F}(i_1, i_2) = (0, 0) \tag{25f}$$

$$W_{i_1 j_1} + \sum_{j_2\in\mathcal{O}_{i_2}: j_2 \geq j_1} W_{i_2 j_2} \leq 1 \qquad \forall (i_1, i_2) \in \mathcal{F} : \mathscr{F}(i_1, i_2) = (0, 1) \tag{25g}$$

$$W_{i_1 j_1} + \sum_{j_2\in\mathcal{O}_{i_2}: j_2 \leq j_1} W_{i_2 j_2} \leq 1 \qquad \forall (i_1, i_2) \in \mathcal{F} : \mathscr{F}(i_1, i_2) = (1, 0) \tag{25h}$$

$$W_{i_1 j_1} + \sum_{j_2\in\mathcal{O}_{i_2}: j_2 \geq N - j_1} W_{i_2 j_2} \leq 1 \qquad \forall (i_1, i_2) \in \mathcal{F} : \mathscr{F}(i_1, i_2) = (1, 1) \tag{25i}$$

$$V_{i_1 c_1} + V_{i_2 c_2} \leq 1 \qquad \forall (i_1, i_2) \in \mathcal{F}, (c_1, c_2) \in \mathcal{I}^{\mathscr{F}(i_1, i_2)} \tag{25j}$$

$$V_{i_1 c_1} + V_{i_2 c_2} \leq 1 \qquad \forall (i_1, i_2) \in \overline{\mathcal{F}}, (c_1, c_2) \in \mathcal{I} \tag{25k}$$

$$V_{ic} \leq U_c \qquad \forall i \in \tilde{\mathcal{P}}, c \in \mathcal{C}_i \cap \mathcal{B} \tag{25l}$$

$$V_{ic} + U_{c'} \leq 1 \qquad \forall i \in \mathcal{P}, c \in \mathcal{C}_i, c' \in \mathcal{B}, (c, c') \in \mathcal{I} \tag{25m}$$

$$V_{ic} \in \{0, 1\} \qquad \forall i \in \mathcal{P}, c \in \mathcal{C}_i \tag{25n}$$

$$W_{ij} \in \{0, 1\} \qquad \forall i \in \mathcal{P}, j \in \mathcal{O}_i \tag{25o}$$

$$U_c \in \{0, 1\} \qquad \forall c \in \mathcal{B} \tag{25p}$$

As before, we can also derive valid inequalities from Theorems 2 and 3 to strengthen the formulation. They are not strictly necessary, but they certainly make it easier to find a valid covering array when the conditions of those theorems hold. Hence, we also omit them from the formulation for clarity.

In practice, we can stop solving the model when a feasible solution is found, since the goal of each model is to find a feasible solution for a given N. Moreover, we can reduce the model size significantly by generating the constraints

(25f–25m) through lazy callback by inspecting potentially feasible solutions. In other words, we inspect the covering arrays associated with solutions of the IP formulation without those constraints, and then dynamically add the constraints that would make those solutions infeasible if the covering array is not valid.

5.3 Conjectured Exact Approach $\mathbf{M_1}$

If we assume Conjecture 1, then we use only balanced columns for every unconstrained parameter $X_i, i \in \tilde{\mathcal{P}}$, which reduces the number of choices considerably.

One simple way to produce that modification would be to restrict the compatibility of unconstrained parameters to only canonical balanced columns in the preprocessing, hence not modifying the previous formulation at all.

5.4 Baseline Exact Approach $\mathbf{M_B}$

Now we present a baseline model in which we model the assignments in the covering array as decision variables. The results previously discussed are not necessary for this model, except for Assumption 3: we state that every pair of assignments that is not forbidden should be covered in at least one row, which thus renders the model infeasible if an implicit pairwise constraint exists.

In this model, we use the additional index and elements:

- r for the number of the row in the covering array;
- the set of rows $\mathcal{R} := \{1, \ldots, N^{\max}\}$, where $N^{\max}$ is chosen in advance.

In the IP model (26a)–(26l), we represent the covering array assignment directly with the binary variable M_{ri} denoting the value of parameter i (column i) in test r (row r). In addition, we use the binary variable $Y_{ri_1i_2\alpha\beta}$ to denote if test r covers the pairwise assignment $(X_{i_1}, X_{i_2}) = (\alpha, \beta)$; and the binary variable Z_r to denote if row r is used in the covering array. If a row is not used, we skip that row when producing the covering array from the solutions obtained.

We relate the Y variables with a value of one to the corresponding assignment on the M variables with (26b)–(26c). We ensure that at least one row covers each allowed pairwise assignment on the Y variables with (26d) when there is no pairwise constraint between the parameters, and with (26e) when there is a pairwise assignment between the parameters. We ensure that no row covers a forbidden pairwise assignment with (26f) on the Y variables; and with (26g)–(26h) on the M variables. Finally, we limit the value of one on the Y variables when the value on the corresponding Z variable is one with (26i).

The objective function (26a) then induces an optimal solution corresponding to an optimal covering array, provided that we choose a sufficiently large value for $N^{\max}$ in advance. However, using a much larger for $N^{\max}$ than needed may result in a much larger model, which may not be solved as fast.

This model has many forms of symmetry. However, we could not obtain any significant improvement in preliminary experiments by exploiting some of those in the unconstrained case. Nevertheless, we believe that this model is an earnest

alternative if we were to ignore the developments proposed in this paper, given that the authors formulated and relied on it as part of their initial investigation.

$$\min \sum_{r \in \mathcal{R}} Z_r \tag{26a}$$

$$\text{s.t. } (Y_{ri_1i_2\alpha\beta} = 1) \rightarrow (M_{ki_1} = \alpha) \qquad \forall r \in \mathcal{R}, (i_1, i_2) \in \overline{\mathcal{F}}, (\alpha, \beta) \in \{0,1\} \times \{0,1\} \tag{26b}$$

$$(Y_{ri_1i_2\alpha\beta} = 1) \rightarrow (M_{ki_2} = \beta) \qquad \forall r \in \mathcal{R}, (i_1, i_2) \in \overline{\mathcal{F}}, (\alpha, \beta) \in \{0,1\} \times \{0,1\} \tag{26c}$$

$$\sum_{r \in \mathcal{R}} Y_{ri_1i_2\alpha\beta} \geq 1 \qquad \forall (i_1, i_2) \in \overline{\mathcal{F}}, (\alpha, \beta) \in \{0,1\} \times \{0,1\} \tag{26d}$$

$$\sum_{r \in \mathcal{R}} Y_{ri_1i_2\alpha\beta} \geq 1 \qquad \forall (i_1, i_2) \in \mathcal{F}, \mathscr{F}(i_1, i_2) \neq (\alpha, \beta), (\alpha, \beta) \in \{0,1\} \times \{0,1\} \tag{26e}$$

$$\sum_{r \in \mathcal{R}} Y_{ri_1i_2\alpha\beta} = 0 \qquad \forall (i_1, i_2) \in \mathcal{F}, \mathscr{F}(i_1, i_2) = (\alpha, \beta), (\alpha, \beta) \in \{0,1\} \times \{0,1\} \tag{26f}$$

$$(M_{ki_1} = \alpha) \rightarrow (M_{ki_1} = 1 - \beta) \qquad \forall (i_1, i_2) \in \mathcal{F}, \mathscr{F}(i_1, i_2) = (\alpha, \beta), (\alpha, \beta) \in \{0,1\} \times \{0,1\} \tag{26g}$$

$$(M_{ki_2} = \beta) \rightarrow (M_{ki_1} = 1 - \alpha) \qquad \forall (i_1, i_2) \in \mathcal{F}, \mathscr{F}(i_1, i_2) = (\alpha, \beta), (\alpha, \beta) \in \{0,1\} \times \{0,1\} \tag{26h}$$

$$Y_{ri_1i_2\alpha\beta} \leq Z_r \qquad \forall r \in \mathcal{R}, (i_1, i_2) \in \overline{\mathcal{F}}, (\alpha, \beta) \in \{0,1\} \times \{0,1\} \tag{26i}$$

$$M_{ri} \in \{0,1\} \qquad \forall r \in \mathcal{R}, i \in \mathcal{P} \tag{26j}$$

$$Y_{ri_1i_2\alpha\beta} \in \{0,1\} \qquad \forall r \in \mathcal{R}, (i_1, i_2) \in \overline{\mathcal{F}}, (\alpha, \beta) \in \{0,1\} \times \{0,1\} \tag{26k}$$

$$Z_r \in \{0,1\} \qquad \forall r \in \mathcal{R} \tag{26l}$$

6 Computational Experiments

We designed our computational experiments to evaluate the four approaches described: the exact method M_0, the conjectured exact method M_1, the baseline exact method M_B, and the heuristic method H. We also compare some of those approaches with commonly used heuristics from the literature by using two solvers, the Automated Combinatorial Testing for Software (ACTS) 3.2 solver [41] and the Constrained Covering Array Generation (CCAG) solver [21]. ACTS was chosen because of its historical importance and continuous development. CCAG was chosen because it is a framework implementing and comparing the most popular heuristics in its accompanying survey [50]. We refer to ACTS as one of the heuristics tested. We also test heuristics AETG, DDA, PSO, SA, and TS from CCAG along with their best-performed handlers identified in [50].

6.1 Implementation Details

We generated 300 random binary instances by combining $|\mathcal{P}| \in \{10, 20, 30, 40, 50\}$ parameters with $|\mathcal{F}| \in \{1, 5, 10, 20, 40, 80\}$ forbidden pairs, using 10 random seeds for each setting. All methods were given a 30-minute limit per instance, except for ACTS because it is preconfigured and runs to completion. For the CCAG heuristics, we record the best solution obtained within this budget and the first time in which such a solution was obtained, and the number of iterations is chosen to exceed the time limit and the solver is interrupted at the time limit. For heuristic H, we used the following settings: $C_1 = 6$, $C_2 = 4$, $P = 9$, $T_N = 180$, $R_1 = 8$, $R_2 = 8$, and $T_F = 60$.

All experiments were run on identical Red Hat Enterprise Linux 9.6 nodes with Intel Xeon Gold 6226 (Cascade Lake) processors at 2.70 GHz, with each run confined to a single core and 32 GB of RAM. The methods were implemented in Python 3.9.21 and MILP models were solved with the Gurobi 12.0.1 solver.

6.2 Results and Analysis

We collected data and visualized the results with a few questions in mind. First, we evaluated the exact approaches to answer the following questions:

(i) *How does the proposed approach M_0 compare with the baseline M_B?*
(ii) *Is there a refutation of Conjecture 1 in the results with approach M_1?*
(iii) *If not and assuming Conjecture 1, how does M_1 compare with M_0?*

Figure 1 shows a cumulative plot of the proportion of instances solved by each exact method, including both when an optimal solution is found and when the instance is proven infeasible, in logarithmic and linear time scale. We note that the visible uptick of instances solved within a hundredth of a second by both M_0 and M_1 correspond to cases in which preprocessing has proven infeasibility.

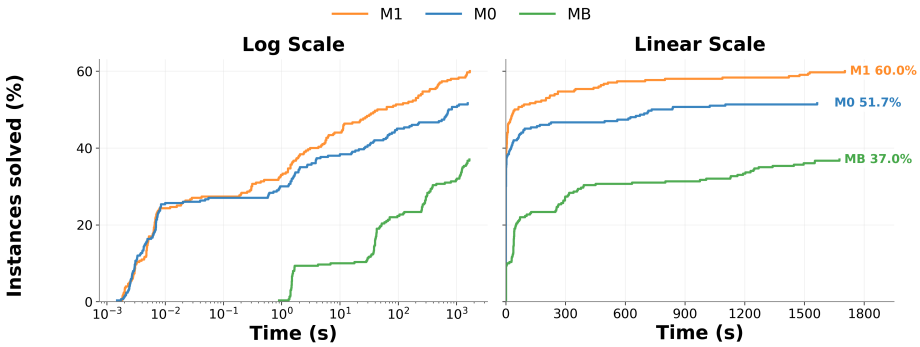

Fig. 1: Cumulative ratio of instances solved by each exact method over time, in logarithmic time scale (left) and linear time scale (right).

In total, 76 of the 300 instances are proven infeasible by preprocessing. That includes all the instances with 80 pairwise constraints.

Figure 2 compares the solution value (number of rows) of the covering arrays obtained by exact methods for each instance. In this and other scatter plots, we exclude infeasible instances, represent timeout results with a large value labeled as T.O., and use larger markers with a number inside when multiple instances produce the same pair of values. If an optimal solution is found by both methods, we expected—and indeed observed—that the solution values match. Hence, the only differences are in the timeouts summarized at the top left of each plot.

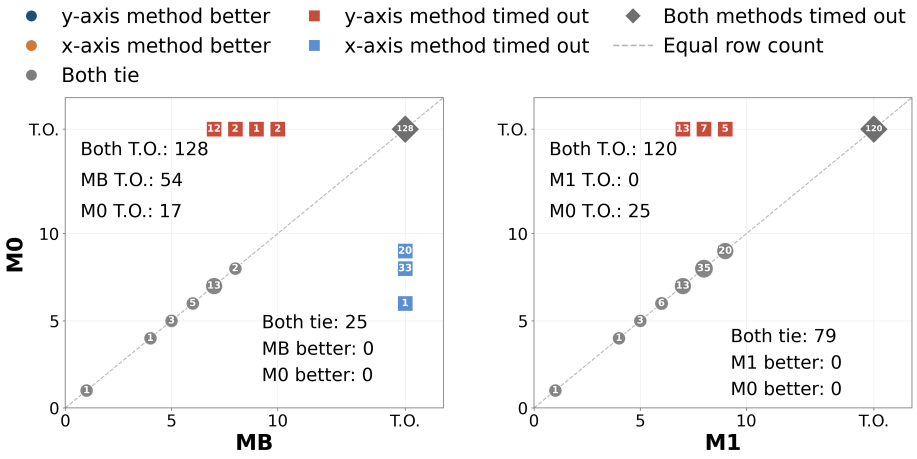

Fig. 2: Pairwise comparison of solution values (numbers of rows) and timeouts for exact methods, contrasting M_B with M_0 (left) and M_1 with M_0 (right).

Figure 3 compares the time taken by exact methods to solve each instance, allowing us to evaluate which method solved each instance faster. Those cases are summarized at the bottom right of each plot.

Answering (i) We observe that M_0 has solved more instances than M_B at any given time (Figure 1), although both methods solved instances that the other did not (Figure 2 left). Among those, we observe instances solved up to a minute by M_B being solved faster by M_0; whereas the instances in which M_0 times out take at least 100 seconds to be solved by M_B, while most of the instances in which M_B times out take at least 1 second to be solved by M_0 (Figure 3 left). Hence, M_0 generally solves more instances and faster than M_B.

Answering (ii) There is no instance solved by both M_0 and M_1 in which the optimal solution value is not the same (Figure 2 right). Hence, we have not observed a contradiction with Conjecture 1, which remains unrefuted.

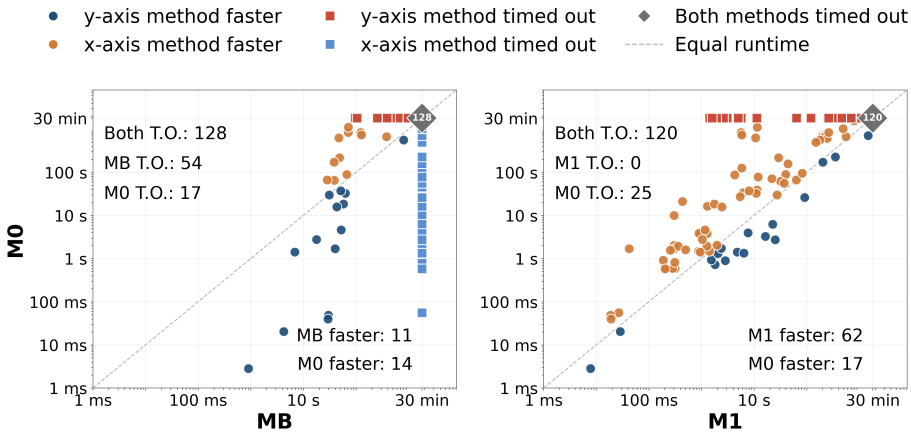

Fig. 3: Pairwise comparison of runtimes and timeouts for exact methods, contrasting M_B with M_0 (left) and M_1 with M_0 (right).

Answering (iii) We observe that M_1 similarly has solved more instances than M_0 at any given time (Figure 1), and in fact the instances solved by M_0 are a proper subset of those solved by M_1 (Figure 2 right). Among the instances solved by both, the majority was solved faster by M_1, sometimes with orders-of-magnitude speedups, whereas the instances solved faster by M_0 were within about an order-of-magnitude speedup (Figure 3 right). Hence, M_1 performs better than M_0, solving the same and even more instances, and generally faster.

Second, we evaluated our heuristic approach H with the following questions:

(iv) *How does H compare with the exact approaches M_0 and M_1?*
(v) *How does H compare with with other commonly used heuristics?*
(vi) *Is there a refutation of Conjecture 1 in the results with any heuristic?*

We use similar scatter plots as before to compare H with the exact methods M_0 and M_1: Figure 4 compares the solution value (number of rows) of the covering arrays obtained and Figure 5 compares the runtimes involved.

Figure 6 compares the solution value (number of rows) of the covering arrays obtained by heuristic methods, comparing H with each other method. We conclude from these plots that SA and TS are the best performing heuristics besides H, for which reason we restrict the subsequent analysis to only those.

In the case of ACTS, we note that a covering array is also produced for the 76 instances that were proven to be infeasible in our preprocessing. Since infeasible solutions is not included in the scatter plots, that did not affect the comparisons. In the case of the heuristics within CCAG (AETG, DDA, PSO, SA, and TS), the plots would have indicated 76 timeouts if those instances were included.

Figure 7 compares the time taken by H to solve each instance with the time that it took SA and TS to obtain their best solution value for the first time.

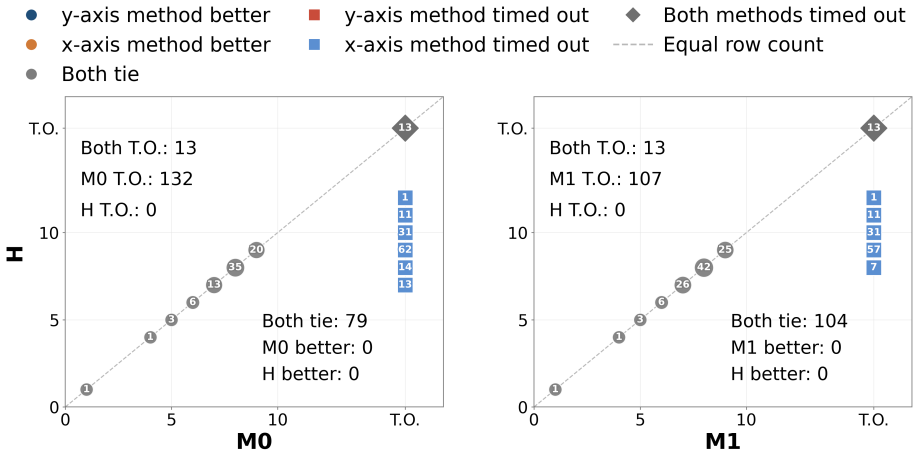

Fig. 4: Pairwise comparison of solution values (numbers of rows) and timeouts, contrasting exact methods M_0 (left) and M_1 (right) with heuristic H.

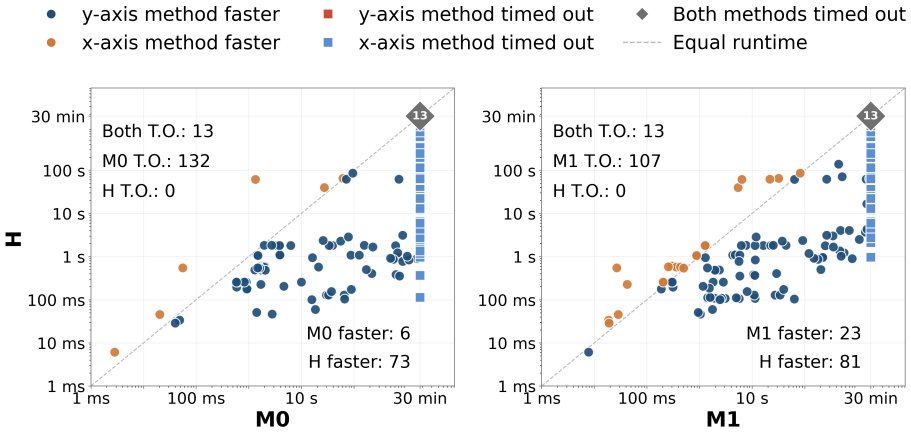

Fig. 5: Pairwise comparison of runtimes and timeouts, contrasting exact methods M_0 (left) and M_1 (right) with heuristic H.

Answering (iv) We note that H matches the solution value (number of rows) of all the solutions obtained with M_0 and M_1, more than doubling the total number of instances solved with respect to each (Figure 4). Among the instances that each exact method also solved, a majority is solved at least one order-of-magnitude faster by H (Figure 5). Hence, H has solved twice as many instances as M_0 and M_1, and the solutions obtained for instances also solved by M_0 or M_1 were generally solved faster and all happened to be optimal—although H is an heuristic and thus bound to occasionally produce suboptimal solutions.

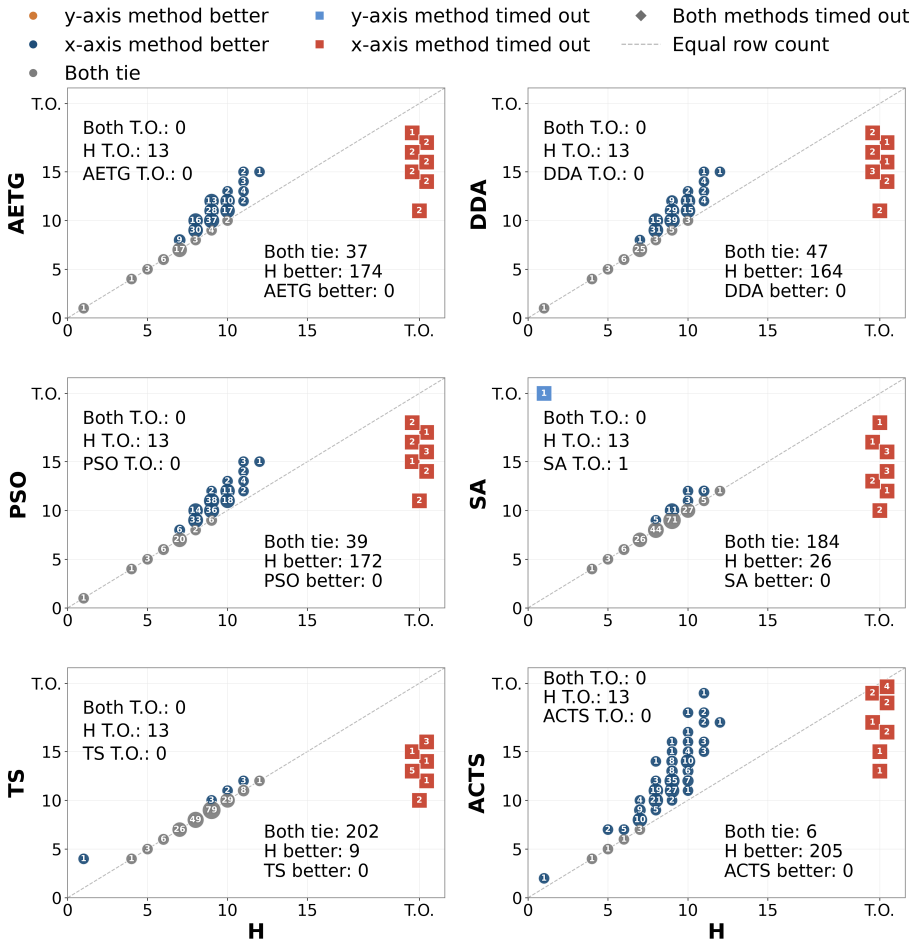

Fig. 6: Pairwise comparison of solution values (numbers of rows) and timeouts, contrasting heuristic H with heuristics AETG, DDA, PSO, SA, TS, and ACTS.

Answering (v) We note that H produced solutions with as many or fewer rows than the other heuristics; although it occasionally timed out without a solution, and the same happened but once with another heuristic if we ignore the 76 timeouts due to infeasibility proven by preprocessing (Figure 6). Compared with the best-performing competing heuristics SA and TS in the cases in which the instances are solved by H and each of those, H solves almost half of the instances faster in each case (Figure 7). Hence, balancing time outs and solution values in feasible instances, H is competitive with SA and TS, narrowly better handling more instances than SA (26 better solutions vs. 13 against 1 timeouts) and fewer instances than TS (9 better solutions vs. 13 timeouts); whereas SA and TS timeout in the 76 instances that H identifies as infeasible.

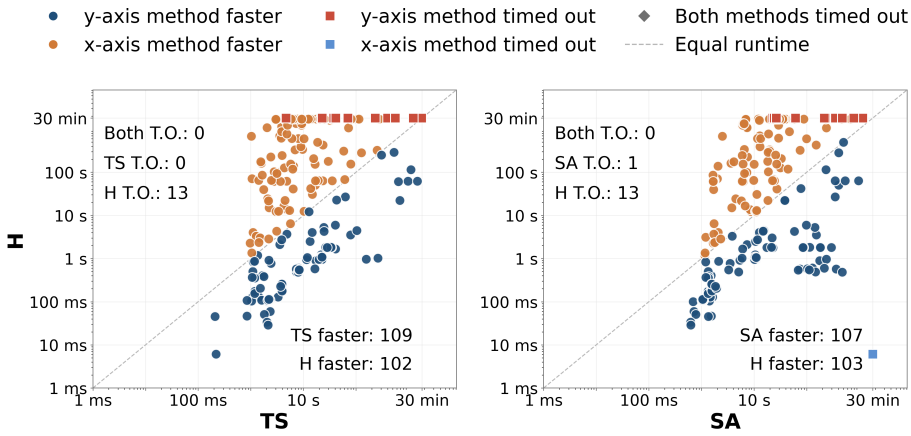

Fig. 7: Pairwise comparison of time for finding the best solution and timeouts, contrasting heuristic H with the best performing heuristics SA and TS.

Answering (vi) There are no instances solved by M_1 for which a better solution is found by heuristic H (Figure 4). Since H solved all the instances also solved by M_1 (Figure 4) and no other heuristic produced a better solution than H in the cases in which *H* produced a solution (Figure 6), then by consequence the previous statement applies to the other heuristics tested. Hence, we continue not having observed a contradiction with Conjecture 1, which remains unrefuted.

Finally, we study in more detail the instances used with respect to number of parameters and pairwise constraints to answer the following questions:

(vii) *How often each type of instance is solved to optimality or proven infeasible?*
(viii) *How does the optimal solutions compare with the unconstrained case?*

We use two new types of plots in this part. Figure 8 distributes the instances with respect to the solutions obtained with M_1 among optimal, timed out, and proved infeasible within each number of parameters and pairwise constraints. Within each such combination of number of parameters and pairwise constraints, Figure 9 distributes the instances with respect to the value of the optimal solution (number of rows) in comparison to the unconstrained case.

Answering (vii) Increasing the number of pairwise constraints shifts the most common outcome from optimal to proved infeasible, whereas increasing the number of parameters creates and widens a gap between those cases in which the most common outcome is timeout (Figure 8).

Answering (viii) Adding from one to ten random constraints generally increases the number of rows in the optimal solutions when compared to the unconstrained case; with twenty constraints or more, we observe both increases and decreases (Figure 9). Although we have observed two pairwise constraints being enough

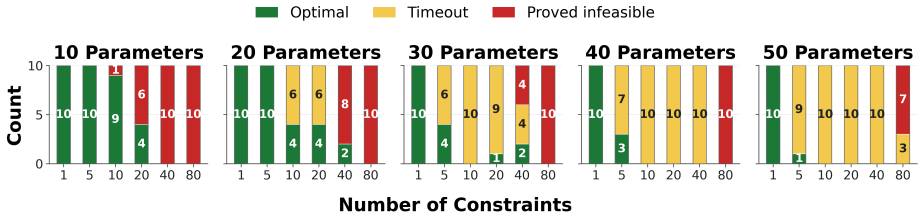

Fig. 8: Instances solved to optimality, timed out, and proven infeasible by M_1 for each number of parameters and pairwise constraints.

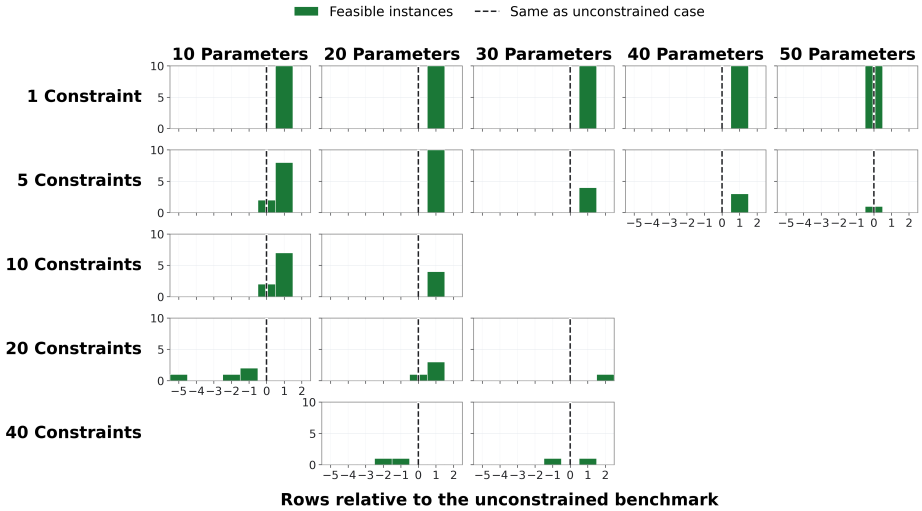

Fig. 9: Difference in optimal solution value (row count) with respect to the unconstrained case for instances solved to optimality by M_1 per each number of parameters and pairwise constraints. Empty cells have no feasible instances.

for the optimal covering array to have fewer rows (see discussion about $\mathbf{M}^{(8)}$ at the beginning of Section 4), we note that those constraints were not random.

7 Conclusion

In this paper, we have studied how to solve the coverage problem in combinatorial testing for binary parameters and subject to pairwise constraints. First, we generalized results on the structure of optimal solutions by showing that:

(i) in the absence of constraints, the constructive algorithm widely used in the literature may in some cases generate all optimal solutions; whereas
(ii) constrained testing may lead to comparatively more tests, but also occasionally to paradoxically fewer tests—even among binary parameters;

(iii) constraints on subsets that should otherwise be covered fundamentally change the structure of testing suites; but, perhaps more importantly,
(iv) the key to finding optimal solutions remains to be on balancing the number of zeros and ones per parameter, as much as still possible.

In fact, the prevalence of balanced columns has been empirically observed in unconstrained optimal solutions [2]; and in a sense relates to the alternative—but less flexible—use of orthogonal arrays [48], in which all pairs of assignments must appear the same number of times.

Second, we showed (a) how to obtain optimal testing suites by solving a series of Integer Linear Programming (ILP) models; and (b) how to obtain similar solutions faster with an heuristic that is competitive with commonly used techniques in quality and runtime. To the best of our knowledge, this is the first non-enumerative approach to obtain the optimal solution in the setting of binary parameters subject to pairwise constraints.

In future work, we hope to develop new theoretical results on the minimum number of tests and on the feasibility conditions for covering arrays, similar to those in Sections 4.2 and 4.3, and explore more efficient approaches to solve the covering problem to optimality.

Acknowledgments. Changkun Guan was supported by Bucknell University's PP&L Undergraduate Research Fund. Hunter Gehman was supported by Bucknell University's Freeman College of Management Gift Funds. Hunter Gehman and Mikey Ferguson were supported by Bucknell University's Presidential Fellowship.

References

1. Ansótegui, C., Manyà, F., Ojeda, J., Salvia, J.M., Torres, E.: Incomplete MaxSAT approaches for combinatorial testing. Journal of Heuristics **28**(4), 377–431 (Aug 2022)
2. Bracho-Rios, J., Torres-Jimenez, J., Rodriguez-Tello, E.: A new backtracking algorithm for constructing binary covering arrays of variable strength. In: Aguirre, A.H., Borja, R.M., Garciá, C.A.R. (eds.) MICAI 2009: Advances in Artificial Intelligence. pp. 397–407. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
3. Bryce, R.C., Colbourn, C.J.: The density algorithm for pairwise interaction testing. Software Testing **17**(3) (2007), https://doi.org/10.1002/stvr.365
4. Bryce, R.C., Colbourn, C.J.: Prioritized interaction testing for pair-wise coverage with seeding and constraints. Information and Software Technology **48**(10), 960–970 (2006), Advances in Model-based Testing
5. Bui-Xuan, B.M., Magnien, C., Meyer, P., Phan, T.H.D.: Link stream edition: Sparse split and bi-sparse split (2019), https://math.ac.vn/uploads/files/IMH20191102.pdf
6. Cawse, J.N.: Experimental Design for Combinatorial and High Throughput Materials Development. John Wiley & Sons, New York (2003)
7. Cohen, D.M., Dalal, S.R., Fredman, M.L., Patton, G.C.: The AETG system: An approach to testing based on combinatorial design. IEEE Trans. Software Eng. **23**, 437–444 (1997). https://doi.org/https://doi.org/10.1109/32.605761

8. Cohen, D., Dalal, S., Parelius, J., Patton, G.: The combinatorial design approach to automatic test generation. IEEE Software **13**(5), 83–88 (1996). https://doi.org/10.1109/52.536462
9. Cohen, M., Gibbons, P., Mugridge, W., Colbourn, C.: Constructing test suites for interaction testing. In: 25th International Conference on Software Engineering, 2003. Proceedings. pp. 38–48 (2003). https://doi.org/10.1109/ICSE.2003.1201186
10. Cohen, M.B., Dwyer, M.B., Shi, J.: Interaction testing of highly-configurable systems in the presence of constraints. In: Proceedings of the 2007 International Symposium on Software Testing and Analysis. p. 129–139. ISSTA '07, Association for Computing Machinery, New York, NY, USA (2007). https://doi.org/10.1145/1273463.1273482
11. Cohen, M.B., Dwyer, M.B., Shi, J.: Constructing interaction test suites for highly-configurable systems in the presence of constraints: A greedy approach. IEEE Transactions on Software Engineering **34**(5), 633–650 (2008). https://doi.org/10.1109/TSE.2008.50
12. Colbourn, C.J.: Combinatorial aspects of covering arrays. Le Matematiche **59**(1–2), 125–172 (2004)
13. Czerwonka, J.: Pairwise testing in real world. https://github.com/microsoft/pict/blob/main/doc/Pairwise%20Testing%20in%20Real%20World.pdf (2006)
14. Danziger, P., Mendelsohn, E., Moura, L., Stevens, B.: Covering arrays avoiding forbidden edges. Theoretical Computer Science **410**(52), 5403–5414 (2009)
15. Duan, F., Lei, Y., Yu, L., Kacker, R.N., Kuhn, D.R.: Optimizing IPOG's vertical growth with constraints based on hypergraph coloring. In: 2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). pp. 181–188 (2017). https://doi.org/10.1109/ICSTW.2017.37
16. Dumer, I.: Asymptotically optimal codes correcting memory defects of fixed multiplicity. Problemy Peredachi Informatskii **25**, 3 (10 1989)
17. Ferrer, J., Chicano, F., Alba, E.: Hybrid algorithms based on integer programming for the search of prioritized test data in software product lines. In: Squillero, G., Sim, K. (eds.) Applications of Evolutionary Computation. pp. 3–19. Springer International Publishing, Cham (2017)
18. Ferrer, J., Chicano, F., Ortega-Toro, J.A.: CMSA algorithm for solving the prioritized pairwise test data generation problem in software product lines. Journal of Heuristics **27**(1), 229–249 (2021). https://doi.org/10.1007/s10732-020-09462-w
19. Flores, P., Cheon, Y.: PWiseGen: Generating test cases for pairwise testing using genetic algorithms. In: 2011 IEEE International Conference on Computer Science and Automation Engineering. vol. 2, pp. 747–752 (2011). https://doi.org/10.1109/CSAE.2011.5952610
20. Grindal, M., Offutt, J., Andler, S.F.: Combination testing strategies: a survey. Software Testing, Verification and Reliability **15**(3) (2005), https://doi.org/10.1002/stvr.319
21. Group in Intelligent Software Technology at Nanjing University: CCAG: A reference implementation of constrained covering array generation (2021), https://github.com/GIST-NJU/CCAG, accessed: 2026-07-14
22. Hagar, J.D., Wissink, T.L., Kuhn, D.R., Kacker, R.N.: Introducing combinatorial testing in a large organization. Computer **48**(4), 64–72 (2015). https://doi.org/10.1109/MC.2015.114

23. Hartman, A., Raskin, L.: Problems and algorithms for covering arrays. Discrete Mathematics **284**(1), 149–156 (2004). https://doi.org/10.1016/j.disc.2003.11.029
24. Hnich, B., Prestwich, S.D., Selensky, E., Smith, B.M.: Constraint models for the covering test problem. Constraints **11**, 119–219 (2006)
25. Kadioglu, S.: Column generation for interaction coverage in combinatorial software testing (2017), https://arxiv.org/abs/1712.07081
26. Katona, G.O.H.: Two applications (for search theory and truth functions) of Sperner type theorems. Periodica Mathematica Hungarica **3**, 19–26 (1973)
27. Kleitman, D.J., Spencer, J.: Families of k-independent sets. Discrete Mathematics **6**(3), 255–262 (1973)
28. Kuhn, D.R., Okun, V.: Pseudo-exhaustive testing for software. In: 2006 30th Annual IEEE/NASA Software Engineering Workshop. pp. 153–158 (2006). https://doi.org/10.1109/SEW.2006.26
29. Kuhn, D., Kacker, R., Lei, Y.: Practical combinatorial testing. Tech. Rep. NIST SP 800-142, National Institute of Standards and Technology, Gaithersburg, MD (10 2010). https://doi.org/10.6028/NIST.SP.800-142
30. Kuhn, D., Reilly, M.: An investigation of the applicability of design of experiments to software testing. In: 27th Annual NASA Goddard/IEEE Software Engineering Workshop, 2002. Proceedings. pp. 91–95 (2002)
31. Kuhn, D., Wallace, D., Gallo, A.: Software fault interactions and implications for software testing. IEEE Transactions on Software Engineering **30**(6), 418–421 (2004). https://doi.org/10.1109/TSE.2004.24
32. Lawrence, J., Kacker, R.N., Lei, Y., Kuhn, D.R., Forbes, M.: A survey of binary covering arrays. The Electronic Journal of Combinatorics **18**(1), 1–30 (2011)
33. Lei, Y., Kacker, R., Kuhn, D.R., Okun, V., Lawrence, J.: IPOG: A general strategy for t-way software testing. In: 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07). pp. 549–556 (March 2007). https://doi.org/10.1109/ECBS.2007.47
34. Lei, Y., Kacker, R., Kuhn, D.R., Okun, V., Lawrence, J.: IPOG/IPOG-D: Efficient test generation for multi-way combinatorial testing. Software Testing, Verification and Reliability **18**(3), 125–148 (2008). https://doi.org/10.1002/stvr.381
35. Lei, Y., Tai, K.: In-parameter-order: a test generation strategy for pairwise testing. In: Proceedings Third IEEE International High-Assurance Systems Engineering Symposium (Cat. No.98EX231). pp. 254–261 (1998). https://doi.org/10.1109/HASE.1998.731623
36. Lopez-Herrejon, R.E., Chicano, F., Ferrer, J., Egyed, A., Alba, E.: Multi-objective optimal test suite computation for software product line pairwise testing. In: 2013 IEEE International Conference on Software Maintenance. pp. 404–407 (2013). https://doi.org/10.1109/ICSM.2013.58
37. Luo, C., Lin, J., Cai, S., Chen, X., He, B., Qiao, B., Zhao, P., Lin, Q., Zhang, H., Wu, W., Rajmohan, S., Zhang, D.: AutoCCAG: An automated approach to constrained covering array generation. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). pp. 201–212 (2021). https://doi.org/10.1109/ICSE43902.2021.00030
38. Ma, L., Zhang, F., Xue, M., Li, B., Liu, Y., Zhao, J., Wang, Y.: Combinatorial testing for deep learning systems (2018), https://arxiv.org/abs/1806.07723
39. Maltais, E., Moura, L.: Finding the best cafe is np-hard. In: López-Ortiz, A. (ed.) LATIN 2010: Theoretical Informatics. pp. 356–371. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)

40. National Institute of Standards and Technology: Industrial case studies - combinatorial and pairwise testing. https://csrc.nist.gov/projects/automated-combinatorial-testing-for-software/combinatorial-methods-in-testing/case-studies-and-examples (2016), accessed: 2026-07-15
41. National Institute of Standards and Technology: Tools for combinatorial testing developed by the NIST ACTS project (2019), https://github.com/usnistgov/combinatorial-testing-tools, accessed: 2026-07-15
42. Nie, C., Leung, H.: A survey of combinatorial testing. ACM Computing Surveys **43**(2) (2011). https://doi.org/10.1145/1883612.1883618
43. Rényi, A.: Foundations of Probability. Holden-Day (1970)
44. Shasha, D.E., Kouranov, A.Y., Lejay, L.V., Chou, M.F., Coruzzi, G.M.: Using Combinatorial Design to Study Regulation by Multiple Input Signals. A Tool for Parsimony in the Post-Genomics Era. Plant Physiology **127**(4), 1590–1594 (12 2001)
45. Tang, D.T., Woo, L.S.: Exhaustive test pattern generation with constant weight vectors. IEEE Transactions on Computers **C-32**(12), 1145–1150 (1983)
46. van Beek, P.: Backtracking search algorithms. In: Rossi, F., van Beek, P., Walsh, T. (eds.) Handbook of Constraint Programming, Foundations of Artificial Intelligence, vol. 2, pp. 85–134. Elsevier (2006)
47. Wallace, D.R., Kuhn, D.R.: Failure modes in medical device software: An analysis of 15 years of recall data. International Journal of Reliability, Quality and Safety Engineering **08**(04), 351–371 (2001). https://doi.org/10.1142/S0218539301000058X
48. Williams, A.W.: Determination of test configurations for pair-wise interaction coverage. In: Ural, H., Probert, R.L., v. Bochmann, G. (eds.) Testing of Communicating Systems: Tools and Techniques. IFIP TC6/WG6.1 13th International Conference on Testing of Communicating Systems (TestCom 2000), August 29–September 1, 2000, Ottawa, Canada. pp. 59–74. Springer US, Boston, MA (2000). https://doi.org/10.1007/978-0-387-35516-0_4
49. Wu, H., Nie, C., Petke, J., Jia, Y., Harman, M.: A survey of constrained combinatorial testing (2019), https://arxiv.org/abs/1908.02480
50. Wu, H., Nie, C., Petke, J., Jia, Y., Harman, M.: Comparative analysis of constraint handling techniques for constrained combinatorial testing. IEEE Transactions on Software Engineering **47**(11), 2549–2562 (Nov 2021)
51. Yang, J., Yin, S., Wang, J., Li, S.: A method for estimating minimum sizes of covering arrays avoiding forbidden edges by decomposing graphs. In: 2021 IEEE International Conference on Information Communication and Software Engineering (ICICSE). pp. 185–190 (2021)
52. Yu, L., Duan, F., Lei, Y., Kacker, R.N., Kuhn, D.R.: Combinatorial test generation for software product lines using minimum invalid tuples. In: 2014 IEEE 15th International Symposium on High-Assurance Systems Engineering. pp. 65–72 (2014). https://doi.org/10.1109/HASE.2014.18
53. Yu, L., Duan, F., Lei, Y., Kacker, R.N., Kuhn, D.R.: Constraint handling in combinatorial test generation using forbidden tuples. In: 2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW). pp. 1–9 (2015). https://doi.org/10.1109/ICSTW.2015.7107441
54. Zeller, A., Hildebrandt, R.: Simplifying and isolating failure-inducing input. IEEE Transactions on Software Engineering **28**(2), 183–200 (2002)
55. Zhang, J., Zhang, Z., Ma, F.: The IPO family. In: Automatic Generation of Combinatorial Test Data. pp. 41–49. Springer Berlin Heidelberg, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43429-1_4