Covering Arrays: Algorithms and Asymptotics

by

Kaushik Sarkar

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved October 2016 by the
Graduate Supervisory Committee:

Charles J. Colbourn, Chair
Andrzej Czygrinow
Andrea W. Richa
Violet R. Syrotiuk

ARIZONA STATE UNIVERSITY

December 2016

ABSTRACT

Modern software and hardware systems are composed of a large number of components. Often different components of a system interact with each other in unforeseen and undesired ways to cause failures. Covering arrays are a useful mathematical tool for testing all possible t-way interactions among the components of a system.

The two major issues concerning covering arrays are explicit construction of a covering array, and exact or approximate determination of the covering array number—the minimum size of a covering array. Although these problems have been investigated extensively for the last couple of decades, this thesis presents significant improvements on both of these questions using tools from the probabilistic method and randomized algorithms.

First, a series of improvements is developed on the previously known upper bounds on covering array numbers. An estimate for the discrete Stein-Lovász-Johnson bound is derived and the Stein-Lovász-Johnson bound is improved upon using an alteration strategy. Then group actions on the set of symbols are explored to establish two asymptotic upper bounds on covering array numbers that are tighter than any of the presently known bounds.

Second, an algorithmic paradigm, called the two-stage framework, is introduced for covering array construction. A number of concrete algorithms from this framework are analyzed, and it is shown that they outperform current methods in the range of parameter values that are of practical relevance. In some cases, a reduction in the number of tests by more than 50% is achieved.

Third, the Lovász local lemma is applied to covering perfect hash families to obtain an upper bound on covering array numbers that is tightest of all known bounds. This bound leads to a Moser-Tardos type algorithm that employs linear algebraic computation over finite fields to construct covering arrays. In some cases, this algorithm outperforms currently used methods by more than an 80% margin.

Finally, partial covering arrays are introduced to investigate a few practically relevant relaxations of the covering requirement. Using probabilistic methods, bounds are obtained on partial covering arrays that are significantly smaller than for covering arrays. Also, randomized algorithms are provided that construct such arrays in expected polynomial time.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my adviser Dr. Charles Colbourn, without whom this thesis would not have been possible. What started as a class project has today turned into a dissertation due to his constant encouragement, support, and guidance. He gave me the freedom to work on whatever idea that was most interesting to me and often rescued me when those were leading me nowhere. I am extremely grateful to him for the confidence that he has shown in me.

Second, I would like to thank my dissertation committee members—Dr. Andrzej Czygrinow, Dr. Andrea Richa and Dr. Violet Syrotiuk. Not only were they always accommodating for my last minute requests in spite of their busy schedules, but also, as my teachers for a number of courses, they were responsible for my intellectual development at ASU in many ways.

I would like to thank Dr. Hari Sundaram, Dr. Marco A. Janssen, and Dr. K. Selcuk Candan for their initial support of me at ASU. I would like to thank CIDSE for giving me this opportunity of pursuing my Ph.D. at ASU.

I would like to thank Dr. Arijit Bishnu and Dr. Rana Barua from my masters' days at Indian Statistical Institute for kindling my interest in theory. I would like to thank Dr. Palash Sarkar for giving me the opportunity to pursue research for my masters' thesis under his guidance and always being a source of inspiration.

I would also like to thank my friends—Shruti Gaur, Tien Le, Joydeep Banerjee, Tathagata Chakraborti, Suhas Ranganath, Soumajyoti Sarkar, Lydia Monikonda, Shubhadip Senapati, Suratna Hazra, Shatabdi Roy Chowdhury, Anasuya Pal, Akanksha Singh, Soumyottam Chatterjee, Rajarshi Guha, Rakesh Banerjee and Riddhi Bhowmick—who made my stay at Tempe fun and enjoyable.

Finally, I am forever indebted to my family without whom I would not be where I am today. My parents were always very supportive of all my decisions and stood by me through thick and thin. My wife Poulami not only endured me during these five years of my Ph.D. but also agreed to marry me. She played the role of an emotional center in my life during a period which was basically an emotional roller-coaster ride. Special thanks go to her for proof-reading several chapters of this thesis. Without her, this thesis would have many more misplaced articles than I would otherwise like to have.

The material in this thesis is based in part upon work supported by the National Science Foundation under Grant No. 1421058.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

This thesis is about covering arrays—a type of combinatorial object that finds important application in software and hardware interaction fault testing. In this thesis, we investigate a number of construction algorithms and upper bounds for covering arrays. To obtain a better appreciation of the context, we begin this chapter with an informal discussion of the problem of interaction fault testing and covering arrays. A more formal introduction to the topic is presented in the next chapter on background material (Chapter 2). We conclude this chapter with a brief summary of the main contributions and an outline of the thesis.

1.1 Interaction Faults

Any sufficiently complicated engineered system is composed of a number of components. Although individual components are usually tested exhaustively, often faults occur due to undesired or unforeseen interactions among components. Consider the following conversation between two browser developers after a crash is reported:

"Did you see the bug report on the bookmark sync module crash?"

"Yes. It says that to replicate you first need to go to the private browsing mode, and then enable the add-on option from the settings menu."

"Hmm... That is interesting! We tested the sync module in private browsing mode."

"And I remember writing test cases for the module where I set the add-on attribute to "enabled"."

"But did we test the module under both the conditions simultaneously?"

"Probably not. That's why we did not detect this crash during testing."

Empirical studies on software faults show that 70% of all reported bugs in deployed software result from interactions between at most 2 components, and 90% of all reported bugs occur due to interactions among at most 3 different components [40]. These results are fairly consistent across different domains of software [41]. Given the preponderance of interaction faults in deployed software it becomes a high priority for the software tester to find these faults during the testing phase.

However, given k components where each component can have only two different initial configurations, there are 2^k different combinations to check, an exponential number. If it were indeed the case that faults can occur due to the interaction among all k components in the system, then software testers may be condemned to checking all 2^k different combinations — a gigantic and often impractical task. Fortunately, the same empirical studies show that virtually no faults occur due to interactions among more than 6 different components. Therefore, testing up to all 6-way interactions is called *pseudo-exhaustive testing* [38]. In this thesis, we investigate a mathematical structure that facilitates pseudo-exhaustive interaction testing of complex engineered systems.

1.2 Covering Arrays

A covering array is a type of combinatorial array that is helpful in planning and automating the testing of all possible t-way interactions. Suppose the system under consideration has k different components, and each component has v different possible initial settings. A choice of any t factors and assignment of t initial settings to them constitute a t-way interaction. There are v^t different t-way interactions involving any specific set of t components. So there are $\binom{k}{t}v^t$ different t-way interactions. A *covering array* for such a system is denoted by $\mathsf{CA}(N; t, k, v)$ and is an $N \times k$ array A

Component			
Role	Interface	Database	Language
Admin.	PC	MongoDB	English
Blogger	Tablet	Cassandra	Spanish
Reader	Mobile	Postgres	Bengali

Table 1.1: 4 different components of a blogging platform, each with 3 different initial configurations.

in which each entry is from a v-ary alphabet Σ, such that for every t-set of columns, each of the v^t tuples is covered in at least one row of the array. The rows of a covering array correspond to test cases, and all the rows taken together constitute a test suite that tests all possible t-way interactions among the k components.

Suppose you have developed a new blogging platform. It supports three different roles: administrator, blogger, and reader. For each role, depending on the type of the device, it has three different interfaces: PC, tablet, and mobile phone. Also, it can be configured to use any of the following modern database engines: MongoDB, Cassandra, and Postgres. Furthermore, the platform can be customized to display in any of the three languages: English, Spanish, and Bengali. Taken together, the platform can support 81 different role-interface-database-language combinations. In the parlance of covering arrays, for this system, we have $k = 4$ and $v = 3$. The system is summarized in Table 1.1.

In this case, it may be possible to test the software under all these 81 combinations separately. However, we can see why this exhaustive test strategy is a hopeless enterprise in more complicated situations due to the combinatorial explosion in the number of testing scenarios. Therefore, we resort to t-way interaction testing for different values of t in the range $2 \leq t \leq 6$ using covering arrays. For example, using

Test Case	Role	Interface	Database	Language
1	Admin.	PC	MongoDB	English
2	Admin.	Mobile	Postgres	Spanish
3	Admin.	Tablet	Cassandra	Bengali
4	Blogger	PC	Postgres	Bengali
5	Blogger	Tablet	MongoDB	Spanish
6	Blogger	Mobile	Cassandra	English
7	Reader	PC	Cassandra	Spanish
8	Reader	Mobile	MongoDB	Bengali
9	Reader	Tablet	Postgres	English

Table 1.2: Test suite to cover all 2-way interactions for the system shown in Table 1.1.

covering arrays we can determine that 9 different scenarios are sufficient for testing all possible 2-way interactions in this example [17]. Table 1.2 shows a collection of 9 such test cases that cover all 2-way interactions among the 4 components. Similarly, 45 different scenarios are sufficient for testing all possible 3-factor combinations [17].

The difference between the number of scenarios in exhaustive testing and t-way interaction testing becomes more pronounced as the number of factors and the number of options available under each factor increase. For example, if we have 10 different factors with 4 different levels available under each factor, the number of different scenarios that needs to be tested under the exhaustive strategy is 4^{10} or $1,048,576$. However, using covering arrays one would require 24 different scenarios for testing all possible 2-factor combinations, 100 different scenarios for testing all possible 3-factor combinations, and 8176 scenarios for testing all possible 6-factor combinations [17].

Covering arrays have uses in interaction testing in complex engineered systems. To ensure that all possible combinations of options of every t components function together correctly, one needs to examine all possible t-way interactions. When the number of components is k, and the number of different options available for each component is v, each row of $\mathsf{CA}(N; t, k, v)$ represents a test case. The N test cases collectively test all t-way interactions.

Apart from software interaction testing, covering arrays find important application in hardware testing (see [38] and references therein). Applications of covering arrays also arise in experimental testing for advanced materials [9], inference of interactions that regulate gene expression [50], fault-tolerance of parallel architectures [28], synchronization of robot behavior [31], drug screening [53], and learning of boolean functions [21]. Covering arrays have been studied using different nomenclature, as qualitatively independent partitions [26], t-surjective arrays [10], and (k, t)-universal sets [35], among others. Covering arrays are closely related to hash families [18] and orthogonal arrays [16].

1.2.1 Research Problems

The cost of testing is directly related to the number of test cases. Therefore, one is interested in covering arrays with the fewest rows. The smallest value of N for which $\mathsf{CA}(N; t, k, v)$ exists is denoted by $\mathsf{CAN}(t, k, v)$. Efforts to determine or bound $\mathsf{CAN}(t, k, v)$ have been extensive; see Colbourn [16, 18], Kuhn *et al.* [38], Nie and Leung [46] for example. Naturally one would prefer to determine $\mathsf{CAN}(t, k, v)$ exactly. Katona [36] and Kleitman and Spencer [37] independently showed that for $t = v = 2$, the minimum number of rows N in a $\mathsf{CA}(N; 2, k, 2)$ is the smallest N for which $k \leq \binom{N-1}{\lceil \frac{N}{2} \rceil}$. Exact determination of $\mathsf{CAN}(t, k, v)$ for other values of t and v has remained open. Therefore, there is a huge theoretical interest in determining an asymptotic upper bound on $\mathsf{CAN}(t, k, v)$. Some progress has been made in determining upper bounds for $\mathsf{CAN}(t, k, v)$ in the general case; for recent results, see Francetić and Stevens [23]. In this thesis, we address this problem at great lengths and present a number of improvements on the currently best known results.

However, for practical applications, such bounds are often unhelpful because one needs explicit covering arrays to use as test suites. Explicit constructions can be recursive, producing larger covering arrays using smaller ones as ingredients (see Colbourn [18] for a survey), or direct. Direct methods for some specific cases arise from algebraic, geometric, or number-theoretic techniques; general direct methods are computational in nature. Indeed when k is relatively small, the best known results arise from computational techniques [17], and these are in turn essential for the success of the recursive methods. Unfortunately, for larger values of k that are relevant for some practical applications, existing computational methods do not scale very well. Typically such difficulties arise either as a result of storage or time limitations or by producing covering arrays that are too big to compete with those arising from simpler recursive methods.

Cohen [14] discusses commercial software where the number of factors often exceeds 50. Aldaco et al. [1] analyze a complex engineered system having 75 factors, using a variant of covering arrays. Android [3] uses a *Configuration* class to describe the device configuration; there are 17 different configuration parameters with $3 - 20$ different levels.

In each of these cases, while existing techniques are effective when the strength is small, these moderately large values of k pose concerns for larger strengths. In this thesis, we present a number of computational construction techniques for covering arrays that are helpful when the number of factors is moderately large (e.g., k in the range $30 - 100$) and the strength of the interactions to be covered is also large (e.g., $t \in \{5, 6\}$).

1.3 Summary of Contributions

Asymptotic upper bounds for $\mathsf{CAN}(t, k, v)$ have earlier been established using the Stein-Lovász-Johnson strategy and the Lovász local lemma. A series of improvements on these bounds is developed in this thesis. First, an estimate for the discrete Stein-Lovász-Johnson bound is derived. Then using an alteration strategy, the Stein-Lovász-Johnson bound is improved upon, leading to a two-stage construction algorithm. A similar two-stage bound is derived that employs the Lovász local lemma and the conditional Lovász local lemma distribution. Finally, group actions on the set of symbols are explored to establish two asymptotic upper bounds on $\mathsf{CAN}(t, k, v)$ that are tighter than the known bounds.

On the algorithmic side, a generic two-stage framework, with a number of concrete algorithms, is developed for efficient construction of covering arrays. In the first stage, a time and memory efficient randomized algorithm covers most of the interactions. In the second stage, a more sophisticated search covers the remainder in relatively few tests. In this way, the storage limitations of the sophisticated search algorithms are avoided; hence the range of the number of components for which the algorithm can be applied is extended, without increasing the number of tests. Many of the framework instantiations can be tuned to optimize a memory-quality trade-off so that fewer tests can be achieved using more memory. These algorithms outperform the currently best known methods when the number of factors ranges from 20 to 60, the number of levels for each factor ranges from 3 to 6, and t-way interactions are covered for $t \in \{5, 6\}$. In some cases, a reduction in the number of tests by more than 50% is achieved.

Next, we apply the Lovász local lemma in conjunction with permutation vectors to obtain an asymptotic upper bound on $\mathsf{CAN}(t, k, v)$ that is tighter than any other known bound. This bound leads to a Moser-Tardos type randomized algorithm that employs linear algebraic computation over finite fields to construct covering arrays. For $t \in \{5, 6\}$, and $v \geq 3$ this algorithm provides the best known covering arrays when k ranges between 20 and 200. A reduction in the number of tests by more than 80% is often achieved for high values of k.

Finally, we introduce and investigate a few variants of covering arrays that are known as partial covering arrays. Sensible and practically relevant relaxations of the covering requirement arise when (1) the set $\{1, 2, \ldots, v\}^t$ need only be contained among the rows of *at least* $(1-\epsilon)\binom{k}{t}$ of the $N \times t$ subarrays and (2) the rows of *every* $N \times t$ subarray need only contain a (large) *subset* of $\{1, 2, \ldots, v\}^t$. Using probabilistic methods, we obtain significant improvements on the covering array upper bound for both relaxations, and for the conjunction of the two. In each case, we provide a randomized algorithm that constructs such arrays in expected polynomial time.

1.4 Organization of the Thesis

The rest of the thesis is organized as follows: Chapter 2 introduces all the necessary notation and concepts and summarizes best known results on $\mathsf{CAN}(t, k, v)$ upper bounds prior to this thesis. This chapter also outlines the key tools and techniques used extensively throughout the thesis.

Chapter 3 discusses two results that improve upon the Stein-Lovász-Johnson bound. The first bound utilizes the idea of discretization. The second bound uses a familiar technique from the probabilistic method — called alteration — to prove a tighter bound. Also, we see the first application of group action that improves the alteration based bound.

Chapter 4 serves as an introduction to the density algorithm in the framework of conditional expectation based derandomization of randomized algorithms. The density algorithm was introduced in Bryce and Colbourn [6] and its variants [7, 19] are probably the most successful computational method for covering array construction. It plays an important role in the next chapter on the two-stage algorithmic framework. Also, we present an original randomized algorithm and its derandomized version called the biased density algorithm in this chapter.

Chapter 5 introduces the two-stage algorithmic framework for covering array construction. We define a number of concrete algorithms within this framework and analyze them. Also, we provide extensive computational results that compare these algorithms against the currently known best results.

Chapter 6 delves deeper into the question of asymptotic upper bounds on $\mathsf{CAN}(t,k,v)$. We apply the Lovász local lemma in conjunction with different transitive group actions to obtain asymptotically tighter upper bounds. The chapter also describes a Moser-Tardos type randomized algorithm for covering array construction. Also, we investigate a Moser-Tardos type algorithm for the first stage of our two-stage framework and derive an upper bound on $\mathsf{CAN}(t,k,v)$ based on this strategy.

Chapter 7 describes permutation vectors and covering perfect hash families. We describe the necessary and sufficient conditions for a set of permutation vectors to be covering and based on this characterization derive an upper bound on the size of covering arrays by applying the Lovász local lemma. We compare this upper bound to the presently known bounds and show that this bound turns out to be the tightest of all the known bounds. Then we describe a Moser-Tardos type construction algorithm based on permutation vectors and present computational results comparing this algorithm against the best known results.

Chapter 8 introduces two variants of partial covering arrays. In this chapter, we derive upper bounds on the size of such partial covering arrays by applying the Lovász local lemma and other probabilistic methods. We also provide polynomial time randomized construction algorithms for such arrays. Finally, we present a result that tells us that a very small relaxation in coverage requirement leads to partial covering arrays that are substantially smaller than covering arrays.

Chapter 9 concludes the thesis. There we provide a comprehensive table of different general $\mathsf{CAN}(t, k, v)$ upper bounds and summarize the main themes of the thesis. Finally, we end with a discussion of some open issues and future research ideas.

Chapter 2

BACKGROUND

In this chapter, first we establish the formal terminology that is useful for talking about covering arrays. Then we summarize the known general upper bounds on covering array numbers for easy reference and comparison in later chapters. In the last section of the chapter, we introduce a few tools and techniques from discrete probability theory and combinatorics, e.g., the Lovász local lemma, the Moser-Tardos algorithm, and group action. These tools will be applied a number of times in the later chapters to derive the main results of this thesis.

2.1 Terminology

When k is a positive integer, we use $[k]$ to denote the set $\{1, \ldots, k\}$. Let N, t, k, and v be positive integers such that $k \geq t \geq 2$ and $v \geq 2$. A *covering array* $\mathsf{CA}(N; t, k, v)$ is an $N \times k$ array A in which each entry is from a v-ary alphabet Σ, and for every $N \times t$ sub-array B of A and every $\mathbf{x} \in \Sigma^t$, there is a row of B that equals $\mathbf{x}$. The parameter t is called the *strength*, k is called the number of *factors*, and v is called the number of *levels* of the covering array. The smallest value of N for which $\mathsf{CA}(N; t, k, v)$ exists is denoted by $\mathsf{CAN}(t, k, v)$.

A *t-way interaction* is defined as the set $\{(c_i, a_i) : 1 \leq i \leq t, c_i \in [k], c_i \neq c_j \text{ for } i \neq j, \text{ and } a_i \in \Sigma\}$. Basically, an interaction is an assignment of values from Σ to t of the k columns. An $N \times k$ array A *covers* the interaction $\iota = \{(c_i, a_i) : 1 \leq i \leq t, c_i \in [k], c_i \neq c_j \text{ for } i \neq j, \text{ and } a_i \in \Sigma\}$ if there is a row r in A such that $A(r, c_i) = a_i$ for $1 \leq i \leq t$. When there is no such row in A, ι is *not* covered in A. Hence a $\mathsf{CA}(N; t, k, v)$ covers all the t-way interactions involving k factors each having v levels.

Let $\mathcal{C}_{t,k}$ be the set of all subsets of size t of $[k]$, i.e. $\mathcal{C}_{t,k} = \binom{[k]}{t}$. Let $\mathcal{I}_{t,k,v}$ denote the set of all $\binom{k}{t} \cdot v^t$ different t-way interaction among the k factors, each having v levels. We adopt a canonical representation for any t-subset of $[k]$ as an increasing sequence of t values from $[k]$. Define $c : \mathcal{I}_{t,k,v} \to \mathcal{C}_{t,k}$ as follows: For $\iota \in \mathcal{I}_{t,k,v}$, $c(\iota) = (c_1, \ldots, c_t)$ where $(c_i, a_i) \in \iota$ for some $a_i \in \Sigma$, and $c_i < c_j$ for $1 \leq i < j \leq t$. We use $c(\iota)_i$ to denote the ith element of $c(\iota)$, i.e., c_i. Similarly, define $s : \mathcal{I}_{t,k,v} \to \Sigma^t$ as follows: For an interaction $\iota \in \mathcal{I}_{t,k,v}$, define $s(\iota) = (a_1, \ldots, a_t)$ where $(c(\iota)_i, a_i) \in \iota$ for $1 \leq i \leq t$. We use $s(\iota)_i$ to denote the ith element of the t-tuple $s(\iota)$. A bijection from $\mathcal{I}_{t,k,v}$ to $\mathcal{C}_{t,k} \times \Sigma^t$ maps $\iota \to (c(\iota), s(\iota))$. Therefore, the interaction ι can be described by the ordered pair $(c(\iota), s(\iota))$ as well.

In this thesis, we primarily focus on situations where every factor has the same number of levels. These cases have been most extensively studied, and hence provide a basis for making comparisons. In practice, however, different components often have different number of levels, which is captured by extending the notion of a covering array. A *mixed covering array* $\mathsf{MCA}(N; t, k, (v_1, v_2, \ldots, v_k))$ is an $N \times k$ array in which the ith column contains v_i symbols for $1 \leq i \leq k$. Given any subset of t columns $\{i_1, \ldots, i_t\} \subseteq \{1, \ldots, k\}$, the $N \times t$ subarray, obtained by selecting columns $i_1, \ldots, i_t$ of the MCA, contains each of the $\prod_{j=1}^{t} v_{i_j}$ distinct t-tuples as a row at least once. Although we examine the uniform case in which $v_1 = \cdots = v_k$, the many of the methods developed this thesis (e.g., the two-stage algorithms from Chapter 5) can be directly applied to mixed covering arrays as well.

2.1.1 $\mathsf{CAN}(t, k, v)$ *Upper Bounds*

Given the interpretation of $\mathsf{CAN}(t, k, v)$ as the minimum number of required test cases to test all possible t-way interaction among k factors, each having v different levels, it is natural that one would prefer to determine $\mathsf{CAN}(t, k, v)$ exactly. Katona [36] and Kleitman and Spencer [37] independently showed that for $t = v = 2$, the minimum number of rows N in a $\mathsf{CA}(N; 2, k, 2)$ is the smallest N for which

$$k \leq \binom{N-1}{\lceil \frac{N}{2} \rceil}.$$

So far it has not been possible to determine the exact value of $\mathsf{CAN}(t, k, v)$ as a function of k for any other values of $t > 2$ or $v > 2$.

In light of this, the asymptotic determination of $\mathsf{CAN}(t, k, v)$ has been of substantial interest. For fixed t and v, it is well-known that $\mathsf{CAN}(t, k, v)$ is $\Theta(\log k)$; the lower bound can be established by observing that all columns are distinct, and the upper bound is a simple probabilistic argument (see Theorem 1, for example). When $t = 2$ and $v \geq 2$, Gargano et al. [26] establish the much more precise statement that

$$\mathsf{CAN}(2, k, v) = \frac{v}{2} \log k \left\{1 + o(1)\right\}. \tag{2.1}$$

(Throughout the thesis, we write log for logarithms base 2, and ln for natural logarithms.) However, when $t > 2$, even the coefficient of the highest order term is not known precisely.

The first general upper bound on $\mathsf{CAN}(t, k, v)$ was obtained by specializing the method of Stein [52], Lovász [42], and Johnson [34] to the case of covering arrays. Because the ideas used here are essential for the rest of the thesis, we provide a complete proof of this known result.

Theorem 1. [34, 42, 52]*(Stein-Lovász-Johnson (SLJ) bound) Let t, k, v be integers with $k \geq t \geq 2$, and $v \geq 2$. Then as $k \to \infty$,*

$$\mathsf{CAN}(t, k, v) \leq \frac{t}{\log\left(\frac{v^t}{v^t - 1}\right)} \log k (1 + \mathrm{o}(1))$$

Proof. Let A be an $N \times k$ array in which each entry is chosen independently and uniformly at random from an alphabet Σ of size v. The probability that a specific interaction of strength t is not covered in A is $\left(1-\frac{1}{v^t}\right)^N$. By the linearity of expectations, the expected number of uncovered interactions in A is $\binom{k}{t}v^t\left(1-\frac{1}{v^t}\right)^N$. If this expectation is less than 1, because the number of uncovered interactions is an integer, there is an array with N rows that covers all the interactions. Solving $\binom{k}{t}v^t\left(1-\frac{1}{v^t}\right)^N < 1$, we get $\mathsf{CAN}(t,k,v) \le \frac{\log\binom{k}{t}+t\log v}{\log\left(\frac{v^t}{v^t-1}\right)}$. Simplifying further,

$$\begin{aligned}\mathsf{CAN}(t,k,v) &\le \frac{\log\binom{k}{t}+t\log v}{\log\left(\frac{v^t}{v^t-1}\right)}\\ &\le \frac{t\log\left(\frac{ke}{t}\right)+t\log v}{\log\left(\frac{v^t}{v^t-1}\right)}\\ &= \frac{t\log k}{\log\left(\frac{v^t}{v^t-1}\right)}\left(1+\frac{1}{\log k}-\frac{\log t}{\log k}+\frac{\log v}{\log k}\right)\\ &= \frac{t}{\log\left(\frac{v^t}{v^t-1}\right)}\log k(1+\mathrm{o}(1))\end{aligned}$$

This completes the proof. □

Because it is already known that $\mathsf{CAN}(t,k,v)=\Theta(\log k)$, for asymptotic comparison one is interested in more accurate determination of the coefficient of the highest order term as k approaches infinity. To simplify this comparison we define

$$d(t,v)=\limsup_{k\to\infty}\frac{\mathsf{CAN}(t,k,v)}{\log k}.$$

Using this notation, Theorem 1 can be restated as: For $k\ge t\ge 2$, and $v\ge 2$, $d(t,v)\le\frac{t}{\log\left(\frac{v^t}{v^t-1}\right)}$.

The first asymptotic improvement over the Stein-Lovász-Johnson bound in the general case was obtained by Godbole et al. [27] by accounting for the limited dependence using the Lovász local lemma:

Theorem 2. [27]*(Godbole, Skipper and Sunley (GSS) bound) Let* t, v *be integers with* $t,v\ge 2$. *Then*

$$d(t,v)\le\frac{t-1}{\log\left(\frac{v^t}{v^t-1}\right)}.$$

The next improvement in the general case was obtained by Francetić and Stevens [23] using the entropy compression method and balanced columns:

Theorem 3. *Let t, v be integers with $t, v \geq 2$. Then*

$$d(t,v) \leq \frac{v(t-1)}{\log\left(\frac{v^{t-1}}{v^{t-1}-1}\right)}.$$

In Chapter 6 we obtain the same bound using a much simpler argument and then improve upon it. In Chapter 7 we derive $\mathsf{CAN}(t,k,v)$ upper bounds that are the tightest known asymptotic upper bounds in the general case at the moment.

2.2 Techniques

Next, we introduce a few techniques from probabilistic methods and combinatorics that we use a number of times in this thesis.

2.2.1 The Lovász Local Lemma

When events are independent it is quite easy to compute the probabilities and analyze the situation. However, that is rarely the case. On the other hand, it is also rare to find oneself in a situation where all the events are interdependent. In between these extremes lies the situation where the dependence among different events is limited. The Lovász local lemma is great tool under these circumstances.

Lemma 4. *(Lovász local lemma; general case) (see* [2]*) Let $A_1, A_2, \ldots, A_n$ be events in an arbitrary probability space. A directed graph $D = (V, E)$ on the set of vertices $V = \{1, 2, \ldots, n\}$ is called a dependency digraph for the events $A_1, A_2, \ldots, A_n$ if for each i, $1 \leq i \leq n$, the event A_i is mutually independent of all the events $\{A_i : (i,j) \notin E\}$. Suppose that $D = (V, E)$ is a dependency digraph for $\{A_1, \ldots, A_n\}$ and suppose there are real numbers $x_1, x_2, \ldots, x_n$ such that $0 \leq x_i < 1$ and $\Pr[A_i] \leq x_i \prod_{(i,j)\in E}(1 - x_j)$*

for all $1 \leq i \leq n$. *Then*

$$\Pr\left[\wedge_{i=1}^{n}\overline{A_i}\right] \geq \prod_{i=1}^{n}(1-x_i).$$

In particular, with positive probability, no event A_i *holds.*

To apply the Lovász local lemma successfully, the problem of interest needs to be modeled as a situation where we are trying to avoid occurrence of a set of "bad" events. The dependency digraph provides us the dependency structure among these "bad" events. Then the Lovász local lemma tells us that if the probabilities of the "bad" events are bounded from above, with positive probability we can find an outcome where all the "bad" events are avoided.

Often it is the case that the "bad" events are of the same type and have the same upper bound. Under such circumstances we may apply the following version of the lemma, which can be easily derived from Lemma 4.

Lemma 5. *(Lovász local lemma (LLL); symmetric case) (see* [2]*) Let* $A_1, A_2, \ldots, A_n$ *be events in an arbitrary probability space. Suppose that each event* A_i *is mutually independent of a set of all other events* A_j *except for at most* d*, and that* $\Pr[A_i] \leq p$ *for all* $1 \leq i \leq n$. *If* $ep(d+1) \leq 1$*, where* e *is the base of the natural logarithm, then* $\Pr[\cap_{i=1}^{n}\bar{A}_i] > 0$.

The symmetric version of Lovász local lemma provides an upper bound on the probability of a "bad" event in terms of the maximum degree of a bad event in a dependence graph so that the probability that all the "bad" events are avoided is non-zero.

We apply the symmetric version of the Lovász local lemma in Sections 6.1, 6.2.2, 7.3, and 8.1.

The Moser-Tardos Algorithm and a Constructive Version of Lovász Local Lemma

In its original form the Lovász local lemma is a non-constructive result. It just guarantees that the desired object exists with non-zero probability. However, the probability may be vanishingly small to allow a uniform random sampling algorithm to efficiently construct such objects in a polynomial time. Fortunately, due to Moser and Tardos [45] we have a constructive version of the local lemma. Moser and Tardos [45] describe a randomized polynomial time algorithm that constructs the desired object whenever the sufficient condition in the Lovász local lemma is satisfied.

First, we describe the setting. Let $\mathscr{X} = \{X_1, X_2, \ldots, X_m\}$ be a finite set of mutually independent random variables in a fixed probability space Ω. Let A be an event in the same space Ω that is determined by the values of the random variables in a subset $S \subseteq \mathscr{X}$. If a particular assignment of values of the variables in S makes A happen, then we say that the assignment *realizes* A. If A is determined by $\mathscr{X}$ then there is a unique minimal set $S \subseteq \mathscr{X}$ that determines A. We denote that minimal set of random variables that determines A by $\mathsf{vbl}(A)$.

Consider a finite set of events $\mathscr{A} = \{A_1, A_2, \ldots, A_n\}$ in Ω that are determined by the random variables in $\mathscr{X}$. An event $A_i \in \mathscr{A}$ is mutually independent of all other events $A_j \in \mathscr{A}$ such that $\mathsf{vbl}(A_i) \cap \mathsf{vbl}(A_j) = \phi$. Therefore, we can define a dependency digraph $D = (V, E)$ with the vertex set $V = \{1, 2, \ldots, n\}$ such that $(i, j) \in E$ if and only if $\mathsf{vbl}(A_i) \cap \mathsf{vbl}(A_j) \neq \phi$.

Consider Algorithm 1. It starts with a random assignment of values to the variables in $\mathscr{X}$. Then it sequentially checks all the events until it finds an event that is realized under the current assignment. It "resamples" only the random variables that determine the realized event, and restarts the checking from the beginning. If the checking loop that starts at line 4 terminates, the algorithm outputs an assignment of values to the random variables in $\mathscr{X}$ such that none of the events in $\mathscr{A}$ are realized.

The following theorem by Moser and Tardos [45] tells us that if the probabilities and the dependence among the events in $\mathscr{X}$ are such that the sufficient conditions of the Lovász local lemma are met, then Algorithm 1 terminates within expected polynomial amount of time.

Lemma 6. *(see [45]) Let $\mathscr{X} = \{X_1, X_2, \ldots, X_m\}$ be a finite set of mutually independent random variables in a fixed probability space Ω, and $\mathscr{A} = \{A_1, A_2, \ldots, A_n\}$ be a finite set of events in Ω that are determined by the random variables $\mathscr{X}$. Let $D = (V, E)$ be the dependency digraph for the events in $\mathscr{A}$. If there exists reals $x_1, x_2, \ldots, x_n$ such that $0 \leq x_i < 1$ and $\Pr[A_i] \leq x_i \prod_{(i,j)\in E}(1 - x_j)$ for $1 \leq i \leq n$, then there exists an assignment of values to the random variables in $\mathscr{X}$ that does not realize any of the events in $\mathscr{A}$. Moreover, Algorithm 1 resamples an event $A_i \in \mathscr{A}$ at most an expected $x_i/(1 - x_i)$ times before it finds such an assignment. Thus, the expected total number of resampling steps is at most $\sum_{i=1}^{n} \frac{x_i}{1-x_i}$*

When we can apply the symmetric version of the Lovász local lemma, the expected number of resampling steps in Algorithm 1 is simply n/d. We will design Moser-Tardos type algorithms in Sections 6.2 and 7.4 for covering array construction, and in Section 8.1 for construction of partial m-covering arrays.

2.2.2 Group Action

Group action on the symbols of a covering array plays an important role in much of the development in this thesis. To be able to talk about such group actions precisely, here we define some of the basic terminology and discuss a few important examples that play a crucial role in the later development.

Algorithm 1: A randomized algorithm for obtaining an assignment of values to the random variables in $\mathscr{X}$ such that none of the events in $\mathscr{A}$ are realized.

```
Input: $\mathscr{X}$ : the set of random variables, $\mathscr{A}$ : a set of events that are
       determined by the random variables in $\mathscr{X}$, and for each $A \in \mathscr{A}$ we
       know $\mathsf{vbl}(A)$
Output: An assignment of values to the random variables in $\mathscr{X}$ such that
        none of the events in $\mathscr{A}$ is realized.
forall $X \in \mathscr{X}$ do
    Let $v_X$ be a random evaluation of $X$;
end
repeat
    Set covered:= true;
    forall event $A \in \mathscr{A}$ do
        if $A$ is realized then
            Set covered := false;
            Set offending-variables := $\mathsf{vbl}(A)$;
            break;
        end
    end
    if covered = false then
        forall $X \in$ offending-variables do
            Set $v_X$ to be a new random evaluation of $X$;
        end
    end
until covered = true;
Output $(v_X)_{X \in \mathscr{X}}$;
```

Let X be a set and Γ be a group. An *action of* Γ *on* X is defined as a map $\varphi : X \times \Gamma \to X$ such that: (1) $\varphi(x, e) = x$ for all $x \in X$, e is the identity element of Γ, and (2) $\varphi(x, g_1 g_2) = \varphi(\varphi(x, g_1), g_2)$ for all $x \in X$ and all $g_1, g_2 \in \Gamma$. If such an action exists then X is called a Γ-set. For example, let n be a positive integer, X be the set $\{0, 1, \ldots, n-1\}$, and Γ be the group of all permutations on n symbols, i.e., $\Gamma = S_n$. For $\sigma \in S_n$, and $x \in \{0, 1, \ldots, n-1\}$, let us define φ in the following way: $\varphi(x, \sigma) = \sigma(x)$, i.e., φ maps the member x to the corresponding member $\sigma(x)$ under the permutation. It is easy to show that φ defines an action of S_n on $\{0, 1, \ldots, n-1\}$. From now on we will shorten $\varphi(x, g)$ to $g(x)$ for $g \in \Gamma$ and $x \in X$.

For any positive integer m, whenever X is a Γ-set, X^m is a Γ-set as well. We can extend the action of Γ on X to X^m in the natural way: For $\chi = (x_1, x_2, \ldots, x_m) \in X^m$ and $g \in \Gamma$, define $g(\chi)$ as the tuple $(g(x_1), g(x_2), \ldots, g(x_m))$. This defines a group action on X^m.

We can define a relation $\sim$ on a Γ-set X in the following way: $x_1 \sim x_2$ for $x_1, x_2 \in X$ if there exists $g \in \Gamma$ such that $g(x_1) = x_2$. Then $\sim$ defines an equivalence relation on X, whose equivalence classes are called *orbits*. If the orbits are finite then the *length* of an orbit refers to the number of elements in the orbit. For $x \in X$, the *orbit of* x refers to the equivalence class to which x belongs.

We often focus on covering arrays that are invariant under the action of a group. More precisely, we consider the action of different subgroups of S_v over Σ^t and Σ^k where, as usual, v is the number of levels, t is the strength of interaction, k is the number of factors and $\Sigma = \{0, 1, \ldots, v-1\}$.

The action of a group Γ on Σ is *sharply transitive* if for every $x, y \in \Sigma$ there is exactly one $\sigma \in \Gamma$ that maps x to y. In such cases $|\Gamma| = |\Sigma| = v$. For example, the action of the cyclic group C_v on Σ is sharply transitive. Under the action of a sharply transitive group Γ, Σ forms only one orbit of length v that includes all the members.

However, for $t \geq 2$, any sharply transitive group action partitions Σ^t into v^{t-1} orbit of length v each.

Similarly, the action of a group Γ on Σ is called *sharply ℓ-transitive* if for all pairwise distinct $x_1, \dots, x_\ell \in \Sigma$ and pairwise distinct $y_1, \dots, y_\ell \in \Sigma$ there is exactly one $\sigma \in \Gamma$ that maps x_i to y_i for $1 \leq i \leq \ell$. For $\ell \leq t$, a sharply ℓ-transitive group action on Σ^t forms orbits of length v, $v(v-1)$, up to $v(v-1)\dots(v-\ell+1)$.

When v is a prime power, let Γ be the Frobenius group, defined on the finite field $\mathbb{F}_v$ in the following way: $\Gamma = \{g : \mathbb{F}_v \to \mathbb{F}_v \;:\; g(x) = ax + b,\, x, a, b \in \mathbb{F}_v,\, a \neq 0\}$. Γ is an efficiently constructible group that acts sharply 2-transitively on $\mathbb{F}$. It has been used for the practical construction of covering arrays [19]. When $v = q + 1$, where q is a prime power, the projective general linear (PGL) group is a sharply 3-transitive group of order $v(v-1)(v-2)$.

For the v-symbol alphabet $\Sigma = \{0, 1, \dots, v-1\}$ and a permutation group Γ on the v symbols, we can extend the action of Γ to the set of all t-way interactions among the k factors in a similar manner: For $\sigma \in \Gamma$ and $\iota = \{(c_i, a_i) \;:\; 1 \leq i \leq t,\, c_i \in [k],\, c_i \neq c_j \text{ for } i \neq j, \text{ and } a_i \in \Sigma\}$, σ maps ι to $\{(c_i, \sigma(a_i)) \;:\; 1 \leq i \leq t\}$.

In the computational techniques and constructions that apply group action the key idea is to construct an array A that covers all the orbits of t-way interactions under the action of some group Γ. To be precise, let A be an $n \times k$ array with entries from Σ. We would like A to have the following property: For every orbit of interaction, there is a row in A that covers at least an interaction from this orbit. The rows of A, when developed over Γ provides an array that covers all t-way interactions, and therefore is a covering array. Group action here essentially works as a search space reduction technique.

The strategy of covering orbits of interactions under the action of the permutation group Γ on the symbols has been used in direct and computational methods [11, 44], and in randomized and derandomized methods [19]. In Colbourn [19] it is noted that using a group action appears to construct covering arrays with fewer rows than using similar methods on the covering array directly. We apply group action based techniques in Sections 3.2.1, 5.2.4, 6.1 and 8.2.

Chapter 3

IMPROVING THE STEIN-LOVÁSZ-JOHNSON BOUND

In this chapter we provide two improvements over the basic Stein-Lovász-Johnson bound established in Theorem 1. The first improvement is obtained by applying a discretization idea. The second improvement — the two-stage bound — is obtained by applying a well known strategy, called alteration, from probabilistic methods. Apart from providing an improved estimate, this bound gives the theoretical justification for the two-stage algorithmic framework introduced in Chapter 5.

The basic probabilistic construction of a covering array analyzed in Theorem 1 is quite naïve. We just construct a random $N \times k$ array and hope that it covers all the interactions. Although this randomly constructed covering array does not require asymptotically more rows than the smallest possible covering array, from the practical point of view there are a number of issues to consider. First, the probability that a random array of a specified size is a covering array may be very small, requiring a high number of samples. Also to know that a given $N \times k$ array is a $\mathsf{CA}(N; t, k, v)$ we need to check whether each of the $\binom{k}{t} \cdot v^t$ interactions are covered in the array. That is a lot of checking to do, especially, considering that we may have to do this checking multiple times.

One may wonder if for practical purposes, it makes more sense to construct small portions of the array at a time, instead of constructing the full array. For each part one may want to choose the rows that maximize the number of newly covered interactions. However, this problem is NP-hard [20]. Instead, in this chapter, we consider two strategies where the goal is to construct the parts with "good enough" coverage using random methods. We analyze these strategies and provide quantitative guarantees on how well they perform in comparison to the naïve random method.

The results presented in this chapter are reported in Sarkar and Colbourn [48].

3.1 The Discrete Stein-Lovász-Johnson Bound

The Stein-Lovász-Johnson bound (Theorem 1 in Chapter 2) was obtained by considering an $N \times k$ array where each of the entries was chosen independently and uniformly at random. Rather than choosing the N rows at random, we can build the covering array one row at a time. To select a row, compute the expected number of uncovered interactions that remain when we choose the next row uniformly at random from Σ^k. There must be a row whose selection leaves at most that expected number of interactions uncovered. Indeed except when the first row is selected, some row must leave a number that is strictly less than the expectation, because previously selected rows cover no interaction that is not yet covered. Add such a row to the covering array and repeat until all the interactions are covered. The number of rows employed by this method yields an upper bound on $\mathsf{CAN}(t, k, v)$. If at each stage the row selected left uncovered precisely the expected number of uncovered interactions, we recover Theorem 1. However, after each row selection, the number of uncovered interactions must be an integer no larger than the expected number, improving on the basic SLJ bound. This better upper bound is the *discrete Stein-Lovász-Johnson* (discrete-SLJ) bound.

A row that leaves no more than the expected number uncovered can be computed efficiently when t and v are fixed, so the discrete-SLJ bound can be efficiently derandomized; this is the basis of the *density algorithm* [6, 7]. The density algorithm works well in practice, providing the smallest known covering arrays in many cases [17]. Although Theorem 1 provides an easily computed upper bound on the array sizes produced by the density algorithm, it is a very loose bound.

We analyze the discrete Stein-Lovász-Johnson bound in order to establish a better estimate.

Theorem 7. *The number of rows N in A obtained by the discrete-SLJ bound satisfies*

$$\frac{\log\left\{\binom{k}{t}+1\right\}}{\log\left(\frac{v^t}{v^t-1}\right)} < N \le \frac{\log\left\{\binom{k}{t}+\epsilon\right\}-\log\epsilon}{\log\left(\frac{v^t}{v^t-1}\right)}$$

for some $0<\epsilon<1$ ($\log$ stands for logarithms base 2).

Proof. Let $y = \left(1-\frac{1}{v^t}\right)$ and $x = 1/y$. Let $r(i)$ denote the number of uncovered interactions that remain after i rows are chosen. Suppose that when row i is chosen, it leaves

$$r(i) = \begin{cases} \lfloor yr(i-1)\rfloor & \text{when} \quad i=1 \text{ or } r(i-1)\not\equiv 0 \pmod{v^t} \\ yr(i-1)-1 & \text{when} \quad i>1 \text{ and } r(i-1)\equiv 0 \pmod{v^t} \end{cases}$$

uncovered interactions.

Write $\epsilon(i-1) = yr(i-1)-r(i)$ for $i\ge 1$. Then expanding the recurrence $r(i) = yr(i-1)-\epsilon(i-1)$,

$$r(n) = y^n r(0) - \sum_{i=0}^{n-1} y^{n-1-i}\epsilon(i).$$

Rewriting in terms of x,

$$x^n r(n) = r(0) - \sum_{i=0}^{n-1} x^{i+1}\epsilon(i).$$

Now $r(0)=\binom{k}{t}v^t$ and $r(n)=0$, so

$$\binom{k}{t}v^t = x^n\epsilon(n-1) + \sum_{i=0}^{n-2} x^{i+1}\epsilon(i).$$

Because $r(n)=0$, $y\le\epsilon(n-1)<1$. Then because $0\le\epsilon(i)\le 1$,

$$x^{n-1}+\sum_{i=0}^{n-2}x^{i+1}\epsilon(i) \le \binom{k}{t}v^t < x^n + \sum_{i=0}^{n-2}x^{i+1}\epsilon(i) < \sum_{i=1}^{n} x^i = \frac{x(x^n-1)}{(x-1)}.$$

Simplify $\binom{k}{t}v^t < \frac{x(x^n-1)}{(x-1)}$ to obtain $\binom{k}{t}+1 < x^n$. Take logarithms to establish that $n > \frac{\log\left\{\binom{k}{t}+1\right\}}{\log\left(\frac{v^t}{v^t-1}\right)}$. If we select each row so that $r(n) = \lfloor yr(n-1)\rfloor$, we cannot cover all interactions in $\log\left\{\binom{k}{t}+1\right\}/\log\left(\frac{v^t}{v^t-1}\right)$ rows. This establishes the lower bound.

Note that $\epsilon(0) = 0$. Let $\epsilon = \min\{\epsilon(k) : 1 \le k < n-1\}$. Then $\frac{1}{v^t} \le \epsilon \le 1$, because every row selected after the first covers more than the expected number of previously uncovered interactions. Then for sufficiently large k

$$\begin{array}{rcl} \epsilon\frac{x(x^{n-1}-1)}{(x-1)} & = & \epsilon\sum_{i=1}^{n-1} x^i \le x^{n-1} + \epsilon\sum_{i=2}^{n-2} x^i \\ & < & x^{n-1} + \epsilon\sum_{i=2}^{n-2} x^i + \epsilon(x^{n-1}-x) \\ & \le & x^{n-1} + \sum_{i=0}^{n-2} x^{i+1}\epsilon(i) \le \binom{k}{t}v^t \end{array} .$$

The strict inequality follows from the fact that $x > 1$. Hence $\epsilon(x^n-1) < \binom{k}{t}$, so $n < \frac{\log\left\{\binom{k}{t}+\epsilon\right\}-\log\epsilon}{\log\left(\frac{v^t}{v^t-1}\right)} + 1$, and because n is an integer the upper bound follows. $\square$

Consequently $\log\left\{\binom{k}{t}+1\right\}/\log\left(\frac{v^t}{v^t-1}\right)$ can be used to estimate the discrete Stein-Lovász-Johnson bound. Figure 3.1 compares the estimate to the discrete Stein-Lovász-Johnson bound and the Stein-Lovász-Johnson bound from Theorem 1 when $t = 6$ and $v = 3$. For a wide range of values of k, the reduction in the number of rows is substantial.

The density algorithm [6, 7] enables one to produce covering arrays of sizes at most those given by the bound efficiently. Despite their efficiency in theory, in practice the methods are limited by the need to store information about all t-way interactions; even when $t = 6$, $v = 3$, and $k = 54$, there are 18,828,003,285 6-way interactions, so the storage requirements are limiting. Moreover, as shown in the analysis, rows added towards the end of the process account for relatively few of the interactions. For these reasons, we explore a two-stage approach using alteration.

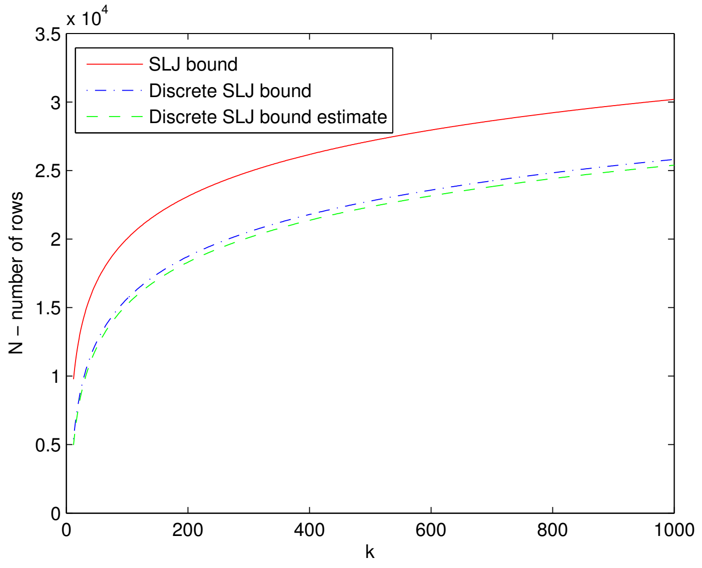


Figure 3.1: Comparison of the Stein-Lovász-Johnson bound, the discrete Stein-Lovász-Johnson bound, and the estimate for the discrete Stein-Lovász-Johnson bound. $t = 6$, $v = 3$.

3.2 Completing a Partial Array: The Two-stage Bound

Alteration is an important strategy in probabilistic methods [2]. The idea is to consider "random" structures that have a few "blemishes," in that they do not have all the desired properties. Such "partial" structures are then altered to obtain the desired property. To apply this technique to covering arrays, in stage 1 we construct a random $n \times k$ array with each entry chosen from the v-ary alphabet Σ independently and uniformly at random. The number of uncovered interactions after stage 1 can be computed using the SLJ or discrete-SLJ bounds. In stage 2, we add one new row for each uncovered interaction to obtain a covering array.

For example, when $t = 6$, $k = 54$ and $v = 3$, Theorem 1 gives $\mathsf{CAN}(6, 54, 3) \leq 17,236$. Using the alteration approach, Figure 5.2 plots an upper bound on the size of the completed covering array against the number n of rows in a partial array that

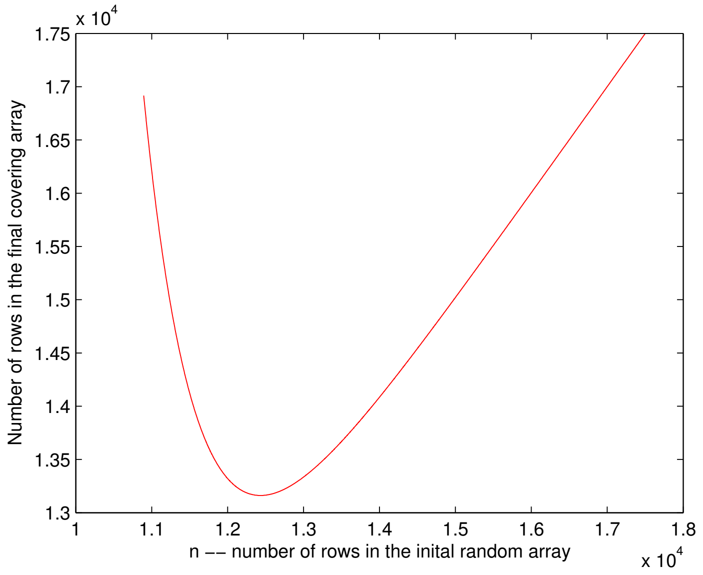


Figure 3.2: Plot of $n + \left\lfloor \binom{k}{t} v^t \left(1 - \frac{1}{v^t}\right)^n \right\rfloor$ against n, the size of a partial array, for $t = 6$, $k = 54$, and $v = 3$. $\binom{k}{t} v^t \left(1 - \frac{1}{v^t}\right)^n$ is the expected number of uncovered interactions in a random $n \times k$ array. The minimum number of rows in the final covering array is $13,162$, achieved when the initial random array has $n = 12,402$ rows. The Stein-Lovász-Johnson bound requires $17,236$ rows, and the best known covering array has $17,197$ rows.

covers at least the expected number of interactions. The smallest covering array is obtained when $n = 12,402$, which when completed yields $\mathsf{CAN}(6, 54, 3) \leq 13,162$. At least in this case, our alteration provides a much better bound. Next, we explore this strategy in general.

Theorem 8. *Let t, k, v be integers with $k \geq t \geq 2$, and $v \geq 2$. Then*

$$\mathsf{CAN}(t, k, v) \leq \frac{\log \binom{k}{t} + t \log v + \log \log \left(\frac{v^t}{v^t - 1}\right) + 1}{\log \left(\frac{v^t}{v^t - 1}\right)}.$$

Proof. In an $n \times k$ array with each entry chosen independently and uniformly at random from an alphabet Σ of size v, the expected number of uncovered t-way interactions is $\binom{k}{t} v^t \left(1 - \frac{1}{v^t}\right)^n$. Let P be an $n \times k$ array with at most $\left\lfloor \binom{k}{t} v^t \left(1 - \frac{1}{v^t}\right)^n \right\rfloor$

uncovered interactions. Let Q contain $\lfloor \binom{k}{t} v^t \left(1 - \frac{1}{v^t}\right)^n \rfloor$ new rows, each covering a different interaction not covered in P. Then $A = \binom{P}{Q}$, where P is vertically concatenated with Q, is a covering array with $n + \lfloor \binom{k}{t} v^t \left(1 - \frac{1}{v^t}\right)^n \rfloor$ rows. So an upper bound on the number of rows in A is $n + \binom{k}{t} v^t \left(1 - \frac{1}{v^t}\right)^n$. Applying elementary calculus, the smallest number of rows is

$$\frac{\log \binom{k}{t} + t \log v + \log \log \left(\frac{v^t}{v^t - 1}\right) + 1}{\log \left(\frac{v^t}{v^t-1}\right)},$$

obtained when P has $n = \frac{\log \binom{k}{t} + t \log v + \log \log \left(\frac{v^t}{v^t-1}\right)}{\log \left(\frac{v^t}{v^t-1}\right)}$ rows. $\square$

For $v, t \geq 2$, $\log \log \left(\frac{v^t}{v^t-1}\right) < 0$. Hence, Theorem 8 gives a tighter bound on $\mathsf{CAN}(t, k, v)$ than that of Theorem 1. Using the Taylor series expansion of $\log(1 - x)$, it can be shown that $1/\log \left(\frac{v^t}{v^t-1}\right) \leq v^t$. In fact, in the range of values of v and t of interest, $1/\log \left(\frac{v^t}{v^t-1}\right) \approx v^t$. So Theorem 8 guarantees the existence of a covering array with $N \approx \frac{\log \binom{k}{t} + 1}{\log \left(\frac{v^t}{v^t-1}\right)} \approx v^t \log \binom{k}{t} + v^t$ rows.

The argument in the proof of Theorem 8 can be made constructive. It underlies an efficient randomized construction algorithm for covering arrays: In the first stage, construct a random $n \times k$ array with $n \approx v^t \log \binom{k}{t}$ rows; then check if the number of uncovered interactions is at most v^t. If not, randomly generate another $n \times k$ array and repeat the check. In the second stage add at most v^t rows to the partial covering array to cover the remaining interactions. Neither stage needs to store information about individual interactions because we need only count the uncovered interactions in the first stage. The second stage is deterministic and efficient. The first stage has expected polynomial running time; it could be efficiently derandomized in principle using the methods in [6, 7], at the price of the storage of the status of individual interactions.

The proof of Theorem 8 suggests a general "two-stage" construction paradigm, in

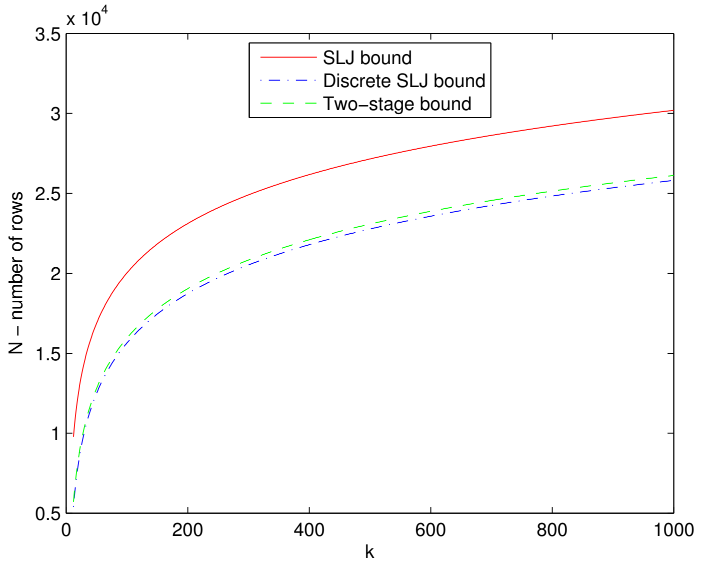


Figure 3.3: Comparison of the Stein-Lovász-Johnson bound, the discrete Stein-Lovász-Johnson bound and the two-stage based bound from Theorem 8. $t = 6$, $v = 3$.

which the first stage uses one strategy to cover almost all of the interactions, and the second uses another to cover the relatively few that remain.

Figure 3.3 compares the two-stage based bound with the Stein-Lovász-Johnson bound and the discrete Stein-Lovász-Johnson bound. In the cases shown, the two-stage bound is much better than the Stein-Lovász-Johnson bound, and not much worse than the discrete Stein-Lovász-Johnson bound. Therefore a purely randomized method (with much smaller memory requirements) produces covering arrays that are competitive with the guarantees from the density algorithm.

3.2.1 Group Action and the Two-stage Bound

We may apply the alteration based two-stage strategy from Section 3.2 to construct covering arrays that are invariant under the action of a permutation group on the set of symbols.

Let Γ be a permutation group on the set of symbols. As discussed in Section 2.2.2, the action of this group partitions the set of t-way interactions into orbits. We construct an array A such that for every orbit, at least one row covers an interaction from that orbit. Then we develop the rows of A over Γ to obtain a covering array that is invariant under the action of Γ. Effort then focuses on covering all the orbits of t-way interactions, instead of the individual interactions.

If Γ acts sharply transitively on the set of symbols (for example, if Γ is a cyclic group of order v) then the action of Γ partitions $\binom{k}{t}v^t$ interactions into $\binom{k}{t}v^{t-1}$ orbits of length v each. Following the lines of the proof of Theorem 8, there exists an $n \times k$ array with $n = \frac{\log\binom{k}{t}+(t-1)\log v+\log\log\left(\frac{v^{t-1}}{v^{t-1}-1}\right)+1}{\log\left(\frac{v^{t-1}}{v^{t-1}-1}\right)}$ that covers at least one interaction from each orbit. Therefore,

$$\mathsf{CAN}(t,k,v) \le v\frac{\log\binom{k}{t} + (t-1)\log v + \log\log\left(\frac{v^{t-1}}{v^{t-1}-1}\right) + 1}{\log\left(\frac{v^{t-1}}{v^{t-1}-1}\right)}. \tag{3.1}$$

We can employ a sharply 2-transitive group action like the action of the Frobenius group when v is a prime power to obtain even smaller covering arrays. Recall that for a prime v, the *Frobenius group* is the group of permutations of $\mathbb{F}_v$ of the form $\{x \mapsto ax + b \,:\, a, b \in \mathbb{F}_v,\, a \neq 0\}$. The action of the Frobenius group partitions the set of t-tuples on v symbols into $\frac{v^{t-1}-1}{v-1}$ orbits of length $v(v-1)$ (full orbits) each and 1 orbit of length v (a short orbit). The short orbit consists of tuples of the form $(x_1, \ldots, x_t)$ where $x_1 = \ldots = x_t$. Therefore, we can obtain a covering array by first constructing an array that covers all the full orbits, and then developing all the rows over the Frobenius group and adding v constant rows. Using the two stage strategy in conjunction with the Frobenius group action we obtain:

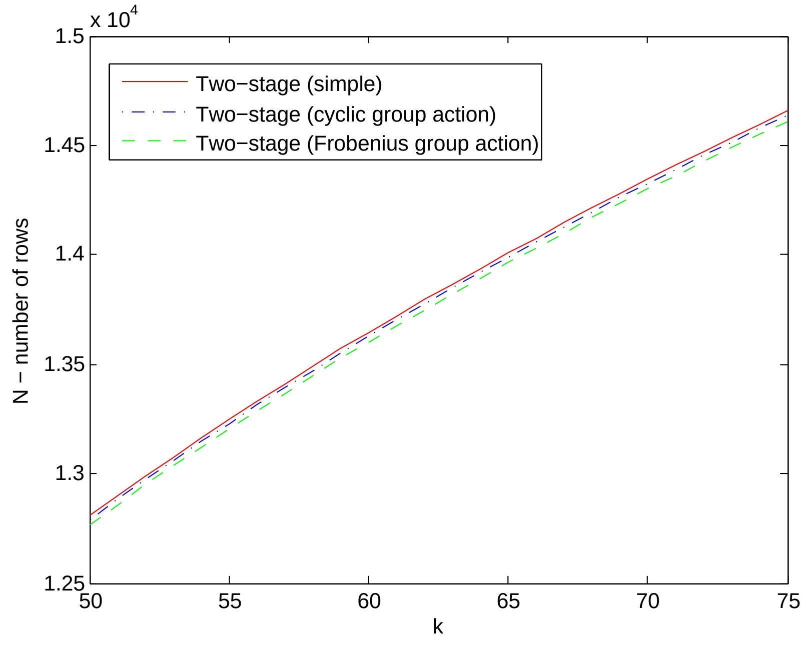


Figure 3.4: Comparison of the simple two-stage bound with the cyclic and the Frobenius two-stage bounds. $t = 6$, $v = 3$ and $50 \le k \le 75$. Group action reduces the required number of rows slightly.

$$\mathsf{CAN}(t,k,v) \le v(v-1)\frac{\log\binom{k}{t} + \log\left(\frac{v^{t-1}-1}{v-1}\right) + \log\log\left(\frac{v^{t-1}}{v^{t-1}-v+1}\right) + 1}{\log\left(\frac{v^{t-1}}{v^{t-1}-v+1}\right)} + v. \qquad (3.2)$$

Figure 3.4 compares the simple two-stage bound with the cyclic and Frobenius two-stage bounds. For $t = 6$, $v = 3$ and $12 \le k \le 100$, the cyclic bound requires form 7 to 21 (on average 16) fewer rows than the simple bound. In the same range, the Frobenius bound requires from 17 to 51 (on average 40) fewer rows.

Chapter 4

ALGORITHMIC CONSTRUCTION OF COVERING ARRAYS

In this chapter, we discuss a number of algorithms for covering array construction. We begin with a short review of most commonly used computational techniques for covering array construction. Then we examine a randomized construction algorithm based on the discrete Stein-Lovász-Johnson paradigm which was introduced in Section 3.1. Conditional expectation based derandomization of this algorithm naturally leads to the density algorithm [6, 7], which is probably the most successful technique for practical construction of covering arrays.

Unlike other practical covering array construction algorithms, we can obtain an upper bound on the size of the covering arrays produced by the density algorithm. An understanding of the strengths and weaknesses of the density algorithm help us to appreciate the motivation of the two-stage framework introduced in Chapter 5. Also, the density algorithm is used as one of the methods in the second stage of a two-stage algorithm discussed in Chapter 5.

In the last section of this chapter, we introduce a new covering array construction technique that is called the biased density algorithm. Similar to the density algorithm, it is a conditional expectation based derandomized version of a randomized algorithm. It is shown that the biased density algorithm performs at least as well as the density algorithm.

4.1 A Brief Review of Algorithms for Covering Array Construction

Available algorithms for the construction of covering arrays are primarily heuristic in nature; indeed exact algorithms have succeeded for very few cases. Computationally intensive metaheuristic search methods such as simulated annealing, tabu search, constraint programming, and genetic algorithms have been employed when the strength is relatively small or the number of factors and levels is small. These methods have established many of the best known bounds on sizes of covering arrays [17], but for larger problems of practical size their time and storage requirements are prohibitive.

For larger problems, the best available methods are greedy. The In Parameter Order (IPO) family of algorithms [38] repeatedly adds one column at a time, and then adds new rows to ensure complete coverage. In this way, at any point in time, the status of $v^t\binom{k-1}{t-1}$ interactions may be stored. Automatic Efficient Test Generator (AETG) [13] pioneered a different method, which greedily selects one row at a time to cover a large number of as-yet-uncovered interactions. They establish that if a row can be chosen that covers the maximum number, a good *a priori* bound on the size of the covering array can be computed.

Unfortunately selecting the maximum is NP-hard, and even if one selects the maximum there is no guarantee that the covering array is the smallest possible [8]. So AETG resorts to a good heuristic selection of the next row by examining the stored status of $v^t\binom{k}{t}$ interactions.

None of the methods so far mentioned therefore guarantee to reach an *a priori* bound. An extension of the AETG strategy, the density algorithm [6, 7, 19], stores additional statistics for all $v^t\binom{k}{t}$ interactions in order to ensure the selection of a good next row, and hence guarantees to produce an array with at most the precomputed number of rows. Variants of the density algorithm have proved to be most effective for problems of moderately large size.

Algorithm 2: A randomized algorithm for covering array construction.

```
Input: $t$ : strength of the covering array, $k$ : number of factors, $v$ : number of
       levels for each factor
Output: $A$ : a $\mathsf{CA}(N; t, k, v)$
1  Set $N := \left\lfloor \frac{\log\binom{k}{t} + t\log v}{\log\left(\frac{v^t}{v^t - 1}\right)} \right\rfloor$;
2  repeat
3      Construct an $N \times k$ array $A$ where each entry is chosen independently and
         uniformly at random from a $v$-ary alphabet;
4      Set covered:= true;
5      for each interaction $\iota \in \mathcal{I}_{t,k,v}$ do
6          if $\iota$ is not covered in $A$ then
7              Set covered:= false;
8              break;
9          end
10     end
11 until covered = true;
12 Output $A$;
```

4.2 The Density Algorithm for Covering Array Construction

To obtain a better understanding of the density algorithm we begin by contrasting it with a basic randomized algorithm. Algorithm 2 shows a simple randomized algorithm for covering array construction. The algorithm constructs an array of a particular size randomly, checks whether all the interactions are covered and repeats until it finds an array that covers all the interactions.

We have already encountered this construction in the proof of Theorem 1. There we have shown that $\mathsf{CA}(N; t, k, v)$ with $N = \frac{\log\binom{k}{t} + t \log v}{\log\left(\frac{v^t}{v^t - 1}\right)}$ is guaranteed to exist. In fact, the probability that the $N \times k$ array constructed in line 3 of Algorithm 2 is a valid covering array is high enough that the expected number of times the loop in line 2 is repeated is a small constant.

An alternative strategy is to add rows one by one instead of constructing the full array at the outset. We start with an empty array, and whenever we add a new row, we ensure that it covers at least the expected number of previously uncovered interactions for a randomly chosen row. The probability that an uncovered interaction is covered by a random row is $1/v^t$. If the number of uncovered interactions is u, then by linearity of expectation, the expected number of newly covered interactions in a randomly chosen row is uv^{-t}. If each row added covers exactly this expected number, we obtain the same number of rows as the SLJ bound, realized in Algorithm 2. But because the actual number of newly covered interactions is always an integer, each added row covers at least $\lceil uv^{-t} \rceil$ interactions. This is especially helpful towards the end when the expected number is a small fraction.

Algorithm 3 follows this strategy. Again the probability that a randomly chosen row covers at least the expected number of previously uncovered interactions is high enough that the expected number of times the row selection loop in line 6 of Algorithm 3 is repeated is bounded by a small constant.

We can obtain an upper bound on the size produced by Algorithm 3 by assuming that each new row added covers exactly $\lceil uv^{-t} \rceil$ previously uncovered interactions. This bound is the *discrete Stein-Lovász-Johnson* (discrete SLJ) bound. Figure 4.1 compares the sizes of covering arrays obtained from the SLJ and the discrete SLJ bounds for different values of k when $t = 6$ and $v = 3$. Consider a concrete example, when $t = 5$, $k = 20$, and $v = 3$. The SLJ bound guarantees the existence of a covering array with $12,499$ rows, whereas the discrete SLJ bound guarantees the existence of

Algorithm 3: A randomized algorithm for covering array construction using the discrete SLJ strategy.

```
   Input: $t$ : strength of the covering array, $k$ : number of factors, $v$ : number of
          levels for each factor
   Output: $A$ : a $\mathsf{CA}(N; t, k, v)$
 1 Let $A$ be an empty array;
 2 Initialize a table $T$ indexed by all $\binom{k}{t} v^t$ interactions, marking every interaction
     "uncovered";
 3 while there is an interaction marked "uncovered" in $T$ do
 4     Let $u$ be the number of interactions marked "uncovered" in $T$;
 5     Set expectedCoverage := $\lceil \frac{u}{v^t} \rceil$;
 6     repeat
 7         Let $r$ be a row of length $k$ where each entry is chosen independently
             and uniformly at random from a $v$-ary alphabet;
 8         Let coverage be the number of "uncovered" interactions in $T$ that are
             covered in row $r$;
 9     until coverage > expectedCoverage;
10     Add $r$ to $A$;
11     Mark all interactions covered by $r$ as "covered" in $T$;
12 end
13 Output $A$;
```

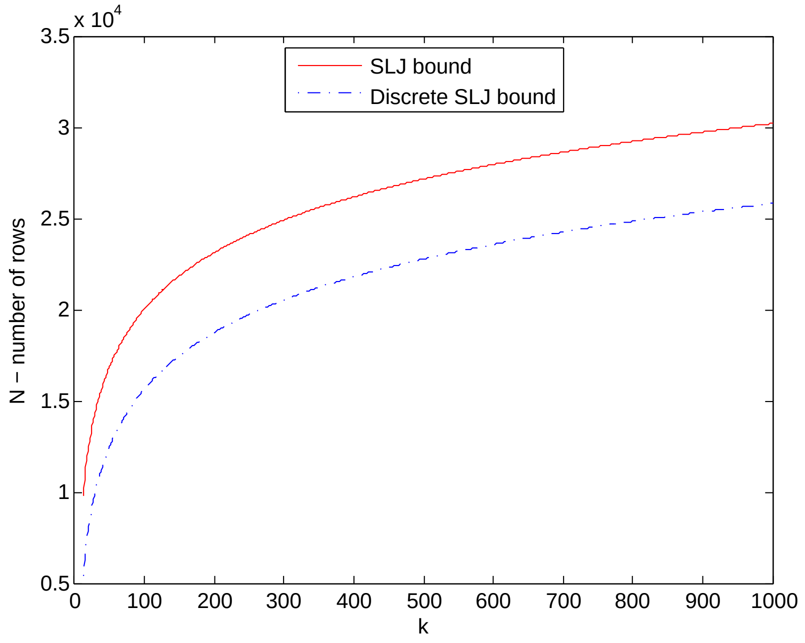


Figure 4.1: Comparison of covering array sizes obtained from the Stein-Lovász-Johnson bound and the discrete Stein-Lovász-Johnson bound for different values of k, when $t = 6$ and $v = 3$.

a covering array with only $8,117$ rows.

The density algorithm (Algorithm 4) replaces the loop at line 6 of Algorithm 3 by a conditional expectation based derandomized method (see Bryce and Colbourn [6, 7], Colbourn [16] for more details). For fixed v and t the density algorithm selects a row efficiently (time polynomial in k) and deterministically. The row is guaranteed to cover at least $\lceil uv^{-t} \rceil$ previously uncovered interactions. In practice, for small values of k the density algorithm works well, often covering many more interactions than the minimum. Many of the currently best known $\mathsf{CAN}(t, k, v)$ upper bounds are obtained by the density algorithm in combination with various post-optimization techniques [17].

However, the practical applicability of Algorithm 3 and the density algorithm is

Algorithm 4: The density algorithm for selection of a row.

```
Input: $X$ : the set of uncovered interactions, $\Sigma$ : the alphabet of symbols with
       $|\Sigma| = v$, $t$ : strength of the covering array, $k$ : number of factors.
Output: $r$ : the next row of the covering array.
Let $r := (*, \ldots, *)$;
for $i = 1$ to $k$ do
    Choose a $j$ such that $r_j = *$;
    Let $maxExpCov := 0$, and $\sigma^* = null$;
    foreach $\sigma \in \Sigma$ do
        Let $z := r$, $z_j := \sigma$, and $expCov := 0$;
        foreach $c = \{c_1, \ldots, c_t\}$ with $c_l < c_{l+1}$ for $1 \le l \le t$ and $j \in c$ do
            Let $F := \{\gamma \in c : z_\gamma = *\}$, $\overline{F} := c \setminus F$, and $count := 0$;
            foreach $s \in \Sigma^k$ do
                // $\iota = (c, s)$ agrees with $z$ if $c_l \in \overline{F}$ implies $z_{c_l} = s_l$.
                if $\iota = (c, s)$ agrees with $z$ and $\iota \in X$ then
                    Set $count := count + 1$;
                end
            end
            Set $expCov := expCov + count/|\Sigma|^{|F|}$;
        end
        if $expCov > maxExpCov$ then
            Set $maxExpCov := expCov$, and $\sigma^* = \sigma$;
        end
    end
    Set $r_j := \sigma^*$;
end
```

limited by the storage of the table T, representing each of the $\binom{k}{t}v^t$ interactions. Even when $t = 6$, $v = 3$, and $k = 54$, there are 18,828,003,285 6-way interactions. This huge memory requirement renders the density algorithm impractical for rather small values of k when $t \in \{5, 6\}$ and $v \geq 3$. We present an idea to circumvent this large requirement for memory and develop it in full in Chapter 5.

4.3 The Biased Density Algorithm

In this section, we present an algorithm for construction of covering arrays, called the *biased density algorithm.* This method, like the density algorithm, belongs to the same discrete Stein-Lovász-Johnson bound paradigm. However, it differs from the density algorithm in the way it fixes the entries in a row. The biased density algorithm is guaranteed to construct covering arrays that are no bigger than the ones constructed by the density algorithm. Although under certain circumstances the biased density algorithm does produce smaller covering arrays than the density algorithm, it is not suited for the practical construction of covering arrays with higher k values due to its quadratic running time. Therefore, the interest in this method is primarily theoretical in nature.

First, we introduce a new randomized algorithm that guarantees a better expected coverage than Algorithm 3. The biased density algorithm is basically the conditional expectation based derandomized version of this randomized algorithm. While selecting a row of a covering array, the derandomized version selects the next interaction to cover in the row, instead of the next entry, by maximizing the conditional expectation of the number of newly covered interactions. This derandomization strategy is described in detail in Section 4.3.2.

4.3.1 A New Randomized Algorithm for Covering Array Construction

Algorithm 3 chooses each row $r \in \Sigma^k$ with equal probability. We call this method *uniform random* row selection strategy. Note that the probability of choosing a row that has already been chosen is not zero in this case. But such a choice results in zero new coverage. We can improve the expected number of newly covered interactions by restricting our choice only to those rows that provide new coverage. In fact, if the rows with higher number of uncovered interactions are chosen with higher probability then the expected number of newly covered interactions would be higher than the uniform random case. We design a sample space that achieves exactly this objective.

We construct the rows of the covering array sequentially. Suppose a number of rows have been already constructed and let $X \neq \phi$ be the set of interactions that are not yet covered. To construct a row we start with an empty row and then fix symbols for different entries in a certain sequence. Let ρ be the set of entries that have been fixed; initially $\rho = \phi$.

For each interaction $\iota = (c, s) \in X$, where $c \in \binom{[k]}{t}$ and $s \in \Sigma^t$, let $N(\iota)$ denote the set of rows that may cover this interaction, and let $n(\iota) := |N(\iota)|$. Let us define $M := \sum_{\iota \in X} n(\iota)$. Initially, when $\rho = \phi$, we have $n(\iota) = v^{k-t}$, and $M = |X| \cdot v^{k-t}$. Similarly, let $m(r)$ denote the number of new interactions that are covered if the row $r \in \Sigma^k$ is selected next. We have $\sum_{r \in \Sigma^k} m(r) = \sum_{\iota \in X} n(\iota) = M$. Algorithm 5 selects the row r with probability $m(r)/M$.

The probability that a row $r \in \Sigma^k$ is chosen by Algorithm 5 is given by

$$\sum_{\iota \in X \,:\, r \in N(\iota)} \frac{1}{|X|} \times \frac{1}{v^{k-t}} = \frac{m(r)}{M}.$$

The expected number of new interactions that are covered by a row chosen by this algorithm is $\sum_{r \in \Sigma^k} m(r).\frac{m(r)}{M} \geq \sum_{r \in \Sigma^k} \frac{m(r)}{v^k}$. Therefore, the expected coverage of

Algorithm 5: A randomized algorithm for the selection of a row.

```
Input: $X$ : the set of uncovered interactions, $\Sigma$: the alphabet of symbols with
        $|\Sigma| = v$, $t$ : strength of the covering array, $k$ : number of factors.
Output: $r$ : the next row of the covering array.
1 Let $r := (*, \ldots, *)$;
2 Choose an interaction $\iota = (c, s)$ uniformly at random from the set of uncovered
    interactions $X$;
3 for $i = 1$ to $t$ do
4     Set $r_{c_i} := s_i$;
5 end
  // This step essentially samples $r$ uniformly at random from $N(\iota)$.
6 Select the remaining entries in $r$ uniformly at random from $\Sigma$;
```

this algorithm is better than the uniform random row selection method. We call this method the *biased randomized* row selection method because it is biased in the favor of the rows that cover a higher number of new interactions.

4.3.2 Strategy for Derandomization

Derandomizing Step 2 of the biased randomized algorithm (Algorithm 5) is straightforward. To derandomize Step 6 we may apply the usual technique of fixing each individual entry, but instead we define an alternative random process for Step 6 with provably better performance.

Let r' be the new row and ρ be the set of entries of r' for which the symbols have been fixed. Suppose we have $\rho \neq \phi$. Instead of picking a row uniformly at random from the set of rows that agree with the choice of symbols for ρ, we pick a row with probability proportional to the number of newly covered interactions.

Let X' denote the set of interactions that agree with the choice of symbols for ρ, but are not already covered. For each interaction $\iota = (c, s) \in X'$, as before, let $N(\iota)$ denote the set of rows that agree with ρ and cover ι, and let $n(\iota) := |N(\iota)|$. So $n(\iota) = v^{k-|\rho\cup c|}$. Let $M := \sum_{\iota \in X'} n(\iota)$. Now, $n(\iota)$ is not same for all interactions as was the case previously.

Let r be a row that agrees with the choice of symbols for ρ. For such a row r let $m(r)$ denote the number of new interactions covered by it. For any other row r that does not agree with the choice of symbols for ρ, let us set $m(r) = 0$. Once again, we have $\sum_{r\in\Sigma^k} m(r) = \sum_{\iota\in X'} n(\iota) = M$. The following randomized algorithm picks a row r, that agrees with the choice of symbols for ρ, with probability $m(r)/M$.

1. Choose an interaction $\iota \in X'$ with probability $n(\iota)/M$. Fix $r'_{c_i} = s_i$ for all $c_i \in c\backslash\rho$.
2. Among all the rows that agree with the choice of symbols for $\rho \cup c$, choose one uniformly at random.

In this method, the probability that a row r that agrees with choice of symbols for ρ is chosen is given by

$$\sum_{\iota\in X' \,:\, r\in N(\iota)} \frac{n(\iota)}{M} \times \frac{1}{n(\iota)} = \frac{m(r)}{M}.$$

We can replace Step 6 of Algorithm 5 by recursively applying this new randomized selection method until all the entries in r' are fixed (or X' becomes empty). Algorithm 6 presents the full recursive version of the row selection algorithm. The expected coverage of this recursive randomized algorithm is at least as good as Algorithm 5. However, in the next section we provide a derandomized version of Algorithm 5, and not the recursive version discussed above. The subtle difference is that for derandomizing Step 2 of Algorithm 5 we assume Step 6 of Algorithm 5, not the recursive version. It is only while derandomizing step 6 we replace it by this recursive process.

Algorithm 6: A recursive randomized algorithm for the selection of a row.

```
Input: $X$ : the set of uncovered interactions, $\Sigma$: the alphabet of symbols with
       $|\Sigma| = v$, $t$ : strength of the covering array, $k$ : number of factors.
Output: $r$ : the next row of the covering array.
1  Let $r := (*, \dots, *)$;
2  while $X \neq \phi$ do
3      Choose an interaction $\iota = (c, s)$ from the set of uncovered interactions $X$
         with probability $n(\iota)/M$;
4      for $i = 1$ to $t$ do
5          Set $r_{c_i} := s_i$;
6      end
7      foreach $\iota' = (c', s') \in X$ do
8          if $\iota'$ does not agree with $r$ then
               // i.e., there is an $i$ such that $r_{c'_i} \neq s'_i$.
9              Delete $\iota'$ from $X$;
10         end
11     end
12 end
13 Fix the remaining entries of $r$, if there are any, with symbols from $\Sigma$ drawn
     uniformly at random;
```

4.3.3 The Derandomized Algorithm

In this section we present a conditional expectation based derandomized version of the biased randomized row selection algorithm (Algorithm 5). As mentioned before, this algorithm is called the *biased density* algorithm.

A naïve derandomization of Algorithm 5 would result in an exponential time algorithm (See Algorithm 15 and 16 in Appendix B). To obtain a polynomial time algorithm we need to avoid extra counting. We present a way of achieving that in Algorithm 7.

Computational Results

Table 4.1 compares the size of the covering arrays produced by the uniform random (Algorithm 2), the biased random (Algorithm 6), the density (Algorithm 4), and the biased density (Algorithm 7) row selection algorithms for $t = 6$, $v = 2$ and $7 \leq k \leq 10$. We run the randomized algorithms $10,000$ times independently on each instance, and choose the smallest covering arrays. As the results show, the biased density produces shorter arrays than the density algorithm when $k = 9, 10$. More results are presented in Table C.1 of Appendix C.

4.3.4 Runtime Analysis

The while loop at line 3 of the biased density algorithm is executed at most $(k - t + 1)$ times (the first selected interaction fixes t entries, and each subsequent selection of interactions fixes at least one new entry). The for loop at the line 5 is executed $|X| \leq v^t \binom{k}{t}$ times.

An upper bound on the execution of the loop at the line 2 of Algorithm 8 is $|X|$. So the time complexity of the algorithm is of the order of $(k - t + 1) \cdot |X|^2 \leq$

Algorithm 7: The biased density algorithm for the selection of a row.

```
Input: $X$ : the set of uncovered interactions, $\Sigma$: the alphabet of symbols with
       $|\Sigma| = v$, $t$ : strength of the covering array, $k$ : number of factors.
Output: $r$ : the next row of the covering array.
1  Let $r := (*, \ldots, *)$;
2  Let $\rho$ denote the set of entries in $r$ that have been fixed. Initially, $\rho = \phi$;
3  while $X \neq \phi$ do
4      Let $maxExpCov := 0$, and $\iota^* := (null, null)$;
5      foreach $\iota = (c, s) \in X$ do
6          Let $expCov$ store the conditional expected coverage computed by
             Algorithm 8;
7          if $expCov > maxExpCov$ then
8              Set $maxExpCov := expCov$, and $\iota^* := \iota$, i.e., set $c^* = c$ and $s^* = s$;
9          end
10     end
11     for $i = 1$ to $t$ do
12         Set $r_{c_i^*} := s_i^*$;
13     end
14     foreach $\iota' = (c', s') \in X$ do
15         if $\iota'$ does not agree with $r$ or is already covered by $r$ then
16             Delete $\iota'$ from $X$;
17         end
18     end
19 end
20 Fix the remaining entries of $r$, if there are any, randomly from $\Sigma$;
```

Algorithm 8: Computation of conditional expected coverage for the biased density algorithm.

```
Input: $X$ : the set of uncovered interactions, $\iota = (c, s)$: the interaction that is
       selected next, $\rho$ : the set of entries that have been fixed in the current
       row, $\Sigma$: the alphabet of symbols with $|\Sigma| = v$, $t$ : strength of the
       covering array, $k$ : number of factors.
Output: The conditional expected coverage if the interaction $\iota$ is selected
        next.
1 Let $count := 0$;
2 foreach $\iota' = (c', s') \in X$ do
3     if $\iota'$ agrees with $\iota$ then
          // i.e. for all $1 \le i, j \le t$, $c_i = c'_j$ implies $s_i = s'_j$.
4         Set $count := count + |\Sigma|^{k-|\rho \cup c \cup c'|}$;
          // $\iota$ and $\iota'$ together with current assignment $\rho$ is covered by
             $|\Sigma|^{k-|\rho \cup c \cup c'|}$ different rows.
5     end
6 end
7 Return $count/|\Sigma|^{k-|\rho \cup c|}$;
```

$(k - t + 1) \cdot \left(v^t \binom{k}{t}\right)^2$. Therefore, the time complexity of constructing a row using the biased density algorithm is approximately $k \cdot |X|^2$ where X is the set of as yet uncovered interactions.

A similar estimate for the density algorithm provides a time complexity that is approximately $k \cdot v \cdot \binom{k-1}{t-1} \cdot v^{t-1} = k \cdot v^t \cdot \binom{k-1}{t-1} \le k \cdot |X|$. So in terms of the time complexity, the biased density algorithm is not as efficient as the density algorithm. In fact, as k grows it quickly becomes infeasible to run the biased density algorithm

k	Uniform Random	Biased Random	Density	Biased Density
7	218	92	86	88
8	280	136	114	117
9	362	176	145	143
10	431	214	172	161

Table 4.1: Comparison of the uniform random (Algorithm 2), the biased random (Algorithm 6), the density (Algorithm 4), and the biased density (Algorithm 7) row selection algorithms. $t = 6$, $v = 2$. $10,000$ independent runs of the randomized algorithm are used, and the best result is reported.

to construct complete covering arrays. However, when relatively smaller number of interactions are left, the biased density algorithm runs reasonably fast and hence has some practical value. Section 4.3.5 discusses a few variants of the biased density algorithm that exploit this observation.

4.3.5 Improving the Running Time

The biased density algorithm produces shorter arrays than the density algorithm in practice, but also takes more time and becomes impractical as k grows larger. To circumvent its quadratic running time, we introduce two new variations of the biased density algorithm: the method of pessimistic estimators, and a hybrid two stage algorithm where the density algorithm in the first stage is followed by the biased density algorithm in the second stage. Next, we develop these ideas and compare these methods with the density algorithm.

The Method of Pessimistic Estimation

Instead of computing the exact value of the conditional expected coverage for each remaining interaction, we may compute a lower bound, a *pessimistic estimate* of the conditional expectation, and use this lower bound to make the greedy interaction selection. The method works as follows: we maintain a table of dimension $|\mathcal{I}_{t,k,v}| \times (t+1)$, where each row is indexed by the interactions from the set of all possible t-way interactions $\mathcal{I}_{t,k,v}$. For $\iota \in \mathcal{I}_{t,k,v}$ and $0 \leq i \leq t$, $A(\iota, i)$ maintains a lower bound on the number of interactions which share exactly i columns with ι and agree with it. After each new row is added, we update this table by over-counting the number of interactions that share i columns with ι, agree with it, and are covered in the most recently added row.

Table 4.2 compares the size of the covering arrays obtained from the density algorithm and the biased density algorithm with pessimistic estimates for 3 different combination of t and v values. We observe that for higher values of k, the method of pessimistic estimations produces marginally smaller covering arrays compared to density algorithm. The running time of this variation of the biased density algorithm is comparable to that of the density algorithm.

The Hybrid Method

In this variation, we start by adding rows to an array using the density algorithm. Once the number of uncovered interactions falls below a certain threshold we switch over to the biased density algorithm. The switchover threshold is set in such a way that the running time of the biased density is within a constant factor of the running time of the density algorithm.

This hybrid strategy often performs better than the density algorithm. Table 4.3

k	D	PE
15	306	306
16	318	320
17	332	331
18	344	341
19	355	352
20	365	363
21	378	375
22	385	385
23	394	392
25	401	402

(a) $t = 4$, $v = 3$

k	D	PE
11	814	804
12	884	877
13	955	948
14	1018	1007
15	1073	1067
16	1136	1131
17	1193	1178
18	1239	1234
19	1287	1282
20	1340	1324

(b) $t = 5$, $v = 3$

k	D	PE
10	2166	2148
11	2485	2490
12	2778	2766
13	3054	3045
14	3313	3309
15	3559	3540
16	3785	3788
17	4015	3995
18	4238	4215
19	4439	4417

(c) $t = 6$, $v = 3$

Table 4.2: Comparison of the covering array sizes constructed by the density algorithm (D) and the biased density algorithm with pessimistic estimates (PE).

compares the density algorithm against the hybrid strategy for $t = 6$, $v = 3$ and k in the range $15 - 19$. In this range the presently best known results are obtained by running post optimization methods on the covering arrays constructed by the density algorithm. Because the hybrid strategy constructs smaller arrays compared to the density algorithm for all the k values in this range, it is not unreasonable to expect that the covering arrays produced by running post-optimization on these arrays may eventually beat the presently best known results.

k	density	hybrid	best-known
15	3410	3379	3223
16	3652	3597	3435
17	3855	3812	3654
18	4072	4026	3846
19	4273	4222	4051

Table 4.3: Comparison of the covering array sizes obtained by the density and the hybrid algorithm for $t = 6$, $v = 3$. In the density algorithm each row is selected from multiple candidate rows generated using the conditional expectation. All the best known results are obtained by running post-optimization on the arrays generated by the density algorithm.

Chapter 5

TWO-STAGE FRAMEWORK FOR COVERING ARRAY CONSTRUCTION

In this chapter, we describe a new algorithmic framework for covering array construction, called the *two-stage framework*. When $k > \max(t+1, v+2)$, a covering array must cover some interactions more than once, for if not they are orthogonal arrays [33]. Treating the rows of a covering array in a fixed order, each row covers some number of interactions not covered by any earlier row. For a variety of known constructions, the initial rows cover many new interactions, while the later ones cover very few [8]. Comparing this rate of coverage for a purely random method and for one of the sophisticated search techniques, one finds little difference in the initial rows, but very substantial differences in the final ones. This suggests strategies to build the covering array in stages, investing more effort as the number of remaining uncovered interactions declines. This observation forms the core of the ideas investigated in this chapter.

In the first stage of our two-stage framework, a randomized row construction method builds a specified number of rows to cover all but a small number of interactions. As we see later, by dint of being randomized this stage uses very little memory. The second stage is a more sophisticated search that adds a few rows to cover the remaining uncovered interactions. We choose search algorithms whose time and space requirements depend on the number of interactions to be covered, to profit from the fact that only few interactions remain uncovered. By mixing randomized and deterministic methods, we hope to retain the fast execution and small storage of the randomized methods while gaining the efficiency of the deterministic search techniques.

We introduce a number of algorithms within this two-stage framework. Here our focus is on practical consequences. The two-stage algorithms are indeed quite efficient for higher strengths ($t \in \{5, 6\}$) and relatively smaller number of levels ($v \in \{3, 4, 5, 6\}$), when the number of factors k is moderately high (approximately in the range of $20-80$ depending on the value of t and v). In fact, for many combinations of t, k and v values the two-stage algorithms beat the previously best known bounds.

Torres-Jimenez et al. [54] explore a related two-stage strategy. In their first stage, an in-parameter-order greedy strategy (as used in ACTS [38]) adds a column to an existing array; in their second stage, simulated annealing is applied to cover the remaining interactions. They apply their methods for $t = v = 3$, when the storage and time requirements for both stages remain acceptable. In addition to the issues in handling larger strengths, their methods provide no *a priori* bound on the size of the resulting array. In contrast with their methods, ours provide a guarantee prior to execution with much more modest storage and time.

The discussion in this chapter is based on the presentation in Sarkar and Colbourn [47].

5.1 Why Does a Two Stage Based Strategy Make Sense?

Compare the two extremes: The density algorithm and Algorithm 2. On one hand, Algorithm 2 does not suffer from any substantial storage restriction, but appears to generate many more rows than the density algorithm. On the other hand, the density algorithm constructs fewer rows for small values of k but becomes impractical when k is moderately large. One wants algorithms that behave like Algorithm 2 in terms of memory, but yield a number of rows competitive with the density algorithm.

For $t = 6$, $k = 16$, and $v = 3$, Figure 5.1 compares the coverage profile for the density algorithm and Algorithm 2. We plot the number of newly covered interactions

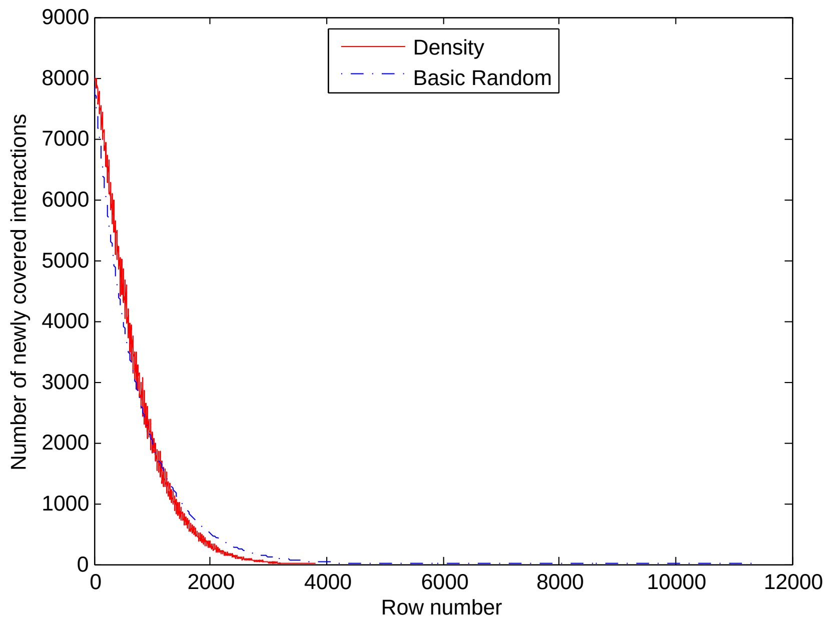


Figure 5.1: For $t = 6$, $k = 16$ and $v = 3$, the actual number of newly covered interactions of the density algorithm and the expected number of newly covered interactions in a random array.

for each row in the density algorithm, and the expected number of newly covered interactions for each row for Algorithm 2. The qualitative features exhibited by this plot are representative of the rates of coverage for other parameters.

Two key observations are suggested by Figure 5.1. First, the expected coverage in the initial random rows is similar to the rows chosen by the density algorithm. In this example, the partial arrays consisting of the first 1000 rows exhibit similar coverage, yet the randomized algorithm needed no extensive bookkeeping. Secondly, as later rows are added, the judicious selections of the density algorithm produce much larger coverage per row than Algorithm 2. Consequently, it appears sensible to invest few computational resources on the initial rows while making more careful selections in the later ones. This forms the blueprint of our general *two-stage* algorithmic framework shown in Algorithm 9.

Algorithm 9: The general two-stage framework for covering array construction.

```
Input: $t$ : strength of the required covering array, $k$ : number of factors, $v$ :
       number of levels for each factor
Output: $A$ : a $\mathsf{CA}(N;t,k,v)$
1 Choose a number $n$ of rows and a number $\rho$ of interactions;
  // First Stage
2 Use a randomized algorithm to construct an $n \times k$ array $A'$;
3 Ensure that $A'$ covers all but at most $\rho$ interactions;
4 Make a list $L$ of interactions that are not covered in $A'$ ($L$ contains at most $\rho$
    interactions);
  // Second Stage
5 Use a deterministic procedure to add $N - n$ rows to $A'$ to cover all the
    interactions in $L$;
6 Output $A$;
```

A specific covering array construction algorithm results by specifying the randomized method in the first stage, the deterministic method in the second stage, and the computation of n and ρ. Any such algorithm produces a covering array, but we wish to make selections so that the resulting algorithms are practical while still providing a guarantee on the size of the array. In Section 5.2 we describe different algorithms from the two-stage family, determine the size of the partial array to be constructed in the first stage, and establish upper bound guarantees. In Section 5.3 we explore how good the algorithms are in practice.

5.2 Two-stage Framework

For the first stage we consider two methods:

Rand	the basic randomized algorithm,
MT	the Moser-Tardos type algorithm.

We defer the development of method MT until Section 6.2. Method Rand uses a simple variant of Algorithm 2, choosing a random $n \times k$ array.

For the second stage we consider four methods:

Naive	the naïve strategy, one row per uncovered interaction,
Greedy	the online greedy coloring strategy,
Den	the density algorithm,
Col	the graph coloring algorithm.

Using these abbreviations, we adopt a uniform naming convention for the algorithms: $\mathsf{TS}\,\langle A, B\rangle$ is the algorithm in which A is used in the first stage, and B is used in the second stage. For example, $\mathsf{TS}\,\langle \mathsf{MT}, \mathsf{Greedy}\rangle$ denotes a two-stage algorithm where the first stage is a Moser-Tardos type randomized algorithm and the second stage is a greedy coloring algorithm. Later when the need arises we refine these algorithm names.

5.2.1 *One Row Per Uncovered Interaction in the Second Stage*

In the second stage, each of the uncovered interactions after the first stage is covered using a new row. Algorithm 10 describes the method in more detail.

This simple strategy improves on the basic randomized strategy when n is chosen judiciously. For example, when $t = 6$, $k = 54$ and $v = 3$, Algorithm 2 constructs a covering array with $17,236$ rows. Figure 5.2 plots an upper bound on the size of the covering array against the number n of rows in the partial array. The smallest

Algorithm 10: Naïve two-stage algorithm ($\mathsf{TS}\langle\mathsf{Rand},\mathsf{Naive}\rangle$).

```
Input: $t$ : strength of the covering array, $k$ : number of factors, $v$ : number of
       levels for each factor
Output: $A$ : a $\mathsf{CA}(N; t, k, v)$
1  Let $n := \frac{\log\binom{k}{t} + t\log v + \log\log\left(\frac{v^t}{v^t-1}\right)}{\log\left(\frac{v^t}{v^t-1}\right)}$;
2  Let $\rho = \frac{1}{\log\left(\frac{v^t}{v^t-1}\right)}$;
3  repeat
4      Let $A$ be an $n \times k$ array where each entry is chosen independently and
         uniformly at random from a $v$-ary alphabet;
5      Let uncovNum := 0 and unCovList be an empty list of interactions;
6      Set covered:= true;
7      for each interaction $\iota \in \mathcal{I}_{t,k,v}$ do
8          if $\iota$ is not covered in $A$ then
9              Set uncovNum :=uncovNum+1;
10             Add $\iota$ to unCovList;
11             if uncovNum $> \rho$ then
12                 Set covered:= false;
13                 break;
14             end
15         end
16     end
17 until covered= true;
18 for each interaction $\iota \in$uncovList do
19     Add a row to $A$ that covers $\iota$;
20 end
21 Output $A$;
```

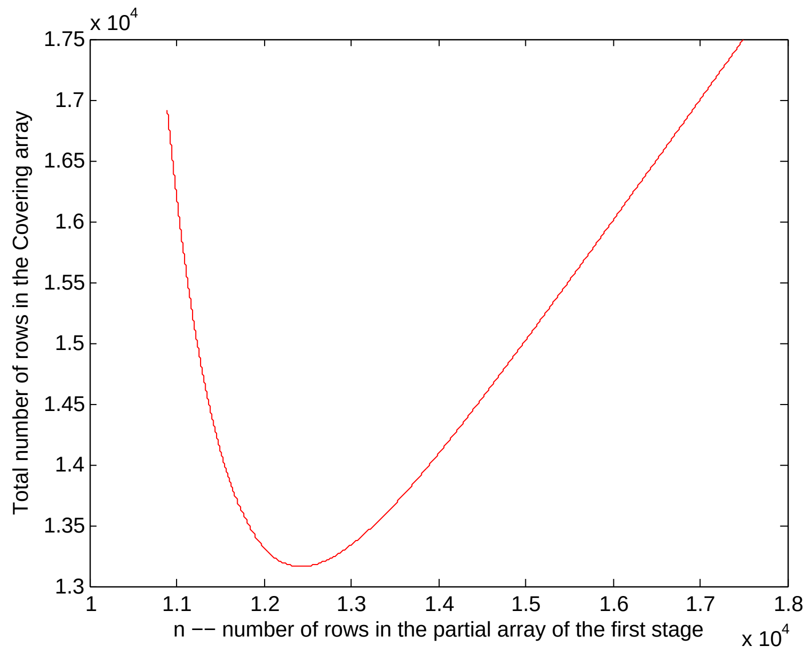


Figure 5.2: An upper bound on the size of the covering array against n, the size of the partial array constructed in the first stage when $t = 6, k = 54$, and $v = 3$, with one new row added per uncovered interaction in the second stage. The minimum size of $13,162$ is obtained when $n = 12,402$. Algorithm 2 requires $17,236$ rows, and the currently best known covering array has $17,197$ rows.

covering array is obtained when $n = 12,402$ which, when completed, yields a covering array with at most $13,162$ rows—a big improvement over Algorithm 2.

Theorem 8 from Chapter 3 tells us the optimal value of n for this strategy in general: $n = \frac{\log\binom{k}{t} + t\log v + \log\log\left(\frac{v^t}{v^t-1}\right)}{\log\left(\frac{v^t}{v^t-1}\right)}$. The expected number of uncovered interactions is exactly $\rho = 1/\log\left(\frac{v^t}{v^t-1}\right)$.

Figure 5.3 compares SLJ, discrete SLJ and two-stage bounds for $k \leq 100$, when $t = 6$ and $v = 3$. The two-stage bound does not deteriorate in comparison to the discrete SLJ bound as k increases; it consistently takes only 307–309 more rows. Thus

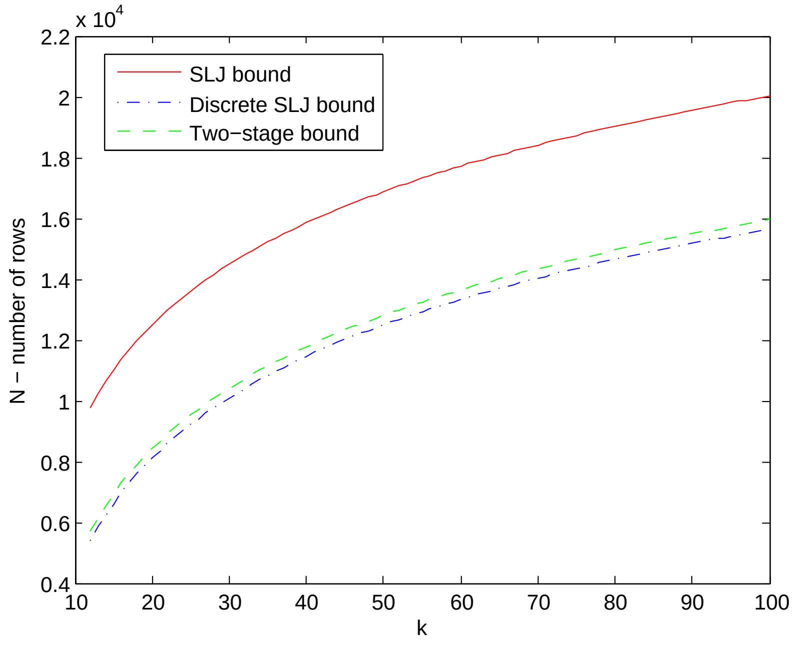


Figure 5.3: Comparison of covering array sizes obtained from the Stein-Lovász-Johnson bound, the discrete Stein-Lovász-Johnson bound and the two-stage bound for $k \leq 100$, when $t = 6$ and $v = 3$. For this range of k, the two-stage bound requires 307–309 more rows than the discrete SLJ bound, that is, 2–6% more rows.

when $k = 12$ the two-stage bound requires only 6% more rows and when $k = 100$ only 2% more rows than the discrete SLJ bound.

To ensure that the loop in line 7 of Algorithm 10 does not repeat too many times we need to know the probability with which a random $n \times k$ array leaves at most ρ interactions uncovered. Using Chebyshev's inequality and the second moment method developed in [2, Chapter 4], we next show that in a random $n \times k$ array the number of uncovered interactions is almost always close to its expectation, i.e., $\binom{k}{t} v^t \left(1 - \frac{1}{v^t}\right)^n$. Substituting the value of n from line 1, this expected value is equal to ρ, as in line 2. Therefore, the probability that a random $n \times k$ array covers the desired number of interactions is constant, and the expected number of times the loop in line 7 is repeated is also a constant (around 2 in practice).

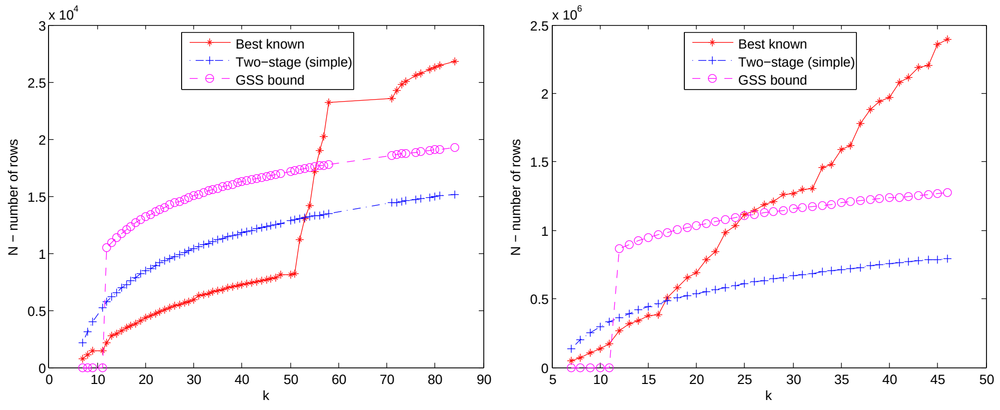
Because the theory of the second moment method is developed in considerable detail in Alon and Spencer [2], here we briefly mention the relevant concepts and results. Suppose that $X = \sum_{i=1}^{m} X_i$, where X_i is the indicator random variable for event A_i for $1 \leq i \leq m$. For indices i, j, we write $i \sim j$ if $i \neq j$ and the events A_i, A_j are not independent. Also suppose that $X_1, \ldots, X_m$ are *symmetric*, i.e., for every $i \neq j$ there is a measure preserving mapping of the underlying probability space that sends event A_i to event A_j. Define $\Delta^* = \sum_{j \sim i} \Pr[A_j | A_i]$. Then by [2, Corollary 4.3.4]:

Lemma 9. [2] *If* $\mathsf{E}[X] \to \infty$ *and* $\Delta^* = o(\mathsf{E}[X])$ *then* $X \sim \mathsf{E}[X]$ *almost always.*

In our case, A_i denotes the event that the ith interaction is *not* covered in a $n \times k$ array where each entry is chosen independently and uniformly at random from a v-ary alphabet. Then $\Pr[X_i] = \left(1 - \frac{1}{v^t}\right)^n$. Because there are $\binom{k}{t} v^t$ interactions in total, by linearity of expectation, $\mathsf{E}[X] = \binom{k}{t} v^t \left(1 - \frac{1}{v^t}\right)^n$, and $\mathsf{E}[X] \to \infty$ as $k \to \infty$.

Distinct events A_i and A_j are independent if the ith and jth interactions share no column. Therefore, the event A_i is not independent of at most $t\binom{k}{t-1}$ other events A_j. So $\Delta^* = \sum_{j \sim i} \Pr[A_j | A_i] \leq \sum_{j \sim i} 1 \leq t\binom{k}{t-1} = o(\mathsf{E}[X])$ when v and t are constants. By Lemma 9, the number of uncovered interactions in a random $n \times k$ array is close to the expected number of uncovered interactions. This guarantees that Algorithm 10 is an efficient randomized algorithm for constructing covering arrays with a number of rows upper bounded by Theorem 8.

In keeping with the general two-stage framework, Algorithm 10 does not store the coverage status of each interaction. We only need store the interactions that are uncovered in A, of which there are at most $\rho = \frac{1}{\log\left(\frac{v^t}{v^t - 1}\right)} \approx v^t$. This quantity



(a) $t = 6$, $v = 3$ (b) $t = 6$, $v = 6$

Figure 5.4: Comparison of the GSS bound and the two-stage bound with the currently best known results

depends only on v and t and is independent of k, so is effectively a constant that is much smaller than $\binom{k}{t}v^t$, the storage requirement for the density algorithm. Hence the algorithm can be applied to a higher range of k values.

Although Theorem 2 provides asymptotically tighter bounds than Theorem 8, in a range of k values that are relevant for practical application, Theorem 8 provides better results. Figure 5.4 compares the bounds on $\mathsf{CAN}(t, k, v)$ with the currently best known results.

5.2.2 The Density Algorithm in the Second Stage

Next, we apply the density algorithm in the second stage. Figure 5.5 plots an upper bound on the size of the covering array against the size of the partial array constructed in the first stage when the density algorithm is used in the second stage, and compares it with $\mathsf{TS}\,\langle\mathsf{Rand}, \mathsf{Naive}\rangle$. The size of the covering array decreases as n decreases. This is expected because, with smaller partial arrays, more interactions remain for the second stage to be covered by the density algorithm. In fact, if we

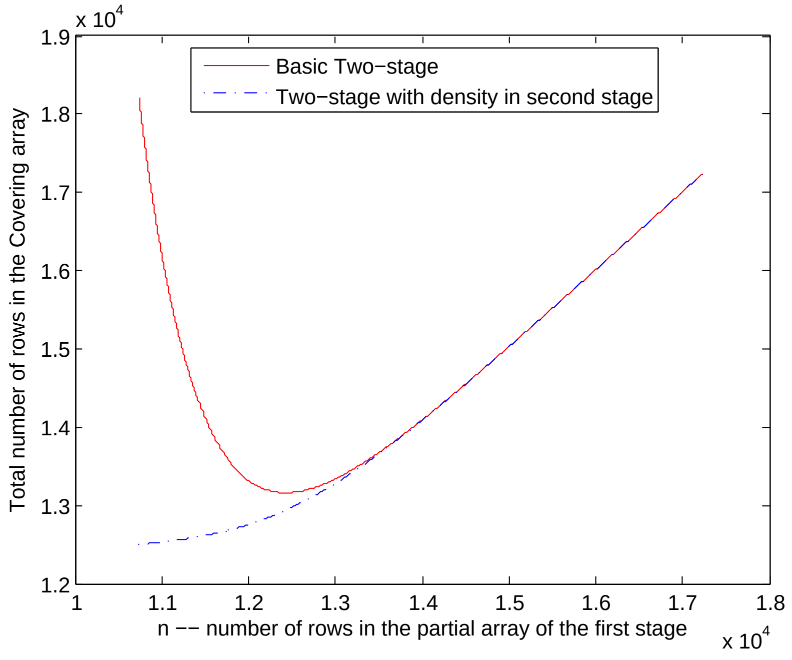


Figure 5.5: Comparison of covering array sizes from two-stage algorithms with Den and Naive in the second stage. With Den there is no minimum point in the curve; the size of the covering array keeps decreasing as we leave more uncovered interactions for the second stage.

cover all the interactions using the density algorithm (as when $n = 0$) we would get an even smaller covering array. However, our motivation was precisely to avoid doing that. Therefore, we need a ”cut-off” for the first stage.

We are presented with a trade-off. If we construct a smaller partial array in the first stage, we obtain a smaller covering array overall. But we then pay for more storage and computation time for the second stage. To appreciate the nature of this trade-off, look at Figure 5.6, which plots an upper bound on the covering array size and the number of uncovered interactions in the first stage against n. The improvement in the covering array size plateaus after a certain point. The three horizontal lines indicate $\rho\,(\approx v^t)$, 2ρ and 3ρ uncovered interactions in the first stage. (In the naïve method of Section 5.2.1, the partial array after the first stage leaves at most ρ uncovered interactions.) In Figure 5.6 the final covering array size appears to plateau when the number of uncovered interactions left by the first stage is around 2ρ. After that we see diminishing returns — the density algorithm needs to cover more interactions in return for a smaller improvement in the covering array size.

Let r be the maximum number of interactions allowed to remain uncovered after the first stage. Then r can be specified in the two-stage algorithm. To accommodate this, we denote by $\mathsf{TS}\langle A, B; r\rangle$ the two-stage algorithm where A is the first stage strategy, B is the second stage strategy, and r is the maximum number of uncovered interactions after the first stage. For example, $\mathsf{TS}\langle \mathsf{Rand}, \mathsf{Den}; 2\rho\rangle$ applies the basic randomized algorithm in the first stage to cover all but at most 2ρ interactions, and the density algorithm to cover the remaining interactions in the second stage.

5.2.3 Coloring in the Second Stage

Now we describe strategies using graph coloring in the second stage. Construct a graph $G = (V, E)$, the *incompatibility graph*, in which V is the set of uncovered interactions and there is an edge between two interactions exactly when they share a column in which they have different symbols. A single row can cover a set of interactions if and only if it forms an independent set in G. Hence the minimum number of rows required to cover all interactions of G is exactly its chromatic number $\chi(G)$, the minimum number of colors in a proper coloring of G. Graph coloring is an NP-hard problem, so we employ heuristics to bound the chromatic number. Moreover, G only has vertices for the *uncovered* interactions after the first stage, so it is size is small relative to the total number of interactions.

The expected number of edges in the incompatibility graph after choosing n rows uniformly at random is $\gamma = \left(\frac{1}{2}\right)\binom{k}{t}v^t \sum_{i=1}^{t}\binom{t}{i}\binom{k-t}{t-i}(v^t - v^{t-i})\left(1-\frac{1}{v^t}\right)^n\left(1-\frac{1}{(v^t-v^{t-i})}\right)^n$.

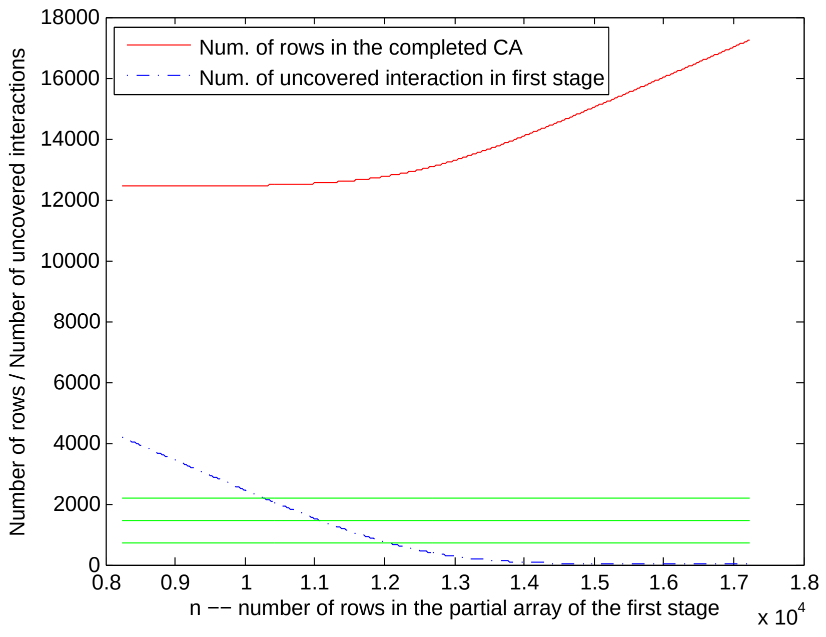


Figure 5.6: Final covering array size against the number of uncovered interactions after the first stage. As the size n of the partial array decreases, the number of uncovered interactions in the first stage increases. Den is used in the second stage. From bottom to top, the green lines denote ρ, 2ρ, and 3ρ uncovered interactions.

Using the elementary upper bound on the chromatic number $\chi \leq \frac{1}{2}+\sqrt{2m+\frac{1}{4}}$, where m is the number of edges [22, Chapter 5.2], we can surely cover the remaining interactions with at most $\frac{1}{2}+\sqrt{2m+\frac{1}{4}}$ rows.

The actual number of edges m that remain after the first stage is a random variable with mean γ. In principle, the first stage could be repeatedly applied until $m \leq \gamma$, so we call $m = \gamma$ the *optimistic estimate*. To ensure that the first stage is expected to be run a small constant number of times, we increase the estimate. With probability more than $\frac{1}{2}$ the incompatibility graph has $m \leq 2\gamma$ edges, so $m = 2\gamma$ is the *conservative estimate*.

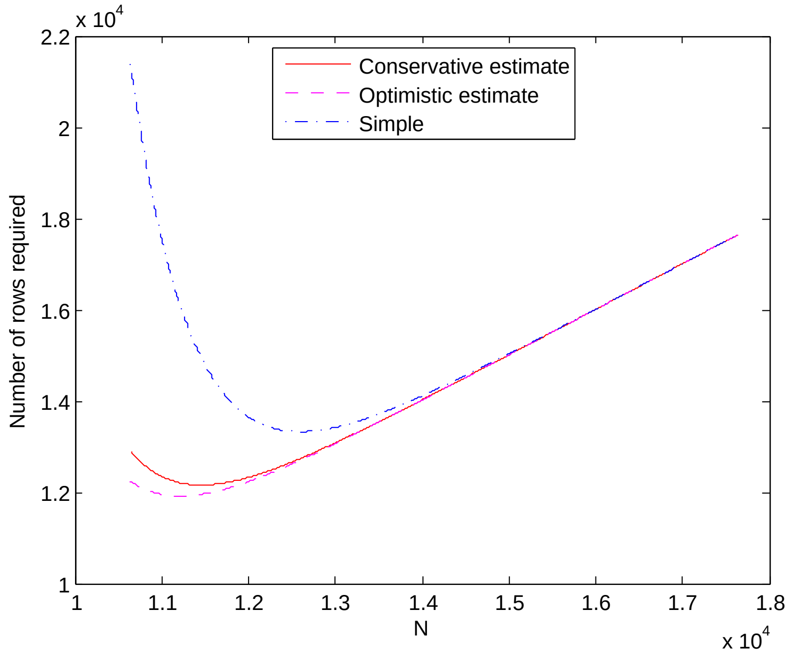


Figure 5.7: Size of the partial array vs. size of the complete covering array. $t = 6$, $k = 56$, $v = 3$. The Stein-Lovász-Johnson bound requires $17,403$ rows, and the discrete Stein-Lovász-Johnson bound requires $13,021$ rows. The simple estimate for the two stage algorithm is $13,328$ rows, the conservative estimate assuming $m = 2\gamma$ is $12,159$ rows, and the optimistic estimate assuming $m = \gamma$ is $11,919$ rows. Even the conservative estimate beats the discrete Stein-Lovász-Johnson bound.

For $t = 6$, $k = 56$, and $v = 3$, Figure 5.7 shows the effect on the minimum number of rows when the bound on the chromatic number in the second stage is used, for the conservative or optimistic estimates. The naïve method is plotted for comparison. Better coloring bounds shift the minima leftward, reducing the number of rows produced in both stages.

Thus far we have considered bounds on the chromatic number. Better estimation of $\chi(G)$ is complicated by the fact that we do not have much information about the structure of G until the first stage is run. In practice, however, G is known after the first stage and hence an algorithmic method to bound its chromatic number can be applied. Because the number of vertices in G equals the number of uncovered interactions after the first stage, we encounter the same trade-off between time and storage, and final array size, as seen earlier for density. Hence we again parameterize by the expected number of uncovered interactions in the first stage.

We employ two different greedy algorithms to color the incompatibility graph. In method Col we first construct the incompatibility graph G after the first stage. Then we apply the commonly used *smallest last order* heuristic to order the vertices for greedy coloring: At each stage, find a vertex v_i of minimum degree in G_i, order the vertices of $G_i - v_i$, and then place v_i at the end. More precisely, we order the vertices of G as $v_1, v_2, \ldots, v_n$, such that v_i is a vertex of minimum degree in G_i, where $G_i = G - \{v_{i+1}, \ldots, v_n\}$. A graph is *$d$-degenerate* if all of its subgraphs have a vertex with degree at most d. When G is d-degenerate but not $(d-1)$-degenerate, the *coloring number* $\mathsf{col}(G)$ is $d + 1$. If we then greedily color the vertices with the first available color, at most $\mathsf{col}(G)$ colors are used.

In method Greedy we employ an on-line, greedy approach that colors the interactions as they are discovered in the first stage. In this way, the incompatibility graph is never constructed. We instead maintain a set of rows. Some entries in rows are fixed to a specific value; others are flexible to take on any value. Whenever a new interaction is found to be uncovered in the first stage, we check if any of the rows is compatible with this interaction. If such a row is found then entries in the row are fixed so that the row now covers the interaction. If no such row exists, a new row with exactly t fixed entries corresponding to the interaction is added to the set of rows. This method is much faster than method Col in practice.

5.2.4 Using Group Action in the Second Stage

Application of group action to the TS ⟨Rand, Naive⟩ type methods was discussed in Section 3.2.1. Group action can be applied to the other methods for the second stage as well. Colbourn [19] incorporates group action into the density algorithm, allowing us to apply method Den in the second stage.

Greedy extends easily to use group action, as we do not construct an explicit incompatibility graph. Whenever we fix entries in a row to cover an uncovered orbit, we commit to a specific orbit representative.

However, applying group action to the incompatibility graph coloring for Col is more complicated. We need to modify the definition of the incompatibility graph for two reasons. First the vertices no longer represent uncovered interactions, but rather uncovered orbits of interactions. Secondly, and perhaps more importantly, pairwise compatibility between every two orbits in a set no longer implies mutual compatibility among all orbits in the set.

One approach is to form a vertex for each uncovered orbit, placing an edge between two when they share a column. Rather than the usual coloring, however, one asks for a partition of the vertex set into classes so that every class induces an acyclic subgraph. Problems of this type are *generalized graph coloring* problems [4]. Within each class of such a vertex partition, consistent representatives of each orbit can be selected to form a row; when a cycle is present, this may not be possible. Unfortunately, heuristics for solving these types of problems appear to be weak, so we adopt another approach. As we build the incompatibility graph, we commit to specific orbit representatives. When a vertex for an uncovered orbit is added, we check its compatibility with the orbit representatives chosen for the orbits already handled with which it shares columns; we commit to an orbit representative and add edges to those with which it is now incompatible. Once completed, we have a (standard) coloring problem for the resulting graph.

Because group action can be applied using each of the methods for the two stages, we extend our naming to $\mathsf{TS}\left\langle A, B; r, \Gamma\right\rangle$, where Γ can be Trivial (i.e. no group action), Cyclic, or Frobenius.

5.3 Computational Results

Figure 5.4 indicates that even a simple two-stage bound can improve on the best known covering array numbers. Therefore we investigate the actual performance of our two-stage algorithms for covering arrays of strength 5 and 6.

First we present results for $t = 6$, when $v \in \{3, 4, 5, 6\}$ and no group action is assumed. Table 5.1 shows the results for different v values. In each case we select the range of k values where the two-stage bound predicts smaller covering arrays than the previously known best ones, setting the maximum number of uncovered interactions as $\rho = 1/\log\left(\frac{v^t}{v^t-1}\right) \approx v^t$. For each value of k we construct a single partial array and then run the different second stage algorithms on it consecutively. In this way, all the second stage algorithms cover the same set of uncovered interactions.

The column **tab** lists the best known $\mathsf{CAN}(t, k, v)$ upper bounds from [17]. The column **bound** shows the upper bounds obtained from the two-stage bound (8). The columns **naïve**, **greedy**, **col** and **den** show results obtained from running the $\mathsf{TS}\left\langle\mathsf{Rand}, \mathsf{Naive}; \rho, \mathsf{Trivial}\right\rangle$, $\mathsf{TS}\left\langle\mathsf{Rand}, \mathsf{Greedy}; \rho, \mathsf{Trivial}\right\rangle$, $\mathsf{TS}\left\langle\mathsf{Rand}, \mathsf{Col}; \rho, \mathsf{Trivial}\right\rangle$ and $\mathsf{TS}\left\langle\mathsf{Rand}, \mathsf{Den}; \rho, \mathsf{Trivial}\right\rangle$ algorithms, respectively.

The naïve method always finds a covering array that is smaller than the two-stage bound. This happens because we repeat the first stage of Algorithm 10 until the array has fewer than v^t uncovered interactions. (If the first stage were not repeated, the algorithm would still produce covering arrays that are not too far from the bound.) For

$v = 3$ Greedy and Den have comparable performance. Method Col produces covering arrays that are smaller. However, for $v \in \{4, 5, 6\}$ Den and Col are competitive.

Table 5.1: Comparison of different TS $\langle$Rand, $-$; ρ, Trivial$\rangle$ algorithms.

k	tab	bound	naïve	greedy	col	den
			$t = 6, v = 3$			
53	13021	13076	13056	12421	12415	12423
54	14155	13162	13160	12510	12503	12512
55	17161	13246	13192	12590	12581	12591
56	19033	13329	13304	12671	12665	12674
57	20185	13410	13395	12752	12748	12757
			$t = 6, v = 4$			
39	68314	65520	65452	61913	61862	61886
40	71386	66186	66125	62573	62826	62835
41	86554	66834	66740	63209	63160	63186
42	94042	67465	67408	63819	64077	64082
43	99994	68081	68064	64438	64935	64907
44	104794	68681	68556	65021	65739	65703
			$t = 6, v = 5$			
31	233945	226700	226503	213244	212942	212940
32	258845	229950	229829	216444	217479	217326
33	281345	233080	232929	219514	219215	219241
34	293845	236120	235933	222516	222242	222244
35	306345	239050	238981	225410	226379	226270
36	356045	241900	241831	228205	230202	229942
			$t = 6, v = 6$			
17	506713	486310	486302	449950	448922	447864

Continued on the next page

Table 5.1 – *continued from the previous page*

k	tab	bound	naïve	greedy	col	den
18	583823	505230	505197	468449	467206	466438
19	653756	522940	522596	485694	484434	483820
20	694048	539580	539532	502023	500788	500194
21	783784	555280	555254	517346	516083	515584
22	844834	570130	569934	531910	530728	530242
23	985702	584240	584194	545763	544547	548307
24	1035310	597660	597152	558898	557917	557316
25	1112436	610460	610389	571389	570316	569911
26	1146173	622700	622589	583473	582333	582028
27	1184697	634430	634139	594933	593857	593546

Table 5.2 shows the results obtained by the different second stage algorithms when the maximum number of uncovered interactions in the first stage is set to 2ρ and 3ρ respectively. When more interactions are covered in the second stage, we obtain smaller arrays as expected. However, the improvement in size does not approach 50%. There is no clear winner.

Table 5.2: Comparison of $\mathsf{TS}\left<\mathsf{Rand}, -; 2\rho, \mathsf{Trivial}\right>$ and $\mathsf{TS}\left<\mathsf{Rand}, -; 3\rho, \mathsf{Trivial}\right>$ algorithms.

k	2ρ			3ρ		
	greedy	col	den	greedy	col	den
			$t = 6, v = 3$			
53	11968	11958	11968	11716	11705	11708

Continued on the next page

Table 5.2 – *continued from the previous page*

k	2ρ			3ρ		
	greedy	col	den	greedy	col	den
54	12135	12126	12050	11804	11787	11790
55	12286	12129	12131	11877	11875	11872
56	12429	12204	12218	11961	12055	11950
57	12562	12290	12296	12044	12211	12034
			$t = 6, v = 4$			
39	59433	59323	59326	58095	57951	57888
40	60090	60479	59976	58742	58583	58544
41	60715	61527	60615	59369	59867	59187
42	61330	62488	61242	59974	61000	59796
43	61936	61839	61836	60575	60407	60393
44	62530	62899	62428	61158	61004	60978
			$t = 6, v = 5$			
31	204105	203500	203302	199230	198361	197889
32	207243	206659	206440	202342	201490	201068
33	210308	209716	209554	205386	204548	204107
34	213267	212675	212508	208285	-	207060
35	216082	215521	215389	211118	-	209936
36	218884	218314	218172	213872	-	212707
			$t = 6, v = 6$			
17	425053	-	420333	412275	-	405093
18	443236	-	438754	430402	-	423493
19	460315	-	455941	447198	-	440532
20	476456	-	472198	463071	-	456725
21	491570	-	487501	478269	-	471946

Continued on the next page

Table 5.2 – *continued from the previous page*

k	2ρ			3ρ		
	greedy	col	den	greedy	col	den
22	505966	-	502009	492425	-	486306
23	519611	-	515774	505980	-	500038
24	532612	-	528868	518746	-	513047
25	544967	-	541353	531042	-	525536
26	556821	-	553377	542788	-	537418
27	568135	-	564827	554052	-	548781

Next, we investigate the covering arrays that are invariant under the action of a cyclic group. In Table 5.3 the column **bound** shows the upper bounds from Equation (3.1). The columns **naïve**, **greedy**, **col** and **den** show results obtained from running $\mathsf{TS}\left\langle\mathsf{Rand},\mathsf{Naive};\rho,\mathsf{Cyclic}\right\rangle$, $\mathsf{TS}\left\langle\mathsf{Rand},\mathsf{Greedy};\rho,\mathsf{Cyclic}\right\rangle$, $\mathsf{TS}\left\langle\mathsf{Rand},\mathsf{Col};\rho,\mathsf{Cyclic}\right\rangle$ and $\mathsf{TS}\left\langle\mathsf{Rand},\mathsf{Den};\rho,\mathsf{Cyclic}\right\rangle$, respectively.

Table 5.3: Comparison of different $\mathsf{TS}\left\langle\mathsf{Rand},-;\rho,\mathsf{Cyclic}\right\rangle$ algorithms.

k	tab	bound	naïve	greedy	col	den
			$t=6, v=3$			
53	13021	13059	13053	12405	12405	12411
54	14155	13145	13119	12489	12543	12546
55	17161	13229	13209	12573	12663	12663
56	19033	13312	13284	12660	12651	12663
57	20185	13393	13368	12744	12744	12750
			$t=6, v=4$			

Continued on the next page

Table 5.3 – *continued from the previous page*

k	tab	bound	naïve	greedy	col	den
39	68314	65498	65452	61896	61860	61864
40	71386	66163	66080	62516	62820	62784
41	86554	66811	66740	63184	63144	63152
42	94042	67442	67408	63800	63780	63784
43	99994	68057	68032	64408	64692	64680
44	104794	68658	68556	64988	64964	64976
			$t = 6, v = 5$			
31	226000	226680	226000	213165	212945	212890
32	244715	229920	229695	216440	217585	217270
33	263145	233050	233015	219450	221770	221290
34	235835	236090	235835	222450	222300	222210
35	238705	239020	238705	225330	225130	225120
36	256935	241870	241470	228140	229235	229020
			$t = 6, v = 6$			
17	506713	486290	485616	449778	448530	447732
18	583823	505210	504546	468156	467232	466326
19	653756	522910	522258	485586	490488	488454
20	694048	539550	539280	501972	500880	500172
21	783784	555250	554082	517236	521730	519966
22	844834	570110	569706	531852	530832	530178
23	985702	584210	583716	545562	549660	548196
24	1035310	597630	597378	558888	557790	557280
25	1112436	610430	610026	571380	575010	573882
26	1146173	622670	622290	583320	582546	582030
27	1184697	624400	633294	594786	598620	597246

Table 5.4 presents results for cyclic group action based algorithms when the number of maximum uncovered interactions in the first stage is set to 2ρ and 3ρ respectively.

Table 5.4: Comparison of $\mathsf{TS}\left\langle \mathsf{Rand}, -; 2\rho, \mathsf{Cyclic}\right\rangle$ and $\mathsf{TS}\left\langle \mathsf{Rand}, -; 3\rho, \mathsf{Cyclic}\right\rangle$ algorithms.

k		2ρ			3ρ	
	greedy	col	den	greedy	col	den
			$t = 6, v = 3$			
53	11958	11955	11958	11700	11691	11694
54	12039	12027	12036	11790	11874	11868
55	12120	12183	12195	11862	12057	12027
56	12204	12342	12324	11949	11937	11943
57	12276	12474	12450	12027	12021	12024
			$t = 6, v = 4$			
39	59412	59336	59304	58076	57976	57864
40	60040	59996	59964	58716	58616	58520
41	60700	61156	61032	59356	59252	59160
42	61320	62196	61976	59932	59840	59760
43	61908	63192	62852	60568	61124	60904
44	62512	64096	63672	61152	61048	60988
			$t = 6, v = 5$			
31	204060	203650	203265	199180	198455	197870
32	207165	209110	208225	202255	204495	203250
33	207165	209865	209540	205380	204720	204080
34	213225	212830	212510	208225	207790	207025
35	216050	217795	217070	211080	213425	212040

Continued on the next page

Table 5.4 – *continued from the previous page*

k	2ρ			3ρ		
	greedy	col	den	greedy	col	den
36	218835	218480	218155	213770	213185	212695
			$t = 6, v = 6$			
17	424842	422736	420252	411954	409158	405018
18	443118	440922	438762	430506	427638	423468
19	460014	457944	455994	447186	456468	449148
20	476328	474252	472158	463062	460164	456630
21	491514	489270	487500	478038	486180	479970
22	505884	503580	501852	492372	489336	486264
23	519498	517458	515718	505824	502806	500040
24	532368	530340	528828	518700	515754	512940
25	544842	542688	541332	530754	538056	532662
26	543684	543684	543684	542664	539922	537396
27	568050	566244	564756	553704	560820	555756

For the Frobenius group action, we show results only for $v \in \{3, 5\}$ in Table 5.5. The column **bound** shows the upper bounds obtained from Equation (3.2).

Table 5.6 presents results for Frobenius group action algorithms when the number of maximum uncovered interactions in the first stage is 2ρ or 3ρ.

Next we present a handful of results when $t = 5$. In the cases examined, using the trivial group action is too time consuming to be practical. However, the cyclic or Frobenius cases are feasible. Tables 5.7 and 5.8 compare two stage algorithms when the number of uncovered interactions in the first stage is at most 2ρ.

All code used in this experimentation is available from the GitHub repository

k	tab	bound	naïve	greedy	col	den
			$t = 6, v = 3$			
53	13021	13034	13029	12393	12387	12393
54	14155	13120	13071	12465	12513	12531
55	17161	13203	13179	12561	12549	12567
56	19033	13286	13245	12633	12627	12639
57	20185	13366	13365	12723	12717	12735
			$t = 6, v = 5$			
31	233945	226570	226425	213025	212865	212865
32	258845	229820	229585	216225	216085	216065
33	281345	232950	232725	219285	219205	219145
34	293845	235980	234905	222265	223445	223265
35	306345	238920	238185	225205	227445	227065
36	356045	241760	241525	227925	231145	230645

Table 5.5: Comparison of different $\mathsf{TS}\,\langle \mathsf{Rand}, -; \rho, \mathsf{Frobenius}\rangle$ algorithms.

https://github.com/ksarkar/CoveringArray

under an open source GPLv3 license.

5.3.1 Choosing a Method

In section 5.3 we presented computational results for different combinations of the parameter values t, k, and v and compared the four different second stage methods under different group actions and different values of r. In almost all cases there is no clear winner among the three second stage methods—Greedy, Den, and Col. However, in practical applications, while applying the two-stage framework for covering array construction one needs to decide how to properly parameterize the framework, i.e., which specific two-stage algorithm to use. In this section we provide general guidelines to users to help them choose the right method for their applications.

Assuming that the basic randomized method is used in the first stage, to obtain a specific two-stage algorithm one needs to make the following three choices: (1) What group action to use? (2) how many interactions to leave for the second stage? and (3) what method to use in the second stage? These choices are dictated chiefly by the values of t, k and v for which a covering array is sought. Additionally, available computing resources (computing time, RAM, parallelism etc.) figure in the consideration.

Of the three choices the most straightforward is deciding the group action. If v is a prime power then one should use Frobenius group action, otherwise Cyclic should be chosen. They produce shorter covering arrays than the Trivial group action, and they also run faster.

The choice of r—the number of interaction to leave for the second stage—is interrelated with the choice of the second stage method, and is determined by the values of t, k and v. As mentioned earlier, in general, the higher the value of r, the smaller the final covering array is in size. At the same time, for higher values of r, all other things being equal, the second stage method takes more time (and space) to complete the covering array. Therefore, for smaller values of t, k and v ($t \leq 4$, $k \leq 100$, and $v \leq 10$) it is advisable to set $r = 3\rho$ (where $\rho \approx v^t$). For higher values of the inputs ($t \geq 5$, $k \geq 100$, and $v \geq 5$), the computation time in the second stage is too high for most practical purposes unless r is set to ρ. In the ranges in between one should start by setting $r = 2\rho$ and then should choose a smaller value if the computation time is prohibitively large for the purpose.

The space complexity of the Col method is $O(r^2)$ because it needs to store the dense incompatibility graph for the remaining uncovered interactions. For the Den method, the space complexity is $O(r)$, and for the Greedy method it is constant and almost negligible for all practical purposes. Also in terms of the running time Greedy is the fastest, Col takes the maximum time, and Den is in between. However, in terms of the size of constructed covering arrays Greedy and Den are comparable, while Col often produces arrays which are slightly smaller. So, if the values of v, and t are relatively high and r is set to a higher value then Greedy is the best choice in most scenarios. On the other hand, if v and t are relatively smaller and r is set to ρ then one may consider employing Col in the second stage.

k	2ρ			3ρ		
	greedy	col	den	greedy	col	den
			$t = 6, v = 3$			
53	11931	11919	11931	11700	11691	11694
54	12021	12087	12087	11790	11874	11868
55	12105	12237	12231	11862	12057	12027
56	12171	12171	12183	11949	11937	11943
57	12255	12249	12255	12027	12021	12024
70	13167	13155	13179	-	-	-
75	13473	13473	13479	-	-	-
80	13773	13767	13779	-	-	-
85	14031	14025	14037	-	-	-
90	14289	14283	14301	-	-	-
			$t = 6, v = 5$			
31	203785	203485	203225	198945	198445	197825
32	206965	208965	208065	201845	204505	203105
33	209985	209645	209405	205045	209845	207865
34	213005	214825	214145	208065	207545	206985
35	215765	215545	215265	210705	210365	209885
36	218605	218285	218025	213525	213105	212645
50	250625	250365	250325	-	-	-
55	259785	259625	259565	-	-	-
60	268185	268025	267945	-	-	-
65	275785	275665	275665	-	-	-

Table 5.6: Comparison of $\mathsf{TS}\ \langle \mathsf{Rand}, -; 2\rho, \mathsf{Frobenius} \rangle$ and $\mathsf{TS}\ \langle \mathsf{Rand}, -; 3\rho, \mathsf{Frobenius} \rangle$ algorithms.

k	tab	greedy	col	den
67	59110	48325	48285	48305
68	60991	48565	48565	48585
69	60991	48765	49005	48985
70	60991	49005	48985	49025
71	60991	49245	49205	49245

Table 5.7: Comparison of $\mathsf{TS}\,\langle \mathsf{Rand}, -; 2\rho, \mathsf{Frobenius}\rangle$ algorithms. $t = 5, v = 5$

k	tab	greedy	col	den
49	122718	108210	108072	107988
50	125520	109014	108894	108822
51	128637	109734	110394	110166
52	135745	110556	110436	110364
53	137713	111306	111180	111120

Table 5.8: Comparison of $\mathsf{TS}\,\langle \mathsf{Rand}, -; 2\rho, \mathsf{Cyclic}\rangle$ algorithms. $t = 5, v = 6$

Chapter 6

GROUP ACTION AND LIMITED DEPENDENCE

When $k \geq 2t$, some interactions have no columns in common. The events of coverage of such interactions are independent. Neither Theorem 1 nor Theorem 8 takes advantage of this fact. Consider an $N \times k$ array A with each entry chosen independently and uniformly at random from Σ. Let A_ι denote the event that the interaction $\iota \in \mathcal{I}_{t,k,v}$ is not covered in A. A_ι depends on all events $\{A_\rho : \rho \in \mathcal{I}_{t,k,v}, c(\iota) \cap c(\rho) \neq \emptyset\}$, and only on those events. Hence when $k \geq 2t$, there are events A_ρ of which A_ι is independent. Godbole et al. [27] exploits this limited dependence by applying the Lovász local lemma to prove Theorem 2 which is asymptotically tighter than Theorem 1. In this chapter we employ an additional technique — group action, together with the Lovász local lemma to obtain a number of powerful upper bounds on $\mathsf{CAN}(t,k,v)$. The results in this chapter are based on the work reported in Sarkar and Colbourn [48].

6.1 Group Action

We begin by considering sharply transitive group actions on the rows of a covering array together with limited dependence. We consider only the covering arrays that are invariant under such group action, thereby concentrating on covering orbits of interaction instead of each interaction individually. First, we prove a result that was first reported by Francetić and Stevens [23]. However, our proof is considerably simpler than their proof. Recall that $d(t,v)$ is defined as $d(t,v) = \limsup_{k \to \infty} \dfrac{\mathsf{CAN}(t,k,v)}{\log k}$.

Theorem 10. *Let t, v be integers with $t, v \geq 2$. Then*

$$d(t,v) \leq \frac{v(t-1)}{\log\left(\frac{v^{t-1}}{v^{t-1}-1}\right)}.$$

Proof. Let Γ be a group that acts sharply transitively on Σ. Let $\mathscr{C}_t = \binom{[k]}{t}$, and $\tau \in \mathscr{C}_t$ be a collection of t columns. The action of Γ partitions the set of interactions involving the columns in τ into v^{t-1} orbits of length v each. We consider an $n \times k$ array A with each entry chosen independently and uniformly at random from the alphabet Σ . We want to cover all the orbits for every $\tau \in \mathscr{C}_t$. The probability that there is at least one orbit involving τ that is not covered is at most $v^{t-1}\left(1 - \frac{1}{v^{t-1}}\right)^n$.

For $\tau \in \mathscr{C}_t$, let A_τ denote the event that not all the orbits involving the columns in τ are covered in A. So $\Pr[A_\tau] \leq v^{t-1}\left(1 - \frac{1}{v^{t-1}}\right)^n$ for all $\tau \in \mathcal{T}$. The event A_τ is not independent of event A_ρ if and only if τ and ρ share a column. So $d \leq \binom{t}{1}\binom{k-1}{t-1} < t\binom{k}{t-1} < \frac{tk^{t-1}}{(t-1)!}$. By the Lovász local lemma (Lemma 5), if $ev^{t-1}\left(1 - \frac{1}{v^{t-1}}\right)^n \frac{tk^{t-1}}{(t-1)!} < 1$, there exists an $n \times k$ array that covers every orbit on every t-column combination of A. Solving for n, and then developing A over the group Γ, we obtain a covering array of size N, where

$$\begin{aligned}
N &= vn \\
&> v\frac{1 + \log\left(v^{t-1}t\frac{k^{t-1}}{(t-1)!}\right)}{\log\left(\frac{v^{t-1}}{v^{t-1}-1}\right)} \\
&= \frac{v(t-1)\log k}{\log\left(\frac{v^{t-1}}{v^{t-1}-1}\right)}\left\{1 + \frac{1}{(t-1)\log k} + \frac{\log v}{\log k} + \frac{\log t}{(t-1)\log k} - \frac{\log((t-1)!)}{\log k}\right\} \\
&= \frac{v(t-1)\log k}{\log\left(\frac{v^{t-1}}{v^{t-1}-1}\right)}\{1 + \mathrm{o}(1)\}
\end{aligned}$$

This yields the required bound on $d(t,v)$. □

If we compare the bounds from Theorems 2 and 10 using the Taylor series expansion of $\log(1-x) = -x - \frac{x^2}{2} - O(x^3)$, we find that

$$\begin{array}{rcl} \frac{t-1}{\log\left(\frac{v^t}{v^t-1}\right)} = \frac{t-1}{-\log\left(1-\frac{1}{v^t}\right)} & \approx & \frac{t-1}{\left(\frac{1}{v^t}+\frac{1}{2.v^{2t}}\right)} = \frac{v^t(t-1)}{1+\frac{1}{2v^t}}, \text{ and} \\ \frac{v(t-1)}{\log\left(\frac{v^{t-1}}{v^{t-1}-1}\right)} = \frac{v(t-1)}{-\log\left(1-\frac{1}{v^{t-1}}\right)} & \approx & \frac{v(t-1)}{\left(\frac{1}{v^{t-1}}+\frac{1}{2v^{2t-2}}\right)} = \frac{v^t(t-1)}{1+\frac{1}{2v^{t-1}}}. \end{array}$$

Hence the bound of Theorem 10 is tighter than that of Theorem 2.

Next, we present an even tighter bound that utilizes a larger permutation group:

Theorem 11. *Let $t \geq 2$ be an integer and v be a prime power. Then*

$$d(t,v) \leq \frac{v(v-1)(t-1)}{\log\left(\frac{v^{t-1}}{v^{t-1}-v+1}\right)}. \tag{6.1}$$

Proof. Let Γ be a group that is sharply 2-transitive on v symbols. Consider the action of Γ on the set of interactions involving the columns $\tau \in \binom{[k]}{t}$. Under the action of Γ the v interactions $\{(c_i, v_i) : c_i \in \tau, 1 \leq i \leq t\}$ with $v_1 = \ldots = v_t$ (the *constant* interactions) form a single orbit of length v. The remaining $v^t - v$ interactions form $\frac{v^{t-1}-1}{v-1}$ orbits, each of length $v(v-1)$. So the probability that a full length orbit is not covered in a $n \times k$ random array is $\left(1-\frac{v-1}{v^{t-1}}\right)^n$, and the probability that at least one of these orbits is not covered in the random array is at most $\left(\frac{v^{t-1}-1}{v-1}\right)\left(1-\frac{v-1}{v^{t-1}}\right)^n$ by the union bound.

Using the Lovász local lemma (Lemma 5), when $e\left(\frac{v^{t-1}-1}{v-1}\right)\left(1-\frac{v-1}{v^{t-1}}\right)^n t\frac{k^{t-1}}{(t-1)!} < 1$, there exists an $n \times k$ array that covers all the full orbits of interactions on all t-column combinations. Developing this array over Γ and adding v additional rows to cover the short orbit, we obtain a covering array that has N rows, with

$$\begin{array}{rcl} N & = & v(v-1)n + v \\ & > & v(v-1)\dfrac{1+\log\left(t\frac{k^{t-1}}{(t-1)!}\right)+\log\left(\frac{v^{t-1}-1}{v-1}\right)}{\log\left(\frac{v^{t-1}}{v^{t-1}-v+1}\right)} + v \\ & = & \dfrac{v(v-1)(t-1)\log k}{\log\left(\frac{v^{t-1}}{v^{t-1}-v+1}\right)}\{1+\mathrm{o}(1)\} \end{array}$$

This proves the theorem. $\square$

Again, using the Taylor series expansion we find that the bound obtained in Theorem 11 is tighter than that of Theorem 10, since

$$\frac{v(v-1)(t-1)}{\log\left(\frac{v^{t-1}}{v^{t-1}-v+1}\right)} = \frac{v(v-1)(t-1)}{-\log\left(1-\frac{v-1}{v^{t-1}}\right)} \approx \frac{v(v-1)(t-1)}{\left\{\frac{v-1}{v^{t-1}}+\frac{(v-1)^2}{2v^{2t-2}}\right\}} = \frac{v^t(t-1)}{1+\frac{v-1}{2v^{t-1}}}.$$

As mentioned in Chapter 2, the Frobenius group defined on the finite field $\mathbb{F}_v$ as $G = \{g : \mathbb{F}_v \to \mathbb{F}_v \,:\, g(x) = ax+b,\, x,a,b \in \mathbb{F}_v,\, a \neq 0\}$ is an example of an efficiently constructible group that acts sharply 2-transitively on the set of v symbols and can be used for practical construction of covering arrays [19].

It is natural to consider the action of larger groups in seeking further improvements. One simple but important idea in Theorem 11 is to treat full length orbits using the Lovász local lemma, affixing a small number of additional rows to cover the short orbits. Thus far we have treated a sharply 1-transitive group (the cyclic group) and a sharply 2-transitive group (the Frobenius group). In order to generalize, the next natural choice is the projective general linear (PGL) group for $v = q+1$ where q is a prime power, which is a sharply 3-transitive group of order $v(v-1)(v-2)$.

Let Γ be the PGL group on v symbols. The action of Γ on t-way interactions forms orbits of lengths v, $v(v-1)$, and $v(v-1)(v-2)$. Constant interactions lie in orbits of length v, interactions involving precisely two distinct symbols lie in orbits of length $v(v-1)$, and the $r = \frac{v^{t-1}-(v-1)(2^{t-1}-1)-1}{(v-1)(v-2)}$ others lie in full length orbits. Constant orbits can be handled as in Theorem 11, and full length orbits can be treated using the Lovász local lemma.

Unlike constant orbits, orbits of length $v(v-1)$ cannot be covered with a number of rows that is independent of k. If we cover the orbits of length $v(v-1)$ as we covered full length orbits, we see no improvement over Theorem 11. Instead we adapt a method from Cohen *et al.* [15] to gain occasional improvements.

Theorem 12. *Let $t \geq 2$ be an integer and $v-1$ be a prime power. Then*

$$d(t,v) \leq \frac{v(v-1)(v-2)(t-1)}{\log\left\{\frac{v^{t-1}}{v^{t-1}-(v-1)(v-2)}\right\}} + \frac{v(v-1)(t-1)}{\log\left(\frac{2^{t-1}}{2^{t-1}-1}\right)}.$$

Proof. Let Γ be the PGL group acting on v symbols.

Covering orbits of length $v(v-1)(v-2)$: The probability that at least one orbit of length $v(v-1)(v-2)$ is not covered in an array with n rows is $p \leq r\left(1 - \frac{(v-1)(v-2)}{v^{t-1}}\right)^n$. As shown before, $d < t\binom{k}{t-1} \leq t\frac{k^{t-1}}{(t-1)!}$. Using the Lovász local lemma (Lemma 5), if $ep(d+1) \leq 1$ there is an array with n rows that covers all orbits of length $v(v-1)(v-2)$. Developing over Γ we obtain an array of the size

$$\begin{aligned}
&v(v-1)(v-2)\frac{1+\log\left\{t\frac{k^{t-1}}{(t-1)!}\right\}+\log r}{\log\left\{\frac{v^{t-1}}{v^{t-1}-(v-1)(v-2)}\right\}}\\
=&v(v-1)(v-2)\frac{1+(t-1)\log k+\log t+\log r-\log((t-1)!)}{\log\left\{\frac{v^{t-1}}{v^{t-1}-(v-1)(v-2)}\right\}}\\
=&\frac{v(v-1)(v-2)(t-1)}{\log\left\{\frac{v^{t-1}}{v^{t-1}-(v-1)(v-2)}\right\}}\log k\,\{1+o(1)\}
\end{aligned}$$

Using the Taylor series expansion:

$$\frac{v(v-1)(v-2)(t-1)}{\log\left\{\frac{v^{t-1}}{v^{t-1}-(v-1)(v-2)}\right\}} \approx \frac{v^t(t-1)}{1+\frac{(v-1)(v-2)}{2v^{t-1}}}.$$

Covering orbits of length $v(v-1)$: Use a binary covering array on every pair of symbols, adding $\binom{v}{2}\mathsf{CAN}(t,k,2)$ rows to cover all interactions in orbits of length $v(v-1)$. Applying Theorem 10 to bound $\mathsf{CAN}(t,k,2)$, in this way we add

$$\binom{v}{2}\frac{2(t-1)}{\log\left(\frac{2^{t-1}}{2^{t-1}-1}\right)}\log k\,\{1+o(1)\} = \frac{v(v-1)(t-1)}{\log\left(\frac{2^{t-1}}{2^{t-1}-1}\right)}\log k\,\{1+o(1)\}$$

rows.

So $d(t,v) \leq \frac{v(v-1)(v-2)(t-1)}{\log\left\{\frac{v^{t-1}}{v^{t-1}-(v-1)(v-2)}\right\}} + \frac{v(v-1)(t-1)}{\log\left(\frac{2^{t-1}}{2^{t-1}-1}\right)}$. $\square$

In the bound of Theorem 12, the first term dominates the second. However, only when $t \in \{3,4\}$ and v is sufficiently large does Theorem 12 give a tighter bound on $d(t,v)$ than that given by Theorem 11. Moreover, Theorem 11 gives a tighter bound in many situations, e.g., for $t = 5$, it is tighter than Theorem 12 as long as $v \leq 29$. For larger t, Theorem 11 gives tighter bounds than Theorem 12 for even larger values of v. Hence the natural avenue of generalization to larger groups does not appear to be fruitful.

So far in our discussion of group action we have emphasized only the aspect of search space reduction. Now we mention a side benefit inherent to sharply transitive group actions that further validates their role. By using sharply transitive (or sharply l-transitive) group actions we can further reduce the dependence among different bad events.

Consider the cyclic group used in Theorem 10. For any set of t columns τ, if we fix the symbols in a specific column c and select symbols in the remaining $t-1$ columns independently and uniformly at random, the probability of a "bad event" (i.e., at least one orbit not being covered) remains unchanged. This suggests that all the "bad events" on the t-set of columns that share only the column c with τ are mutually independent of the "bad event" on τ. Therefore, we can set $d \leq t\binom{k-1}{t-1} - \binom{k-t}{t-1}$. A similar reduction in dependence may be obtained when we apply the Lovász local lemma to cover the full length orbits under sharply l-transitive group actions in Theorem 11 and Theorem 12.

The storage requirements are quite modest; in order to determine whether resampling is necessary, one maintains a single list indexed by the orbits of Σ^t. A set of t

columns can be treated without regard to the coverage in other sets of t columns.

6.2 Limited Dependence and the Moser-Tardos Algorithms

The guarantee of existence of covering arrays of a certain size that are required to prove Theorem 2, as well as Theorem 10 and 3 are derived using the Lovász local lemma (Lemma 5) which is a non-constructive result. However, as we have discussed in Section 2.2.1 whenever we can guarantee the existence of a combinatorial object using the Lovász local lemma we can design a Moser-Tardos type algorithm that constructs such objects efficiently.

Specializing Algorithm 1 to covering arrays, we obtain Algorithm 11. With the specified value of N from Theorem 2, it is guaranteed that the expected number of times the loop in line 3 of Algorithm 11 is repeated is linearly bounded in k.

Godbole et al. [27] presents Theorem 2 as a non-constructive result. Indeed no previous construction algorithms appear to be based on it. However the Moser-Tardos method of Algorithm 11 does provide a construction algorithm that runs in expected polynomial time. For sufficiently large values of k Algorithm 11 produces smaller covering arrays than Algorithm 3.

But the question remains: Does Algorithm 11 produce smaller covering arrays than the best known results to date within the range that it can be effectively computed? Surprisingly, we show that the answer is affirmative. In Algorithm 11 we do not need to store the coverage information of individual interactions in memory because each time an uncovered interaction is encountered we re-sample the columns involved in that interaction and start the check afresh (checking the coverage in interactions in the same order each time). Consequently, Algorithm 11 can be applied for larger values of k than the density algorithm.

Smaller covering arrays can be obtained by exploiting a group action using Lovász

Algorithm 11: Moser-Tardos type algorithm for covering array construction.

```
Input: $t$ : strength of the covering array, $k$ : number of factors, $v$ : number of
       levels for each factor
Output: $A$ : a $\mathsf{CA}(N; t, k, v)$
1  Let $N := \frac{\log\left\{\binom{k}{t}-\binom{k-t}{t}\right\}+t.\log v+1}{\log\left(\frac{v^t}{v^t-1}\right)}$;
2  Construct an $N \times k$ array $A$ where each entry is chosen independently and
     uniformly at random from a $v$-ary alphabet;
3  repeat
4      Set covered:= true;
5      for each interaction $\iota \in \mathcal{I}_{t,k,v}$ do
6          if $\iota$ is not covered in $A$ then
7              Set covered:= false;
8              Set missing-interaction := $\iota$;
9              break;
10         end
11     end
12     if covered = false then
13         Choose all the entries in the $t$ columns involved in missing-interaction
             independently and uniformly at random from the $v$-ary alphabet;
14     end
15 until covered = true;
16 Output $A$;
```

local lemma, as shown in Theorem 10 and 11. Table 6.1 shows the sizes of the covering arrays constructed by a variant of Algorithm 11 that employs cyclic and Frobenius group actions. While this single stage algorithm produces smaller arrays than the currently best known ones [17], these are already superseded by the two-stage algorithms (Chapter 5).

k	Best	MT
56	19033	16281
57	20185	16353
58	23299	16425
59	23563	16491
60	23563	16557

(a) Frobenius. $t = 6$, $v = 3$

k	Best	MT
44	411373	358125
45	417581	360125
46	417581	362065
47	423523	363965
48	423523	365805

(b) Frobenius. $t = 6$, $v = 5$

k	Best	MT
25	1006326	1020630
26	1040063	1032030
27	1082766	1042902
28	1105985	1053306
29	1149037	1063272

(c) Cyclic. $t = 6$, $v = 6$

Table 6.1: Comparison of covering array sizes from Algorithm 11 (MT) with the best known results [17] (Best).

6.2.1 A Moser-Tardos Type Algorithm for the First Stage

All the two-stage algorithms in Chapter 5 employ the naïve randomized algorithm in the first stage. Can we apply a Moser-Tardos type algorithm in the first stage and still obtain a similar guarantee on the size of the constructed covering array? The linearity of expectation arguments used in the SLJ bounds permit one to consider situations in which a few of the "bad" events are allowed to occur, a fact that we have exploited in the first stage of the algorithms. However, the Lovász local lemma does not address this situation directly. The conditional Lovász local lemma (LLL) distribution, introduced in Haeupler *et al.* [30], is a very useful tool.

Lemma 13. *(Conditional LLL distribution; symmetric case) (see [2, 30]) Let* $\mathscr{A} = \{A_1, A_2, \ldots, A_l\}$ *be a set of* l *events in an arbitrary probability space. Suppose that each event* A_i *is mutually independent of a set of all other events* A_j *except for at most* d*, and that* $\Pr[A_i] \leq p$ *for all* $1 \leq i \leq l$*. Also suppose that* $ep(d+1) \leq 1$ *(Therefore, by LLL (Lemma 5)* $\Pr[\cap_{i=1}^{l}\bar{A}_i] > 0$*). Let* $B \notin \mathscr{A}$ *be another event in the same probability space with* $\Pr[B] \leq q$*, such that* B *is also mutually independent of a set of all other events* $A_j \in \mathscr{A}$ *except for at most* d*. Then* $\Pr[B| \cap_{i=1}^{l} \bar{A}_i] \leq eq$*.*

We apply the conditional LLL distribution to obtain an upper bound on the size of partial array that leaves at most $\log\left(\frac{v^t}{v^t-1}\right) \approx v^t$ interactions uncovered. For a positive integer k, let $I = \{j_1, \ldots, j_\rho\} \subseteq [k]$ where $j_1 < \ldots < j_\rho$. Let A be an $n \times k$ array where each entry is from the set $[v]$. Let A_I denote the $n \times \rho$ array in which $A_I(i, \ell) = A(i, j_\ell)$ for $1 \leq i \leq N$ and $1 \leq \ell \leq \rho$; A_I is the projection of A onto the columns in I.

Let $M \subseteq [v]^t$ be a set of m t-tuples of symbols, and $C \in \binom{[k]}{t}$ be a set of t columns. Suppose the entries in the array A are chosen independently from $[v]$ with uniform probability. Let B_C denote the event that at least one of the tuples in M is not covered in A_C. There are $\eta = \binom{k}{t}$ such events, and for all of them $\Pr[B_C] \leq m\left(1 - \frac{1}{v^t}\right)^n$. Moreover, when $k \geq 2t$, each of the events is mutually independent of all other events except for at most $\rho = \binom{k}{t} - \binom{k-t}{t} - 1 < t\binom{k}{t-1}$. Therefore, by the Lovász local lemma, when $e\rho m\left(1 - \frac{1}{v^t}\right)^n \leq 1$, none of the events B_C occur. Therefore, when

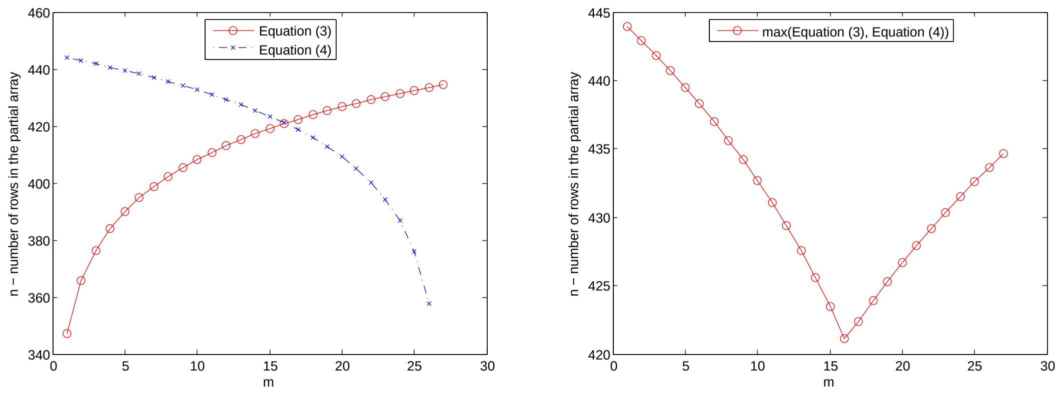
$$n \geq \frac{\log(e\rho m)}{\log\left(\frac{v^t}{v^t-1}\right)} \tag{6.2}$$

there exists an $n \times k$ array A over $[v]$ such that for all $C \in \binom{[k]}{t}$, A_C covers all the m tuples in M. In fact, we can use a Moser-Tardos type algorithm to construct such an array.

Let ι be an interaction whose t-tuple of symbols is not in M. Then the probability that ι is not covered in an $n \times k$ array is at most $\left(1 - \frac{1}{v^t}\right)^n$ when each entry of the array is chosen independently from $[v]$ with uniform probability. Therefore, by the conditional LLL distribution the probability that ι is not covered in the array A where for all $C \in \binom{[k]}{t}$, A_C covers all the m tuples in M is at most $e\left(1 - \frac{1}{v^t}\right)^n$. Moreover, there are $\eta(v^t - m)$ such interactions ι. By the linearity of expectation, the expected number of uncovered interactions in A is less than v^t when $\eta(v^t - m)e\left(1 - \frac{1}{v^t}\right)^n \leq v^t$. Solving for n, we obtain

$$n \geq \frac{\log\left\{\eta e\left(1 - \frac{m}{v^t}\right)\right\}}{\log\left(\frac{v^t}{v^t - 1}\right)}. \tag{6.3}$$

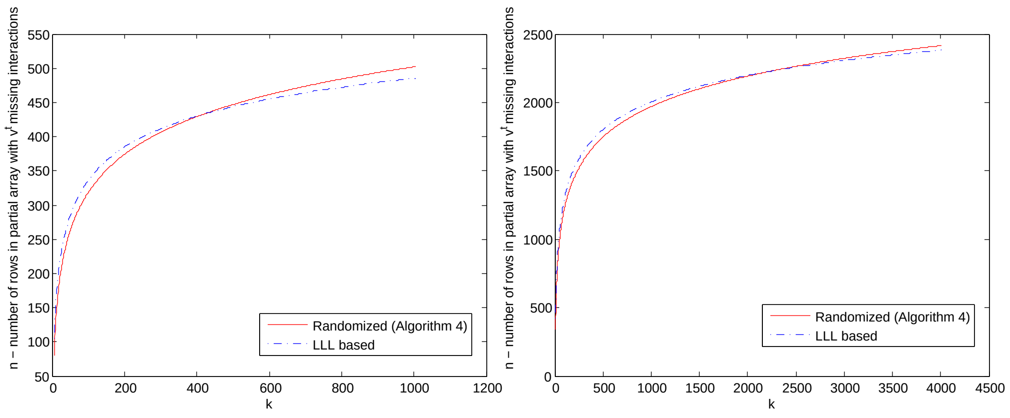
Therefore, there exists an $n \times k$ array with $n = \max\left\{\frac{\log(e\rho m)}{\log\left(\frac{v^t}{v^t-1}\right)}, \frac{\log\left\{\eta e\left(1 - \frac{m}{v^t}\right)\right\}}{\log\left(\frac{v^t}{v^t-1}\right)}\right\}$ that has at most v^t uncovered interactions. To compute n explicitly, we must choose m. We can select a value of m to minimize n graphically for given values of t, k and v. For example, Figure 6.1 plots Equations 6.2 and 6.3 against m for $t = 3$, $k = 350$, $v = 3$, and finds the minimum value of n.



(a) Equations 6.2 and 6.3 against m. (b) Maximum of the two sizes against m.

Figure 6.1: Determination of the size of the partial array. The minimum is at $n = 422$, when $m = 16$. $t = 3$, $k = 350$, $v = 3$.

We compare the size of the partial array from the naïve two-stage method (Algorithm 10) with the size obtained by the graphical methods in Figure 6.2. The Lovász local lemma based method is asymptotically better than the simple randomized method. However, except for the small values of t and v, in the range of k values relevant for practical applications, the simple randomized algorithm requires fewer rows than the Lovász local lemma based method.



(a) $t = 3$, $v = 3$. (b) $t = 4$, $v = 3$.

Figure 6.2: Comparison of the size of the partial array constructed in the first stage. The size of the partial array specified in Algorithm 10 in Section 5.2.1 is compared against the size derived in Section 6.2.1.

6.2.2 The Lovász Local Lemma Based Two-stage Bound

We can apply the techniques from Section 6.2.1 to obtain a two-stage bound similar to Theorem 8 using the Lovász local lemma and conditional LLL distribution. Theorem 14 extends a result from Sarkar and Colbourn [48].

Theorem 14. *Let t, k, v be integers with $k \geq t \geq 2$, $v \geq 2$ and let $\eta = \binom{k}{t}$, and $\rho = \binom{k}{t} - \binom{k-t}{t}$. If $\frac{\eta v^t \log\left(\frac{v^t}{v^t-1}\right)}{\rho} \leq v^t$ Then*

$$\mathsf{CAN}(t,k,v) \leq \frac{\log \binom{k}{t} + t\log v + \log\log\left(\frac{v^t}{v^t-1}\right) + 2}{\log\left(\frac{v^t}{v^t-1}\right)} - \frac{\eta}{\rho}.$$

Proof. Let $M \subseteq [v]^t$ be a set of m t-tuples of symbols. Following the arguments of Section 6.2.1, when $n \geq \frac{\log(e\rho m)}{\log\left(\frac{v^t}{v^t-1}\right)}$ there exists an $n \times k$ array A over $[v]$ such that for all $C \in \binom{[k]}{t}$, A_C covers all m tuples in M.

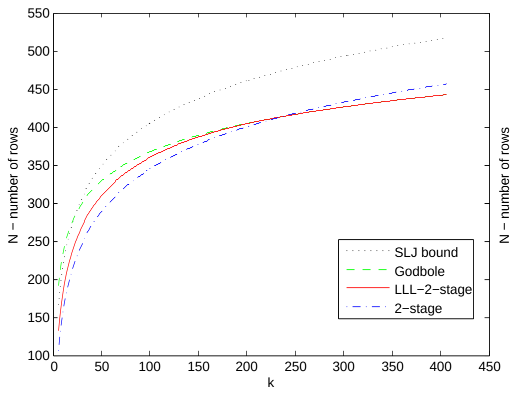
At most $\eta(v^t - m)$ interactions are uncovered in such an array. Using the conditional LLL distribution, the probability that one such interaction is not covered in A is at most $e\left(1 - \frac{1}{v^t}\right)^n$. Therefore, by the linearity of expectation, we can find one such array A that leaves at most $e\eta(v^t - m)\left(1 - \frac{1}{v^t}\right)^n = \frac{\eta}{\rho}\left(\frac{v^t}{m} - 1\right)$ interactions uncovered. Adding one row per uncovered interaction to A, we obtain a covering array with at most N rows, where

$$N = \frac{\log(e\rho m)}{\log\left(\frac{v^t}{v^t-1}\right)} + \frac{\eta}{\rho}\left(\frac{v^t}{m} - 1\right).$$

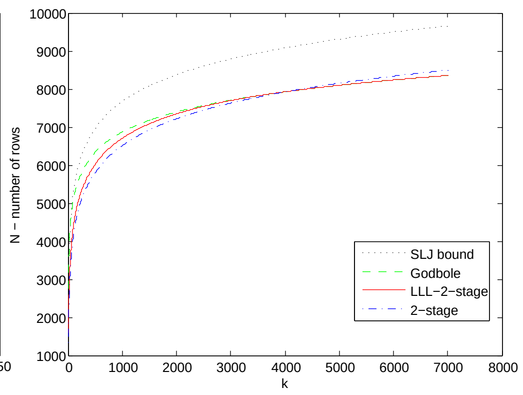
The value of N is minimized when $m = \frac{\eta v^t \log\left(\frac{v^t}{v^t-1}\right)}{\rho}$. Because $m \leq v^t$, we obtain the desired bound. $\square$

When $m = v^t$ this recaptures the bound of Theorem 2.

Figure 6.3 compares the LLL based two-stage bound obtained in Theorem 14 to the standard two-stage bound (Theorem 8), the Godbole et al. bound (Theorem 2), and the SLJ bound (Theorem 1). Although the LLL based two-stage bound is tighter than the LLL based Godbole et al. bound, even for quite large values of k the standard two-stage bound is tighter than the LLL based two-stage bound. In practical terms, this specific LLL based two-stage method does not look very promising, unless the parameters are quite large.



(a) $t = 3$, $v = 3$.



(b) $t = 4$, $v = 4$.

Figure 6.3: Comparison of constructed covering array sizes among the Lovász local lemma based two-stage bound (Theorem 14), the standard two-stage bound (Theorem 8), the Godbole et al. bound (Theorem 2), and the Stein-Lovász-Johnson Bound bound (Theorem 1).

Chapter 7

COVERING PERFECT HASH FAMILIES

In the last chapter, we saw that as we graduate from the cyclic group to a larger group like the Frobenius group we obtain smaller covering arrays. However, the improvement obtained using the PGL group is not substantial when compared to that of the smaller Frobenius group. Are there any other groups and related group actions that would result in tighter bounds? It appears that the general linear group plays an important role in the further improvement of the $\mathsf{CAN}(t,k,v)$ upper bounds in a very interesting way.

In this section, we introduce permutation vectors and covering perfect hash families. We use covering perfect hash families to construct covering arrays. The general linear group plays an important role in the construction of covering perfect hash families. The role of the general linear group can be understood as a group acting on t-sets of columns instead of individual symbols which has been the case thus far.

7.1 Permutation Vectors

We consider only the case when v is a prime power and the symbols are elements of the field $\mathbf{GF}(v)$. Let $\mathbf{b}^{(i)} = (b_0^{(i)}, b_1^{(i)}, \ldots, b_{t-1}^{(i)})$ be the base v representation of $i \in \{0, 1, \ldots, v^t - 1\}$, i.e., $i = b_0^{(i)} + v^1 b_1^{(i)} + \ldots + v^{t-1} b_{t-1}^{(i)}$ where $b_j^{(i)} \in \{0, 1, \ldots, v-1\}$ for $0 \leq j \leq t-1$. Let A be a $v^t \times t$ matrix where the ith row is $\mathbf{b}^{(i)}$. Let $\mathbf{h} = (h_0, h_1, \ldots, h_{t-1})^\intercal$ be a vector of length t with $h_j \in \{0, 1, \ldots, v-1\}$. A *permutation vector* $\overrightarrow{\mathbf{h}}$ corresponding to the vector $\mathbf{h}$ is a vector of length v^t and is defined as $\overrightarrow{\mathbf{h}} = A\mathbf{h}$ where the addition and the multiplications are over $\mathbf{GF}(v)$.

Next, let $\mathbf{h} = (h_1, h_2, \ldots, h_{t-1})^\intercal$ be a vector of length $t-1$ with $h_j \in \{0, 1, \ldots, v-$

1}, and $\mathbf{h}^+ = (1, h_1, \ldots, h_{t-1})^\intercal$. A *Sherwood permutation vector* $\overrightarrow{\mathbf{h}^+}$, corresponding to the vector $\mathbf{h}$, is a permutation vector corresponding to the vector $\mathbf{h}^+$, i.e., a vector of length v^t that is defined as $\overrightarrow{\mathbf{h}^+} = A\mathbf{h}^+$ where the addition and the multiplications are over $\mathbf{GF}(v)$.

Consider a set of t different permutation vectors $\overrightarrow{\mathbf{h}^{(1)}}, \overrightarrow{\mathbf{h}^{(2)}}, \ldots, \overrightarrow{\mathbf{h}^{(t)}}$ corresponding to the vectors $\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \ldots, \mathbf{h}^{(t)}$ of length t. Let H be a $t \times t$ matrix such that its ith column is $\mathbf{h}^{(i)}$, for $1 \leq i \leq t$. Then AH is a $v^t \times t$ matrix which has $\overrightarrow{\mathbf{h}^{(i)}}$ as its ith column. This set of t permutation vectors is *covering* if AH is a $\mathsf{CA}(v^t; t, t, v)$, i.e., each of the t-tuples from $\{0, 1, \ldots, v-1\}^t$ occurs exactly once as a row of AH. Otherwise, the set of t permutation vectors is called *non-covering*. Next we provide a useful characterization of a set of covering permutation vectors. This result is a generalization of the corresponding result from Sherwood *et al.* [51]:

Lemma 15. *A set of t different permutation vectors $\overrightarrow{\mathbf{h}^{(1)}}, \overrightarrow{\mathbf{h}^{(2)}}, \ldots, \overrightarrow{\mathbf{h}^{(t)}}$ corresponding to the vectors $\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \ldots, \mathbf{h}^{(t)}$, each of length t, is covering if and only if H (as defined above) is invertible (i.e., non-degenerate or non-singular).*

Proof. Let $i, j \in \{0, 1, \ldots, v^t - 1\}$ such that $\mathbf{b}^{(i)}H = \mathbf{b}^{(j)}H$. Setting $\mathbf{a} = \mathbf{b}^{(i)} - \mathbf{b}^{(j)}$, we obtain the following system of t linear equations in t unknowns $a_0, a_1, \ldots, a_{t-1}$ over $\mathbf{GF}(v)$

$$\mathbf{a}H = \mathbf{0}. \tag{7.1}$$

If Equation 7.1 has a non-trivial solution then the ith and jth rows of AH represent the same t-tuple from $\{0, 1, \ldots, v-1\}^t$. Therefore, the set of t permutation vectors is non-covering if and only if Equation 7.1 has a non-trivial solution. Hence, the conclusion follows. $\square$

Now consider a set of t different Sherwood permutation vectors $\overrightarrow{\mathbf{h}^{(1)+}}, \overrightarrow{\mathbf{h}^{(2)+}}, \ldots, \overrightarrow{\mathbf{h}^{(t)+}}$ corresponding to the vectors $\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \ldots, \mathbf{h}^{(t)}$ of length $t-1$. Let H^+ be a $t \times t$ matrix such that its ith column is $\mathbf{h}^{(i)+}$, for $1 \leq i \leq t$. Then AH^{+} is a $v^t \times t$ matrix that has $\overrightarrow{\mathbf{k}^{(i)+}}$ as its ith column. This set of t permutation vectors is called *covering* if AH^{+} is a $\mathsf{CA}(v^t; t, t, v)$, i.e., each of the t-tuples from $\{0, 1, \ldots, v-1\}^t$ occurs exactly once as a row of AH^{+}. Otherwise, the set of t permutation vectors is called *non-covering.* We have the following corollary of Lemma 15:

Corollary 16. *[51] A set of t different Sherwood permutation vectors $\overrightarrow{\mathbf{h}^{(1)+}}, \overrightarrow{\mathbf{h}^{(2)+}}, \ldots, \overrightarrow{\mathbf{h}^{(t)+}}$ corresponding to the vectors $\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \ldots, \mathbf{h}^{(t)}$ of length $t-1$ is covering if and only if H^{+} (as defined above) is invertible (i.e. non-degenerate or non-singular).*

7.2 Covering Perfect Hash Families

A *covering perfect hash family,* $\mathsf{CPHF}(n; t, k, v)$ is an $n \times k$ array where each entry is chosen from $\{0, 1, \ldots, v^t - 1\}$ such that for any combination of t columns there is at least one row so that the t permutation vectors, corresponding to the entries (when interpreted as vectors of length t given by their base-v representations) in this set of columns, is covering.

Similarly, a *Sherwood covering perfect hash family,* $\mathsf{SCPHF}(n; t, k, v)$ is an $n \times k$ array where each entry is chosen from $\{0, 1, \ldots, v^{t-1} - 1\}$ such that for any combination of t columns there is at least one row so that the t Sherwood permutation vectors, corresponding to the entries (when interpreted as vectors of length $t-1$ given by their base-v representations) in this set of columns, is covering.

Every permutation vector has 0 as its first entry. If we replace each entry of a $\mathsf{CPHF}(n; t, k, v)$ by the permutation vector corresponding to the entry then we obtain a $\mathsf{CA}(n \cdot v^t; t, k, v)$ that has n all-0 rows of which $n-1$ can be deleted. Therefore, we actually obtain a $\mathsf{CA}(n \cdot (v^t - 1) + 1; t, k, v)$ from a $\mathsf{CPHF}(n; t, k, v)$. Notice that every Sherwood permutation vector has the same v symbols $(0, 1, \ldots, v-1)^{\intercal}$ in the first v positions. Therefore, we employ a *shortened permutation vector* that consists of the last $v^t - v$ entries. We can obtain a $\mathsf{CA}(n \cdot (v^t - v) + v; t, k, v)$ from a $\mathsf{SCPHF}(n; t, k, v)$ by replacing each entry by the corresponding shortened Sherwood permutation vector.

7.3 Asymptotic Upper Bounds on $\mathsf{CAN}(t, k, v)$

We apply Lovász local lemma once again to obtain an upper bound on the size of covering perfect hash families. To that end, we need to count the $t \times t$ invertible matrices H (and H^+). The following result was first obtained by Galois in 1832 and communicated in the last letter of his life.

Lemma 17. *[24] Let v be a prime power. Then the number of invertible $t \times t$ matrices over $\mathbf{GF}(v)$ is $\prod_{s=0}^{t-1}(v^t - v^s)$.*

Proof. In an invertible $t \times t$ matrix the ith column is linearly independent of the previous $i-1$ columns. But the previous $i-1$ columns span a subspace of size v^{i-1}. Therefore, there are $v^t - v^{i-1}$ choices for the ith column. □

We have a similar result for H^+.

Lemma 18. *Let v be a prime power. Then the number of invertible $t \times t$ matrices over $\mathbf{GF}(v)$ that has all 1s in the first row is $\prod_{s=1}^{t-1}(v^t - v^s)$.*

Proof. The argument from the previous lemma can be applied to the t rows, noting that there is a single fixed choice for the first row instead of $v^t - 1$ different choices. □

Next we provide improved upper bounds on $\mathsf{CAN}(t, k, v)$.

Theorem 19. *Let t, k, v be integers with $kt \geq 2$, $k \geq 2t$ and v is a prime power. Then*

$$\mathsf{CAN}(t, k, v) \leq (v^t - 1) \cdot \left\lceil \frac{1 + \log d}{\log(1/q)} \right\rceil + 1 \tag{7.2}$$

and

$$\mathsf{CAN}(t,k,v) \leq (v^t - v) \cdot \left\lceil \frac{1+\log d}{\log(1/q')} \right\rceil + v \tag{7.3}$$

where $d = \binom{k}{t} - \binom{k-t}{t}$, $q = 1 - \frac{\prod_{i=1}^{t-1}(v^t - v^i)}{(v^t-1)^{t-1}}$ *and* $q' = 1 - \prod_{i=1}^{t-1}(1 - \frac{1}{v^i})$.

Proof. Let A be an $n \times k$ array with entries chosen independently from $\{1, 2 \ldots, v^t - 1\}$ with uniform probability. Each entry can be represented by a vector of length t corresponding to its base v representation. Therefore, with t such entries we can construct a matrix whose jth column corresponds to the jth entry, $1 \leq j \leq t$. (We do not consider 0 as an entry because a matrix containing an all zero column is always singular.) Let $\tau \in \binom{[k]}{t}$ be a set of t columns. Let $H_{\tau,i}$ be the $t \times t$ matrix constructed with the t entries from the ith row of A that are indexed by the columns in τ. By Lemma 17, the probability that $H_{\tau,i}$ is invertible is

$$\frac{\prod_{s=0}^{t-1}(v^t - v^s)}{(v^t-1)^t} = \frac{\prod_{s=1}^{t-1}(v^t - v^s)}{(v^t-1)^{t-1}}.$$

Let B_τ denote the event that none of the matrices $H_{\tau,i}$, $1 \leq i \leq n$ is invertible. Since the entries are chosen independently, it is easy to see that

$$\Pr[B_\tau] = \left(1 - \frac{\prod_{s=1}^{t-1}(v^t - v^s)}{(v^t-1)^{t-1}}\right)^n.$$

Then, by the Lovász local lemma (Lemma 5) and Lemma 15, we obtain that when $n \geq \frac{1+\log d}{\log(1/q)}$ A is a $\mathsf{CPHF}(n;t,k,v)$ where $d = \binom{k}{t} - \binom{k-t}{t}$ and $q = 1 - \frac{\prod_{i=1}^{t-1}(v^t - v^i)}{(v^t-1)^{t-1}}$. Now the upper bound 7.2 follows from the natural construction of a covering array from a covering perfect hash family described earlier.

Similarly, if we consider an $n \times k$ array A' where each entry is chosen independently from $\{0, 1, \ldots, v^{t-1} - 1\}$ with uniform probability, then each entry can be interpreted as a vector of length $t-1$ given by its base v representation. Let $H_{\tau,i}$ be the $(t-1)\times t$ matrix constructed with the t entries from the ith row of A such that the t entries are indexed by the columns in τ. Let $H^{+}_{\tau,i}$ denote the $t\times t$ matrix obtained from $H_{\tau,i}$ by adding a row that consists of all ones at the beginning. If C_τ denotes the event that none of the $H^{+}_{\tau,i}$s is invertible for $1 \le i \le n$, then by Lemma 18 we have

$$\Pr[C_\tau] = \left(1 - \prod_{i=1}^{t-1}\left(1 - \frac{1}{v^i}\right)\right)^n .$$

Once again, applying the Lovász local lemma and Corollary 16, we obtain that when $n \ge \frac{1+\log d}{\log(1/q')}$, where $d = \binom{k}{t} - \binom{k-t}{t}$ and $q' = 1 - \prod_{i=1}^{t-1}(1 - \frac{1}{v^i})$, A' is an $\mathsf{SCPHF}(n; t, k, v)$. The upper bound in (7.3) follows from the natural construction of a covering array from a Sherwood covering perfect hash family described earlier. □

Now $q' > q$, because

$$\frac{\prod_{s=1}^{t-1}(v^t - v^s)}{(v^t-1)^{t-1}} > \frac{\prod_{s=1}^{t-1}(v^t - v^s)}{(v^t)^{t-1}} = \prod_{s=1}^{t-1}\left(1 - \frac{1}{v^s}\right). \tag{7.4}$$

Therefore, asymptotically, the bound in (7.2) is tighter than the bound in (7.3).

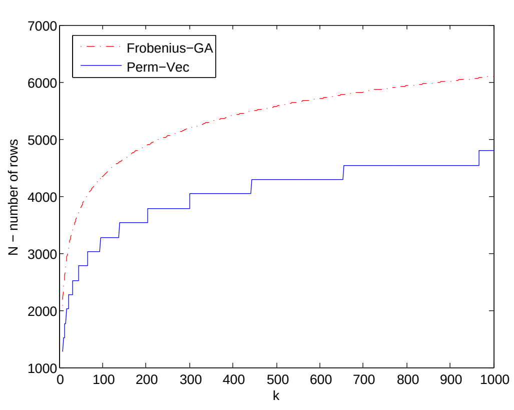
In Figure 7.1 we compare the Frobenius group action based bound derived in Theorem 11 against (7.3). We show two different combinations of t and v values: $t = 4$, $v = 4$ and $t = 6$, $v = 5$. The permutation vector based bounds are much tighter than the Frobenius group action based bound.

We compute upper bounds on $d(t, v)$ from the bounds in (7.2) and in (7.3) to make an asymptotic comparison with the Frobenius bound (Theorem 11) —

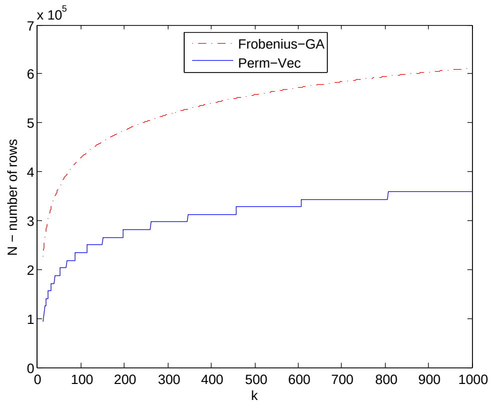
$$d(t,v) \le \frac{(v^t-1)(t-1)}{\log(1/q)}, \tag{7.5}$$

and

$$d(t,v) \le \frac{(v^t-v)(t-1)}{\log(1/q')}, \tag{7.6}$$



(a) $t = 4$, $v = 4$.



(b) $t = 6$, $v = 5$.

Figure 7.1: Comparison of $\mathsf{CAN}(t, k, v)$ upper bounds obtained from the Frobenius group action based method (Theorem 11) and the Sherwood permutation vector based method (Equation 7.3)

.

where q and q' are as defined in Theorem 19.

For $t = 2$, the inequalities (6.1) and (7.6) give the same upper bound. For $t = 3$, we obtain from (6.1)

$$d(3, v) \leq \frac{2v(v-1)}{\log\left(\frac{v^2}{v^2-v+1}\right)}, \tag{7.7}$$

and from (7.6) we obtain

$$d(3, v) \leq \frac{2v(v-1)(v+1)}{\log\left(\frac{v^3}{v^2+v-1}\right)}, \tag{7.8}$$

because for $t = 3$, $q' = 1 - \left(1 - \frac{1}{v}\right)\left(1 - \frac{1}{v^2}\right) = \frac{1}{v} + \frac{1}{v^2} - \frac{1}{v^3}$. Ignoring $2v(v-1)$ from the numerator of the inequality (7.7) we obtain

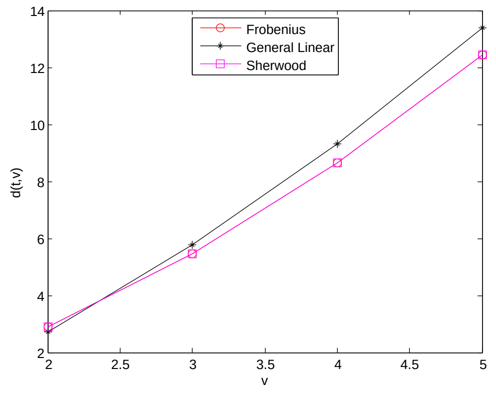
$$\begin{aligned}\frac{1}{\log\left(\frac{v^2}{v^2-v+1}\right)} &> \frac{1}{\log\left(\frac{v^2}{v^2-v}\right)}\\ &= \frac{1}{-\log\left(1-\frac{1}{v}\right)}\\ &= \frac{1}{\frac{1}{v}+\frac{1}{2v^2}+\frac{1}{3v^3}+\dots}\\ &> \frac{1}{\frac{1}{v}+\frac{1}{v^2}+\frac{1}{v^3}+\dots}\\ &= v-1.\end{aligned}$$

Similarly, ignoring $2v(v-1)$ from the numerator of the inequality (7.8) we obtain

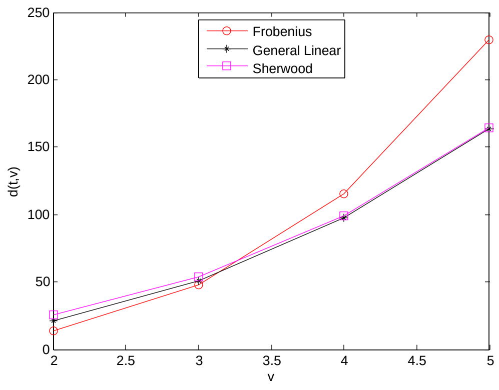
$$\begin{aligned}\frac{v+1}{\log\left(\frac{v^3}{v^2+v-1}\right)} &< \frac{v+1}{\log\left(\frac{v^3}{v^2+v}\right)}\\ &= \frac{v+1}{\log v-\log\left(1+\frac{1}{v}\right)}\\ &= \frac{v+1}{\log v-\frac{1}{v}+\frac{1}{2v^2}-\frac{1}{3v^3}+\dots}\\ &< \frac{v+1}{\log v-\frac{1}{v}}.\end{aligned}$$

Because $v-1 > \frac{v+1}{\log v-\frac{1}{v}}$ for sufficiently large v ($v > 5$), (7.8) is tighter than (7.7). Therefore, for $t = 3$ the Sherwood bound provides approximately a $\log v$ factor of reduction in the $d(t, v)$ upper bound for sufficiently large values of v.

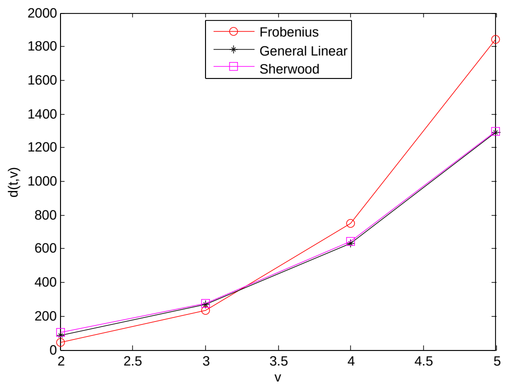
Because of the complicated form of the denominators in the bounds in (7.4) and (7.6), it requires quite a cumbersome computation to show that they are indeed tighter bounds than (6.1) for $t > 3$. Figure 7.2 compares the three upper bounds on $d(t, v)$ obtained from (6.1), (7.5) and (7.6). For $t = 2$, bounds in (6.1) and (7.6) give the same result, as was already verified by substituting $t = 2$ in these bounds. Both of them are tighter than the bound in (7.5). However, for $t \geq 3$ the bounds in (7.5) and (7.6) are tighter than (6.1) when $v > 3$. The bound in (7.5) is tighter than the bound in (7.6) which is not surprising in light of the inequality (7.4).



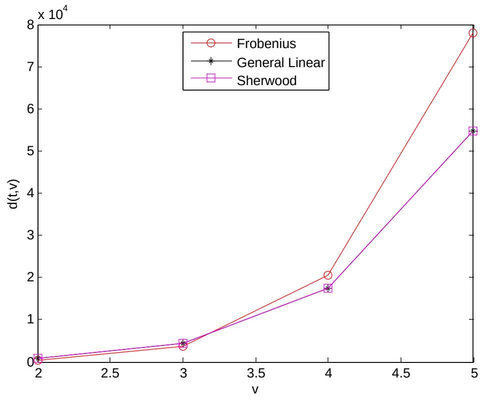
(a) $t = 2$.



(b) $t = 3$.



(c) $t = 4$.

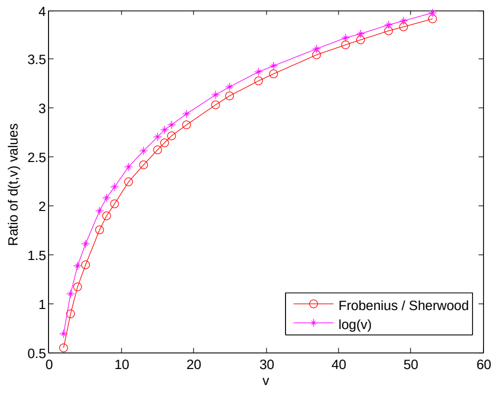


(d) $t = 6$.

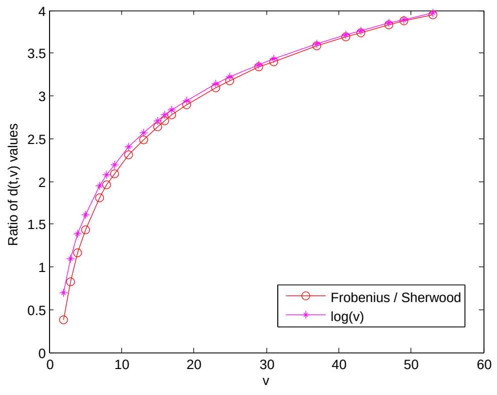
Figure 7.2: Comparison of three $d(t, v)$ upper bounds—Frobenius (6.1), General Linear (7.5), and Sherwood (7.6) for $t = 2, 3, 4, 6$.

Figure 7.3 plots the ratio of the $d(t, v)$ upper bounds obtained from the Frobenius bound (6.1) and the Sherwood bound (7.6) against different values of v (prime power). It is evident from the figure that for $t \geq 3$ this ratio is approximately $\log v$ for sufficiently large values of v.

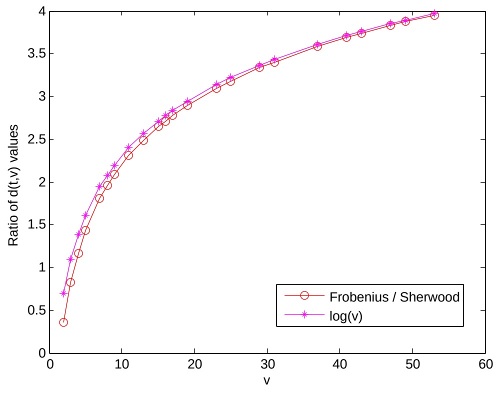
To extend our results to values of v that are not prime powers, choose the smallest prime p with $p \geq v$. Then construct the CPHF or SCPHF over the field of order p. When expanding the result into a covering array, treat any $p - v$ of the symbols as



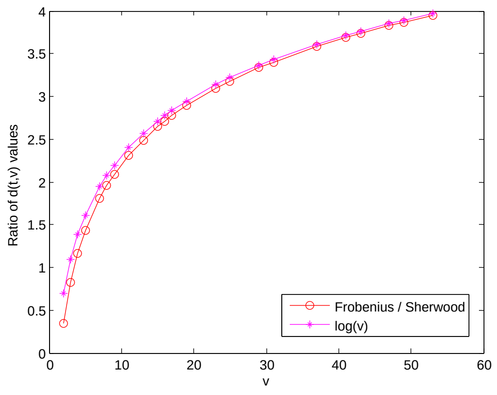
(a) $t = 3$.



(b) $t = 5$.



(c) $t = 6$.



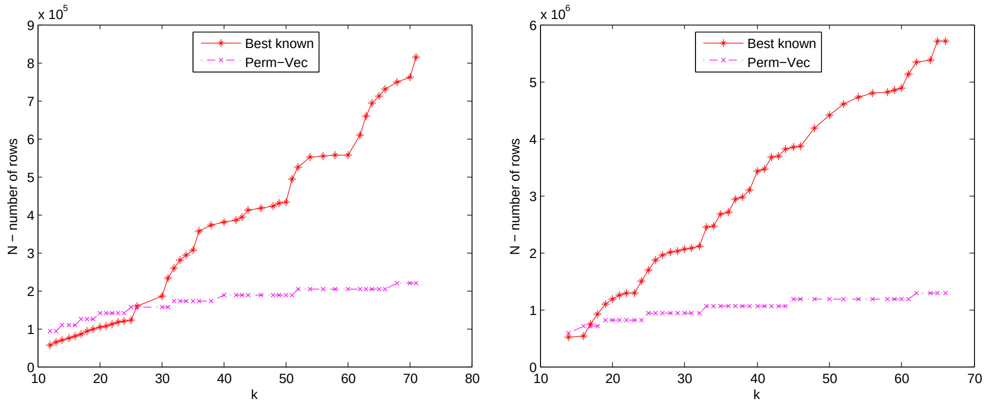
(d) $t = 10$.

Figure 7.3: Ratio of $d(t, v)$ upper bounds obtained by the Frobenius bound (6.1) and the Sherwood bound (7.6) plotted against different (prime power) v values. For comparison $\log v$ is also plotted.

"don't care" positions to obtain a covering array on v symbols. Although the increase in the array size may make the result poor for small values of v, asymptotically the increase is negligible. This can be seen by using the Prime Number Theorem to provide an upper bound on p in terms of v.

7.4 A Moser-Tardos Type Construction Algorithm for Covering Perfect Hash Families

The bounds obtained in Theorem 19 are not only asymptotically the best known, but also they beat the computationally constructed best known covering arrays for relatively small values of k. Figure 7.4 compares the bound in (7.3) against the best known upper bounds reported in Colbourn [17] for $t = 6$, $v = 5$ and $t = 6$, $v = 7$. As we can see, permutation vector based bounds beat the best known results to date by quite a margin when $k \geq 25$. This happens whenever $t = 6$ and $v \geq 5$.



(a) $t = 6$, $v = 5$. (b) $t = 6$, $v = 7$.

Figure 7.4: Comparison of the upper bounds derived from Equation 7.3 and the currently best known upper bounds compiled at Colbourn [17]

.

Therefore, the important question is: Can we construct the covering arrays that are guaranteed by Theorem 19 efficiently? Once again, because the Lovász local lemma forms the core of the probabilistic existence argument, we can easily obtain a Moser-Tardos type construction algorithm that finds covering perfect hash families of the required size (see Section 2.2.1). Algorithm 12 presents the method for constructing a Sherwood covering perfect hash family. This algorithm is quite fast in practice and has been used to obtain covering arrays for k values of up to 200 when $t = 6$ and $v \in \{5, 7\}$.

Table 7.1 presents the computational results obtained from Moser-Tardos type construction algorithms for CPHF and SCPHF. As expected, these algorithms start to outperform all known methods for covering array construction even for fairly small values of k (in the range 17-26 depending on the value of v).

Algorithm 12: Moser-Tardos type algorithm for SCPHF construction.

```
Input: $t$ : strength, $k$ : number of columns, $v$ : number of levels
Output: $A$ : an SCPHF$(n; t, k, v)$
Let $d := \binom{k}{t} - \binom{k-t}{t}$, $q' := 1 - \prod_{i=1}^{t-1}(1 - \frac{1}{v^i})$, and $n := \left\lceil \frac{1+\log d}{\log(1/q')} \right\rceil$;
Construct an $n \times k$ array $A$ where each entry is chosen independently and uniformly at
  random from the set $\{0, 1, \ldots, v^{t-1} - 1\}$;
repeat
    Set covered:= true;
    for each $t$-set of columns $\tau \in \binom{[k]}{t}$ do
        Set degenerate := true;
        for each $i$, $1 \le i \le n$ do
            Construct $H^+_{\tau,i}$ with the entries in the $i$th row of $A$ indexed by the columns in $\tau$;
            if $H^+_{\tau,i}$ is invertible then
                Set degenerate := false;
                break;
            end
        end
        if degenerate = true then
            Set covered:= false;
            Set violated := $\tau$;
            break;
        end
    end
    if covered = false then
        Resample all the entries in the $t$ columns of the set violated;
    end
until covered = true;
Output $A$;
```

k	CPHF	SCPHF	Best-known
13	93745	93725	63835
16	109369	109345	80406
19	124993	124965	98994
24	140617	140585	119500
26	156241	156205	160408
31	156241	156205	233945
39	171865	171825	380039
51	187489	187445	493879
66	203113	203065	730000
113	234361	234305	1043198
114	234361*	249925	1043198
149	249985	249925	1074535
150	249985*	265545	1074535
197	265609	265545	1096907
198	265609*	281165	1097427
199	281233	281165	1101245

(a) $t = 6$, $v = 5$

k	CPHF	SCPHF	Best-known
13	588241	588217	521017
14	588241	521017	521017
17	588241	588217	738871
18	705889	705859	926149
24	823537	823501	1497979
32	941185	941143	2113678
44	1058833	1058785	3819650
61	1176481	1176427	5132527
85	1294129	1294069	7227943
121	1411777	1411711	8179848
171	1529425	1529353	8454598
172	1647073	1646995	8454598

(b) $t = 6$, $v = 7$

Table 7.1: Comparison of covering array sizes obtained from the Moser-Tardos type algorithm for CPHF construction (CPHF), SCPHF construction, i.e., Algorithm 12 (SCPHF) and the best known results from Colbourn [17] (Best-known). To find the size of the covering array for a missing k value in the range, first locate the k value that is at least as large as the given value. The size of the covering array constructed by the algorithm is the same for both these values. The * entries indicate the instances where CPHF method produces smaller covering arrays.

Chapter 8

PARTIAL COVERING ARRAYS

In this chapter, we consider a few variants of covering arrays. Our focus is on finding reasonable relaxations of the covering requirement and establishing upper bounds on the size of the arrays that fulfill such relaxed covering requirements. One of the main results of this chapter tells us that with a minimal relaxation we can construct arrays that are asymptotically smaller in size than covering arrays.

We have seen a number of upper bounds on the $\mathsf{CAN}(t,k,v)$ so far. An (essentially) equivalent but more convenient form of the upper bound in Theorem 2 is:

$$\mathsf{CAN}(t,k,v) \leq (t-1)v^t \log k(1+o(1)). \tag{8.1}$$

A lower bound on the $\mathsf{CAN}(t,k,v)$ results from the inequality $\mathsf{CAN}(t,k,v) \geq v \cdot \mathsf{CAN}(t-1,k-1,v)$ obtained by derivation, together with (2.1), to establish that $\mathsf{CAN}(t,k,v) \geq v^{t-2} \cdot \mathsf{CAN}(2,k-t+2,v) = v^{t-2} \cdot \frac{v}{2}\log(k-t+2)(1+\mathrm{o}(1))$. When $\frac{t}{k} < 1$, we obtain:

$$\mathsf{CAN}(t,k,v) = \Omega(v^{t-1}\log k). \tag{8.2}$$

Because (8.2) ensures that the number of rows in covering arrays can be considerable, researchers have suggested the need for relaxations in which not all interactions must be covered [12, 32, 39, 43] in order to reduce the number of rows. The practical relevance is that each row corresponds to a test to be performed, adding to the cost of testing.

For example, an array *covers a t-set of columns* when it covers each of the v^t interactions on this t-set. Hartman and Raskin [32] consider arrays with a fixed number of rows that cover the *maximum* number of t-sets of columns. A similar question was also considered in Maximoff *et al.* [43]. In Kuhn *et al.* [39] and Maximoff *et al.* [43] a more refined measure of the (partial) coverage of an $N \times k$ array A is introduced. For a given $q \in [0, 1]$, let $\alpha(A, q)$ be the number of $N \times t$ submatrices of A with the property that at least qv^t elements of $[v]^t$ appear in their set of rows; the (q, t)-*completeness* of A is $\alpha(A, q)/\binom{k}{t}$. Then for practical purposes one wants "high" (q, t)-completeness with few rows.

In these works, no theoretical results on partial coverage appear to have been stated; earlier contributions focus on experimental investigations of heuristic construction methods. Our purpose is to initiate a mathematical investigation of arrays offering "partial" coverage. More precisely, we address:

- Can one obtain a significant improvement on the upper bound (8.1) if the set $[v]^t$ is only required to be contained among the rows of *at least* $(1-\epsilon)\binom{k}{t}$ subarrays of A of dimension $N \times t$, for $0 < \epsilon < 1$?

- Can one obtain a significant improvement if, among the rows of *every* $N \times t$ subarray of A, only a (large) *subset* of $[v]^t$ is required to be contained?

- Can one obtain a significant improvement if the set $[v]^t$ is only required to be contained among the rows of *at least* $(1 - \epsilon)\binom{k}{t}$ subarrays of A of dimension $N \times t$, **and** among the rows of each of the $\epsilon\binom{k}{t}$ subarrays that remain, a (large) *subset* of $[v]^t$ is required to be contained, for $0 < \epsilon < 1$?

We answer these questions both theoretically and algorithmically in the following sections.

The results presented in this chapter are based on the work reported in Sarkar, Colbourn, De Bonis and Vaccaro [49].

8.1 Partial Covering Arrays

Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. Let N, t, k, and v be integers such that $k \geq t \geq 2$ and $v \geq 2$. Let A be an $N \times k$ array where each entry is from the set $[v]$. For $I = \{j_1, \ldots, j_\rho\} \subseteq [k]$ where $j_1 < \ldots < j_\rho$, let A_I denote the $N \times \rho$ array in which $A_I(i, \ell) = A(i, j_\ell)$ for $1 \leq i \leq N$ and $1 \leq \ell \leq \rho$; A_I is the projection of A onto the columns in I.

When $1 \leq m \leq v^t$, a *partial m-covering array*, $\mathsf{PCA}(N; t, k, v, m)$, is an $N \times k$ array A with each entry from $[v]$ so that for each t-set of columns $C \in \binom{[k]}{t}$, at least m distinct tuples $x \in [v]^t$ appear as rows in A_C. Hence a covering array $\mathsf{CA}(N; t, k, v)$ is precisely a partial v^t-covering array $\mathsf{PCA}(N; t, k, v, v^t)$.

Theorem 20. *For integers t, k, v, and m where $k \geq t \geq 2$, $v \geq 2$ and $1 \leq m \leq v^t$ there exists a $\mathsf{PCA}(N; t, k, v, m)$ with*

$$N \leq \frac{\ln\left\{\binom{k}{t}\binom{v^t}{m-1}\right\}}{\ln\left(\frac{v^t}{m-1}\right)}. \tag{8.3}$$

.

Proof. Let $r = v^t - m + 1$, and A be a random $N \times k$ array where each entry is chosen independently from $[v]$ with uniform probability. For $C \in \binom{[k]}{t}$, let B_C denote the event that at least r tuples from $[v]^t$ are missing in A_C. The probability that a particular r-set of tuples from $[v]^t$ is missing in A_C is $\left(1 - \frac{r}{v^t}\right)^N$. Applying the union bound to all r-sets of tuples from $[v]^t$, we obtain $\Pr[B_C] \leq \binom{v^t}{r}\left(1 - \frac{r}{v^t}\right)^N$. By linearity of expectation, the expected number of t-sets C for which A_C misses at least r tuples from $[v]^t$ is at most $\binom{k}{t}\binom{v^t}{r}\left(1 - \frac{r}{v^t}\right)^N$. When A has at least $\frac{\ln\left\{\binom{k}{t}\binom{v^t}{m-1}\right\}}{\ln\left(\frac{v^t}{m-1}\right)}$ rows this expected number is less than 1. Therefore, an array A exists with the required number of rows such that for all $C \in \binom{[k]}{t}$, A_C misses at most $r - 1$ tuples from $[v]^t$, i.e., A_C covers at least m tuples from $[v]^t$. □

Theorem 20 can be improved upon using the Lovász local lemma (Lemma 5 in Chapter 2).

Theorem 21. *For integers t, k, v and m where $v, t \geq 2$, $k \geq 2t$ and $1 \leq m \leq v^t$ there exists a* $\mathsf{PCA}(N; t, k, v, m)$ *with*

$$N \leq \frac{1 + \ln\left\{t\binom{k}{t-1}\binom{v^t}{m-1}\right\}}{\ln\left(\frac{v^t}{m-1}\right)}. \tag{8.4}$$

Proof. When $k \geq 2t$, each event B_C with $C \in \binom{[k]}{t}$ (that is, at least $v^t - m + 1$ tuples are missing in A_C) is independent of all but at most $\binom{t}{1}\binom{k-1}{t-1} < t\binom{k}{t-1}$ events in $\{B_{C'} : C' \in \binom{[k]}{t} \setminus \{C\}\}$. Applying Lemma 5, $\Pr[\wedge_{C\in\binom{[k]}{t}}\overline{B_C}] > 0$ when

$$\mathrm{e}\binom{v^t}{r}\left(1 - \frac{r}{v^t}\right)^N t\binom{k}{t-1} \leq 1. \tag{8.5}$$

Solve (8.5) to obtain the required upper bound on N. □

When $m = v^t$, apply the Taylor series expansion to obtain $\ln\left(\frac{v^t}{m-1}\right) \geq \frac{1}{v^t}$, and thereby recover the upper bound (8.1). Theorem 21 implies:

Corollary 22. *Given $q \in [0, 1]$ and integers $2 \leq t \leq k$, $v \geq 2$, there exists an $N \times k$ array on $[v]$ with (q, t)-completeness equal to 1 (i.e.,* maximal*), whose number N of rows satisfies*

$$N \leq \frac{1 + \ln\left\{t\binom{k}{t-1}\binom{v^t}{qv^t-1}\right\}}{\ln\left(\frac{v^t}{qv^t-1}\right)}.$$

Rewriting (8.4), setting $r = v^t - m + 1$, and using the Taylor series expansion of $\ln\left(1 - \frac{r}{v^t}\right)$, we get

$$N \leq \frac{1 + \ln\left\{t\binom{k}{t-1}\binom{v^t}{r}\right\}}{\ln\left(\frac{v^t}{v^t-r}\right)} \leq \frac{v^t(t-1)\ln k}{r}\left\{1 - \frac{\ln r}{\ln k} + o(1)\right\}. \tag{8.6}$$

Hence when $r = v(t-1)$ (or equivalently, $m = v^t - v(t-1) + 1$), there is a partial m-covering array with $\Theta(v^{t-1}\ln k)$ rows. This matches the lower bound (8.2) asymptotically for covering arrays by missing, in each t-set of columns, *no more* than $v(t-1)-1$ of the v^t possible rows.

The dependence of the bound (8.4) on the number of v-ary t-vectors that must appear in the t-tuples of columns is particularly of interest when test suites are run sequentially until a fault is revealed, as in Bryce *et al.* [5]. Indeed the arguments here may have useful consequences for the rate of fault detection.

As discussed in Section 2.2.1, we can obtain a randomized algorithm that constructs partial m-covering arrays with the number of rows N guaranteed by Theorem 21. Patterned on Algorithm 1, Algorithm 13 constructs a partial m-covering array with exactly the same number of rows as in (8.4) in expected polynomial time. Indeed, for fixed t, the expected number of times the resampling step (line 13) is repeated is linear in k (see Moser and Tardos [45] and Section 2.2.1 for more details).

8.2 Almost Partial Covering Arrays

For $0 < \epsilon < 1$, an *ϵ-almost partial m-covering array*, $\mathsf{APCA}(N; t, k, v, m, \epsilon)$, is an $N \times k$ array A with each entry from $[v]$ so that for at least $(1-\epsilon)\binom{k}{t}$ column t-sets $C \in \binom{[k]}{t}$, A_C covers at least m distinct tuples $x \in [v]^t$. Again, a covering array $\mathsf{CA}(N; t, k, v)$ is precisely an $\mathsf{APCA}(N; t, k, v, v^t, \epsilon)$ when $\epsilon < 1/\binom{k}{t}$. Our first result on *$\epsilon$-almost* partial m-covering arrays is the following.

Theorem 23. *For integers t, k, v, m and real ϵ where $k \geq t \geq 2$, $v \geq 2$, $1 \leq m \leq v^t$ and $0 \leq \epsilon \leq 1$, there exists an $\mathsf{APCA}(N; t, k, v, m, \epsilon)$ with*

$$N \leq \frac{\ln\left\{\binom{v^t}{m-1}/\epsilon\right\}}{\ln\left(\frac{v^t}{m-1}\right)}. \tag{8.7}$$

Proof. Parallelling the proof of Theorem 20 we compute an upper bound on the expected number of t-sets $C \in \binom{[k]}{t}$ for which A_C misses at least r tuples $x \in [v]^t$. When this expected number is at most $\epsilon\binom{k}{t}$, an array A is guaranteed to exist with at

Algorithm 13: Moser-Tardos type algorithm for partial m-covering arrays.

```
Input: Integers $N, t, k, v$ and $m$ where $v, t \geq 2$, $k \geq 2t$ and $1 \leq m \leq v^t$
Output: $A$ : a $\mathsf{PCA}(N; t, k, v, m)$
1  Let $N := \frac{1+\ln\left\{t\binom{k}{t-1}\binom{v^t}{m-1}\right\}}{\ln\left(\frac{v^t}{m-1}\right)}$;
2  Construct an $N \times k$ array $A$ where each entry is chosen independently and
     uniformly at random from $[v]$;
3  repeat
4      Set covered:= true;
5      for each column $t$-set $C \in \binom{[k]}{t}$ do
6          if $A_C$ does not cover at least $m$ distinct $t$-tuples $x \in [v]^t$ then
7              Set covered:= false;
8              Set missing-column-set := $C$;
9              break;
10         end
11     end
12     if covered = false then
13         Choose all the entries in the $t$ columns of missing-column-set
             independently and uniformly at random from $[v]$;
14     end
15 until covered = true;
16 Output $A$;
```

least $(1-\epsilon)\binom{k}{t}$ t-sets of columns $C \in \binom{[k]}{t}$ such that A_C misses at most $r-1$ distinct tuples $x \in [v]^t$. Thus A is an $\mathsf{APCA}(N; t, k, v, m, \epsilon)$. To establish the theorem, solve the following for N:

$$\binom{k}{t}\binom{v^t}{r}\left(1-\frac{r}{v^t}\right)^N \le \epsilon\binom{k}{t}.$$

□

When $\epsilon < 1/\binom{k}{t}$ we recover the bound from Theorem 20 for partial m-covering arrays. In terms of (q,t)-completeness, Theorem 23 yields the following.

Corollary 24. *For $q \in [0,1]$ and integers $2 \le t \le k$, $v \ge 2$, there exists an $N \times k$ array on $[v]$ with (q,t)-completeness equal to $1-\epsilon$, with*

$$N \le \frac{\ln\left\{\binom{v^t}{m-1}/\epsilon\right\}}{\ln\left(\frac{v^t}{m-1}\right)}.$$

When $m = v^t$, an ϵ-almost covering array exists with $N \le v^t \ln\left(\frac{v^t}{\epsilon}\right)$ rows. Improvements result by focussing on covering arrays in which the symbols are acted on by a finite group. In this setting, one chooses orbit representatives of rows that collectively cover orbit representatives of t-way interactions under the group action; see Colbourn [19], for example. Such group actions have been used in direct and computational methods for covering arrays [11, 44], and in randomized and derandomized methods [19, 47, 48].

We employ the sharply transitive action of the cyclic group of order v, adapting the earlier arguments using methods from Sarkar and Colbourn [48]:

Theorem 25. *For integers t, k, v and real ϵ where $k \ge t \ge 2$, $v \ge 2$ and $0 \le \epsilon \le 1$ there exists an $\mathsf{APCA}(N; t, k, v, v^t, \epsilon)$ with*

$$N \le v^t \ln\left(\frac{v^{t-1}}{\epsilon}\right). \tag{8.8}$$

Proof. The action of the cyclic group of order v partitions $[v]^t$ into v^{t-1} orbits, each of length v. Let $n = \lfloor \frac{N}{v} \rfloor$ and let A be an $n \times k$ random array where each entry is chosen independently from the set $[v]$ with uniform probability. For $C \in \binom{[k]}{t}$, A_C *covers the orbit* X if at least one tuple $x \in X$ is present in A_C. The probability that the orbit X is not covered in A is $\left(1 - \frac{v}{v^t}\right)^n = \left(1 - \frac{1}{v^{t-1}}\right)^n$. Let D_C denote the event that A_C does not cover at least one orbit. Applying the union bound, $\Pr[D_C] \leq v^{t-1}\left(1 - \frac{1}{v^{t-1}}\right)^n$. By linearity of expectation, the expected number of column t-sets C for which D_C occurs is at most $\binom{k}{t} v^{t-1}\left(1 - \frac{1}{v^{t-1}}\right)^n$. As earlier, set this expected value to be at most $\epsilon\binom{k}{t}$ and solve for n. An array exists that covers all orbits in at least $(1-\epsilon)\binom{k}{t}$ column t-sets. Develop this array over the cyclic group to obtain the desired array. □

As in Sarkar and Colbourn [48], further improvements result by considering a group, like the Frobenius group, that acts sharply 2-transitively on $[v]$. When v is a prime power, the *Frobenius group* is the group of permutations of $\mathbb{F}_v$ of the form $\{x \mapsto ax + b \,:\, a, b \in \mathbb{F}_v,\ a \neq 0\}$.

Theorem 26. *For integers t, k, v and real ϵ where $k \geq t \geq 2$, $v \geq 2$, v is a prime power and $0 \leq \epsilon \leq 1$ there exists an $\mathsf{APCA}(N; t, k, v, v^t, \epsilon)$ with*

$$N \leq v^t \ln\left(\frac{2v^{t-2}}{\epsilon}\right) + v. \tag{8.9}$$

Proof. The action of the Frobenius group partitions $[v]^t$ into $\frac{v^{t-1}-1}{v-1}$ orbits of length $v(v-1)$ (full orbits) each and 1 orbit of length v (a short orbit). The short orbit consists of tuples of the form $(x_1, \ldots, x_t) \in [v]^t$ where $x_1 = \ldots = x_t$. Let $n = \left\lfloor \frac{N-v}{v(v-1)} \right\rfloor$ and let A be an $n \times k$ random array where each entry is chosen independently from the set $[v]$ with uniform probability. Our strategy is to construct A so that it covers all full orbits for the required number of arrays $\{A_C : C \in \binom{[k]}{t}\}$. Develop A over the Frobenius group and add v rows of the form $(x_1, \ldots, x_k) \in [v]^t$ with $x_1 = \ldots = x_k$ to obtain an $\mathsf{APCA}(N; t, k, v, v^t, \epsilon)$ with the desired value of N. Following the lines of the proof of Theorem 25, A covers all full orbits in at least $(1-\epsilon)\binom{k}{t}$ column t-sets C when

$$\binom{k}{t}\frac{v^{t-1}-1}{v-1}\left(1-\frac{v-1}{v^{t-1}}\right)^n \leq \epsilon\binom{k}{t}.$$

Because $\frac{v^{t-1}-1}{v-1} \leq 2v^{t-2}$ for $v \geq 2$, we obtain the desired bound. $\square$

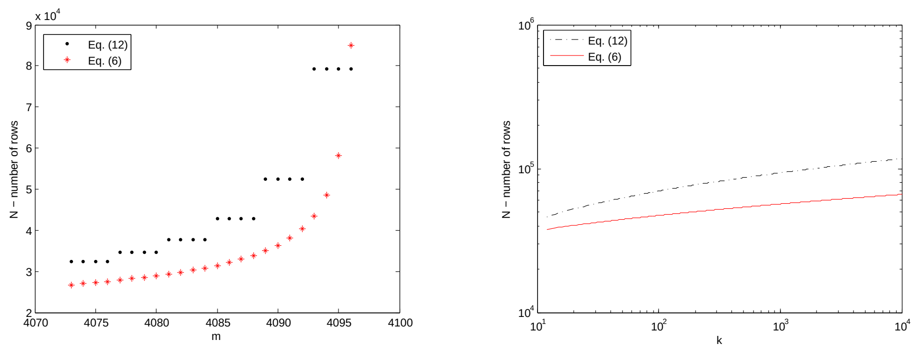
Using group action when $m = v^t$ affords useful improvements. Does this improvement extend to cases when $m < v^t$? Unfortunately, the answer appears to be no. Consider the case for $\mathsf{PCA}(N; t, k, v, m)$ when $m \leq v^t$ using the action of the cyclic group of order v on $[v]^t$. Let A be a random $n \times k$ array over $[v]$. When $v^t - vs + 1 \leq m \leq v^t - v(s-1)$ for $1 \leq s \leq v^{t-1}$, this implies that for all $C \in \binom{[k]}{t}$, A_C misses at most $s-1$ orbits of $[v]^t$. Then we obtain that $n \leq \left(1+\ln\left(t\binom{k}{t-1}\binom{v^{t-1}}{s}\right)\right)/\ln\left(\frac{v^{t-1}}{v^{t-1}-s}\right)$. Developing A over the cyclic group we obtain a $\mathsf{PCA}(N; t, k, v, m)$ with

$$N \leq v\frac{1+\ln\left\{\binom{k}{t-1}\binom{v^{t-1}}{s}\right\}}{\ln\left(\frac{v^{t-1}}{v^{t-1}-s}\right)}. \tag{8.10}$$

Figure 8.1 compares (8.10) and (8.4). In Figure 8.1(a) we plot the size of the partial m-covering array as obtained by (8.10) and (8.4) for $v^t - 6v + 1 \leq m \leq v^t$ and $t = 6$, $k = 20$, $v = 4$. Except when $m = v^t = 4,096$, the covering array case, (8.4) outperforms (8.10). Similarly, Figure 8.1(b) shows that for $m = v^t - v = 4,092$, (8.4) consistently outperforms (8.10) for all values of k when $t = 6$, $v = 4$. We observe similar behavior for different values of t and v.

Next we consider even stricter coverage restrictions, combining Theorems 21 and 25.

Theorem 27. *For integers t, k, v, m and real ϵ where $k \geq t \geq 2$, $v \geq 2$, $0 \leq \epsilon \leq 1$ and $m \leq v^t + 1 - \frac{\ln k}{\ln(v/\epsilon^{1/(t-1)})}$ there exists an $N \times k$ array A with entries from $[v]$ such that*



(a) $t = 6,\ k = 20,\ v = 4$ (b) $t = 6,\ v = 4,\ m = v^t - v$

Figure 8.1: Comparison of (8.10) and (8.4). Figure (a) compares the sizes of the partial m-covering arrays when $v^t - 6v + 1 \leq m \leq v^t$. Except for $m = v^t = 4,096$ the bound from (8.4) outperforms the bound obtained by assuming group action. Figure (b) shows that for $m = v^t - v = 4,092$, (8.4) outperforms (8.10) for all values of k.

1. *for each* $C \in \binom{[k]}{t}$, A_C *covers at least* m *tuples* $x \in [v]^t$,

2. *for at least* $(1-\epsilon)\binom{k}{t}$ *column* t*-sets* C, A_C *covers all tuples* $x \in [v]^t$, *and*

3. $N = O(v^t \ln \left(\frac{v^{t-1}}{\epsilon}\right))$.

Proof. We vertically juxtapose a partial m-covering array and an ϵ-almost v^t-covering array. For $r = \frac{\ln k}{\ln(v/\epsilon^{1/(t-1)})}$ and $m = v^t - r + 1$, (8.6) guarantees the existence of a partial m-covering array with $v^t \ln \left(\frac{v^{t-1}}{\epsilon}\right) \{1 + \mathrm{o}(1)\}$ rows. Theorem 25 guarantees the existence of an ϵ-almost v^t-covering array with at most $v^t \ln \left(\frac{v^{t-1}}{\epsilon}\right)$ rows. □

Corollary 28. *There exists an* $N \times k$ *array* A *such that:*

1. *for any* t*-set of columns* $C \in \binom{[k]}{t}$, A_C *covers at least* $m \leq v^t + 1 - v(t-1)$ *distinct* t*-tuples* $x \in [v]^t$,

2. *for at least* $\left(1 - \frac{v^{t-1}}{k^{1/v}}\right)\binom{k}{t}$ *column* t*-sets* C*,* A_C *covers all the distinct* t*-tuples* $x \in [v]^t$*, and*

3. $N = O(v^{t-1} \ln k)$.

Proof. Apply Theorem 27 with $m = v^t + 1 - \frac{\ln k}{\ln(v/\epsilon^{1/(t-1)})}$. There are at most $\frac{\ln k}{\ln(v/\epsilon^{1/(t-1)})} - 1$ missing t-tuples $x \in [v]^t$ in the A_C for each of the at most $\epsilon\binom{k}{t}$ column t-sets C that do not satisfy the second condition of Theorem 27. To bound from above the number of missing tuples to a certain small function $f(t)$ of t, it is sufficient that $\epsilon \leq v^{t-1}\left(\frac{1}{k}\right)^{\frac{t-1}{f(t)+1}}$. Then the number of missing t-tuples $x \in [v]^t$ in A_C is bounded from above by $f(t)$ whenever ϵ is not larger than

$$v^{t-1}\left(\frac{1}{k}\right)^{\frac{t-1}{f(t)+1}}. \tag{8.11}$$

On the other hand, in order for the number $N = O\left(v^{t-1}\ln\left(\frac{v^{t-1}}{\epsilon}\right)\right)$ of rows of A to be asymptotically equal to the lower bound (8.2), it suffices that ϵ is not smaller than

$$\frac{v^{t-1}}{k^{\frac{1}{v}}}. \tag{8.12}$$

When $f(t) = v(t-1) - 1$, (8.11) and (8.12) agree asymptotically, completing the proof. □

Once again we obtain a size that is $O(v^{t-1}\log k)$, a goal that has not been reached for covering arrays. This is evidence that even a small relaxation of covering arrays provides arrays of the best sizes one can hope for.

Next we consider the efficient construction of the arrays whose existence is ensured by Theorem 27. Algorithm 14 is a randomized method to construct an $\mathsf{APCA}(N; t, k, v, m, \epsilon)$ of a size N that is very close to the bound of Theorem 23. By Markov's inequality the condition in line 9 of Algorithm 14 is met with probability at most $\frac{1}{2}$. Therefore, the expected number of times the loop in line 2 repeats is at most 2.

To prove Theorem 23, t-wise independence among the variables is sufficient. Hence, Algorithm 14 can be derandomized using t-wise independent random variables. We can also derandomize the algorithm using the method of conditional expectation. In this method, we construct A by considering the k columns one by one and fixing all N entries of a column. Given a set of already fixed columns, to fix the entries in the next column we consider all possible v^N choices, and choose one that provides the maximum conditional expectation of the number of column t-sets $C \in \binom{[k]}{t}$ such that A_C covers at least m tuples $x \in [v]^t$. Because $v^N = O(\mathsf{poly}(1/\epsilon))$, this derandomized algorithm constructs the desired array in polynomial time. Similar randomized and derandomized strategies can be applied to construct the array guaranteed by Theorem 25. Together with Algorithm 13 this implies that the array in Theorem 27 is also efficiently constructible.

8.3 Final Remarks

We have shown that by relaxing the coverage requirement of a covering array somewhat, powerful upper bounds on the sizes of the arrays can be established. Indeed the upper bounds are substantially smaller than the best known bounds for a covering array; they are of the same order as the *lower* bound for the $\mathsf{CAN}(t, k, v)$. As importantly, the techniques not only provide asymptotic bounds but also randomized polynomial time construction algorithms for such arrays.

Our approach seems flexible enough to handle variations of these problems. For instance, some applications require arrays that satisfy, for different subsets of columns, different coverage or separation requirements [16]. In Gravier and Ycart [29] several interesting examples of combinatorial problems are presented that can be unified and expressed in the framework of S-constrained matrices. Given a set of vectors S each of length t, an $N \times k$ matrix M is *S-constrained* if for every t-set $C \in \binom{[k]}{t}$, M_C

Algorithm 14: Randomized algorithm for ϵ-almost partial m-covering arrays.

```
   Input: Integers $N, t, k, v$ and $m$ where $v, t \geq 2$, $k \geq 2t$ and $1 \leq m \leq v^t$, and
          real $0 < \epsilon < 1$
   Output: $A$ : an $\mathsf{APCA}(N; t, k, v, m, \epsilon)$
 1 Let $N := \frac{\ln\left\{2\binom{v^t}{m-1}/\epsilon\right\}}{\ln\left(\frac{v^t}{m-1}\right)}$;
 2 repeat
 3     Construct an $N \times k$ array $A$ where each entry is chosen independently and
         uniformly at random from $[v]$;
 4     Set $isAPCA$:= true;
 5     Set $defectiveCount$:= 0;
 6     for each column $t$-set $C \in \binom{[k]}{t}$ do
 7         if $A_C$ does not cover at least $m$ distinct $t$-tuples $x \in [v]^t$ then
 8             Set $defectiveCount$:= $defectiveCount$ + 1;
 9             if defectiveCount > $\lfloor \epsilon\binom{k}{t} \rfloor$ then
10                 Set $isAPCA$:= false;
11                 break;
12             end
13         end
14     end
15 until $isAPCA$ = $true$;
16 Output $A$;
```

contains as a row each of the vectors in S. The parameter to optimize is, as usual, the number of rows of M. One potential direction is to ask for arrays that, in every t-tuple of columns, cover at least m of the vectors in S, or that all vectors in S are covered by all but a small number of t-tuples of columns. Exploiting the structure of the members of S appears to require an extension of the results developed here.

Chapter 9

CONCLUSION

In this chapter, we summarize the key contributions of this thesis and point out the main themes that emerge out of it. In Section 9.2, we discuss a number of open problems that still remain unresolved. Some of these problems may lead to interesting and fruitful research directions.

9.1 Key Ideas and Main Themes

In this thesis, we have established a successive series of upper bounds on the $\mathsf{CAN}(t, k, v)$, each bound being tighter than the previous one. The probabilistic methods have played an important role. A unifying thread among the various methods used to obtain these bounds is the strategy of exploiting the *limited dependence* structure among the coverage events using the Lovász local lemma. Although the strategy of applying the Lovász local lemma to covering arrays was pioneered by Godbole et al. [27], this thesis has made extensive use of it. We have seen successful application of this strategy in Chapter 6 (covering arrays), 7 (covering perfect hash families) and 8 (partial covering arrays).

Another important theme underlying a number of strategies developed in this thesis is *search space reduction*, a technique that reduces the solution space by assuming some kind of symmetry. In Chapter 6, we have achieved search space reduction by considering only those covering arrays that are invariant under the action of different permutation groups on the symbols. In Chapter 7, we have achieved substantial search space reduction by considering covering perfect hash families. In terms of group action, this can be understood as the general linear group acting on the t-tuples of columns instead of individual symbols.

Overall, the combination of these two ideas—limited dependence and search space reduction—has proven to be quite powerful. Not only have we been able to provide a much shorter proof of the previous best result that was proved using a complicated entropy compression argument [23] but also we have proved an even stronger result. The power of our strategy can be appreciated from this result. In the context of covering perfect hash families, our method provides an upper bound on the $\mathsf{CAN}(t, k, v)$ that is not only the tightest known asymptotic upper bound in the general case but also outperforms many computationally derived upper bounds in the practical range of k values. Table 9.1 lists all the bounds and their asymptotically equivalent forms.

This thesis has made a number of major contributions in the area of practical construction algorithms for covering arrays as well. We have introduced the two-stage algorithmic framework for covering array construction. Individual algorithms from this framework surpassed a number of currently best known results. Also, this framework is flexible enough to be quite effective for mixed covering arrays, which are more important from a practical point of view. Our experimental evaluation shows that $\mathsf{TS}\langle\mathsf{Rand}, \mathsf{Greedy}; 2\rho\rangle$ and $\mathsf{TS}\langle\mathsf{Rand}, \mathsf{Den}; 2\rho\rangle$ with cyclic or Frobenius group action achieve the ideal trade-off in terms of the running time and the size of the constructed covering array. A key idea in this paradigm is an intelligent allocation of resources. This requires dividing the covering array construction process into number of stages and applying an algorithm in each stage that achieves the best trade-off between resource consumption (running time and memory) and quality of the solution.

Covering perfect hash family based Moser-Tardos type algorithms appear to be the most effective construction method for higher strength covering arrays. Although their applicability is limited to covering arrays with a constant level for all factors, they outperform all the known construction methods in this category. It has often been the case that the best asymptotic bounds do not translate to the best practical construction method for covering arrays. However, this algorithm has proven to be an important exception to this pattern. Not only do they provide the tightest known asymptotic bounds, but also in certain cases, they provide the best known construction methods as well.

9.2 Open Problems and Future Research Directions

We have not been able to provide a completely satisfactory answer to a few questions that are pertinent to this thesis. We believe these questions open up a number of exciting directions for future research. In this section, we briefly mention some these open issues and future research ideas.

On the theoretical side, establishing tighter bounds on the coloring based methods of Section 5.2.3 is a challenging problem. Further improvement may ensue from a more careful estimation of the chromatic number of the incompatibility graph. However, this task is complicated by the fact that the structure of the incompatibility graph is dependent on the random row selection and interaction coverage process of the first stage.

Coming up with a more efficient randomized algorithm for the first stage is an important problem from both theoretical as well as practical point of view. In Section 6.2.1 and 6.2.2 we have explored the possibility of Moser-Tardos type algorithm for the first stage. As mentioned already, those methods have severe practical limitations.

However, one may still be hopeful that a different approach of reducing the number of "bad" events to be avoided explicitly may lead to a better algorithm. A potentially fruitful approach may look as follows: Bad events would denote non-coverage of an interaction over t-sets of columns. We would select a set of column t-sets such that the dependency graph of the corresponding bad events has a bounded maximum

Bound name	Exact form $(\mathsf{CAN}(t,k,v))$	Asymptotic form $(d(t,v))$	Reference
Stein-Lovász-Johnson	$\left\lceil \frac{\log\left\{v^t\cdot\binom{k}{t}\right\}}{\log\left(\frac{v^t}{v^t-1}\right)} \right\rceil$	$\frac{t}{\log\left(\frac{v^t}{v^t-1}\right)}$	Theorem 1
Two-stage	$\left\lceil \frac{\log\left\{v^t\cdot\binom{k}{t}\cdot\log\left(\frac{v^t}{v^t-1}\right)\right\}+1}{\log\left(\frac{v^t}{v^t-1}\right)} \right\rceil$	$\frac{t}{\log\left(\frac{v^t}{v^t-1}\right)}$	Theorem 8
Two-stage + cyclic group action	$v\cdot\left\lceil \frac{\log\left\{v^{t-1}\cdot\binom{k}{t}\cdot\log\left(\frac{v^{t-1}}{v^{t-1}-1}\right)\right\}+1}{\log\left(\frac{v^{t-1}}{v^{t-1}-1}\right)} \right\rceil$	$\frac{vt}{\log\left(\frac{v^{t-1}}{v^{t-1}-1}\right)}$	Equation 3.1
Two-stage + Frobenius group action	$v(v-1)\cdot\left\lceil \frac{\log\left\{\left(\frac{v^{t-1}-1}{v-1}\right)\cdot\binom{k}{t}\cdot\log\left(\frac{v^{t-1}}{v^{t-1}-v+11}\right)\right\}+1}{\log\left(\frac{v^{t-1}}{v^{t-1}-v+1}\right)} \right\rceil+v$	$\frac{vt}{\log\left(\frac{v^{t-1}}{v^{t-1}-v+1}\right)}$	Equation 3.2
Godbole (LLL)	$\left\lceil \frac{\log\left\{v^t\cdot t\cdot\binom{k}{t-1}\right\}+1}{\log\left(\frac{v^t}{v^t-1}\right)} \right\rceil$	$\frac{t-1}{\log\left(\frac{v^t}{v^t-1}\right)}$	Theorem 2
LLL + cyclic group action	$v\cdot\left\lceil \frac{\log\left\{v^{t-1}\cdot t\cdot\binom{k}{t-1}\right\}+1}{\log\left(\frac{v^{t-1}}{v^{t-1}-1}\right)} \right\rceil$	$\frac{v(t-1)}{\log\left(\frac{v^{t-1}}{v^{t-1}-1}\right)}$	Theorem 10
LLL + Frobenius group action	$v(v-1)\cdot\left\lceil \frac{\log\left\{\left(\frac{v^{t-1}-1}{v-1}\right)\cdot t\cdot\binom{k}{t-1}\right\}+1}{\log\left(\frac{v^{t-1}}{v^{t-1}-v+1}\right)} \right\rceil+v$	$\frac{v(t-1)}{\log\left(\frac{v^{t-1}}{v^{t-1}-v+1}\right)}$	Theorem 11
Permutation vector	$(v^t-1)\cdot\left\lceil \frac{\log\left\{t\cdot\binom{k}{t-1}\right\}+1}{\log\left\{\frac{(v^t-1)^{t-1}}{(v^t-1)^{t-1}-\prod_{i=1}^{t-1}(v^t-v^i)}\right\}} \right\rceil+1$	$\frac{(v^t-1)(t-1)}{\log\left\{\frac{(v^t-1)^{t-1}}{(v^t-1)^{t-1}-\prod_{i=1}^{t-1}(v^t-v^i)}\right\}}$	Equation 7.2
Sherwood permutation vector	$(v^t-v)\cdot\left\lceil \frac{\log\left\{t\cdot\binom{k}{t-1}\right\}+1}{\log\left\{1/\prod_{i=1}^{t-1}(1-\frac{1}{v^i})\right\}} \right\rceil+v$	$\frac{(v^t-v)(t-1)}{\log\left\{1/\prod_{i=1}^{t-1}(1-\frac{1}{v^i})\right\}}$	Equation 7.3

Table 9.1: Different upper bounds for $CAN(t,k,v)$.

degree (less than the original dependency graph). We would devise a Moser-Tardos type algorithm for covering all the interactions on our chosen column t-sets, and then apply the conditional LLL distribution to obtain an upper bound on the number of uncovered interactions.

However, the difficulty lies in the fact that "all vertices have degree $\leq \rho$" is a non-trivial, "hereditary" property for induced subgraphs, and for such properties finding a maximum induced subgraph with the property is an NP-hard optimization problem [25]. There is still hope for a randomized or "nibble" type strategy that may find a reasonably good induced subgraph with a bounded maximum degree. Further exploration of this idea seems to be a promising research avenue.

In general, one need not restrain oneself within only two stages. One can definitely think of more than two stages in a covering array construction algorithm. However establishing the benefit or the lack thereof of having more than two stages is also an interesting open problem.

Probably the most underexplored topic in this thesis is the practical construction of covering arrays from covering perfect hash families using Moser-Tardos type methods. Although the basic method beats a number of currently best known results, there are enough reasons to believe that this method is not yet pushed to its limits. One obvious idea is to think about removing redundancy of coverage. This idea ties back to the technique of post-optimization. It is conceivable that once the covering array has been produced by the Moser-Tardos algorithm, we can remove a few rows of the array and still maintain full coverage. Otherwise, we may repeat Moser-Tardos type resampling on the smaller array itself. This and related ideas are ripe for further investigation, and we plan to pursue them in subsequent work.

REFERENCES

[1] Aldaco, A. N., C. J. Colbourn and V. R. Syrotiuk, "Locating arrays: A new experimental design for screening complex engineered systems", SIGOPS Oper. Syst. Rev. **49**, 1, 31–40, URL `http://doi.acm.org/10.1145/2723872.2723878` (2015).

[2] Alon, N. and J. H. Spencer, *The probabilistic method*, Wiley-Interscience Series in Discrete Mathematics and Optimization (John Wiley & Sons, Inc., Hoboken, NJ, 2008), third edn., URL `http://dx.doi.org.ezproxy1.lib.asu.edu/10.1002/9780470277331`, with an appendix on the life and work of Paul Erdős.

[3] Android, "Android configuration class", Http://developer.android.com/reference /android/content/res/Configuration.html (2016).

[4] Brown, J. I., "The complexity of generalized graph colorings", Discrete Appl. Math. **69**, 3, 257–270, URL `http://dx.doi.org.ezproxy1.lib.asu.edu/10.1016/0166-218X(96)00096-0` (1996).

[5] Bryce, R. C., Y. Chen and C. J. Colbourn, "Biased covering arrays for progressive ranking and composition of web services.", Int. J. Simulation Process Modelling **3**, 1/2, 80–87 (2007).

[6] Bryce, R. C. and C. J. Colbourn, "The density algorithm for pairwise interaction testing", Software Testing, Verification, and Reliability **17**, 159–182 (2007).

[7] Bryce, R. C. and C. J. Colbourn, "A density-based greedy algorithm for higher strength covering arrays", Software Testing, Verification, and Reliability **19**, 37–53 (2009).

[8] Bryce, R. C. and C. J. Colbourn, "Expected time to detection of interaction faults", Journal of Combinatorial Mathematics and Combinatorial Computing **86**, 87–110 (2013).

[9] Cawse, J. N., "Experimental design for combinatorial and high throughput materials development", GE Global Research Technical Report **29**, 769–781 (2002).

[10] Chandra, A. K., L. T. Kou, G. Markowsky and S. Zaks, "On sets of boolean n-vectors with all k-projections surjective", Acta Informatica **20**, 1, 103–111, URL `http://dx.doi.org/10.1007/BF00264296` (1983).

[11] Chateauneuf, M. A., C. J. Colbourn and D. L. Kreher, "Covering arrays of strength 3", Des. Codes Crypt. **16**, 235–242 (1999).

[12] Chen, B. and J. Zhang, "Tuple density: a new metric for combinatorial test suites", in "Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, Waikiki, Honolulu , HI, USA, May 21-28, 2011", pp. 876–879 (2011), URL `http://doi.acm.org/10.1145/1985793.1985931`.

[13] Cohen, D. M., S. R. Dalal, M. L. Fredman and G. C. Patton, "The AETG system: An approach to testing based on combinatorial design", IEEE Transactions on Software Engineering **23**, 437–44 (1997).

[14] Cohen, M. B., *Designing test suites for software interaction testing*, Ph.D. thesis, The University of Auckland (2004).

[15] Cohen, M. B., C. J. Colbourn and A. C. H. Ling, "Constructing strength three covering arrays with augmented annealing", Discrete Math. **308**, 2709–2722 (2008).

[16] Colbourn, C. J., "Combinatorial aspects of covering arrays", Le Matematiche (Catania) **58**, 121–167 (2004).

[17] Colbourn, C. J., "Covering array tables", Http://www.public.asu.edu/∼ccolbou/src/tabby (2005-2016).

[18] Colbourn, C. J., "Covering arrays and hash families", in "Information Security and Related Combinatorics", NATO Peace and Information Security, pp. 99–136 (IOS Press, 2011).

[19] Colbourn, C. J., "Conditional expectation algorithms for covering arrays", Journal of Combinatorial Mathematics and Combinatorial Computing **90**, 97–115 (2014).

[20] Colbourn, C. J., D. M. Cohen and R. C. Turban, "A deterministic density algorithm for pairwise interaction coverage", in "IASTED Proc. of the Intl. Conference on Software Engineering (SE 2004)", pp. 345–352 (2004).

[21] Damaschke, P., "Adaptive versus nonadaptive attribute-efficient learning", Machine Learning **41**, 2, 197–215, URL `http://dx.doi.org/10.1023/A:1007616604496` (2000).

[22] Diestel, R., *Graph Theory*, Graduate Texts in Mathematics (Springer, 2010), fourth edn.

[23] Francetić, N. and B. Stevens, "Asymptotic size of covering arrays: an application of entropy compression", ArXiv e-prints (2015).

[24] Galois, E., "Lettre de Galois à M. Auguste Chevalier", Journal de Mathématiques Pures et Appliquées **XI**, 408–415 (1846).

[25] Garey, M. R. and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W. H. Freeman & Co., New York, NY, USA, 1979).

[26] Gargano, L., J. Körner and U. Vaccaro, "Sperner capacities", Graphs and Combinatorics **9**, 31–46 (1993).

[27] Godbole, A. P., D. E. Skipper and R. A. Sunley, "*t*-covering arrays: upper bounds and Poisson approximations", Combinatorics, Probability and Computing **5**, 105–118 (1996).

[28] Graham, N., F. Harary, M. Livingston and Q. Stout, "Subcube fault-tolerance in hypercubes", Information and Computation **102**, 2, 280 – 314, URL `http://www.sciencedirect.com/science/article/pii/S0890540183710102` (1993).

[29] Gravier, S. and B. Ycart, "S-constrained random matrices", DMTCS Proceedings **0**, 1, URL `http://www.dmtcs.org/dmtcs-ojs/index.php/proceedings/article/view/dmAG0127` (2006).

[30] Haeupler, B., B. Saha and A. Srinivasan, "New constructive aspects of the Lovász local lemma", J. ACM **58**, 6, Art. 28, 28, URL `http://dx.doi.org.ezproxy1.lib.asu.edu/10.1145/2049697.2049702` (2011).

[31] Hartman, A., "Software and hardware testing using combinatorial covering suites", in "Interdisciplinary Applications of Graph Theory, Combinatorics, and Algorithms", edited by M. C. Golumbic and I. B.-A. Hartman, pp. 237–266 (Springer, Norwell, MA, 2005).

[32] Hartman, A. and L. Raskin, "Problems and algorithms for covering arrays", Discrete Mathematics **284**, 13, 149 – 156, URL `http://www.sciencedirect.com/science/article/pii/S0012365X0400130X` (2004).

[33] Hedayat, A. S., N. J. A. Sloane and J. Stufken, *Orthogonal Arrays* (Springer-Verlag, New York, 1999).

[34] Johnson, D. S., "Approximation algorithms for combinatorial problems", J. Comput. System Sci. **9**, 256–278 (1974).

[35] Jukna, S., *Extremal Combinatorics: With Applications in Computer Science* (Springer Publishing Company, Incorporated, 2010), 1st edn.

[36] Katona, G. O. H., "Two applications (for search theory and truth functions) of Sperner type theorems", Periodica Math. **3**, 19–26 (1973).

[37] Kleitman, D. and J. Spencer, "Families of k-independent sets", Discrete Math. **6**, 255–262 (1973).

[38] Kuhn, D. R., R. Kacker and Y. Lei, *Introduction to Combinatorial Testing* (CRC Press, 2013).

[39] Kuhn, D. R., I. D. Mendoza, R. N. Kacker and Y. Lei, "Combinatorial coverage measurement concepts and applications", in "2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)", pp. 352–361 (2013).

[40] Kuhn, D. R. and M. Reilly, "An investigation of the applicability of design of experiments to software testing", in "Proc. 27th Annual NASA Goddard/IEEE Software Engineering Workshop", pp. 91–95 (IEEE, Los Alamitos, CA, 2002).

[41] Kuhn, D. R., D. R. Wallace and A. M. Gallo, "Software fault interactions and implications for software testing", IEEE Trans. Software Engineering **30**, 418–421 (2004).

[42] Lovász, L., "On the ratio of optimal integral and fractional covers", Discrete Math. **13**, 4, 383–390 (1975).

[43] Maximoff, J. R., M. D. Trela, D. R. Kuhn and R. Kacker, "A method for analyzing system state-space coverage within a t-wise testing framework", in "4th Annual IEEE Systems Conference", pp. 598–603 (2010).

[44] Meagher, K. and B. Stevens, "Group construction of covering arrays", J. Combin. Des. **13**, 70–77 (2005).

[45] Moser, R. A. and G. Tardos, "A constructive proof of the general Lovász local lemma", J. ACM **57**, 2, Art. 11, 15, URL `http://dx.doi.org.ezproxy1.lib.asu.edu/10.1145/1667053.1667060` (2010).

[46] Nie, C. and H. Leung, "A survey of combinatorial testing", ACM Computing Surveys **43**, 2, #11 (2011).

[47] Sarkar, K. and C. J. Colbourn, "Two-stage algorithms for covering array construction", (submitted for publication) ArXiv e-prints, arXiv:1606.06730 URL `http://arxiv.org/abs/1606.06730` (2016).

[48] Sarkar, K. and C. J. Colbourn, "Upper bounds on the size of covering arrays", (submitted for publication) ArXiv e-prints, arXiv:1603.07809 URL `http://arxiv.org/abs/1603.07809` (2016).

[49] Sarkar, K., C. J. Colbourn, A. de Bonis and U. Vaccaro, "Partial covering arrays: Algorithms and asymptotics", in "Combinatorial Algorithms: 27th International Workshop, IWOCA 2016, Helsinki, Finland, August 17-19, 2016", edited by V. Mäkinen, J. S. Puglisi and L. Salmela, vol. 9843 of *Lecture Notes in Computer Science*, pp. 437–448 (Springer International Publishing, Switzerland, 2016), URL `http://dx.doi.org/10.1007/978-3-319-44543-4_34`.

[50] Shasha, D. E., A. Y. Kouranov, L. V. Lejay, M. F. Chou and G. M. Coruzzi, "Using combinatorial design to study regulation by multiple input signals: A tool for parsimony in the post-genomics era", Plant Physiol. **127**, 1590–2594 (2001).

[51] Sherwood, G. B., S. S. Martirosyan and C. J. Colbourn, "Covering arrays of higher strength from permutation vectors", Journal of Combinatorial Designs **14**, 3, 202–213, URL `http://dx.doi.org/10.1002/jcd.20067` (2006).

[52] Stein, S. K., "Two combinatorial covering theorems", J. Combinatorial Theory Ser. A **16**, 391–397 (1974).

[53] Tong, A. J., Y. G. Wu and L. D. Li, "Room-temperature phosphorimetry studies of some addictive drugs following dansyl chloride labelling", Talanta **43**, 9, 14291436, URL `http://europepmc.org/abstract/MED/18966621` (1996).

[54] Torres-Jimenez, J., H. Avila-George and I. Izquierdo-Marquez, "A two-stage algorithm for combinatorial testing", Optimization Letters pp. 1–13, URL `http://dx.doi.org/10.1007/s11590-016-1012-x` (2016).

APPENDIX A

NOTATION

$[k]$ **:** The set $\{1, 2, \ldots, k\}$ for a positive integer k (page 11).

$\mathsf{CA}(N; t, k, v)$ **:** A covering array with N rows, k factors, v levels and strength t (page 11).

$\mathsf{CAN}(t, k, v)$ **:** The minimum N for which a $\mathsf{CA}(N; t, k, v)$ exists. (page 11).

$\mathcal{C}_{t,k}$ **:** The set of all t-sets of $[k]$ (page 12).

$\mathcal{I}_{t,k,v}$ **:** The set of all possible t-way interactions among k factors, each having v levels (page 12).

$\mathsf{MCA}(N; t, k, (v_1, v_2, \ldots, v_k))$ **:** A mixed covering array with N rows, k columns having $v_1, \ldots, v_k$ levels and strength t (page 12).

$d(t, v)$ **:** The limiting value of the least upper bound of the ratio of $\mathsf{CAN}(t, k, v)$ and $\log k$ as $k \to \infty$ (page 14).

$\mathsf{TS}\langle A, B; r, \Gamma\rangle$ **:** A two-stage algorithm where A is the method used in the first stage, B is the methods used in the second stage, r is the maximum number of interactions that remain uncovered at the end of the first stage, and Γ is the group that acts on the rows (page 68).

$\mathsf{CPHF}(n; t, k, v)$ **:** A covering perfect hash family with n rows and k columns with parameters t and v (page 97).

$\mathsf{SCPHF}(n; t, k, v)$ **:** A Sherwood covering perfect hash family with n rows and k columns with parameters t and v (page 97).

$\mathsf{PCA}(N; t, k, v, m)$ **:** A partial covering array with N rows, k columns and parameters t, v and m (page 111).

$\mathsf{APCA}(N; t, k, v, m, \epsilon)$ **:** A ϵ-almost covering array with N rows, k columns and parameters t, v, m and ϵ (page 111).

APPENDIX B

THE BIASED DENSITY ALGORITHM: NAÏVE DERANDOMIZATION

Algorithm 15: The biased density algorithm for selection of a row.

```
Input: $X$ : the set of uncovered interactions, $\Sigma$: the alphabet of symbols with
       $|\Sigma| = v$, $t$ : strength of the covering array, $k$ : number of factors.
Output: $r$ : the next row of the covering array.
1  Let $r := (*, \ldots, *)$;
2  Let $\rho$ denote the set of entries in $r$ that have been fixed. Initially, $\rho = \phi$;
3  while $X \neq \phi$ do
4      Let $maxExpCov := 0$, and $\iota^* := (null, null)$;
5      foreach $\iota = (c, s) \in X$ do
6          Let $expCov$ store the conditional expected coverage computed by
             Algorithm 16;
7          if $expCov > maxExpCov$ then
8              Set $maxExpCov := expCov$, and $\iota^* := \iota$, i.e., set $c^* = c$ and $s^* = s$;
9          end
10     end
11     for $i = 1$ to $t$ do
12         Set $r_{c_i^*} := s_i^*$;
13     end
14     foreach $\iota' = (c', s') \in X$ do
15         if $\iota'$ does not agree with $r$ or is already covered by $r$ then
16             Delete $\iota'$ from $X$;
17         end
18     end
19 end
20 Fix the remaining entries of $r$, if there are any, randomly from $\Sigma$;
```

Algorithm 16: Computation of conditional expected coverage.

```
Input: $X$ : the set of uncovered interactions, $\iota = (c, s)$: the interaction that is
       selected next, $\rho$ : the set of entries that have been fixed in the current
       row, $\Sigma$: the alphabet of symbols with $|\Sigma| = v$, $t$ : strength of the
       covering array, $k$ : number of factors.
Output: The conditional expected coverage if the interaction $\iota$ is selected
        next.
1 Let $count := 0$;
  // The following loop runs for $|\Sigma|^{k-|\rho \cup c|}$ times; therefore, takes
     exponential time
2 foreach $r' \in \Sigma^k$ do
3     if $r'$ agrees with the symbols selected for the entries in $\rho$ and $\iota$ then
4         Set $count := count + m(r')$;
5     end
6 end
  // The probability that $r'$ is chosen is $1/|\Sigma|^{k-|\rho \cup c|}$.
7 Return $count/|\Sigma|^{k-|\rho \cup c|}$;
```

APPENDIX C

COMPARISON OF THE DENSITY AND THE BIASED DENSITY ALGORITHM

k	Uniform Random	Biased Random	Density	Biased Density
		$t = 2, v = 3$		
4	15	12	11	9
5	18	15	14	13
6	21	16	15	15
7	25	18	16	15
8	26	21	17	16
9	28	22	18	17
10	31	23	17	17
		$t = 3, v = 3$		
4	62	35	32	34
5	85	48	42	40
6	104	60	50	46
7	119	72	56	51
8	125	81	58	57
9	139	91	62	61
10	149	98	64	62
14	-	-	80	73

Table C.1: Comparison of different covering array construction algorithms—the uniform random (Algorithm 2), the biased random (Algorithm 6), the density (Algorithm 4), and the biased density (Algorithm 7) row selection algorithms. $t = 2$, $v = 3$ and $t = 3$, $v = 3$. $10,000$ independent runs of the randomized algorithm are used, and the best result is reported.