Variable Strength Covering Arrays

Sebastian Raaphorst

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Ph.D. degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

© Sebastian Raaphorst, Ottawa, Canada, 2013
Abstract

Recently, covering arrays have been the subject of considerable research attention as they hold both theoretical interest and practical importance due to their applications to testing. In this thesis, we perform the first comprehensive study of a generalization of covering arrays called variable strength covering arrays, where we dictate the interactions to be covered in the array by modeling them as facets of an abstract simplicial complex.

We outline the necessary background in the theory of hypergraphs, combinatorial testing, and design theory that is relevant to the study of variable strength covering arrays. We then approach questions that arise in variable strength covering arrays in a number of ways. We demonstrate their connections to hypergraph homomorphisms, and explore the properties of a particular family of abstract simplicial complexes, the qualitative independence hypergraphs. These hypergraphs are tightly linked to variable strength covering arrays, and we determine and identify several of their important properties and subhypergraphs.

We give a detailed study of constructions for variable strength covering arrays, and provide several operations and divide-and-conquer techniques that can be used in building them. In addition, we give a construction using linear feedback shift registers from primitive polynomials of degree 3 over arbitrary finite fields to find variable strength covering arrays, which we extend to strength-3 covering arrays whose sizes are smaller than many of the best known sizes of covering arrays.

We then give an algorithm for creating variable strength covering arrays over arbitrary abstract simplicial complexes, which builds the arrays one row at a time, using a density concept to guarantee that the size of the resultant array is asymptotic in the logarithm of the number of facets in the abstract simplicial complex. This algorithm is of immediate practical importance, as it can be used to create test suites for combinatorial testing.

Finally, we use the Lovász Local Lemma to nonconstructively determine upper bounds on the sizes of arrays for a number of different families of hypergraphs. We lay out a framework that can be used for many hypergraphs, and then discuss possible strategies that can be taken in asymmetric problems.
Acknowledgements

This thesis certainly could not have been completed without the help of many people.

Firstly and most importantly, I would like to extend my immeasurable thanks to both of my supervisors, Lucia Moura and Brett Stevens. As a lowly undergraduate student many years ago, I had the extreme fortune of being chosen to serve as a teaching assistant to work with Lucia Moura. I could never have predicted that that was to be the first step on a long and rewarding journey into academia. Through all the opportunities she gave me, I caught her infectious enthusiasm for research and had my eyes opened to the wonderful fields of design theory and covering arrays. Without her endless patience and compassion with my quirkiness, this thesis would never have come to see the light of day. Brett has been my biggest inspiration: his love of mathematics has helped me to realize the beauty in combinatorial design theory through its far-reaching connections to many other disciplines. He is a seemingly endless source of ideas, and his incredible knowledge has been an immeasurable help. Working with him has been an amazing experience with which I have been blessed.

Many thanks to the other members of my doctoral committee, Sylvia Boyd and Yvan Labiche, for their time, effort, and recommendations. Sylvia’s wonderful gift for teaching has always served as an example to me, and has made me both a better student and teacher. I would also like to thank Daniel Panario for his generosity with both his time and knowledge, and in particular, for teaching me about finite fields. Similarly, thanks to Charles Colbourn for his help and suggestions. Much of what I have learned has been through his publications.

I would also like to thank the other individuals who served as examiners for my thesis defence: Jeff Dinitz, who graciously came from Vermont to attend despite the treacherous weather; Amiya Nayak; and Mateja Sajna. I must extend special thanks to Mateja for her thorough attention to detail and excellent suggestions. They have improved the quality of the writing and mathematics of this thesis dramatically.

Karen Meagher has always been a competitor that drove me forward, a colleague that encouraged me, and a source of distraction and amusement. Our university laboratory went from being a place of mirth to a windowless pit of gloom in her absence. Her thesis was fundamental to this work and motivated many of the ideas herein. Jacob Chodoriwsky has been a great friend with whom I shared many good times playing Dance Dance Revolution and board games, and who was always willing to discuss research.

感謝我的中文老師。每個星期，我最盼望的就是上中文課了。王老師一直支持我。即使當我擔心不能畢業的時候，我們的談話也使我樂觀起來。我相信是我的中文功課
My friends Carlie Robbins and Mario Ossia are to thank for satisfying my social needs during the production of this thesis. Fortunately for me, they were endlessly understanding of my frantic work schedule and as a result, generously would come to me or organize activities that were both easy and fun for me to enjoy. Their thoughtfulness and flexibility allowed me to direct my time and energy toward my research, my hobbies, and satisfying the pesky general requirements of being a human, which would sometimes be forgotten when I became overly focused. A heartfelt thanks to my dear friend Norma Hannant, without who my life would have much less laughter. Other people whose friendship I appreciate greatly include Alison Steeves, Kat Angus, and Charles Catalano.

I’d also like to thank my parents, Peter and Ginette, for too many reasons to fully enumerate here. They have constantly encouraged me and supported me, both emotionally and financially. Their love for and faith in me was truly uplifting and continually urged me forward and closer to my goals. I am indeed privileged to be their son and could not have hoped for better role models.

Finally, I thank my husband Jeff / JD for his patience and motivation. I’m sure he was on the verge of consigning himself to the possibility that I might remain a student for all of eternity. He was a much-needed distraction who made sure I ate, slept, and watched enough television to prevent my brain from irreparably melting under the vast weight of large amounts of mathematics. Our four wonderful cats - Duncan, Rapunzel, Zoe, and Max - single-pawedly prevented me from developing carpal tunnel syndrome through their loving and endearing demands that I frequently stop typing and instead engage myself in something useful and productive like petting and napping, going so far as to lay down on my keyboard if I refused to heed their sage wisdom. Every day with JD and our furry family is a blessing fraught with laughter and adventures, and I look forward to many more to come.

只要功夫深，鐵杵磨成針
Sebastian Raaphorst (饒漢文)
# Contents

## 1 Introduction
1.1 Combinatorial Testing Examples .......................... 4
1.2 Thesis Contributions ........................................ 8
   1.2.1 Summary of Contents .................................. 8
   1.2.2 Major Contributions ................................... 11

## 2 Covering Arrays and Design Theory .......................... 14
2.1 Graphs, Hypergraphs, and ASC ............................... 15
   2.1.1 General Relational Systems ............................ 25
2.2 Covering Arrays and their Generalizations .................. 27
   2.2.1 Bounds on CAN ......................................... 33
   2.2.2 Direct and Algebraic Constructions .................... 36
   2.2.3 Recursive Constructions ................................ 38
   2.2.4 Covering Arrays on Graphs ............................. 42
   2.2.5 Variable Strength Covering Arrays ..................... 43
   2.2.6 Covering Arrays with Forbidden Edges ................. 44
   2.2.7 Consecutive Covering Arrays ........................... 45
   2.2.8 \(k\)-Trees ........................................... 47
   2.2.9 Triangulations of the Sphere .......................... 48
2.3 Design Theory and Links to Covering Arrays ............... 52
2.4 Testing and Covering Arrays ................................ 56
   2.4.1 Combinatorial Testing ................................. 57
   2.4.2 Other Applications of Covering Arrays to Testing .... 62

## 3 Variable Strength Covering Arrays and Qualitative Independence Hypergraphs .......................... 64
3.1 Homomorphisms and Colourings ............................. 65
6.5 Analysis of Results ................................................. 202

7 Conclusion and Open Questions ........................................205
  7.1 Major Contributions and Summary ..................................205
    7.1.1 Theory of VCA and Constructions .............................. 205
    7.1.2 Greedy Algorithm for VCA ................................... 208
    7.1.3 Probabilistic Methods for VCA Bounds ..........................209
  7.2 Open Questions ...................................................... 210
## List of Tables

1.1 The polynomial library VCA .................................................. 5  
1.2 Circuit VCA ........................................................................ 7  
1.3 3-tier architecture VCA ....................................................... 9  

4.1 Sizes of CA($N; 3, k, g$) using the LFSR construction ................. 114  
4.2 Sizes of CA($N; 3, k, g$) with fusion and increasing .................. 115  
4.3 Explanation of Table 4.8 ....................................................... 116  
4.4 Random permutation groups in LFSR construction for CA($N; 4, 40, 3$) ................................................................. 121  
4.5 Generator refinement in LFSR construction for CA($N; 4, 40, 3$) ................................................................. 123  
4.6 $s(t, q)$ such that there exist CA($s(t, q)(q^t - 1) + 1; t, \frac{q^t - 1}{q-1}, q$) ................................................................. 123  
4.7 Improvements from Conjecture 4.4.3 ........................................ 125  
4.8 CAN($3, k, g$) bound improvements ....................................... 126  

5.1 Comparison of VARDENS and Bryce-Colbourn density algorithm .................................................................................. 152  
5.2 Effect of varying ℓ in VARDENS .............................................. 153  
5.3 Comparison of VARDENS and simulated annealing .................. 159  
5.4 VARDENS for STS(15) ........................................................... 162  
5.5 VARDENS for STS($k$), $k \in \{7, 9, 13, 15\}$ .......................... 163  

6.1 Local Lemma vs VARDENS bounds on STS ......................... 184  
6.2 Local Lemma vs VARDENS bounds on SQS ........................... 185  
6.3 VARDENS results for triangulations ..................................... 193  
6.4 Bad event dependencies for asymmetric ASC .......................... 197  
6.5 Comparison of symmetric versus general Local Lemma ............ 200  
6.6 Symbolic and numerical optimization of general Local Lemma .................................................................................. 201  
6.7 Binary searching with the asymmetric Local Lemma constraints ................................................................. 202
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Circuit example</td>
<td>6</td>
</tr>
<tr>
<td>1.2</td>
<td>Circuit hypergraph</td>
<td>6</td>
</tr>
<tr>
<td>1.3</td>
<td>3-tier architecture example</td>
<td>7</td>
</tr>
<tr>
<td>1.4</td>
<td>3-tier architecture hypergraph</td>
<td>8</td>
</tr>
<tr>
<td>2.1</td>
<td>Example of a 2-section of a hypergraph</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>Example of a CA($10; 3, 5, 2$) and a VOA($8, \Lambda, 2$), with $\Lambda$ a $(7, 3, 4)$-BIBD</td>
<td>29</td>
</tr>
<tr>
<td>2.3</td>
<td>Example of a VCA</td>
<td>32</td>
</tr>
<tr>
<td>2.4</td>
<td>2-tree examples</td>
<td>47</td>
</tr>
<tr>
<td>2.5</td>
<td>Recursive generation of triangulations of the sphere</td>
<td>50</td>
</tr>
<tr>
<td>2.6</td>
<td>Example of a triangulation of the sphere</td>
<td>51</td>
</tr>
<tr>
<td>3.1</td>
<td>VCA and hypergraph homomorphisms</td>
<td>66</td>
</tr>
<tr>
<td>3.2</td>
<td>Vertex transitive hypergraphs have vertex transitive cores</td>
<td>76</td>
</tr>
<tr>
<td>3.3</td>
<td>Split: $\text{VCAN}(H, (ab, g_1, \ldots, g_{k-1})) = \text{VCAN}(G, (a, b, g_1, \ldots, g_{k-1}))$</td>
<td>82</td>
</tr>
<tr>
<td>3.4</td>
<td>Join: $\text{VCAN}(H, (a, b, g_2, \ldots, g_{k-1})) \leq \text{VCAN}(G, (ab, g_2, \ldots, g_{k-1}))$</td>
<td>85</td>
</tr>
<tr>
<td>3.5</td>
<td>Splitting and joining example for two factors with 5 and 6 levels</td>
<td>88</td>
</tr>
<tr>
<td>3.6</td>
<td>Turning a hypergraph into a hypertree via join operations</td>
<td>90</td>
</tr>
<tr>
<td>3.7</td>
<td>The cut-vertex construction for uniform / almost-uniform VCA</td>
<td>92</td>
</tr>
<tr>
<td>3.8</td>
<td>A $d$-colouring of $H_c^{k,t}$, for $d = t + \left\lfloor \frac{r}{q} \right\rfloor$, $k = qt + r$</td>
<td>96</td>
</tr>
<tr>
<td>5.1</td>
<td>VARDENS example</td>
<td>144</td>
</tr>
<tr>
<td>5.2</td>
<td>Effect of LF variation of VARDENS</td>
<td>154</td>
</tr>
<tr>
<td>5.3</td>
<td>Effect of MU variation of VARDENS</td>
<td>155</td>
</tr>
<tr>
<td>5.4</td>
<td>Effect of 2R variation of VARDENS</td>
<td>156</td>
</tr>
<tr>
<td>5.5</td>
<td>Effect of 2S variation of VARDENS</td>
<td>157</td>
</tr>
<tr>
<td>5.6</td>
<td>VARDENS variation time requirements</td>
<td>158</td>
</tr>
<tr>
<td>5.7</td>
<td>Three hypertrees used in VARDENS experiments</td>
<td>163</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.8</td>
<td>Time and array sizes of \textit{VARDENS} on hypertree $T_1$ arrays, $g = 3$</td>
<td>164</td>
</tr>
<tr>
<td>5.9</td>
<td>Time and array sizes of \textit{VARDENS} on hypertree $T_1$ arrays, $g = 5$</td>
<td>165</td>
</tr>
<tr>
<td>5.10</td>
<td>Time and array sizes of \textit{VARDENS} on hypertree $T_2$ arrays, $g = 3$</td>
<td>166</td>
</tr>
<tr>
<td>5.11</td>
<td>Time and array sizes of \textit{VARDENS} on hypertree $T_2$ arrays, $g = 5$</td>
<td>167</td>
</tr>
<tr>
<td>5.12</td>
<td>Time and array sizes of \textit{VARDENS} on hypertree $T_3$ arrays</td>
<td>168</td>
</tr>
<tr>
<td>6.1</td>
<td>Design with varying facet dependencies</td>
<td>186</td>
</tr>
<tr>
<td>6.2</td>
<td>\textit{VARDENS} for $c\text{CCAN}(3, k, 3)$</td>
<td>192</td>
</tr>
<tr>
<td>6.3</td>
<td>Comparison of \textit{VARDENS} and the general Local Lemma</td>
<td>204</td>
</tr>
</tbody>
</table>
Symbols

\[ EQ(H), \text{15} \]
\[ H^*, \text{21} \]
\[ H^{k,t}, \text{44} \]
\[ H_c^{k,t}, \text{44} \]
\[ K_n, \text{14} \]
\[ K_n^t, \text{18} \]
\[ T(v, n), \text{46} \]
\[ V(\Delta), \text{20} \]
\[ [H]_n, \text{15} \]
\[ \text{Aut}(H), \text{17} \]
\[ \text{CAN}_\lambda(t, k, g), \text{27} \]
\[ \text{CA}_\lambda(N; t, k, g), \text{27} \]
\[ \text{CCA}(N; t, k, g), \text{44} \]
\[ \text{CCAN}(t, k, g), \text{44} \]
\[ \text{COD}(N; t, k, g), \text{39} \]
\[ \text{CODN}(t, k, g), \text{39} \]
\[ \text{DCA}(N; 2, k, g), \text{38} \]
\[ \Delta, \text{20} \]
\[ \Delta(H), \text{20} \]
\[ \Delta_i, \text{20} \]
\[ \text{LS}(n), \text{35} \]
\[ \Lambda, \text{20} \]
\[ \text{MCAN}_\lambda(t, (g_0, \ldots, g_{k-1})), \text{28} \]
\[ \text{MCA}_\lambda(N; t, (g_0, \ldots, g_{k-1})), \text{28} \]
\[ \text{MOLS}(n), \text{35} \]
\[ \text{OA}_\lambda(t, k, g), \text{26} \]
\[ \text{OOA}_\lambda(t, k, l, g), \text{42} \]
\[ \text{RCODN}(t, k, g), \text{40} \]
\[ \text{UVQIH}(n, g), \text{77} \]
\[ \text{VCAN}_\lambda(\Lambda, g), \text{29} \]
\[ \text{VCA}_\lambda(N; \Lambda, g), \text{29} \]
\[ \text{VOA}(N; \Lambda, g), \text{29} \]
\[ \alpha_t(H), \text{19} \]
\[ \text{cCCA}(N; t, k, g), \text{45} \]
\[ \text{cCCAN}(t, k, g), \text{45} \]
\[ \chi(G), \text{14} \]
\[ \chi(H), \text{23} \]
\[ \text{diam}(H), \text{17} \]
\[ \gamma, \text{23} \]
\[ \hookrightarrow, \text{16} \]
\[ \omega(G), \text{19} \]
\[ \omega_t(H), \text{18} \]
\[ \text{rank}(H), \text{15} \]
\[ \rightarrow^W, \text{79} \]
\[ \text{skel}_\Delta(\Delta), \text{21} \]
\[ \rightarrow, \text{16} \]
\[ \hat{H}, \text{17} \]
\[ d_H(v), \text{16} \]
\[ d'_H(v), \text{16} \]
\[ p_H(u, v), \text{17} \]
\[ \text{ASC}, \text{20} \]
Chapter 1

Introduction

The study of covering arrays is an active area of research, rich in both applications and theory. Their origins are in orthogonal arrays, which have been used in the generation of statistical experiments. Covering arrays are of particular interest in fields such as software testing, especially in the field of combinatorial testing, as test cases from orthogonal arrays can make guarantees as to the quality of testing.

An orthogonal array is an $n \times k$ array with entries from a set of size $v$ such that, for a given $t$ called the strength of the array, the $n \times t$ subarray over any set of $t$ columns contains as rows all of the $v^t$ tuples precisely $\lambda = n/v^t$ times. We call $n$ the size of the array. These strict balance requirements generally do not translate to actual requirements or offer significant benefit when they are used in the design of test suites. Additionally, orthogonal arrays exist only for specific combinations of parameters $\lambda$, $t$, $k$, and $v$.

In order to broaden their use to a larger range of problems, the balance requirement of orthogonal arrays was relaxed, and the resultant structure named a covering array; instead of requiring every $v^t$ tuple to appear precisely $n/v^t$ times, we simply demand that every $v^t$ tuple appears a minimum of $\lambda$ times for some given $\lambda$, although typically we are only interested in cases where $\lambda = 1$. In the case of orthogonal arrays, the size of the array is fully determined by the parameters $t$, $v$, and $\lambda$, whereas in covering arrays, there is no explicit constraint on the size of the array. This gives rise to the two most commonly examined problems in the study of covering arrays:

1. Given $\lambda$, $t$, $k$, and $v$, what is the smallest $n$ such that a covering array exists?

2. Given $\lambda$, $n$, $t$, and $v$, what is the largest $k$ such that a covering array exists?

Covering arrays are generally computationally difficult to find, although certain cases
are well understood or solved completely; for example, determining the smallest size of array \( n \) required for \( v = t = 2 \) was solved independently by Katona [53] and also Kleitman and Spencer [54] using the Sperner and Erdős-Ko-Rado theorems. Much of the research on covering arrays involves developing constructions to create them directly [17, 53, 54], recursively [15, 50, 80], or via hybrid techniques [23, 24]; creating heuristic algorithms to find covering arrays [10, 11, 18, 19, 20, 21, 22, 23, 24, 32, 41, 58, 74]; and finding bounds on their sizes [39, 40].

When using a covering array in testing, the rows of the array represent the test cases, while the columns of the array represent the parameters of the problem, or factors. The values in the array over a row represent the values, or levels, that a factor should assume in the specific test case. Details on using covering arrays for testing can be found in [19, 21, 36, 37, 88, 89], and studies supporting their effectiveness in testing can be found in [12, 13, 19, 21, 36, 37, 42, 57, 90].

Many generalizations of covering arrays have risen to extend their uses and enhance their applicability to testing. We detail some of these below:

- **Mixed-level covering arrays:** Regular covering arrays work under the assumption that all factors have the same range of levels. When used in testing, however, it is very unlikely that all configurable parameters will have the same number of levels. Mixed-level covering arrays [34, 71] overcome this limitation by allowing the number of levels for each factor to be specified, which manifests in the actual array as each column having its own range of values while still maintaining the balance conditions.

- **Covering arrays with forbidden edges:** Certain combinations of levels may be prohibited, and thus they can never appear together in a row. For \( k \) factors, this can be modeled using a \( k \)-partite graph. Some results can be found in [38, 61, 62].

- **Covering arrays on graphs:** When performing pairwise testing (\( t = 2 \)), it may be the case that certain factors are known to not interact. We can then model the factors as the vertices of a graph, with edges indicating which factors interact; for an edge in the graph, the subarray of the covering array over the two columns corresponding to the factors in the edge must have every possible 2-tuple appearing as rows. Covering arrays on graphs have been studied in [66, 67, 68].

- **Variable strength covering arrays:** We may know that certain sets of factors interact, interact strongly, or do not interact. We can model the interactions
amongst factors that we wish to cover using a hypergraph where the vertices are the factors and the edges represent the sets of factors to test. This allows us to direct and focus testing, and is a generalization of covering arrays on graphs.

In this thesis, we are predominantly interested in variable strength covering arrays, both fixed-level and mixed-level. Very little has been done in this field. We present a review of the current literature in Section 2.2.5.

1.1 Combinatorial Testing Examples

We give three examples of small-scale testing problems to motivate the use of variable strength covering arrays.

Example 1.1.1. Consider an external library that implements basic operations for polynomials over a finite field $\mathbb{F}_q$. We are writing a program that will use this library, limited to the case $q = 2$, and we would like to test it beforehand. In particular, assume we have two polynomials $f = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ and $g = b_1x + b_0$, and we are interested in testing the following operations:

- Calculate the product of the two polynomials $fg$:
  \[
  fg = (a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0)(b_1x + b_0) \\
  = a_4b_1x^5 + (a_4b_0 + a_3b_1)x^4 + (a_3b_0 + a_2b_1)x^3 + (a_2b_0 + a_1b_1)x^2 \\
  + (a_1b_0 + a_0b_1)x + a_0b_0
  \]
- Calculate the derivative of $f$:
  \[
  f' = 4a_4x^3 + 3a_3x^2 + 2a_2x + a_1 \\
  = a_3x^2 + a_1
  \]

We can then consider, as inputs to our program, the coefficients $a_0, a_1, a_2, a_3, a_4, b_0, b_1$, and as outputs, the coefficients of the polynomials that result from our two operations. The calculations above then give the set of inputs that determine each of the outputs. In order to model this dependency, we create a hypergraph with one vertex per input, and take, for each output, one hyperedge over the vertices that determine the output value. Thus, our hypergraph has vertices:

\[
V = \{a_0, a_1, a_2, a_3, a_4, b_0, b_1\}
\]
and the following hyperedges:

\[ E = \left\{ \{a_4, b_1\}, \{a_4, b_0, a_3, b_1\}, \{a_3, b_0, a_2, b_1\}, \{a_2, b_0, a_1, b_1\}, \{a_1, b_0, a_0, b_1\}, \{a_0, b_0\}, \{a_3, a_1\} \right\}. \]

Instead of testing all possible combinations of polynomials, of which there are \(2^7 = 128\), in order to make sure every combination of values for each of the outputs is tested in some test case, we only need 16 combinations of polynomials, as dictated by the array in Table 1.1.

<table>
<thead>
<tr>
<th>T#</th>
<th>(a_0)</th>
<th>(a_1)</th>
<th>(a_2)</th>
<th>(a_3)</th>
<th>(a_4)</th>
<th>(b_0)</th>
<th>(b_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1.1: A variable strength covering array for testing the finite field polynomial library for \(\mathbb{F}_2\).

**Example 1.1.2.** For our second example, we look at the circuit diagram in Figure 1.1.

This is a small circuit, with only five inputs - \(a\), \(b\), \(c\), \(d\), and \(e\) - each of one bit, and three outputs. This circuit could be exhaustively tested using \(2^5 = 32\) tests, but based on the fact that some of the inputs do not interact, doing so is unnecessary. We instead show that we can fully test the circuit using only eight tests by using a variable strength
covering array. We represent the circuit as a hypergraph, with one vertex for each input, and one hyperedge for each output, as shown in Figure 1.2.

![Figure 1.1: An example of a small circuit with five inputs and three outputs.](image)

In Table 1.2, we give a variable strength covering array for the circuit that requires only eight different test cases, with the guarantee that for each output, every possible combination of inputs have been tested together in some test case.

**Example 1.1.3.** A third example demonstrates the use of variable strength covering arrays to systems testing. Assume that we have a simple 3-tier architecture system as shown in Figure 1.3 where a user of the system works through some kind of interface to connect to a server to interact with a database. Furthermore, on the server, one of several possible reporting packages prepares reports on the data for the user, which the user may then print via a printer available through the interface. This system has a total
Table 1.2: A variable strength covering array for the circuit in Figure 1.1 with hypergraph in Figure 1.2. There are eight test cases.

of $3^5 = 243$ possible configurations to test based on the possible choice of three different elements for each component, and having 243 different such test configurations is likely to be too costly. For more complex architectures with more choices for each component, the total number of test configurations becomes very large. Instead of testing all possible configurations, we identify the components that directly interact with each other and model these as a hypergraph, as given in Figure 1.4.

![Figure 1.3: A 3-tier system with the possible choices for each component illustrated, as well as the communication between components.](image)

In Table 1.3 we provide a variable strength covering array for the system that requires only 27 different configurations, with the guarantee that all possible configurations over each of the hyperedges in the hypergraph are tested together in some test configuration.
1.2 Thesis Contributions

We present an outline of the thesis, and then provide a list of the accomplishments and contributions to research that we have generated in this work.

1.2.1 Summary of Contents

We focus on variable strength covering arrays (VCA), and derive several results about other types of arrays in the process.

In Chapter 2 we outline the necessary mathematical background and definitions. In Section 2.1 we begin with an introduction to hypergraphs and one particular type of hypergraph that will be of particular interest to us, called an abstract simplicial complex (ASC), which is a hypergraph where the edge set is closed under subsets. Covering arrays and bounds on their size are strongly tied to homomorphisms and cores, and so we introduce the different types of homomorphisms of interest to us, and derive several theorems regarding hypergraph homomorphisms and cores. We then turn our attention to covering arrays in Section 2.2 rigorously defining orthogonal arrays, variable strength orthogonal arrays (VOA), and the variations of covering arrays listed above, and include several examples to illustrate some of the particular benefits of using variable strength covering arrays. We then provide a brief survey on topics of interest that are used in this thesis, including some of the known upper and lower bounds on covering arrays in Section 2.2.1 direct constructions of covering arrays in Section 2.2.2 recursive constructions...
Table 1.3: A variable strength covering array for the 3-tier architecture in Figure 1.3 with hypergraph in Figure 1.4. There are 27 test cases, numbered by column T, divided into two blocks for compactness.

We turn our attention to the theory of variable strength covering arrays in Chapter 3. We introduce a family of abstract simplicial complexes called qualitative independence hypergraphs, and show their connection to variable strength covering arrays. As with general covering arrays, homomorphisms also play a key role in variable strength covering arrays, and we discuss this connection in detail. We also demonstrate some operations we can perform on hypergraphs, which form a powerful tool that allows us to look at variable strength covering array problems in a variety of ways. We examine some divide-and-conquer techniques that can be used to build variable strength arrays over hypergraphs with certain properties, and finally look at direct constructions of arrays...
over certain types of hypergraphs.

In Chapter 4 we derive a construction for strength-3 covering arrays that improves many of the best previously known upper bounds. The general idea behind the construction is that we use linear feedback shift register sequences to build variable strength orthogonal arrays that have connections to combinatorial designs and projective geometries. By identifying what sets of factors are not fully covered by the variable strength orthogonal array, we can extend the array to the full strength-3 covering array. We begin an investigation of how this construction might be extended to higher strength covering arrays and identify some of the challenges that arise in our attempts.

We give an algorithm in Chapter 5 called VarDENS, which finds variable strength covering arrays over arbitrary hypergraphs, and thus serves as a practical tool for designing test suites for testing with variable strength arrays. The algorithm builds the array one row at a time, and builds each row by progressively selecting factors and levels for them until all factors have been assigned to levels. This algorithm uses the concept of density as introduced in [10, 11, 32], which allows us to guarantee that the row generated by the algorithm will cover at least the average number of uncovered interactions taken over all possible rows. This allows us to make a guarantee about the size of the resultant array, which, for fixed $t$ and $v$, grows logarithmically in the number of maximal edges of the hypergraph. Furthermore, we explain in a high level of detail the theory and implementation of density-based greedy algorithms for covering array generation, and feel that our detailed explanation will clarify implementation details of density-based algorithms. We finally perform a set of experiments to compare variants of VarDENS to determine what parameters are best, and then run some experiments comparing VarDENS with results obtained by simulated annealing and direct constructions. We then examine the case of using VarDENS to generate covering arrays over a type of combinatorial design called Steiner triple systems, which are of particular interest because they consist of the smallest number of 3-sets such that every 2-set is covered in some 3-set; thus, using these Steiner triple systems in the generation of arrays guarantees that the array performs pairwise testing while also including some strength-3 coverage.

The Lovász Local Lemma is a powerful tool that can be used in probabilistic arguments. The idea of the Local Lemma is that, if we have a collection of undesirable events that we wish to avoid, even if the events have some dependencies amongst them, if they are individually not too likely to occur, then the probability that none of them occur is nonzero. Godbole, Skipper, and Sunley [46] used the symmetric Local Lemma, which assumes that events occur with equal probability and have symmetric dependency sets,
to determine an upper bound on the size of general covering arrays. In Chapter 6, we similarly apply the probabilistic method, using the symmetric Local Lemma to nonconstructively determine upper bounds on the sizes of variable strength covering arrays over several different families of hypergraphs. We compare our bounds derived from the Local Lemma to those guaranteed by our VarDens algorithm in Chapter 5, and show that the Local Lemma bounds are significantly better in all cases. Finally, we look at some cases where events have differing probabilities and dependency sets. Even if the events differ in probabilities and dependency sets, by using upper bounds, we can apply the symmetric Local Lemma to obtain an upper bound on the size of an array, but due to the loss of information that occurs via this upper bounding, cases where the results are arbitrarily bad can be found. Because of this, we use more general forms of the Local Lemma combined with numerical methods to obtain bounds on the size of arrays in these cases, and then compare these bounds to those given by VarDens and the symmetric Local Lemma and note substantial improvements.

1.2.2 Major Contributions

We outline the main ideas and accomplishments of this thesis:

- We give the first comprehensive study of variable strength covering arrays, which are an important generalization of covering arrays that allow for domain knowledge to be taken into account in the testing process, and develop the framework of variable strength covering arrays and orthogonal arrays.

- We present a study of the connection between variable strength covering arrays and homomorphisms in Chapter 3, and introduce the concept of qualitative independence hypergraphs. We demonstrate the strong connection between covering arrays over ASC and the qualitative independence hypergraphs, and derive a number of important hypergraph-theoretic properties about these hypergraphs.

- In Section 3.4.1, we introduce two operations that are unique to variable strength covering arrays, namely the factor splitting operation of Theorem 3.4.1 and the factor joining operation of Theorem 3.4.3; when combined with a standard level-dropping covering array operation, these three operations offer a flexible tool for building VCA by demonstrating their relationship to other VCA; thus, an easier-to-solve problem or already known solution can be exploited through these three
to solve new classes of problems. We show several examples where the benefits of this technique become clear.

• We give several constructions that can be used to build VCA of different types. If a hypergraph has a cut vertex, in Section 3.4.2 we provide a means to approach the problem in a divide-and-conquer fashion on the subhypergraphs separated by the cut vertex, and this can be applied repeatedly on cut vertices on the subhypergraphs: VCA can be built on the subhypergraphs, and provided that these VCA have certain properties, we then present techniques to assemble these smaller VCA to derive a complete VCA on the full hypergraph. This technique can be generalized to cut-cliques but has stronger requirements, namely that the columns over the factors in the cut-clique have a balance property. VOA necessarily have this property, and thus are good candidates for this method. Additionally, in Section 3.4.3 we solve many instances of the problem of determining the size of VCA for a family of hypergraphs consisting of a series of consecutive edges, which can be linked cyclically and outline the remaining cases that need to be examined for the problem to be fully solved.

• In Chapter 4, we show that by using a degree-3 primitive polynomial over any finite field, we are able to construct a variable strength orthogonal array over a hypergraph whose edges are precisely the complement in the set of all sets of size $n+1$ of a projective plane of order $n$, and form a combinatorial design. By appending to the array a copy of the array with the rows reversed, we derive a construction for strength-3 covering arrays that improves upon many of the previously known bounds.

We generalize the idea of the construction to use degree-$t$ primitive polynomials to create variable strength orthogonal arrays, and identify the structure of the hypergraph. Finally, we detail attempts to modify the variable strength orthogonal array to a full covering array as we were able to do in the strength-3 case, and highlight the challenges that arise in doing so. We have submitted a paper consisting of the research from this chapter as [78].

• Chapter 5 presents an algorithm named VARDENS for finding VCA over arbitrary hypergraphs. The algorithm uses the concept of density in building the VCA one row at a time, and by doing so, is able to provide a guarantee on the size of the resultant arrays. This size guarantee serves as an upper bound on the size
of the VCA for the given hypergraph. This algorithm is of considerable practical importance, as it is one of the first algorithms that generates arbitrary VCA with a guarantee on their size, and may be directly used in the design of test suites for combinatorial testing. We have prepared and submitted a paper from the research of this chapter as [77].

• In Chapter 6 we show how we can use probabilistic arguments by applying the Local Lemma to derive nonconstructive upper bounds on the sizes of VCA. We demonstrate, for VCA over combinatorial designs, that the upper bound obtained through the probabilistic method is always better than the one guaranteed by VAR-DENS. Furthermore, the probabilistic method, when applied to deriving bounds for VCA over hypergraphs with consecutive (possibly cyclic) edges, is able to “detect” properties of the hypergraph that VAR-DENS is unable to, thus resulting in a much better bound that is similar to that obtained from the direct construction. We also give a family of hypergraphs based on triangulations of the sphere where the best array size is completely independent of the hypergraph, but neither the Local Lemma nor VAR-DENS is able to make this distinction. Finally, we use generalizations of the Local Lemma to show how we can find more precise bounds for VCA over hypergraphs where the edge sizes and intersection counts vary.

• In Chapter 7 we summarize our work in more detail and list open questions that have arisen during our research.
Chapter 2

Covering Arrays and Design Theory

This chapter consists of the general definitions and theory of covering arrays, and requisite mathematical background. We define graphs, hypergraphs, abstract simplicial complices, and homomorphisms in Section 2.1 and examine some of their important properties that will be of use when applying them to the theory of variable strength covering arrays. We also derive a number of results regarding cores and homomorphisms that are tightly linked to VCA.

In Section 2.2 we introduce the various generalizations of covering arrays, provide background on commonly studied problems of interest and a summary of important results to date. A review of the literature for variable strength covering arrays is given in Section 2.2.5. We additionally describe some of the known techniques that are used to generate arrays, both directly and recursively. These constructions will be of use in extending new results to larger classes of arrays, and generalizing constructions to variable strength covering arrays.

We define several families of combinatorial designs that will be of interest in our study of covering arrays in Section 2.3, and present some of their important properties, as well as some conditions required for their existence, when known.

Finally, to motivate the practical importance of covering array research, we provide an introduction to combinatorial testing, its ties to covering arrays, and their effectiveness as test suites in Section 2.4.
2.1 Graphs, Hypergraphs, and ASC

For a set $S$ and nonnegative integer $t$, let $\binom{S}{t} = \{ T \subseteq S : |T| = t \}$. If for sets $S$ and $T$, we have a map $f : S \rightarrow T$, then for a subset $S' \subseteq S$, define $f(S') = \{ f(s) : s \in S' \}$.

**Definition 2.1.1.** A graph $G$ is a pair $(V, E)$, where $V = V(G)$ is the finite set of vertices of $G$, and $E = E(G) \subseteq \binom{V}{2}$ is the set of edges of $G$.

**Definition 2.1.2.** Let $G = (V, E)$ be a graph. A graph $H = (V', E')$ is called a subgraph of $G$ if $V' \subseteq V$ and $E' \subseteq (E \cap \mathcal{P}(V'))$.

**Definition 2.1.3.** Let $K_n$ be the complete graph on $n$ vertices, that is $V(K_n) = \{0, \ldots, n-1\}$, and $E(K_n) = \binom{V(K_n)}{2}$.

**Definition 2.1.4.** For two graphs, $G$ and $H$, a graph homomorphism is a map $f : V(G) \rightarrow V(H)$ such that if $e \in E(G)$, then $f(e) \in E(H)$. If there is a homomorphism from $G$ to $H$, we say that $G$ is homomorphic to $H$, written $G \rightarrow H$. If $f$ is a bijective homomorphism with $|E(G)| = |E(H)|$, we call $f$ a graph isomorphism.

A c-colouring of a graph $G$ is a homomorphism $G \rightarrow K_c$. A graph is c-colourable if it has a c-colouring. A 2-colourable graph is called bipartite. The minimum $c$ such that a graph $G$ is c-colourable is called the chromatic number of $G$, written $\chi(G)$:

$$\chi(G) = \min\{ c \in \mathbb{N} : G \rightarrow K_c \}.$$

While we will discuss graphs, we are predominantly interested in hypergraphs, which are generalizations of graphs where the edges are not restricted to pairs of vertices. As a result, all of the concepts and definitions we examine for hypergraphs may equally be applied to graphs without modification. An introduction to hypergraph theory can be found in [8].

**Definition 2.1.5.** A hypergraph $H$ is a pair $(V, E)$, where $V = V(H)$ is the finite set of vertices of $H$, and $E = E(H) \subseteq \mathcal{P}(V)$ is the set of hyperedges (or simply edges) of $H$. A hyperedge of size 1 is called a loop. It is typical, as in [8], to consider a set of hyperedges $E$ to be a hypergraph with vertex set:

$$V = \bigcup_{e \in E} e.$$
Definition 2.1.6. A subhypergraph $G$ of a hypergraph $H$ has the property that $V(G) \subseteq V(H)$, $E(G) \subseteq E(H) \cap \mathcal{P}(V(G))$. A subhypergraph $G$ is induced if $E(G) = E(H) \cap \mathcal{P}(V(G))$. We write $E_i(H)$ to denote the set of edges of size $i$ of $H$:

$$E_i(H) = \{ e \in E(H) : |e| = i \}.$$ 

Furthermore, we consider $E_i(H)$ to be a subhypergraph of $H$.

One family of subhypergraphs of interest are the $n$-sections of $H$. We present an example in Figure 2.1.

![Figure 2.1: An example of a hypergraph $H$ on the left, and its 2-section $[H]_2$ on the right.](image)

Definition 2.1.7. If $H = (V, E)$ is a hypergraph, the $n$-section of $H$, written $[H]_n$, is the hypergraph $(V, E')$ such that, for all $e \in E$:

1. If $|e| < n$, then $e \in E'$.
2. Otherwise, for all $f \in \binom{e}{n}$, $f \in E'$.

By ignoring loops of $H$, we can think of $[H]_2$ as a graph, which is also called the primal graph of $H$ or the Gaifman graph of $H$.

Definition 2.1.8. A hypergraph is $t$-uniform, or alternatively, a $t$-hypergraph, if $E(H) \subseteq \binom{\binom{V}{t}}{t}$. The rank of a hypergraph $H$, denoted $\text{rank}(H)$, is the size of a largest edge in $H$:

$$\text{rank}(H) = \max\{|e| : e \in E(H)\}.$$
The *degree* of a vertex \( v \in V(H) \), denoted \( d_H(v) \), is the number of edges of \( H \) containing \( v \):

\[
d_H(v) = |\{ e \in E(H) : v \in e \}|.
\]

We denote the *t-degree* of a vertex \( v \in V(H) \), written \( d^t_H(v) \), as the number of edges of size \( t \) containing \( v \):

\[
d^t_H(v) = |\{ e \in E_t(H) : v \in e \}|.
\]

A hypergraph \( H \) is *\( s \)-regular* if there exists some integer \( s \geq 0 \) such that \( d_H(v) = s \) for all \( v \in V(H) \).

**Proposition 2.1.9.** For a hypergraph \( H \), and any vertex \( v \in V(H) \):

\[
d_H(v) = \sum_{i=1}^{\text{rank}(H)} d^i_H(v).
\]

Homomorphisms will play into many of our results and proofs, so we proceed to define hypergraph homomorphisms and identify the various types of homomorphisms that may occur.

**Definition 2.1.10.** For two hypergraphs \( G, H \), a hypergraph homomorphism \( f : G \to H \) is a map \( f : V(G) \to V(H) \) such that if \( e = \{v_0, \ldots, v_{m-1}\} \in E(G) \), then \( f(e) = \{f(v_0), \ldots, f(v_{m-1})\} \in E(H) \), and \( |e| = |f(e)| \). If a homomorphism from \( G \) to \( H \) exists, we say \( G \) is *homomorphic* to \( H \), written \( G \to H \).

Note that we specifically require hypergraph homomorphisms to retain edge size, so if there is some \( e \in E(G) \) such that \( u, v \in e, u \neq v \), then \( f(u) \neq f(v) \). There appears to be no clear consensus as to whether or not this property is required for hypergraph homomorphisms, but we will show later that it is essential in order to distinguish the hypergraphs of interest in this thesis.

An injective homomorphism is called a *monomorphism*, and if there is a monomorphism from \( G \) to \( H \), we write \( G \hookrightarrow H \), with the notation implying that \( G \) can be thought of as a subhypergraph of \( H \) via the action of the monomorphism. A surjective homomorphism is called an *epimorphism*, and if there is an epimorphism from \( G \) to \( H \), we write \( G \twoheadrightarrow H \).

If \( G \to H \) and \( H \to G \), we say that \( G \) and \( H \) are *homomorphically equivalent*. As we will see, homomorphically equivalent hypergraphs share many of the same properties.

If \( f : G \to H \) is a bijective homomorphism and \( |E(G)| = |E(H)| \), then we say that \( G \) and \( H \) are *isomorphic*, written \( G \cong H \), and call \( f \) an *isomorphism*. 
A homomorphism \( f : H \to H \) is called an endomorphism, and if \( f \) is an isomorphism, it is called an automorphism. The set of automorphisms of a hypergraph \( H \) forms a group under composition, denoted \( \text{Aut}(H) \).

**Proposition 2.1.11.** If \( G, H \) are hypergraphs and \( G \to H \), then \( \text{rank}(G) \leq \text{rank}(H) \).

**Proof.** Let \( t = \text{rank}(G) \), and \( e \in E(G) \) be an edge such that \( |e| = t \). Let \( f : G \to H \) be a homomorphism. Then \( |f(e)| = t \) and \( f(e) \in E(H) \), so \( \text{rank}(H) \geq t \).

We now define walks and paths in hypergraphs, which lead to the notion of the diameter of a hypergraph, and provide us with the tools to identify a particular family of hypergraphs that is of interest to us, namely the infinite family of hypertrees.

**Definition 2.1.12.** If \( H \) is a hypergraph, we say that \( P = u_0e_1u_1e_2 \ldots u_{k-1}e_ku_k \) is a walk from \( u_0 \) to \( u_k \) in \( H \) if \( u_0 \neq u_k \), and for all \( j, e_j \in E(H) \) and \( u_{j-1}, u_j \in e_j \). If \( u_0, \ldots, u_k \) and \( e_1, \ldots, e_k \) are distinct vertices and hyperedges of \( H \) respectively, we then call \( P \) a path from \( u_0 \) to \( u_k \) in \( H \). We say that \( P \) has length \( k \). If we require that vertices and edges are distinct but have \( u_0 = u_k \), we call \( P \) a cycle; a hypergraph with no cycles is acyclic. Given a hypergraph \( H \), two vertices \( u, v \in V(H) \) are connected if there is a path from \( u \) to \( v \) in \( H \). A hypergraph is connected if every pair of vertices in the hypergraph are connected. An acyclic connected graph is called a tree, and an acyclic connected hypergraph is a hypertree.

**Definition 2.1.13.** For a hypergraph \( H \), connectivity is an equivalence relation on \( V(H) \), and thus determines a partition \( \mathcal{P} = \{P_1, \ldots, P_s\} \) on the vertices. If, for \( 1 \leq i \leq s \), we let \( H_i \) be the induced subhypergraph of \( H \) on the set of vertices \( P_i \), we then call \( H_1, \ldots, H_s \) the connected components of \( H \). For convenience, we write \( \hat{H} \) to denote a connected component of \( H \) with the most vertices, i.e. \( \hat{H} = H_j \) if and only if \( |P_j| \geq |P_i| \) for all \( 1 \leq i \leq s \).

**Definition 2.1.14.** Let \( H = (V, E) \) and for \( u, v \in V \), write \( p_H(u, v) \) to represent the length of a shortest path from \( u \) to \( v \) in \( H \). If no such path exists, we then say \( p_H(u, v) = \infty \). Define:

\[
\text{diam}(H) = \max\{p_H(u, v) : u, v \in V\}.
\]

Then we call \( \text{diam}(H) \) the diameter of \( H \).

**Theorem 2.1.15.** If \( H \) is a hypergraph with \( u, v \in V(H) \) and there is a walk from \( u \) to \( v \) in \( H \) of length \( k \), then there exists a path from \( u \) to \( v \) of length at most \( k \).
Proof. Let \( H = (V, E) \) and write \( P = u_0e_1u_1\ldots u_{k-1}e_ku_k \) with \( u = u_0, v = u_k \) to be a walk from \( u \) to \( v \) of length \( k \). We then perform the following algorithm repeatedly to iteratively modify \( P \) until \( P \) is no longer changed by an iteration:

1. Repeat while there are duplicated vertices in \( P \): If \( u_i = u_j \) for \( i < j \), replace \( u_ie_{i+1}\ldots e_ju_j \) by \( u_i \).

2. Repeat while there are duplicated edges in \( P \): If \( e_i = e_j \) for \( i < j \), replace \( e_iu_i\ldots u_{j-1}e_j \) by \( e_i \).

After each modification, the fact that \( P \) is a walk is preserved. When the algorithm terminates, the edges and vertices of \( P \) are necessarily distinct and are still a walk. Thus, the modified \( P \) is a path from \( u \) to \( v \) in \( H \), and as the length can have only decreased by the operations, it has length at most \( k \). \(\square\)

**Corollary 2.1.16.** If \( H, G \) are hypergraphs and \( f: H \to G \) a homomorphism, then for any \( u, v \in V(H) \), \( p_G(f(u), f(v)) \leq p_H(u, v) \).

**Proof.** Let \( s = p_H(u, v) \) and let \( P = ue_1w_1\ldots e_sv \) be a path from \( u \) to \( v \) in \( H \). Then \( f(P) = f(u)f(e_1)f(w_1)f(e_2)\ldots f(e_s)f(v) \) is a walk from \( u \) to \( v \) in \( G \) of length \( s \), so by Theorem 2.1.15, there is a path from \( u \) to \( v \) of length at most \( s \) in \( G \), and thus \( p_G(f(u), f(v)) \leq s \). \(\square\)

We then have the following result relating diameters of subhypergraphs.

**Corollary 2.1.17.** If \( H \) is a hypergraph and \( H' \) is a spanning subhypergraph of \( H \), then \( \text{diam}(H) \leq \text{diam}(H') \).

**Proof.** Since \( H' \) is spanning, \( V(H') = V(H) \). Thus, for any two vertices \( u, v \in V(H) \), \( u, v \in V(H') \), so \( p_{H'}(u, v) \leq \text{diam}(H') \). Since any path from \( u \) to \( v \) in \( H' \) is also a path in \( H \), we have \( p_H(u, v) \leq p_{H'}(u, v) \), so \( \text{diam}(H) \leq \text{diam}(H') \). \(\square\)

Another concept of interest that derives from graph theory is the notion of cliques.

**Definition 2.1.18.** A complete \( t \)-uniform hypergraph of order \( n \) is a \( t \)-uniform hypergraph \( H \) such that \( |V(H)| = n \) and \( E(H) = \binom{V(H)}{t} \), denoted \( K_n^t \). Given a hypergraph \( H \), a \( t \)-clique of size \( n \) in \( H \) is a set of vertices \( C \subseteq V(H) \) of size \( n \) such that the induced subhypergraph of \( H \) on \( C \) contains a subhypergraph isomorphic to \( K_n^t \). We write \( \omega_t(H) \) to denote the size of the largest \( t \)-clique in \( H \), and note that \( \omega_t(K_n^t) = n \) and for any \( i \neq t \), \( \omega_i(K_n^t) = 0 \).
If $G$ is a graph, then $G$ can contain only nonempty 2-cliques, and we refer to these simply as cliques. Then $G$ contains a clique of size $n$ if it has a subgraph isomorphic to $K_n$, and we simply write $\omega(G)$ to denote the size of the largest clique of $G$.

**Proposition 2.1.19.** If $G, H$ are hypergraphs such that $G \rightarrow H$, then for any $t$, $\omega_t(G) \leq \omega_t(H)$.

**Proof.** Let $n = \omega_t(G)$ and assume $C \subseteq V(G)$ is a $t$-clique of size $n$ in $G$. Let $f : G \rightarrow H$ be a homomorphism. For every pair of distinct vertices $u, v \in C$, there is necessarily an edge in $G$ containing $u$ and $v$, so $f(u) \neq f(v)$. Thus, $f(C)$ has size $n$, and as $f$ is a homomorphism, for every size-$t$ edge $e \in E(G)$, $f(e)$ is a size-$t$ edge of $H$. Hence, $f(C)$ is a $t$-clique of size $n$ in $H$, so $\omega_t(H) \geq n$. \hfill $\Box$

**Definition 2.1.20** ([64]). In a hypergraph $H$, a $t$-coclique of size $n$ is a set of $n$ vertices $C$ such that no two vertices of $C$ occur together in any $t$-edge of $H$. We write $\alpha_t(H)$ to denote the size of the largest $t$-coclique of $H$. We note that $\alpha_t(K^t_n) = 1$, and for any $i \neq t$, $\alpha_t(K^t_n) = n$.

In the following theorem, for a hypergraph $H$, a set of $t$-cliques $\mathcal{C}$ in $H$ is said to uniformly cover $H$ if there exists some positive integer $n$ such that for any $v \in V(H)$:

$$|\{C \in \mathcal{C} : v \in C\}| = n.$$  

**Theorem 2.1.21** ([64]). Let $H$ be a hypergraph such that there is a non-empty set of maximum $t$-cliques that uniformly cover $V(H)$. Then:

$$\omega_t(H)\alpha_t(H) \leq |V(H)|.$$ 

Moreover, if equality holds in the above equation, the intersection of any $t$-coclique and any $t$-clique in $H$ has exactly one vertex.

**Proof.** Let $v = |V(H)|$ and write $\mathbf{1}_s$ to be the $s \times 1$ column vector with every entry 1. Assume we have a set of $m \geq 1$ maximum $t$-cliques in $H$ that uniformly cover $V(H)$, and let $N$ be the $v \times m$ matrix whose columns are the characteristic vectors of the $t$-cliques. Let $S$ be a maximum $t$-coclique, and let $x$ be the $v \times 1$ characteristic column vector of $S$. Since $S$ is a $t$-coclique, the intersection of $S$ with any $t$-clique has size at most 1. Thus, for $u = (u_0, \ldots, u_{m-1}) = x^T N$, we have that $0 \leq u_i \leq 1$ for all $0 \leq i \leq m - 1$. Thus:

$$x^T N \mathbf{1}_m \leq m. \quad (2.1)$$
By assumption, each vertex of $H$ must be in the same number of these $m$ maximum $t$-cliques, so each vertex must be in precisely $\frac{m\omega_t(H)}{v}$ of the maximum $t$-cliques. Thus, since the rows of $N$ represent the incidence of vertices in the maximum $t$-cliques, we have that

$$N1_m = \frac{m\omega_t(H)}{v}1_v.$$ 

Multiplying on the left by $x^T$ gives:

$$x^TN1_m = x^T\frac{m\omega_t(H)}{v}1_v = \frac{m\omega_t(H)}{v}(x^T1_v) = \frac{m\omega_t(H)\alpha_t(H)}{v}.$$ 

From Equation 2.1, this gives that:

$$\frac{m\omega_t(H)\alpha_t(H)}{v} \leq m$$

which implies that:

$$\omega_t(H)\alpha_t(H) \leq v$$

as required. If we have equality, then this implies that $x^TN = 1_m^T$, implying that $S$ intersects with every maximum $t$-clique exactly once. Since $S$ was arbitrarily chosen, any maximum $t$-coclique and any maximum $t$-clique must intersect in exactly one vertex. □

**Definition 2.1.22.** Let $V$ be a finite set. An abstract simplicial complex (ASC) over $V$ is a set $\Delta \subseteq \mathcal{P}(V)$ such that if $e \in \Delta$ and $f \subseteq e$, then $f \in \Delta$. We call $V(\Delta) = V$ the set of vertices of $\Delta$, and the sets in $\Delta$ are called faces or simplices. A maximal face of $\Delta$, i.e. a face $e \in \Delta$ such that if $f \in \Delta$, $e \subseteq f$, then $e = f$, is called a facet of $\Delta$. We write $\Lambda$ to denote the set of facets of $\Delta$, and call $\Lambda$ the facet subhypergraph of $\Delta$. We write $\Delta_i$ to denote the set of faces of size $i$ in $\Delta$:

$$\Delta_i = \{e \in \Delta : |e| = i\} = E_i(\Delta).$$

Then $\Delta_i$ can be considered, in its own right, a hypergraph, and a subhypergraph of $\Delta$.

A sub-ASC $\Delta'$ of an ASC $\Delta$ has the properties that $\Delta' \subseteq \Delta$ and $\Delta'$ is itself an ASC.

For a hypergraph $H$, we can extend $H$ to an ASC over $V(H)$ as follows:

$$\Delta(H) = \bigcup_{e \in E(H)} \mathcal{P}(e) = \bigcup_{i=0}^{\text{rank}(H)} [H]_i.$$ 

We call this the ASC induced by $H$. 

As ASC are simply hypergraphs whose edge set is closed under subsets, homomorphisms over ASC are simply hypergraph homomorphisms, and thus all the properties of hypergraph homomorphisms also hold for ASC. It becomes clear here why we have chosen to use a definition of hypergraph homomorphism that preserves edge size: if not, every pair of nonempty ASC $\Delta_1$ and $\Delta_2$ would be homomorphic simply by mapping all elements of $V(\Delta_1)$ to an edge of size 1 in $\Delta_2$.

We now introduce the notion of a $n$-skeleton of an ASC, which is similar to the idea of the $n$-section of a hypergraph.

**Definition 2.1.23.** For an ASC $\Delta$, the $n$-skeleton of $\Delta$, denoted $\text{skel}_n(\Delta)$, is the sub-ASC of $\Delta$ defined by:

$$\text{skel}_n(\Delta) = \bigcup_{i=0}^{n+1} \Delta_i.$$

**Definition 2.1.24.** A hypergraph $H$ is called a core if every endomorphism of $H$ is an automorphism. For an arbitrary hypergraph $H$, a core of $H$, denoted $H^\bullet$, is a subhypergraph of $H$ such that $H^\bullet$ is a core and there exists a homomorphism $H \to H^\bullet$.

**Theorem 2.1.25.** Given any hypergraph $H = (V,E)$, $H$ has a core.

*Proof.* If $H = (V,E)$ is not a core itself, consider the nonempty set of all endomorphisms that are not automorphisms $S$. Every endomorphism $f \in S$ has the property that $|f(V)| < |V|$. Pick $g \in S$ to be an endomorphism such that $|g(V)|$ is minimum. We then claim that the subhypergraph $H' = (g(V),g(E))$ of $H$ is a core: if not, there is an endomorphism of $H'$ that is not an automorphism, say $h$. Then $|h(g(V))| < |g(V)|$. But then $g \circ h$, the composition of $g$ with $h$, is an endomorphism of $H$ that is not an automorphism, and furthermore, $|(g \circ h)(V)| = |h(g(V))| < |g(V)|$, thus contradicting the choice of $g$ as having minimum size $|g(V)|$. Thus, $H'$ is a core of $H$. \hfill \Box

**Proposition 2.1.26.** Let $H$ be a hypergraph. The homomorphism $f: H \to H^\bullet$ must be an epimorphism.

*Proof.* If $f$ is not surjective, then the restriction map $f|_{H^\bullet}: H^\bullet \to H^\bullet$ is also not surjective, and thus not an automorphism, contradicting the fact that $H^\bullet$ is a core. \hfill \Box

**Theorem 2.1.27.** Every hypergraph has a unique core up to isomorphism.

*Proof.* Let $G$ be a hypergraph, and assume it has cores $G_1^\bullet$ and $G_2^\bullet$. Then we have homomorphisms:

$$f: G_1^\bullet \hookrightarrow G \twoheadrightarrow G_2^\bullet, \quad g: G_2^\bullet \hookrightarrow G \twoheadrightarrow G_1^\bullet.$$
Then $fg$ is necessarily an automorphism of $G_1^*$, and $gf$ is an automorphism of $G_2^*$, so $|V(G_1^*)| = |V(G_2^*)|$ so $f$ and $g$ are bijections. Furthermore, if $e \in E(G_1^*)$, then $f(e) \in E(G_2^*)$, and if $e' \in E(G_2^*)$, then $g(e') \in E(G_1^*)$, so $|E(G_1^*)| = |E(G_2^*)|$, and thus, $G_1^* \cong G_2^*$.

\begin{proof}

\textbf{Lemma 2.1.28.} Let $G$, $H$ be hypergraphs. If $f : G \to H$ is a homomorphism, then for any $i \geq 0$:

1. $f|_{E_i(G)} : E_i(G) \to E_i(H)$ is a homomorphism, and

2. $f|_{[G]_i} : [G]_i \to [H]_i$ is a homomorphism.

\textbf{Proof.} Consider an edge $e \in E_i(G)$, so $|e| = i$. Then $e \in E(G)$, so $f|_{E_i(G)}(e) = f(e) \in E(H)$ and $|e| = |f(e)| = i$ since $f$ is a homomorphism, and homomorphisms preserve edge sizes. Thus, $f|_{E_i(G)} \in E_i(H)$, as required, so $f|_{E_i(G)} : E_i(G) \to E_i(H)$ is a homomorphism.

Now, if $e \in [G]_i$, then we have one of two cases:

1. If $e \notin E(G)$, then $|e| = i$ and there is an edge $e' \in E(G)$ such that $e \subseteq e'$. Since $f : G \to H$ is a homomorphism, we have that $f(e') \in E(H)$ and $|e'| = |f(e')|$ since homomorphisms preserve edge size. Thus, for all $u, v \in e'$, if $f(u) = f(v)$, then $u = v$. Hence, as $e \subseteq e'$, the same is true for $e$, so $|f(e)| = |e| = i$. Additionally, $f(e) \subseteq f(e')$, and since $f(e') \in E(H)$, then $f|_{[G]_i} = f(e) \in [H]_i$.

2. If $e \in E(G)$, then for some $1 \leq j \leq i$, $e \in E_j(G)$. Since, from the first part, $f|_{E_j(G)} : E_j(G) \to E_j(H)$ is a homomorphism, then $f|_{[G]_i}(e) = f(e) \in E_j(H) \subseteq [H]_i$.

Thus, $f : [G]_i \to [H]_i$ is a homomorphism.

\textbf{Corollary 2.1.29.} Let $\Delta$, $\Delta'$ be ASC. If $f : \Delta \to \Delta'$ is a homomorphism, then for any $i \geq 1$, $f : \text{ske} \_i(\Delta) \to \text{ske} \_i(\Delta')$ is also a homomorphism.

\textbf{Theorem 2.1.30.} Let $H = (V, E)$ be a hypergraph. If, for any $i \geq 0$, $E_i(H)$ is a core, then $H$ is a core.

\textbf{Proof.} If $H$ is not a core, then its core $H^\bullet$ must be strictly smaller than $H$, i.e. $|V(H^\bullet)| < |V(H)|$. Since $H^\bullet$ is a subhypergraph of $H$, $E_i(H^\bullet)$ is a subhypergraph of $E_i(H)$, and as $H^\bullet$ is strictly smaller than $H$, then $|E_i(H^\bullet)| < |E_i(H)|$. There is a homomorphism $f : H \to H^\bullet$, so, by Lemma 2.1.28, $f : E_i(H) \to E_i(H^\bullet)$ is a homomorphism, contradicting the fact that $E_i(H)$ is a core.

\end{proof}
The core of a hypergraph is homomorphically equivalent to the hypergraph itself, and thus a hypergraph’s core preserves many important properties of the hypergraph.

Proposition 2.1.31. If $H$ is a hypergraph, $\omega_t(H) = \omega_t(H^*)$.

Proof. We have the homomorphism $H \to H^*$, and the inclusion homomorphism $H^* \hookrightarrow H$. Thus, Proposition 2.1.19 gives the required result.

The notion of a colouring on a graph does not generalize in a unique way to hypergraphs and cannot be expressed so nicely in terms of hypergraph homomorphisms, thus leading to several different notions of colourings stated in terms of functions or partitions of the vertices. We are interested in particular in two types of hypergraph colourings, namely weak and strong colourings.

Definition 2.1.32. For a hypergraph $H = (V,E)$, a weak $c$-colouring is a surjective function $f: V \to \mathbb{Z}_c$ such that for every $e \in E$, $|f(e)| = 1$ if and only if $|e| = 1$, i.e. the only monochromatic edges of $H$ are the loops. We can also think of a $c$-colouring $f$ of $H$ as a partition of $V$ into $c$ nonempty classes $C_0, \ldots, C_{c-1}$, with vertex $v \in V$ in class $C_i$ if and only if $f(v) = i$. The weak chromatic number of $H$, written $\chi(H)$, is the smallest integer $c$ for which a weak $c$-colouring of $H$ exists. A strong $c$-colouring is a surjective function $f: V \to \mathbb{Z}_c$ such that for every $e \in E$, $|f(e)| = |e|$. As with weak colourings, we can associate with each strong colouring of $H$ a partition of $V$ into $c$ nonempty classes. The strong chromatic number of $H$, written $\gamma(H)$ is the smallest integer $c$ for which a strong $c$-colouring of $H$ exists.

A strong colouring of a hypergraph is always also a weak colouring, and thus $\gamma(H) \geq \chi(H)$. Furthermore, a strong colouring $f$ of $H$ is simply a colouring of the graph $[H]_2$, so $\gamma(H) = \chi([H]_2)$, where $f$ is a homomorphism $[H]_2 \to K_c$; thus, when $H$ is a graph, the two notions of colouring are identical and correspond to the usual definition of graph colouring. For an ASC $\Delta$, it also turns out that the notions of colouring are equivalent, so we can refer to the two simply as a $c$-colouring of $\Delta$.

Proposition 2.1.33. For an ASC $\Delta$, every weak colouring of $\Delta$ is also a strong colouring of $\Delta$, so $\chi(\Delta) = \gamma(\Delta)$.

Proof. Let $f$ be a weak colouring of $\Delta$, and consider any face $e \in \Delta$, $|e| \geq 2$ with $u, v \in e$, $u \neq v$. Since $\{u, v\} \in \Delta$, we have $f(u) \neq f(v)$.

In terms of hypergraph homomorphisms, we have the following results.
Theorem 2.1.34. Let $H$ be a hypergraph and $f: V(H) \rightarrow \mathbb{Z}_c$ a map. Then $f$ is a strong $c$-colouring if and only if for $H' = (\mathbb{Z}_c, \{f(e) : e \in E(H)\})$, we have $H \rightarrow H'$.

Proof. If $f$ is a strong $c$-colouring of $H$, then $f$ is a homomorphism $H \rightarrow H'$. If $H \rightarrow H'$, then $f$ must be such a homomorphism, and then for any edge $e \in E(H)$, we have that $|f(e)| = |e|$, thus guaranteeing that $f$ is a strong $c$-colouring.

Theorem 2.1.35. Let $H$ be a hypergraph. Then $H$ is $t$-uniform and has a strong $c$-colouring if and only if $H \rightarrow K^t_c$.

Proof. Assume $H$ is $t$-uniform and has a strong $c$-colouring $f: V(H) \rightarrow \mathbb{Z}_c$. Then we apply Theorem 2.1.34 and note that $H'$ is also $t$-uniform, so $H' \hookrightarrow K^t_c$.

If we have $f: H \rightarrow K^t_c$, then since for all $e \in E(H)$, $|f(e)| = t$, we have that $H$ is $t$-uniform. Furthermore, $f$ defines a strong $c$-colouring of $H$.

With regards to ASC, we get the following results.

Theorem 2.1.36. Let $\Delta$ be an ASC with facet subhypergraph $\Lambda$ such that $\text{rank}(\Delta) = \text{rank}(\Lambda) = t$. If $\Delta$ is $c$-colourable via $f: V(\Delta) \rightarrow \mathbb{Z}_c$, then:

$$\Delta \rightarrow \{f(e) : e \in \Delta\} \hookrightarrow \Delta(K^t_c).$$

Proof. It is obvious that $\Delta' = \{f(e) : e \in \Delta\}$ is an ASC since $\Delta$ is an ASC, and furthermore that $\text{rank}(\Delta') = t$, so $\Delta' \subseteq \Delta(K^t_c)$.

2.1.1 General Relational Systems

There is a generalization of the concept of hypergraphs called general relational systems. We provide a brief introduction to them below, and show the equivalence between hypergraphs, general relational systems, and homomorphisms to motivate the idea that it may be possible to use general relational systems to also capture additional testing concepts in covering arrays.

Definition 2.1.37. A general relational system (GRS) $S$ is a finite set $V = V(S)$ of vertices, together with a finite set of relations $R_i(S)$ for $i \in I$ some indexing set, such that $R_i(S)$ is a $k_i$-ary relation on $V$. The finite set $I$ and the integers $k_i$, $i \in I$, describing the arities of the relations form the type or pattern $P$ of the GRS.

A hypergraph $H$ can be viewed as a specific type of GRS $S$ as follows:
1. \( V(S) = V(H) \).

2. \( I = \{ |e| : e \in E(H) \}, k_i = i \).

3. For \( e \in E(H) \), let \( S_e \) be the set of all orderings of the elements in \( e \). Then define:

\[
R_i(S) = \bigcup_{e \in E(H), |e| = i} S_e.
\]

**Example 2.1.38.** If we have the hypergraph \( H \) with:

\[
V(H) = \{0, 1, 2, 3, 4\}
\]

\[
E(H) = \{\{0, 1, 2\}, \{0, 3\}, \{0, 4\}, \{1, 3, 4\}, \{2\}, \{3, 4\}\},
\]

then this gives GRS \( S \) with \( V(S) = V(H) \) and relations:

\[
R_1(S) = \{(2)\}
\]

\[
R_2(S) = \{(0, 3), (3, 0), (0, 4), (4, 0), (3, 4), (4, 3)\}
\]

\[
R_3(S) = \{(0, 1, 2), (0, 2, 1), (1, 2, 0), (1, 0, 2), (2, 0, 1), (2, 1, 0),
\]

\[
= (1, 3, 4), (1, 4, 3), (3, 1, 4), (3, 4, 1), (4, 1, 3), (4, 3, 1)\}.
\]

**Definition 2.1.39.** For two GRS \( S \) and \( T \), a **GRS homomorphism** \( f : S \to T \) is a map \( f : V(S) \to V(T) \) such that if \((v_0, \ldots, v_{k_i-1}) \in R_i(S)\), then \((f(v_0), \ldots, f(v_{k_i-1})) \in R_i(T)\).

If \( S \) is homomorphic to \( T \), we write \( S \to T \).

**Theorem 2.1.40.** Let \( G \) and \( H \) be hypergraphs with associated GRS \( S \) and \( T \) respectively. Then \( G \to H \) if and only if \( S \to T \).

**Proof.** Let \( f : G \to H \). Since \( V(S) = V(G) \) and \( V(T) = V(H) \) by construction, we form the map:

\[
g : V(S) \to V(T)
\]

\[
s \mapsto f(s)
\]

This is a homomorphism: if \((v_0, \ldots, v_{k_i-1}) \in R_i(S)\), then \( e = \{v_0, \ldots, v_{k_i-1}\} \in E(G)\), so \( f(e) = \{f(v_0), \ldots, f(v_{k_i-1})\} = \{g(v_0), \ldots, g(v_{k_i-1})\} \in E(H) \) with \( g(v_m) \neq g(v_n) \) for \( 0 \leq m < n \leq k_i - 1 \), thus \((g(v_0), \ldots, g(v_{k_i-1})) \in R_i(T)\).
Similarly, if $f : S \to T$ is a homomorphism, we form the map:

$$g : V(G) \to V(H)$$

$$s \mapsto f(s)$$

This is also a homomorphism: if $\{v_0, \ldots, v_{k-1}\} \in E(G)$, then $(v_0, \ldots, v_{k-1}) \in R_i(S)$, so $(f(v_0), \ldots, f(v_{k-1})) \in R_i(T)$, and thus $\{f(v_0), \ldots, f(v_{k-1})\} = g(\{v_0, \ldots, v_{k-1}\}) \in E(H)$ as required.

### 2.2 Covering Arrays and their Generalizations

We are predominantly interested in covering arrays, and will focus in particular on one generalization of covering arrays called *variable strength covering arrays*. Covering arrays are a generalization of the well-studied orthogonal arrays, which were first proposed and examined by Rao in his Master’s thesis and refined and studied through a series of papers in the 1940s: he began by initially introducing hypercubes of strength $d$, and in [79], extended the definition to cover the wider class of orthogonal arrays. Rao illustrated the use of hypercubes of strength $d$ in creating confounded designs for factorial experiments, and demonstrated that they could be constructed using configurations of points and planes from finite projective geometries.

We now proceed to define the structures of interest. There are several equivalent ways to define covering arrays and orthogonal arrays: in this section, we first define coverage in an $N \times k$ array over a set $V$, and then describe the arrays in terms of their coverage properties. Later, when convenient to do so, we will explore alternative definitions that will better illustrate the connections of these arrays to other mathematical structures.

**Definition 2.2.1.** If $M$ is an $N \times k$ array over a set $V$, for a set of columns $B = \{b_0, \ldots, b_{s-1}\} \subseteq \{0, \ldots, k-1\}$, we say that $B$ is $\lambda$-covered if the $N \times s$ subarray over the columns of $B$ has each $s$-tuple over $V$ as a row at least $\lambda$ times; otherwise, $B$ is $\lambda$-uncovered. If $\lambda = 1$, we simply say $B$ is covered or uncovered. We say that $B$ is balance-covered if $|V|^s$ divides $N$ and the $N \times s$ subarray over the columns of $B$ has every $s$-tuple over $V$ as a row precisely $N/|V|^s$ times; otherwise, $B$ is balance-uncovered.

**Definition 2.2.2.** An $\text{OA}_\lambda(t, k, g)$ orthogonal array is a $\lambda g^t \times k$ array over $\{0, \ldots, g-1\}$ such that every $B \in \binom{[0, \ldots, k-1]}{t}$ is balance-covered. When $\lambda = 1$, this parameter is often omitted. We say that the array has strength $t$, $k$ factors, $g$ levels, and index $\lambda$. 


Orthogonal arrays are used in the design of statistical experiments, where the rows of the array prescribe the configurations of the tests. A collection of results regarding orthogonal arrays can be found in [51]. They are closely connected with sets of mutually orthogonal latin squares, transversal designs, and projective geometries; the interested reader is referred to [86], where these equivalences are given in detail; we provide a brief overview in Section 2.2.2 of the results of importance to this work. For many combinations of parameters $t$, $k$, and $g$, orthogonal arrays do not exist for small values of $\lambda$, which can limit their usefulness in this application, and thus motivates the study of their generalization to covering arrays.

**Definition 2.2.3.** A CA$\lambda(N; t, k, g)$ covering array is an $N \times k$ array over $\{0, \ldots, g - 1\}$ such that every $B \in \binom{\{0, \ldots, k-1\}}{t}$ is $\lambda$-covered. When $\lambda = 1$, this parameter is often omitted. We say that the array has size $N$, strength $t$, $k$ factors, $g$ levels, and index $\lambda$. Given $t$, $k$, $g$, and $\lambda$, the smallest $N$ for which a CA$\lambda(N; t, k, g)$ exists is denoted CAN$\lambda(t, k, g)$. A CA$\lambda(N; t, k, g)$ with $N = \text{CAN}_\lambda(t, k, g)$ is said to be optimal.

There are two problems of interest in the study of covering arrays, namely:

1. For fixed $\lambda$, $t$, $k$, and $g$, what is CAN$\lambda(t, k, g)$?

2. For fixed $\lambda$, $N$, $t$, and $g$, what is the largest $k$ such that a CA$\lambda(N; t, k, g)$ exists?

Covering arrays with $\lambda = 1$ have been studied under the guise of a variety of equivalent structures, including $(k, t)$-universal sets, which are covering arrays with $g = 2$, and more generally, as $t$-surjective arrays, which are fully equivalent structures. A general survey of covering arrays is given in [30]. One of the primary uses of covering arrays is in the design of test suites. Given a system with $k$ factors, each with $g$ possible levels, exhaustively testing the system would require $g^k$ test cases, and unless both $g$ and $k$ are small, this is typically too costly. Instead of exhaustive testing, covering arrays can be used to perform $t$-way interaction testing, where for every $t$-set of factors, all possible $g^t$ combinations of levels over those factors are tested. We discuss combinatorial testing in Section 2.4.

**Example 2.2.4.** In Figure 2.2, there is an example of a CA$(10; 3, 5, 2)$. If we have a system with $k = 5$ binary factors, the rows represent the $N = 10$ test cases that can be used to perform 3-way interaction testing over the system. To exhaustively test this system would require $2^5 = 32$ tests. As $k$ increases, the savings become even more apparent: if our system had $k = 20$ factors, exhaustive testing would require $2^{20} = 1,048,576$ test cases, whereas 3-way testing can be performed with 18 test cases.
In general, for $\lambda$, $t$, and $g$ fixed, we can construct a covering array of size $O(\log k)$, as shown in [11].

$$
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 1
\end{array}
$$

Figure 2.2: On the left, a CA(10; 3, 5, 2). On the right, a VOA(8, $\Lambda$, 2), where $\Lambda$ is the set of blocks of a (7, 3, 4)-BIBD.

The standard model of covering arrays has several drawbacks when used in the design of test suites, which has led to generalizations of covering arrays. Firstly, it is not often the case in real-world systems that the number of levels is uniform across all factors. **Mixed-level covering arrays**, as proposed in [71], allow for factors to have different numbers of levels. We examine mixed-level variable strength covering arrays in greater detail in Section 3.3.

**Definition 2.2.5.** An $\text{MCA}_\lambda(N; t, (g_0, \ldots, g_{k-1}))$ mixed-level covering array is an $N \times k$ array such that column $i$ has entries over $G_i = \{0, \ldots, g_i - 1\}$. Furthermore, every $B = \{b_0, \ldots, b_{t-1}\} \in \binom{\{0, \ldots, k-1\}}{t}$ has the property that the $N \times t$ subarray over the columns in $B$ contains every tuple from $G_{b_0} \times \ldots \times G_{b_{t-1}}$ at least $\lambda$ times. We say that the array has size $N$, strength $t$, $k$ factors with factor $i$ having $g_i$ levels, and index $\lambda$. Given $\lambda$, $t$, and $g_0, \ldots, g_{k-1}$, the smallest $N$ for which a $\text{MCA}_\lambda(N; t, (g_0, \ldots, g_{k-1}))$ exists is denoted $\text{MCAN}_\lambda(t, (g_0, \ldots, g_{k-1}))$. An $\text{MCA}_\lambda(N; t, (g_0, \ldots, g_{k-1}))$ with $N = \text{MCAN}_\lambda(t, (g_0, \ldots, g_{k-1}))$ is said to be optimal. When convenient to do so, we can rewrite the $k$-tuple of number of levels, $(g_0, \ldots, g_{k-1})$, as a function $w: \{0, \ldots, k - 1\} \to \mathbb{N}^+$, where $w(i) = g_i$. Additionally, it is common to express the levels in exponential notation, writing $g_0^{a_0}g_1^{a_1} \cdots g_d^{a_d}$ to indicate that the array has a total of $k = \sum_{i=0}^{d} a_i$ factors, $a_i$ of which have $g_i$ levels, for $0 \leq i \leq d$. When $\lambda = 1$, it is also common to omit this.
Another drawback is that covering arrays do not allow for domain knowledge to be taken into account in the design of test suites. This black-box feature is generally considered to be one of the strengths of covering arrays, as it allows them to be used to test arbitrary systems. However, knowledge of the system under testing may indicate that certain factors are known not to interact, in which case it is unnecessary to test all interactions between them; Cheng et al. [18] propose a model where sets of factors that jointly affect one of the output values of a software system should be considered to interact. For pairwise interaction testing, i.e. covering arrays with \( t = 2 \), the relationships of interacting factors may be modeled via a graph with one vertex per factor and an edge between two factors that interact, representing that we wish to test all pairs of levels between the factors. Some known results on covering arrays on graphs are given in Section 2.2.4. For higher strengths, we can dictate the combination of factors we want to test by modeling them as the facets of an ASC. This leads to the concept of a variable strength covering array, which is the primary focus of this thesis.

**Definition 2.2.6.** Let \( \Delta \) be an ASC over \( \{0, \ldots, k - 1\} \) with set of facets \( \Lambda \), and let \( t = \text{rank}(\Delta) \). A \( \text{VCA}_\lambda(N; \Lambda, g) \) variable strength covering array, where \( \lambda = (\lambda_1, \ldots, \lambda_t) \), is an \( N \times k \) array over \( \{0, \ldots, g - 1\} \) with columns \( 0, \ldots, k - 1 \) such that if \( B = \{b_0, \ldots, b_{s-1}\} \in \Lambda \), then \( B \) is \( \lambda_s \)-covered. When \( \lambda_i = 1 \) for all \( i \in \{|B| : B \in \Lambda\} \), the parameter \( \lambda \) is frequently omitted. We take \( \text{VCAN}_\lambda(\Lambda, g) \) to be the smallest \( N \) such that a \( \text{VCA}_\lambda(N; \Lambda, g) \) exists.

We note the following observation.

**Proposition 2.2.7.** An array \( H \) is a \( \text{CA}(N; t, k, g) \) if and only if it is a \( \text{VCA}(N; K^t_k, g) \).

**Proof.** The edges of \( K^t_k \) are precisely what we need covered in the covering array, thus giving the result. \( \square \)

As covering arrays are a generalization of orthogonal arrays with less stringent requirements, we can also define a variable strength equivalent of orthogonal arrays, of which variable strength covering arrays are a generalization.

**Definition 2.2.8.** For an ASC \( \Delta \) with facet set \( \Lambda \), a \( \text{VOA}(N; \Lambda, g) \) variable strength orthogonal array is an \( N \times k \) array over \( \{0, \ldots, g - 1\} \) with columns \( 0, \ldots, k - 1 \) such that if \( B = \{b_0, \ldots, b_{s-1}\} \in \Delta \), then \( B \) is balance-covered.
Example 2.2.9. Figure 2.2 has an example of a VOA that is not an OA, where \( \Lambda \) is the set of all 3-sets except those of the form \( \{i, i + 1, i + 3\} \), or in other words, all 3-sets but the blocks of a Fano plane.

In addition, as with covering arrays, we can allow the factors of variable strength covering arrays to have different numbers of levels. This is represented by replacing \( g \) in the definition by a \( k \)-tuple \( g = (g_0, \ldots, g_{k-1}) \) representing the number of possible levels for each of the \( k \) factors. This can also be written in exponential notation for convenience.

Example 2.2.10. In Figure 2.3, we have an example of a six-factor VCA(27; \( \Lambda, 3^59 \)). We refer to the factors as 0,1,2,3,4,5, with factors 0,1,2,3,4 having 3 levels and factor 5 having 9 levels, and take \( \Lambda = \{(0, 1, 2, 3, 4) \times \{5\} \cup ([0,1,2,3,4]/3) \setminus \{0, 2, 4\}\). Thus, omitting brackets and commas for compactness, we have:

\[
\Lambda = \{05, 15, 25, 35, 45, 012, 013, 014, 023, 034, 123, 124, 134, 234\}.
\]

This represents the following abstract simplicial complex:

\[
\Delta = \left\{ \emptyset, 0, 1, 2, 3, 4, 5, 01, 02, 03, 04, 05, 12, 13, 14, 15, 23, 24, 25, 34, 35, 45, 012, 013, 014, 023, 034, 123, 124, 134, 234 \right\}.
\]

Note that since we need all interactions over the set of factors 012, each of which has three levels, a lower bound on the number of tests in a VCA is \( 3^3 = 27 \), so the VCA in Figure 2.3 is optimal. Furthermore, the smallest \( N \) for which a CA(\( N; 3^59 \)) exists is 33; by omitting one 3-set of factors, 024, we are able to reduce the number of tests required to achieve coverage. Additionally, if we imposed a uniform strength of 3 over all factors, i.e., MCA(\( N; 3; 3^59 \)), we would have \( N \geq 9 \cdot 3^2 = 81 \), so the reduced testing requirement for the interactions involving the last factor with 9 levels guarantees a reduction of at least two-thirds of the size of a complete strength-3 array.

When discussing covering arrays, and in particular algorithms that build covering arrays as a sequence of steps, it is useful to be able to refer to the assignment of levels to factors in a given row, which we call an interaction.

Definition 2.2.11. For a set of factors \( W = \{i_0, \ldots, i_{m-1}\} \), where factor \( i_j \) has \( g_j \) levels, we define an interaction \( S \) over \( W \) to be an assignment of levels to factors, written as a set of pairs \( S = \{(i_0, \sigma_0), \ldots, (i_{m-1}, \sigma_{m-1})\} \), where \( \sigma_j \in \{0, \ldots, g_j - 1\} \). \( S \) is said to
be \(m\)-way, or alternatively called an \(m\)-interaction. The rows of a \(CA(N; t, k, g)\) can then be seen as \(k\)-way interactions, and the restriction to \(t\) columns can be seen as \(t\)-way interactions. The covering array then can be defined by the property that for any set \(W\) of \(t\) factors, every \(t\)-interaction over \(W\) appears in a row of the array.

**Definition 2.2.12.** If \(S\) is an interaction over a set \(W\) of factors, then for \(U \subseteq W\), we call an interaction \(T\) over \(U\) a subinteraction of \(S\) if \(T \subseteq S\). We say that \(S\) extends \(T\).

**Definition 2.2.13.** Let \(W = \{0, \ldots, k - 1\}\) be a set of factors, and let \(S\) be the set of all possible interactions over all possible subsets of \(W\). Define the following interaction index function:

\[
\phi : S \rightarrow W \\
S \mapsto \{f : (f, \sigma) \in S\}
\]

Then if \(U, V \subseteq W\), and \(S, T\) are interactions over \(U, V\) respectively, we say that \(S\) respects \(T\) (alternatively, \(S\) and \(T\) respect each other) if, for all \(f \in \phi(S) \cap \phi(T)\), if \((f, \sigma) \in S\), then \((f, \sigma) \in T\).

**Definition 2.2.14.** For an interaction \(S\) over a set \(W\) of factors, and \(f \in W\), if \(f \in \phi(S)\), then \(f\) is fixed in \(S\). Otherwise, \(f\) is free with respect to \(S\).
2.2.1 Bounds on CAN

Of particular interest is the study of the growth of the size of covering arrays across various construction techniques as a function of the number $k$ of factors. We present some simple bounds on covering arrays, which we use throughout this thesis.

Very few lower bounds have been established for covering arrays. Trivially, to accommodate the appearance of $\lambda$ copies of each of the $g^t$ tuples over the rows of the array, we have:

$$\text{CAN}_\lambda(t, k, g) \geq \lambda g^t.$$

A similar bound exists for the mixed level case: if we assume without loss of generality $g_0 \geq \ldots \geq g_{k-1}$, we have that:

$$\text{MCAN}(t, (g_0, \ldots, g_{k-1})) \geq \prod_{i=0}^{t-1} g_i.$$

The same bound, for $\text{VCA}(N; \Lambda, (g_0, \ldots, g_{k-1}))$, with no need to order factors, has the form:

$$\text{VCAN}(N; \Lambda, (g_0, \ldots, g_{k-1})) \geq \max \left\{ \prod_{i \in B} g_i : B \in \Lambda \right\}. \quad (2.2)$$

We can construct examples arbitrarily far from this bound: in Section 2.2.2, we show that the values of $\text{CAN}(2, k, 2)$ have been fully determined, and for any value $N \geq 4$, taking $k = \left( \frac{N-1}{2} \right)$, we have $\text{CAN}(2, k, 2) = \text{VCA}(K_k, 2) = N$, but Equation 2.2 gives us a bound of 4.

In Equation 3.4 of [84], Stevens gives the following general lower bound for strength-2 covering arrays:

$$\text{CAN}(2, k, g) \geq \left\lceil \frac{g \log k}{2} \right\rceil + g + 1.$$

Poljak and Tuza [75] use extremal set theory to show that, for any $k$ and $g$:

$$k \leq \frac{1}{2} \left( \left\lceil \frac{2\text{CAN}(2,k,g)}{g} \right\rceil - \left\lceil \frac{\text{CAN}(2,k,g)}{g} \right\rceil \right).$$

Stevens, Moura, and Mendelsohn [85] improved this bound slightly. Define:

$$\delta_{t,g} = \begin{cases} 1 & \text{if } t \equiv -1 \pmod{g} \\ 0 & \text{otherwise.} \end{cases}$$
They then show that:

\[ k \leq \frac{1}{2} \left( 2 \left\lfloor \frac{\text{CAN}(2,k,g)}{g} \right\rfloor + \delta_{\text{CAN}(2,k,g),g} \right). \]

By defining the following function:

\[ h_g(t) = \frac{1}{2} \left( 2 \left\lfloor \frac{t}{g} \right\rfloor + \delta_{t,g} \right), \]

they derive the following lower bound for strength-2 covering arrays:

\[ \text{CAN}(2,k,g) \geq \min \{ t : h_g(t) \geq k \}. \]

Some bounds are recursively defined. If we increase the number of levels for any factor, without loss of generality say \( g_0 \), the array size must grow or stay the same:

\[ \text{MCAN}(t, (g_0 + 1, \ldots, g_k)) \geq \text{MCAN}(t, (g_0, \ldots, g_k-1)). \]

Dropping a factor from the problem, without loss of generality say factor 0, does not increase array size:

\[ \text{MCAN}(t, (g_0, g_1, \ldots, g_k)) \geq \text{MCAN}(t, (g_1, \ldots, g_k)). \]

If, for a \( \text{CA}(N; t, k, g) A \), we consider any column \( i \) and any level \( 0 \leq x < g \), the \( N' \times (k - 1) \) subarray resulting from deleting column \( i \) from \( A \) and taking only the rows of \( A \) that had symbol \( x \) in column \( i \), the result is a \( \text{CA}(N'; t - 1, k - 1, g) \). We can always pick an \( x \) such that the number of rows kept is at most \( \lfloor N/g \rfloor \), thus giving the bound:

\[ \text{CAN}(t - 1, k - 1, g) \leq \frac{1}{g} \text{CAN}(t, k, g). \]

This translates into the following for the mixed-level case:

\[ \text{MCAN}(t, (g_0, g_1, \ldots, g_k)) \geq g_0 \text{MCAN}(t - 1, (g_1, \ldots, g_k)). \]

In [27], the authors detail a technique called fusion that permits reducing the number of levels by one in a covering array while allowing two rows to be dropped; this is a generalization from [26].
Theorem 2.2.15 (Fusion, [27]).

\[ \text{CAN}(t, k, g - 1) \leq \text{CAN}(t, k, g) - 2. \]

Proof. Let \( M \) be a \( \text{CA}(N; t, k, g) \) with entries from \( \{0, \ldots, g - 1\} \). For each column of \( M \), permute the levels to obtain a constant row \((g - 1, g - 1, \ldots, g - 1)\). We then can drop this row from the array. Choose any other row \( r = (\sigma_0, \ldots, \sigma_{k-1}) \) of \( M \), and, in column \( i \) for \( 0 \leq i \leq k - 1 \), replace all instances of \( g - 1 \) by \( \sigma_i \) if \( \sigma_i \neq g - 1 \), and by 0 otherwise. We then have that all of the \( t \)-tuples not involving \( g - 1 \) that were covered by row \( r \) are now covered by other rows of the modified array, so we can also drop \( r \) from the array. The new array has entries from \( \{0, \ldots, g - 2\} \), and is a \( \text{CA}(N - 2; t, k, g - 1) \). \qed

In the same paper, the authors provide a technique to add a factor to a covering array.

Theorem 2.2.16 (Factor increase, [27]).

\[ \text{CAN}(t, k + 1, g) \leq \text{CAN}(t, k, g) + (g - 1)\text{CAN}(t - 2, k - 1, g). \]

Proof. Define the following ingredients:

- Let \( A = [a_0, \ldots, a_{k-1}] \) be a \( \text{CA}(N_A; t, k, g) \), and
- Let \( B = [b_0, \ldots, b_{k-2}] \) be a \( \text{CA}(N_B; t - 2, k - 1, g) \).

We then form:

- An \( N_A \times (k + 1) \) matrix \( M = [a_0, \ldots, a_{k-1}, a_{k-1}] \), and
- For each \( 0 \leq u < v \leq g - 1 \), a \( N_B \times (k + 1) \) matrix \( B_{u,v} = [b_0, \ldots, b_{k-2}, \vec{u}, \vec{v}] \), where \( \vec{u} \) (resp. \( \vec{v} \)) is the length-\( N_B \) constant column vector with all entries \( u \) (resp. \( v \)).

Then the vertical concatenation of \( M \) with the \( g(g - 1) \) different \( B_{u,v} \) is a \( \text{CA}(N_A + g(g - 1)N_B; t, k + 1, g) \). \qed

As shown by the Lovász Local Lemma, a randomly generated \( N \times k \) array with entries chosen with uniform probability from \( \{0, \ldots, g - 1\} \) has a non-zero probability of being a \( \text{CA}(N; t, k, g) \) when \( N \) is large enough with respect to \( t, k, \) and \( g \): Godbole, Skipper, and Sunley [46] establish bounds using this result for \( t \) and \( v \) fixed:

\[ \text{CAN}(t, k, g) \leq \frac{(t - 1) \log k}{\log g^t}(1 + o(1)). \]
Let us consider $t$ and $g$ fixed. In [19, 20], Cohen et al. prove that greedy techniques can find covering arrays of size $O(\log k)$; their algorithm, however, relies on the solution of an NP-hard subproblem. They then give the greedy AETG algorithm which is polynomial time via the use of a heuristic to solve the NP-hard subproblem, but does not guarantee $O(\log k)$ size. Through a modification of the algorithm, using the concept of density as detailed in [10, 11, 32], in polynomial time, covering arrays can be found of size:

$$\text{CAN}(t, k, g) \leq \frac{\log \binom{k}{t} + t \log g}{\log g^t - 1}.$$ 

We extend the density algorithm to find variable strength covering arrays in Chapter 5 and find a general bound for VCA in this way.

### 2.2.2 Direct and Algebraic Constructions

There are several direct constructions for covering arrays and orthogonal arrays. We detail several of the constructions of importance to this work here.

**Definition 2.2.17.** A *latin square of order* $n$, written $\text{LS}(n)$, is an $n \times n$ square with entries from $N = \{0, \ldots, n-1\}$ such that every row and column is a permutation in the symmetric group $S_N$. Two $\text{LS}(n)$ are said to be *orthogonal* if their superposition contains all elements from $N \times N$. A collection of $k$ pairwise mutually orthogonal latin squares is referred to as a set of $k$ MOLS($n$).

**Theorem 2.2.18 ([86]).** A pair of orthogonal latin squares of order $n$ exist unless $n = 2, 6$.

**Theorem 2.2.19 (MOLS construction for orthogonal arrays [86]).** We have that:

1. An $OA(2, k, g)$ is equivalent to a set of $k - 2$ MOLS($g$).

2. For $g \neq 2, 6$, there exists an $OA(2, 4, g)$.

The construction of Theorem 2.2.19 has ties to BIBD (balanced incomplete block designs) and affine planes; details can be found in [86].

We now detail a simple construction for $OA(2, q + 1, q)$ for $q$ any prime power using finite fields. In Chapter 4, we provide an extension of this construction to variable strength orthogonal arrays.

**Theorem 2.2.20 ([86]).** For any prime power $q$, there is an $OA(2, q + 1, q)$. 
Proof. If $q = p^e$ for $p$ prime, there is a finite field $\mathbb{F}_q = \{a_1, \ldots, a_q\}$, and we can use it to construct an OA$(2, q, q)$ as follows: let $v_1 = (1, \ldots, 1)$ and $v_2 = (a_1, \ldots, a_q)$. We index the $q^2$ rows of the orthogonal array by $(i, j)$ for $i, j \in \mathbb{F}_q$. Then take row $(i, j)$ to be $i \cdot v_1 + j \cdot v_2$. This can be extended to an OA$(2, q + 1, q)$ by adjoining a column with entry $j$ for row $(i, j)$.

Another simple but useful construction for orthogonal arrays called the Zero-Sum construction is given in [33].

Lemma 2.2.21 (Zero-Sum construction, see [33, III.7.16]). For $t \geq 2$, $g \geq 1$, an OA$(t, t + 1, g)$ exists.

Proof. Let $(r_{i,0}, \ldots, r_{i,t-1})$ be the $i$th $t$-tuple of $\mathbb{Z}^t_g$ according to the lexicographic ordering. Then create a $g^t \times (t + 1)$ array where row $i$ consists of, for columns $0, \ldots, t - 1$, the values $r_{i,0}, \ldots, r_{i,t-1}$, and for column $t$, the value $-(r_{i,0} + \ldots + r_{i,t-1})$.

Additionally, the Bush construction gives a technique using polynomials of degree at least one over finite fields to generate orthogonal arrays. Details of the construction can be found in [14].

Lemma 2.2.22 (Bush construction, see [14, Section 3]). Let $q = p^n$ be a prime power. For any $t < q$, an OA$(t, q + 1, q)$ exists.

The problem of determining CAN$(2, k, 2)$ is fully solved for all values of $k$. A construction for optimal CA$(N; 2, k, 2)$ is given by Kleitman and Spencer [54], and also by Katona [53], and gives that in $N$ tests, we can accommodate the following number of parameters:

$$k = \left(\begin{array}{c} N - 1 \\ \left\lceil \frac{N}{2} \right\rceil \end{array}\right).$$

Using this equation, we have:

$$\text{CAN}(2, k, 2) = \min \left\{ N : \left(\frac{N}{\left\lceil \frac{N-1}{2} \right\rceil}\right) \geq k \right\}.$$ 

This yields that CAN$(2, k, 2)$ is approximately $\log k$ as $k$ approaches infinity. The columns of the array achieving this bound are all distinct binary $N$-tuples of weight $\left\lceil \frac{N}{2} \right\rceil$ with a 0 in the first position.

A $(g, l)$-cover starter is a vector $v = (\infty, v_1, \ldots, v_{l-1})$ with $v_i \in \mathbb{Z}_{g-1}$ such that for any $s$, the differences of elements at positional difference $s$ cover all of $\mathbb{Z}_{g-1}$. Meagher and Stevens [69] show that a cover starter can be developed into a CA$(l(g - 1) + 1; 2, l, g)$. 

For $t = 3$, the following approach is often taken in constructing covering arrays: let $\Sigma$ be a set with $g = |\Sigma|$, and $C = C(k, \Sigma)$ be the $k \times g$ matrix having constant columns with entries from $\Sigma$. If $M = (m_{i,j})$ is a matrix over $\Sigma$ and $H = \{h_1, \ldots, h_n\}$ is a group acting on $\Sigma$, then define $M^h = (h(m_{i,j}))$ and $[M^H] = [M^{h_1}] \ldots [M^{h_n}]$. To find a covering array, the general idea is to determine an $k \times l$ matrix $M$ and a group $H$ acting on $\Sigma$ such that either $[M^H, C]^T$ or $[M^H]^T$ is a CA($N; 3, k, g$). Using this technique, Colbourn [30] shows that for $g > 2$, $q \geq g - 1$ a prime power, there is a $CA((2^g - 1)(q^3 - q) + g; 3, 2g, g)$.

### 2.2.3 Recursive Constructions

Many of the constructions in this section are outlined in greater detail in [30].

One family of constructions of particular interest in this thesis originates from the following theorem of Roux’s PhD thesis [80]:

$$CAN(3, 2k, 2) \leq CAN(3, k, 2) + CAN(2, k, 2).$$

This has led to a number of Roux-type constructions based on similar ideas, which we detail next for strength $t = 2$ and strength $t = 3$. Roux-type constructions for higher strength are given in [34, 63, 82]; we do not explicitly examine them here.

**Roux-Type Constructions for $t = 2$**

If we have a CA($N; 2, k, g$) $A$ and a CA($M; 2, l, g$) $B = [B_1 \ldots B_l]$, we can get a CA($N + M; 2, kl, g$) by tiling them:

$$\begin{bmatrix}
A & A & \cdots & A \\
B_1 \ldots B_1 & B_2 \ldots B_2 & \cdots & B_1 \ldots B_1 \\
\text{k copies} & \text{k copies} & \cdots & \text{k copies}
\end{bmatrix}.$$

Given a 2-set of factors, say $v_i$ and $v_j$ with $0 \leq i < j \leq kl - 1$, we have two cases:

1. If there is some integer $s$ such that $j = i + sk$, then the columns from the rows generated by tiling $A$ do not cover the pairs between $v_i$ and $v_j$, as they will have the same column in the tilings of $A$. However, they are then covered by the interleaving of the columns of $B$.

2. Otherwise, the tiling of copies of $A$ will cover the pairs between $v_i$ and $v_j$. 
Some of the entries may not be needed to cover all pairs; we can replace these with a *. Define profile\((d_1, \ldots, d_k)\) to be the \(k\)-tuple with \(d_i\) being the number of * entries in column \(i\).

If we have:

- a MCA\((N; 2, (g_1, \ldots, g_k))\) \(A\) with profile\((d_1, \ldots, d_k)\), and
- for \(i = 1, \ldots, k\), a MCA\((M_i; 2, (w_{i,1}, \ldots, w_{i, l_i}))\) with profile\((f_{i,1}, \ldots, f_{i, l_i})\),

then for \(T = N + \max\{M_i - d_i : i = 1, \ldots, k\}\), there is a:

\[
\text{MCA}(T; 2, \sum_{i=1}^{l_i} l_i, (w_{i,j}))
\]

with \((w_{i,j})\) representing all alphabet sizes from the \(M_i\).

**Roux-Type Constructions for Strength \(t = 3\)**

Chateauneuf and Kreher [15] generalized an idea presented by Roux [80] to show the following theorem:

**Theorem 2.2.23 ([15]).**

\[
\text{CAN}(3, 2k, g) \leq \text{CAN}(3, k, g) + (g - 1) \text{CAN}(2, k, g).
\]

**Proof.** If \(A\) is a CA\((N; 3, k, g)\), \(B\) is a CA\((M; 2, k, g)\), and \(\pi = (0 \ldots g - 1)\), then the following is a CA\((N + (g - 1)M; 3, 2k, g)\):

\[
\begin{bmatrix}
A & A \\
B & \pi(B) \\
\vdots & \vdots \\
B & \pi^{g-1}(B)
\end{bmatrix},
\]

where \(\pi(B)\) is the matrix \(B\) with its entries permuted by \(\pi\). \(\square\)

This idea was further refined to the \(k\)-ary Roux construction by using difference covering arrays (DCA). DCA and their applications to covering arrays are studied by Yin in [92, 93].

**Definition 2.2.24.** A difference covering array\ DCA\((N; 2, k, g)\) is an \(N \times k\) array with entries \((d_{ij})\) from \(\mathbb{Z}_g\) such that for any two distinct columns \(j\) and \(l\), the set \(c_{jl} = \{d_{ij} - d_{il} : i \in \{1, \ldots, N\}\}\) contains \(\mathbb{Z}_g \setminus \{0\}\). A DCAN\((2, k, g)\) is the minimum \(N\) for which a DCA\((N; 2, k, g)\) exists.
Difference covering arrays can be used to perform the following construction:

**Theorem 2.2.25** \cite{24}.

\[ \text{CAN}(3, kl, g) \leq \text{CAN}(3, l, g) + \text{CAN}(3, k, g) + \text{CAN}(2, l, g) \times \text{DCAN}(2, k, g). \]

**Proof.** If we have a CA\((N; 3, l, g)\) \(A\), a CA\((M; 3, k, g)\) \(B = [B_1 \ldots B_k]\), a CA\((R; 2, l, g)\) \(F\), and a DCA\((Q; 2, k, g)\) \(D = (d_{ij})\), with all arrays having entries from \(\mathbb{Z}_g\), then the following is a CA\((N + M + RQ; 3, kl, g)\):

\[
\begin{bmatrix}
A & A & \ldots & A \\
B_1 \ldots B_1 & B_2 \ldots B_2 & \ldots & B_k \ldots B_k \\
\text{\(t\) copies} & \text{\(t\) copies} & \ldots & \text{\(t\) copies} \\
F + d_{11} & F + d_{12} & \ldots & F + d_{1k} \\
\vdots & \vdots & \ddots & \vdots \\
F + d_{Q1} & F + d_{Q2} & \ldots & F + d_{Qk}
\end{bmatrix}.
\]

\(\square\)

The next theorem, as shown in detail in \cite{35}, follows from **Theorem 2.2.25** when the number of levels is a prime power \(q \geq 3\). In this case, it is possible to find a difference covering array where, for any two columns, no differences are repeated in the difference set. This is called a *difference matrix*, which has size \(q - 1\). Furthermore, the CAN\((3, k, g)\) can be replaced with an OA\((3, q, q)\) created by the Bush construction, from which it is possible to further drop \(q^2\) rows, yielding:

**Theorem 2.2.26** \cite{35}. For \(q \geq 3\) a prime power:

\[ \text{CAN}(3, qk, q) \leq \text{CAN}(3, k, g) + (q - 1)\text{CAN}(2, k, q) + q^3 - q^2. \]

Colbourn \cite{29} improves on the earlier result of \cite{24} by using resolvable covering arrays as ingredients in the construction.

**Definition 2.2.27.** A covering ordered design COD\((N; t, k, g)\) is an \(N \times k\) array over \(S = \{0, \ldots, g - 1\}\) with the property that every \(N \times t\) subarray contains all \(t\)-tuples from:

\[ S^t \setminus \{(i, \ldots, i) : i \in S\} \]

at least once. We write CODN\((t, k, g)\) to denote the minimum \(N\) for which a COD\((N; t, k, g)\) exists.
Definition 2.2.28. If $C$ is an $M \times k$ array over $g$ symbols, for an integer $1 \leq N \leq M$ and integers $M_1, \ldots, M_N$ such that $M = \sum_{i=1}^{N} M_i$, a resolution of $C$ with $N$ classes is a partition of $C$ into $N$ subarrays $C_1, \ldots, C_N$ such that for all $1 \leq i \leq N$, we have that $C_i$ is an $M_i \times k$ array where every symbol occurs at most once in each column.

Definition 2.2.29. A resolvable covering ordered design $RCOD(N; t, k, g)$ is a covering order design $COD(M; t, k, g)$ that has a resolution with $N$ classes. We write $RCODN(t, k, g)$ to denote the smallest $N$ for which an $RCOD(N; t, k, g)$ exists.

We then have the following theorem:

Theorem 2.2.30. When $\min(k, l) \geq 3$:

\[ CAN(3, kl, g) \leq CAN(3, k, g) + CAN(3, l, g) + CODN(2, l, g) \times RCODN(2, k, g). \]

The proof is similar to the proof of Theorem 2.2.25, as the COD and the RCOD satisfy the necessary properties exhibited by the ingredients in the earlier construction.

Other Recursive Constructions

Another recursive construction that depends only on the existence of a strength-3 covering array is presented by Hartman and Raskin in [50] as Theorem 3.8, where $T(v, n)$ represent the Turán numbers. These are the numbers of edges in a complete $n$-partite graph with $v$ vertices where the vertex partition classes are almost-uniform; i.e. if $a = \left\lfloor \frac{v}{n} \right\rfloor$, then there are $v - na$ partition classes of size $a + 1$, and $n(1 + a) - v$ partition classes of size $a$. We include a modification by Colbourn [31], which improves upon the original bound of Hartman and Raskin by using perfect hash families to reduce the array size by $T(v, n)$ rows.

Theorem 2.2.31. If there exists both a $CA(N; t, k, g)$ and a set of $T(t, g) - 1$ mutually orthogonal latin squares of order $k$, then:

\[ CAN(t, k^2, g) \leq (T(t, g) + 1)(N - 1). \]

This bound leads to the following corollary, which we can use to recursively generate larger covering arrays:

Corollary 2.2.32. If $CAN(3, k, g) \leq N_g$, and either $g = 2$, or $g \geq 3$ and $k \not\in \{2, 6\}$, then for any $i \geq 2$:

\[ CAN(3, k^{2^i}, g) \leq \begin{cases} 3^i N_g - \frac{3^{i+1} - 1}{2} & \text{if } g = 2 \\ 4^i N_g - \frac{4^{i+1} - 1}{3} & \text{if } g \geq 3. \end{cases} \]
Proof. For the case \( g = 2 \), we have that \( T(t, 2) = \lfloor \frac{t^2}{4} \rfloor \), so \( T(3, 2) = \lfloor \frac{3^2}{4} \rfloor = 2 \). By iterating the recursive construction in Theorem 2.2.31 \( i \) times, we get the desired result.

When \( g \geq t \), we have that \( T(t, g) = \frac{t(t-1)}{2} \), so for \( g \geq 3 \), \( T(3, g) = 3 \). From Theorem 2.2.18, there are always a pair of mutually orthogonal Latin squares of order \( k \) for \( k \notin \{2, 6\} \). We can then apply Theorem 2.2.31 recursively \( i \) times to get the desired result.

### 2.2.4 Covering Arrays on Graphs

Consider a VCA(\( N; G, g \)) where \( G \) is a graph. Meagher and Stevens introduce and study these covering arrays on graphs in [66, 68], and Meagher, Moura, and Zekaoui generalize to the mixed level case in [67]. We summarize some of their results here.

Using properties of graph homomorphisms, we have that:

\[
\text{CAN}(2, \omega(G), g) \leq \text{VCAN}(G, g) \leq \text{CAN}(2, \chi(G), g)
\]

where \( \omega(G) \) is the clique number, or size of the largest 2-clique in \( G \), and \( \chi(G) \) is the chromatic number, or smallest number such that there is a homomorphism \( G \to K_{\chi(G)} \). This bound is generalized to mixed-level covering arrays in [67]. There are strong connections between covering arrays on graphs, graph cores, and graph homomorphisms.

Two vectors \( v = \{v_0, \ldots, v_{n-1}\} \) and \( u = \{u_0, \ldots, u_{n-1}\} \) over \( \mathbb{Z}_v \) are qualitatively independent if \( \{(v_0, u_0), (v_1, u_1), \ldots, (v_{n-1}, u_{n-1})\} \) contains every ordered pair of \( \mathbb{Z}_v \times \mathbb{Z}_v \). A qualitative independence graph QI(\( N, g \)), \( N \geq g^2 \), is a graph with one vertex representing each vector in \( \mathbb{Z}_v^N \). Two vertices are adjacent if their corresponding vectors are qualitatively independent. Then a VCA(\( N; G, g \)) exists if and only if there is a graph homomorphism \( G \to \text{QI}(N, g) \).

The colouring bound given above is not always tight: consider \( G = \text{QI}(5, 2) \), where clearly we have that \( \text{CAN}(G, 2) \leq 5 \). We have that \( \chi(G) = 5 \), but \( \text{CAN}(2, 5, 2) = \text{CAN}(K_5, 2) = 6 \).

In the case \( g = 2 \), the exact values of \( \omega(\text{QI}(N, 2)) \) and \( \chi(\text{QI}(N, 2)) \) are known, as are the cores of the graphs, which give that:

\[
\text{CAN}(2, \chi(G), 2) - 1 \leq \text{VCAN}(G, 2) \leq \text{CAN}(2, \chi(G), 2)
\]

Several necessary conditions for strict inequality in the right-hand side are presented, and many such graphs are known, as detailed in [65].

For \( g > 2 \), the known bounds are not always tight: \( \text{QI}(9, 3) \) is 6-chromatic, but \( \text{CAN}(2, 6, 3) = 12 \).
In [67], for the mixed alphabet case, bounds for graphs of low chromatic number \((\chi(G) \in \{2, 3, 4\})\) are given. Both Meagher et al. [67] and Cheng [17] determine the covering array number in the case of \(G\) being a tree, cycle, or bipartite graph, showing that the lower bound as given in Equation 2.2 can be achieved.

We generalize the concept of qualitative independence to higher strengths, and examine the resultant qualitative independence hypergraphs and their ties to variable strength covering arrays in Chapter 3.

### 2.2.5 Variable Strength Covering Arrays

Besides the existing results focusing on covering arrays on graphs, only a few other studies have been done on variable strength covering arrays.

Cohen et al. [22] study a specific family of ASC, which consists of a base strength over all factors, and then disjoint regions of higher strength. More specifically, their ASC have the following form: given a set of points \(V\), a base strength \(t\), and \(V_1, \ldots, V_s \subseteq V\) such that \(V_i \cap V_j = \emptyset\) for all \(1 \leq i < j \leq s\), with strengths \(t_1, \ldots, t_s\), \(t < t_i \leq |V_i|\) for \(1 \leq i \leq s\), then define the ASC:

\[
\Delta = \Delta \left( \left( V_t \right) \cup \left( \left( V_1 \atop t_1 \right) \cup \ldots \cup \left( V_s \atop t_s \right) \right) \right)
\]

They then give a simulated annealing algorithm to find VCA over this restricted family of ASC, and experimentally determine results for several of these ASC.

Cheng et al. [18] propose a problem reduction technique based on proper colourings to be used in conjunction with greedy algorithms to find VCA over arbitrary ASC.

In [17], Cheng provides several results regarding the sizes of covering arrays for several different families of ASC. She proves that if \(H\) is a hypertree, then the lower bound of Equation 2.2 is achievable.

Analogous to graphs, a hypergraph is said to be \(q\)-partite if \(V\) can be partitioned into \(q\) sets so that no edge intersects any one of the sets in more than one vertex. Cheng shows that, for a \(q\)-partite hypergraph \(H\) with maximum degree \(t \leq q\), if an \(OA(t, q, g)\) exists, then \(VCA(H, g) = g^t\).

Cheng also uses ordered orthogonal arrays to find VCA where the factors are all powers of a common positive integer.

**Definition 2.2.33.** An ordered orthogonal array \(OOA_\lambda(t, k, l, g)\) to be a \(\lambda g^t \times kl\) array \(M\) with entries from \(G = \{0, \ldots, g - 1\}\) if the columns of \(M\) can be partitioned into
k groups of $l$ columns each, denoted $C_0, \ldots, C_{k-1}$ with $C_i = \{c_{ij} : 0 \leq j \leq l - 1\}$, so that the following condition holds: whenever $t = \sum_{i=0}^{k-1} t_i$ where each $t_i$ is a nonnegative integer such that $t_i \leq l$, then the $t \times \lambda g^{t}$ subarray of $M$ formed by taking the first $t_i$ columns of $C_i$ for $i = 0, \ldots, k - 1$ contains each $t$-tuple over $G$ as a row precisely $\lambda$ times. As with similar definitions, we omit $\lambda$ from the notation when $\lambda = 1$.

Let $H$ be a $q$-partite hypergraph. If there exists a positive integer $m$ such that for all $0 \leq i \leq k - 1$, $g_i$ is a power of $m$ and an OOA($t, k, t, q$) exists, then:

$$\text{VCAN}(H, (g_0, \ldots, g_{k-1})) = \max \left\{ \prod_{i \in e} g_i : e \in H \right\}.$$

Cheng also shows that when the number of levels across all factors are all pairwise relatively prime, regardless of the hypergraph, we can construct VCA that reach the lower bound in Equation 2.2 in time $O(km)$ for $m = \max\{\prod_{i \in e} g_i : e \in H\}$. She also states that her proof works for hypergraphs $H$ where, for each edge $e \in H$, the number of levels for each of the factors in $e$ are relatively prime.

More recently, some other research has begun to emerge on variable strength covering arrays. Maity [60] establishes some connections between variable strength covering arrays and homomorphisms, and claims to find VCAN($H, g$) when $H$ is any 3-uniform hypergraph. Unfortunately, the proof is incomplete; we are currently collaborating to attempt to find a full solution to this problem.

Wang, Xu, and Nie independently examine the problem of using density-based greedy algorithms to find variable strength covering arrays in [87]. Unlike our VAR_DENS algorithm in Chapter 5, they do not use their algorithm to establish bounds on the size of the array generated by the algorithm, but they do explore additional modifications and compare the arrays from their package to those generated through other means.

More recently, some research has emerged into using swarm intelligence methods to find variable strength covering arrays; these algorithms, however, focus on the restricted definition of a VCA as given by Cohen et al. in [22]. Particle swarm based optimization strategies are examined by Ahmed and Zamli in [2, 3], and Ahmed, Zamli, and Lim in [4]. Chen, Gu, and Chen look at using an ant colony approach in [16].

### 2.2.6 Covering Arrays with Forbidden Edges

Another problem of interest that is currently receiving attention is that of covering arrays where certain interactions are prohibited. These forbidden interactions are modeled using
a $k$-partite graph over $v_0 + \ldots + v_{k-1}$ vertices, where $k$ is the number of factors, with factor $i$ having $v_i$ levels for $0 \leq i \leq k - 1$. These are called covering arrays with forbidden edges, and are discussed in [38, 61, 62].

Given our study of general relational systems in Section 2.1.1 and our definition of variable strength covering arrays, we propose the following possible line of research:

**Open Question 2.2.1.** Is it possible to use general relational systems instead of ASC to extend the concept of variable strength covering arrays to include forbidden edges?

### 2.2.7 Consecutive Covering Arrays

Godbole, Koutras, and Milienos [45] introduce the concept of a consecutive covering array, where the structure of the columns captures some linear progression of data (for example, data across a series of consecutive dates), or data organized by consecutive proximity (for example, consecutive switches in a circuit).

**Definition 2.2.34.** Let $H_{k,t} = (V, E)$ be the hypergraph with $V = \{0, \ldots, k - 1\}$ and:

$$E = \{\{i, i+1, \ldots, i+t-1\} : 0 \leq i \leq k - t\}.$$

A consecutive covering array $\text{CCA}(N; t, k, g)$ is a $\text{VCA}(N; H_{k,t}, g)$, and thus:

$$\text{CCAN}(t, k, g) = \text{VCAN}(H_{k,t}, g).$$

In [45], Godbole et al. focus on binary arrays of the form $\text{CCA}(N; t, k, 2)$. They produce, for a value $n$, an $n \times k$ array randomly with values from $\{0, 1\}$ by treating the problem as $kn$ Bernoulli trials where, for some $0 < p < 1$, a variable assumes value 1 with probability $p$ and value 0 with probability $1 - p$. They then determine the probability distribution function of a random variable that enumerates the number of uncovered consecutive $t$-subarrays by using a Markov chain embedding technique. In [44], they focus on the more general problem of $\text{CCA}(N; t, k, g)$, again establishing the probability distribution function of the random variable enumerating the uncovered consecutive $t$-subarrays with the Markov chain method, and extend this by examining upper bounds on the total variation error.

We propose another similar family of covering arrays, which we call cyclic consecutive covering arrays, where the columns are considered to be cyclically ordered.

**Definition 2.2.35.** Let $H_{c}^{k,t} = (V, E)$ be the hypergraph with $V = \{0, \ldots, k - 1\}$ and:

$$E = \{\{i, (i+1) \mod k, \ldots, (i+t-1) \mod k\} : 0 \leq i \leq k - 1\}.$$
A cyclic consecutive covering array \( cCCA(N; t, k, g) \) is a \( VCA(N; H_{c}^{k,t}, g) \), and thus:

\[
cCCAN(t, k, g) = VCAN(H_{c}^{k,t}, g).
\]

We give a homomorphic shrinking property of these graphs in Theorem 2.2.36.

**Theorem 2.2.36.** For \( k \geq 2t \), \( H_{c}^{k,t} \to H_{c}^{k-t,t} \).

**Proof.** Write \( E_{j} = \{ j, (j + 1) \mod k, \ldots, (j + t - 1) \mod k \} \) for \( 0 \leq j < k \) to uniquely identify all of the \( k \) edges of \( H_{c}^{k,t} \). Consider the map \( f: \{0, \ldots, k-1\} \to \{0, \ldots, k-t-1\} \) defined by \( i \mapsto i \mod (k - t) \). We then consider three separate subsets of edges:

1. For \( 0 \leq j \leq k - 2t \), we have that \( f(E_{j}) = E_{j} \in E(H_{c}^{k-t,t}) \).
2. For \( k - 2t < j < k - t + 1 \), we have that \( j + t - 1 < k \), so \( f(E_{j}) \in E(H_{c}^{k-t,t}) \).
3. For \( k - t + 1 \leq j < k \), we can write \( j = k - i \) for some integer \( i > 0 \).

\[
f(E_{j}) = \{ f(j), \ldots, f(j + t - 1) \mod k \} = \{ f(j), \ldots, f(j + i - 1), f(j + i), \ldots, f(j + t - 1) \} = \{ f(j), \ldots, f(k - 1), f(0), \ldots, f((t - i) - 1) \}
\]

We now have that \( f(j) = j \mod (k - t) = j - (k - t) = (k - i) - (k - t) = t - i \) and \( f(k - 1) = f(j + i - 1) = j + i - 1 \mod (k - t) = j + i - 1 - (k - t) = (k - i) + i - 1 - (k - t) = t - 1 \).

\[
f(E_{j}) = \{ t - i, t - i + 1, \ldots, t - 1, 0, 1, \ldots, (t - i) - 1 \} = \{0, \ldots, t - 1\}.
\]

This edge is always in \( E(H_{c}^{k-t,t}) \).

Thus, \( f \) is a homomorphism.

Corollary 2.2.37 follows immediately from the recursive application of Theorem 2.2.36.

**Corollary 2.2.37.** \( H_{c}^{k,t} \to H_{c}^{t+(k \mod t),t} \).

**Proof.** We iterate Theorem 2.2.36 until we have \( t \leq k' < 2t \), i.e. \( k' = t + (k \mod t) \).

We investigate the problem of constructing optimal CCA and cCCA in Section 3.4.3.
2.2.8  $k$-Trees

Trees are a well-studied family of bipartite graphs. Another family of graphs that we will consider are the $k$-trees, which are generalizations of trees. The following definition comes from [52].

**Definition 2.2.38.** A graph $G$ is a $k$-tree if $G$ can be constructed, beginning with a $K_k$, through a sequence of vertex additions. The idea is as follows:

1. Begin with a $K_k$.

2. For each new vertex $v$, select a $k$-clique $C$ in the existing graph, and add $v$ and $k$ edges from $v$ to all vertices in $C$ to form a $(k+1)$-clique containing $v$.

We give two examples of 2-trees in Figure 2.4. One important property of $k$-trees is that the initial choice of $K_k$ is irrelevant; any such $K_k$ in the $k$-tree may serve as a starting point for the iterative construction.

![Two examples of possible 2-trees](image)

Figure 2.4: Two examples of possible 2-trees. The dotted lines represent edges that could never be added.

Clearly, if $G$ is a $k$-tree, then $|V(G)| \geq k$. We have the following result regarding the colourability of $k$-trees:

**Theorem 2.2.39.** If $G$ is a $k$-tree, then:

$$
\chi(G) = \begin{cases} 
  k & \text{if } |V(G)| = k \\
  k + 1 & \text{if } |V(G)| > k.
\end{cases}
$$
Proof. If $|V(G)| = k$, then $G$ must be $K_k$, and $\chi(K_k) = k$. Otherwise, we construct a colouring map $f : V(G) \rightarrow \{0, \ldots, k\}$ by specifying how to colour each vertex during the construction of $G$ as follows:

1. For the initial $K_k$, set $f(i) = i$ for $0 \leq i \leq k - 1$.

2. When adding $v$ to the $k$-clique $C$, set $f(v)$ to the only colour in $\{0, \ldots, k\} \setminus f(C)$.

We then have that $f$ is a homomorphism $G \rightarrow K_{k+1}$, as required, so $\chi(G) \leq k + 1$. Furthermore, since $G$ contains $(k+1)$-cliques, which are isomorphic to $K_{k+1}$, $\chi(G) \geq k+1$, so $\chi(G) = k + 1$.

For a $k$-tree $G$ with $|V(G)| > k$, we can associate a hypergraph with $G$ by replacing each $K_{k+1}$ by a hyperedge of size $k + 1$. Then the primal graph of this hypergraph is the original $k$-tree.

Definition 2.2.40. A $k$-tree hypergraph is a hypergraph $H$ built as follows:

1. Initially set $H$ to contain precisely vertices $0, \ldots, k$ and hyperedge $\{0, \ldots, k\}$.

2. For each new vertex $v$, select a hyperedge $E$ in $H$. Pick any vertex $u \in E$ and set $E' = (E \setminus \{u\}) \cup \{v\}$. Then add $v$ and $E'$ to $H$.

If $H$ is a $k$-tree hypergraph, then $[H]_2$ is a $k$-tree. Thus, the strong colouring number of a $k$-tree hypergraph is a direct corollary of Theorem 2.2.39.

Corollary 2.2.41. If $G$ is a $k$-tree hypergraph, then $\gamma(G) = k + 1$.

As discussed in Section 2.2.4, constructions for optimal covering arrays over bipartite graphs have been established. The problem of finding a construction for optimal covering arrays over arbitrary, strongly 3-colourable hypergraphs appears to be a much more difficult problem. The 2-tree hypergraphs are a subset of the strongly 3-colourable hypergraphs, and given their iterative construction, building optimal arrays over them appears to be considerably more simple. We examine covering arrays on 2-tree hypergraphs in Section 3.4.4.

2.2.9 Triangulations of the Sphere

Definition 2.2.42. A triangulation hypergraph of the sphere is an abstract simplicial complex $\Delta$ with facet set $\Lambda$ such that:
1. $\Lambda \subseteq \binom{V(\Delta)}{3}$,

2. For any $a, b \in V(\Delta)$, $a \neq b$, we have:
   $$|\{e \in \Lambda : \{a, b\} \subset e\}| \in \{0, 2\}.$$

3. $\Delta$ can be embedded on a sphere such that no edges of $\text{skel}_1(\Delta)$ cross.

Bowen and Fisk [9] discuss techniques to generate all (isomorph-free) triangulations of the sphere with $v$ vertices using structural induction and building up the triangulation vertex by vertex: for $v \in \{4, 5\}$, there is precisely one triangulation up to isomorphism. We can then apply one of three recursive operations to add a vertex to the triangulation, as shown in Figure 2.5, and via these operations, we can build every possible triangulation of the sphere.

**Theorem 2.2.43** (See [9]). Any triangulation hypergraph of the sphere can be built recursively by beginning with hypergraph $\Delta^0 = \Delta(\{wxy, wxyz, wzy, xyz\})$, which is itself the smallest valid triangulation. Given any valid triangulation hypergraph $\Delta^i$, perform any one of the following operations:

**O1:** Given a triangle $xyz \in \Delta^i$, add a vertex $u$ and set:
   $$\Delta^{i+1} = (\Delta^i \setminus \Delta(\{xyz\})) \cup \Delta(\{uxy, uxz, uyz\}).$$

**O2:** Given a pair $xy \in V(\Delta^i)$ such that there are two triangles, $wxy, xyz \in \Delta^i$, add a vertex $u$ and set:
   $$\Delta^{i+1} = (\Delta^i \setminus \Delta(\{wxy, xyz\})) \cup \Delta(\{uwx, uwy, uxz, uyz\}).$$

**O3:** Given a triangle $xyz \in \Delta^i$ such that there are two other triangles, $vxz, wxy \in \Delta^i$, add a vertex $u$ and set:
   $$\Delta^{i+1} = (\Delta^i \setminus \Delta(\{vxz, wxy, xyz\})) \cup \Delta(\{uvx, uvz, uwx, uwz, uyz\}).$$

Then $\Delta^{i+1}$ is a valid triangulation hypergraph of the sphere.

**Example 2.2.44.** A triangulation of the sphere $\Delta$ with 7 vertices (and dimension-0 faces), 10 dimension-2 facets, and 15 dimension-1 faces is given in Figure 2.6. We specify facets $\{x, y, z\}$ as $xyz$ for compactness. The triangulation was created as follows via the operations of Theorem 2.2.43.
Figure 2.5: The three operations of Bowen and Fisk [9] for building all triangulations of the sphere up to isomorphism.

1. Begin with $\Delta = \{uxy, uxz, uyz, xyz\}$ and subfaces.
2. Perform O2 to add vertex $w$ to pair $uy$: $\Delta = \{uw, wzu, wxy, wyz, uxz, xyz\}$ and subfaces.
3. Perform O3 to add vertex $a$ to face $wyz$:

   $$\Delta = \{auw, auz, awx, ayz, uwx, uxz, xyz\}$$

   and subfaces.
4. Perform O2 to add vertex $b$ to pair $az$:

$$\Delta = \{auw, aub, awx, axy, aby, buz, byz, uwx, uxz, xyz\}$$

along with all subfaces.

Figure 2.6: Example of a triangulation of the sphere with 7 vertices, 10 dimension-2 facets, and 15 dimension-1 faces.

From the recursive construction, we can derive the following theorem regarding the structure of triangulation hypergraphs of the sphere.

**Theorem 2.2.45.** If $\Delta$ is a triangulation hypergraph of the sphere over $v$ vertices, then:

1. The number of triangles (dimension-2 facets) in $\Delta$ is $2v - 4$.
2. The number of dimension-1 faces in $\Delta$ is $3v - 6$.

*Proof.* We begin with the base case $\Delta^0$, which has 4 vertices, 4 facets of dimension 2, and 6 facets of dimension 1, thus satisfying the two equations. Assume we have a
triangulation hypergraph $\Delta^i$ with $v$ vertices and $2v - 4$ dimension-2 facets. Then any of the three operations O1, O2, and O3 all result in a triangulation hypergraph $\Delta^{i+1}$ with one more vertex and a net addition of two dimension-2 facets. Hence, $\Delta^{i+1}$ has $v + 1$ vertices and $2v - 4 + 2 = 2(v + 1) - 4$ dimension-2 facets, thus proving the first equation by induction.

Let $\Delta$ be a valid triangulation hypergraph of the sphere over $v$ vertices. By definition, each pair of vertices $\{a, b\} \in \Delta$ must be in precisely two facets (triangles) of $\Delta$. Each triangle of $\Delta$, say $\{x, y, z\}$, contains three dimension-1 faces, namely $\{x, y\}$, $\{x, z\}$, and $\{y, z\}$. However, each of these dimension-1 faces must also be shared with another triangle. As from the first formula, there are $2v - 4$ triangles in $\Delta$, we then have that the number of dimension-1 faces must be:

$$\frac{3(2v - 4)}{2} = \frac{6v - 12}{2} = 3v - 6.$$

We fully solve the problem of finding VCA over triangulation hypergraphs of the sphere in Section 3.5.

### 2.3 Design Theory and Links to Covering Arrays

We detail some definitions of classical designs and explore various properties of these designs, which will be useful in several contexts in the study of covering arrays. An excellent introduction to general design theory can be found in [86].

**Definition 2.3.1.** A set system $S$ is a pair $(V, B)$ such that $V$ is a set and $B$ is a collection of subsets (with repetitions permitted) of $V$. We call $V$ the set of points of the system, and $B$ the set of blocks of the system.

For $v \in V$, the replication number of $v$ is the number of blocks in which $v$ appears:

$$r_v = |\{B \in B : v \in B\}|.$$

If every point has the same replication number, say $r$, then we say that $S$ is $r$-regular.

Additionally, the variable $b$ is typically used to represent the number of blocks in a set system:

$$b = |B|.$$

A set system where $B$ is a set is called simple.
Many mathematical structures fall under the umbrella of set systems, including graphs, hypergraphs, and combinatorial designs. We can define morphisms over set systems, which generalizes and captures the behaviour of the morphisms already seen for graphs and hypergraphs in Section 2.1.

**Definition 2.3.2.** Let \( S = (V, B) \) and \( T = (W, C) \) be set systems. Then we have the following types of maps \( f : V \to W \):

- **homomorphism:** For all \( B \in B \), \( |B| = |f(B)| \) and \( f(B) \in C \).
  
  If such a homomorphism exists, we write \( S \to T \).

- **monomorphism:** \( f \) is an injective homomorphism.
  
  If such a monomorphism exists, we write \( S \hookrightarrow T \).

- **epimorphism:** \( f \) is a surjective homomorphism.
  
  If such an epimorphism exists, we write \( S \twoheadrightarrow T \).

- **isomorphism:** \( f \) is a bijective homomorphism and \( f(B) = C \).
  
  If an isomorphism exists, we write \( S \cong T \).

A homomorphism \( S \to S \) is called an *endomorphism*, and an isomorphism \( S \to S \) is called an *automorphism*. The set of all automorphisms of a set system \( S \) form a group under composition, called the *automorphism group of* \( S \), denoted \( \text{Aut}(S) \).

Due to the generality of the definition, it is not particularly interesting to study set systems in their full abstraction, and we instead define and examine set systems with additional restrictions imposed upon them. Thus, we begin by examining combinatorial designs, which are set systems in which certain balance properties over the points are enforced in the blocks of the system. We first describe one of the most general forms of a combinatorial design, called an \( s \)-design, and then identify specific subfamilies of \( s \)-designs such as BIBD, projective planes, and Steiner systems.

**Definition 2.3.3.** Let \( k > t > s > 0 \). An \( s \)-(\( k, t, \lambda \)) design is a set system \( (V, B) \) such that \( |V| = k \), and \( B \) is a multiset with elements from \( \binom{V}{t} \) such that for every \( S \in \binom{V}{s} \), there are exactly \( \lambda \) sets \( B \in B \) such that \( S \subset B \). We call \( s \) the *strength* of the design, and \( \lambda \) the *index* of the design. It is common to take \( V = \{0, \ldots, k-1\} \) and then describe the design simply by listing the block multiset \( B \).
It is more typical to call these designs $t-(v, k, \lambda)$ designs, but since we will be investigating VCA over these designs, we have renamed the parameters to coincide with the parameters of covering arrays and to avoid clashes in parameter names.

While the definition of $s-(k, t, \lambda)$ designs only prescribes specific balance behaviour over all $s$-sets of points, this extrapolates to balance properties over all $m$-sets of points, for $0 \leq m \leq s$, as illustrated in Theorem 2.3.4.

**Theorem 2.3.4.** Let $S = (V, \mathcal{B})$ be a $s-(k, t, \lambda)$ design, and $M$ be any $m$-subset of $V$, with $0 \leq m \leq s$. Then the number of blocks containing $M$ is precisely:

$$
\lambda_m = |\{B \in \mathcal{B} : M \subseteq B\}| = \frac{\lambda \binom{k-m}{s-m}}{\binom{t-m}{s-m}}.
$$

For any $s-(k, t, \lambda)$ design, this theorem can be used to calculate $r$ by taking $m = 1$, and $b$ by taking $m = 0$, as every block contains the unique 0-set, namely $\emptyset$.

**Corollary 2.3.5.** For an $s-(k, t, \lambda)$ design, we have:

$$
r = \frac{\lambda \binom{k-1}{s-1}}{\binom{t-1}{s-1}}, \quad b = \frac{\lambda \binom{k}{s}}{\binom{t}{s}}.
$$

We now define several subfamilies of $s-(k, t, \lambda)$ designs that will be of interest, and enumerate some of the properties of each; most of these properties follow immediately from Corollary 2.3.5 and observing that $r$ and $b$ must always be integers.

**Definition 2.3.6.** Let $k > t > 0$. A $(k, t, \lambda)$-BIBD (balanced incomplete block design) is a $2-(k, t, \lambda)$ design.

Several conditions have been identified that are necessary for the existence of BIBD, but a full characterization of the existence of BIBD is an open problem. We present several of the simple necessary conditions, as well as some of the properties of BIBD in Theorem 2.3.7.

**Theorem 2.3.7.** A $(k, t, \lambda)$-BIBD exists only if:

1. $kr = bt$,
2. $r(t-1) = \lambda(k-1)$, and
3. Fisher’s inequality: $b \geq k$. 

Finite projective planes are equivalent to BIBD with certain properties, and thus can be defined entirely as a special class of BIBD.

**Definition 2.3.8.** An \((n^2 + n + 1, n + 1, 1)\)-BIBD is called a *projective plane of order* \(n\).

We now examine a family of designs that, unlike BIBD, allow for strengths \(t > 2\), but are characterized by the fact that they only allow for index \(\lambda = 1\).

**Definition 2.3.9.** Let \(k > t > s > 0\). A *Steiner system* \(S(s, t, k)\) is an \(s-(k, t, 1)\) design.

Two of the most simple and well studied families of Steiner systems are the cases where \(t \in \{3, 4\}\) and \(s = t - 1\), as defined below.

**Definition 2.3.10.** Let \(k > 0\). A *Steiner triple system of order* \(k\), written \(STS(k)\), is an \(S(2, 3, k)\).

One of the most famous and commonly encountered examples of a combinatorial design is the unique \(STS(7)\) (up to isomorphism, defined below), which has blockset \(B = \{012, 034, 056, 135, 146, 245, 236\}\), where we write \(abc\) to mean the set \(\{a, b, c\}\) for convenience. Theorem 2.3.11 details the exact existence conditions for STS, and provides some basic properties that we will use to derive the bound on variable strength covering arrays over STS. Proofs of these results and more facts on triple systems can be found in [28].

**Theorem 2.3.11.** An \(STS(k)\) *exists if and only if* \(k \equiv 1, 3 \pmod{6}\), *and has the following properties:*

- \(r = \frac{k-1}{2} = \frac{k-1}{2}\)
- \(b = \frac{k(k-1)}{6}\).

**Proof.** The calculations for \(b\) and \(r\) follow from Corollary 2.3.5. Explicit constructions for \(STS(k)\) when \(k \equiv 1, 3 \pmod{6}\) can be found in [28]. When \(k \not\equiv 1, 3 \pmod{6}\), \(b\) and \(r\) cannot both be integers, which is impossible.

**Definition 2.3.12.** Let \(k > 0\). A *Steiner quadruple system of order* \(k\), written \(SQS(k)\), is a \(S(3, 4, k)\).

As was the case with STS, we can fully characterize the existence of SQS. Proofs of the following theorem can be found in [28, 86].
Theorem 2.3.13. An SQS \((k)\) exists if and only if \(k \equiv 2, 4 \pmod{6}\), and has the following properties:

- \(r = \binom{k-1}{2} = \frac{(k-1)(k-2)}{6}\)
- \(b = \binom{k}{3} = \frac{k(k-1)(k-2)}{24}\)
- \(\lambda_2 = \frac{k-2}{2}\).

There are many techniques that can be used to build designs from other designs. One particular construction of interest, called the derived design, is given in Theorem 2.3.14.

Theorem 2.3.14. Let \(S = (V, B)\) be an \(s-(k,t,\lambda)\) design. For any \(x \in V\), define:

\[
T = (V', B') = (V \setminus \{x\}, \{B \setminus \{x\} : B \in B, x \in B\}).
\]

Then \(T\) is an \((s-1)-(k-1, t-1, \lambda)\) design, called the derived design of \(S\) at \(x\), or simply a derived design of \(S\). We then say that \(S\) is an extension design of \(T\).

Proof. Consider any \((s-1)\)-set \(M\) of \(V'\). Then, take \(M' = M \cup \{x\}\), which is an \(s\)-set of \(V\). Since \(S\) is an \(s-(k,t,\lambda)\) design, there are precisely \(\lambda\) blocks of \(B\) containing \(M'\). For each of these blocks \(B\), since \(x \in B\), \(B' = B \setminus \{x\}\) is in \(B'\). These are precisely the blocks of \(B'\) containing \(M\), as required.

2.4 Testing and Covering Arrays

Orthogonal arrays are used in the design of statistical experiments. Covering arrays, as a relaxation of orthogonal arrays, and their generalizations allow us to reap the benefits of orthogonal array testing on a larger number of systems.

We begin by defining some of the key concepts in software testing. Since the number of possible inputs for many programs can be effectively infinite, in order to quantify testing, we use coverage criteria, which define what inputs to test, and indicate when to stop testing. Coverage criteria lead to test requirements, which dictate the specific element of the artifact that the test cases must satisfy or cover. A test set consists of a number of test cases, which are comprised of test case values (the inputs required for the test execution), expected results, prefix values (any inputs necessary to put the software in the proper state to receive the test case values), and postfix values (any inputs needed after the test case values to perform execution and verify results). Coverage criteria can
be interrelated through subsumption: $C_1$ subsumes $C_2$, which we write $C_1 \geq C_2$, if every test set satisfying $C_1$ also satisfies $C_2$. Subsumption relationships do not allow us to draw any conclusions comparing the effectiveness of testing.

A general overview of the theory of software testing, from which these definitions and strategies can be found in greater detail, is given in [5]. Testing with covering arrays and orthogonal arrays falls under the umbrella of combinatorial testing: in Section 2.4.1 we detail the principles involved in combinatorial testing, examine several different coverage criteria that constitute combinatorial testing, and then discuss support for using covering arrays as a means of test suite design. Then, in Section 2.4.2 we enumerate several other uses of covering arrays in testing.

### 2.4.1 Combinatorial Testing

Combinatorial testing embodies coverage criteria that make some guarantee on the test cases with regards to combinations of assignments of levels to factors. We give an overview of the technique here, as well as detail some of the coverage criteria that fall under this umbrella.

The first step in combinatorial testing is to perform input space partitioning: this involves partitioning the input space into regions assumed to contain equally useful values with respect to testing, and forming test cases by choosing values from each region. A partition of the space into blocks is based on some characteristic. After partitioning, we work under the assumption that any input value in a block is as good as any other.

The input space can be modelled in multiple ways. The steps to doing so generally involve identifying testable blocks of functionality, which may correspond to methods, classes, UML test cases, etc, and parameters that affect the functionality; these two together form the input domain. The input domain model (IDM) then couples this with identifying characteristics and partitioning the input space into blocks, with test inputs being taken to be tuples taking on a value from a block for each characteristic.

Interface-based IDM is largely mechanical and considers parameters in isolation, where it is easy to derive characteristics. This may result in an incomplete IDM due to missing information not captured by the parameters considered in this way, e.g. by subinteractions amongst the parameters.

Functionality-based IDM identifies characteristics corresponding to functionality. This can generally be done from the requirements and allows for more semantic information to be captured in the partitioning; however, it is more difficult to design reasonable
Characteristics, which may span multiple parameters and thus be more complex.

Characteristics are easily identified in interface-based IDM, and should be derived using domain knowledge instead of implementation details. In functionality-based, preconditions and postconditions can be good sources of characteristics, especially at separating normal and abnormal functionality. Another technique is to investigate the relationships between inputs and check for missing ones. Ideally, the result is a model with many characteristics with fewer blocks; the reverse may be too costly to test.

After the characteristics have been chosen, the problem remains of how to partition blocks and select representatives from each block. Many strategies exist and the optimal depends on the nature of the data being considered. Examples include using valid values, boundary values, normal use values, invalid values, etc. For data types such as integers, it is normal to partition based on the relation to a special value in the domain. Special care must often be taken in functionality-based analysis to ensure that the input space is actually being partitioned.

It is possible to use more than one IDM; indeed, if the complexity is high, it might be preferable to have several smaller ones. This allows a divide-and-conquer approach to be used for modelling characteristics and blocks, and allows varying levels of coverage across IDM. There can be overlap between the IDM if it makes sense with respect to test cases.

There are several coverage criterion that can be applied to an IDM:

**All Combinations Coverage:** (ACoC) All combinations of blocks from all characteristics must appear. This corresponds to a CA($N; k, k, g$), which we usually call exhaustive testing, and generally results in too many tests to be a valid test strategy.

**Each Choice Coverage:** (ECC) One value from each block for each characteristic must appear at least once. This corresponds to a CA($N; 1, k, g$), and usually allows too much flexibility in how input values are gathered into test cases.

**Pairwise Coverage:** (PWC) A value from each block for each characteristic must appear with a value from every block for every other characteristic. This corresponds to a CA($N; 2, k, g$). We call this testing modality pairwise interaction testing.

**t-wise Coverage:** (TWC) A value from each block for each group of $t$ characteristics must be combined. This corresponds to a CA($N; t, k, g$), and we call this testing approach
\textit{t-wise interaction testing.}

For \( t > 2 \), we have the following subsumption relationships:

\[
\text{ACoC} \geq \text{TWC} \geq \text{PWC} \geq \text{ECC}.
\]

PWC and TWC still allow values to be combined blindly; this may be problematic in practice, so we examine a strengthening of ECC that does not allow this:

**Base Choice Coverage:** (BCC) A base choice block is chosen for each characteristic. The base test combines all these values. Every other test maintains all but one base choice constant. The base choice is typically the simplest, smallest, first in some ordering, or most likely choice.

**Multiple Base Choice Coverage:** (MBCC) At least one base choice is chosen for each characteristic. The base tests are formed by using each base choice for each characteristic at least once. For subsequent tests, hold all but one base choice constant for each base test and use each non-base choice in each other characteristic. Note that tests can appear multiple times here.

We have:

\[
\text{ACoC} \geq \text{MBCC} \geq \text{BCC} \geq \text{ECC}.
\]

BCC is not comparable with PWC or TWC via subsumption.

Some block combinations may be infeasible, which must be documented in the IDM. There are two types of constraints that we encounter, namely prohibited block combinations, and necessary block combinations. From ACoC, PWC, and TWC, we can simply drop infeasible combinations from consideration. For BCC, if the block choice conflicts with a base case, the base case value needs to be reconsidered.

The idea behind input partition testing is to divide the input domain into classes of tests that are considered equivalent, and then select tests from each class according to a probability distribution. Hamlet and Taylor \cite{HamletTaylor} call into question the benefits of partition testing over random testing where tests are simply selected according to some distribution. A class is called homogeneous if all of the test cases that fall into it cause either all passes or all failures. They propose that partition testing is flawed in that it is no better than the information used to define the partitioning and obtaining homogeneous partitions is inherently difficult. If the partitions are not homogeneous, the partitioning was not sufficiently refined, and thus, it is not possible to inspire confidence through successful results and testing fails. If they are homogeneous, then the classes themselves
yield a correctness proof of the program. Statistical analyses show that partition testing can perform worse, better, or about the same as random testing; even when it performs better, random testing can achieve 80% of its effectiveness, which could be boosted by running more random tests. Regardless, they recognize that partition testing has value in practice in that testing can begin early through partition design and classes with high failure rate are useful in fault detection, and can be useful particularly if we are able to find input classes with high failure rates.

Nair et al. [73] challenge Hamlet and Taylor’s claims and propose that partition testing is at least as useful and possibly much more so than random testing. An allocation strategy determines the proportion of tests that come from each class in partition testing. They provide an allocation strategy, proportional allocation, which is implementable but inefficient and is guaranteed to perform at least as well as random testing; knowledge of the software allows more effective partitioning that increases the effectiveness of partition testing, and partitions should be built to be as homogeneous as possible. The allocation strategy should sample more frequently from partitions that are more likely to contain tests resulting in failures: constant allocation is one strategy that works if classes with high proportions of failures are small. They also demonstrate that when partition testing is better than random testing, it also inspires more confidence: as partition testing with proportional allocation is as least as good as random testing, this can always be achieved. Nair concludes that Hamlet and Taylor’s error was in computing the necessary sample sizes under partition testing and the use of a conservative estimate.

The combinatorial approach to testing, i.e. covering all $t$-way interactions, tends to give good code coverage in practice, even for low strength $t$, e.g. $t \in \{2, 3\}$, as shown in [21]. Experiments by Dunietz et al. [42] support this: they used code and path coverage as measures of test set effectiveness and contrasted random testing and covering array testing to show that while they both performed well, covering arrays obtained better code and path coverage and demonstrated less variability (and thus more reliability) in results. In all cases, full code coverage was achieved by $t = 2$. Kuhn et al. [57] give a survey of the experimental results of interaction testing, which demonstrate that in all experiments conducted, covering arrays of strengths between 2 and 6 were sufficient to discover all faults.

Dalal and Mallows [37] provide a survey of earlier covering array results and a brief history of the use of covering arrays in testing. They introduce some metrics for a covering array CA($N; t, k, g$). If we let $m$ be the number of distinct $t$-tuples covered by the array, then the $t$-coverage is defined as $m/(g^t \binom{k}{t})$, or the ratio of $t$-tuples covered to
the total number of possible $t$-tuples. Williams and Probert explore $t$-coverage in more detail in [90]. The $t$-diversity is defined as $m/(N^k)$, or the ratio of distinct $t$-tuples covered to the total number of $t$-tuples covered. They cite earlier works and remark that pairwise testing in practice provides 80-90% block coverage, whereas in system testing it is generally difficult to achieve higher than 50-60% block coverage. Some experiments demonstrate that for $t = 3$, 100% block coverage was achieved, but higher $t$ was needed for path coverage.

Cohen et al. introduce the Automated Efficient Test Generator (AETG) in [21], which uses a greedy technique to build covering arrays. In [19, 21], they discuss the merits of using AETG and examine the results of applying it to testing several applications. They demonstrate that using AETG to test a handful of UNIX commands resulted in 90% block coverage and in the case of the sort command, which is difficult to test given its complexity, performed significantly better with respect to error detection and block coverage than an earlier technique of Wong et al. where test suites were built from a pool of randomly generated test cases. For an inventory database interface, testing the input validator using pairwise interaction resulted in the discovery of errors that had gone undetected by previous testing techniques.

Burroughs et al. [13] apply AETG to protocol conformance testing of a switch. They measure breadth of coverage by the number of pairs of values tested. Traditional testing techniques in this area consist of exhaustive testing, which is too costly; default value testing, which has far too low coverage; and explicitly pairwise testing the two most important factors while fixing the others and then performing default value testing on the remaining factors, which results in a very unbalanced test suite with regards to the occurrence of levels. Covering arrays guarantee full coverage using this metric and require far fewer test cases than the exhaustive approach.

Burr and Young [12] show the benefits of using covering arrays in testing by using AETG to generate a test table to achieve pairwise coverage for a component of an internal e-mail service. Using conventional testing and default values testing, code coverages of 62% and 85% were achieved respectively. Using a covering array to test, 93% of the code was executed, and the other 7% was deemed unreachable due to the architecture of the testing.
2.4.2 Other Applications of Covering Arrays to Testing

Korel et al. [55] propose a technique to use variable strength covering arrays to project a set of expected results from a limited size test set to every possible test case. If the input-output relationships can be modelled to accurately map the interactions of inputs to outputs as per Cheng et al. [18], then finding a variable strength covering array for the problem defines a partition, for each output, on the test cases such that the test cases in each class should generate the same value for the output. Then the value determined for the output by running one test from each class can be used as the expected value for all test cases in the class.

A failed test case is then either an indicator of a fault in the program or an improperly modelled input-output relationship. While it can be difficult to find all such relationships, there are source code and black box techniques and tools for doing so, and an iterative process can be performed: find a VCA, and run it to get all expected results. If there is any inconsistency between outputs with same inputs, due to repeated combinations of parameters appearing in the VCA, then some relationship is not modelled correctly. Investigate, repair, and repeat until the results are consistent. The authors argue that running all possible test cases and using lookup to determine the expected results has benefit: it assists in further detecting improper input-output modelling; reachability and propagation of programmer faults may occur in the full test set that did not occur in the reduced set as dictated by the VCA; and this is more validating of customer use of the application, which may be quite different than the tests given in the VCA. However, in their experiment on a mature chemistry analysis product, the expected results of the VCA tests correctly predicted the expected results for all possible tests.

Williams and Probert formally define the system component interaction test coverage problem and discuss using covering arrays for component testing in [90], deriving metrics as discussed in Section 2.4.1 for measuring the coverage of test sets in combinatorial component testing.

Yilmaz et al. [91] investigate creating fault characterization models from test data: given the results of running various software configurations for a test, they create a fault characterization tree with nodes modelling variables and branches representing values in an attempt to determine what variable-value settings lead to which errors; this information can be used in bug fixing. Their experiments support several conclusions: it is better to build one characterization model per error instead of trying to incorporate multiple errors into the same model; even low strength covering arrays often produce
models as effective as those generated by exhaustive testing; higher strength covering arrays produce more precise models but are considerably more costly; and precision can be increased to approach higher strength covering array models by using several lower strength covering arrays together, which is generally cheaper. They also apply variable strength covering arrays as per Cohen’s formulation as detailed in Section 2.2.5, imposing a higher strength subarray on a subset of factors that is considered to be higher risk, and find that they can allow base array strength to drop, and thus cost (in terms of number of configurations), without compromising accuracy. Alternatively, by maintaining the number of configurations approximately the same, the accuracy can be improved.
Chapter 3

Variable Strength Covering Arrays and Qualitative Independence Hypergraphs

We now establish the theory of variable strength covering arrays, by examining how they are tied to homomorphisms and qualitative independence. This motivates our presentation of qualitative independence hypergraphs in Section 3.2. This important family of hypergraphs, a generalization of qualitative independence graphs as given in [66, 67, 68], is connected directly to VCA via homomorphisms, and thus is of deep importance in the study of VCA. We establish hypergraph-theoretic results and properties of these hypergraphs, which provides insight into their structure.

We then describe constructions and modifications of VCA in Section 3.4. Since VCA do not require all tested interactions to be of the same strength, we are able to derive two operations on the factors of VCA, where we split a factor into two as given in Theorem 3.4.1 and conversely, join two factors together as stated in Theorem 3.4.3. These two operations, along with the level-dropping operation from standard covering arrays, provide powerful tools for extending VCA results and viewing VCA problems under a flexible number of guises. We illustrate this usefulness through three examples.

We give a divide-and-conquer construction for VCA over hypergraphs with a cut-vertex in Section 3.4.2, which can be extended to hypergraphs with cut cliques.

Finally, we look at VCA over specific families of hypergraphs. In Section 3.4.3, we turn our attention to the problem of consecutive and cyclic consecutive covering arrays, giving a direct, optimal construction for consecutive covering arrays, and optimal constructions
for most combinations of parameters of cyclic consecutive covering arrays, listing the remaining cases to be examined. In Section 3.4.4 we examine the problem of building variable strength covering arrays over 2-trees, and provide a conjecture regarding the nature of an optimal construction based on strong experimental evidence. In Section 3.3 we solve the problem fully for triangulation hypergraphs of the sphere.

3.1 Homomorphisms and Colourings

We examine how homomorphisms and colourings connect hypergraphs and variable strength covering array number. Maity [60] independently investigated and established some of these connections between homomorphisms and variable strength covering arrays.

Theorem 3.1.1. Let $G$, $H$ be hypergraphs and $g$ be a positive integer. If $G \to H$, then $\text{VCAN}(G, g) \leq \text{VCAN}(H, g)$.

Proof. Without loss of generality, let $V(G) = \{0, \ldots, k_G - 1\}$, $V(H) = \{0, \ldots, k_H - 1\}$. If $f : G \to H$ is a homomorphism and we have a VCA($n; H, g$) where $n = \text{VCAN}(H, g)$, say $M$, then let $M'$ be the $n \times |V(G)|$ array where we set column $i$ to column $f(i)$ of $M$. Since $f$ is a homomorphism, $M'$ is a VCA($n; G, g$), and thus $\text{VCAN}(G, g) \leq n$.

An example of Theorem 3.1.1 is shown in Figure 3.1.

We can combine this theorem with Theorems 2.1.34, 2.1.35, and 2.1.36 to derive some bounds on the size of VCA using colourings. Generally, we have the following:

Corollary 3.1.2. If $H$ is a strongly $c$-colourable hypergraph with $t = \text{rank}(H)$, then $\text{VCAN}(H, g) \leq \text{CAN}(t, c, g)$.

Proof. We have that $\Delta(H) \to \Delta(K^t_c)$ by Theorem 2.1.36, so by Theorem 3.1.1

$$\text{VCAN}(H, g) = \text{VCAN}(\Delta(H), g) \leq \text{VCAN}(\Delta(K^t_c), g) = \text{VCAN}(K^t_c, g)$$

From Proposition 2.2.7, $\text{VCAN}(K^t_c, g) = \text{CAN}(t, c, g)$.

3.2 Qualitative Independence and Qualitative Independence Hypergraphs

For a vector $v \in \mathbb{Z}_n^g$, write $v[i]$ to represent the value in the $i$th coordinate of $v$ for $0 \leq i < n$. Vectors play an important role in qualitative independence, and so we define
the following two permutation actions on vectors. If $v \in \mathbb{Z}_g^n$ and $\sigma \in S_n$, write:

$$v^\sigma = (v[\sigma(0)], \ldots, v[\sigma(n-1)])$$

For $\phi \in S_g$, write:

$$\phi(v) = (\phi(v[0]), \ldots, \phi(v[n-1])).$$

We now discuss qualitative independence, which is another way of capturing the balance coverage requirements of covering arrays. Qualitative independence can be defined over vectors or partitions, and we will use it to define an infinite family of hypergraphs that is strongly tied to covering arrays. We begin by defining the notion of qualitative independence as it relates to vectors.
Definition 3.2.1. A set of \(t\) vectors \(\{v_0, \ldots, v_{t-1}\} \subset \mathbb{Z}_g^n\) is said to be \(t\)-qualitatively independent if, for every \(t\)-tuple \((x_0, \ldots, x_{t-1}) \in \mathbb{Z}_g^t\), there exists \(0 \leq i < n\) such that \((v_0[i], \ldots, v_{t-1}[i]) = (x_0, \ldots, x_{t-1})\). If \(t = 2\), we simply say that \(v_0\) and \(v_1\) are qualitatively independent.

All of the definitions we have seen for the various generalizations of covering arrays can be reformulated in terms of qualitative independence, as shown by this alternative definition of variable strength covering arrays:

Definition 3.2.2. Given an ASC \(\Delta\) with \(|V(\Delta)| = k\), a variable strength covering array \(VCA(n; \Delta, g)\) is an \(n \times k\) array \(M = [m_0, m_1, \ldots, m_{k-1}]\) such that:

1. \(m_i \in \mathbb{Z}_g^n\).

2. If \(e = \{i_0, \ldots, i_{t-1}\} \in \Delta\), the vector set \(S = \{m_{i_0}, m_{i_1}, \ldots, m_{i_{t-1}}\}\) is \(t\)-qualitatively independent.

We now define qualitative independence in terms of partitions of an \(n\)-set, and the correspondence between sets of vectors and partitions.

Definition 3.2.3. Let \(P^n_g\) be the set of all partitions of \(\mathbb{Z}_n\) into \(g\) (possibly empty) classes. A set of \(t\) partitions \(\{P_0, \ldots, P_{t-1}\} \subset P^n_g\) is said to be \(t\)-qualitatively independent if, for any choice of classes \(P_i \in P_i\), we have that:

\[ P_1 \cap P_2 \cap \ldots \cap P_t \neq \emptyset. \]

Let \(v \in \mathbb{Z}_g^n\) be a vector containing all symbols of \(\mathbb{Z}_g\), i.e. for every \(a \in \mathbb{Z}_g\), there is some \(0 \leq i < n\) such that \(v[i] = a\). Then \(v\) defines a partition of \(\mathbb{Z}_n\) into \(g\) partition classes, say \(P_v = \{P_0, \ldots, P_{g-1}\}\), where:

\[ P_a = \{i : v[i] = a\}. \]

Proposition 3.2.4. For \(u, v \in \mathbb{Z}_g^n\), we have that \(P_u = P_v\) if and only if there is some permutation \(\sigma \in S_g\) such that:

\[ (\sigma(u[0]), \ldots, \sigma(u[n - 1]))^T = v. \]

For each \(P \in P^n_g\), if \(P\) has \(h\) nonempty classes, there are exactly \(g!/(g - h)!\) vectors of \(\mathbb{Z}_g^n\), say \(v_0, \ldots, v_{g-1}\) such that \(P_{v_i} = P\) for \(0 \leq i \leq g - 1\).
Proof. Assume \( u, v \in \mathbb{Z}_g^n \) with \( P_u = \{ P_0, \ldots, P_{g-1} \} \), \( P_v = \{ Q_0, \ldots, Q_{g-1} \} \), and \( P_u = P_v \). Thus, there is some permutation \( \sigma \in S_g \) such that for all \( i \in \mathbb{Z}_g \):

\[
P_{\sigma(i)} = Q_i.
\]

Then, by definition of the mapping from \( \mathbb{Z}_g^n \) to \( P_g^n \), we have that:

\[
\sigma(u) = (\sigma(u[0]), \ldots, \sigma(u[n-1]))^T = v.
\]

The other direction of the equivalence is obvious as simply relabelling the symbols of a vector only changes the order of the classes of a partition.

Consider a partition \( P \in P_g^n \) with \( h \) nonempty classes. Then any vector \( v \in \mathbb{Z}_g^n \) that defines \( P \) must have exactly \( h \) symbols of \( \mathbb{Z}_g \) appear amongst its positions. The number of ways to pick the \( h \) symbols is \( \binom{g}{h} \), and we can assign the symbols \( h! \) ways, so there are \( \binom{g}{h} h! = g!/(g - h)! \) vectors of \( \mathbb{Z}_g^n \) corresponding to \( P \).  

\[\square\]

**Theorem 3.2.5.** Consider a set of vectors \( \{v_0, \ldots, v_{t-1}\} \subseteq \mathbb{Z}_g^n \). We then have that \( \{v_0, \ldots, v_{t-1}\} \) is \( t \)-qualitatively independent if and only if \( \{P_{v_0}, \ldots, P_{v_{t-1}}\} \) is \( t \)-qualitatively independent.

Proof. Let \( \{v_0, \ldots, v_{t-1}\} \) be \( t \)-qualitatively independent. Thus, for any \( t \)-tuple of vectors, say \((x_0, \ldots, x_{t-1}) \in \mathbb{Z}_g^t\), there is some index \( 0 \leq i < n \) such that, for all \( 0 \leq j < t \), \( v_j[i] = x_j \). This means that the class \( P_0 \in P_{v_0} \) corresponding to \( x_0 \) contains \( i \), as does \( P_1 \in P_{v_1} \) corresponding to \( x_1 \), etc. Thus, \( i \in P_0 \cap \ldots \cap P_{t-1} \).

Let \( \{P_{v_0}, \ldots, P_{v_{t-1}}\} \) be \( t \)-qualitatively independent, and for each partition \( P_{v_i} \), write \( P_{v_i,j} \) to represent the class corresponding to symbol \( j \in \mathbb{Z}_g \) of the vector \( v_i \). For any \( t \)-tuple \((x_0, \ldots, x_{t-1}) \in \mathbb{Z}_g^t \), we then have, due to \( t \)-qualitative independence, that:

\[
J = P_{v_0,x_0} \cap \ldots \cap P_{v_{t-1},x_{t-1}} \neq \emptyset.
\]

Thus, there is some \( j \in J \). This implies that \( v_i[j] = x_i \) for all \( 0 \leq i \leq t - 1 \). Thus, \( \{v_0, \ldots, v_{t-1}\} \) is \( t \)-qualitatively independent.  

\[\square\]

**Proposition 3.2.6.** If, for any \( g \) and \( n \), \( S \subseteq \binom{\mathbb{Z}_g^n}{t} \) is \( t \)-qualitatively independent, then, for all \( 1 \leq s \leq t - 1 \), every \((t-s)\)-subset \( T \in \binom{S}{s} \) is \((t-s)\)-qualitatively independent. As well, if \( S' \subseteq \binom{\mathbb{Z}_g^n}{t} \) is \( t \)-qualitatively independent, then, for all \( 1 \leq s \leq t - 1 \), every \((t-s)\)-subset \( T' \in \binom{S'}{s} \) is \((t-s)\)-qualitatively independent.
Proof. Let \( T = \{v_0, \ldots, v_{t-2}\} \), \( S = \{v_0, \ldots, v_{t-2}, v_{t-1}\} \), and consider a \((t - 1)\)-tuple \((x_0, \ldots, x_{t-2}) \in \mathbb{Z}^{t-1}_g\). Then since \( S \) is \( t \)-qualitatively independent, we have that, for the \( t \)-tuple \((x_0, \ldots, x_{t-2}, 0) \in \mathbb{Z}^t_g\), there is some \( 0 \leq i < n \) such that \((v_0[i], \ldots, v_{t-2}[i], v_{t-1}[i]) = (x_0, \ldots, x_{t-2}, 0)\), so \((v_0[i], \ldots, v_{t-2}[i]) = (x_0, \ldots, x_{t-2})\), as required. The argument iterated \( s \) times gives the result for a \((t - s)\)-subset. Applying Theorem 3.2.5 gives the result for partitions. \( \square \)

Stirling numbers of the second kind provide a way to count partitions of an \( n \)-set into a fixed number of nonempty classes, and thus, will come in handy in counting arguments and formulas for qualitative independence hypergraphs.

**Definition 3.2.7.** Let \( S(n, k) \) denote the number of ways to partition an \( n \)-set into \( k \) nonempty unlabelled subsets; these are called the **Stirling numbers of the second kind**.

We now define two hypergraphs, one for vectors and one for partitions, where hyperedges are dictated by qualitative independence. The study of these hypergraphs is motivated by their connections to variable strength covering arrays, as illustrated in Theorem 3.2.9.

We examine several properties of these hypergraphs and the relations between them.

**Definition 3.2.8.** A **qualitative independence vector hypergraph** \( VQIH(n, g) \) is a hypergraph \((V, E)\) with:

1. \( V = \mathbb{Z}_g^n \).

2. For any set \( e \subseteq V \) of size \( t \), \( e \in E \) if and only if \( e \) is \( t \)-qualitatively independent.

A **qualitative independence partition hypergraph** \( PQIH(n, g) \) is a hypergraph \((V', E')\) with:

1. \( V' = \mathcal{P}_g^n \).

2. For any set \( e \subseteq V' \) of size \( t \), \( e \in E' \) if and only if \( e \) is \( t \)-qualitatively independent.

In [66], Meagher introduces the qualitative independence graphs \( QI(n, g) \). We note that these are generalizations of these graphs, and in particular, we have that \( QI(n, g) = E_2(PQIH(n, g)) \).

**Theorem 3.2.9.** Given an ASC \( H \), there exists a homomorphism \( H \rightarrow VQIH(n, g) \) if and only if there exists a VCA\((n; H, g)\).
Proof. Let $G$ be a VQIH($n, g$) and write $V(H) = \{v_0, \ldots, v_{k-1}\}$. If a homomorphism $f : H \to G$ exists, then take the array:

$$A = [f(v_0), \ldots, f(v_{k-1})].$$

We claim that this is a VCA($n; H, g$). Since $V(G) = \mathbb{Z}_g^n$, $f(v_i) \in \mathbb{Z}_g^n$, as required by the first property of VCA. Then if $e \in E(H)$ with $|e| = t$, we have that $f(e) \in E(G)$, so by property of the edges of $G$, the vectors in $f(e)$ are $t$-qualitatively independent, as required by the second property of VCA. Thus, $A$ is a VCA($n; H, g$).

Now, assume that we have a VCA($n; H, g$), $A = [m_0, \ldots, m_{k-1}]$. Consider the mapping $f : V(H) \to V(G)$ defined by $f(v_i) = m_i$. We claim that this is a hypergraph homomorphism: if $e \in E(H)$ with $|e| = t$, then since $A$ is a variable strength covering array, $f(e) = \{m_i : i \in e\}$ is $t$-qualitatively independent, so $f(e) \in E(G)$. Thus, $f$ is a homomorphism $H \to G$.

**Corollary 3.2.10.** A CA($n; t, k, g$) exists if and only if VQIH($n, g$) contains a $t$-clique of size $k$.

**Proof.** A CA($n; t, k, g$) is equivalent to a VCA($n; K^t_k, g$) as implied by Proposition 2.2.7. By Theorem 3.2.9, this VCA exists if and only if we have a homomorphism $K^t_k \to VQIH(n, g)$. By Proposition 2.1.19, since $\omega_t(K^t_k) = k$, then $\omega_t(VQIH(n, g)) \geq k$, so VQIH($n, g$) contains a $t$-clique of size $k$. □

**Proposition 3.2.11.** 1. The total number of vertices of VQIH($n, g$) is $g^n$.

2. The total number of vertices of PQIH($n, g$) is $\sum_{i=1}^{g} S(n, i)$.

**Proof.** 1. We have that $|\mathbb{Z}_g^n| = g^n$.

2. The number of partitions of $\mathbb{Z}_n$ into $i$ nonempty classes, for $1 \leq i \leq g$, is $S(n, i)$, so the total number of partitions of $\mathbb{Z}_n$ into $g$ classes with some classes possibly empty is simply the sum. □

**Proposition 3.2.12.** VQIH($n, g$) \$\rightarrow\$ PQIH($n, g$).

**Proof.** Consider the map that takes $v \in \mathbb{Z}_g^n$ to $P_v \in \mathcal{P}_g^n$. By Theorem 3.2.5, this is a homomorphism, and by Proposition 3.2.4, it is surjective. □

**Proposition 3.2.13.** PQIH($n, g$) \$\rightarrow\$ VQIH($n, g$).
Proof. By Proposition 3.2.12, we have an epimorphism \( f : VQIH(n, g) \to PQIH(n, g) \), so for any \( P \in V(PQIH(n, g)) = \mathcal{P}_g^n \), we can consider its inverse image in \( f \):

\[
f^{-1}(P) = \{ v \in \mathbb{Z}_g^n : f(v) = P \}.
\]

Define the following map \( h : \mathcal{P}_g^n \to \mathbb{Z}_g^n \): using a lexicographic ordering on the vectors of \( \mathbb{Z}_g^n \), for each \( P \in \mathcal{P}_g^n \), set \( h(P) \) to the lexicographically smallest vector in \( f^{-1}(P) \). If \( e \in E(PQIH(n, g)) \) with \( |e| = t \), then the partitions in \( e \) are \( t \)-qualitatively independent. Any set of vectors corresponding to the partitions in \( e \) must also be \( t \)-qualitatively independent, so \( h \) is a homomorphism. Furthermore, if, for some \( P, Q \in \mathcal{P}_g^n \), we have that \( h(P) = h(Q) \), then since \( f \) is well-defined, \( P = f(h(P)) = f(h(Q)) = Q \), so \( h \) is injective.

By Propositions 3.2.12 and 3.2.13, \( VQIH(n, g) \) and \( PQIH(n, g) \) are homomorphically equivalent, and thus they share any homomorphically determined parameters. Thus, when not inconvenient to do so, we will focus our attention on \( PQIH(n, g) \) when discussing any parameters or proofs where the results can be homomorphically applied as well to \( VQIH(n, g) \).

**Theorem 3.2.14.** \( \text{rank}(VQIH(n, g)) = \text{rank}(PQIH(n, g)) = \lfloor \log_g n \rfloor \).

**Proof.** Let \( t = \text{rank}(VQIH(n, g)) \), and take \( e \in E(VQIH(n, g)) \) to be an edge of size \( t \). We then necessarily have that \( g^t \) different \( t \)-tuples must be covered by the vectors in \( e \). As the vectors have length \( n \), this means:

\[
 n \geq g^t.
\]

Furthermore, as \( t \) is maximum, there is no \((t + 1)\)-qualitatively independent set in \( VQIH(n, g) \) by definition. If \( n \geq g^{t+1} \), we can easily select a set of \((t + 1)\)-qualitatively independent vectors simply by listing the \( g^{t+1} \) tuples as the first \( g^{t+1} \) rows of an \( n \times (t + 1) \) matrix and filling in the remaining \( n - g^{t+1} \) rows arbitrarily. Then the columns form a set of \((t + 1)\)-qualitatively independent vectors. Thus, we must have:

\[
 n < g^{t+1}.
\]

Taking the logarithm of both equations gives that \( t \leq \log_g n \) and \( t + 1 > \log_g n \), giving:

\[
 (\log_g n) - 1 < t \leq \log_g n.
\]

Hence, we have that:

\[
 t = \lfloor \log_g n \rfloor.
\]

By Propositions 2.1.11, 3.2.12 and 3.2.13, \( \text{rank}(VQIH(n, g)) = \text{rank}(PQIH(n, g)) \). \( \square \)
Corollary 3.2.15. VQIH\((n,g)\) and PQIH\((n,g)\) are both ASC.

**Proof.** This is a direct corollary of Proposition 3.2.6.

Theorem 3.2.16. 1. Let \(v\) be a vertex of VQIH\((n,g)\), i.e. \(v \in \mathbb{Z}_g^n\), and for each \(s \in \mathbb{Z}_g\), write:

\[
as = |\{0 \leq j < n : v[j] = s\}|.
\]

Then \(v\) appears in no edges of size \(t\) if and only if, for some \(s \in \mathbb{Z}_g\), \(a_s < g^{t-1}\).

2. A vertex \(P \in V(PQIH(n,g))\) appears in no edges of size \(t\) if and only if, for some \(P \in \mathcal{P}, |P| < g^{t-1}\).

**Proof.**

1. We have that \(v\) is in an edge of size \(t\) if and only if there is a set of \(t\)-qualitatively independent vectors including \(v\), say \(S = \{v,u_1,\ldots,u_{t-1}\}\). By definition, this is equivalent to saying that for each of the \(g^t\) possible \(t\)-tuples \((x_0, x_1, \ldots, x_{t-1}) \in \mathbb{Z}_g^t\), there is some index \(i\) such that \((v[i], u_1[i], \ldots, u_{t-1}[i]) = (x_0, x_1, \ldots, x_{t-1})\). We can restate this by saying that for every value \(s \in \mathbb{Z}_g\), for each of the \(g^{t-1}\) possible \((t-1)\)-tuples \((x_1, \ldots, x_{t-1}) \in \mathbb{Z}_g^{t-1}\), there is some index \(i\) such that \((v[i], u_1[i], \ldots, u_{t-1}[i]) = (s, x_1, \ldots, x_{t-1})\).

As there are \(g^{t-1}\) such tuples for each \(s \in \mathbb{Z}_g\), we have that \(a_s \geq g^{t-1}\).

Now we show that if for all \(s \in \mathbb{Z}_g\), we have that \(a_s \geq g^{t-1}\), then there is a \(t\)-qualitatively independent set of vectors containing \(v\). List the \(g^{t-1}\) different \((t-1)\) tuples of \(\mathbb{Z}_g^{t-1}\), say \(t_0, \ldots, t_{g^{t-1}-1}\), and let \(\{v_1, \ldots, v_{t-1}\}\) be vectors of \(\mathbb{Z}_g^n\) such that if \(v[i] = s\) and \(m = |\{0 \leq j < i : v[j] = s\}| < g^{t-1}\), then \((v_1[i], \ldots, v_{t-1}[i]) = t_m\). This is possible since \(a_s \geq g^{t-1}\). If \(v[i] = s\) and \(m = |\{0 \leq j < i : v[j] = s\}| \geq g^{t-1}\), then \((v_1[i], \ldots, v_{t-1}[i]) = (0, \ldots, 0)\). Then, by construction, this is a set of \(t\)-qualitatively independent vectors of \(\mathbb{Z}_g^n\).

2. For a partition \(\mathcal{P} \in V(PQIH(n,g))\), if for some \(P \in \mathcal{P}, |P| < g^{t-1}\), then, by the homomorphism \(f: PQIH(n,g) \to VQIH(n,g)\) of Proposition 3.2.13, for some symbol \(s \in \mathbb{Z}_g\), we have that \(P = \{0 \leq i < n : f(\mathcal{P})[i] = s\}\). Then since \(|P| < g^{t-1}\), the symbol \(s\) appears in \(f(\mathcal{P})\) less than \(g^{t-1}\) times, so by the first part of this theorem, \(f(\mathcal{P})[i]\) appears in no edges of size \(t\) in VQIH\((n,g)\). If \(e \in E(PQIH(n,g))\), \(\mathcal{P} \in e, |e| = t\), then \(f(\mathcal{P}) \in f(e)\) and \(|f(e)| = t\), which is impossible, so \(\mathcal{P}\) appears in no edges of size \(t\) in PQIH\((n,g)\).

By Proposition 3.2.12, there is a homomorphism \(g: VQIH(n,g) \to PQIH(n,g)\). Consider a partition \(\mathcal{P} \in V(PQIH(n,g))\) that appears in no edges of size \(t\). If \(f(\mathcal{P}) \in f(e)\) and \(|f(e)| = t\), this is impossible, so \(f(\mathcal{P})\) appears in no edges of size \(t\) in PQIH\((n,g)\).
appears in some edge $e \in E(VQIH(n, g))$ of size $t$, then $g(f(P)) = P \in g(e)$ and $|g(e)| = t$, thus contradicting our assumption, so $f(P)$ appears in no edges of size $t$ in $VQIH(n, g)$. Then, by the first part of the theorem, there is some $s \in \mathbb{Z}_g$ such that $s$ appears fewer than $g^{t-1}$ times in $f(P)$. But then the class corresponding to $s$ in $P$ has size smaller than $g^{t-1}$.

For a hypergraph $H$, we say that a vertex $v \in V(H)$ is isolated if there are no edges of $E(H)$ containing $v$. Isolated vertices have no impact on homomorphisms as they can be mapped arbitrarily to any vertex: hence, a homomorphism of nonempty (i.e. containing vertices) hypergraphs $H \rightarrow G$ exists if and only if a homomorphism of hypergraphs $H' \rightarrow G$ exists, where $H'$ is $H$ with its isolated vertices removed. Thus, unless otherwise explicitly stated, from this point on when discussing $VQIH(n, g)$ or $PQIH(n, g)$, we assume that we are discussing the hypergraphs only on vertices that occur in at least edges of size 1, which, by Theorem 3.2.16, in $VQIH(n, g)$ are precisely the vectors of $\mathbb{Z}^n_g$ containing all symbols of $\mathbb{Z}_g$, and in $PQIH(n, g)$ are precisely the partitions of $\mathcal{P}^n_g$ with no empty classes. From the following theorem, then, every partition that is not isolated (as a vertex) corresponds exactly to $g!$ vectors, a fact that we will use repeatedly in counting arguments to translate results between $VQIH(n, g)$ and $PQIH(n, g)$.

**Theorem 3.2.17.**

1. The total number of vertices of $PQIH(n, g)$ that are not isolated is $S(n, g)$.

2. The total number of vertices of $VQIH(n, g)$ that are not isolated is $g!S(n, g)$.

3. Every vertex (partition) of $PQIH(n, g)$ that is not isolated corresponds via the map $v \rightarrow \mathcal{P}_v$ to exactly $g!$ vertices (vectors) of $VQIH(n, g)$ that are not isolated.

**Proof.** In $PQIH(n, g)$, by Theorem 3.2.16 a partition appears in no edges if and only if some class is empty. Thus, a partition appears in an edge if and only if it has $g$ nonempty classes. The number of such partitions is precisely what $S(n, g)$ counts.

In $VQIH(n, g)$, by Theorem 3.2.16 a vector appears in no edges if and only if some symbol does not appear in the vector. Thus, a vector appears in an edge if and only if every symbol appears. To count the number of vectors of $\mathbb{Z}^n_g$ with this property, we partition the $n$ positions into $g$ nonempty classes, representing the placement of symbols from $\mathbb{Z}_g$ in the vector, which can be done $S(n, g)$ ways. The classes are unlabeled, so we label each class with a distinct value from $\mathbb{Z}_g$, which can be done in $g!$ ways, giving the result.
The final statement of the theorem follows directly from the fact that in the partitions that are not isolated, no classes are empty, and in the vectors that are not isolated, all of the symbols appear. Proposition 3.2.4 then gives the result.

We investigate several properties regarding the occurrences of edges in $VQIH(n,g)$ and $PQIH(n,g)$.

**Theorem 3.2.18.** Let $1 \leq t \leq \lfloor \log_g n \rfloor$. If $v \in \mathbb{Z}^n_g$ with the number of occurrences of symbol $i$ from $\mathbb{Z}_g$ in $v$ being $a_i$, then we have that:

$$d_{VQIH(n,g)}^t(v) = \frac{(g^{t-1})! \prod_{i=0}^{a_i-1} S(a_i, g^{t-1})}{(t-1)!}.$$  

If $\mathcal{P} = \{P_0, \ldots, P_{g-1}\} \in \mathcal{P}_g^n$, we have that:

$$d_{PQIH(n,g)}^t(\mathcal{P}) = \frac{(g^{t-1})! \prod_{i=0}^{g-1} S(|P_i|, g^{t-1})}{(g!)^{t-1}(t-1)!}.$$  

**Proof.** In order for $v$ to appear in an edge of size $t$, it must appear in a corresponding $t$-qualitatively independent set of vectors, say $T$. We count the number of different possibilities for $T$. For each symbol $i \in \mathbb{Z}_g$ in $v$, we have that it must appear with all possible $g^{t-1}$ $(t-1)$-tuples in $\mathbb{Z}_g^{t-1}$. These define the values of the other $t-1$ vectors of $T$ in the coordinates where $i$ appears in $v$. We must then determine the number of ways to assign the $g^{t-1}$ tuples amongst the coordinates of $i$ in $v$ such that every tuple appears at least once: this is equivalent to finding the total number of partitions of the set of $a_i$ coordinates of value $i$ in $v$ into $g^{t-1}$ classes: this is $S(a_i, g^{t-1})$. However, we must label the $g^{t-1}$ classes with specific tuples, and there are $(g^{t-1})!$ ways of doing so.

Taking the product across all possible values of $i \in \mathbb{Z}_g$ gives the numerator of the equation: the fact that via this technique, for every set of $t-1$ vectors $v_1, \ldots, v_{t-1}$ such that $\{v, v_1, \ldots, v_{t-1}\}$ is $t$-qualitatively independent, we count every permutation of $v_1, \ldots, v_{t-1}$ means that we must divide by $(t-1)!$ to eliminate the repeats. This gives the full equation.

The result for partitions follows from the formula for $d_{VQIH(n,g)}^t(v)$ and the fact that since the vectors counted by the formula are all not isolated, each partition that is not isolated corresponds to precisely $g!$ of them. Let $f: \mathbb{Z}_g^n \to \mathcal{P}_g^n$ be the standard map from vectors to partitions, and consider a $t$-qualitatively independent set $e = \{v, v_1, \ldots, v_{t-1}\}$ of vectors. Then $f(e)$ is a $t$-qualitatively independent set of partitions, and as the map is surjective, for each such $t$-qualitatively independent set of partitions, we can find
a $t$-qualitatively independent set of vectors that map to it via $f$. Furthermore, for any of the vectors $v_i$, $1 \leq i \leq t-1$, if we replace $v_i$ in $e$ by any of the $g!$ vectors obtained from rearranging the symbols of $v_i$ to get $e'$, then $f(e) = f(e')$. Additionally, for $1 \leq i < j \leq t-1$, there is no rearrangement of $v_i$ that will yield $v_j$. Thus, each edge $e$ of PQIH($n, g$) containing $P_v$ corresponds to precisely $g!^{t-1}$ edges of VQIH($n, g$) containing $v$, which gives the second formula.

Theorem 3.2.19. In VQIH($n, g$), the number of edges of size $t$ is given by:

$$|E_t(VQIH(n, g))| = \frac{(g^t)!S(n, g^t)}{t!}.$$  

In PQIH($n, g$), the number of edges of size $t$ is:

$$|E_t(PQIH(n, g))| = \frac{(g^t)!S(n, g^t)}{g!^t t!}.$$  

Proof. Consider an edge of size $t$, and write its $t$ vectors as columns of an $n \times t$ matrix:

$$M = [v_0, \ldots, v_{t-1}].$$  

We examine the number of ways to construct such a matrix row-by-row. Each row is an element of $Z_{g^t}^t$, so there are $g^t$ possibilities per row. The Stirling number of the second kind $S(n, g^t)$ denotes the number of ways of partitioning the set $\{0, \ldots, n-1\}$ into $g^t$ groups, which is equivalent to assigning to each row $\{0, \ldots, n-1\}$ the $t$-tuple of $Z_{g^t}$ to which it belongs in the partition. Since this partitioning does not label the $g^t$ groups explicitly, we must do so to associate them with the $t$-tuples of $Z_{g^t}$, and there are $(g^t)!$ ways to do so. Additionally, constructing every possible matrix in this way constructs every permutation of the vectors $v_0, \ldots, v_{t-1}$. We are only interested in edges, or sets of vectors: thus we divide by $t!$ to eliminate the permutations, giving us the formula.

The formula for partitions follows for the formula for $|E_t(VQIH(n, g))|$, and the fact that for each $P \in P_n^g$ there are $g!$ vectors corresponding to $P$. The argument is then the same as in the proof of Theorem 3.2.18. \qed

Theorem 3.2.20. The number of vertices of VQIH($n, g$) that are not in any $t$-edges is

$$\sum_{0 \leq a_0, \ldots, a_{g-1} \leq n} \frac{n!}{a_0! \cdots a_{g-1}!},$$  

subject to $a_0 + \ldots + a_{g-1} = n$ and $\min\{a_i : i \in Z_g\} < g^{t-1}$. \qed
Proof. Consider a vertex \( v \) and for each \( i \in \mathbb{Z}_g \), let \( a_i = |\{0 \leq j < n : v[j] = i\}| \). Then 
\[ 0 \leq a_i \leq n, \text{ and } a_0 + \ldots + a_{g-1} = n. \]
Furthermore, by Theorem 3.2.16, \( v \) is in no edges of size \( t \) if and only if, for some \( i \in \mathbb{Z}_g \), \( a_i < g^{t-1} \), i.e. \( \min\{a_i : 0 \leq i \leq g-1\} < g^{t-1} \).

The number of vertices with these values \( a_0, \ldots, a_{g-1} \) is precisely the number of ways of partitioning the \( n \) labelled positions of the vector into labelled classes of size \( a_0, \ldots, a_{g-1} \), which can be done in precisely \( \frac{n!}{a_0! \cdots a_{g-1}!} \) ways.

Definition 3.2.21. A hypergraph \( H = (V,E) \) is vertex transitive if, for any pair of vertices \( u, v \in V \), there exists a map \( f \in \text{Aut}(H) \) such that \( f(u) = v \).

Theorem 3.2.22. If \( H \) is a hypergraph and \( H \) is vertex transitive, then \( H^* \) is also vertex transitive.

Proof. The commutative diagram in Figure 3.2 illustrates the idea of this proof. Let \( f : H \to H^* \), and consider any pair of vertices \( u, v \in V(H^*) \). Since \( H^* \) is a subhypergraph of \( H \), define \( i : H^* \hookrightarrow H \) to be the inclusion homomorphism. Since \( H^* \) is a core of \( H \), any homomorphism \( H \to H^* \) must be surjective (otherwise the restriction of the homomorphism to \( H^* \) would not be an automorphism), so for any \( v \in V(H^*) \), there exists a vertex \( v' \in V(H) \) such that \( f(v') = v \). Since \( H \) is vertex transitive, there is some automorphism \( a : H \to H \) such that \( a(u) = v' \). Then \( (fa)(H^*) \to H^* \) is a homomorphism, and \( (fa)(u) = (fa)(H^*) = f(v') = v \), as required.

\[
\begin{array}{ccc}
H : u & \xrightarrow{a} & H : v' \\
\downarrow i & & \downarrow f \\
H^* : u & \xrightarrow{ia} & H^* : v
\end{array}
\]

Figure 3.2: Proof of Theorem 3.2.22

Above, we explicitly stated that we no longer consider vertices that appear in no edges. By doing so, we have ensured that every vertex is in some edge, and thus in some facet. However, another type of uninteresting vertices are those that are only in facets of size 1. This happens precisely when, in the case of VQIH(\( n, g \)), a vector has some symbol appear less then \( g \) times, and equivalently, in PQIH(\( n, g \)), when a partition class has size less than \( g \). Any hypergraph with a facet of size 1, say \( \{v\} \) for \( v \in V(H) \), and at least one other facet \( e \), has a diameter of infinity, as there is no path from \( v \) to any vertex in \( e \).
The number of vertices in facets of size 1 can be determined from Theorem 3.2.20. We then have that \( V_{\text{QIH}}(n, g) \) and \( P_{\text{QIH}}(n, g) \) are connected subhypergraphs of \( V_{\text{QIH}}(n, g) \) and \( P_{\text{QIH}}(n, g) \) respectively, each with no vertices in facets of size 1.

In [66], Meagher gives the following result about the diameter of the qualitative independence graph \( [\hat{Q}_{\text{I}}(n, g)]_2 \), which can be applied to our two families of hypergraphs.

**Lemma 3.2.23** (See Lemma 5.2.4 of [66]). When \( n \geq g^2 \), the diameter of \( [\hat{P}_{\text{QIH}}(n, g)]_2 \) is 2.

This result then follows immediately:

**Corollary 3.2.24.** We have that:

1. For any \( n \geq g > 1 \), \( \text{diam}(V_{\text{QIH}}(n, g)) = \text{diam}(P_{\text{QIH}}(n, g)) = \infty \).
2. When \( n \geq g^2 > 1 \), \( \text{diam}(V_{\text{QIH}}(n, g)) = \text{diam}(P_{\text{QIH}}(n, g)) = 2 \).

**Proof.** If \( n \geq g > 1 \), we have that both \( V_{\text{QIH}}(n, g) \) and \( P_{\text{QIH}}(n, g) \) contain multiple facets of size 1, and there is no path between the vertices in these facets, so the diameter of these ASC must be \( \infty \).

As \( [\hat{P}_{\text{QIH}}(n, g)]_2 \) is a spanning subhypergraph of \( \hat{P}_{\text{QIH}}(n, g) \), by Lemma 3.2.23 and Corollary 2.1.17, \( \text{diam}(P_{\text{QIH}}(n, g)) \leq \text{diam}([\hat{P}_{\text{QIH}}(n, g)]_2) = 2 \). As there are always two vertices of \( P_{\text{QIH}}(n, g) \) that are not in an edge together, the diameter is then not 1, so it must be 2.

**Open Question 3.2.1.** Show that \( \text{diam}(V_{\text{QIH}}(n, g)) = 2 \).

One desirable property of covering arrays is for them to consist solely of columns that are *uniform*; Cheng [17] and Meagher, Moura, and Zekaoui [67] independently find CAN\((G, g)\) for certain families of graphs \( G \) using uniform covering arrays and a divide-and-conquer strategy, where pieces of the covering array are built independently and then assembled together over shared, uniform columns. We will similarly use uniform columns to show that uniform variable strength covering arrays over certain families of ASC can be built using a divide-and-conquer strategy.

**Definition 3.2.25.** A partition \( \mathcal{P} \in \mathcal{P}_g^n \) is said to be *uniform* if \( g \mid n \) and all classes in \( \mathcal{P} \) have size exactly \( n/g \). Otherwise, let \( n = gs + r \) for some integers \( s \) and \( 0 < r < g \). If \( g - r \) classes in \( \mathcal{P} \) have size \( s \) and \( r \) classes have size \( s + 1 \), we then say that \( \mathcal{P} \) is *almost-uniform*. A vector \( v \in \mathbb{Z}_g^n \) is *uniform* (respectively *almost-uniform*) if its associated partition \( \mathcal{P}_v \) is uniform (respectively *almost-uniform*).
Proposition 3.2.26. Let \( u = u^n_g \in \mathbb{Z}^n_g \) with \( u[i] = (i \mod g) \). For \( v \in \mathbb{Z}^n_g \), \( v \) is uniform or almost-uniform if and only if there exist permutations \( \sigma \in S_n \), \( \phi \in S_g \) such that \( v = \phi(u^\sigma) \).

Proof. We have that the partition associated with \( u^n_g \) is clearly uniform or almost-uniform, since if \( n = gs + r \) for integers \( s \geq 0 \) and \( 0 \leq r < g \), we have that the symbol \( a \) of \( u^n_g \) appears \( s + 1 \) times if \( 0 \leq a < r \), and \( s \) times otherwise.

If the permutations \( \sigma \) and \( \phi \) exist, since \( v \) is simply a permutation of the positions of \( u^n_g \) according to \( \sigma \) with a possible relabeling \( \phi \), \( P_{u^n_g} = P_v \), so \( v \) is uniform or almost-uniform.

If \( v \) is uniform or almost uniform, then it has symbols \( a_0, \ldots, a_{r-1} \) appearing \( s + 1 \) times in \( v \), and symbols \( a_r, \ldots, a_{g-1} \) appearing \( s \) times in \( v \). Let \( \phi \) be the permutation taking \( a_i \mapsto i \). Then \( \phi(v) \) is simply a permutation of the positions of \( u^n_g \) according to some permutation \( \sigma \in S_n \).

Proposition 3.2.27. If \( v \in \mathbb{Z}^n_g \) is uniform or almost-uniform, then for any \( n' > n \), there is a uniform or almost-uniform vector \( v' \in \mathbb{Z}^{n'}_g \) such that \( v[i] = v'[i] \) for \( 0 \leq i < n \).

Proof. Clearly, \( u = u^n_g \) and \( w = u^n_{g'} \) are uniform or almost uniform. If \( v \) is uniform, by Proposition 3.2.26, there are permutations \( \sigma \in S_n \), \( \phi \in S_g \) such that \( v = \phi(u^\sigma) \). Let \( \gamma \in S_{n'} \) such that \( \gamma(i) = \sigma(i) \) for all \( 0 \leq i < n \). Then \( v' = \phi(u^\gamma) \) has the required properties.

We can then define subhypergraphs of \( \text{VQIH}(n, g) \) and \( \text{PQIH}(n, g) \) consisting of only uniform (or almost uniform) vectors or partitions, respectively.

Definition 3.2.28. Define \( \text{UVQIH}(n, g) \) to be the uniform qualitative independence hypergraph with:

\[
V(\text{UVQIH}(n, g)) = \{ v \in \mathbb{Z}^n_g : v \text{ is uniform or almost uniform} \}
\]

\[
E(\text{UVQIH}(n, g)) = P(V(\text{UVQIH}(n, g))) \cap E(\text{VQIH}(n, g)).
\]

Then clearly \( \text{UVQIH}(n, g) \) is a subhypergraph of \( \text{VQIH}(n, g) \). Similarly, define the hypergraph \( \text{UPQIH}(n, g) \) to be the subhypergraph of \( \text{PQIH}(n, g) \) with:

\[
V(\text{UPQIH}(n, g)) = \{ v \in \mathcal{P}^n_g : v \text{ is uniform or almost uniform} \}
\]

\[
E(\text{UPQIH}(n, g)) = P(V(\text{UPQIH}(n, g))) \cap E(\text{PQIH}(n, g)).
\]
Not surprisingly, for a given \( n \) and \( g \), these two hypergraphs, like the qualitative independence hypergraphs which contain them, are homomorphically equivalent.

**Theorem 3.2.29.** \( \text{UVQIH}(n, g) \) and \( \text{UPQIH}(n, g) \) are homomorphically equivalent.

**Proof.** We follow a similar idea as in Proposition 3.2.12. The map that takes \( v \in \mathbb{Z}_g^n \) to \( \mathcal{P}_v \in \mathcal{P}_g^n \) clearly takes uniform vectors to uniform partitions, and thus by Theorem 3.2.5, it is a homomorphism, so \( \text{UVQIH}(n, g) \rightarrow \text{UPQIH}(n, g) \). The other direction follows from exactly the same idea as in Proposition 3.2.13.

Uniformity, as a property, is interesting to study as it is possible that the uniform qualitative independence hypergraphs may be cores. In particular, Meagher [66] shows the following result, which allows us to show that at least for \( g = 2 \), they are indeed cores.

**Lemma 3.2.30** (See Theorems 5.3.9 and 5.3.10 of [66]). We have that:

\[
[PQIH(n, 2)^\bullet]_2 = [UPQIH(n, 2)]_2.
\]

Furthermore, when \( n \) is even, \( [UPQIH(n, 2)]_2 \cong K_{\frac{n}{2}}^{\frac{n}{2}} \).

**Corollary 3.2.31.** \( UPQIH(n, 2) \) is a core.

**Proof.** By Lemma 3.2.30, \( [UPQIH(n, 2)]_2 \) is a core, so by Theorem 2.1.30, \( UPQIH(n, 2) \) must be a core.

This leaves us with two open problems regarding the connections between qualitative independence hypergraphs and their cores.

**Open Question 3.2.2.** Is \( PQIH(n, 2)^\bullet = UPQIH(n, 2) \)? That is, is there a homomorphism \( PQIH(n, 2) \rightarrow UPQIH(n, 2) \)? For qualitative independence graphs, this was proven using chains of posets, as shown in [66]. Can something similarly be done for this problem?

**Open Question 3.2.3.** Is \( UPQIH(n, g) \) a core? If so, is \( PQIH(n, g)^\bullet = UPQIH(n, g) \)?
3.3 Mixed-Level Variable Strength Covering Arrays

As discussed in Section 2.2, restricting all factors of a covering array to have the same set of levels is overly restrictive and does not accurately model real world systems. We extend the notion of qualitative independence to allow for varying numbers of levels and lay the ground for the theory of mixed-level variable strength covering arrays around this.

Definition 3.3.1. Consider two vectors $v_0 \in \mathbb{Z}^{n}_{g_0}$, $v_1 \in \mathbb{Z}^{n}_{g_1}$ where $v_i[x]$ denotes the value in the $x$th coordinate of $v_i$. We say that $v_0$ and $v_1$ are qualitatively independent if, for every $(x_0, x_1) \in \mathbb{Z}_{g_0} \times \mathbb{Z}_{g_1}$, there exists an integer $0 \leq r < n$ such that $(v_0[r], v_1[r]) = (x_0, x_1)$. By extension, a set of $t$ vectors $v_0 \in \mathbb{Z}^{n}_{g_0}, \ldots, v_{t-1} \in \mathbb{Z}^{n}_{g_{t-1}}$ is said to be $t$-qualitatively independent if, for every $t$-tuple $(x_0, \ldots, x_{t-1}) \in \mathbb{Z}_{g_0} \times \ldots \times \mathbb{Z}_{g_{t-1}}$, there exists an integer $0 \leq r < n$ such that $(v_0[r], \ldots, v_{t-1}[r]) = (x_0, \ldots, x_{t-1})$.

As we did in Section 3.2, we now give an alternate definition of mixed-level covering arrays that uses qualitative independence to capture the coverage conditions on the columns.

Definition 3.3.2. A mixed-level covering array $\text{MCA}(n; t, (g_0, \ldots, g_{k-1}))$ is an $n \times k$ array $M = [m_0, m_1, \ldots, m_{k-1}]$ such that:

1. $m_i \in \mathbb{Z}^{n}_{g_i}$.
2. For $\{i_0, \ldots, i_{t-1}\} \subseteq \{0, \ldots, k-1\}$, we have that the set of vectors $S = \{m_{i_0}, \ldots, m_{i_{t-1}}\}$ is $t$-qualitatively independent.

We denote the mixed-level covering array number $\text{MCAN}(t, (g_0, \ldots, g_{k-1}))$ to be the smallest $n$ such that a MCA($n; t, (g_0, \ldots, g_{k-1})$) exists.

Alternatively, we can represent the $k$-tuple $(g_0, \ldots, g_{k-1})$ as a weight function, written as $w: \{0, \ldots, k-1\} \rightarrow \mathbb{N}^+$, such that $w(i) = g_i$, in which case we write $\text{MCA}(n; t, k, w)$. We write $w$ in vector notation to differentiate it from a scalar suggesting uniform number of levels.

Definition 3.3.3. A weighted hypergraph $H$ is a hypergraph with a weight function defined on its set of vertices, say $w: V(H) \rightarrow \mathbb{N}^+$. If $V(H) = \{v_0, \ldots, v_{k-1}\}$, for convenience, we may list the weights as a $k$-tuple $(g_0, \ldots, g_{k-1})$ where $g_i = w(v_i)$.
**Definition 3.3.4.** If we have two weighted hypergraphs $H$ and $G$ with weight functions $w_H$ and $w_G$ respectively, we say that $f : V(H) \rightarrow V(G)$ is a non-weight-decreasing homomorphism if $f$ is a homomorphism and $w_H(v) \leq w_G(f(v))$. We write $f : H \rightarrow^W G$.

**Definition 3.3.5.** Let $H$ be a weighted hypergraph with weight function $w$, and write $k = |V(H)|$. A mixed-level variable-strength covering array, denoted $VCA(n; H, w)$, is an $n \times k$ array $M = [m_0, \ldots, m_{k-1}]$ such that:

1. $m_i \in \mathbb{Z}^n_{w(i)}$.
2. For $e = \{v_0, \ldots, v_{t-1}\} \in E(H)$, the set of vectors $S = \{m_{v_0}, \ldots, m_{v_{t-1}}\}$ is $t$-qualitatively independent.

The mixed-level variable-strength covering array number, written $VCAN(H, w)$, is the smallest $n$ such that a $VCA(n; H, w)$ exists.

**Definition 3.3.6.** For $n$ and $g_0, \ldots, g_{s-1} \in \mathbb{N}^+$, define the mixed-level qualitative independence hypergraph $MIH(n, \{g_0, \ldots, g_{s-1}\})$ to be the weighted hypergraph with $V(H) = \mathcal{P}^n_{g_0} \cup \ldots \cup \mathcal{P}^n_{g_{s-1}}$ and weight $g_i$ on the vertices of $\mathcal{P}^n_{g_i}$, with $\{P_0, \ldots, P_{t-1}\} \in E(H)$ if and only if it is $t$-qualitatively independent.

**Theorem 3.3.7.** Let $H$ be a weighted hypergraph with weights $(g_0, \ldots, g_{k-1})$. Then we have that a $VCA(n; H, (g_0, \ldots, g_{k-1}))$ exists if and only if $H \rightarrow^W MIH(n, \{g_0, \ldots, g_{k-1}\})$.

**Proof.** If $f : H \rightarrow^W MIH(n, \{g_0, \ldots, g_{k-1}\})$ is a homomorphism, for a vertex $v_i \in V(H)$ with $g_i$ levels, set the $i$th column of the array to any vector $u$ associated with the partition $f(v_i)$ of weight $w$ under the image of the homomorphism, which can be done by arbitrarily labeling the classes of the partition from $\mathbb{Z}_w$. If $w > g_i$, replace the symbols of $u$ that are greater than $g_i - 1$ in any way so that $u \in \mathbb{Z}^n_{g_i}$. The resultant collection of vectors is a $VCA(n; H, (g_0, \ldots, g_{k-1}))$.

If an $VCA(n; H, (g_0, \ldots, g_{k-1}))$ exists, take vertex $i \in V(H)$ to the partition defined by the $i$th column vector of the mixed-level covering array. The result is a weighted homomorphism from $H \rightarrow^W MIH(n, \{g_0, \ldots, g_{k-1}\})$.

### 3.4 Modifications and Constructions

Due to the fact that strength is dictated by the edges of a hypergraph in variable-strength covering arrays, since edge size is not fixed, it is possible to modify the factors in certain
ways while making changes to the underlying hypergraph and still preserving the strength requirements of the original problem. This offers a great deal of flexibility in how we may approach several problems, using the two operations of factor splitting and joining as defined below.

### 3.4.1 Factor Splitting and Joining

Given a weighted hypergraph, we detail two modifications to the hypergraph called factor splitting and factor joining, and investigate the effect that these two operations have on the size of a mixed-level variable strength covering array over the hypergraph. When discussing a mixed-level independence hypergraph MIH, for convenience, we consider the vertices interchangeably as partitions and vectors, where the vector associated with a partition is any of the vectors that map to the partition.

![Figure 3.3: Split: VCAN(H, (ab,g₁,...,gₖ₋₁)) = VCAN(G, (a,b,g₁,...,gₖ₋₁)).](image)

The general idea of the following theorem is illustrated in Figure 3.3.

**Theorem 3.4.1** (Factor splitting). Let $H$ be a weighted hypergraph with weight function $w: V(H) \to \mathbb{N}^+$. If, for some $v \in V(H)$, we have that $w(v) = ab$ for $a, b > 1$, define the weighted hypergraph $H'$:

1. $V(H') = (V(H) \setminus \{v\}) \cup \{v', v''\}$. 


2. For every \( e \in E(H) \), if \( v \not\in e \), then \( e \in E(H') \). If \( v \in e \), then \( (e \setminus \{v\}) \cup \{v', v''\} \in E(H') \).

and the weight function \( w': V(H') \to \mathbb{N}^+ \):

\[
w'(u) = \begin{cases} w(u) & \text{if } u \neq v', v'' \\ a & \text{if } u = v' \\ b & \text{if } u = v''. \end{cases}
\]

Then we have that \( VCAN(H, w) = VCAN(H', w') \).

Proof. We begin by showing that \( VCAN(H', w') \leq VCAN(H, w) \). Write \( k = |V(H)| \), and assume that \( M \) is an \( n \times k \) array with columns indexed by \( V(H) \) such that \( M \) is a \( VCA(n; H, w) \). For \( u \in V(H) \), \( 0 \leq j < n \), write \( m_{uj} \) to denote the \( j \)th entry in the column of \( M \) corresponding to the vertex \( u \) of \( H \). Define the following \( n \times (k + 1) \) array \( M' \) with columns indexed by \( V(H') \), where \( m'_{uj} \) denotes the \( j \)th entry in the column corresponding to the vertex \( u \) of \( H' \):

\[
m'_{uj} = \begin{cases} m_{uj} & \text{if } u \neq v', v'' \\ \lfloor m_{uj} / b \rfloor & \text{if } u = v' \\ m_{uj} \mod b & \text{if } u = v''. \end{cases}
\]

We now claim that \( M' \) is a \( VCA(n; H', w') \). By construction, for \( u \in V(H) \cap V(H') \), we have that the entries in column \( u \) are from \( \mathbb{Z}_{w(u)} \) as required. If \( u = v' \), then, by construction, since the entries from column \( v \) in \( M \) are from \( \mathbb{Z}_{ab} \), the entries from column \( v' \) are from \( \mathbb{Z}_a = \mathbb{Z}_{w(v')} \), as required. If \( u = v'' \), the entries are from \( \mathbb{Z}_b = \mathbb{Z}_{w(v'')} \) as required. Now, if \( e' \in E(H') \), we have two cases to consider:

1. \( e' \in E(H) \). Then \( v, v', v'' \notin e' \), so the columns corresponding to the vertices of \( e' \) in \( M' \) are precisely the same columns as in \( M \), so the \( n \times |e'| \) subarray of \( M' \) over the columns of \( e' \) is equal to the \( n \times |e'| \) subarray of \( M \) over the columns of \( e' \). Since the columns of the subarray are \( |e'| \)-qualitatively independent in \( M \), they are also in \( M' \).

2. \( e' \notin E(H) \). This can only happen if \( v', v'' \in e' \), and \( e = (e' \setminus \{v', v''\}) \cup \{v\} \in E(H) \).

Let \( A \) be the \( n \times |e| \) subarray of \( M \) with columns corresponding to the vertices of \( e \). Then this must be \( |e| \)-qualitatively independent. Consider any interaction \( I' \) over
e', and for \( u \in e \), write \( i'_u \) to denote the assignment of level to \( u \) in \( I' \). We now define an interaction \( I \) over \( e \), writing \( i_u \) to denote the assignment of level to \( u \) in \( I \) as follows:

\[
  i_u = \begin{cases}
    i'_u & \text{if } u \neq v \\
    i'_v b + i''_v & \text{if } u = v.
  \end{cases}
\]

Then since \( M \) is a VCA(\( n; H, w \)), there is some row, say \( r \), such that the columns of \( e \) assume the values of \( I \) in \( M \). Then, by construction, the columns of \( e' \) assume the values of \( I' \) in row \( r \) of \( M' \).

Thus, \( M' \) is a VCA(\( n; H', w' \)).

We now show the converse, namely that VCAN(\( H, w \)) \( \leq \) VCAN(\( H', w' \)). Assume that \( M' \) is an \( n \times (k + 1) \) array with columns indexed by \( V(H') \) such that \( M' \) is a VCA(\( n; H', w' \)), and for any \( u \in V(H') \), \( 0 \leq j < n \), write \( m'_{uj} \) to denote the \( j \)th entry of the column of \( M' \) corresponding to \( u \). Define the following \( n \times k \) matrix \( M \) with columns indexed by \( V(H) \), where \( m_{uj} \) denotes the \( j \)th entry in the column corresponding to \( u \):

\[
  m_{uj} = \begin{cases}
    m'_{uj} b + m''_{uj} & \text{if } u = v \\
    m'_{uj} & \text{if } u \neq v.
  \end{cases}
\]

We now claim that \( M \) is a VCA(\( n; H, w \)). Consider any edge \( e \in E(H) \). We have the following cases:

1. \( e \in E(H') \). Then \( v, v', v'' \not\in e \), so by the same argument as above, \( e \) is \(|e|\)-qualitatively independent in \( M \).

2. \( e \not\in E(H') \). Then, by construction, this happens only when \( v \in e \), and \( e' = (e \setminus \{v\}) \cup \{v', v''\} \in E(H') \). Consider any interaction \( I = (i_u) \in \prod_{u \in e} \Z_{w(u)} \) over \( e \), and define the following interaction \( I' = (i'_u) \in \prod_{u \in e'} \Z_{w'(u)} \) over \( e' \):

\[
  i'_u = \begin{cases}
    i_u & \text{if } u \neq v', v'' \\
    \left\lfloor \frac{i_u}{b} \right\rfloor & \text{if } u = v' \\
    i_v \mod b & \text{if } u = v''.
  \end{cases}
\]

Since \( M' \) is a VCA(\( n; H', w' \)), there is a row, say \( r \), such that the columns of \( e' \) assume the values of \( I' \) in \( M' \). Then, by construction, the columns of \( e \) assume the values of \( I \) in row \( r \) of \( M \).
Thus, $M$ is a VCA($n; H, w$). Hence, VCAN($H, w$) = VCAN($H', w'$).

**Example 3.4.2.** If we have hypergraph $H$ with $V(H) = \{0, 1, 2\}$, $w(H) = (2, 2, 4)$, and $E(H) = \{\{0, 1\}\}$, then we have a VCA(16; $H, (2, 2, 4)$):

$$M = \begin{bmatrix}
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3
\end{bmatrix}^T.$$ 

We can split parameter 2, originally over four levels, into two parameters $2'$ and $2''$, each with two levels. This gives hypergraph $H'$ with $V(H') = \{0, 1, 2', 2''\}$, $w(H') = (2, 2, 2, 2)$, and $E(H') = \{\{0, 1, 2', 2''\}\}$, with VCA(16; $H', (2, 2, 2, 2)$):

$$M = \begin{bmatrix}
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix}^T.$$ 

**Theorem 3.4.3** (Factor joining). Let $H$ be a weighted hypergraph with weight function $w: V(H) \to \mathbb{N}$. Consider any two distinct vertices $v', v'' \in V(H)$ with $w(v') = a$ and $w(v'') = b$. Define the following weighted hypergraph $H'$:

1. $V(H') = (V(H) \setminus \{v', v''\}) \cup \{v\}$. 

An example of the application of the following theorem is illustrated in Figure 3.4.

**Figure 3.4:** Join: VCAN($H, (a, b, g_2, \ldots, g_{k-1})$) $\leq$ VCAN($G, (ab, g_2, \ldots, g_{k-1})$).
2. For every edge \( e \in E(H) \), if \( v', v'' \not\in e \), then \( e \in E(H') \). If \( \{v', v''\} \cap e \neq \emptyset \), then 
\((e \setminus \{v', v''\}) \cup \{v\} \in E(H') \).

and the weight function \( w' : V(H') \to \mathbb{N} : \)

\[
w'(u) = \begin{cases} w(u) & \text{if } u \neq v \\ ab & \text{if } u = v. \end{cases}
\]

Then we have that \( \text{VCAN}(H, w) \leq \text{VCAN}(H', w') \). Furthermore, if, for every \( e \in E(H) \), 
\( v' \in e \) if and only if \( v'' \in e \), then \( \text{VCAN}(H, w) = \text{VCAN}(H', w') \).

Proof. Write \( k = |V(H')| \), and assume that \( M' \) is an \( n \times k \) array with columns indexed by \( V(H') \) such that \( M' \) is a VCA\((n; H', w')\). For \( u \in V(H') \), \( 0 \leq j < n \), write \( m'_{uj} \) to denote the \( j \)th entry in the column of \( M' \) corresponding to the vertex \( u \) of \( H' \). Define the following \( n \times (k + 1) \) array \( M \) with columns indexed by \( V(H) \), where \( m_{uj} \) denotes the \( j \)th entry in the column corresponding to the vertex \( u \) of \( H \):

\[
m_{uj} = \begin{cases} m'_{uj} & \text{if } u \neq v', v'' \\ \lfloor \frac{m'_{uj}}{b} \rfloor & \text{if } u = v' \\ m'_{uj} \mod b & \text{if } u = v''. \end{cases}
\]

We now claim that \( M \) is a VCA\((n; H, w)\). By construction of \( w' \), we have that, for all 
\( u \in V(H) \setminus \{v', v''\} \), the entries in column \( u \) are from \( \mathbb{Z}_{w(u)} = \mathbb{Z}_{w(u)} \) as required. If 
\( u = v' \), then the entries in column \( v' \) are of the form \( \lfloor \frac{u}{b} \rfloor \) for some \( 0 \leq u < ab \), and thus 
\( 0 \leq \lfloor \frac{u}{b} \rfloor < a \), so the values can be considered to be from \( \mathbb{Z}_{w(v')} = \mathbb{Z}_a \), as required. If 
\( u = v'' \), the entries in column \( v' \) are of the form \( u \mod b \), so they can be considered to be from \( \mathbb{Z}_{w(v'')} = \mathbb{Z}_b \).

Now, consider each edge \( e \in E(H) \). We have the following cases:

1. \( e \in E(H') \). Then, since the \( n \times |e| \) subarray over the columns of \( e \) in \( M' \) is \(|e|\)-qualitatively independent since \( M' \) is a VCA\((n; H', w')\), and this subarray remains unchanged in \( M \), we have that the subarray is also \(|e|\)-qualitatively independent in \( M \).

2. \( e \not\in E(H') \). Then we must have that \( e' = (e \setminus \{v', v''\}) \cup \{v\} \in E(H') \). Let 
\( e_a = e \cup \{v', v''\} \supseteq e \), and consider any interaction \( I = (i_u) \in \prod_{u \in e_a} \mathbb{Z}_{w(u)} \). Define
the interaction $I' = (i'_u) \in \prod_{u \in e'} \mathbb{Z}_{w'(u)}$: 

$$i'_u = \begin{cases} 
  i_u & \text{if } u \neq v', v'' \\
  \lfloor \frac{i_u}{b} \rfloor & \text{if } u = v' \\
  i_v \mod b & \text{if } u = v''. 
\end{cases}$$

Then as $e' \in E(H')$, there is some row $r'$ of $M'$ where the values over the columns in $e'$ correspond with the interaction $I'$. Then, by construction, in the row $r'$ of $M$, the values over the columns in $e_a$ correspond with the interaction $I$. Since every interaction over $e$ is a subinteraction of some interaction over $e_a$ since $e \subseteq e_a$, we get the desired result.

Thus, $M$ is a VCA($n; H, w$).

If, for every $e \in E(H)$, $v' \in e$ if and only if $v'' \in e$, then this is precisely the reverse of the split operation in Theorem 3.4.1, giving that VCAN($H, w$) = VCAN($H', w'$).

In the proof of Theorems 3.4.1 and 3.4.3 we used the bijection:

$$\mathbb{Z}_{ab} \rightarrow \mathbb{Z}_a \times \mathbb{Z}_b$$

$$x \mapsto \left( \left\lfloor \frac{x}{b} \right\rfloor, x \mod b \right)$$

However, we note that any bijection $\mathbb{Z}_{ab} \rightarrow \mathbb{Z}_a \times \mathbb{Z}_b$ could be substituted.

This strategy allows for a large degree of flexibility in its use, especially when combined with dropping levels. The following scenario is illustrated in Figure 3.5. If, for example, we have a factor $f$ over 5 levels and a factor $f'$ over 6 levels in an edge, here are several strategies we could use to build a covering array including these factors:

- Treat them as they are in the problem.
- Join them together to form a single factor over $5 \times 6 = 30$ levels.
- Replace them by five factors over 2 levels. Then combine these five factors to get a single factor over 32 levels. We drop the number of levels by 2 to 30 by replacing every occurrence of level 30 and 31 by any level $0, \ldots, 29$. Finally, split this factor over 30 levels into the original two factors.

In order to motivate the usefulness of the join operation, we now give two examples where first joining vertices provides us with a preferable way to look at a variable strength covering array problem.
Example 3.4.4. Assume that we have a set of five factors $F_5 = \{0, 1, 2, 3, 4\}$, each with 5 levels, and a set of three factors $F_3 = \{5, 6, 7\}$, each with 3 levels. Take the ASC $\Delta$ with facet set $\Lambda$ as follows:

$$\Lambda = \binom{F_5}{2} \cup \{\{v, 5, 6, 7\} : v \in F_5\}.$$ 

It is not immediately obvious how to determine $\text{VCAN}(\Lambda, 5^5 3^3)$; however, if instead we join all the factors in $F_3$ to become a single factor $F$ over $3^3 = 27$ levels, then the facet hypergraph $\Lambda'$ of the new ASC is 2-uniform, and thus a graph isomorphic to $K_6$, so we have that $\text{VCAN}(\Lambda, 5^5 3^3) = \text{MCAN}(2, 6, 5^5 27)$. Begin with an OA(2, 6, 5), which we can construct using Theorem 2.2.20. By using a technique from [71], we augment one of the factors from 5 levels to 27 levels, which can be done optimally. This gives the MCA(135; 2, 6, 5^5 27), and by using a bijection $\mathbb{Z}_{27} \rightarrow \mathbb{Z}_3 \times \mathbb{Z}_3 \times \mathbb{Z}_3$, we derive the VCA(135; $\Lambda, 5^5 3^3$), which must be optimal.

Example 3.4.5. Another example occurs when we have a weighted hypergraph $H = (V, E)$ with weight function $w : V \rightarrow \mathbb{N}^+$ dictating the levels for each vertex. Let:

$$m = \max \left\{ \prod_{e \in \Lambda} w(e) : e \in \Lambda \right\}.$$
Assume $H$ is not a hypertree. By iteratively performing join operations on vertices of $H$, we can always derive a hypertree $H' = (V', E')$ with weight function $w': V \rightarrow \mathbb{N}^+$. Let:

$$m' = \max \left\{ \prod_{e \in E'} w'(e) : e \in \Lambda' \right\}.$$  

In [17], Cheng tells us how to build a VCA($m'; H', w'$) for a hypertree $H'$, and by using the appropriate bijections, we obtain a VCA($m'; H, w$). Furthermore, if $m = m'$, then we know that this array is optimal.

We demonstrate an example of this technique in Figure 3.6 to show a case where, in the original hypergraph, $m = 3 \cdot 5 \cdot 4 \cdot 2 = 120$. We perform five join operations to get a hypertree with $m' = 4 \cdot 2 \cdot 15 = 120$, and by constructing an array of size $m'$ over this hypertree, we can then transform it into an array over the original hypergraph.

We now motivate the splitting operation by giving an example that illustrates a situation where, by first splitting a factor, we reduce a variable strength covering array problem to something we can solve via a direct construction, which provides a solution to the original problem.

**Example 3.4.6.** We attempt to find MCAN(3, (8, 8, 8, 8, 3)). This is equivalent to finding a VCA($K^3_4$, (8, 8, 8, 8, 3)). We can split the factor with 8 levels into three factors, each with 8 levels. The modification takes $K^3_4$ to $H = (\mathbb{Z}_6, \{012, 01345, 02345, 12345\})$, and we now have the equivalent problem of finding VCAN($H, 8$). We know, simply by the number of required interactions to cover any of the strength-5 edges, that VCAN($H, 8$) $\geq 8^5$. Using the Bush construction of Lemma 2.2.22, we have that a CA($8^5; 5, 9, 8$) exists. By dropping three factors, we have a CA($8^5; 5, 6, 8$), which is still optimal. Then, this is also a VCA($8^5; H, 8$), which must be optimal. By using a bijection $\mathbb{Z}_8 \times \mathbb{Z}_8 \times \mathbb{Z}_8 \rightarrow \mathbb{Z}_{8^3}$, we derive the mixed-level covering array, giving that MCAN(3, (8, 8, 8, 8, 3)) = $8^5$.

### 3.4.2 Uniform Arrays, Cut-Vertices, and Cut-Cliques

We now show how we can exploit uniform columns in covering arrays to build covering arrays over hypergraphs using a divide-and-conquer approach. We begin by translating the concept of uniformity over vectors and partitions to uniformity of covering arrays.

**Definition 3.4.7.** Let $H$ be a weighted hypergraph with weights $(g_0, \ldots, g_{k-1})$, and let $M = [m_0, \ldots, m_{k-1}]$ be a VCA($n; H, (g_0, \ldots, g_{k-1})$). If, for all $0 \leq i < k$, we have that
Figure 3.6: Turning a hypergraph into a hypertree through a series of join operations. Vertices of the same shape (excluding circles) in the original hypergraph are joined to obtain the final hypertree.
$m_i$ is uniform, then we say that $M$ is uniform. If, for all $0 \leq i < k$, we have that $m_i$ is uniform or almost-uniform, we then call $M$ almost-uniform.

Lemma 3.4.8. If we have a uniform or almost-uniform VCA($n; H, (g_0, \ldots, g_{k-1})$), then, for any $n' \geq n$, there exists a uniform (if $g_i | n'$ for all $0 \leq i < k$) or almost-uniform (otherwise) VCA($n'; H, (g_0, \ldots, g_{k-1})$).

Proof. This follows immediately from Proposition 3.2.27, simply by extending each column vector independently.

Theorem 3.4.9. A uniform or almost-uniform VCA($n; H, g$) exists if and only if $H \rightarrow UVQIH(n, g)$.

Proof. The proof is the same as for Theorem 3.2.9.

Definition 3.4.10. For a hypergraph $H = (V, E)$, a vertex $u \in V$ is a cut-vertex of $H$ if $E$ can be partitioned into two classes, $A$ and $B$, such that $V(A) \cap V(B) = \{u\}$. Furthermore, if we have a cut-vertex $u$ and a partition $P = \{P_0, \ldots, P_{s-1}\}$ of $E$, $s \geq 2$, such that for any pair $0 \leq i < j < s$, $V(P_i) \cap V(P_j) = \{u\}$, then we say that $u$ separates $P$.

We now demonstrate that we can divide-and-conquer recursively on cut-vertices of a hypergraph, build uniform or almost-uniform covering arrays over the components, and then assemble them to give a covering array over the whole hypergraph. The idea behind the construction is illustrated in Figure 3.7.

Theorem 3.4.11. Let $H = (V, E)$ be a weighted hypergraph (write $k = |V|$) with weight function $w: V(G) \rightarrow \mathbb{N}^+$, cut-vertex $u$, and a partition $P = \{P_0, \ldots, P_{s-1}\}$ of $E$, $s \geq 2$, such that $u$ separates $P$. If, for each $P_i \in P$, we have a uniform or almost-uniform VCA($n_i; P_i, w|_{V(P_i)}$), there exists a uniform (if $w(i) | n$ for all $0 \leq i < k$) or almost-uniform (otherwise) VCA($n; H, w$) where $n = \max\{n_0, \ldots, n_{s-1}\}$.

Proof. Let $A_i$ be a uniform or almost-uniform VCA($n_i; P_i, w$) for all $i \in \{0, \ldots, s-1\}$. By Lemma 3.4.8, we can extend $A_i$ to a uniform or almost-uniform VCA($n; P_i, w$) $A'_i$. We can then permute the rows of $A'_i$ and the values of the columns for the cut vertex so that each such column corresponding to $u$ is precisely the vector $x = (x_i)^T$ where $x_i = i \pmod{w(u)}$.

Let $k = |V|$. We now construct an $n \times k$ matrix $A$ as follows: for the $j$-th column of $A$, for $j \neq u$, find the partition $P_i$ such that $j \in V(P_i)$. As $V(P_a) \cap V(P_b) = \{u\}$ for
all $0 \leq a < b \leq s - 1$, there is exactly one such $P_i$, so this is well-defined. Take the column of the $A'_i$ corresponding to vertex $j$. For the column corresponding to $u$, we have that by our permutation on the rows of each $A'_i$ and value permutations on the columns corresponding to $u$ in each, all of the columns for $u$ are identical in $A'_0, \ldots A'_{s-1}$, i.e. $x$ as defined above. Thus, take $x$ for the $u$th column of $A$.

We now claim that $A$ is a uniform or almost-uniform VCA($n; H, w$). It is clearly uniform or almost-uniform, since all of the columns are taken from uniform or almost-uniform covering arrays. Furthermore, it is a covering array as it covers all hyperedges $E$ since $\mathcal{P}$ partitions $E$.

The conditions of Theorem 3.4.11 can be relaxed to only require that the frequency of the symbols for the columns corresponding to the cut vertex $u$ be the same in each
of the $A_i$; we can then still apply the same proof, as the arrays $A_i$ can combined in the
same way to give the final covering array.

Uniformity can be extended beyond single columns to a higher-strength notion that
includes multiple columns. This is a much more demanding requirement, but allows us
to generalize the cut-vertex divide-and-conquer strategy to cut-cliques.

**Definition 3.4.12.** Let $n$ and $g_0, \ldots, g_{s-1}$ be integers, and consider a set of vectors
$U = \{v_0, \ldots, v_{s-1}\}$ with $v_i \in \mathbb{Z}_{g_i}^n$. Denote the set of all tuples over the base sets of the
vectors as follows:

$$T = \prod_{i=0}^{s-1} \mathbb{Z}_{g_i}$$

and for convenience, write $g = |T| = g_0 g_1 \ldots g_{s-1}$. We then say that $U$ is $s$-uniform
(respectively $s$-almost-uniform) if the array $[v_0, \ldots, v_{s-1}]$ contains every tuple from $T$ as
a row precisely $n/g$ times (respectively $\lfloor n/g \rfloor$ or $\lfloor n/g \rfloor + 1$ times).

Just as, by Proposition 3.2.27, we can extend the length of a uniform vector and
maintain the property of uniformity, we can do the same thing with a set of $s$-uniform
vectors.

**Proposition 3.4.13.** If, for $U = \{v_0, \ldots, v_{s-1}\}$ with $v_i \in \mathbb{Z}_{g_i}^n$, we have that $U$ is $s$-uniform,
then for any $n' \geq n$, there exists a set of vectors $U' = \{u_0, \ldots, u_{s-1}\}$ with
$u_i \in \mathbb{Z}_{g_i}^{n'}$ such that, for all $0 \leq i < s$, $0 \leq j < n$, $u_i[j] = v_i[j]$, with the property that if
$g|n'$, $U'$ is $s$-uniform, and otherwise, $U'$ is $s$-almost-uniform.

**Proof.** Consider the $n \times s$ matrix $N = [v_0, \ldots, v_{s-1}]$, where the columns are $s$-uniform.
Let $g = g_0 g_1 \ldots g_{s-1}$. Define the set of all tuples over the base sets of the vectors:

$$T = \prod_{i=0}^{s-1} \mathbb{Z}_{g_i}.$$

Take any ordering on the tuples and, for $0 \leq i < g$, write $t_i$ to be the $i$th tuple with
respect to this ordering. For $0 \leq i < s$, define $u'_i \in \mathbb{Z}_{g_i}^{n'}$ such that for $0 \leq j < n'$,
$u'_i[j] = t_j \mod g[i]$.

Consider the matrix $M = [u'_0, \ldots, u'_{s-1}]$. If $g|n'$, by construction, every $t_i$ appears
as a row in the matrix precisely $n'/g$ times, and if not, every $t_i$ appears as a row either
$\lfloor n'/g \rfloor$ or $\lfloor n'/g \rfloor + 1$ times, so the columns of $M$ are $s$-uniform or $s$-almost-uniform as
required.
Furthermore, since \( g \mid n \), it is easy to see that the vectors in the \( n \times s \) subarray \( M' \) of \( M \) are \( s \)-uniform by construction, so there exists some permutation \( \sigma \in S_n \) that permutes the rows of \( M' \) to give the matrix \( N \). By extending \( \sigma \) to a permutation \( \sigma' \in S_{n'} \) where, for \( 0 \leq i < n \), \( \sigma'(i) = \sigma(i) \), and for \( n \leq i < n' \), \( \sigma'(i) = i \), the action of \( \sigma' \) to the rows of \( M \) results in a set of column vectors that are \( s \)-uniform or \( s \)-almost-uniform that extends \( N \), as required.

We now define the notion of a \( t \)-cut-clique of size \( c \), which is the higher dimensional notion of a cut-vertex in our generalization of the construction.

**Definition 3.4.14.** Let \( H = (V, E) \) be a hypergraph, and let \( C \subseteq V \) be a \( t \)-clique of size \( c \) in \( H \). If we have a partition \( \mathcal{P} = \{P_0, \ldots, P_{s-1}\} \) of \( E \), \( s \geq 2 \), such that for all \( 0 \leq i < j < s \), we have that \( V(P_i) \cap V(P_j) = C \), then we say that \( C \) separates \( \mathcal{P} \), and we call \( C \) a \( t \)-cut-clique of size \( c \) of \( H \).

We now give the construction. Unlike the construction for cut-vertices, we require the subarrays over the cut-clique of size \( c \) to be \( c \)-uniform. In the cut-vertex construction, we could build the arrays over the components separately, extend them to the same size, and paste them together because we could always transform the arrays over the components to a form where they had the same column over the cut-vertex and were thus compatible. In the case of cut-cliques, this does not work in the almost-uniform case. As an example, we have the following two sets of 2-almost-uniform column vectors over \( \mathbb{Z}_6^2 \), but there is no way to transform \( U \) and \( V \) to be compatible: there is no permutation of rows and of values over the columns of \( U \) that will map it to \( V \).

\[
U = \begin{bmatrix}
0 & 0 \\
0 & 0 \\
1 & 1 \\
1 & 1 \\
0 & 1 \\
1 & 0
\end{bmatrix}, \quad V = \begin{bmatrix}
0 & 0 \\
0 & 0 \\
1 & 1 \\
0 & 1 \\
1 & 0
\end{bmatrix}
\]

Thus, unlike the cut-clique case, almost-uniformity is not enough for the generalized construction.

**Theorem 3.4.15.** Let the following conditions hold:

1. Let \( H = (V, E) \) be a weighted hypergraph equipped with weight function \( w : V(G) \to \mathbb{N}^+ \).
2. Let $C \subset V$ be a $t$-cut-clique of size $c$ in $H$, and write $E(C) = \mathcal{P}(C) \cap E$.

3. Let $\mathcal{P} = \{P_0, \ldots, P_{s-1}\}$ be a partition of $E$, $s \geq 2$, such that $C$ separates $\mathcal{P}$.

Write $H_i$ to be the subhypergraph $(V(P_i), P_i \cup E(C))$. If, for each $P_i \in \mathcal{P}$, we have a VCA$(n_i, H_i, w|_{V(H_i)})$ whose columns corresponding to $C$ form a $c$-uniform set of vectors, then for $n = \max\{n_0, \ldots, n_{s-1}\}$, there exists a VCA$(n; H, w)$ whose columns corresponding to $C$ form a $c$-uniform set of vectors.

Proof. The idea is identical to that of Theorem 3.4.11. For each $P_i \in \mathcal{P}$, let $A_i$ be a VCA$(n_i; H_i, w|_{V(H_i)})$ whose columns corresponding to $C$ form a $c$-uniform set of vectors. We then extend this, by Proposition 3.4.13 to extend the columns in $C$ and arbitrarily filling in the positions over the other columns, to get $A'_i$ of size $n$ that is $c$-uniform on $C$. We can then assemble the $A'_i$ by sorting the rows according to any ordering on the columns in $C$, and then the $n \times c$ subarray on $C$ is identical for each $A'_i$, so they can be combined to get a full VCA$(n; H, w)$.

We note that the construction does allow a component VCA$(n_i; P_i \cup E(C), w)$ to be $c$-almost uniform, provided it can be extended to a $c$-uniform array of size $n$.

3.4.3 Constructions for Consecutive Covering Arrays

We now look at the problem of constructing consecutive and cyclic consecutive covering arrays, and identify cases in which we can build provably optimal arrays.

Lemma 3.4.16. $H^{k,t}_c$ is vertex transitive.

Proof. For any integer $i$, the map $\sigma_i: \{0, \ldots, k-1\} \rightarrow \{0, \ldots, k-1\}$ defined by $\sigma_i(v) = (v+i) \mod k$ is an automorphism of $H^{k,t}_c$, so for any two vertices $u, v \in \{0, \ldots, k-1\}$ the automorphism $\sigma_{v-u}$ takes vertex $u$ to $v$. \qed

Theorem 3.4.17. $CCAN(t, k, g) = g^t$.

Proof. Let $H = (\mathbb{Z}_t, \{\mathbb{Z}_t\})$ be a hypergraph. Then the map $\{0, \ldots, k-1\} \rightarrow \mathbb{Z}_t$ defined by $v \mapsto v \pmod{t}$ is a homomorphism $H^{t,k} \rightarrow H$. We have that the $g^t \times t$ array with rows all $t$-tuples from $\mathbb{Z}_g^t$ is a VCA$(g^t; H, g)$, so:

$g^t \leq CCAN(t, k, g) = \text{VCAN}(H^{t,k}, g) \leq \text{VCAN}(H, g) = g^t$. \qed
Theorem 3.4.18. Let $k > t > 0$ and $g > 1$ be integers. If $t|k$, then $c\text{CCAN}(t, k, g) = g^t$, and otherwise, $c\text{CCAN}(t, k, g) \leq c\text{CCAN}(t, t + (k \mod t), g)$.

Proof. If $t|k$, define the complete $t$-hypergraph $H_t = (\mathbb{Z}_t, \{\mathbb{Z}_t\})$, and define the map $f : V(H_{t}^{c, k}) \rightarrow V(H_t)$ by taking $u \in \{0, \ldots, k - 1\}$ to $u \mod t$. Then this is easily seen to be a homomorphism. Furthermore, we have that CAN$(t, t, g) = V\text{CAN}(H_t, g) = g^t$, as we can simply list all of the $g^t$ tuples as rows of a $g^t \times t$ array, say $M$, which is the required covering array. By taking the $g^t \times k$ array $M'$ assigning to column $v$ the contents of column $f(v)$ of $M$, we then have that $M'$ is a $c\text{CCA}(g^t; t, k, g)$.

Otherwise, the second result follows immediately from Corollary 2.2.37.

We now determine the strong chromatic number for all $H_{c, t}^{k, t}$, which, if we consider $H_{c, t}^{k, t}$ as an ASC, is equal to the weak chromatic number.

Lemma 3.4.19. For $k > t > 0$, write $k = qt + r$ for $0 \leq r < t$. Then $H_{c, t}^{k, t}$ is strongly $t + \lceil \frac{r}{q} \rceil$-colourable.
Proof. Let \( d = t + \left\lceil \frac{r}{q} \right\rceil \), and let \( V = V(H_{c,t}^{k}) = \{0, \ldots, k - 1\} \). We give a function \( V \rightarrow \mathbb{Z}_d \), where vertices of cyclic distance less than \( t \) do not map to the same value, which is then necessarily a strong \( d \)-colouring of \( H_{c,t}^{k} \). We divide into the following cases:

1. If \( q = 1 \), then \( d = t + \left\lceil \frac{r}{q} \right\rceil = t + r = k \), and certainly \( H_{c,t}^{k} \) is strongly \( k \)-colourable by taking \( v \mapsto v \).

2. If \( r = 0 \), i.e. \( t \mid k \), then \( k = qt \) and \( d = t + \left\lceil \frac{r}{q} \right\rceil = t \), and taking vertex \( i \mapsto i \mod t \) is a strong \( t \)-colouring of \( H_{c,t}^{k} \).

3. Otherwise, we want to partition the \( k \) vertices into \( q \) uniform or almost uniform classes, which we will then colour as in Figure 3.8. If \( q \mid r \), then \( r = qs \) for some positive integer \( s \), so \( k = q(t + s) = q(t + \left\lceil \frac{r}{q} \right\rceil) \).

If \( q \not\mid r \), then let \( f = \left\lfloor \frac{r}{q} \right\rfloor \), and \( f + 1 = \left\lceil \frac{r}{q} \right\rceil \). We can place \( t \) vertices in each of the \( q \) classes, but then must partition \( r \) remaining vertices into \( x \) classes of size \( f \), and \( q - x \) classes of size \( f + 1 \), giving:

\[
x \left\lfloor \frac{r}{q} \right\rfloor + (q - x) \left\lceil \frac{r}{q} \right\rceil = r
\]
\[
xf + (q - x)(f + 1) = r
\]
\[
x = q(f + 1) - r
\]

Thus, we have \( q(f + 1) - r \) classes of size \( f \), and \( q - (q(f + 1) - r) = r - qf \) classes of size \( f + 1 \). Formally, to get the colouring of Figure 3.8 we derive the following colouring on the vertices of \( H_{c,t}^{k} \):

\[
\{0, \ldots, k - 1\} \rightarrow \mathbb{Z}_d
\]

\[
v \mapsto \begin{cases} 
  v \mod f & \text{if } 0 \leq v < (q(f + 1) - r)f. \\
  (v + rf) \mod (f + 1) & \text{if } v \geq (q(f + 1) - r)f.
\end{cases}
\]

Then, in this colouring, clearly vertices of cyclic distance less than \( t \) do not have the same colour, so this is a strong \( d \)-colouring.

\( \square \)

We will use the independence number of the hypergraph along with a result on the size of colour classes to establish the chromatic number.
Definition 3.4.20. If $H = (V, E)$ is a hypergraph, a set $X \subseteq V(H)$ is said to be independent if, for all $u, v \in X$, there is no edge $e \in E$ with $u, v \in e$. The independence number of $H$, written $\alpha(H)$, is defined as:

$$\alpha(H) = \max\{|X| : X \text{ is an independent set of } H\}.$$  

Lemma 3.4.21. If $k = qt + r$ for $0 \leq r < t$, then $\alpha(H^{k,t}_c) = q$.

Proof. Two vertices $u$ and $v$ of $H^{k,t}_c$ are not in the same edge if they are at cyclic distance at least $t$. Thus, $S = \{0, t, 2t, \ldots, (q - 1)t\}$ is an independent set of $H^{k,t}_c$. If there was a set of $q + 1$ vectors whose pairwise cyclic distance was at least $t$, this would require $k$ to be at least $(q + 1)t$, which contradicts the value of $k$, so $S$ is of maximum size. \hfill \Box

There is a connection between independent sets and colourings of a hypergraph.

Lemma 3.4.22. If $H$ is a hypergraph and $f$ is a strong $c$-colouring of $H$, then for all $0 \leq i < c$, the $i$th colour class $P_i = \{v \in V(H) : f(v) = i\}$ is an independent set in $H$. Additionally, $\gamma(H)\alpha(H) \geq |V(H)|$.

Proof. By property of strong colourings, two vertices with the same colour - that is, two vertices in the same colour class - cannot appear in an edge together, thus showing that each colour class $P_i$ is an independent set. Hence, we have that $|P_i| \leq \alpha(H)$. Since the colour classes partition $V(H)$, we have:

$$\sum_{i=0}^{c-1} |P_i| = |V(H)|$$

$$\sum_{i=0}^{c-1} \alpha(H) \geq |V(H)|$$

$$c\alpha(H) \geq |V(H)|$$

As this holds for any strong $c$-colouring, in particular it holds for $c = \gamma(H)$, so $\gamma(H)\alpha(H) \geq |V(H)|$. \hfill \Box

We are now ready to establish the strong chromatic number of $H^{k,t}_c$.

Theorem 3.4.23. Let $k = qt + r$ for $0 \leq r < t$. Then $\gamma(H^{k,t}_c) = t + \ceil*{\frac{r}{q}}$.

Proof. From Lemma 3.4.21 and Lemma 3.4.22 we have that $\gamma(H^{k,t}_c)q \geq k$. Thus, we have $\gamma(H^{k,t}_c) \geq \frac{k}{q} = \frac{qt + r}{q} = t + \frac{r}{q}$, and since $\gamma(H^{k,t}_c)$ must be an integer, $\gamma(H^{k,t}_c) \geq t + \ceil*{\frac{r}{q}}$.

In Lemma 3.4.19 we found a $(t + \ceil*{\frac{r}{q}})$-colouring of $H^{k,t}_c$, thus establishing equality. \hfill \Box
Establishing $\gamma(H_c^{k,t})$ allows us to identify some cases where we can find provably optimal covering arrays over $H_c^{k,t}$. In particular, we will use the Zero-Sum construction for orthogonal arrays.

**Theorem 3.4.24.** Let $k > t > 0$, $g > 1$, and write $k = tq + r$ for $0 \leq r < t$. Then if $0 \leq r \leq q$, $c\text{CCAN}(t,k,g) = g^t$.

**Proof.** If $r = 0$, then $t | k$, so by Theorem 3.4.18, we have the result.

Otherwise, we have that $\left\lfloor \frac{r}{q} \right\rfloor = 1$, so from Theorem 3.4.23 $\gamma(H_c^{k,t}) = t + 1$, and from Theorem 2.1.35 $H_c^{k,t} \rightarrow K_{t+1}^t$, so $c\text{CCA}(t,k,g) \leq \text{VCAN}(K_{t+1}^t,g) = \text{CAN}(t,t+1,g)$. Since, by the Zero-Sum construction as given in Lemma 2.2.21 an OA$(t,t+1,g)$ exists, CAN$(t,t+1,g) = g^t$. \hfill $\Box$

**Corollary 3.4.25.** Let $k > t > 0$, $g > 1$, and write $k = tq + r$ for $0 \leq r < t$. If $q \geq (t-1)$, then $c\text{CCAN}(t,k,g) = g^t$.

**Proof.** We have $r \leq t - 1 \leq q$, so by Theorem 3.4.24, the result follows immediately. \hfill $\Box$

**Theorem 3.4.26.** Let $k > t > 0$ and write $k = qt + r$ for $0 \leq r < t$. Let $g = p^m$ be a prime power with $g + 1 \geq t + \left\lceil \frac{r}{q} \right\rceil$. Then $c\text{CCAN}(t,k,g) = g^t$.

**Proof.** Let $d = t + \left\lceil \frac{r}{q} \right\rceil$. From Theorem 3.4.23 $\gamma(H_c^{k,t}) = d$, and from Theorem 2.1.35 $H_c^{k,t} \rightarrow K_d^t$, so $c\text{CCA}(t,k,g) \leq \text{VCAN}(K_d^t,g) = \text{CAN}(t,d,g)$. By the Bush construction of Lemma 2.2.22 an OA$(t,g+1,g)$ exists, and since $g + 1 \geq d$, by dropping any $(g+1) - d$ columns of the orthogonal array, we derive a CA$(g^t;t,d,g)$. \hfill $\Box$

We now enumerate the remaining cases to be solved to determine the full spectrum of $c\text{CCAN}(t,k,g)$.

**Open Question 3.4.1.** Let $k > t > 0$, $g > 1$, and write $k = qt + r$ for $0 \leq r < t$. Find $c\text{CCAN}(t,k,g)$ for the cases:

1. $t + 2 \leq k \leq 2t - 1$.

2. $q + 1 \leq r < t$, $2t + 3 \leq k \leq t(t-1) - 1$, and one of the following:
   
   (a) $g$ not a prime power, or
   
   (b) $g < t + \left\lceil \frac{r}{q} \right\rceil - 1$. 
Furthermore, by simply executing a backtracking search, we have found $c\text{CCA}(g^t; t, k, g)$ in the following cases:

1. $t = 3$, $k = 5$, $g \in \{2, 3\}$.
2. $t = 4$, $k = 6$, $g \in \{2, 3\}$.
3. $t = 4$, $k = 7$, $g \in \{2, 3\}$.
4. $t = 5$, $k = 7$, $g \in \{2, 3\}$.
5. $t = 5$, $k = 8$, $g \in \{2, 3\}$.
6. $t = 5$, $k = 9$, $g = 2$.
7. $t = 6$, $k = 8$, $g = 2$.
8. $t = 6$, $k = 9$, $g = 2$.

### 3.4.4 Constructions for VCA over 2-Trees

As discussed in Section [2.2.4](#), for a bipartite graph $G = (V, E)$ with $s$ vertices, it was determined independently in both [17](#) and [67](#) that:

$$\text{CAN}(G, (g_0, \ldots, g_{s-1})) = \max \{g_i g_j : \{i, j\} \in E\}.$$ 

Determining an analogous result for strongly 3-colourable hypergraphs is more difficult and may not even hold in all cases; however, there may be certain families of strongly 3-colourable hypergraphs where such a result holds. In Section [2.2.8](#) we introduced $k$-tree hypergraphs and showed that their strong chromatic number is $k + 1$.

We present the following conjecture:

**Conjecture 3.4.27.** Let $x, y, z$ be positive integers, and let $N \geq xyz$ be an integer. If we have a $N \times 2$ array $M = [c_x, c_y]$ with $c_x \in \mathbb{Z}_x^N$ and $c_y \in \mathbb{Z}_y^N$, such that the following properties hold:

1. The column $c_x$ is uniform if $x|N$, or almost-uniform otherwise;
2. The column $c_y$ is uniform if $y|N$, or almost-uniform otherwise; and
3. $M$ is 2-uniform if $(xy)|N$, or 2-almost-uniform otherwise.
Then there exists a $c_z \in \mathbb{Z}_N^3$ such that:

1. The column $c_z$ is uniform if $z|N$, or almost-uniform otherwise;
2. $[c_x, c_z]$ is 2-uniform if $(xz)|N$, or 2-almost-uniform otherwise;
3. $[c_y, c_z]$ is 2-uniform if $(yz)|N$, or 2-almost-uniform otherwise; and
4. $[c_x, c_y, c_z]$ is covered.

The following conjecture is a direct corollary of Conjecture 3.4.27:

**Conjecture 3.4.28.** Let $H$ be a 2-tree hypergraph with $s$ vertices, and let $g_0, \ldots, g_{s-1} \geq 1$. Write:

$$N = \max \{ g_x g_y g_z : \{x, y, z\} \in E(H) \}.$$ 

Then there exists a VCA($N; H, (g_0, \ldots, g_{s-1})$).

**Proof.** We proceed with structural induction on the 2-tree hypergraph construction. Select any edge $e = \{x, y, z\} \in E(H)$ such that $g_x g_y g_z = N$ and construct the $N \times 3$ array with rows consisting of all the tuples of $\mathbb{Z}_{g_x} \times \mathbb{Z}_{g_y} \times \mathbb{Z}_{g_z}$. Then clearly any 2-subset of columns of this array is 2-uniform, each of the columns is uniform, and the array is covered.

Following the iterative 2-tree construction for $H$, for each new vertex $u$ and choice of existing hyperedge $e' = \{v, w, x\}$ such that we will add $u$ and $\{u, v, w\}$ to $H$, we have that the $N \times 2$ subarray $[c_v, c_w]$ is 2-uniform or 2-almost uniform and the columns of $c_v$ and $c_w$ are uniform or almost-uniform by structural induction. Thus, the premises of Conjecture 3.4.27 hold, so we can find a column $c_u \in \mathbb{Z}_{g_u}$ that is uniform, $[c_u, c_v]$ is 2-uniform or 2-almost-uniform, $[c_u, c_w]$ is 2-uniform or 2-almost-uniform, and $[c_u, c_v, c_w]$ is covered.

Thus, by structural induction, the full $N \times s$ array created in this way is covered and is therefore a VCA($N; H, (g_0, \ldots, g_{s-1})$).

In order to test the conjecture, we designed an algorithm that, given a 2-tree hypergraph, attempted to construct the array as in the proof of Conjecture 3.4.28 by identifying a facet where the product of the number of levels associated with the vertices was maximum, say $N$, and then building an $N \times 3$ array consisting of all possible tuples. For each new vertex $i$, say with $g_i$ levels, we ran a 0-1 integer linear program over $Ng_i$ variables using the GLPKK solver to attempt to find a column of length $N$ satisfying the
conclusion of Conjecture 3.4.27. If such a column was found, we added it to the array. If no column could be found, we terminated unsuccessfully.

We ran our algorithm on 2000 randomly generated 2-tree hypergraphs, designed in the following way:

1. Begin with the 2-tree hypergraph \( H = (\{x, y, z\}, \{x, y, z\}) \), randomly selecting between 2 and 20 levels for each factor. Take \( N \ (8 \leq N \leq 8000) \) to be the product of the number of levels for the three factors.

2. Add 47 more vertices to the 2-tree hypergraph, one at a time: for each \( u \), randomly select an existing facet \( e \) and two vertices \( \{v, w\} \subset e \), and then add facet \( \{u, v, w\} \) to the hypergraph. Let \( g_v \) and \( g_w \) be the number of levels for \( v \) and \( w \) respectively, and select a random number of levels between 2 and \( \lfloor N/(g_v g_w) \rfloor \) for \( u \).

In all cases, our algorithm was able to produce a variable strength covering array over the 2-tree hypergraph. However, we have not yet found a deterministic, algebraic algorithm that satisfies Conjecture 3.4.27.

**Open Question 3.4.2.** Let \( H \) be a 2-tree hypergraph over \( s \) vertices, and \( g_0, \ldots, g_{s-1} \geq 1 \). Find an algorithm that serves as a proof of Conjecture 3.4.27. Then this gives that:

\[
\text{VCAN}(H, (g_0, \ldots, g_{s-1})) = \max \{g_x g_y g_z : \{x, y, z\} \in E(H)\}.
\]

### 3.5 Constructions for VCA over Triangulations of the Sphere

We introduced triangulation hypergraphs of the sphere in Section 2.2.9. These comprise a family of 4-colourable hypergraphs, and thus by using orthogonal arrays, we can fully determine the variable-strength covering array number for all such hypergraphs.

The four-colour theorem can be used to determine the chromatic number of planar graphs; it also applies to graphs embedded on the surface of spheres and cylinders.

**Theorem 3.5.1** (Four-Colour Theorem, see [6]). Every planar graph, spherical graph, and cylindrical graph is four-colourable.

We then have the following corollary:

**Corollary 3.5.2.** If \( \Delta \) is a triangulation hypergraph of the sphere, then for any \( g \geq 1 \), \( \text{VCAN}(\Delta, g) = g^3 \).
Proof. Let $\Delta$ be a triangulation hypergraph of the sphere with facet subhypergraph $\Lambda$. The underlying graph of $\Delta$, namely $[\Delta]_2$, ignoring loops, is a spherical graph, and thus 4-colourable. Thus, as $\Delta$ is an ASC, a colouring of $[\Delta]_2$ is a strong colouring of $\Delta$, so $\Delta$ is strongly 4-colourable, and so is $\Lambda$. We have that $\Lambda$ is 3-uniform and strongly 4-colourable so by Theorem 2.1.35, $\Lambda \rightarrow K^3_4$, so $\Delta \rightarrow \Delta(K^3_4)$. Thus, $\text{VCAN}(\Delta, g) \leq \text{VCAN}(K^3_4, g)$.

By the Zero-Sum Construction as given in Lemma 2.2.21, we have that an OA$(3, 4, g)$ exists, which is also a VCA$(g^3; K^3_4, g)$. As $\text{VCAN}(\Delta, g) \geq g^3$, then $\text{VCAN}(\Delta, g) = g^3$. □
Chapter 4

Covering Arrays from Linear Feedback Shift Register Sequences

In this chapter, we use linear feedback shift registers to construct VOA and covering arrays. The construction here can be viewed as a generalization of the construction in Theorem 2.2.20. Munemasa [72] employs linear feedback shift registers on primitive trinomials over $\mathbb{F}_2$ to create strength-2 orthogonal arrays, which have the additional property of being very close to having strength-3. Based on a similar idea, given a linear feedback shift register for any primitive polynomial over a finite field $\mathbb{F}_q$, in Theorem 4.2.1, we derive a construction for a variable strength orthogonal array $\text{VOA}(q^3; \Lambda, q)$ with $q^2 + q + 1$ factors, where we can precisely describe the structure of $\Lambda$: we show in Theorem 4.2.2 that $\Lambda$ is a $(q^2 + q + 1, 3, q^2)$-BIBD. In Theorem 4.2.10, by permuting the columns of the array and then vertically concatenating the permuted array with the original, we derive a new construction for strength-3 covering arrays of the form $\text{CA}(2q^3-1; 3, q^2 + q + 1, q)$.

Colbourn [25] maintains a repository of best known upper bounds for covering arrays for levels $2 \leq q \leq 25$, and our construction substantially improves the bounds for all prime powers $4 \leq q \leq 25$, as shown in Table 4.1. We extend our results to non-prime powers and different numbers of factors through the use of fusion (Theorem 2.2.15), factor increases (Theorem 2.2.16), and recursive constructions discussed in Section 2.2.3; the improvements on known bounds are summarized in Table 4.8.

In Section 4.1, we introduce the finite field background that underlies our construction. Section 4.2 details the VOA construction and identifies the structure of the ASC covered by the VOA, and the technique that we use to extend the VOA into a strength-3 CA. In Section 4.3, we detail the new covering array upper bounds obtained, and provide
Covering Arrays from Linear Feedback Shift Registers

details on the modifications and recursive constructions we used to extend our results to produce additional new bounds. We then, in Section 4.4, examine the possibility of generalizing our technique to higher strength arrays. We establish the equivalence between linear dependence of sets of vectors and uncovered sets of columns in Theorem 4.4.1 and detail experimental results and problems that arise from our initial attempts to use our technique to generate full strength-\( m \) covering arrays for \( m > 3 \).

These results have been prepared as a paper and submitted in [78].

4.1 Linear Feedback Shift Registers

We provide an introduction to linear feedback shift registers and investigate some of their properties, which we will use in Section 4.2 in order to construct both VOA and strength-3 CA.

Let \( G \) be a group written multiplicatively. For an element \( g \in G \), we write \( \langle g \rangle = \{g^i : i \in \mathbb{Z}\} \) to denote the subgroup of \( G \) generated by the element \( g \). If \( \langle g \rangle = G \), we say that \( g \) generates \( G \). Furthermore, the order of an element \( g \in G \) is defined as \( \text{ord}(g) = |\langle g \rangle| = \min\{i > 0 : g^i = 1\} \). Note that if there is no such \( i \), we say that \( g \) has infinite order. We also have, for \( i > 0 \), that \( \text{ord}(g^i) = \frac{|\langle g \rangle|}{\gcd(i, |\langle g \rangle|)} \).

Definition 4.1.1. Let \( q = p^m \), where \( p \) is a prime power and \( m \geq 1 \). Consider \( \mathbb{F}_q \), the finite field of \( q \) elements. An element \( \alpha \in \mathbb{F}_q \) is called a primitive element of \( \mathbb{F}_q \) if \( \alpha \) generates \( \mathbb{F}_q^\times \), the multiplicative group of \( \mathbb{F}_q \), or, equivalently, \( \text{ord}(\alpha) = q - 1 \). A polynomial \( f \in \mathbb{F}_p[x] \) of degree \( m \geq 1 \) is called monic if the coefficient of \( x^m \) in \( f \) is 1. We call \( f \) irreducible if for any factorization \( f = gh \) with \( g, h \in \mathbb{F}_p[x] \), we have that \( \min(\deg g, \deg h) = 0 \). We say that \( f \) is the minimal polynomial over \( \mathbb{F}_p \) of \( \alpha \in \mathbb{F}_q \) if \( f \) is monic and \( f \) is the polynomial of smallest degree such that \( f(\alpha) = 0 \). We call \( f \) a primitive polynomial over \( \mathbb{F}_p \) if \( f \) is the minimal polynomial over \( \mathbb{F}_p \) of a primitive element of \( \mathbb{F}_{p^m} = \mathbb{F}_q \). Equivalently, \( f \in \mathbb{F}_p[x] \) of degree \( m \geq 1 \) is primitive if and only if \( f \) is monic, \( f(0) \neq 0 \), and \( f \) is irreducible in \( \mathbb{F}_p[x] \) such that it has a root \( \alpha \in \mathbb{F}_p^m \) that is primitive.

If \( \alpha \in \mathbb{F}_q \) is primitive, i.e. \( \langle \alpha \rangle = \mathbb{F}_q^\times \), we have that, for \( s = \frac{q-1}{p-1} \), \( \langle \alpha^s \rangle = \mathbb{F}_p^\times \).

Definition 4.1.2. Let \( f(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_{m-1} x^{m-1} + x^m \in \mathbb{F}_q[x] \) be a polynomial of degree \( m \). A linear feedback shift register (LFSR) with characteristic polynomial \( f \) and initial values \( T = (b_0, \ldots, b_{m-1}) \in \mathbb{F}_q^m \) is a sequence \( S(f, T) = (a_0, a_1, a_2, \ldots) \) over
\( F_q \) defined as

\[
a_i = \begin{cases} 
  b_i, & \text{if } 0 \leq i < m \\
  -c_{m-1}a_{i-1} - c_{m-2}a_{i-2} - \ldots - c_1a_{i-(m-1)} - c_0a_{i-m}, & \text{if } i \geq m. 
\end{cases}
\]

**Definition 4.1.3.** A sequence \((a_0, a_1, \ldots)\) is **periodic** if there exists a positive integer \(n\) such that \(a_{n+i} = a_i\) for all \(i \geq 0\). The smallest such \(n\), if one exists, is called the **least period**, or simply the period of the sequence.

Given a polynomial \(f \in F_q[x]\) of degree \(m\), an LFSR sequence over \(f\) can have least period at most \(q^m - 1\) because the LFSR sequence over \(f\) is a recurrence relation where the value of a term \(a_i\) is defined completely by the previous \(m\) terms. There are \(q^m\) possible values for these previous \(m\) terms, and if they are all 0, then every subsequent term of the sequence must be 0, making the least period 1. Thus, excluding this possibility gives \(q^m - 1\) possible values for the previous \(m\) terms, thus giving us a maximum possible least period of \(q^m - 1\) if all those possible values are realized in the sequence. The following proposition shows that an LFSR sequence can indeed attain this maximum possible least period.

**Proposition 4.1.4** (See [59, Theorem 8.33]). If \(f\) is a primitive polynomial over \(F_q\) with \(\deg f = m\) and initial values \(T = (b_0, \ldots, b_{m-1}) \neq (0, \ldots, 0)\), then the LFSR generated by \(f\) and \(T\) has the least period of maximum possible length \(q^m - 1\).

We define the trace function from the extension field \(F_{q^m}\) to the base subfield \(F_q\), and then present a structural theorem of linear feedback shift registers that gives a bijection between the initial values \(F_{q^m} \setminus \{0\}\) of the LFSR and \(F_q^\times\), which will be useful in future proofs.

**Definition 4.1.5.** Let \(\text{Tr}\) denote the trace function from \(F_{q^m}\) to \(F_q\), defined by:

\[
\text{Tr} : F_{q^m} \rightarrow F_q \\
  a \mapsto a^{q^0} + a^{q^1} + a^{q^2} + \ldots + a^{q^{m-1}}
\]

Note that the trace is \(F_q\)-linear, i.e. for all \(s, t \in F_q, a, b \in F_{q^m}\), \(\text{Tr}(sa + tb) = s\text{Tr}(a) + t\text{Tr}(b)\), as shown in Theorem 2.23 of [59].

**Proposition 4.1.6** (See [59, Theorem 8.21]). Let \(f\) be a primitive polynomial over \(F_q\) of degree \(m\) with root \(\alpha\). For any initial values \(T = (b_0, \ldots, b_{m-1})\), there exists a unique element \(\beta \in F_{q^m}\) such that \(b_i = \text{Tr}(\beta \alpha^i)\) for all \(0 \leq i \leq m - 1\), and the LFSR \(S(f, T) = (a_i)\) has the property that for all \(i \geq 0\), \(a_i = \text{Tr}(\beta \alpha^i)\).
We now determine several useful results about the distribution of symbols in an LFSR. Consider a primitive polynomial \( f \) over \( \mathbb{F}_q \) of degree \( m \) and initial values \( T \neq (0, \ldots, 0) \in \mathbb{F}_q^m \). By Proposition 4.1.4, we know that the LFSR \( S(f, T) \) has period \( q^m - 1 \). So, every vector in \( \mathbb{F}_q^m \) with the exception of the all-zero vector appears as a substring exactly once in each period of the sequence. Then the number of appearances of symbol \( x \) in one period is equal to the number of nonzero vectors of length \( m \) with \( x \) as the first coordinate. Thus, in one period of the LFSR, each nonzero element \( x \in \mathbb{F}_q^\times \) occurs \( q^m - 1 \) times, and 0 occurs \( q^m - 1 \) times.

Let \( a = (a_i) \) be a sequence over \( \mathbb{F}_q \) of period \( q^m - 1 \), and let \( k = \frac{q^m - 1}{q - 1} \). For \( v, w \in \mathbb{F}_q \) and \( 0 \leq d < q^m - 1 \), define:

\[
N_{v,w}(d) = |\{i : (a_i, a_{i+d}) = (v, w), 0 \leq i < q^m - 1\}|.
\]

Then \( a \) has the 2-tuple balance property if \( a \) satisfies the following conditions:

1. If \( d \not\equiv 0 \pmod{k} \), then for all \( (v, w) \neq (0, 0) \), \( N_{v,w}(d) = q^m - 2 \), and \( N_{0,0}(d) = q^m - 1 \).

2. If \( d \equiv 0 \pmod{k} \), then there exists some \( \lambda \in \mathbb{F}_q \) (\( \lambda = 1 \) if and only if \( d = 0 \)) such that \( (a_i, a_{i+d}) = (a_i, \lambda a_i) \) for all \( 0 \leq i < q^m - 1 \), and additionally, for \( v \neq 0 \), \( N_{v,\lambda v}(d) = q^m - 1 \), and \( N_{0,0}(d) = q^m - 1 \). For \( w \neq \lambda v \), \( N_{v,w}(d) = 0 \).

The following theorem originally appeared in [94], and a very nice proof can be found in [47].

**Theorem 4.1.7 ([94] Theorem 12).** If \( f \) is a degree-\( m \) primitive polynomial over \( \mathbb{F}_q \), then the LFSR sequence defined by \( f \) for any initial values \( T \neq (0, \ldots, 0) \) has the 2-tuple balance property.

**Corollary 4.1.8.** If \( f \) is a degree-\( m \) primitive polynomial over \( \mathbb{F}_q \) and \( T \in \mathbb{F}_q^m \), \( T \neq (0, \ldots, 0) \), then the LFSR \( S(f, T) = (a_i) \) has the following properties:

1. Every length \( \frac{q^m - 1}{q - 1} \) subinterval contains exactly \( \frac{q^m - 1}{q - 1} \) zeros.

2. For all \( i \geq 0, j \geq 0 \), the positions of zeros are identical over any two length \( \frac{q^m - 1}{q - 1} \) subintervals starting in positions \( i \) and \( i + j\frac{q^m - 1}{q - 1} \).

**Proof.** By Theorem 4.1.7, \( S(f, T) \) has the 2-tuple balance property, so if \( a_i = 0 \), then \( a_{i+j\frac{q^m - 1}{q - 1}} = 0 \) for all \( j \geq 0 \). This gives that the positions of zeros are identical over any
two subintervals of length \( \frac{q^{m-1}}{q-1} \) starting in positions that differ by a multiple of \( \frac{q^{m-1}}{q-1} \). Furthermore, as every period of the LFSR contains \( q^{m-1} - 1 \) zeros, and the period length is \( q^m - 1 \), there are \( q - 1 \) subintervals of length \( \frac{q^{m-1}}{q-1} \) in each period. Thus, each such subinterval contains exactly \( \frac{q^{m-1} - 1}{q-1} \) zeros.

\[ \square \]

4.2 VOA and Strength-3 CA Constructions

We now give the structure of the array that, in Theorem 4.2.1, we show to be a variable strength orthogonal array, and determine precisely where coverage occurs. We then show that by adding the reversal of each row of the array with the exception of an all-zero row, we achieve coverage of all 3-sets of factors, thus leading to a construction for \( \text{CA}(2(q^3 - 1) + 1; 3, q^2 + q + 1, q) \) covering arrays.

If \( S = (a_0, a_1, \ldots) \) is a sequence, for a positive integer \( n \), consider:

\[ C_i^n(S) = (a_i, a_{i+1}, \ldots, a_{i+n-1}). \]

We say that \( C_i^n(S) \) is the subinterval of \( S \) of length \( n \) beginning in position \( i \).

For a given length, we now build an array of all subintervals of that length from an LFSR, along with an additional all-zero row. This will serve as the basis for the VOA construction.

Let \( f \) be a degree-\( m \) primitive polynomial over \( \mathbb{F}_q \) with root \( \alpha \in \mathbb{F}_{q^m} \). Then we have that \( \{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\} \) is a basis for \( \mathbb{F}_{q^m}^n \) (a proof is given in [59]). For some initial values \( T = (b_0, \ldots, b_{m-1}) \) not all zero, take the LFSR \( S(f, T) \), which has period \( q^m - 1 \). Let \( k = \frac{q^m - 1}{q-1} \). Consider the following \( q^m \times k \) array:

\[ M = M(f, T) = \begin{bmatrix} C_0^k(S(f, T)) \\ C_1^k(S(f, T)) \\ \vdots \\ C_{q^{m-2}}^k(S(f, T)) \\ 0, 0, \ldots, 0 \end{bmatrix}. \]

Because the sequence is defined by a primitive polynomial of degree \( m \), the rows of the array form a subspace of \( \mathbb{F}_q^k \) of dimension \( m \). Furthermore, the subspace is independent of the choice of \( T \) since the sequence \( S(f, T') \) for any nonzero \( T' \) contains all nonzero vectors of length \( m \) as a subinterval. Thus, we can omit \( T \) from the definition of \( M \) and simply define \( M \) by its characteristic polynomial \( f \) and write \( M = M(f) \). We call \( M \)
Theorem 4.2.1. Let \( f \) be a primitive polynomial of degree \( 3 \) over \( \mathbb{F}_q \) with root \( \alpha \in \mathbb{F}_{q^3} \), and let \( k = \frac{q^3-1}{q-1} = q^2 + q + 1 \). Consider the \( q \times k \) array \( M = M(f) \), the subinterval array of \( f \), and \( \Delta = \Delta(M) \) with facet set \( \Lambda \). Then \( M \) is a VOA\((q^3, \Lambda, q)\), \( \left( \binom{0,\ldots,k-1}{2} \right) \subset \Delta \), and for a set \( \{i_0, i_1, i_2\} \), \( 0 \leq i_0 < i_1 < i_2 < q^2 + q + 1 \), the following are equivalent:

1. \( \{i_0, i_1, i_2\} \in \Lambda \) (i.e. \( \{i_0, i_1, i_2\} \) is balance-covered in \( M \)).

2. There is no row \( r \) in \( M \), \( 0 \leq r < q^3 \), other than the all-zero row such that \( M_{r,i_0} = M_{r,i_1} = M_{r,i_2} = 0 \).

3. \( \{\alpha^{i_0}, \alpha^{i_1}, \alpha^{i_2}\} \) is linearly independent over \( \mathbb{F}_q \).

Proof. By Theorem [4.1.7], the LFSR generated by \( f \) for any nonzero initial values has the 2-tuple balance property. Hence, for any pair of columns \( 0 \leq i_0 < i_1 < q^2 + q + 1 \) and any two values \( v, w \in \mathbb{F}_q \), we have that, in the \( q^3 \times 2 \) subarray over columns \( i_0, i_1 \), \( (v, w) \) appears as a row precisely \( q^3/q^2 = q \) times, so \( \{i_0, i_1\} \) is balance-covered in \( M \).

We now consider which 3-sets (triples) of columns are balance-covered, or equivalently, as \( q^3/q^3 = 1 \), covered in \( M \). Let \( \mathcal{U} \) be the set of all triples of columns:

\[
\mathcal{U} = \{\{i_0, i_1, i_2\} : 0 \leq i_0 < i_1 < i_2 < q^2 + q + 1\}.
\]

Let \( \mathcal{T} \) be the set of triples of columns that are covered in \( M(f) \). Let \( \mathcal{N} \) be the set of triples of columns that are not covered. Additionally, let \( \mathcal{I} \) be the set of triples of columns \( \{i_0, i_1, i_2\}, 0 \leq i_0 < i_1 < i_2 < q^2 + q + 1 \), such that \( \{\alpha^{i_0}, \alpha^{i_1}, \alpha^{i_2}\} \) is linearly independent, and let \( \mathcal{D} \) be the set of triples of columns \( \{i_0, i_1, i_2\}, 0 \leq i_0 < i_1 < i_2 < q^2 + q + 1 \), such that \( \{\alpha^{i_0}, \alpha^{i_1}, \alpha^{i_2}\} \) is linearly dependent. We then have that \( \mathcal{U} = \mathcal{T} \cup \mathcal{N} = \mathcal{I} \cup \mathcal{D} \).

Let \( \mathcal{Z} \) be the set of triples of columns such that, for \( \{i_0, i_1, i_2\} \in \mathcal{Z}, M \) has a row \( r \) other than the all-zero row, \( 0 \leq r < q^3 - 1 \), such that \( M_{r,i_0} = M_{r,i_1} = M_{r,i_2} = 0 \). From Corollary [4.1.8], each non-zero row of \( M \), which is a length-\( (q^2 + q + 1) \) subinterval of the LFSR over \( f \), contains exactly \( \frac{q^2-1}{q-1} = q + 1 \) zeros, and the positions of zeros in any row of \( M \) are identical to a row in the first \( q^2 + q + 1 \) rows, so with regards to \( \mathcal{Z} \), without loss of generality we can assume \( 0 \leq r < q^2 + q + 1 \). Hence, we have that \( |\mathcal{Z}| \leq \left( \frac{q+1}{3} \right) (q^2+q+1) \).

We claim there exists exactly one row \( 0 \leq r < q^2 + q + 1 \) such that \( M_{r,i_0} = M_{r,i_1} = M_{r,i_2} = 0 \). Let \( r \) be one such row. Since \( \{i_0, i_1\} \) is balance-covered in \( M \), \( M_{r',i_0} = M_{r',i_1} = M_{r',i_2} = 0 \).
0 for exactly \( q - 1 \) values \( 0 \leq r' < q^3 - 1 \), i.e. \( r' = r + j(q^2 + q + 1) \) for \( j = 0, \ldots, q - 2 \), and only \( r \) is in the first \( q^2 + q + 1 \) rows of \( M \). Thus, we have:

\[
|Z| = \binom{q+1}{3}(q^2 + q + 1).
\]

Counting the number of ways of choosing three linearly independent vectors from \( \{\alpha^i : 0 \leq i < q^2 + q + 1\} \) gives:

\[
|I| = \frac{(q^3 - 1)(q^3 - q)(q^3 - q^2)}{3!(q - 1)^3}.
\]

This gives that:

\[
|D| = |U| - |I| = \binom{q+1}{3}(q^2 + q + 1).
\]

Let \( N \in \mathcal{N} \), i.e. \( N \) is a triple of columns not covered in \( M \). Since the LFSR is defined by a degree-3 linear relation, we can construct a linear map \( g : \mathbb{F}_{q^3}^3 \to \mathbb{F}_{q^3}^3 \) from columns \( \{0, 1, 2\} \) to the columns indexed by \( N \) as a row is fully determined by the initial values in columns \( \{0, 1, 2\} \). As \( f \) has degree 3, we have that \( \{0, 1, 2\} \in \mathcal{T} \), i.e. is covered in \( M \). Since \( N \) is not covered, this map cannot be surjective, so \( \dim(\ker g) \geq 1 \), and therefore \( |g^{-1}(0, 0, 0)| > 1 \). Hence, \( N \in Z \). Now consider \( Z \in Z \). Since there are multiple rows in \( M \) with values 0, 0, 0 in the subarray indexed by the columns in \( Z \), by counting, \( Z \) must be uncovered. Thus, \( Z \in \mathcal{N} \). Hence, \( \mathcal{N} = \mathcal{Z} \), showing that 1 and 2 are equivalent.

Let \( D = \{i_0, i_1, i_2\} \in \mathcal{D} \). Thus, there exist \( s_0, s_1 \in \mathbb{F}_q \) such that \( \alpha^{i_2} = s_0\alpha^{i_0} + s_1\alpha^{i_1} \). Consider any row \( r, 0 \leq r < q^3 - 1 \) in \( M \) with the property that \( M_{r,i_0} = M_{r,i_1} = 0 \), and let \( r \) be defined by \( \beta \in \mathbb{F}_{q^n} \) such that \( M_{r,j} = \text{Tr}(\beta\alpha^j) \) for all \( j \). Then:

\[
M_{r,i_2} = \text{Tr}(\beta\alpha^{i_2}) = \text{Tr}(\beta(s_0\alpha^{i_0} + s_1\alpha^{i_1})) = s_0\text{Tr}(\beta\alpha^{i_0}) + s_1\text{Tr}(\beta\alpha^{i_1}) = 0.
\]

From the argument for coverage of 2-sets of columns, there are exactly \( q - 1 \geq 1 \) such \( r \). Thus, \( D \in Z \), so \( \mathcal{D} \subseteq \mathcal{Z} \). Since \( |\mathcal{D}| = |\mathcal{Z}| \), we have that \( \mathcal{D} = \mathcal{Z} \).

We can then conclude that \( \mathcal{D} = \mathcal{N} \). Hence, \( \mathcal{T} = \mathcal{I} \), i.e. a triple of columns \( \{i_0, i_1, i_2\} \) is covered if and only if \( \{\alpha^{i_0}, \alpha^{i_1}, \alpha^{i_2}\} \) is linearly independent, i.e. 1 and 3 are equivalent.

The sets of 0s of this array correspond to certain BIBD and projective planes.

**Theorem 4.2.2.** Let \( f \) be a primitive polynomial \( f \) of degree 3 over \( \mathbb{F}_q \).

1. If \( M = M(f) \) is the subinterval array of \( f \), the set

\[
\mathcal{B} = \{\{a_1, \ldots, a_{q+1}\} : M_{i,a_1} = \ldots = M_{i,a_{q+1}} = 0 \text{ for some } 0 \leq i < q^3 - 1\}
\]

is the set of blocks of a projective plane of order \( q \).
2. Let $\Delta(M)$ be the ASC with facet set $\Lambda$ associated with $M = M(f)$ given in Theorem 4.2.1, and let $V = \{0, \ldots, q^2 + q\}$. Then $(V, \binom{V}{3} \setminus \Lambda)$ is a simple $(q^2 + q + 1, 3, q - 1)$-BIBD, and $(V, \Lambda)$ is a simple $(q^2 + q + 1, 3, q^2)$-BIBD.

Proof. 1. Let $V = \{0, \ldots, q^2 + q\}$. By Corollary 4.1.8, part 2, we only need to consider $0 \leq i < q^2 + q + 1$. For any pair of points $0 \leq i_0 < i_1 \leq q^2 + q$, as shown in the proof of Theorem 4.2.1, there is exactly one row $r$, $0 \leq r < q^2 + q + 1$ such that $M_{r,i_0} = M_{r,i_1} = 0$. Hence, $(\{0, \ldots, q^2 + q\}, B)$ is a $(q^2 + q + 1, q + 1, 1)$-BIBD, which is a projective plane of order $q$.

2. If, for each block of the projective plane as defined above, we take every 3-subset, then this collection of 3-sets forms a simple $(q^2 + q + 1, 3, q - 1)$-BIBD. By Theorem 4.2.1, these are precisely the triples of columns not covered in $M(f)$, i.e. $(V, \binom{V}{3} \setminus \Lambda)$.

We claim that $\Lambda$ is a simple $(q^2 + q + 1, 3, q^2)$-BIBD. Indeed, if $i_0, i_1 \in V$, there are exactly $q^2 + q - 1$ triples in $\binom{V}{3}$ containing $\{i_0, i_1\}$. Since the index of $(V, \binom{V}{3} \setminus \Lambda)$ is $q - 1$, the index of $(V, \Lambda)$ is $q^2 + q - 1 - (q - 1) = q^2$.

\[\square\]

**Definition 4.2.3.** A $(v, k, \lambda)$-difference set is a $k$-subset $H$ of an abelian group $G$ (written additively) of order $v$ such that for every $i \in G \setminus \{0\}$, we can write $i = w - u$ for $\lambda$ distinct choices of $w, u \in H$.

**Theorem 4.2.4** ([76, Theorem 2.1.1]). Let $q$ be a prime power, and select an integer $d \geq 0$. Let $\alpha$ be a generator of $\mathbb{F}_q^{\times}$ of degree $d + 1$. Then the set of integers:

$$D = \left\{ i : 0 \leq i < \frac{q^{d+1} - 1}{q - 1}, \text{Tr}(\alpha^i) = 0 \right\}$$

forms a difference set in $\mathbb{Z}_{\frac{q^{d+1} - 1}{q - 1}}$ with parameters:

$$\left( \frac{q^{d+1} - 1}{q - 1}, \frac{q^d - 1}{q - 1}, \frac{q^{d-1} - 1}{q - 1} \right).$$

**Corollary 4.2.5.** Let $f$ be a primitive polynomial of degree 3 over $\mathbb{F}_q$. Let $k = q^2 + q + 1$ and for any values $T = (a_0, a_1, a_2) \neq (0, 0, 0)$, consider the LFSR $S(f, T) = (a_i)$ generated by $f$. Define:

$$H = \{0 \leq i < k : a_i = 0\} \subset \mathbb{Z}_k.$$

We then have that $H$ is a $(q^2 + q + 1, q + 1, 1)$-difference set in $\mathbb{Z}_{q^2+q+1}$. Moreover, the blocks in $\mathcal{B}$ in Theorem 4.2.2 are the translates of this difference set.
Proof. Since $f$ is primitive over $\mathbb{F}_q$ of degree 3, it has a root $\alpha$ in $\mathbb{F}_{q^3}$ that generates $\mathbb{F}_{q^3}^\times$. We can then apply Theorem 4.2.4 for $d = 2$ to get the difference set $D$ over $\mathbb{Z}_{q^3 - 1}/\mathbb{Z}_{q^2 + q + 1}$ with parameters:

$$\left(\frac{q^3 - 1}{q - 1}, \frac{q^2 - 1}{q - 1}, \frac{q - 1}{q - 1}\right) = (q^2 + q + 1, q + 1, 1).$$

Let $\beta \in \mathbb{F}_{q^3}$ be the $\beta$ corresponding to initial values $T$. By Proposition 4.1.6, $\text{Tr}(\alpha^i \beta) = a_i$ for all $i$. Since $\beta \in \mathbb{F}_{q^3}$ and $\alpha$ generates $\mathbb{F}_{q^3}^\times$, we have that $\beta = \alpha^s$ for some $0 \leq s \leq q - 1$.

We now have:

$$H = \{i \in \mathbb{Z}_{q^3 - 1}/\mathbb{Z}_{q^2 + q + 1} : a_i = 0\}$$

$$= \{i \in \mathbb{Z}_{q^3 - 1}/\mathbb{Z}_{q^2 + q + 1} : \text{Tr}(\alpha^i \beta) = 0\}$$

$$= \{i \in \mathbb{Z}_{q^3 - 1}/\mathbb{Z}_{q^2 + q + 1} : \text{Tr}(\alpha^i \alpha^s) = 0\}$$

$$= \{i \in \mathbb{Z}_{q^3 - 1}/\mathbb{Z}_{q^2 + q + 1} : \text{Tr}(\alpha^{i+s}) = 0\}$$

This is just a translate by $s$ of the difference set $D$ in Theorem 4.2.4 and is thus a difference set with the same parameter set. The same can be said of the blocks in $\mathcal{B}$. □

Lemma 4.2.6. Let $q$ be a prime power, and let $H$ be a $(q^2 + q + 1, q + 1, 1)$-difference set in $\mathbb{Z}_{q^2 + q + 1}$. Let $\{a, b, c\} \subset H$ with $a < b < c$. Then, $(b - c + a) \mod (q^2 + q + 1) \notin H$.

Proof. Assume the contrary, i.e. $\{a, b, c\} \subset H$ with $a < b < c$, and $(b - c + a) \mod (q^2 + q + 1) \in H$. Then $c - b = (c) - (b)$, and $c - b = (a) - (b - c + a)$, so the difference $c - b$ appears twice in $H$, contradicting the fact that $\lambda = 1$. □

Lemma 4.2.7. Let $f$ be a primitive polynomial of degree 3 over $\mathbb{F}_q$. Consider a triple of columns $D = \{a, b, c\}$ with $0 \leq a < b < c < q^2 + q + 1$ of $M(f)$. If $D$ is uncovered in $M(f)$, then $D' = \{a, b, b - c + a\}$ is covered in $M(f)$.

Proof. Let $k = q^2 + q + 1$. If $D$ is uncovered, then by Theorem 4.2.1, there is a row $r$, $0 \leq r < q^3 - 1$ such that $M_{r,a} = M_{r,b} = M_{r,c} = 0$. Thus, by Corollary 4.2.5, there is a unique block $B \in \mathcal{B}$ of Theorem 4.2.2 containing $\{a, b, c\}$. By Lemma 4.2.6, $(b - c + a) \mod k \notin B$, so by Corollary 4.2.5, $M_{r,b-c+a} \neq 0$. Then, from Theorem 4.2.1, $D'$ is covered in $M(f)$. □
Definition 4.2.8. For a sequence $S = (a_i)$ with period $p$, we define the reversal sequence of $S$, $S^{\text{rev}} = (b_i)$, by setting $b_i = (a_{-i \mod p})$; if $S$ is defined by a nonzero polynomial $f$, $S^{\text{rev}}$ is equivalent to the sequence defined by the reciprocal polynomial $\hat{f} = f(1/x)x^{\deg(f)}$.

Lemma 4.2.9. Let $f$ be a primitive polynomial of degree 3 over $\mathbb{F}_q$ and let $\hat{f}$ be the reciprocal polynomial of $f$. If a triple of columns $D = \{a, b, c\}$ with $0 \leq a < b < c < q^2 + q + 1$ is not covered in $M(f)$, then $D$ is covered in $M(\hat{f})$.

Proof. The rows of $M(\hat{f})$ are all the length-$(q^2 + q + 1)$ subintervals of the reversal of the LFSR defined by $f$ and the all-zero row, or equivalently, the reversal of the rows of $M(f)$. Thus, for any row $r$ of $M(f)$, $0 \leq r < q^3 - 1$, there exists a row $r'$, $0 \leq r' < q^3 - 1$ in $M(\hat{f})$ such that $M(\hat{f})_{r',i} = M(f)_{r,a+b-i}$ for all $0 \leq i < q^2 + q + 1$. Thus, taking $i = a$, we have $M(\hat{f})_{r',a} = M(f)_{r,b}$; taking $i = b$, we have $M(\hat{f})_{r',b} = M(f)_{r,a}$; and taking $i = c$, we have $M(\hat{f})_{r',c} = M(f)_{r,a+b-c}$. By Lemma 4.2.7, $D' = \{a, b, -c + a\}$ is covered in $M(f)$, so $D$ is covered in $M(\hat{f})$.

We then have the following construction for strength-3 covering arrays.

Theorem 4.2.10. For any prime power $q$ and degree-3 primitive polynomial $f$ over $\mathbb{F}_q$, the vertical concatenation of $M(f)$ with $M(\hat{f})$ with one copy of the all-zero row removed is a $\text{CA}(2q^3 - 1; 3, q^2 + q + 1, q)$.

Proof. Lemma 4.2.9 shows that any triple is covered in $M(f)$ or in $M(\hat{f})$.

Now, using our construction, we can find strength-3 covering arrays for any number of levels $v$ simply by constructing an array for a prime power $q \geq v$ and then using the fusion operation of Theorem 2.2.15 to drop $q - v$ symbols to get an array over $v$ levels. Improvements in known results for a non-prime power $g$ come not only from constructing an array for the next prime power $q$, but possibly for higher prime powers as well, which result in arrays with more factors. We can also incorporate the factor increase operation of Theorem 2.2.16, which gives the following:

Corollary 4.2.11. Let $g \geq 2$, $s \geq 0$, and let $q$ be a prime power with $q \geq g$. Then:

$$\text{CAN}(3, q^2 + q + s, g) \leq 2q^3 - 1 - 2(q - g) + sg^2(g - 1).$$

Proof. This follows from combining and iterating Theorem 2.2.15 to drop $q - g$ factors, and Theorem 2.2.16 to add $s$ levels, using the fact that for any $k, g \geq 1$, $\text{CAN}(1, k, g) = g$.

4.3 New Upper Bounds and Recursive Constructions

Using the construction that forms the proof of Theorem 4.2.10, we obtain covering arrays of sizes $N$ given in Table 4.1. We then apply the fusion and factor increase operations as combined in Corollary 4.2.11, and give the new bounds generated by this technique in Table 4.2, where, in the Source column, we write $A_n$ to indicate adding $n$ factors, and $F_m$ to indicate performing $m$ fusions, with operations performed from left to right. We now use the covering arrays of Theorem 4.2.10 and those constructed as in Table 4.2 as ingredients in the recursive constructions detailed in Section 2.2.3. Improvements to known bounds are given in Table 4.8, which appears at the end of the chapter in Section 4.5 due to size. We note that the data from [25] that we use in this section is from December, 2012; improvements may have been made since. As there is no central database of the best known values for some of the ingredients in the recursive constructions, we have confirmed with Colbourn the table results that arise from using our covering arrays as an ingredient.

<table>
<thead>
<tr>
<th>$g$</th>
<th>$k$</th>
<th>$N$</th>
<th>$g$</th>
<th>$k$</th>
<th>$N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7</td>
<td>15</td>
<td>16</td>
<td>273</td>
<td>8191</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>53</td>
<td>17</td>
<td>307</td>
<td>9825</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>127</td>
<td>19</td>
<td>381</td>
<td>13717</td>
</tr>
<tr>
<td>5</td>
<td>31</td>
<td>249</td>
<td>23</td>
<td>553</td>
<td>24333</td>
</tr>
<tr>
<td>7</td>
<td>57</td>
<td>685</td>
<td>25</td>
<td>651</td>
<td>31249</td>
</tr>
<tr>
<td>8</td>
<td>73</td>
<td>1023</td>
<td>27</td>
<td>757</td>
<td>39365</td>
</tr>
<tr>
<td>9</td>
<td>91</td>
<td>1457</td>
<td>29</td>
<td>871</td>
<td>48777</td>
</tr>
<tr>
<td>11</td>
<td>133</td>
<td>2661</td>
<td>31</td>
<td>993</td>
<td>59581</td>
</tr>
<tr>
<td>13</td>
<td>183</td>
<td>4393</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Sizes $N$ for $CA(N; 3, k, g)$ using the new construction. Values in bold indicate new values.

Each entry in Table 4.8 provides details as to how the bound was determined and the improvement over the previous values of the tables at [25]. The table is organized by $g$, the number of levels, and each entry lists the $k$ and $N$ for our new $CA(N; 3, k, g)$. The column Prev$N$ consists of the old best known bound for this value $k$. The Method column illustrates how the array was constructed, by providing the technique used followed by a list of the ingredients. We refer to our arrays $CA(2q^3 - 1; 3, q^2 + q + 1, q)$ as RMS($q$). The
Table 4.2: Sizes $N$ for $\text{CA}(N; 3, k, g)$ from applying Corollary 4.2.11 to the arrays generated from the construction. We include only those results that improve the previously known best bounds.

<table>
<thead>
<tr>
<th>$g$</th>
<th>$k$</th>
<th>$N$</th>
<th>Source</th>
<th>$g$</th>
<th>$k$</th>
<th>$N$</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>32</td>
<td>349</td>
<td>A1</td>
<td>18</td>
<td>381</td>
<td>13715</td>
<td>F1</td>
</tr>
<tr>
<td>7</td>
<td>58</td>
<td>979</td>
<td>A1</td>
<td>19</td>
<td>382</td>
<td>19223</td>
<td>F1A1</td>
</tr>
<tr>
<td>8</td>
<td>91</td>
<td>1455</td>
<td>F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>92</td>
<td>2105</td>
<td>A1</td>
<td>19</td>
<td>381</td>
<td>20215</td>
<td>A1</td>
</tr>
<tr>
<td></td>
<td>133</td>
<td>2657</td>
<td>F2</td>
<td>20</td>
<td>553</td>
<td>24325</td>
<td>F4</td>
</tr>
<tr>
<td>10</td>
<td>133</td>
<td>2659</td>
<td>F1</td>
<td>21</td>
<td>553</td>
<td>24327</td>
<td>F3</td>
</tr>
<tr>
<td></td>
<td>134</td>
<td>3559</td>
<td>F1A1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>134</td>
<td>3871</td>
<td>A1</td>
<td>22</td>
<td>553</td>
<td>24329</td>
<td>F2</td>
</tr>
<tr>
<td>12</td>
<td>183</td>
<td>4391</td>
<td>F1</td>
<td>23</td>
<td>553</td>
<td>24331</td>
<td>F1</td>
</tr>
<tr>
<td></td>
<td>184</td>
<td>5975</td>
<td>F1A1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>184</td>
<td>6421</td>
<td>A1</td>
<td>24</td>
<td>553</td>
<td>24345</td>
<td>F2</td>
</tr>
<tr>
<td></td>
<td>273</td>
<td>8185</td>
<td>F3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>273</td>
<td>8187</td>
<td>F2</td>
<td>25</td>
<td>553</td>
<td>24357</td>
<td>F4</td>
</tr>
<tr>
<td></td>
<td>307</td>
<td>9819</td>
<td>F3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>273</td>
<td>8189</td>
<td>F1</td>
<td>26</td>
<td>553</td>
<td>24361</td>
<td>F2</td>
</tr>
<tr>
<td></td>
<td>307</td>
<td>9821</td>
<td>F2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>381</td>
<td>13709</td>
<td>F4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>307</td>
<td>9823</td>
<td>F1</td>
<td>27</td>
<td>553</td>
<td>24369</td>
<td>F4</td>
</tr>
<tr>
<td></td>
<td>381</td>
<td>13711</td>
<td>F3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>381</td>
<td>13713</td>
<td>F2</td>
<td>28</td>
<td>553</td>
<td>24379</td>
<td>F3</td>
</tr>
<tr>
<td></td>
<td>382</td>
<td>18337</td>
<td>F2A1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

other techniques are explained below and the exact forms of the values in the Method column explained in Table 4.3.

With the best current known values for $\text{CAN}(2, k, g)$ as taken from [25], we use the following generalized Roux doubling construction as given by Chateauneuf and Kreher [15] and detailed in Theorem 2.2.23:

\[
\text{CAN}(3, 2k, g) \leq \text{CAN}(3, k, g) + (g - 1)\text{CAN}(2, k, g). \quad \text{(CKD)}
\]

For prime powers $g \geq 3$, we recursively build arrays with the following generalized
Roux construction as shown by Colbourn, Martirosyan, Trung, and Walker in [35] and stated in Theorem 2.2.26:

\[
\text{CAN}(3, gk, g) \leq \text{CAN}(3, k, g) + (g - 1)\text{CAN}(2, k, g) + g^3 - g^2. \tag{CMTW}
\]

This strategy can be iterated to derive bounds for \(\text{CAN}(3, g^i k, g)\).

Colbourn [29] improves upon another recursive construction, as given by Cohen, Colbourn, and Ling in [24], which gives the following:

\[
\text{CAN}(3, kl, g) \leq \text{CAN}(3, l, g) + \text{CAN}(3, k, g) + \text{RCODN}(2, l, g) \times \text{CODN}(2, k, g). \tag{CCL}
\]

His paper provides tables of the best known values for \(\text{RCODN}(2, l, g)\) for \(1 \leq l \leq 10000, 2 \leq g \leq 25\). It is not always the case that, for \(g < h\), \(\text{RCODN}(2, k, g) \leq \text{RCODN}(2, k, h)\). Thus, in order to minimize the array sizes in the CCL construction, we can find \(h \geq g\) that minimizes \(\text{RCODN}(2, l, h) \times \text{CODN}(2, k, h)\) and use this value instead, dropping levels as required in the final array.

We also applied Hartman and Raskin’s construction [50] as given in Theorem 2.2.32, this construction, however, yielded no improvements to known bounds.

<table>
<thead>
<tr>
<th>Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMS(q)</td>
<td>CA((2q^3 - 1; 3, q^2 + q + 1, q))</td>
</tr>
<tr>
<td>BF</td>
<td>Fusion of B.</td>
</tr>
<tr>
<td>BA</td>
<td>Factor increase of B.</td>
</tr>
<tr>
<td>CKD((k,N2))</td>
<td>CK doubling for (k) factors with a CA((N2; 2, k, g)).</td>
</tr>
<tr>
<td>CMTW((k,N2))</td>
<td>CMTW for (k) factors with a CA((N2; 2, k, g)).</td>
</tr>
<tr>
<td>CCL((l,k,N1,N2,R,D)):</td>
<td>CCL with values (l, k, a \text{CA}(N1; 3, l, g),)</td>
</tr>
<tr>
<td></td>
<td>a CA((N2; 3, k, g)), an RCOD((R; 2, l, g)),</td>
</tr>
<tr>
<td></td>
<td>and a COD((D; 2, k, g)).</td>
</tr>
<tr>
<td>CCL((l,k,N1,N2,h,R,D)):</td>
<td>CCL with values (l, k, h, a \text{CA}(N1; 3, l, g),)</td>
</tr>
<tr>
<td></td>
<td>a CA((N2; 3, k, g)), an RCOD((R; 2, l, h)),</td>
</tr>
<tr>
<td></td>
<td>and a COD((D; 2, k, h)).</td>
</tr>
</tbody>
</table>

Table 4.3: Explanation of the Method column of Table 4.8
4.4 Extending the Construction to Higher $t$

Given a degree-2 primitive polynomial $f$ over $\mathbb{F}_q$, $M(f)$, the subinterval array of $f$, yields an OA$(2, q+1, q)$, and thus does not require anything to be vertically concatenated. This corresponds to the standard finite field construction as given in Theorem 2.2.20. The array is also a CA$(q^2; 2, q+1, q)$, which is optimal.

Our strategy to generate strength-3 covering arrays is essentially a generalization and extension of this. We take the VOA $M$ as described in Theorem 4.2.1 and a permutation group of order 2 over the columns, namely $\{R, I = R^2\}$ where $R$ is the reversal permutation (which is equivalent to taking the subintervals of the sequence of the reciprocal polynomial). For each permutation, we permute the columns of $M$, and take the vertical concatenation of these permuted arrays, removing one copy of the all-zero row. The result is a CA$(2q^3-1; 3, q^2 + q + 1, q)$.

We give the following theorem regarding the structure of arrays over LFSR sequences defined by primitive polynomials of higher degree, which provides us with more information about what is uncovered by constructing an array over the rows of the LFSR with an all-zero row.

**Theorem 4.4.1.** Let $f$ be a primitive polynomial of degree $m$ over a finite field $\mathbb{F}_q$ with root $\alpha$ in the extension field $\mathbb{F}_{q^m}$, and write $k = \frac{q^m-1}{q-1}$. Let $M = [c_0, \ldots, c_{k-1}]$ be the $q^m \times k$ array that arises by taking as rows all subwords of length $k$ from the LFSR sequence as defined by $f$, along with an all-zero row. Then, for $2 < s \leq m$, a set of $s$ columns $C = \{c_{i_0}, \ldots, c_{i_{s-1}}\}$ is uncovered in $M$ if and only if the set of vectors $\{\alpha^{i_0}, \ldots, \alpha^{i_{s-1}}\}$ is linearly dependent over $\mathbb{F}_q$.

**Proof.** As every value appears in the LFSR sequence, we have coverage of all 1-sets of columns. From Theorem 4.1.7, we know that the LFSR sequence has the 2-tuple balance property, which guarantees coverage of all the 2-sets of columns. Furthermore, any $m+1$ set of columns cannot be covered since we only have $q^m$ rows in our array, and thus, we need only consider values $2 < s \leq m$. Assume we have an $s$-set $C$ of columns as defined above. The set of vectors as given above is linearly dependent if and only if we have that, for some values $a_0, \ldots, a_{s-1} \in \mathbb{F}_q$ not all zero:

$$a_0\alpha^{i_0} + \ldots + a_{s-1}\alpha^{i_{s-1}} = 0.$$ 

Assume without loss of generality that $a_{s-1} \neq 0$, so $a_{s-1}$ has inverse $a_{s-1}^{-1}$. Then we can write:

$$\alpha^{i_{s-1}} = -a_{s-1}^{-1}(a_0\alpha^{i_0} + \ldots + a_{s-2}\alpha^{i_{s-2}}).$$
If there is no row \( r \) in \( M \) other than the all-zero row where \( r_{i_0} = \ldots = r_{i_{s-2}} = 0 \), then \( C \) is uncovered, as no tuples of the form \( (0, \ldots, 0, h) \) for \( h \neq 0 \) then appear. Hence, assume such a row \( r \) exists. Then there is a \( \beta \in \mathbb{F}_q^m \) such that \( r_j = \text{Tr}(\beta \alpha^j) \) for all \( 0 \leq j \leq k - 1 \). Consider:

\[
    r_{i_{s-1}} = \text{Tr}(\beta \alpha^{i_{s-1}}) \\
    = \text{Tr}(\beta(-a_{s-1}^{-1}(a_0 \alpha^{i_0} + \ldots + a_{s-2} \alpha^{i_{s-2}}))) \\
    = -a_{s-1}^{-1}(a_0 \text{Tr}(\beta \alpha^{i_0}) + \ldots + a_{s-2} \text{Tr}(\beta \alpha^{i_{s-2}})) \\
    = -a_{s-1}^{-1}(a_0 r_{i_0} + \ldots + a_{s-2} r_{i_{s-2}}) \\
    = 0.
\]

Thus, again it is impossible to have any tuples of the form \( (0, \ldots, 0, h) \) over \( C \) for \( h \neq 0 \), so \( C \) is uncovered.

Now assume we have some set \( C \) of columns that is uncovered. We can then extend this set to a full \( m \)-set \( C' = \{i_0, \ldots, i_{m-1}\} \) that will also be uncovered by virtue of \( C \) being uncovered. As generated by the recurrence relation defining the sequence, we have a map from the first \( m \) columns to \( C \), and furthermore, this map must be linear; thus it is a linear transformation from \( \mathbb{F}_q^m \) to \( \mathbb{F}_q^m \). We note that the \( q^m \times s \) subarray over the first \( m \) columns must be covered as these values can be considered as the seeds to the linear feedback shift register sequence. Since \( C \) is uncovered, there is some \( m \)-tuple missing from the \( q^m \times s \) subarray over \( C \), and thus this transformation is not surjective, so the column space is not full-dimension, and thus the dimension of the null space is nonzero.

Hence, if we consider the inverse image of the zero vector \( (0, \ldots, 0) \in \mathbb{F}_q^m \) under the transformation, we obtain a subset of \( \mathbb{F}_q^m \) that contains vectors other than just the zero vector, and so the \( q^m \times s \) subarray over \( C \) has more than one all-zero row, and thus there is some row of \( M \) other than the all-zero row where the columns \( C \) all assume the value zero.

This is equivalent to having a linear dependency over the columns of \( C \), i.e. there exist constants \( a_0, \ldots, a_{s-1} \in \mathbb{F}_q \) not all zero such that:

\[
    a_0 c_{i_0} + \ldots + a_{s-1} c_{i_{s-1}} = 0. 
\]

We have that, for any row \( r \), there exists a distinct \( \beta \in \mathbb{F}_q^m \) such that \( r_j = \text{Tr}(\beta \alpha^j) \) for all \( 0 \leq j \leq k - 1 \). From Equation \( \ref{4.1} \), since the linear dependency holds across the columns as vectors, it holds across the individual row entries themselves, giving that for
all \( \beta \in \mathbb{F}_q^m \):

\[
a_0 \text{Tr}(\beta a^0) + \ldots + a_{s-1} \text{Tr}(\beta a^{s-1}) = 0 \\
\text{Tr}(\beta(a_0 a^0 + \ldots + a_{s-1} a^{s-1})) = 0
\]

We have that \( b = a_0 a^0 + \ldots + a_{s-1} a^{s-1} \in \mathbb{F}_q^m \), and if \( b \neq 0 \), then \( b \) has an inverse, say \( b^{-1} \in \mathbb{F}_q^m \). Thus, as the expression holds for all \( \beta \in \mathbb{F}_q^m \), this gives that for any \( w \in \mathbb{F}_q^m \), we can write \( w = (wb^{-1})b \), so by choosing \( \beta = wb^{-1} \), we have:

\[
0 = \text{Tr}(\beta b) = \text{Tr}((wb^{-1})b) = \text{Tr}(w).
\]

However, this implies that the trace function for \( \mathbb{F}_q^m \) over \( \mathbb{F}_q \) is always zero, which is impossible. Thus, we must have that \( b = 0 \). Hence:

\[
a_0 a^0 + \ldots + a_{s-1} a^{s-1} = 0
\]

where \( a_0, \ldots, a_{s-1} \) are not all zero, implying that \( \{a^0, \ldots, a^{s-1}\} \) is linearly dependent.

The sets of zeros over the rows of the subinterval array correspond to a BIBD as per the following theorem. We note that for \( m > 3 \), we have \( \lambda > 1 \), so the BIBD is not a projective plane.

**Theorem 4.4.2.** Let \( f \) be a primitive polynomial over \( \mathbb{F}_q \) of degree \( m \), and let \( M = M(f) \) be the subinterval array of \( f \). Let:

\[
k = \frac{q^m - 1}{q - 1}, \quad z = \frac{q^m - 1}{q - 1}, \quad \lambda = \frac{q^{m-2} - 1}{q - 1}.
\]

Then each row of \( M \) has exactly \( z \) zeros, and the set:

\[
\mathcal{B} = \{\{a_1, \ldots, a_{2z-1}\} : M_{i,a_1} = \ldots = M_{i,a_{z-1}} = 0 \text{ for some } 0 \leq i \leq k - 1\}
\]

is the set of blocks of a \((k, z, \lambda)\)-BIBD.

**Proof.** The count of the number of zeros in each length-\( k \) subinterval of the array follows from Corollary 4.1.8 and the pattern of zeros repeats every \( k \) rows, so we focus only on the first \( k \) rows. The sequence has the 2-tuple property by Theorem 4.1.7 so for any pair \( 0 \leq i < j \leq k - 1 \), we have balanced coverage over the pair in the full array with the zero row, i.e. every pair of \( \mathbb{Z}_q \times \mathbb{Z}_q \) appears in \( q^m/q^2 = q^{m-2} \) rows of \( M \) in positions \( i, j \). Thus, without the zero row, we have that the pair \((0, 0)\) appears in \( q^{m-2} - 1 \) rows of \( M \), of which \( \frac{q^{m-2} - 1}{q - 1} \) appear in the first \( k \) rows. \( \square \)
We now investigate the possibility of generalizing this construction to create higher strength covering arrays. To do so, we first further explore the strength-3 construction to gain a deeper understanding of the issues that may arise in such a generalization.

We begin by examining several properties of the permutation $R$, the reversal permutation used to extend the constructed VOA into a full strength-3 CA. We first investigate how special the reversal permutation is in the construction. For $q \in \{2, 3\}$, we sampled random order-2 permutations with exactly one fixed point (since $q^2 + q + 1$ is odd), and applied these permutations to the columns of the VOA to determine if the result was a strength-3 CA. For each $q$, 10,000 trials were performed. For $q = 2$, 53.61% of the generated permutations resulted in a full CA, and for $q = 3$, 1.43% of the permutations resulted in a full CA, indicating that many permutations perform the same role as the reversal permutation in the construction.

We proceeded to investigate the effect, for $q \in \{2, 3\}$, of varying the number of fixed points in randomly generated order-2 permutations on their ability to generate a full strength-3 CA. For each $i \geq 0$, we generated 50,000 order-2 permutations with $2i + 1$ fixed points, and in all cases, we were unable to find permutations with more than one fixed point that could create a full CA from these VOA. Thus, in all our trials for $t > 3$, we always minimize the number of fixed points.

We then attempted to generate higher strength covering arrays by applying a similar strategy as used in the construction in Theorem 4.2.10. For $t > 3$, let $f$ be a primitive polynomial of degree $t$ over $\mathbb{F}_q$, $k = \frac{q^t - 1}{q - 1}$, and $M = M(f)$ be the subinterval array of $f$. For various values of $n$, using a permutation group of order $n$ over $\{0, \ldots, k - 1\}$, we took all images of $M$ with its columns permuted by each permutation and vertically concatenated the images, removing all but one copy of the all-zero row, to get an array of dimension $(n(q^t - 1) + 1) \times k$. We then investigated how close we were to a $CA(n(q^t - 1) + 1; t, k, q)$ by measuring the number of $t$-sets from $\{0, \ldots, k - 1\}$ that remained uncovered in the array.

As a case study, we focus on the results for $q = 3$, $t = 4$, which has $k = \frac{3^4 - 1}{3 - 1} = 40$ columns, and consider the permutation groups of size $n \in \{3, 4\}$: there is precisely one group of order three, namely $\mathbb{Z}_3$, the cyclic group, and there are exactly two groups of order four, namely $\mathbb{Z}_4$ (the cyclic group of order 4), and $\mathbb{Z}_2 \times \mathbb{Z}_2$ (the Klein four-group). We created permutation subgroups of $S_k$ isomorphic to these, minimizing the number of fixed points, and then determined the number of uncovered 4-sets in the array consisting of $n$ vertically concatenated copies of $M(f)$, each with their columns permuted by a permutation in the group, removing $n - 1$ copies of the all-zero row. The results across
all permutation groups are given in Table 4.4. In no case were we able to generate a strength-4 covering array.

<table>
<thead>
<tr>
<th>Group</th>
<th>#Trials</th>
<th>Min</th>
<th>Max</th>
<th>Avg</th>
<th>Mdn</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{Z}_3$</td>
<td>361,378</td>
<td>2251</td>
<td>3607</td>
<td>2718.87</td>
<td>2709</td>
</tr>
<tr>
<td>$\mathbb{Z}_4$</td>
<td>296,114</td>
<td>560</td>
<td>1570</td>
<td>840.35</td>
<td>835</td>
</tr>
<tr>
<td>$\mathbb{Z}_2 \times \mathbb{Z}_2$</td>
<td>288,970</td>
<td>669</td>
<td>2199</td>
<td>948.53</td>
<td>936</td>
</tr>
</tbody>
</table>

Table 4.4: The number of uncovered 4-sets of columns using random permutation groups, where the total number of 4-sets is 91390.

In an attempt to improve our results, focusing on $\mathbb{Z}_4$ since it tended to come the closest to a CA based on the random experiments and primitive polynomial $f = x^4 + 2x + 2$, we implemented two local search algorithms, namely hill-climbing [56] and great deluge [40].

These are both heuristic algorithms in which, starting from a random solution, the solution space is traversed by defining a neighbourhood for each solution and then optimizing by successively moving from the current solution to another in its neighbourhood subject to certain rules, until the algorithm is no longer able to do so. In both these algorithms, the solution space is the subgroups of $S_k$ isomorphic to $\mathbb{Z}_4$, which can be represented by an order-4 generator permutation, and the measure of quality of a solution is the number of covered 4-sets in the array created from the permutation group. Hill climbing only allows upward and lateral moves, i.e. moves in which the quality of the neighbouring solution is greater than or equal to the current solution. In practice, many solution spaces have a large number of local maxima whose quality may be far less than the global maximum. As hill climbing stops upon reaching a local maxima, we also chose to use the great deluge algorithm since it allows for moves to solutions of lower quality to overcome this problem, and performs comparably to simulated annealing but has fewer parameters that are easier to tune. The algorithm is analogous to a landscape that is flooding, with the water level representing the minimally accepted solution quality at any given time: at each step, the algorithm may move to any solution in the neighbourhood provided that it is above the current water level. If the new solution is an improvement on the best solution seen so far, rain falls according to a constant rain speed, thus increasing the water level. The algorithm iterates until no improvement is seen in a large number of iterations.
In both cases, for a permutation $p \in S_k$ and a parameter $h \in \{2, 4, 6\}$, we defined the neighbourhood function $N_h : S_k \to \mathcal{P}(S_k)$ used in the searches as:

$$N_h(p) = \{s^{-1}ps : s \in S_k, \text{fix}(s) = k - h\}$$

where $\text{fix}(s) = |\{i \in \mathbb{Z}_k : s(i) = i\}|$. We note that if $\langle p \rangle \cong \mathbb{Z}_4$, then for all $p' \in N_n(p)$, $\langle p' \rangle \cong \mathbb{Z}_4$ as well. We began with a random permutation $p$ such that $\langle p \rangle \cong \mathbb{Z}_4$, and then at each iteration of the algorithms, we randomly generated a sample neighbourhood $N_h^S(p) \subset N_h(p)$ of size 100. In hill-climbing, if there was a permutation $p' \in N_h^S(p)$ of higher quality, we moved to $p'$. If not, we stayed at $p$, and then iterated. If 10 rounds passed without improvement, we terminated and output $p$.

In the case of the great deluge algorithm, we set the initial water level to the minimum number of covered $t$-sets we saw in random sampling of permutation groups. Due to the intensive nature of the algorithm, a fairly high rain speed was necessary, so we chose to use 50. We kept track of the best permutation seen so far, $p_b$. At each iteration, we selected the $p' \in N_h^S(p)$ of highest quality: if the number of covered $t$-sets was greater than the water level, we moved to $p'$, rained if $p'$ was greater than the previous $p_b$, and then iterated. If 10 rounds passed without improvement in the best value seen so far, we terminated and output $p_b$.

Using hill climbing and great deluge, we were able to find permutations that generated arrays with far fewer uncovered 4-sets than by simply using random sampling. A summary of the results is shown in Table 4.5 with the number of uncovered 4-sets given.

By using this technique, we performed experiments with the great deluge algorithm to determine an upper bound $s(t, q)$ such that we were able to generate a $\text{CA}(s(t, q)(q^t - 1) + 1; t, q^t - 1)$ using a subgroup of $S_{q^t - 1}$ isomorphic to $\mathbb{Z}_{s(t, q)}$. Initially, based on the fact that $s(2, q)$ and $s(3, q)$ are independent of value of $q$, we hoped that we could find a function that was defined exclusively in terms of $t$; however, results suggest that this is not the case for $t \geq 4$. The experimentally determined upper bounds on $s(t, q)$ are presented in Table 4.6. We found a $\text{CA}(61; 4, 15, 2)$ from the LFSR defined by the primitive polynomial $f = x^4 + x + 1$ over $\mathbb{F}_2$, and vertically concatenating the images of $M(f)$ under each permutation in the order-4 group generated by:

$$(0 3 12 13)(1 11 7 5)(2 10 14 8)(4)(6 9).$$

A $\text{CA}(481; 4, 40, 3)$ can be built from the primitive polynomial $x^4 + 2x + 2$ over $\mathbb{F}_3$ and
Table 4.5: The effects of refining permutations generating a group isomorphic to $\mathbb{Z}_4$ through local search algorithms on the number of uncovered 4-sets, where the total number of 4-sets is 91390.

<table>
<thead>
<tr>
<th>Alg</th>
<th>$n$</th>
<th>#Trials</th>
<th>Min</th>
<th>Max</th>
<th>Avg</th>
<th>Mdn</th>
</tr>
</thead>
<tbody>
<tr>
<td>HC</td>
<td>2</td>
<td>1184</td>
<td>434</td>
<td>622</td>
<td>531.70</td>
<td>532</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1213</td>
<td>445</td>
<td>601</td>
<td>527.98</td>
<td>529</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1231</td>
<td>469</td>
<td>615</td>
<td>538.80</td>
<td>539</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>1351</td>
<td>487</td>
<td>630</td>
<td>573.78</td>
<td>575</td>
</tr>
<tr>
<td>GD</td>
<td>2</td>
<td>892</td>
<td>456</td>
<td>572</td>
<td>515.77</td>
<td>515</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>883</td>
<td>449</td>
<td>579</td>
<td>522.53</td>
<td>523</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>768</td>
<td>469</td>
<td>594</td>
<td>536.17</td>
<td>536</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>865</td>
<td>505</td>
<td>628</td>
<td>574.38</td>
<td>584</td>
</tr>
</tbody>
</table>

Table 4.6: Values $s(t, q)$ such that there exists a CA($s(t, q)(q^t - 1) + 1; t, q^t - 1, q$).

<table>
<thead>
<tr>
<th>$q$</th>
<th>$t = 2$</th>
<th>$t = 3$</th>
<th>$t = 4$</th>
<th>$t = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Unfortunately, in all three cases, smaller covering arrays are known for these parameters.

For an $N \times k$ array $M$ over $\mathbb{F}_q$, a hole of size $t$ is a set of $1 \leq t \leq k$ columns that is uncovered in $M$. Perhaps the reason that it became markedly more difficult to extend our construction to the case $t = 4$ was because of the large number of smaller holes of size $t - 1 = 3$. In the case $t = 3$, by limiting the array to $q^2 + q + 1$ columns, there were
no holes of size $t - 1 = 2$ in the array. In order to attempt a similar strategy for $t = 4$, we wrote an algorithm that, given a degree-4 primitive polynomial $f \in \mathbb{F}_q[x]$, built up the array from the LFSR over $f$, column by column, until a size-3 hole appeared, at which point, the column was discarded and the algorithm halted. This is equivalent to finding the degree of the lowest degree trinomial $g \in \mathbb{F}_q[x]$ such that $f | g$, as discussed in [72]. This strategy, unfortunately, generated arrays with very few columns: for $q = 3$, already many of the degree-4 primitive polynomials themselves are trinomials, and for those that are not, there was a degree-5 trinomial that they divided, so it was only possible to find arrays over 4 or 5 columns that did not contain size-3 holes. For $q = 5$, the number of columns ranged between 5 and 11; however, this is still very small compared to the $5^4 - 1 = 156$ columns over the full construction. Even if we were able to generate a covering array of size $2(5^4 - 1) + 1 = 1251$ over 11 factors, this would offer no improvements on the known upper bounds.

Finally, we provide a conjecture on the possibility of extending our construction to higher degree primitive polynomials, the necessary number of permuted copies of the subinterval array that must be appended to obtain a full covering array, and thus the size of the covering array generated by this technique.

**Conjecture 4.4.3.** Let $f$ be a primitive polynomial over $\mathbb{F}_q$ of degree $t \geq 3$, let $k = \frac{q^t - 1}{q - 1} - 1$, and let $M = M(f)$ be the $(q^t - 1) \times k$ subinterval array of $f$. For a permutation $\sigma \in S_k$, we write $M^{\sigma}$ to indicate the copy of $M$ with its columns permuted by the action of $\sigma$. Then there exists a set of $s(t, q) = \frac{q^{t-2} - 1}{q - 1}$ nonidentity permutations in $S_k$, say $\{\sigma_1, \ldots, \sigma_s\}$ such that:

$$\begin{bmatrix} M^T | (M^{\sigma_1})^T | \ldots | (M^{\sigma_s})^T \end{bmatrix}^T$$

is a $CA((s(t, q) + 1)(q^t - 1) + 1; t, k, q)$.

Our reason for proposing $s = \frac{q^{t-2} - 1}{q - 1}$ as the necessary number of permutations mirrors the proof for the case $t = 3$. For $t = 3$, we showed that the set of zeros in a row of the subinterval matrix formed a difference set with $\lambda = 1$, and thus we found one permutation such that if it left any uncovered triple uncovered, this would violate the $\lambda = 1$ property of the difference set. Here, the set of zeros form a $(\frac{q^t - 1}{q - 1}, \frac{q^{t-1} - 1}{q - 1}, \frac{q^{t-2} - 1}{q - 1})$-difference set. Thus, if we can find a set of $s(t, q)$ permutations such that they similarly violate the property $\lambda = s(t, q)$ of the difference set by allowing us to write some difference $s + 1$ times, then they guarantee full coverage, and thus can be used to form a full strength-$t$ covering array. We further note that our experimentally determined permutation groups,
as in Table 4.6 are equal to $s(t, 2)$ for $t \in \{4, 5\}$, $q = 2$, and are equal to $s(4, 3) + 1$ for $t = 4$, $q = 3$.

The tables of [25] list the best known bounds $\text{CAN}(t, k, g)$ for $2 \leq t \leq 6$, $1 \leq k \leq 10000$, and $2 \leq g \leq 25$. If Conjecture 4.4.3 holds, we would significantly improve the known bounds. We list the improvements in Table 4.7.

<table>
<thead>
<tr>
<th>$t$</th>
<th>$q$</th>
<th>$k$</th>
<th>$N$</th>
<th>$N_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
<td>40</td>
<td>401</td>
<td>468</td>
</tr>
<tr>
<td>4</td>
<td>85</td>
<td>1531</td>
<td>1848</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>156</td>
<td>4369</td>
<td>6279</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>400</td>
<td>21601</td>
<td>24643</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>585</td>
<td>40951</td>
<td>45340</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>820</td>
<td>72161</td>
<td>78281</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>341</td>
<td>22507</td>
<td>37000</td>
</tr>
<tr>
<td>5</td>
<td>781</td>
<td>99969</td>
<td>163377</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2801</td>
<td>974749</td>
<td>1317419</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4681</td>
<td>2424759</td>
<td>2817958</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>7381</td>
<td>5432417</td>
<td>5710815</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>364</td>
<td>29849</td>
<td>59025</td>
</tr>
<tr>
<td>4</td>
<td>1365</td>
<td>352171</td>
<td>781270</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3906</td>
<td>2452969</td>
<td>5042345</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.7: A list of the conjectured improved bounds $N$ for $\text{CAN}(t, \frac{q^t-1}{q-1}, q)$ with original bound $N_0$ implied by Conjecture 4.4.3 for $4 \leq t \leq 6$, and all prime powers $q$ with $2 \leq q \leq 25$ such that $\frac{q^t-1}{q-1} \leq 10000$.

**Open Question 4.4.1.** Prove Conjecture 4.4.3 or find a counterexample.
4.5 New strength-3 covering array bounds

In Table 4.8, we provide the new strength-3 bounds that result from the covering arrays derived from LFSR sequences and the results that arise from recursive constructions using these arrays as an ingredient. These tables were confirmed by Colbourn, who maintains a repository of the best known covering array bounds at [25]. We note that there were two entries he supplied that we were not able to confirm:

- From the CA(2661; 3, 133, 11), we can perform two fusion operations to obtain a CA(2657; 3, 133, 9). Colbourn improves this result further by performing an unknown postoptimization step that allows a further reduction of six rows, giving a CA(2651; 3, 133, 9). As we are not aware of the postoptimization step, we list the original array in the table for \( g = 9 \).

- Colbourn employs a recursive construction using perfect hash families with our CA(249; 3, 31, 5) array as an ingredient to generate a CA(892; 3, 846, 5); we have not been able to determine the exact nature of this construction. Perfect hash family constructions for covering arrays can be found in [31].

The \textit{PrevN} column gives the previously best known result for \( g \) and \( k \), and the \textit{Method} column provides the details of the construction. The meanings of the entries in this column are detailed in Table 4.3.

We note that we have focused on the impact of our arrays on strength-3 constructions: we have not investigated the impact of our arrays as ingredients in higher strength recursive constructions.

<table>
<thead>
<tr>
<th>( g )</th>
<th>( k )</th>
<th>( N )</th>
<th>\textit{PrevN}</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>21</td>
<td>127</td>
<td>152</td>
<td>RMS(4)</td>
</tr>
<tr>
<td>4</td>
<td>42</td>
<td>211</td>
<td>212</td>
<td>CKD(( k=21, N2=28 ))</td>
</tr>
<tr>
<td>4</td>
<td>84</td>
<td>259</td>
<td>284</td>
<td>CMTW(( k=21, N2=28 ))</td>
</tr>
<tr>
<td>4</td>
<td>336</td>
<td>395</td>
<td>420</td>
<td>CCL(( l=16, k=21, N1=124, N2=127, R=6, D=24 ))</td>
</tr>
<tr>
<td>4</td>
<td>441</td>
<td>422</td>
<td>438</td>
<td>CCL(( l=21, k=21, N1=127, N2=127, R=7, D=24 ))</td>
</tr>
<tr>
<td>4</td>
<td>609</td>
<td>468</td>
<td>492</td>
<td>CCL(( l=21, k=29, N1=127, N2=152, R=7, D=27 ))</td>
</tr>
<tr>
<td>4</td>
<td>672</td>
<td>471</td>
<td>493</td>
<td>CCL(( l=32, k=21, N1=152, N2=127, R=8, D=24 ))</td>
</tr>
<tr>
<td>4</td>
<td>798</td>
<td>475</td>
<td>500</td>
<td>CCL(( l=21, k=38, N1=127, N2=152, R=7, D=28 ))</td>
</tr>
<tr>
<td>5</td>
<td>31</td>
<td>249</td>
<td>365</td>
<td>RMS(5)</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>349</td>
<td>365</td>
<td>RMS(5)A</td>
</tr>
<tr>
<td>5</td>
<td>62</td>
<td>429</td>
<td>485</td>
<td>CKD(( k=31, N2=45 ))</td>
</tr>
<tr>
<td>$g$</td>
<td>$k$</td>
<td>$N$</td>
<td>Prev$N$</td>
<td>Method</td>
</tr>
<tr>
<td>-----</td>
<td>------</td>
<td>------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>5</td>
<td>155</td>
<td>529</td>
<td>605</td>
<td>CMTW($k=31,N2=45$)</td>
</tr>
<tr>
<td>5</td>
<td>156</td>
<td>590</td>
<td>605</td>
<td>CCL($l=26,k=6,N1=249,N2=125,R=9,D=24$)</td>
</tr>
<tr>
<td>5</td>
<td>186</td>
<td>614</td>
<td>657</td>
<td>CCL($l=31,k=6,N1=249,N2=125,R=10,D=24$)</td>
</tr>
<tr>
<td>5</td>
<td>250</td>
<td>690</td>
<td>729</td>
<td>CCL($l=25,k=10,N1=249,N2=185,R=8,D=32$)</td>
</tr>
<tr>
<td>5</td>
<td>300</td>
<td>714</td>
<td>762</td>
<td>CCL($l=10,k=30,N1=185,N2=249,R=7,D=43$)</td>
</tr>
<tr>
<td>5</td>
<td>310</td>
<td>735</td>
<td>762</td>
<td>CCL($l=10,k=31,N1=185,N2=249,R=7,D=43$)</td>
</tr>
<tr>
<td>5</td>
<td>360</td>
<td>754</td>
<td>770</td>
<td>CCL($l=12,k=30,N1=225,N2=249,R=7,D=40$)</td>
</tr>
<tr>
<td>5</td>
<td>504</td>
<td>818</td>
<td>954</td>
<td>CCL($l=25,k=30,N1=249,N2=185,R=8,D=40$)</td>
</tr>
<tr>
<td>5</td>
<td>575</td>
<td>842</td>
<td>954</td>
<td>CCL($l=25,k=31,N1=249,N2=185,R=8,D=43$)</td>
</tr>
<tr>
<td>5</td>
<td>780</td>
<td>858</td>
<td>954</td>
<td>CCL($l=26,k=30,N1=249,N2=249,R=9,D=40$)</td>
</tr>
<tr>
<td>5</td>
<td>806</td>
<td>885</td>
<td>954</td>
<td>CCL($l=26,k=31,N1=249,N2=249,R=9,D=43$)</td>
</tr>
<tr>
<td>5</td>
<td>930</td>
<td>988</td>
<td>978</td>
<td>CCL($l=31,k=30,N1=249,N2=249,R=10,D=40$)</td>
</tr>
<tr>
<td>5</td>
<td>961</td>
<td>928</td>
<td>986</td>
<td>CCL($l=31,k=31,N1=249,N2=249,R=10,D=43$)</td>
</tr>
<tr>
<td>5</td>
<td>1000</td>
<td>982</td>
<td>986</td>
<td>CCL($l=25,k=40,N1=249,N2=365,R=8,D=46$)</td>
</tr>
<tr>
<td>5</td>
<td>1050</td>
<td>990</td>
<td>1002</td>
<td>CCL($l=25,k=42,N1=249,N2=365,R=8,D=47$)</td>
</tr>
<tr>
<td>5</td>
<td>1075</td>
<td>998</td>
<td>1002</td>
<td>CCL($l=25,k=43,N1=249,N2=365,R=8,D=48$)</td>
</tr>
<tr>
<td>5</td>
<td>1125</td>
<td>1006</td>
<td>1010</td>
<td>CCL($l=25,k=45,N1=249,N2=365,R=8,D=49$)</td>
</tr>
<tr>
<td>5</td>
<td>1175</td>
<td>1014</td>
<td>1045</td>
<td>CCL($l=25,k=47,N1=249,N2=365,R=8,D=50$)</td>
</tr>
<tr>
<td>5</td>
<td>1200</td>
<td>1022</td>
<td>1053</td>
<td>CCL($l=25,k=48,N1=249,N2=365,R=8,D=51$)</td>
</tr>
<tr>
<td>5</td>
<td>1440</td>
<td>1054</td>
<td>1126</td>
<td>CCL($l=48,k=30,N1=365,N2=249,R=11,D=40$)</td>
</tr>
<tr>
<td>5</td>
<td>1488</td>
<td>1087</td>
<td>1150</td>
<td>CCL($l=48,k=31,N1=365,N2=249,R=11,D=43$)</td>
</tr>
<tr>
<td>5</td>
<td>1550</td>
<td>1102</td>
<td>1150</td>
<td>CCL($l=25,k=62,N1=249,N2=429,R=8,D=53$)</td>
</tr>
<tr>
<td>5</td>
<td>1800</td>
<td>1118</td>
<td>1150</td>
<td>CCL($l=60,k=30,N1=429,N2=249,R=11,D=40$)</td>
</tr>
<tr>
<td>5</td>
<td>3875</td>
<td>1258</td>
<td>1265</td>
<td>CCL($l=25,k=155,N1=249,N2=529,R=8,D=60$)</td>
</tr>
<tr>
<td>7</td>
<td>57</td>
<td>685</td>
<td>1015</td>
<td>RMS(7)</td>
</tr>
<tr>
<td>7</td>
<td>58</td>
<td>979</td>
<td>1015</td>
<td>RMS(7)A</td>
</tr>
<tr>
<td>7</td>
<td>114</td>
<td>1231</td>
<td>1351</td>
<td>CKD($k=57,N2=91$)</td>
</tr>
<tr>
<td>7</td>
<td>399</td>
<td>1525</td>
<td>1855</td>
<td>CMTW($k=57,N2=91$)</td>
</tr>
<tr>
<td>7</td>
<td>456</td>
<td>1700</td>
<td>1915</td>
<td>CCL($l=57,k=8,N1=685,N2=343,R=14,D=48$)</td>
</tr>
<tr>
<td>7</td>
<td>504</td>
<td>1829</td>
<td>1921</td>
<td>CCL($l=9,k=56,N1=472,N2=685,R=8,D=84$)</td>
</tr>
<tr>
<td>7</td>
<td>513</td>
<td>1877</td>
<td>1927</td>
<td>CCL($l=9,k=57,N1=472,N2=685,R=8,D=90$)</td>
</tr>
<tr>
<td>7</td>
<td>560</td>
<td>1920</td>
<td>1939</td>
<td>CCL($l=10,k=56,N1=479,N2=685,R=9,D=84$)</td>
</tr>
<tr>
<td>7</td>
<td>570</td>
<td>1948</td>
<td>2126</td>
<td>CCL($l=57,k=10,N1=685,N2=479,R=14,D=56$)</td>
</tr>
<tr>
<td>7</td>
<td>588</td>
<td>2102</td>
<td>2126</td>
<td>CCL($l=49,k=12,N1=685,N2=637,R=12,D=65$)</td>
</tr>
<tr>
<td>7</td>
<td>686</td>
<td>2186</td>
<td>2263</td>
<td>CCL($l=49,k=14,N1=685,N2=637,R=12,D=72$)</td>
</tr>
<tr>
<td>7</td>
<td>700</td>
<td>2203</td>
<td>2263</td>
<td>CMTW($k=100,N2=113$)</td>
</tr>
<tr>
<td>7</td>
<td>707</td>
<td>2215</td>
<td>2263</td>
<td>CMTW($k=101,N2=115$)</td>
</tr>
<tr>
<td>7</td>
<td>721</td>
<td>2221</td>
<td>2311</td>
<td>CMTW($k=103,N2=116$)</td>
</tr>
<tr>
<td>7</td>
<td>728</td>
<td>2227</td>
<td>2311</td>
<td>CMTW($k=104,N2=117$)</td>
</tr>
<tr>
<td>7</td>
<td>756</td>
<td>2233</td>
<td>2353</td>
<td>CMTW($k=108,N2=118$)</td>
</tr>
</tbody>
</table>
## Covering Arrays from Linear Feedback Shift Registers

<table>
<thead>
<tr>
<th>$g$</th>
<th>$k$</th>
<th>$N$</th>
<th>Prev $N$</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>763</td>
<td>2239</td>
<td>2354</td>
<td>CMTW($k=109, N_2=119$)</td>
</tr>
<tr>
<td>7</td>
<td>777</td>
<td>2245</td>
<td>2354</td>
<td>CMTW($k=111, N_2=120$)</td>
</tr>
<tr>
<td>7</td>
<td>784</td>
<td>2246</td>
<td>2354</td>
<td>CCL($l=14, k=56, N_1=637, N_2=685, R=11, D=84$)</td>
</tr>
<tr>
<td>7</td>
<td>798</td>
<td>2251</td>
<td>2354</td>
<td>CMTW($k=114, N_2=121$)</td>
</tr>
<tr>
<td>7</td>
<td>1274</td>
<td>2360</td>
<td>2460</td>
<td>CCL($l=49, k=26, N_1=685, N_2=679, R=12, D=83$)</td>
</tr>
<tr>
<td>7</td>
<td>1792</td>
<td>2372</td>
<td>2618</td>
<td>CCL($l=32, k=56, N_1=679, N_2=685, R=12, D=84$)</td>
</tr>
<tr>
<td>7</td>
<td>2744</td>
<td>2378</td>
<td>2863</td>
<td>CCL($l=49, k=56, N_1=685, N_2=685, R=12, D=84$)</td>
</tr>
<tr>
<td>7</td>
<td>2793</td>
<td>2450</td>
<td>2947</td>
<td>CCL($l=49, k=57, N_1=685, N_2=685, R=12, D=90$)</td>
</tr>
<tr>
<td>7</td>
<td>3192</td>
<td>2546</td>
<td>3007</td>
<td>CCL($l=49, k=56, N_1=685, N_2=685, R=14, D=84$)</td>
</tr>
<tr>
<td>7</td>
<td>3249</td>
<td>2630</td>
<td>3007</td>
<td>CCL($l=49, k=57, N_1=685, N_2=685, R=14, D=90$)</td>
</tr>
<tr>
<td>7</td>
<td>3528</td>
<td>2852</td>
<td>3043</td>
<td>CCL($l=49, k=72, N_1=685, N_2=1015, R=12, D=96$)</td>
</tr>
<tr>
<td>7</td>
<td>3871</td>
<td>2864</td>
<td>3079</td>
<td>CCL($l=49, k=79, N_1=685, N_2=1015, R=12, D=97$)</td>
</tr>
<tr>
<td>7</td>
<td>3920</td>
<td>2876</td>
<td>3091</td>
<td>CCL($l=57, k=80, N_1=685, N_2=1015, R=12, D=98$)</td>
</tr>
<tr>
<td>7</td>
<td>3969</td>
<td>2936</td>
<td>3109</td>
<td>CCL($l=49, k=81, N_1=685, N_2=1015, R=12, D=103$)</td>
</tr>
<tr>
<td>7</td>
<td>4536</td>
<td>3044</td>
<td>3290</td>
<td>CCL($l=81, k=56, N_1=1016, N_2=1023, R=12, D=96$)</td>
</tr>
<tr>
<td>7</td>
<td>4560</td>
<td>3072</td>
<td>3388</td>
<td>CCL($l=49, k=80, N_1=685, N_2=1015, R=14, D=98$)</td>
</tr>
<tr>
<td>7</td>
<td>4617</td>
<td>3140</td>
<td>3388</td>
<td>CCL($l=81, k=57, N_1=1016, N_2=685, R=16, D=90$)</td>
</tr>
<tr>
<td>7</td>
<td>4900</td>
<td>3188</td>
<td>3388</td>
<td>CCL($l=49, k=100, N_1=685, N_2=1231, R=12, D=106$)</td>
</tr>
<tr>
<td>7</td>
<td>5047</td>
<td>3248</td>
<td>3388</td>
<td>CCL($l=49, k=103, N_1=685, N_2=1231, R=12, D=111$)</td>
</tr>
<tr>
<td>7</td>
<td>5341</td>
<td>3272</td>
<td>3388</td>
<td>CCL($l=49, k=109, N_1=685, N_2=1231, R=12, D=113$)</td>
</tr>
<tr>
<td>7</td>
<td>5390</td>
<td>3284</td>
<td>3385</td>
<td>CCL($l=49, k=110, N_1=685, N_2=1231, R=12, D=114$)</td>
</tr>
<tr>
<td>7</td>
<td>5586</td>
<td>3296</td>
<td>3388</td>
<td>CCL($l=49, k=114, N_1=685, N_2=1231, R=12, D=115$)</td>
</tr>
<tr>
<td>7</td>
<td>5700</td>
<td>3400</td>
<td>3388</td>
<td>CCL($l=49, k=100, N_1=685, N_2=1231, R=14, D=106$)</td>
</tr>
<tr>
<td>7</td>
<td>5888</td>
<td>3428</td>
<td>3388</td>
<td>CCL($l=49, k=103, N_1=685, N_2=1231, R=14, D=111$)</td>
</tr>
<tr>
<td>7</td>
<td>5920</td>
<td>3448</td>
<td>3388</td>
<td>CCL($l=49, k=109, N_1=685, N_2=1231, R=14, D=113$)</td>
</tr>
<tr>
<td>7</td>
<td>6384</td>
<td>3488</td>
<td>3656</td>
<td>CCL($l=49, k=142, N_1=685, N_2=1351, R=12, D=121$)</td>
</tr>
<tr>
<td>7</td>
<td>7350</td>
<td>3500</td>
<td>3727</td>
<td>CCL($l=49, k=150, N_1=685, N_2=1351, R=12, D=122$)</td>
</tr>
<tr>
<td>7</td>
<td>10000</td>
<td>3583</td>
<td>3829</td>
<td>CCL($l=197, k=57, N_1=1386, N_2=685, R=18, D=84$)</td>
</tr>
<tr>
<td>8</td>
<td>73</td>
<td>1023</td>
<td>1492</td>
<td>RMS(8)</td>
</tr>
<tr>
<td>8</td>
<td>91</td>
<td>1455</td>
<td>1520</td>
<td>RMS(9)F</td>
</tr>
<tr>
<td>8</td>
<td>584</td>
<td>2311</td>
<td>2780</td>
<td>CMTW($k=73, N_2=120$)</td>
</tr>
<tr>
<td>8</td>
<td>640</td>
<td>2501</td>
<td>2780</td>
<td>CCL($l=64, k=10, N_1=1023, N_2=512, R=14, D=69$)</td>
</tr>
<tr>
<td>8</td>
<td>730</td>
<td>2543</td>
<td>2938</td>
<td>CCL($l=10, k=73, N_1=512, N_2=1023, R=9, D=112$)</td>
</tr>
<tr>
<td>8</td>
<td>2920</td>
<td>3067</td>
<td>4076</td>
<td>CCL($l=40, k=73, N_1=1016, N_2=1023, R=14, D=112$)</td>
</tr>
<tr>
<td>8</td>
<td>4672</td>
<td>3614</td>
<td>4460</td>
<td>CCL($l=64, k=73, N_1=1023, N_2=1023, R=14, D=112$)</td>
</tr>
<tr>
<td>8</td>
<td>5329</td>
<td>3838</td>
<td>4572</td>
<td>CCL($l=73, k=73, N_1=1023, N_2=1023, R=16, D=112$)</td>
</tr>
<tr>
<td>8</td>
<td>5696</td>
<td>4223</td>
<td>4572</td>
<td>CMTW($k=712, N_2=176$)</td>
</tr>
<tr>
<td>8</td>
<td>5824</td>
<td>4242</td>
<td>4642</td>
<td>CCL($l=64, k=91, N_1=1023, N_2=1455, R=14, D=126$)</td>
</tr>
<tr>
<td>8</td>
<td>5840</td>
<td>4270</td>
<td>4674</td>
<td>CCL($l=80, k=73, N_1=1455, N_2=1023, R=16, D=112$)</td>
</tr>
<tr>
<td>8</td>
<td>6272</td>
<td>4339</td>
<td>4792</td>
<td>CCL($l=64, k=98, N_1=1023, N_2=1552, R=14, D=126$)</td>
</tr>
<tr>
<td>8</td>
<td>6336</td>
<td>4367</td>
<td>4799</td>
<td>CCL($l=64, k=99, N_1=1023, N_2=1552, R=14, D=128$)</td>
</tr>
<tr>
<td>8</td>
<td>6424</td>
<td>4382</td>
<td>4758</td>
<td>CCL($l=88, k=73, N_1=1455, N_2=1023, R=17, D=112$)</td>
</tr>
<tr>
<td>$g$</td>
<td>$k$</td>
<td>$N$</td>
<td>Prev $N$</td>
<td>Method</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>8</td>
<td>665</td>
<td>4423</td>
<td>4758</td>
<td>CCL(l=64,k=104,N1=1023,N2=1552,R=14,D=132)</td>
</tr>
<tr>
<td>8</td>
<td>6912</td>
<td>4465</td>
<td>4758</td>
<td>CCL(l=64,k=108,N1=1023,N2=1552,R=14,D=135)</td>
</tr>
<tr>
<td>8</td>
<td>7104</td>
<td>4479</td>
<td>4758</td>
<td>CCL(l=64,k=111,N1=1023,N2=1552,R=14,D=136)</td>
</tr>
<tr>
<td>8</td>
<td>7232</td>
<td>4521</td>
<td>4800</td>
<td>CCL(l=64,k=113,N1=1023,N2=1552,R=14,D=139)</td>
</tr>
<tr>
<td>8</td>
<td>7360</td>
<td>4549</td>
<td>4821</td>
<td>CCL(l=64,k=115,N1=1023,N2=1552,R=14,D=142)</td>
</tr>
<tr>
<td>8</td>
<td>8192</td>
<td>4591</td>
<td>4905</td>
<td>CCL(l=64,k=128,N1=1023,N2=1552,R=14,D=144)</td>
</tr>
<tr>
<td>8</td>
<td>9152</td>
<td>4647</td>
<td>4989</td>
<td>CCL(l=64,k=143,N1=1023,N2=1552,R=14,D=148)</td>
</tr>
<tr>
<td>8</td>
<td>9216</td>
<td>4703</td>
<td>5024</td>
<td>CCL(l=64,k=144,N1=1023,N2=1552,R=14,D=152)</td>
</tr>
<tr>
<td>8</td>
<td>9728</td>
<td>4745</td>
<td>5059</td>
<td>CCL(l=64,k=152,N1=1023,N2=1552,R=14,D=155)</td>
</tr>
<tr>
<td>8</td>
<td>10000</td>
<td>4759</td>
<td>5059</td>
<td>CCL(l=63,k=159,N1=1023,N2=1552,R=14,D=156)</td>
</tr>
<tr>
<td>9</td>
<td>91</td>
<td>1457</td>
<td>2169</td>
<td>RMS(9)</td>
</tr>
<tr>
<td>9</td>
<td>92</td>
<td>2105</td>
<td>2169</td>
<td>RMS(9)A</td>
</tr>
<tr>
<td>9</td>
<td>133</td>
<td>2657*</td>
<td>2889</td>
<td>RMS(11)FF</td>
</tr>
<tr>
<td>9</td>
<td>182</td>
<td>2681</td>
<td>2889</td>
<td>CKD(k=91,N2=153)</td>
</tr>
<tr>
<td>9</td>
<td>819</td>
<td>3329</td>
<td>4041</td>
<td>CMTW(k=91,N2=153)</td>
</tr>
<tr>
<td>9</td>
<td>910</td>
<td>3786</td>
<td>4177</td>
<td>CCL(l=91,k=10,N1=1457,N2=729,R=20,D=80)</td>
</tr>
<tr>
<td>9</td>
<td>1092</td>
<td>4381</td>
<td>4498</td>
<td>CCL(l=12,k=91,N1=1196,N2=1457,R=12,D=144)</td>
</tr>
<tr>
<td>9</td>
<td>1134</td>
<td>4482</td>
<td>4953</td>
<td>CCL(l=81,k=14,N1=1457,N2=1377,R=16,D=103)</td>
</tr>
<tr>
<td>9</td>
<td>1183</td>
<td>4562</td>
<td>5009</td>
<td>CCL(l=13,k=91,N1=1377,N2=1457,R=12,D=144)</td>
</tr>
<tr>
<td>9</td>
<td>1215</td>
<td>4658</td>
<td>5099</td>
<td>CCL(l=81,k=15,N1=1457,N2=1377,R=16,D=114)</td>
</tr>
<tr>
<td>9</td>
<td>1296</td>
<td>4690</td>
<td>5057</td>
<td>CCL(l=81,k=16,N1=1457,N2=1377,R=16,D=116)</td>
</tr>
<tr>
<td>9</td>
<td>1377</td>
<td>4818</td>
<td>5057</td>
<td>CCL(l=81,k=17,N1=1457,N2=1377,R=16,D=124)</td>
</tr>
<tr>
<td>9</td>
<td>1404</td>
<td>4849</td>
<td>5057</td>
<td>CMTW(k=156,N2=190)</td>
</tr>
<tr>
<td>9</td>
<td>1539</td>
<td>4866</td>
<td>5145</td>
<td>CCL(l=81,k=19,N1=1457,N2=1377,R=16,D=127)</td>
</tr>
<tr>
<td>9</td>
<td>1620</td>
<td>4914</td>
<td>5145</td>
<td>CCL(l=81,k=20,N1=1457,N2=1377,R=16,D=130)</td>
</tr>
<tr>
<td>9</td>
<td>1638</td>
<td>4937</td>
<td>5145</td>
<td>CMTW(k=182,N2=201)</td>
</tr>
<tr>
<td>9</td>
<td>1701</td>
<td>5050</td>
<td>5169</td>
<td>CCL(l=81,k=21,N1=1457,N2=1449,R=16,D=134)</td>
</tr>
<tr>
<td>9</td>
<td>2025</td>
<td>5066</td>
<td>5249</td>
<td>CCL(l=81,k=25,N1=1457,N2=1449,R=16,D=135)</td>
</tr>
<tr>
<td>9</td>
<td>2106</td>
<td>5082</td>
<td>5538</td>
<td>CCL(l=81,k=26,N1=1457,N2=1449,R=16,D=136)</td>
</tr>
<tr>
<td>9</td>
<td>2187</td>
<td>5114</td>
<td>5538</td>
<td>CCL(l=81,k=27,N1=1457,N2=1449,R=16,D=138)</td>
</tr>
<tr>
<td>9</td>
<td>2430</td>
<td>5178</td>
<td>5673</td>
<td>CCL(l=81,k=30,N1=1457,N2=1449,R=16,D=142)</td>
</tr>
<tr>
<td>9</td>
<td>3731</td>
<td>5210</td>
<td>5922</td>
<td>CCL(l=41,k=91,N1=1457,N2=1449,R=16,D=144)</td>
</tr>
<tr>
<td>9</td>
<td>7371</td>
<td>5218</td>
<td>6489</td>
<td>CCL(l=81,k=91,N1=1457,N2=1457,R=16,D=144)</td>
</tr>
<tr>
<td>9</td>
<td>8281</td>
<td>5794</td>
<td>6625</td>
<td>CCL(l=91,k=91,N1=1457,N2=1457,R=20,D=144)</td>
</tr>
<tr>
<td>9</td>
<td>8748</td>
<td>6202</td>
<td>6625</td>
<td>CCL(l=81,k=108,N1=1457,N2=2169,R=16,D=161)</td>
</tr>
<tr>
<td>9</td>
<td>9153</td>
<td>6314</td>
<td>6817</td>
<td>CCL(l=81,k=113,N1=1457,N2=2169,R=16,D=168)</td>
</tr>
<tr>
<td>9</td>
<td>10000</td>
<td>6506</td>
<td>7074</td>
<td>CCL(l=113,k=89,N1=2169,N2=1457,R=20,D=144)</td>
</tr>
<tr>
<td>10</td>
<td>133</td>
<td>2659</td>
<td>3794</td>
<td>RMS(11)F</td>
</tr>
<tr>
<td>10</td>
<td>134</td>
<td>3559</td>
<td>3794</td>
<td>RMS(11)FA</td>
</tr>
<tr>
<td>10</td>
<td>665</td>
<td>5733</td>
<td>6775</td>
<td>CCL(l=133,k=5,N1=2659,N2=1050,R=22,D=92)</td>
</tr>
<tr>
<td>$g$</td>
<td>$k$</td>
<td>$N$</td>
<td>Prev$N$</td>
<td>Method</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
<td>--------</td>
<td>---------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>10</td>
<td>798</td>
<td>5849</td>
<td>6874</td>
<td>CCL($l=133,k=6,N1=2659,N2=1100,R=22,D=95$)</td>
</tr>
<tr>
<td>10</td>
<td>1320</td>
<td>6074</td>
<td>7138</td>
<td>CCL($l=10,k=132,N1=1215,N2=2659,h=11,R=10,D=220$)</td>
</tr>
<tr>
<td>10</td>
<td>1452</td>
<td>6152</td>
<td>7216</td>
<td>CCL($l=11,k=132,N1=1293,N2=2659,h=11,R=10,D=220$)</td>
</tr>
<tr>
<td>10</td>
<td>1463</td>
<td>6179</td>
<td>7303</td>
<td>CMTW($k=133,N2=231$)F</td>
</tr>
<tr>
<td>10</td>
<td>1572</td>
<td>6239</td>
<td>7385</td>
<td>CCL($l=10,k=132,N1=1215,N2=2659,h=11,R=10,D=220$)</td>
</tr>
<tr>
<td>10</td>
<td>1596</td>
<td>6250</td>
<td>7491</td>
<td>RMS(11)A</td>
</tr>
<tr>
<td>10</td>
<td>1729</td>
<td>6944</td>
<td>7571</td>
<td>CCL($l=133,k=13,N1=2659,N2=1865,R=22,D=110$)</td>
</tr>
<tr>
<td>10</td>
<td>1834</td>
<td>7245</td>
<td>7701</td>
<td>CCL($l=14,k=131,N1=1895,N2=2659,R=13,D=207$)</td>
</tr>
<tr>
<td>9</td>
<td>1572</td>
<td>6239</td>
<td>7385</td>
<td>CCL($l=10,k=132,N1=1215,N2=2659,h=11,R=10,D=220$)</td>
</tr>
<tr>
<td>9</td>
<td>1862</td>
<td>7258</td>
<td>7989</td>
<td>CCL($l=14,k=131,N1=1895,N2=2659,R=13,D=208$)</td>
</tr>
<tr>
<td>9</td>
<td>1965</td>
<td>7807</td>
<td>8077</td>
<td>CCL($l=15,k=131,N1=2250,N2=2659,R=14,D=207$)</td>
</tr>
<tr>
<td>9</td>
<td>1995</td>
<td>7821</td>
<td>8077</td>
<td>CCL($l=15,k=133,N1=2250,N2=2659,R=14,D=208$)</td>
</tr>
<tr>
<td>9</td>
<td>2128</td>
<td>7967</td>
<td>8176</td>
<td>CCL($l=133,k=16,N1=2659,N2=2250,R=22,D=139$)</td>
</tr>
<tr>
<td>9</td>
<td>2261</td>
<td>7989</td>
<td>8275</td>
<td>CCL($l=133,k=17,N1=2659,N2=2250,R=22,D=140$)</td>
</tr>
<tr>
<td>9</td>
<td>2394</td>
<td>8165</td>
<td>8319</td>
<td>CCL($l=133,k=18,N1=2659,N2=2250,R=22,D=148$)</td>
</tr>
<tr>
<td>9</td>
<td>2660</td>
<td>8284</td>
<td>8418</td>
<td>CCL($l=133,k=20,N1=2659,N2=2250,R=22,D=153$)</td>
</tr>
<tr>
<td>9</td>
<td>5187</td>
<td>9447</td>
<td>9615</td>
<td>CCL($l=133,k=39,N1=2659,N2=2630,R=22,D=189$)</td>
</tr>
<tr>
<td>9</td>
<td>6517</td>
<td>9469</td>
<td>10203</td>
<td>CCL($l=133,k=49,N1=2659,N2=2630,R=22,D=190$)</td>
</tr>
<tr>
<td>9</td>
<td>6650</td>
<td>9513</td>
<td>10203</td>
<td>CCL($l=133,k=50,N1=2659,N2=2630,R=22,D=192$)</td>
</tr>
<tr>
<td>9</td>
<td>8645</td>
<td>9542</td>
<td>11368</td>
<td>CCL($l=133,k=65,N1=2659,N2=2659,R=22,D=192$)</td>
</tr>
<tr>
<td>9</td>
<td>8911</td>
<td>9564</td>
<td>11368</td>
<td>CCL($l=133,k=67,N1=2659,N2=2659,R=22,D=193$)</td>
</tr>
<tr>
<td>9</td>
<td>9310</td>
<td>9586</td>
<td>11368</td>
<td>CCL($l=133,k=70,N1=2659,N2=2659,R=22,D=194$)</td>
</tr>
<tr>
<td>9</td>
<td>9576</td>
<td>9608</td>
<td>11400</td>
<td>CCL($l=133,k=72,N1=2659,N2=2659,R=22,D=195$)</td>
</tr>
<tr>
<td>9</td>
<td>9842</td>
<td>9652</td>
<td>11433</td>
<td>CCL($l=133,k=74,N1=2659,N2=2659,R=22,D=197$)</td>
</tr>
<tr>
<td>9</td>
<td>10000</td>
<td>9696</td>
<td>11455</td>
<td>CCL($l=132,k=76,N1=2659,N2=2659,R=22,D=199$)</td>
</tr>
</tbody>
</table>

<p>| 10  | 133   | 2661   | 3971    | RMS(11)F                                    |
| 10  | 134   | 3871   | 3971    | RMS(11)A                                    |
| 10  | 1463  | 6181   | 7491    | CMTW($k=133,N2=231$)                        |
| 11  | 1596  | 6872   | 7701    | CCL($l=133,k=12,N1=2661,N2=1331,R=24,D=120$) |
| 11  | 1694  | 7669   | 8108    | CCL($l=121,k=14,N1=2661,N2=2088,R=20,D=146$) |
| 11  | 1716  | 7820   | 8182    | CCL($l=13,k=132,N1=2079,N2=2661,R=14,D=220$) |
| 11  | 1848  | 7829   | 8208    | CCL($l=14,k=132,N1=2088,N2=2661,R=14,D=220$) |
| 11  | 1862  | 7969   | 8268    | CCL($l=14,k=133,N1=2088,N2=2661,R=14,D=230$) |
| 11  | 1936  | 8222   | 8278    | CCL($l=121,k=16,N1=2661,N2=2541,R=20,D=151$) |
| 11  | 1980  | 8282   | 8288    | CCL($l=15,k=132,N1=2541,N2=2661,R=14,D=220$) |
| 11  | 6600  | 9712   | 10802   | CCL($l=50,k=132,N1=2661,N2=2661,R=20,D=220$) |
| 11  | 10000 | 9722   | 11891   | CCL($l=76,k=132,N1=2661,N2=2661,R=20,D=220$) |
| 12  | 183   | 4391   | 6350    | RMS(13)F                                    |
| 12  | 184   | 5975   | 6350    | RMS(13)FA                                   |
| 12  | 912   | 9375   | 11180   | CCL($l=6,k=152,N1=1728,N2=4391,R=11,D=296$) |
| 12  | 1086  | 9386   | 11225   | CCL($l=6,k=181,N1=1728,N2=4391,R=11,D=297$) |
| 12  | 1098  | 9397   | 11356   | CCL($l=6,k=183,N1=1728,N2=4391,R=11,D=298$) |</p>
<table>
<thead>
<tr>
<th>$g$</th>
<th>$k$</th>
<th>$N$</th>
<th>$\text{Prev}N$</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>1281</td>
<td>9947</td>
<td>11754</td>
<td>CCL($l=183, k=7, N_1=4391, N_2=2046, R=26, D=135$)</td>
</tr>
<tr>
<td>12</td>
<td>2013</td>
<td>10181</td>
<td>12107</td>
<td>CCL($l=11, k=183, N_1=2046, N_2=4391, h=13, R=12, D=312$)</td>
</tr>
<tr>
<td>12</td>
<td>2196</td>
<td>10294</td>
<td>12114</td>
<td>CCL($l=12, k=183, N_1=2159, N_2=4391, h=13, R=12, D=312$)</td>
</tr>
<tr>
<td>12</td>
<td>2379</td>
<td>10301</td>
<td>12260</td>
<td>CCL($l=13, k=183, N_1=2166, N_2=4391, h=13, R=12, D=312$)</td>
</tr>
<tr>
<td>12</td>
<td>2534</td>
<td>10426</td>
<td>12265</td>
<td>CCL($l=14, k=183, N_1=2174, N_2=4391, R=13, D=297$)</td>
</tr>
<tr>
<td>12</td>
<td>2562</td>
<td>10439</td>
<td>12398</td>
<td>CCL($l=14, k=183, N_1=2174, N_2=4391, R=13, D=298$)</td>
</tr>
<tr>
<td>12</td>
<td>2745</td>
<td>11970</td>
<td>12493</td>
<td>CCL($l=183, k=15, N_1=4391, N_2=3523, R=26, D=156$)</td>
</tr>
<tr>
<td>12</td>
<td>2928</td>
<td>12536</td>
<td>13298</td>
<td>CCL($l=183, k=16, N_1=4391, N_2=3569, R=26, D=176$)</td>
</tr>
<tr>
<td>12</td>
<td>3111</td>
<td>12880</td>
<td>13440</td>
<td>CCL($l=183, k=17, N_1=4391, N_2=3627, R=26, D=187$)</td>
</tr>
<tr>
<td>12</td>
<td>3258</td>
<td>13113</td>
<td>13440</td>
<td>CCL($l=18, k=181, N_1=3673, N_2=4391, R=17, D=297$)</td>
</tr>
<tr>
<td>12</td>
<td>3294</td>
<td>13130</td>
<td>13440</td>
<td>CCL($l=18, k=183, N_1=3673, N_2=4391, R=17, D=298$)</td>
</tr>
<tr>
<td>12</td>
<td>3843</td>
<td>13857</td>
<td>14000</td>
<td>CCL($l=183, k=21, N_1=4391, N_2=3850, R=26, D=216$)</td>
</tr>
<tr>
<td>12</td>
<td>10000</td>
<td>15862</td>
<td>16404</td>
<td>CCL($l=80, k=125, N_1=4391, N_2=4391, R=24, D=295$)</td>
</tr>
<tr>
<td>13</td>
<td>183</td>
<td>4393</td>
<td>6565</td>
<td>RMS(13)</td>
</tr>
<tr>
<td>13</td>
<td>184</td>
<td>6421</td>
<td>6565</td>
<td>RMS(13)A</td>
</tr>
<tr>
<td>13</td>
<td>273</td>
<td>8185</td>
<td>9289</td>
<td>RMS(16)FFF</td>
</tr>
<tr>
<td>13</td>
<td>366</td>
<td>8293</td>
<td>9901</td>
<td>CKD($k=183, N_2=325$)</td>
</tr>
<tr>
<td>13</td>
<td>2379</td>
<td>10321</td>
<td>12493</td>
<td>CMTW($k=183, N_2=325$)</td>
</tr>
<tr>
<td>13</td>
<td>2562</td>
<td>11462</td>
<td>12937</td>
<td>CCL($l=183, k=14, N_1=4393, N_2=2197, R=29, D=168$)</td>
</tr>
<tr>
<td>13</td>
<td>2704</td>
<td>13142</td>
<td>13634</td>
<td>CCL($l=169, k=16, N_1=4393, N_2=3853, R=24, D=204$)</td>
</tr>
<tr>
<td>13</td>
<td>2873</td>
<td>13165</td>
<td>14880</td>
<td>CCL($l=169, k=17, N_1=4393, N_2=3876, R=24, D=204$)</td>
</tr>
<tr>
<td>13</td>
<td>3042</td>
<td>13471</td>
<td>15030</td>
<td>CCL($l=169, k=18, N_1=4393, N_2=3894, R=24, D=216$)</td>
</tr>
<tr>
<td>13</td>
<td>3111</td>
<td>13573</td>
<td>15505</td>
<td>CCL($l=17, k=183, N_1=3876, N_2=4393, R=17, D=312$)</td>
</tr>
<tr>
<td>13</td>
<td>3294</td>
<td>13903</td>
<td>15793</td>
<td>CCL($l=18, k=183, N_1=3894, N_2=4393, R=18, D=312$)</td>
</tr>
<tr>
<td>13</td>
<td>3380</td>
<td>14114</td>
<td>15793</td>
<td>CCL($l=169, k=20, N_1=4393, N_2=4225, R=24, D=229$)</td>
</tr>
<tr>
<td>13</td>
<td>3477</td>
<td>14234</td>
<td>16081</td>
<td>CCL($l=19, k=183, N_1=4225, N_2=4393, R=18, D=312$)</td>
</tr>
<tr>
<td>13</td>
<td>3660</td>
<td>14546</td>
<td>16261</td>
<td>CCL($l=20, k=183, N_1=4225, N_2=4393, R=19, D=312$)</td>
</tr>
<tr>
<td>13</td>
<td>3718</td>
<td>14594</td>
<td>16333</td>
<td>CCL($l=169, k=22, N_1=4393, N_2=4225, R=24, D=249$)</td>
</tr>
<tr>
<td>13</td>
<td>3843</td>
<td>14858</td>
<td>16477</td>
<td>CCL($l=21, k=183, N_1=4225, N_2=4393, R=20, D=312$)</td>
</tr>
<tr>
<td>13</td>
<td>4056</td>
<td>15074</td>
<td>16585</td>
<td>CCL($l=169, k=24, N_1=4393, N_2=4225, R=24, D=269$)</td>
</tr>
<tr>
<td>13</td>
<td>4225</td>
<td>15242</td>
<td>16777</td>
<td>CCL($l=169, k=25, N_1=4393, N_2=4225, R=24, D=276$)</td>
</tr>
<tr>
<td>13</td>
<td>4503</td>
<td>15290</td>
<td>16897</td>
<td>CCL($l=169, k=27, N_1=4393, N_2=4225, R=24, D=278$)</td>
</tr>
<tr>
<td>13</td>
<td>4732</td>
<td>15338</td>
<td>16921</td>
<td>CCL($l=169, k=28, N_1=4393, N_2=4225, R=24, D=280$)</td>
</tr>
<tr>
<td>13</td>
<td>4758</td>
<td>15577</td>
<td>17185</td>
<td>CMTW($k=366, N_2=438$)</td>
</tr>
<tr>
<td>13</td>
<td>4901</td>
<td>15710</td>
<td>17185</td>
<td>CCL($l=169, k=29, N_1=4393, N_2=4381, R=24, D=289$)</td>
</tr>
<tr>
<td>13</td>
<td>5070</td>
<td>15926</td>
<td>17389</td>
<td>CCL($l=169, k=30, N_1=4393, N_2=4381, R=24, D=298$)</td>
</tr>
<tr>
<td>13</td>
<td>5408</td>
<td>15950</td>
<td>17533</td>
<td>CCL($l=169, k=32, N_1=4393, N_2=4381, R=24, D=299$)</td>
</tr>
<tr>
<td>13</td>
<td>5577</td>
<td>16142</td>
<td>17641</td>
<td>CCL($l=169, k=33, N_1=4393, N_2=4381, R=24, D=307$)</td>
</tr>
<tr>
<td>13</td>
<td>10000</td>
<td>16262</td>
<td>18122</td>
<td>CCL($l=55, k=182, N_1=4381, N_2=4393, R=24, D=312$)</td>
</tr>
<tr>
<td>14</td>
<td>273</td>
<td>8187</td>
<td>11996</td>
<td>RMS(16)FF</td>
</tr>
<tr>
<td>14</td>
<td>307</td>
<td>9819</td>
<td>12248</td>
<td>RMS(17)FFF</td>
</tr>
<tr>
<td>$g$</td>
<td>$k$</td>
<td>$N$</td>
<td>Prev $N$</td>
<td>Method</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>14</td>
<td>1290</td>
<td>16736</td>
<td>18046</td>
<td>CCL($l=5,k=258,N_1=2842,N_2=8187,R=13,D=439$)</td>
</tr>
<tr>
<td>14</td>
<td>1365</td>
<td>16749</td>
<td>19222</td>
<td>CCL($l=5,k=273,N_1=2842,N_2=8187,R=13,D=440$)</td>
</tr>
<tr>
<td>14</td>
<td>1638</td>
<td>17679</td>
<td>20189</td>
<td>CCL($l=7,k=6,N_1=8187,N_2=2940,R=36,D=182$)</td>
</tr>
<tr>
<td>14</td>
<td>1680</td>
<td>18364</td>
<td>20189</td>
<td>CCL($l=8,k=210,N_1=3185,N_2=8187,R=16,D=437$)</td>
</tr>
<tr>
<td>14</td>
<td>1768</td>
<td>18380</td>
<td>20345</td>
<td>CCL($l=8,k=221,N_1=3185,N_2=8187,R=16,D=438$)</td>
</tr>
<tr>
<td>14</td>
<td>2064</td>
<td>18396</td>
<td>21083</td>
<td>CCL($l=8,k=258,N_1=3185,N_2=8187,R=16,D=439$)</td>
</tr>
<tr>
<td>14</td>
<td>2184</td>
<td>18412</td>
<td>21356</td>
<td>CCL($l=8,k=273,N_1=3185,N_2=8187,R=16,D=440$)</td>
</tr>
<tr>
<td>14</td>
<td>3003</td>
<td>18572</td>
<td>21909</td>
<td>CCL($l=11,k=273,N_1=3185,N_2=8187,R=16,h=15,D=480$)</td>
</tr>
<tr>
<td>14</td>
<td>3276</td>
<td>18845</td>
<td>22141</td>
<td>CCL($l=12,k=273,N_1=3458,N_2=8187,R=16,D=480$)</td>
</tr>
<tr>
<td>14</td>
<td>3822</td>
<td>18936</td>
<td>22745</td>
<td>CCL($l=14,k=273,N_1=3549,N_2=8187,R=16,D=480$)</td>
</tr>
<tr>
<td>14</td>
<td>4095</td>
<td>19385</td>
<td>23018</td>
<td>CCL($l=16,k=273,N_1=4011,N_2=8187,R=16,D=480$)</td>
</tr>
<tr>
<td>14</td>
<td>4590</td>
<td>19666</td>
<td>23244</td>
<td>CCL($l=17,k=270,N_1=4087,N_2=8187,R=16,D=462$)</td>
</tr>
<tr>
<td>14</td>
<td>5508</td>
<td>21883</td>
<td>24312</td>
<td>CCL($l=18,k=273,N_1=4018,N_2=8187,R=18,D=447$)</td>
</tr>
<tr>
<td>15</td>
<td>273</td>
<td>8189</td>
<td>11998</td>
<td>RMS($16$)F</td>
</tr>
<tr>
<td>15</td>
<td>307</td>
<td>9821</td>
<td>12734</td>
<td>RMS($17$)FF</td>
</tr>
<tr>
<td>15</td>
<td>381</td>
<td>13709</td>
<td>16808</td>
<td>RMS($19$)FFFF</td>
</tr>
<tr>
<td>15</td>
<td>448</td>
<td>14853</td>
<td>17448</td>
<td>CKD($k=224,N_2=476$)</td>
</tr>
<tr>
<td>15</td>
<td>544</td>
<td>14867</td>
<td>17798</td>
<td>CKD($k=272,N_2=477$)</td>
</tr>
<tr>
<td>15</td>
<td>556</td>
<td>14881</td>
<td>17974</td>
<td>CKD($k=273,N_2=478$)</td>
</tr>
<tr>
<td>15</td>
<td>580</td>
<td>16513</td>
<td>17974</td>
<td>CKD($k=278,N_2=479$)</td>
</tr>
<tr>
<td>15</td>
<td>614</td>
<td>16541</td>
<td>18446</td>
<td>CKD($k=307,N_2=480$)</td>
</tr>
<tr>
<td>15</td>
<td>670</td>
<td>17976</td>
<td>18446</td>
<td>CCL($l=5,k=134,N_1=3375,N_2=8189,R=14,D=458$)</td>
</tr>
<tr>
<td>15</td>
<td>755</td>
<td>17990</td>
<td>18446</td>
<td>CCL($l=5,k=151,N_1=3375,N_2=8189,R=14,D=459$)</td>
</tr>
<tr>
<td>15</td>
<td>920</td>
<td>18004</td>
<td>18446</td>
<td>CCL($l=5,k=184,N_1=3375,N_2=8189,R=14,D=460$)</td>
</tr>
<tr>
<td>15</td>
<td>1090</td>
<td>18018</td>
<td>18446</td>
<td>CCL($l=5,k=218,N_1=3375,N_2=8189,R=14,D=461$)</td>
</tr>
<tr>
<td>15</td>
<td>1350</td>
<td>18032</td>
<td>20715</td>
<td>CCL($l=5,k=270,N_1=3375,N_2=8189,R=14,D=462$)</td>
</tr>
<tr>
<td>15</td>
<td>1365</td>
<td>18046</td>
<td>20715</td>
<td>CCL($l=5,k=273,N_1=3375,N_2=8189,R=14,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>1638</td>
<td>18284</td>
<td>21899</td>
<td>CCL($l=7,k=6,N_1=8189,N_2=3375,R=32,D=210$)</td>
</tr>
<tr>
<td>15</td>
<td>1666</td>
<td>19236</td>
<td>21899</td>
<td>CCL($l=17,k=98,N_1=4087,N_2=8189,R=16,D=435$)</td>
</tr>
<tr>
<td>15</td>
<td>1751</td>
<td>19252</td>
<td>22365</td>
<td>CCL($l=17,k=103,N_1=4087,N_2=8189,R=16,D=436$)</td>
</tr>
<tr>
<td>15</td>
<td>1911</td>
<td>19277</td>
<td>22931</td>
<td>CCL($l=7,k=273,N_1=3888,N_2=8189,h=16,R=15,D=480$)</td>
</tr>
<tr>
<td>$g$</td>
<td>$k$</td>
<td>$N$</td>
<td>Prev$N$</td>
<td>Method</td>
</tr>
<tr>
<td>-----</td>
<td>------</td>
<td>------</td>
<td>--------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>15</td>
<td>2184</td>
<td>19360</td>
<td>23000</td>
<td>CCL($l=8,k=273,N_1=3971,N_2=8189,h=16,R=15,D=480$)</td>
</tr>
<tr>
<td>15</td>
<td>2457</td>
<td>19429</td>
<td>23015</td>
<td>CCL($l=9,k=273,N_1=4040,N_2=8189,h=16,R=15,D=480$)</td>
</tr>
<tr>
<td>15</td>
<td>2730</td>
<td>19444</td>
<td>23047</td>
<td>CCL($l=10,k=273,N_1=4055,N_2=8189,h=16,R=15,D=480$)</td>
</tr>
<tr>
<td>15</td>
<td>4368</td>
<td>19469</td>
<td>23246</td>
<td>CMTW($k=273,N_2=496$)F</td>
</tr>
<tr>
<td>15</td>
<td>4590</td>
<td>19668</td>
<td>23246</td>
<td>CCL($l=9,k=273,N_1=4040,N_2=8189,h=16,R=15,D=480$)</td>
</tr>
<tr>
<td>15</td>
<td>4641</td>
<td>19684</td>
<td>23493</td>
<td>CCL($l=10,k=273,N_1=4055,N_2=8189,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>4914</td>
<td>19956</td>
<td>23541</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=465$)</td>
</tr>
<tr>
<td>15</td>
<td>5219</td>
<td>21348</td>
<td>24261</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=465$)</td>
</tr>
<tr>
<td>15</td>
<td>5526</td>
<td>22308</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>5780</td>
<td>24938</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>5833</td>
<td>25095</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>6140</td>
<td>25748</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>6188</td>
<td>25924</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>6256</td>
<td>25956</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>6279</td>
<td>26030</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>6552</td>
<td>26094</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>6825</td>
<td>26380</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>6858</td>
<td>27016</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>7020</td>
<td>27340</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>7098</td>
<td>27365</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>7644</td>
<td>27694</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>8736</td>
<td>27996</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>9009</td>
<td>28348</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>9146</td>
<td>28650</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>9248</td>
<td>28682</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>9282</td>
<td>28696</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>9828</td>
<td>28774</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>15</td>
<td>10000</td>
<td>29690</td>
<td>25221</td>
<td>CCL($l=17,k=307,N_1=4087,N_2=9821,R=16,D=463$)</td>
</tr>
<tr>
<td>16</td>
<td>273</td>
<td>8191</td>
<td>12226</td>
<td>RMS(16)</td>
</tr>
<tr>
<td>16</td>
<td>307</td>
<td>9823</td>
<td>12799</td>
<td>RMS(17)F</td>
</tr>
<tr>
<td>16</td>
<td>381</td>
<td>13711</td>
<td>17416</td>
<td>RMS(19)FFF</td>
</tr>
<tr>
<td>16</td>
<td>546</td>
<td>15631</td>
<td>18448</td>
<td>CKD($k=273,N_2=496$)</td>
</tr>
<tr>
<td>16</td>
<td>576</td>
<td>17263</td>
<td>18448</td>
<td>CKD($k=288,N_2=496$)</td>
</tr>
<tr>
<td>16</td>
<td>578</td>
<td>17473</td>
<td>18448</td>
<td>CKD($k=289,N_2=510$)</td>
</tr>
<tr>
<td>16</td>
<td>612</td>
<td>17713</td>
<td>18448</td>
<td>CKD($k=306,N_2=526$)</td>
</tr>
<tr>
<td>16</td>
<td>614</td>
<td>17743</td>
<td>18448</td>
<td>CKD($k=307,N_2=528$)</td>
</tr>
<tr>
<td>16</td>
<td>4368</td>
<td>19471</td>
<td>23506</td>
<td>CMTW($k=273,N_2=496$)</td>
</tr>
<tr>
<td>16</td>
<td>4608</td>
<td>20417</td>
<td>23506</td>
<td>CCL($l=256,k=18,N_1=8191,N_2=4096,R=30,D=271)$</td>
</tr>
<tr>
<td>16</td>
<td>4914</td>
<td>20447</td>
<td>24482</td>
<td>CCL($l=18,k=273,N_1=4096,N_2=8191,R=17,D=480$)</td>
</tr>
<tr>
<td>16</td>
<td>5184</td>
<td>22334</td>
<td>24737</td>
<td>CCL($l=18,k=288,N_1=4096,N_2=9823,R=17,D=495$)</td>
</tr>
<tr>
<td>16</td>
<td>5202</td>
<td>22351</td>
<td>25055</td>
<td>CCL($l=18,k=289,N_1=4096,N_2=9823,R=17,D=496$)</td>
</tr>
<tr>
<td>$g$</td>
<td>$k$</td>
<td>$N$</td>
<td>Prev$N$</td>
<td>Method</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
<td>-------</td>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>16</td>
<td>5219</td>
<td>22589</td>
<td>25310</td>
<td>CCL($l=307,k=17,N1=9823,N2=4096,R=34,D=255$)</td>
</tr>
<tr>
<td>16</td>
<td>5238</td>
<td>22606</td>
<td>25310</td>
<td>CCL($l=18,k=291,N1=4096,N2=9823,R=17,D=511$)</td>
</tr>
<tr>
<td>16</td>
<td>5526</td>
<td>22623</td>
<td>25599</td>
<td>CCL($l=18,k=307,N1=4096,N2=9823,R=17,D=512$)</td>
</tr>
<tr>
<td>16</td>
<td>5833</td>
<td>25723</td>
<td>28427</td>
<td>CCL($l=307,k=19,N1=9823,N2=6652,R=34,D=272$)</td>
</tr>
<tr>
<td>16</td>
<td>5888</td>
<td>26147</td>
<td>28444</td>
<td>CCL($l=256,k=23,N1=8191,N2=7936,R=30,D=334$)</td>
</tr>
<tr>
<td>16</td>
<td>6080</td>
<td>26161</td>
<td>28444</td>
<td>CMTW($k=380,N2=574$)</td>
</tr>
<tr>
<td>16</td>
<td>6096</td>
<td>26386</td>
<td>28444</td>
<td>CMTW($k=381,N2=589$)</td>
</tr>
<tr>
<td>16</td>
<td>6174</td>
<td>26783</td>
<td>29384</td>
<td>CCL($l=18,k=343,N1=4096,N2=13711,R=17,D=528$)</td>
</tr>
<tr>
<td>16</td>
<td>6279</td>
<td>26815</td>
<td>29639</td>
<td>CCL($l=273,k=23,N1=8191,N2=7936,R=32,D=334$)</td>
</tr>
<tr>
<td>16</td>
<td>6480</td>
<td>27038</td>
<td>30370</td>
<td>CCL($l=18,k=360,N1=4096,N2=13711,R=17,D=558$)</td>
</tr>
<tr>
<td>16</td>
<td>6588</td>
<td>27548</td>
<td>31291</td>
<td>CCL($l=18,k=381,N1=4096,N2=13711,R=17,D=573$)</td>
</tr>
<tr>
<td>16</td>
<td>6912</td>
<td>28277</td>
<td>31508</td>
<td>CMTW($k=449,N2=624$)</td>
</tr>
<tr>
<td>16</td>
<td>7098</td>
<td>28607</td>
<td>31636</td>
<td>CMTW($k=453,N2=630$)</td>
</tr>
<tr>
<td>16</td>
<td>7168</td>
<td>28727</td>
<td>31681</td>
<td>CMTW($k=457,N2=632$)</td>
</tr>
<tr>
<td>16</td>
<td>7184</td>
<td>28831</td>
<td>31816</td>
<td>CMTW($k=461,N2=635$)</td>
</tr>
<tr>
<td>16</td>
<td>7440</td>
<td>29041</td>
<td>32056</td>
<td>CCL($l=256,k=27,N1=7936,N2=8191,R=27,D=480$)</td>
</tr>
<tr>
<td>17</td>
<td>307</td>
<td>9825</td>
<td>15874</td>
<td>RMS(17)</td>
</tr>
<tr>
<td>17</td>
<td>381</td>
<td>13713</td>
<td>20189</td>
<td>RMS(19)FF</td>
</tr>
<tr>
<td>17</td>
<td>382</td>
<td>18337</td>
<td>20189</td>
<td>RMS(19)FFA</td>
</tr>
<tr>
<td>17</td>
<td>614</td>
<td>18801</td>
<td>23362</td>
<td>CKD($k=307,N2=561$)</td>
</tr>
<tr>
<td>17</td>
<td>646</td>
<td>22689</td>
<td>23362</td>
<td>CKD($k=323,N2=561$)</td>
</tr>
<tr>
<td>17</td>
<td>663</td>
<td>22721</td>
<td>23362</td>
<td>CMTW($k=39,N2=517$)</td>
</tr>
<tr>
<td>17</td>
<td>680</td>
<td>22913</td>
<td>23362</td>
<td>CMTW($k=40,N2=529$)</td>
</tr>
<tr>
<td>17</td>
<td>697</td>
<td>23105</td>
<td>23362</td>
<td>CMTW($k=41,N2=541$)</td>
</tr>
<tr>
<td>17</td>
<td>765</td>
<td>23265</td>
<td>23362</td>
<td>CMTW($k=45,N2=551$)</td>
</tr>
<tr>
<td>17</td>
<td>5219</td>
<td>23425</td>
<td>29474</td>
<td>CMTW($k=307,N2=561$)</td>
</tr>
<tr>
<td>17</td>
<td>5526</td>
<td>25394</td>
<td>31177</td>
<td>CCL($l=307,k=18,N1=9825,N2=4913,R=37,D=288$)</td>
</tr>
<tr>
<td>17</td>
<td>5780</td>
<td>27338</td>
<td>31177</td>
<td>CCL($l=289,k=20,N1=9825,N2=6793,R=32,D=335$)</td>
</tr>
<tr>
<td>17</td>
<td>5814</td>
<td>28030</td>
<td>31443</td>
<td>CCL($l=19,k=306,N1=6781,N2=9825,R=21,D=544$)</td>
</tr>
<tr>
<td>$g$</td>
<td>$k$</td>
<td>$N$</td>
<td>Prev $N$</td>
<td>Method</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
<td>-------</td>
<td>----------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>17</td>
<td>6120</td>
<td>28042</td>
<td>32138</td>
<td>CCL($l=20,k=306,N_1=6793,N_2=9825,R=21,D=544$)</td>
</tr>
<tr>
<td>17</td>
<td>6426</td>
<td>28353</td>
<td>33323</td>
<td>CMTW($k=378,N_2=626$)</td>
</tr>
<tr>
<td>17</td>
<td>6460</td>
<td>28369</td>
<td>34427</td>
<td>CMTW($k=380,N_2=627$)</td>
</tr>
<tr>
<td>17</td>
<td>6477</td>
<td>28385</td>
<td>34742</td>
<td>CMTW($k=381,N_2=628$)</td>
</tr>
<tr>
<td>17</td>
<td>6858</td>
<td>30722</td>
<td>36593</td>
<td>CCL($l=381,k=18,N_1=13713,N_2=4913,R=42,D=288$)</td>
</tr>
<tr>
<td>17</td>
<td>7239</td>
<td>32865</td>
<td>36913</td>
<td>CMTW($k=381,N_2=703$)FF</td>
</tr>
<tr>
<td>17</td>
<td>7620</td>
<td>33185</td>
<td>37233</td>
<td>CCL($l=20,k=381,N_1=6856,N_2=13715,R=19,D=664$)F</td>
</tr>
<tr>
<td>17</td>
<td>7803</td>
<td>33954</td>
<td>37233</td>
<td>CCL($l=289,k=27,N_1=9825,N_2=9537,R=32,D=456$)</td>
</tr>
<tr>
<td>17</td>
<td>8092</td>
<td>33986</td>
<td>37489</td>
<td>CCL($l=289,k=28,N_1=9825,N_2=9537,R=32,D=457$)</td>
</tr>
<tr>
<td>17</td>
<td>8381</td>
<td>34082</td>
<td>38065</td>
<td>CCL($l=289,k=29,N_1=9825,N_2=9537,R=32,D=460$)</td>
</tr>
<tr>
<td>17</td>
<td>8670</td>
<td>34146</td>
<td>38145</td>
<td>CCL($l=289,k=30,N_1=9825,N_2=9537,R=32,D=462$)</td>
</tr>
<tr>
<td>17</td>
<td>8976</td>
<td>35121</td>
<td>38689</td>
<td>CMTW($k=528,N_2=731$)</td>
</tr>
<tr>
<td>17</td>
<td>9010</td>
<td>35345</td>
<td>38913</td>
<td>CMTW($k=530,N_2=745$)</td>
</tr>
<tr>
<td>17</td>
<td>9061</td>
<td>35361</td>
<td>38929</td>
<td>CMTW($k=533,N_2=746$)</td>
</tr>
<tr>
<td>17</td>
<td>9180</td>
<td>35377</td>
<td>38945</td>
<td>CMTW($k=540,N_2=747$)</td>
</tr>
<tr>
<td>17</td>
<td>9350</td>
<td>35393</td>
<td>39025</td>
<td>CMTW($k=550,N_2=748$)</td>
</tr>
<tr>
<td>17</td>
<td>9418</td>
<td>35569</td>
<td>39201</td>
<td>CMTW($k=554,N_2=759$)</td>
</tr>
<tr>
<td>17</td>
<td>9520</td>
<td>35585</td>
<td>39217</td>
<td>CMTW($k=560,N_2=760$)</td>
</tr>
<tr>
<td>17</td>
<td>9588</td>
<td>35601</td>
<td>39233</td>
<td>CMTW($k=564,N_2=761$)</td>
</tr>
<tr>
<td>17</td>
<td>9673</td>
<td>35617</td>
<td>39249</td>
<td>CMTW($k=569,N_2=762$)</td>
</tr>
<tr>
<td>17</td>
<td>9707</td>
<td>35633</td>
<td>39265</td>
<td>CMTW($k=571,N_2=763$)</td>
</tr>
<tr>
<td>17</td>
<td>9724</td>
<td>35649</td>
<td>39281</td>
<td>CMTW($k=572,N_2=764$)</td>
</tr>
<tr>
<td>17</td>
<td>9809</td>
<td>35777</td>
<td>39296</td>
<td>CMTW($k=577,N_2=772$)</td>
</tr>
<tr>
<td>17</td>
<td>9911</td>
<td>35793</td>
<td>39296</td>
<td>CMTW($k=583,N_2=773$)</td>
</tr>
<tr>
<td>17</td>
<td>9979</td>
<td>35809</td>
<td>39296</td>
<td>CMTW($k=587,N_2=774$)</td>
</tr>
<tr>
<td>17</td>
<td>10000</td>
<td>35825</td>
<td>39296</td>
<td>CMTW($k=588,N_2=775$)</td>
</tr>
<tr>
<td>18</td>
<td>381</td>
<td>13715</td>
<td>20191</td>
<td>RMS(19)F</td>
</tr>
<tr>
<td>18</td>
<td>382</td>
<td>19223</td>
<td>20191</td>
<td>RMS(19)FA</td>
</tr>
<tr>
<td>18</td>
<td>1905</td>
<td>31337</td>
<td>35982</td>
<td>CCL($l=381,k=5,N_1=13715,N_2=5994,R=38,D=306$)</td>
</tr>
<tr>
<td>18</td>
<td>2286</td>
<td>32183</td>
<td>38336</td>
<td>CCL($l=381,k=6,N_1=13715,N_2=6156,R=38,D=324$)</td>
</tr>
<tr>
<td>18</td>
<td>4180</td>
<td>32606</td>
<td>38743</td>
<td>CCL($l=11,k=380,N_1=6579,N_2=13715,h=19,R=18,D=684$)</td>
</tr>
<tr>
<td>18</td>
<td>4560</td>
<td>32831</td>
<td>38968</td>
<td>CCL($l=12,k=380,N_1=6804,N_2=13715,h=19,R=18,D=684$)</td>
</tr>
<tr>
<td>18</td>
<td>5320</td>
<td>32859</td>
<td>38981</td>
<td>CCL($l=14,k=380,N_1=6832,N_2=13715,h=19,R=18,D=684$)</td>
</tr>
<tr>
<td>18</td>
<td>5700</td>
<td>32860</td>
<td>38997</td>
<td>CCL($l=15,k=380,N_1=6833,N_2=13715,h=19,R=18,D=684$)</td>
</tr>
<tr>
<td>18</td>
<td>7239</td>
<td>32867</td>
<td>39324</td>
<td>CMTW($k=381,N_2=703$)F</td>
</tr>
<tr>
<td>18</td>
<td>7620</td>
<td>33187</td>
<td>39663</td>
<td>CCL($l=20,k=381,N_1=6856,N_2=13715,R=19,D=664$)</td>
</tr>
<tr>
<td>18</td>
<td>8001</td>
<td>38248</td>
<td>42874</td>
<td>CCL($l=381,k=21,N_1=13715,N_2=11537,R=38,D=342$)</td>
</tr>
<tr>
<td>18</td>
<td>8382</td>
<td>41481</td>
<td>42874</td>
<td>CCL($l=381,k=22,N_1=13715,N_2=11616,R=38,D=425$)</td>
</tr>
<tr>
<td>18</td>
<td>8664</td>
<td>42744</td>
<td>43197</td>
<td>CCL($l=361,k=24,N_1=13715,N_2=11713,h=19,R=36,D=481$)</td>
</tr>
<tr>
<td>18</td>
<td>9144</td>
<td>43250</td>
<td>44242</td>
<td>CCL($l=381,k=24,N_1=13715,N_2=11713,R=38,D=469$)</td>
</tr>
<tr>
<td>18</td>
<td>9386</td>
<td>44059</td>
<td>44242</td>
<td>CCL($l=361,k=26,N_1=13715,N_2=12884,h=19,R=36,D=485$)</td>
</tr>
<tr>
<td>$g$</td>
<td>$k$</td>
<td>$N$</td>
<td>Prev$N$</td>
<td>Method</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
<td>-------</td>
<td>---------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>18</td>
<td>9747</td>
<td>44167</td>
<td>44242</td>
<td>CCL($l=361,k=27,N1=13715,N2=12884,h=19,R=36,D=488$)</td>
</tr>
<tr>
<td>18</td>
<td>10000</td>
<td>44239</td>
<td>44242</td>
<td>CCL($l=358,k=28,N1=13715,N2=12884,h=19,R=36,D=490$)</td>
</tr>
<tr>
<td>19</td>
<td>381</td>
<td>13717</td>
<td>24158</td>
<td>RMS(19)</td>
</tr>
<tr>
<td>19</td>
<td>382</td>
<td>20215</td>
<td>24158</td>
<td>RMS(19)A</td>
</tr>
<tr>
<td>19</td>
<td>553</td>
<td>24325</td>
<td>30187</td>
<td>RMS(23)FFFF</td>
</tr>
<tr>
<td>19</td>
<td>762</td>
<td>26371</td>
<td>33176</td>
<td>CKD($k=381,N2=703$)</td>
</tr>
<tr>
<td>19</td>
<td>855</td>
<td>32095</td>
<td>36038</td>
<td>CMTW($k=45,N2=660$)</td>
</tr>
<tr>
<td>19</td>
<td>874</td>
<td>32329</td>
<td>36038</td>
<td>CMTW($k=46,N2=673$)</td>
</tr>
<tr>
<td>19</td>
<td>893</td>
<td>32563</td>
<td>36038</td>
<td>CMTW($k=47,N2=686$)</td>
</tr>
<tr>
<td>19</td>
<td>950</td>
<td>32779</td>
<td>36038</td>
<td>CMTW($k=50,N2=698$)</td>
</tr>
<tr>
<td>19</td>
<td>7239</td>
<td>32869</td>
<td>43310</td>
<td>CMTW($k=381,N2=703$)</td>
</tr>
<tr>
<td>19</td>
<td>7620</td>
<td>37496</td>
<td>47809</td>
<td>CCL($l=381,k=20,N1=13717,N2=6859,R=47,D=360$)</td>
</tr>
<tr>
<td>19</td>
<td>7942</td>
<td>42896</td>
<td>47809</td>
<td>CCL($l=361,k=22,N1=13717,N2=11863,R=36,D=481$)</td>
</tr>
<tr>
<td>19</td>
<td>8303</td>
<td>42922</td>
<td>47835</td>
<td>CCL($l=361,k=23,N1=13717,N2=11889,R=36,D=481$)</td>
</tr>
<tr>
<td>19</td>
<td>8664</td>
<td>42960</td>
<td>48006</td>
<td>CCL($l=361,k=24,N1=13717,N2=11927,R=36,D=481$)</td>
</tr>
<tr>
<td>19</td>
<td>9025</td>
<td>44462</td>
<td>48006</td>
<td>CCL($l=361,k=25,N1=13717,N2=13357,R=36,D=483$)</td>
</tr>
<tr>
<td>19</td>
<td>9386</td>
<td>44534</td>
<td>48006</td>
<td>CCL($l=361,k=26,N1=13717,N2=13357,R=36,D=485$)</td>
</tr>
<tr>
<td>19</td>
<td>9747</td>
<td>44642</td>
<td>48006</td>
<td>CCL($l=361,k=27,N1=13717,N2=13357,R=36,D=488$)</td>
</tr>
<tr>
<td>19</td>
<td>10000</td>
<td>44714</td>
<td>48006</td>
<td>CCL($l=358,k=28,N1=13717,N2=13357,R=36,D=490$)</td>
</tr>
<tr>
<td>20</td>
<td>553</td>
<td>24327</td>
<td>35941</td>
<td>RMS(23)FFFF</td>
</tr>
<tr>
<td>20</td>
<td>651</td>
<td>31239</td>
<td>36040</td>
<td>RMS(25)FFFFF</td>
</tr>
<tr>
<td>20</td>
<td>2425</td>
<td>49902</td>
<td>50440</td>
<td>CCL($l=5,k=485,N1=8000,N2=24327,R=19,D=925$)</td>
</tr>
<tr>
<td>20</td>
<td>2655</td>
<td>49921</td>
<td>57370</td>
<td>CCL($l=5,k=531,N1=8000,N2=24327,R=19,D=926$)</td>
</tr>
<tr>
<td>20</td>
<td>2760</td>
<td>49959</td>
<td>58894</td>
<td>CCL($l=5,k=552,N1=8000,N2=24327,R=19,D=928$)</td>
</tr>
<tr>
<td>20</td>
<td>2765</td>
<td>49978</td>
<td>58894</td>
<td>CCL($l=5,k=553,N1=8000,N2=24327,R=19,D=929$)</td>
</tr>
<tr>
<td>20</td>
<td>2772</td>
<td>52655</td>
<td>58894</td>
<td>CCL($l=6,k=462,N1=8000,N2=24327,R=22,D=924$)</td>
</tr>
<tr>
<td>20</td>
<td>2910</td>
<td>52677</td>
<td>58894</td>
<td>CCL($l=6,k=485,N1=8000,N2=24327,R=22,D=925$)</td>
</tr>
<tr>
<td>20</td>
<td>3186</td>
<td>52699</td>
<td>59084</td>
<td>CCL($l=6,k=531,N1=8000,N2=24327,R=22,D=926$)</td>
</tr>
<tr>
<td>20</td>
<td>3312</td>
<td>52743</td>
<td>59084</td>
<td>CCL($l=6,k=552,N1=8000,N2=24327,R=22,D=928$)</td>
</tr>
<tr>
<td>20</td>
<td>3318</td>
<td>52765</td>
<td>59084</td>
<td>CCL($l=6,k=553,N1=8000,N2=24327,R=22,D=929$)</td>
</tr>
<tr>
<td>20</td>
<td>3512</td>
<td>55385</td>
<td>59274</td>
<td>CCL($l=8,k=439,N1=8930,N2=24327,R=24,D=922$)</td>
</tr>
<tr>
<td>20</td>
<td>3696</td>
<td>55433</td>
<td>59274</td>
<td>CCL($l=8,k=462,N1=8930,N2=24327,R=24,D=924$)</td>
</tr>
<tr>
<td>20</td>
<td>3880</td>
<td>55457</td>
<td>59274</td>
<td>CCL($l=8,k=485,N1=8930,N2=24327,R=24,D=925$)</td>
</tr>
<tr>
<td>20</td>
<td>4248</td>
<td>55481</td>
<td>59844</td>
<td>CCL($l=8,k=531,N1=8930,N2=24327,R=24,D=926$)</td>
</tr>
<tr>
<td>20</td>
<td>6072</td>
<td>55521</td>
<td>60414</td>
<td>CCL($l=11,k=552,N1=8930,N2=24327,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>20</td>
<td>6083</td>
<td>56005</td>
<td>60414</td>
<td>CCL($l=11,k=553,N1=8930,N2=24327,h=23,R=22,D=1034$)</td>
</tr>
<tr>
<td>20</td>
<td>6624</td>
<td>56091</td>
<td>60414</td>
<td>CCL($l=12,k=552,N1=9500,N2=24327,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>20</td>
<td>7728</td>
<td>56281</td>
<td>64746</td>
<td>CCL($l=14,k=552,N1=9690,N2=24327,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>20</td>
<td>8832</td>
<td>56471</td>
<td>66325</td>
<td>CCL($l=16,k=552,N1=9880,N2=24327,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>20</td>
<td>9384</td>
<td>56661</td>
<td>66836</td>
<td>CCL($l=17,k=552,N1=10070,N2=24327,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>20</td>
<td>10000</td>
<td>56851</td>
<td>67714</td>
<td>CCL($l=19,k=527,N1=10070,N2=24327,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>$g$</td>
<td>$k$</td>
<td>$N$</td>
<td>$\text{Prev}N$</td>
<td>Method</td>
</tr>
<tr>
<td>-----</td>
<td>------</td>
<td>-------</td>
<td>-----------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>21</td>
<td>553</td>
<td>24329</td>
<td>35943</td>
<td>RMS(23)FF</td>
</tr>
<tr>
<td>21</td>
<td>651</td>
<td>31241</td>
<td>45319</td>
<td>RMS(25)FFFF</td>
</tr>
<tr>
<td>21</td>
<td>757</td>
<td>39353</td>
<td>47670</td>
<td>RMS(27)FFFFFF</td>
</tr>
<tr>
<td>21</td>
<td>768</td>
<td>43849</td>
<td>47670</td>
<td>CKD($k=384,N_2=976$)</td>
</tr>
<tr>
<td>21</td>
<td>864</td>
<td>43869</td>
<td>47670</td>
<td>CKD($k=432,N_2=977$)</td>
</tr>
<tr>
<td>21</td>
<td>960</td>
<td>43889</td>
<td>47670</td>
<td>CKD($k=480,N_2=978$)</td>
</tr>
<tr>
<td>21</td>
<td>1056</td>
<td>43909</td>
<td>47670</td>
<td>CKD($k=528,N_2=979$)</td>
</tr>
<tr>
<td>21</td>
<td>1104</td>
<td>43929</td>
<td>47670</td>
<td>CKD($k=552,N_2=980$)</td>
</tr>
<tr>
<td>21</td>
<td>1106</td>
<td>43949</td>
<td>47670</td>
<td>CKD($k=553,N_2=981$)</td>
</tr>
<tr>
<td>21</td>
<td>2496</td>
<td>52710</td>
<td>57372</td>
<td>CCL($l=6,k=416,N_1=9261,N_2=24329,R=20,D=956$)</td>
</tr>
<tr>
<td>21</td>
<td>2772</td>
<td>52730</td>
<td>59997</td>
<td>CCL($l=6,k=462,N_1=9261,N_2=24329,R=20,D=957$)</td>
</tr>
<tr>
<td>21</td>
<td>3048</td>
<td>52750</td>
<td>63835</td>
<td>CCL($l=6,k=508,N_1=9261,N_2=24329,R=20,D=958$)</td>
</tr>
<tr>
<td>21</td>
<td>3312</td>
<td>52770</td>
<td>63858</td>
<td>CCL($l=6,k=552,N_1=9261,N_2=24329,R=20,D=959$)</td>
</tr>
<tr>
<td>21</td>
<td>3318</td>
<td>52810</td>
<td>64424</td>
<td>CCL($l=6,k=553,N_1=9261,N_2=24329,R=20,D=961$)</td>
</tr>
<tr>
<td>21</td>
<td>3871</td>
<td>56864</td>
<td>68342</td>
<td>CCL($l=553,k=7,N_1=24329,N_2=11285,R=50,D=425$)</td>
</tr>
<tr>
<td>21</td>
<td>4416</td>
<td>58105</td>
<td>69193</td>
<td>CCL($l=8,k=552,N_1=11512,N_2=24329,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>21</td>
<td>4968</td>
<td>58334</td>
<td>69422</td>
<td>CCL($l=9,k=552,N_1=11741,N_2=24329,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>21</td>
<td>5520</td>
<td>58367</td>
<td>69455</td>
<td>CCL($l=10,k=552,N_1=11774,N_2=24329,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>21</td>
<td>6624</td>
<td>58724</td>
<td>69811</td>
<td>CCL($l=12,k=552,N_1=12131,N_2=24329,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>21</td>
<td>7728</td>
<td>58726</td>
<td>69813</td>
<td>CCL($l=14,k=552,N_1=12133,N_2=24329,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>21</td>
<td>8280</td>
<td>58728</td>
<td>69816</td>
<td>CCL($l=15,k=552,N_1=12135,N_2=24329,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>21</td>
<td>10000</td>
<td>58734</td>
<td>69819</td>
<td>CCL($l=19,k=527,N_1=12141,N_2=24329,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>22</td>
<td>553</td>
<td>24331</td>
<td>35945</td>
<td>RMS(23)F</td>
</tr>
<tr>
<td>22</td>
<td>651</td>
<td>31243</td>
<td>45321</td>
<td>RMS(25)FFFF</td>
</tr>
<tr>
<td>22</td>
<td>757</td>
<td>39355</td>
<td>50864</td>
<td>RMS(27)FFFFFF</td>
</tr>
<tr>
<td>22</td>
<td>1008</td>
<td>45520</td>
<td>50864</td>
<td>CKD($k=504,N_2=1009$)</td>
</tr>
<tr>
<td>22</td>
<td>1106</td>
<td>45541</td>
<td>50864</td>
<td>CKD($k=553,N_2=1010$)</td>
</tr>
<tr>
<td>22</td>
<td>2765</td>
<td>56473</td>
<td>62964</td>
<td>CCL($l=553,k=5,N_1=24331,N_2=10890,R=46,D=462$)</td>
</tr>
<tr>
<td>22</td>
<td>3318</td>
<td>56899</td>
<td>68030</td>
<td>CCL($l=553,k=6,N_1=24331,N_2=11132,R=46,D=466$)</td>
</tr>
<tr>
<td>22</td>
<td>3360</td>
<td>58346</td>
<td>68030</td>
<td>CCL($l=24,k=140,N_1=12165,N_2=24331,R=23,D=950$)</td>
</tr>
<tr>
<td>22</td>
<td>3480</td>
<td>58369</td>
<td>69179</td>
<td>CCL($l=24,k=145,N_1=12165,N_2=24331,R=23,D=951$)</td>
</tr>
<tr>
<td>22</td>
<td>3871</td>
<td>58460</td>
<td>69591</td>
<td>CCL($l=553,k=7,N_1=24331,N_2=11865,R=46,D=484$)</td>
</tr>
<tr>
<td>22</td>
<td>4416</td>
<td>58660</td>
<td>69770</td>
<td>CCL($l=8,k=552,N_1=12065,N_2=24331,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>22</td>
<td>5520</td>
<td>58698</td>
<td>69808</td>
<td>CCL($l=10,k=552,N_1=12103,N_2=24331,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>22</td>
<td>6624</td>
<td>58728</td>
<td>69819</td>
<td>CCL($l=12,k=552,N_1=12133,N_2=24331,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>22</td>
<td>7728</td>
<td>58730</td>
<td>69820</td>
<td>CCL($l=14,k=552,N_1=12135,N_2=24331,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>22</td>
<td>8280</td>
<td>58732</td>
<td>69842</td>
<td>CCL($l=15,k=552,N_1=12137,N_2=24331,h=23,R=22,D=1012$)</td>
</tr>
<tr>
<td>22</td>
<td>10000</td>
<td>58739</td>
<td>69849</td>
<td>CMTW($k=434,N_2=1035$)F</td>
</tr>
<tr>
<td>23</td>
<td>553</td>
<td>24333</td>
<td>38590</td>
<td>RMS(23)</td>
</tr>
<tr>
<td>23</td>
<td>651</td>
<td>31245</td>
<td>46194</td>
<td>RMS(25)FF</td>
</tr>
<tr>
<td>23</td>
<td>757</td>
<td>39357</td>
<td>52889</td>
<td>RMS(27)FFFF</td>
</tr>
<tr>
<td>$g$</td>
<td>$k$</td>
<td>$N$</td>
<td>Prev $N$</td>
<td>Method</td>
</tr>
<tr>
<td>-----</td>
<td>--------</td>
<td>--------</td>
<td>----------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>23</td>
<td>1106</td>
<td>47103</td>
<td>59204</td>
<td>CKD($k=553,N2=1035$)</td>
</tr>
<tr>
<td>23</td>
<td>1150</td>
<td>54015</td>
<td>59204</td>
<td>CKD($k=575,N2=1035$)</td>
</tr>
<tr>
<td>23</td>
<td>1151</td>
<td>54477</td>
<td>59204</td>
<td>CKD($k=576,N2=1056$)</td>
</tr>
<tr>
<td>23</td>
<td>1196</td>
<td>55005</td>
<td>63221</td>
<td>CKD($k=598,N2=1080$)</td>
</tr>
<tr>
<td>23</td>
<td>1198</td>
<td>55446</td>
<td>64564</td>
<td>CKD($k=599,N2=1101$)</td>
</tr>
<tr>
<td>23</td>
<td>1248</td>
<td>55489</td>
<td>64564</td>
<td>CKD($k=624,N2=1102$)</td>
</tr>
<tr>
<td>23</td>
<td>1288</td>
<td>55973</td>
<td>67366</td>
<td>CKD($k=644,N2=1124$)</td>
</tr>
<tr>
<td>23</td>
<td>1300</td>
<td>55995</td>
<td>68709</td>
<td>CKD($k=650,N2=1125$)</td>
</tr>
<tr>
<td>23</td>
<td>1302</td>
<td>56017</td>
<td>68709</td>
<td>CKD($k=651,N2=1126$)</td>
</tr>
<tr>
<td>23</td>
<td>1334</td>
<td>58389</td>
<td>68709</td>
<td>CMTW($k=58,N2=1019$)</td>
</tr>
<tr>
<td>23</td>
<td>1380</td>
<td>58455</td>
<td>69565</td>
<td>CMTW($k=60,N2=1022$)</td>
</tr>
<tr>
<td>23</td>
<td>10000</td>
<td>58741</td>
<td>69851</td>
<td>CMTW($k=434,N2=1035$)</td>
</tr>
<tr>
<td>24</td>
<td>651</td>
<td>31247</td>
<td>46196</td>
<td>RMS(25)F</td>
</tr>
<tr>
<td>24</td>
<td>757</td>
<td>39359</td>
<td>57207</td>
<td>RMS(27)FFF</td>
</tr>
<tr>
<td>24</td>
<td>871</td>
<td>48767</td>
<td>65206</td>
<td>RMS(29)FFFFF</td>
</tr>
<tr>
<td>24</td>
<td>894</td>
<td>58575</td>
<td>65279</td>
<td>CKD($k=447,N2=1196$)</td>
</tr>
<tr>
<td>24</td>
<td>1152</td>
<td>58778</td>
<td>73060</td>
<td>CKD($k=576,N2=1197$)</td>
</tr>
<tr>
<td>24</td>
<td>1302</td>
<td>58801</td>
<td>73060</td>
<td>CKD($k=651,N2=1198$)</td>
</tr>
<tr>
<td>24</td>
<td>1352</td>
<td>66913</td>
<td>73060</td>
<td>CKD($k=676,N2=1198$)</td>
</tr>
<tr>
<td>24</td>
<td>1354</td>
<td>66936</td>
<td>77117</td>
<td>CKD($k=677,N2=1199$)</td>
</tr>
<tr>
<td>24</td>
<td>1406</td>
<td>66959</td>
<td>79164</td>
<td>CKD($k=703,N2=1200$)</td>
</tr>
<tr>
<td>24</td>
<td>1410</td>
<td>67465</td>
<td>79164</td>
<td>CKD($k=705,N2=1222$)</td>
</tr>
<tr>
<td>24</td>
<td>1456</td>
<td>67488</td>
<td>79164</td>
<td>CKD($k=728,N2=1223$)</td>
</tr>
<tr>
<td>24</td>
<td>1458</td>
<td>67511</td>
<td>79512</td>
<td>CKD($k=729,N2=1224$)</td>
</tr>
<tr>
<td>24</td>
<td>1512</td>
<td>68017</td>
<td>79512</td>
<td>CKD($k=756,N2=1246$)</td>
</tr>
<tr>
<td>24</td>
<td>1514</td>
<td>68960</td>
<td>79512</td>
<td>CKD($k=757,N2=1287$)</td>
</tr>
<tr>
<td>24</td>
<td>2010</td>
<td>72004</td>
<td>79512</td>
<td>CCL($l=6,k=335,N1=13824,N2=31247,R=23,D=1171$)</td>
</tr>
<tr>
<td>24</td>
<td>2628</td>
<td>72027</td>
<td>79512</td>
<td>CCL($l=6,k=438,N1=13824,N2=31247,R=23,D=1172$)</td>
</tr>
<tr>
<td>24</td>
<td>3384</td>
<td>72050</td>
<td>82968</td>
<td>CCL($l=6,k=564,N1=13824,N2=31247,R=23,D=1173$)</td>
</tr>
<tr>
<td>24</td>
<td>3906</td>
<td>72073</td>
<td>87022</td>
<td>CCL($l=6,k=651,N1=13824,N2=31247,R=23,D=1174$)</td>
</tr>
<tr>
<td>24</td>
<td>4512</td>
<td>73406</td>
<td>87782</td>
<td>CCL($l=6,k=564,N1=15180,N2=31247,R=23,D=1173$)</td>
</tr>
<tr>
<td>24</td>
<td>5208</td>
<td>73429</td>
<td>88378</td>
<td>CCL($l=6,k=651,N1=15180,N2=31247,R=23,D=1174$)</td>
</tr>
<tr>
<td>24</td>
<td>5859</td>
<td>74477</td>
<td>88876</td>
<td>CCL($l=651,k=9,N1=31247,N2=15180,R=50,D=561$)</td>
</tr>
<tr>
<td>24</td>
<td>7150</td>
<td>75227</td>
<td>89603</td>
<td>CCL($l=11,k=650,N1=15180,N2=31247,h=25,R=24,D=1200$)</td>
</tr>
<tr>
<td>24</td>
<td>10000</td>
<td>75647</td>
<td>90023</td>
<td>CMTW($k=400,N2=1225$)</td>
</tr>
<tr>
<td>25</td>
<td>651</td>
<td>31249</td>
<td>49346</td>
<td>RMS(25)</td>
</tr>
<tr>
<td>25</td>
<td>757</td>
<td>39361</td>
<td>59571</td>
<td>RMS(27)FF</td>
</tr>
<tr>
<td>25</td>
<td>871</td>
<td>48769</td>
<td>67561</td>
<td>RMS(29)FFFFF</td>
</tr>
<tr>
<td>25</td>
<td>993</td>
<td>59569</td>
<td>68905</td>
<td>RMS(31)FFFFF</td>
</tr>
<tr>
<td>25</td>
<td>1302</td>
<td>60649</td>
<td>76202</td>
<td>CKD($k=651,N2=1225$)</td>
</tr>
<tr>
<td>25</td>
<td>1350</td>
<td>68761</td>
<td>76202</td>
<td>CKD($k=675,N2=1225$)</td>
</tr>
<tr>
<td>$g$</td>
<td>$k$</td>
<td>$N$</td>
<td>PrevN</td>
<td>Method</td>
</tr>
<tr>
<td>-----</td>
<td>------</td>
<td>-------</td>
<td>-------</td>
<td>---------------------</td>
</tr>
<tr>
<td>25</td>
<td>1352</td>
<td>69313</td>
<td>76202</td>
<td>CKD($k=676,N2=1248$)</td>
</tr>
<tr>
<td>25</td>
<td>1400</td>
<td>69937</td>
<td>81191</td>
<td>CKD($k=700,N2=1274$)</td>
</tr>
<tr>
<td>25</td>
<td>1402</td>
<td>70489</td>
<td>82512</td>
<td>CKD($k=701,N2=1297$)</td>
</tr>
<tr>
<td>25</td>
<td>1456</td>
<td>70513</td>
<td>82512</td>
<td>CKD($k=728,N2=1298$)</td>
</tr>
<tr>
<td>25</td>
<td>1508</td>
<td>71161</td>
<td>87277</td>
<td>CKD($k=754,N2=1325$)</td>
</tr>
<tr>
<td>25</td>
<td>1514</td>
<td>71185</td>
<td>87301</td>
<td>CKD($k=757,N2=1326$)</td>
</tr>
<tr>
<td>25</td>
<td>1750</td>
<td>75505</td>
<td>89881</td>
<td>CMTW($k=70,N2=1219$)</td>
</tr>
<tr>
<td>25</td>
<td>10000</td>
<td>75649</td>
<td>90025</td>
<td>CMTW($k=400,N2=1225$)</td>
</tr>
</tbody>
</table>
Chapter 5

**VarDens: A Density-Based Greedy Algorithm for Variable Strength Covering Arrays**

In this chapter, we generalize the ideas of Bryce and Colbourn in [10, 11, 32] to derive a density-based greedy algorithm that, given an ASC and numbers of levels for each factor, builds a mixed-level variable strength covering array one row at a time; this algorithm optimizes for the interactions prescribed in the ASC. In Section 5.1, we give a precise, detailed, step by step description of how the algorithm works and the steps involved. We believe this presentation will serve to clarify details needed for the implementation of greedy density methods for covering array generation. Section 5.2 contains an analysis of the algorithm to prove the logarithmic bound, i.e. that $\text{VCAN}(\Lambda, (g_1, \ldots, g_k)) \leq m(\ln |\Lambda| + \ln m)$, where $m = \max\{\prod_{i \in W} g_i : W \in \Lambda\}$.

In Section 5.3, we detail the variants of our algorithm that will be used in the experiments to determine how the different variants will trade off between execution speed and array size. Finally, in Section 5.4 we begin by comparing the sizes of arrays generated by our algorithm with the fixed strength-$t$ greedy algorithm of Bryce and Colbourn [11], and then show how our algorithm performs in comparison with the simulated annealing technique. We then use our algorithm to investigate arrays over two families of hypergraphs, namely hypertrees and Steiner triple systems, which have the property that they are the smallest collection of 3-sets of points such that for any pair of points, there is a 3-set containing the pair. We consider practical modifications to our algorithm and variations to test for both speed and resultant array size.
The results of this chapter have been prepared and submitted in [77].

5.1 VarDens: A Density Algorithm for VCA

Let $\Delta$ be an abstract simplicial complex with facet set $\Lambda \subseteq \Delta$. We now present an algorithm that we call VarDens (or the Variable strength covering array Density-based greedy algorithm) that creates a VCA($N; \Lambda, g$) one test, or row, at a time, and creates each row factor-by-factor, iteratively selecting and fixing a factor to a chosen level. This is a generalization of the density-based greedy algorithm by Bryce and Colbourn [11] for finding CA for strength $t > 2$, which is an extension of the deterministic density algorithm for strength-2 CA [10, 32]. Here we also provide more detail into the steps of the algorithm, which is necessary for the generalized setting. The basic idea is to begin with an empty array, and then at each stage, select a row that, out of all possible rows, covers at least the average number of uncovered interactions as in [10, 11]; this allows us to guarantee an upper bound on the size of the arrays produced, as will be seen in Section 5.2 while remaining in polynomial time.

Let $S$ be the universe of all interactions of any size over the set of factors $K = \{0, \ldots, k - 1\}$ with level values from $\{0, \ldots, g_1 - 1\}, \ldots, \{0, \ldots, g_k - 1\}$, and for $W \subseteq K$, let $\mathcal{I}(W)$ be the set of all interactions over the factors in $W$. We start with an empty array $\mathcal{T}$. Our strategy will be to build up an interaction $S$, initially $\emptyset$, progressively fixing factor-by-factor, until all factors are fixed (i.e. it is a full $k$-interaction); then this is a row which we add to $\mathcal{T}$.

For the sake of simplicity, when it is unambiguous to do so, we refer to factors and their indices interchangeably. Then, for factor $f$, $g_f$ indicates the number of levels of $f$, and in an interaction $S$, $\sigma_f$ refers to the value of $f$ with respect to $S$ (i.e. $(f, \sigma_f) \in S$).

Let $\mathcal{I}_\Lambda$ be the set of all interactions over all $W \in \Lambda$, and define a coverage function $\gamma : \mathcal{I}_\Lambda \to \{0, 1\}$ to serve as an indicator whether or not an interaction is uncovered: that is to say $\gamma(S) = 0$ if $S$ is covered in some row (i.e. there exists $T \in \mathcal{T}$ with $S \subseteq T$) and 1 otherwise (i.e. $S$ is not covered in any row). Assume that we have fixed some interaction $S$ thus far. We now detail our strategy to pick a level for an arbitrary free factor $f$.

Let $W \in \Lambda$. Take $E(S, W)$ to be the set of all possible interactions over $W$ that respect $S$ (i.e. the extensions of $S$ to $W$), written:

$$E(S, W) = \left\{ \left( \bigcup_{f \in \phi(S) \cap W} \{(f, \sigma_f)\} \right) \bigcup \left( \bigcup_{f \in W \setminus \phi(S)} \{(f, a_f)\} \right) : a_f \in \{0, \ldots, g_f - 1\} \text{ for all } f \in W \setminus \phi(S) \right\}.$$
We note that \(|E(S,W)| = \prod_{f \in W \setminus \phi(S)} g_f\). Define \(r(S,W)\) to be the number of interactions in \(E(S,W)\) that are not yet covered in some row, i.e. \(r(S,W) = \sum_{I \in E(S,W)} \gamma(I)\).

**Definition 5.1.1.** The density of a set \(W\) over an interaction \(S\) is the ratio of uncovered interactions over \(W\) respecting \(S\) to the total number of interactions over \(W\) respecting \(S\):

\[
\delta(S,W) = \frac{\sum_{I \in E(S,W)} \gamma(I)}{|E(S,W)|} = \frac{r(S,W)}{|E(S,W)|} = \frac{r(S,W)}{\prod_{f \in W \setminus \phi(S)} g_f}.
\] (5.1)

**Definition 5.1.2.** The interaction density of an interaction \(S\) is \(\delta(S) = \sum_{W \in A} \delta(S,W)\), which, for a given factor \(f\), can be rewritten:

\[
\delta(S) = \sum_{W \in A} \frac{\delta(S,W)}{r_f} + \sum_{W \in A} \frac{\delta(S,W)}{r_f^\prime}.
\] (5.2)

For the purposes of selecting a level \(\sigma\) for the factor \(f\), in Equation 5.2, the first summation is independent of our choice of level for \(f\), so we need only focus on the impact of any choice of level on the second summation.

**Definition 5.1.3.** The factor density of \(f\) with respect to interaction \(S\) is defined as follows:

\[
\delta_f(S) = \sum_{W \in A} \frac{\delta(S,W)}{r_f}.
\] (5.3)

**Proposition 5.1.4.** For a factor \(f\) and an interaction \(S\), the factor density \(\delta_f(S)\) is the average number of uncovered interactions extending \(S\) across all choices of levels for \(f\), i.e.:

\[
\delta_f(S) = \frac{1}{g_f} \sum_{\sigma \in \{0, \ldots, g_f - 1\}} \delta_f(S \cup \{(f, \sigma)\}).
\]

**Proof.** We compute as follows:

\[
\frac{1}{g_f} \sum_{\sigma \in \{0, \ldots, g_f - 1\}} \delta_f(S \cup \{(f, \sigma)\}) = \frac{1}{g_f} \sum_{\sigma \in \{0, \ldots, g_f - 1\}} \sum_{W \in A} \frac{\delta(S \cup \{(f, \sigma)\}, W)}{r_f} = \frac{1}{g_f} \sum_{W \in A} \sum_{\sigma \in \{0, \ldots, g_f - 1\}} \frac{\delta(S \cup \{(f, \sigma)\}, W)}{r_f} \prod_{f \in W \setminus \phi(S)} g_f.
\] (5.4)

Note that, for \(W \in A\) with \(f \in W\), if \(I \in E(S,W)\), then \(I\) has some value \(\sigma\) for \(f\) (i.e. \((f, \sigma) \in I\)). Thus, \(I \in E(S \cup \{(f, \sigma)\}, W)\) and \(I \notin E(S \cup \{(f, \sigma')\}, W)\) for all \(\sigma' \in \{0, \ldots, g_f - 1\} \setminus \{\sigma\}\). Using this, as \(f \in W\), then we have that:

\[
\sum_{\sigma \in \{0, \ldots, g_f - 1\}} r(S \cup \{(f, \sigma)\}, W) = \sum_{\sigma \in \{0, \ldots, g_f - 1\}} \sum_{I \in E(S \cup \{(f, \sigma)\}, W)} \gamma(I) = \sum_{I \in E(S,W)} \gamma(I) = r(S,W).
\]
Thus, applying this to Equation 5.4 gives that:

\[
\frac{1}{g_f} \sum_{\sigma \in \{0, \ldots, g_f - 1\}} \delta_f(S \cup \{(f, \sigma)\}) = \sum_{W \in \Lambda} \frac{r(S, W)}{|E(S, W)|} = \sum_{W \in \Lambda} \delta(S, W) = \delta_f(S).
\]

Our strategy, then, in selecting a level for factor \(f\), is to choose a value \(\sigma_f \in \{0, \ldots, g_f - 1\}\) that achieves at least average density, and the proposition proves that such a choice is always possible:

\[
\delta_f(S \cup \{(f, \sigma_f)\}) \geq \frac{1}{g_f} \sum_{\sigma \in \{0, \ldots, g_f - 1\}} \delta_f(S \cup \{(f, \sigma)\}).
\]

\textsc{VarDens} selects \(\sigma_f\) to maximize the value of the left hand side of this inequality, and is detailed in Algorithm 1.

\begin{algorithm}
\caption{\textsc{VarDens}(\(\Lambda, g = (g_1, \ldots, g_k)\))}
\begin{algorithmic}
\Let {T} \leftarrow \emptyset.
\While {there are interactions over \(\Lambda\) which are uncovered in \(T\)}
\State Create a new row \(S = \emptyset\)
\State Let \(F \leftarrow \{0, \ldots, k - 1\}\).
\While {\(F \neq \emptyset\)}
\State Pick any \(f \in F\).
\State Choose \(\sigma_f \in \{0, \ldots, g_f - 1\}\) such that \(\delta_f(S \cup \{(f, \sigma_f)\})\) is maximized.
\State \(S \leftarrow S \cup \{(f, \sigma_f)\}\)
\State \(F \leftarrow F \setminus \{f\}\)
\EndWhile
\State \(T \leftarrow T \cup \{S\}\)
\EndWhile
\State return \(T\)
\end{algorithmic}
\end{algorithm}

We illustrate the execution of the algorithm with an example of selecting a level for a factor.

\textbf{Example 5.1.5.} Assume we have the graph \(G\) as shown in Figure 5.1 and we are using \textsc{VarDens} to construct a VCA(G, 2). The algorithm now selects the level for factor \(f_2\) as shown in the example. We calculate the density for both choices of level, 0 and 1:
5.2 Logarithmic Guarantee

In the equations in this section, we write intervals \( \{i, i+1, \ldots, j\} \) as \([i, j]\) for compactness; this should not be confused with references. Let \( Z = \emptyset \) be the unique empty interaction, and prior to creating a new row, define the global density at this stage of the algorithm as follows:

\[
\delta = \delta(Z). \tag{5.5}
\]

We now show that a row constructed by Algorithm 1 covers at least \( \delta \) uncovered interactions. Assume without loss of generality that, for the current row, we fix the factors in the order \( 0, 1, \ldots, k-1 \) to values \( \sigma_0, \sigma_1, \ldots, \sigma_{k-1} \) respectively. Let \( \delta^{(h)} \) be the density of the row \( S \) (i.e., \( \delta(S) \)) after fixing factor \( h \) to \( \sigma_h \). Then \( \delta^{(h)} \) can be calculated as follows:

\[
\delta^{(h)} = \sum_{W \subseteq [0, h], W \neq \emptyset} r(S, W) + \sum_{W \subseteq [h+1, k-1], W \neq \emptyset} \prod_{i \in W} g_i + \sum_{\substack{W \subseteq [0, h] \neq \emptyset \atop W \cap [h+1, k-1] \neq \emptyset}} \prod_{i \in W \setminus [0, h]} g_i \tag{5.6}
\]
\[ \delta = \sum_{W \in \Lambda} \delta(Z) = \sum_{W \in \Lambda} \sum_{I \in \mathcal{I}(W)} \frac{\gamma(I)}{|\mathcal{I}(W)|} \]

(5.8)

**Theorem 5.2.1.** Let \( \Lambda \) be an ASC over \( k \) factors, and let \( \mathcal{T} \) represent the partial VCA(\( \Lambda, (g_0, \ldots, g_{k-1}) \)). A row created by Algorithm I covers at least \( \delta \) new interactions, where:

\[ \delta = \delta(Z) = \delta(-1) = \sum_{W \in \Lambda} \sum_{I \in \mathcal{I}(W)} \frac{\gamma(I)}{|\mathcal{I}(W)|}. \]

**Proof.** We first show that \( \delta \) has the value given in Equation 5.8:

\[ \delta(Z) = \sum_{W \in \Lambda} \delta(Z, W) = \sum_{W \in \Lambda} \sum_{I \in \mathcal{I}(Z, W)} \frac{\gamma(I)}{\prod_{f \in W \setminus \phi(Z)} g_f} = \sum_{W \in \Lambda} \sum_{I \in \mathcal{I}(W)} \frac{\gamma(I)}{\prod_{f \in W} g_f} = \sum_{W \in \Lambda} \sum_{I \in \mathcal{I}(W)} \frac{\gamma(I)}{|\mathcal{I}(W)|}. \]

Assume that we have just extended interaction \( S \) to \( S' \) by fixing factor \( h + 1 \) to \( \sigma_{h+1} \), i.e. \( S' = S \cup \{(h + 1, \sigma_{h+1})\} \). It suffices to show that \( \delta(h+1) \geq \delta(h) \). Consider \( \delta(h+1) \) as defined in Equation 5.7.

\[ \delta(h+1) = \sum_{W \in \Lambda} r(S', W) + \sum_{W \in \Lambda} \sum_{W' \subseteq [h+1, k-1]} \frac{r(S', W)}{\prod_{i \in W} g_i} + \sum_{W \in \Lambda} \sum_{W' \subseteq [h+1, k-1]} \frac{r(S', W)}{\prod_{i \in W \setminus [0, h+1]} g_i} + \sum_{W \in \Lambda} \sum_{W' \subseteq [h+1, k-1]} \frac{r(S', W)}{\prod_{i \in W \setminus [0, h+1]} g_i}. \]

We can substitute interaction \( S \) for \( S' \) wherever we do not use the value of \( h + 1 \):

\[ \delta(h+1) = \sum_{W \in \Lambda} r(S, W) + \sum_{W \in \Lambda} \sum_{W' \subseteq [h+2, k-1]} \frac{r(S, W)}{\prod_{i \in W} g_i} + \sum_{W \in \Lambda} \sum_{W' \subseteq [h+2, k-1]} \frac{r(S, W)}{\prod_{i \in W \setminus [0, h+1]} g_i} + \sum_{W \in \Lambda} \sum_{W' \subseteq [h+2, k-1]} \frac{r(S, W)}{\prod_{i \in W \setminus [0, h+1]} g_i}. \]
Now consider \( \delta^{(h)} \) as defined in Equation 5.6:

\[
\delta^{(h)} = \sum_{W \subseteq [0,h]} r(S, W) + \sum_{W \subseteq [h+1,k-1]} \frac{r(S, W)}{\prod_{i \in W} g_i} + \sum_{W \subseteq [h+1,k-1]} \frac{r(S, W)}{\prod_{i \in W \setminus [0,h]} g_i}
\]

\[
= \sum_{W \subseteq [0,h]} r(S, W) + \left( \sum_{W \subseteq [h+1,k-1]} \frac{r(S, W)}{\prod_{i \in W} g_i} + \sum_{W \subseteq [h+1,k-1]} \frac{r(S, W)}{\prod_{i \in W \setminus [0,h]} g_i} \right)
\]

\[
+ \left( \sum_{W \subseteq [h+2,k-1]} \frac{r(S, W)}{\prod_{i \in W} g_i} + \sum_{W \subseteq [h+2,k-1]} \frac{r(S, W)}{\prod_{i \in W \setminus [0,h]} g_i} \right)
\]

By eliminating same terms from \( \delta^{(h)} \) and \( \delta^{(h+1)} \), to show \( \delta^{(h+1)} \geq \delta^{(h)} \), noting that the right hand side can be grouped as follows, it suffices to show:

\[
\sum_{W \subseteq A} \frac{r(S', W)}{\prod_{i \in W \setminus [0,h+1]} g_i} \geq \sum_{W \subseteq A} \frac{r(S, W)}{\prod_{i \in W} g_i} + \sum_{W \subseteq A} \frac{r(S, W)}{\prod_{i \in W \setminus [0,h]} g_i} = \sum_{W \subseteq A} \frac{r(S, W)}{\prod_{i \in W \setminus [0,h]} g_i}.
\]

As per the description of our algorithm, using Equations 5.1 and 5.3 and Proposition 5.1.4, we can rewrite the right hand side as follows:

\[
\sum_{W \subseteq A} \frac{r(S', W)}{\prod_{i \in W \setminus [0,h+1]} g_i} \geq \frac{1}{g_{h+1}} \sum_{\sigma \subseteq [g_{h+1}]} \left( \sum_{W \subseteq A} \frac{r(S \cup \{(h + 1, \sigma)\}, W)}{\prod_{i \in W \setminus [0,h+1]} g_i} \right).
\]

According to the algorithm, we extended \( S \) to \( S' \) by selecting \( \sigma_{h+1} \) to maximize:

\[
\sum_{W \subseteq A} \frac{r(S \cup \{(h + 1, \sigma_{h+1})\}, W)}{\prod_{i \in W \setminus [0,h+1]} g_i}
\]

so we have that the inequality holds, as required. □
We use the following lemma to simplify asymptotic analysis.

**Lemma 5.2.2.** For \( m > 1 \), \( (\ln \frac{m}{m-1})^{-1} < m \).

**Proof.** This is a consequence of the Taylor expansion of \( (\ln \frac{1}{1-x})^{-1} \) with \( x = 1/m \), which gives:

\[
\left( \ln \frac{m}{m-1} \right)^{-1} = \left( \sum_{i \geq 1} \frac{1}{im^i} \right)^{-1} < \left( \frac{1}{m} \right)^{-1} = m.
\]

\( \square \)

**Theorem 5.2.3.** Let \( \Lambda \) be an ASC over \( k \) factors with \( g_1, \ldots, g_k \) levels respectively. Take \( g = \max\{g_i : i \in \{1, \ldots, k\}\} \), \( s = \max\{|W| : W \in \Lambda\} \), and \( m = \max\{\prod_{i \in W} g_i : W \in \Lambda\} \). Then Algorithm \( \square \) returns a VCA(\( N; \Lambda, (g_1, \ldots, g_k) \)) where:

\[
N \leq \frac{\ln |\Lambda| + \ln m}{\ln \frac{m}{m-1}} \\
\leq m(\ln |\Lambda| + \ln m) \\
\leq g^s(\ln |\Lambda| + s \ln g).
\]

**Proof.** The total initial number of interactions that need to be covered is:

\[
n_0 = \sum_{W \in \Lambda} \prod_{i \in W} g_i \leq \sum_{W \in \Lambda} m \leq |\Lambda|m.
\]

Assume we have some partial array \( T \) of size \( i \) that we have built so far. Let \( n_i \) be the remaining number of interactions to be covered. The addition of a row of density \( \delta' \geq \delta \) reduces the number of uncovered interactions by at least \( \delta \), where we have that:

\[
\delta = \sum_{W \in \Lambda} \frac{r(Z, W)}{\prod_{i \in W} g_i} \geq \frac{1}{m} \sum_{W \in \Lambda} r(Z, W) = \frac{n_i}{m}.
\]

Thus, the number of uncovered interactions is reduced to at most \( n_i \left(1 - \frac{1}{m}\right) \). After \( i \) rows have been selected, we thus have that \( n_i \leq n_0 \left(1 - \frac{1}{m}\right)^i \). All interactions are covered and the algorithm halts when \( n_i < 1 \). Let \( N \) be the number of iterations performed by the algorithm (and thus the number of test cases produced). We claim that:

\[
N \leq -\frac{\ln n_0}{\ln(1 - \frac{1}{m})} + 1.
\]
Indeed, for any $i > -\frac{\ln n_0}{\ln(1 - \frac{1}{m})}$, we have that $n_i \leq n_0(1 - \frac{1}{m})^i < n_0(1 - \frac{1}{m})^{-\frac{\ln n_0}{\ln(1 - \frac{1}{m})}}$, which gives that $\ln n_i < \ln n_0(1 - \frac{1}{m})^{-\ln(1 - \frac{1}{m})} = 0$, thus implying $n_i < 1$. In this case, the algorithm would have terminated at the end of iteration $i$ or earlier. Thus:

$$N \leq \frac{\ln n_0}{-\ln(1 - \frac{1}{m})} + 1 \leq \frac{\ln |\Lambda| + \ln m}{\ln \frac{m-1}{m}} + 1 \leq m(\ln |\Lambda| + \ln m) + 1 \quad \text{(Lemma 5.2.2.)}$$

Therefore, $N \leq m(\ln |\Lambda| + \ln m)$, and from the second line, we have that $N \leq \frac{\ln |\Lambda| + \ln m}{\ln \frac{m-1}{m}}$. The fact that $m \leq v^s$ gives the rest of the inequality in the proof.

### 5.2.1 Fine-tuning the Logarithmic Bound on Graphs Using Homomorphisms

We have the following connection between homomorphisms between graphs and covering arrays over them:

**Lemma 5.2.4.** [67, 68] Let $G, G_U$ be graphs and $g > 1$ an integer. If we have a homomorphism $G \rightarrow G_U$, then $\text{CAN}(G, g) \leq \text{CAN}(G_U, g)$. Thus, in particular, if we have a graph $G$ with chromatic number $\chi(G)$, then for all $g > 1$, $\text{CAN}(G, g) \leq \text{CAN}(K_{\chi(G)}, g)$.

We use this fact to contrast two different strategies for a finding VCA over a graph $G$ with uniform levels $g$:

1. Directly use VCADGBA to find a $\text{CA}(N; G, g)$. By Theorem 5.2.3, this gives an upper bound of $N \leq g^2(\ln |E(G)| + \ln g^2)$.

2. Find a $c$-colouring of $G$, which gives a homomorphism $G \rightarrow K_c$, and then use **VARDENS** to find a $\text{CA}(N; K_c, g)$. Then this gives an upper bound of $N < g^2(\binom{c}{2} + \ln g^2)$.

Thus, the first strategy works better whenever $|E(G)| < \binom{c}{2}$; however, if this happens, then the homomorphic image of $G$ under the homomorphism $f: G \rightarrow K_c$ is not edge-surjective, i.e. some edge $e = ij \in E(K_c)$ has no inverse image in $G$. Then there are no edges between any of the vertices $f^{-1}(i)$ and $f^{-1}(j)$, so we can combine these into a single colour class and derive a $c - 1$ colouring. By iterating this, we can always derive
a $c$-colouring such that $|E(G)| \geq \binom{c}{2}$ necessarily, and thus the second technique always works better. Thus, we suggest the following algorithm:

1. Obtain a $c$-colouring of $G$ such that $|E(G)| \geq \binom{c}{2}$. Note, of course, that the smaller the value of $c$ the better.

2. Invoke VarDens to find a VCA($K_c, g$), say $M$.

3. Construct the VCA($G, g$), say $M'$, by pulling the columns of $M$ back through the colouring, i.e. if $i$ maps to colour class $c_i$, then column $i$ in $M'$ becomes the column corresponding to $c_i$ in $M$.

Let $d = \max\{\deg(v) : v \in V(G)\}$: while determining chromatic number is a hard problem, there is a polynomial time greedy colouring algorithm that guarantees a $d + 1$ colouring of $G$, and there are graphs where $\chi(G) = d + 1$. Additionally, the above algorithm is generalizable to arbitrary homomorphisms $G \rightarrow H$, and if $|E(H)| < |E(G)|$, the bound by using VarDens through the homomorphism is better than the bound given by applying VarDens directly on the graph.

### 5.3 Algorithm Variations and Experimental Design

In this section, we discuss variations of the VarDens algorithm that are used in our experiments.

Making full density calculations, while still polynomially bounded, is computationally intensive, even for fairly small problem sets. Bryce et al. [11] introduce the concept of using an $\ell$-restricted version of their density algorithm for finding covering arrays with fixed strength $t > 2$. In computing the density for a factor $f$, there are $\binom{k-1}{t-1}$ sets of factors contributing to the density of $f$. In each of these $t$-sets of factors, the factors may be fixed or free: uncovered interactions over fixed factors are guaranteed to be covered in the completed row, while contributions to density from free factors are only a predictor surrogate regarding interaction coverage. Bryce et al. propose that it may be worthwhile to focus on guaranteed interaction coverage, and this is done by introducing a modification of Algorithm 1. A set of parameters $S$ is $\ell$-restricted if it contains at most $\ell$ free factors. In the $\ell$-restricted algorithm variant, when performing density calculations for a factor $f$, only $\ell$-restricted sets of factors containing $f$ are considered in the calculation for $f$. This has the effect of significantly reducing computation time, but at the expense of the logarithmic guarantee. Their experimental results indicate that $\ell = 2$ demonstrates
substantial time savings over the unrestricted case, which we denote $\ell = \infty$, while generating arrays of comparable size. We investigate the effect of $\ell \in \{1, 2, \infty\}$ with regards to various families of ASC in our experiments: note that $\ell = 1$ mimics the behaviour of the AETG method [20, 19], while $\ell = \infty$ comprises the full density algorithm.

Imposing the $\ell$-restriction necessitates fixing some set of initial parameters in order to have a nonempty collection of facets with only $\ell$ free factors to use in density calculations; otherwise, the density of all parameters is 0, and the choice of which parameter to fix deteriorates to random selection. Our second design pattern involves one of the following initial fixing strategies (IFS):

1. **Largest Facet (LF)**: Consider the set of facets with uncovered interactions. Amongst the largest facets of this set, pick a facet with the most uncovered interactions, breaking ties at random. Select an uncovered interaction over this facet at random and fix it.

2. **Most Uncovered (MU)**: Consider the set of facets with uncovered interactions. Amongst the facets with the most uncovered interactions, pick a largest facet of this set, breaking ties at random. Select an uncovered interaction over this facet at random and fix it.

3. **Any Uncovered (AU)**: Consider the set of all uncovered interactions over all facets. Pick one with uniform probability and fix it.

4. **s-Random (sR)**: For an $s > 0$, randomly pick $s$ factors and fix them to randomly selected levels.

5. **s-Interaction (sS)**: For an $s > 0$, pick the $s$-interaction that appears in the most uncovered interactions over the facets of the simplicial complex. (The idea here is that such a choice would be a good starter interaction to extend to uncovered interactions over many facets.)

We note that when the facet size in the ASC is uniform, LF and MU are equivalent. After initial factors are fixed, we proceed regularly with Algorithm 1 and fix the remaining free factors one at a time, selecting the factor $f$ of largest factor density. We then select the level $\sigma$ for $f$ of highest density, and fix $f$ to $\sigma$ in our test case interaction $S$. In both cases, ties are broken randomly.

We allow two additional control parameters, $M$ and $C$: at each stage, VARDENS builds $M$ row candidates, and then adds the row among them that covers the most
previously uncovered interactions to the array. We terminate when no remaining interactions are uncovered, and the collection of rows generated is our VCA. We repeat the process $C$ times to generate $C$ different arrays, and the one with the smallest number of rows can be used. The random tie-breaking is what causes different row candidates to be chosen, resulting in different arrays. Unless otherwise stated, we take $M = 3$ and $C = 10$.

5.4 Experimental Results

We now proceed to compare the performance of VarDens against previously studied problems, and then apply VarDens to several different families of ASC and analyze the results. Our algorithm was implemented in Python 2.6 and executed on Dell PowerEdge 2950 servers with dual quad-core Intel E5355 CPU.

5.4.1 Comparison to Density-Based Algorithm for Covering Arrays

Given that VarDens is a generalization of the ideas of Bryce and Colbourn [11], as a partial test of correctness for our algorithm, we duplicated their experiments for the cases $t = 4$ and $g \in \{2, 3, 4, 5\}$. In all cases, our results were comparable, as expected, with randomness accountable for small deviations.

In one experiment, they demonstrate their results in generating CA($4, k, g$) for $k \in \{5, \ldots, 10\}$ and $g \in \{2, \ldots, 5\}$: for each $\ell \in \{1, 2, 3, \infty\}$, they use 2S and perform three trials, reporting the best result. Running VarDens with the ASC with facet set $\Lambda = \binom{\{0,\ldots,k-1\}}{4}$ solves the equivalent problem, and should produce comparably sized arrays. We note that for uniform strength $t$, as in this case, the initial fixing methods LF and MU are equivalent. For each $\ell \in \{1, 2, \infty\}$, we apply LF, 2R, and 2S and perform three trials, reporting the best result. In Table 5.1, we conglomerate and compare the results of VarDens to the results of their algorithm, which, in the case of 2S, should be comparable. This proves to be the case, as indicated in the best row for 2S: the mean ratio between the size of our arrays and theirs in the best case is 1.0082, with a median of 1.0156.

We also investigate the effect of varying $\ell$ within VarDens: for each of the initial fixing methods, for $\ell \in \{1, 2, \infty\}$, we record the best array size reported by VarDens for each test. In Table 5.2, we compare the ratios of successively larger values of $\ell$ to de-
Table 5.1: Comparison between VarDens and Bryce and Colbourn’s algorithm [11], where ratios are taken to be the size of arrays from VarDens to the size of arrays generated by their algorithm.

termine if the general trend is towards improvement, and if so, the level of improvement. As we can see, for LF, the improvement moving from $\ell = 1$ to $\ell = 2$ is negligible, and moving from $\ell = 2$ to $\ell = \infty$ is similarly so. For 2R, however, the improvement when moving from $\ell = 1$ to $\ell = 2$ is much more substantial. There is much less improvement when moving from $\ell = 2$ to $\ell = \infty$. For 2S, minimal improvement is seen when moving from $\ell = 1$ to $\ell = 2$, but increasing $\ell = 2$ to $\ell = \infty$ in these experiments resulted in, over half the time, actually worse results with a larger array. Thus, it may not always be advantageous to increase $\ell$, especially when considered with the increase in computation time for higher $\ell$, as we will see in future experiments.

### 5.4.2 Comparison with Simulated Annealing

Cohen et al. [22] used simulated annealing to generate VCA over a special family of ASC, where the factors have a base strength uniformly applied over them, and then disjoint sets of factors may have higher strengths applied to them. The ASC that they used and the resulting VCA sizes are given in Table 5.3. For convenience, we represent ASC over
{0, \ldots, k-1} as follows:

\[(t, (t_0, S_0), (t_1, S_1), \ldots, (t_{m-1}, S_{m-1}))\]

where \( t \geq 0 \), \( S_i \subseteq \{0, \ldots, k-1\} \), and \( t < t_i \leq |S_i| \), to indicate that:

\[
\left( \{0, \ldots, k-1\} \atop t \right) \cup \left( S_0 \atop t_0 \right) \cup \left( S_1 \atop t_1 \right) \cup \ldots \cup \left( S_{m-1} \atop t_{m-1} \right) \subseteq \Delta
\]

and we add any subsets of the above as required to make \( \Delta \) an ASC. For a set \( S \), we write \( S = [i,j] = \{i, i+1, \ldots, j\} \) for compactness.

We run our algorithm on their data for \( \ell \in \{1, 2, \infty\} \) and one of LF, MU, 2R, and 2S. The results for the array sizes obtained by VarDens with different IFS and simulated annealing, for each of the problems in Table 5.3, are given in Figures 5.2-5.5. The running time for different variations of VarDens is given in Figure 5.6, where each test is divided into three “columns”, with the entries in the first representing \( \ell = 1 \) times, the second \( \ell = 2 \), and the third \( \ell = \infty \).

We note that the simulated annealing arrays are usually smaller than those generated by VarDens, which is to be expected, as simulated annealing is computationally more expensive than our greedy heuristic. For the LF and MU techniques, there is little advantage to using \( \ell = 2 \) over \( \ell = 1 \), and the execution time increases significantly with \( \ell \). In general, we can see that increasing \( \ell \) from 1 to 2 produces slightly better arrays at an increase in computation time, but increasing \( \ell \) further has little impact on array quality but generally requires more computation. Overall, the effects seen from varying \( \ell \) are not dramatic, but this is due to the fact that the facets all have small size (either 2 or 3) to begin with; more significant benefits of the different \( \ell \)-restrictions will be seen below.

<table>
<thead>
<tr>
<th>IFS</th>
<th>( \ell_1 : \ell_2 )</th>
<th>min</th>
<th>max</th>
<th>mean</th>
<th>mdn</th>
</tr>
</thead>
<tbody>
<tr>
<td>LF</td>
<td>2 : 1</td>
<td>0.9268</td>
<td>1.0625</td>
<td>0.9984</td>
<td>0.9856</td>
</tr>
<tr>
<td></td>
<td>\infty : 2</td>
<td>0.8889</td>
<td>1.0526</td>
<td>0.9911</td>
<td>0.9968</td>
</tr>
<tr>
<td>2R</td>
<td>2 : 1</td>
<td>0.7809</td>
<td>1.0000</td>
<td>0.8894</td>
<td>0.8847</td>
</tr>
<tr>
<td></td>
<td>\infty : 2</td>
<td>0.9336</td>
<td>1.0588</td>
<td>0.9838</td>
<td>0.9849</td>
</tr>
<tr>
<td>2S</td>
<td>2 : 1</td>
<td>0.9353</td>
<td>1.2963</td>
<td>0.9894</td>
<td>0.9686</td>
</tr>
<tr>
<td></td>
<td>\infty : 2</td>
<td>0.8889</td>
<td>1.1154</td>
<td>0.9997</td>
<td>1.0034</td>
</tr>
</tbody>
</table>

Table 5.2: The effect of varying \( \ell \) in VarDens.
Figure 5.2: Size of arrays generated by LF algorithm variation.
Figure 5.3: Size of arrays generated by MU algorithm variation.
Figure 5.4: Size of arrays generated by 2R algorithm variation.
Figure 5.5: Size of arrays generated by 2S algorithm variation.
Figure 5.6: Time required by algorithm variations.
### Table 5.3: Types of VCAs solved, where the value in the table is substituted for Λ. $M_C$ is the smallest array size found by simulated annealing in [22], and $M_V$ is the smallest array size found by VCADGBA.

<table>
<thead>
<tr>
<th>$n$</th>
<th>VCA($N; Λ, 3$), $k = 15$</th>
<th>$M_C$</th>
<th>$M_V$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(2, $\emptyset$)</td>
<td>16</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>(2, (3, [0, 2]))</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>3</td>
<td>(2, (3, [0, 2]), (3, [3, 5]))</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>(2, (3, [0, 2]), (3, [3, 5]), (3, [6, 8]))</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>5</td>
<td>(2, (3, [0, 3]))</td>
<td>27</td>
<td>28</td>
</tr>
<tr>
<td>6</td>
<td>(2, (3, [0, 4]))</td>
<td>33</td>
<td>38</td>
</tr>
<tr>
<td>7</td>
<td>(2, (3, [0, 3]), (3, [4, 8]), (3, [9, 14]))</td>
<td>33</td>
<td>43</td>
</tr>
<tr>
<td>8</td>
<td>(2, (3, [0, 5]))</td>
<td>33</td>
<td>43</td>
</tr>
<tr>
<td>9</td>
<td>(2, (3, [0, 6]))</td>
<td>41</td>
<td>50</td>
</tr>
<tr>
<td>10</td>
<td>(2, (3, [0, 8]))</td>
<td>50</td>
<td>57</td>
</tr>
<tr>
<td>11</td>
<td>(2, (3, [0, 14]))</td>
<td>67</td>
<td>75</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$n$</th>
<th>VCA($N; Λ, (4, 4, 4, 5, 5, 6, 6)$)</th>
<th>$M_C$</th>
<th>$M_V$</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>(2, $\emptyset$)</td>
<td>36</td>
<td>38</td>
</tr>
<tr>
<td>13</td>
<td>(2, (3, [0, 2]))</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>14</td>
<td>(2, (3, [0, 4]))</td>
<td>100</td>
<td>121</td>
</tr>
<tr>
<td>15</td>
<td>(2, (3, [3, 5]))</td>
<td>125</td>
<td>125</td>
</tr>
<tr>
<td>16</td>
<td>(2, (3, [0, 2]), (3, [3, 5]))</td>
<td>125</td>
<td>125</td>
</tr>
<tr>
<td>17</td>
<td>(2, (3, [0, 6]))</td>
<td>171</td>
<td>199</td>
</tr>
<tr>
<td>18</td>
<td>(2, (3, [5, 7]))</td>
<td>180</td>
<td>180</td>
</tr>
<tr>
<td>19</td>
<td>(2, (3, [0, 7]))</td>
<td>214</td>
<td>246</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$n$</th>
<th>VCA($N; Λ, (3, \ldots, 3, 10, 10)$), $k = 22$</th>
<th>$M_C$</th>
<th>$M_V$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>(2, $\emptyset$)</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>21</td>
<td>(2, (3, [0, 19]))</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

#### 5.4.3 Families of Abstract Simplicial Complexes

We now look at the experimental results, in terms of time and array size, of running **VarDens** with various parameters to produce covering arrays over certain families of ASC.
Steiner triple systems

One family of hypergraphs of particular interest are Steiner triple systems, as they are the smallest 3-uniform set systems which have the property that for any pair of points, there is a 3-set containing the pair. Thus, a covering array over an STS($k$) is a strength-2 covering array over $k$ points with the property that strength-3 coverage is guaranteed over some factors.

Complete isomorph-free lists of STS for a fixed $k$ have been generated for $k \leq 19$: such a list has the property that every STS($k$) is isomorphic to some STS in our list, and no two STS in our list are isomorphic. For $k = 7, 9$, the list has size 1; for $k = 13$, the list has size 2; and for $k = 15$, there are 80 STS in the isomorph-free list. These numbers and the isomorph-free lists of designs themselves can be found in [33].

Taking the blocks of an STS to be facets of an ASC, for a given $k$, we ran VarDens 20 times on each isomorph-free list for $k \in \{7, 9, 13, 15\}$ and $g \in \{2, 3, 5\}$. The results for $k = 15$ and $g \in \{3, 5\}$ can be found in Table 5.4: if $S_k$ is our isomorph-free list of STS of order $k$, we let $m(S, g, T, \ell)$ and $a(S, g, T, \ell)$ be the size of the smallest array and average array respectively over $S \in S_k$ with $g$ levels, obtained by running the $\ell$-restricted VarDens with initial fixing method $T$.

- $N_m = \min \{m(S, g, T, \ell) : S \in S_k\}$, i.e. the smallest array found across all STS of order $k$ for the given algorithm parameters.
- $N_M = \max \{m(S, g, T, \ell) : S \in S_k\}$, i.e. taking a list of all the smallest arrays found for each STS, $N_M$ is the size of the largest array in the list.
- $A_m = \frac{\sum_{S \in S_k} m(S, g, T, \ell)}{|S_k|}$, i.e. the average size of smallest array.
- $A_a = \frac{\sum_{S \in S_k} a(S, g, T, \ell)}{|S_k|}$, i.e. the overall average array size (the average size of the average array sizes over all STS of order $k$).

In Table 5.4, for a given set of parameters, $T_m$ represents the smallest execution time required by any one STS($k$) for VarDens to run 20 iterations, $T_M$ the largest execution time, and $T_a$ the average, all measured in seconds. We note that in all cases, there is significant improvement seen between $\ell = 1$ and $\ell = 2$, but little improvement when $\ell = \infty$, despite a significant increase in run-time; this reflects the observation by Bryce and Colbourn in [11] where it was suggested that, for practical purposes, the 2-restricted algorithm provided the best trade-off in terms of results and time. With the exception of 2R, the choice of initial fixing method did not have a significant impact on resultant array
size. In terms of execution time, LF, MU, and AU were all comparable, and performed faster than 2R and 2S.

In Table 5.5, for an order $k$, we list the number of isomorph-free STS (#), and then if $B \in S_k$, we have that VarDens was able to find a VCA($N; B, g$), where $N_m \leq N \leq N_M$. Taking the smallest array size found by VarDens for every $S \in S_k$ across all techniques and executions of VarDens, we list the average in column $A$. The small variance in size of the minimal arrays generated and the time required by VarDens suggest that size of array and time have little dependence on specific choice of STS($k$).

Open Question 5.4.1. The Steiner triple system study of Section 5.4.3 suggests that the sizes of covering arrays over STS($k$) may depend only on $k$ and $g$, and could possibly be independent of specific choice of the actual Steiner triple system. Whether or not this is the case is an open question. It also merits investigation to see if this behaviour extends to other families of ASC, such as Steiner quadruple systems.

In support of this open question, for the 80 STS(15) as given in [33] where we write $S_n$ for $1 \leq n \leq 80$ to represent the $n$th STS(15), using a great deluge algorithm to try to find optimal VCA over these STS, we have found:

1. For $g = 2$, if $n \in \{9, 10, 19, 36, 53, 63, 66, 69\}$, $\text{VCAN}(S_n, 2) \leq 11$. Otherwise, $\text{VCAN}(S_n, 2) \leq 12$.

2. For $g = 3$, $\text{VCA}(S_n, 3) \leq 39$ for:

   $n \in \{2, 8, 10, 20, 22, 23, 26, 32, 34, 39, 45, 46, 48, 51, 54, 64, 65, 68, 69, 74, 76, 80\}$.

   Otherwise, $\text{VCAN}(S_n, 3) \leq 38$.

Hypertrees

Cheng showed how to construct VCA over hypertrees of provably optimal size in [17]:

**Theorem 5.4.1.** Let $H$ be a hypertree over $k$ factors, and for a factor $f \in V(H)$, let $g_f$ be the number of levels over $f$. Then:

$$\text{VCAN}(H, (g_0, \ldots, g_{k-1})) = \max\left\{ \prod_{f \in e} g_f : e \in H \right\}.$$
Table 5.4: The results of running VarDens over the 80 STS(15).
Table 5.5: A summary of the range of sizes of VCA($\mathcal{B}, g$) for $\mathcal{B} \in \mathcal{S}_k$ in comparison with the best known sizes of CA(2, $k, g$) and CA(3, $k, g$).

<table>
<thead>
<tr>
<th>$k$</th>
<th>$#$</th>
<th>$g$</th>
<th>CA(2, $k, g$)</th>
<th>$N_m$</th>
<th>$N_M$</th>
<th>$A$</th>
<th>CA(3, $k, g$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>8.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>12</td>
<td>31</td>
<td>31</td>
<td>31.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>29</td>
<td>143</td>
<td>143</td>
<td>143.00</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>2</td>
<td>7</td>
<td>10</td>
<td>10</td>
<td>10.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>15</td>
<td>35</td>
<td>35</td>
<td>35.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>38</td>
<td>154</td>
<td>154</td>
<td>154.00</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>12</td>
<td>13</td>
<td>12.50</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>17</td>
<td>40</td>
<td>40</td>
<td>40.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>38</td>
<td>171</td>
<td>171</td>
<td>171.00</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>80</td>
<td>2</td>
<td>8</td>
<td>13</td>
<td>14</td>
<td>13.04</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>19</td>
<td>41</td>
<td>43</td>
<td>42.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>45</td>
<td>178</td>
<td>180</td>
<td>179.45</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.7: Three hypertrees $T_1$, $T_2$, and $T_3$ used in experiments. The factors in $T_1$ and $T_2$ have uniform levels taken from $\{2, 3, 5\}$. The factors of $T_3$ have variable numbers of levels, specified in bold.

This array size must be optimal, as the hyperedge that yields the maximum value requires this many interactions to be covered.

We investigate the performance of VarDens on the three hypertrees given in Figure 5.7. We note that for both $T_1$ and $T_2$, the optimal array size is $g^4$, i.e., 16 for $g = 2$, 81 for $g = 3$, and 625 for $g = 5$. For $T_3$, the optimal size is $2^6 = 4^3 = 64$. The results are demonstrated in Figures 5.8, 5.9, 5.10, 5.11, 5.12.
Figure 5.8: Array sizes and times for $T_1$ with $g = 3$ using LF, MU, AU, 2R, 3R, 2S, and 3S. The entries in each bar from left-to-right represent $\ell = 1$, $\ell = 2$, and $\ell = \infty$. 

VarDens: A Density-Based Greedy Algorithm for VCA
Figure 5.9: Array sizes and times for $T_1$ with $g = 5$ using LF, MU, AU, 2R, 3R, 2S, and 3S. The entries in each bar from left-to-right represent $\ell = 1$, $\ell = 2$, and $\ell = \infty$. 
Figure 5.10: Array sizes and times for $T_2$ with $g = 3$ using LF, MU, AU, 3R, and 3S. The entries in each bar from left-to-right represent $\ell = 1$, $\ell = 2$, and $\ell = \infty$. 
Figure 5.11: Array sizes and times for $T_2$ with $g = 5$ using LF, MU, AU, 3R, and 3S. The entries in each bar from left-to-right represent $\ell = 1, \ell = 2$, and $\ell = \infty$. 
Figure 5.12: Array sizes and times for $T_3$ using LF, MU, AU, sR, and sS for $s \in \{2, \ldots, 5\}$. The entries in each bar from left-to-right represent $\ell = 1$, $\ell = 2$, and $\ell = \infty$. 
For $T_1$, it is not immediately clear from the ASC what are the best values to select for $s$ when using the $sR$ and $sS$ IFS: 3S gives the best results with respect to array size of any IFS we tried, but the computation time is dramatically higher than the other techniques. For $g = 5$, 3S requires between 6.61 to 84.41 times as long to run, with a mean of 35.33 and a median of 29.65 times. LF and MU both perform comparably to one another, and nearly as well as 3S, but require far less execution time. When increasing from $\ell = 1$ to $\ell = 2$, the ratio of array sizes for $g = 5$ is between 0.7448 and 0.9677, with a mean of 0.8702 and a median of 0.9164. This requires between 1.61 and 2.39 times longer to run, with a mean of 2.01 and a median of 2.04. When increasing from $\ell = 2$ to $\ell = \infty$, every array generated for an IFS other than $sR$ is optimal: this represents a ratio of array sizes of between 0.9268 and 0.9933, with a mean of 0.9663 and a median of 0.9912, and requires between 2.00 and 4.79 times longer to run, with a mean of 2.90 and a median of 2.45. This supports the experiments of Bryce et al. [11] that suggest that in terms of array size versus execution time trade-off, $\ell = 2$ is a good choice.

For $T_2$, we note that LF and MU amount to the same technique due to the facet sizes being uniform. Here, the obvious choice for $s$ is 3 (i.e. 1 less than the uniform facet size) when using the $sR$ and $sS$ IFS. Again, we note that 3S gives the best results in terms of array size, but takes significantly longer than the other techniques to the point of being nearly infeasible, requiring between 195.64 to 630.16 times longer than the other IFS to run. None of the IFS produce an optimal array unless $\ell = \infty$, and while increasing $\ell$ from 1 to 2 yields significantly better matrices (ratio of 0.6729 to 0.8401 with mean 0.8052 and median 0.8385, requiring between 1.43 and 1.51 times longer to run), increasing $\ell$ from 2 to $\infty$ provides significantly less benefit (ratio of 0.9126 to 0.9569 with mean 0.9466 and median 0.9548) and a dramatic increase in run time (11.47 to 12.86 times longer).

Due to the high variability in facet size in $T_3$, as with $T_1$ it is not clear what values to select for $s$ for the IFS $sR$ and $sS$, so we try all $s \in \{2, \ldots, 5\}$. Here, 3S performs better than the other $sS$ and $sR$ IFS, but is outperformed slightly by LF, MU, and AU, regardless of value for $\ell$. Furthermore, LF, MU, and AU all achieve optimal arrays for $\ell = 1$.

In all cases, the initial fixing techniques LF, MU, and AU, despite their more computationally intensive nature in selecting the initial fixings to perform, do significantly better than the $sR$ family of results for low $\ell$, both in terms of time and array size, and for higher $\ell$, comparable in terms of both array size and execution time. The $sS$ family does well in terms of array size but due to its computationally intensive nature, requires considerably higher execution time, especially as $s$ increases from 2 to 3. It is unclear in
the context of hypertrees with nonuniform facets what choice of \( s \) yields the best results, but due to the long run time required, it is difficult to obtain enough data to draw any conclusions about \( s \). The shorter run time and comparable results for other IFS suggest that the simpler techniques are superior in the general case.

5.4.4 Experimental Analysis and Possible Enhancements

In all experiments, the three techniques LF, MU, and AU all generated arrays of comparable size, and, especially for low \( \ell \), of a smaller size than the sR and sS technique families, suggesting that simple is best in this case. Furthermore, these techniques do not require the selection of the parameter \( s \), whose value is non-obvious when the facets of the ASC are not uniform.

In the cases where optimal arrays are easily constructed (e.g. hypertrees) or have a lower bound based on the number of interactions over a large facet, VARDENS was able to find the optimal covering array size. In cases where the range of sizes of facets is much tighter and the facets more dense over the parameters, VARDENS did not perform as well as simulated annealing for known results, as expected.

One concern with using the \( \ell \)-restricted approach lies in the fact that for the general problem, the facets in an ASC can differ considerably in size. By using a straightforward \( \ell \)-restriction, during row generation, we note that the restriction has much more of an impact on large facets as they are excluded from consideration until many of their parameters have already been fixed. Smaller facets are likely to contribute to density much sooner, as they are considered when far fewer of their parameters are fixed. This is contrary to our intuition on which facets should receive priority: number of levels notwithstanding, there are generally many more interactions to cover over large facets than over smaller ones, and thus, large facets should be given a greater priority in computing densities, and not a reduced one as is done in the \( \ell \)-restricted algorithm. We note that we attempted to find an ASC where this was problematic by experimenting with several ASC with high variability in facet size, but in all experiments, we were able to achieve the lower bound on the problem dictated by the largest facet: thus, whether or not any ASC exist where this proves to be an issue remains an open question. If an ASC with facet size variability causes the algorithm with fixed \( \ell \) to perform poorly, the following variation could be attempted:

\( \ell \)-density-restricted algorithm: Let \( \mathcal{F} \) be the set of free factors, i.e. the factors not yet assigned a level during the creation of a test case. For \( \ell \in (0, 1] \), a facet \( W \) is said to
be \( \ell \)-density-restricted if \( \frac{|W \cap F|}{|W|} \leq \ell \). When calculating factor density, \( W \) contributes to the density of its free factors if and only if it is \( \ell \)-density-restricted.

For larger facets, more free factors will be allowed than for smaller facets, and thus they will come into play earlier in density calculations. However, considering facets with a large number of free factors is computationally intensive, so this approach should be used with caution.

**Open Question 5.4.2.** When the maximum and minimum facet sizes of an ASC differ considerably, the VarDens algorithm may make poor choices in its execution, especially when there are nonempty intersections between large and small facets. What strategies can be used to avoid this?

**Open Question 5.4.3.** From a practical point of view, it would be interesting to investigate the structure of ASC arising in different software testing applications. Any special structures of interest might motivate the direction of algorithms over ASC and be exploited to generate minimal array sizes.
Chapter 6

The Lovász Local Lemma and Probabilistic Bounds

The Lovász Local Lemma is a versatile, useful tool that can be used to develop probabilistic arguments for determining the existence of combinatorial objects. These proofs are nonconstructive in that they make guarantees about the existence of an object without having to constructively produce them. In the following chapter, we use several different variants of the Local Lemma to determine upper bounds on the size of variable strength covering arrays over a number of different hypergraphs.

In Section 6.1, we begin by deriving a framework that uses the symmetric Local Lemma to derive bounds for VCA over arbitrary hypergraphs, with the result given in Lemma 6.1.9.

In Section 6.1.1 we show that this framework finds better bounds for VCA over all $s-(k,t,\lambda)$ designs than those guaranteed by $\text{VarDens}$, as given in Chapter 5. Based on the nature of the hypergraphs, in Theorems 6.1.12 and 6.1.14 we are able to give precise results for Steiner systems of the form $S(t-1,t,k)$, and more general results for arbitrary $s-(k,t,\lambda)$ designs in Theorems 6.1.16 and 6.1.17.

We then examine the application of the symmetric Local Lemma to finding consecutive and cyclic consecutive covering arrays in Section 6.1.2, and show in Theorems 6.1.20 and 6.1.21 that the bound generated by the Local Lemma, which is independent of the number of factors, is dramatically better than the size guaranteed by the $\text{VarDens}$ bound, which grows logarithmically with the number of factors. In practice, however, the actual size of the consecutive and cyclic consecutive covering arrays generated by $\text{VarDens}$ is independent of the number of factors. Furthermore, we show that the Lo-
cal Lemma bound demonstrates some insight into the homomorphic properties of $H_{c}^{k,t}$.

This, however, is not the case with triangulation hypergraphs of the sphere, as discussed in Section 6.2 we know that the optimal array size is independent of the hypergraph and is only dependent on $g$, the number of levels, but neither the Local Lemma nor VARDENS are able to recognize this fact in theory, and our experiments suggest that the arrays generated by VARDENS grow with the size of the triangulation hypergraph.

In Section 6.3 we compare the bound from VARDENS to the bound from the symmetric Local Lemma in the case where apart from the number of facets, we have no knowledge of the structure of the hypergraph, and we show that these bounds are almost equal.

Finally, in Section 6.4 we consider and analyze some cases where applying forms other than the symmetric form of the Local Lemma yield significantly better results than would be obtained through the symmetric form, and derive some techniques for applying the general Local Lemma and its variants to finding bounds.

### 6.1 The Local Lemma and the Probabilistic Method

In [43], Erdős and Lovász investigate several problems on the colourability of hypergraphs, and to determine upper bounds on the chromaticity of hypergraphs with bounded degree, they developed a probabilistic argument that has come to be known as the Lovász Local Lemma. In general, if we have a number of independent “bad” events that all occur each with probability less than 1, then there is some positive probability that none of the events occur. However, if there are dependencies amongst the events, it becomes more difficult to determine conditions that guarantee that none of the events occur with some positive probability. The Local Lemma examines the conditions on the dependencies and probabilities of the events that guarantee that all of the “bad” events can be avoided with some nonzero probability, and has been used in many different fields in order to determine the existence of combinatorial objects or bounds on their sizes. The technique is nonconstructive in the sense that it does not actually generate an object that can be used, but simply makes a guarantee about its existence or size; however, a constructive technique using the Local Lemma was first proposed by Beck in [7] for a more restricted version of the Local Lemma, and more recently, a fully general algorithm has been described by Moser and Tardos in [70], which contains additional references regarding algorithmic forms of the Local Lemma.

Godbole, Skipper, and Sunley [46] use the symmetric Local Lemma as defined below
to derive an upper bound on the size of covering arrays. We extend their idea to use
the symmetric Local Lemma to find bounds on variable strength covering arrays over
specific hypergraphs, and compare the bounds with those guaranteed by our algorithm,
\textsc{VarDens}, and we also examine the possibility of using the more general forms of the
Local Lemma when it would make sense to do so.

The Local Lemma comes in several distinct flavours, but the most commonly used
and easy to apply version is the symmetric Local Lemma, as given in Theorem 6.1.1,
which assumes a similarity between the events in terms of probabilities and dependencies.
The general Local Lemma, as given in Theorem 6.1.3, allows for much more flexibility in
terms of the interdependencies of events and variances in their individual probabilities,
but is more difficult to use due to the requirement of finding the $x$ values needed for
its application. Most applications of the general Local Lemma fall into two specific
subtheorems, called the asymmetric Local Lemma, as given in Theorem 6.1.4, and the
weighted Local Lemma, as given in Theorem 6.1.5, which both avoid the problem of
requiring a specific assignment of $x$ values. More information about the various forms
of the Local Lemma and the probabilistic method can be found in [48].

The symmetric Local Lemma was first proposed in [43], but the initial proof required
the condition $4pd \leq 1$. In [83], Spencer improved this condition to $ep(d + 1) \leq 1$, where
$e$ is the base of the natural logarithm, which is the form in which the Local Lemma is
most commonly stated.

\textbf{Theorem 6.1.1} (Lovász Local Lemma - Symmetric Case [43, 83]). Consider a finite set
of events $\mathcal{A} = \{A_0, \ldots, A_{m-1}\}$ in a probability space $\Omega$ such that each event occurs with
probability at most $p < 1$, and each event is independent of all but at most $d$ of the other
events. If $ep(d + 1) \leq 1$, where $e$ is the base of the natural logarithm, then the probability
that none of the events occur is nonzero.

Shearer, in [81], identified certain cases where the condition $epd \leq 1$ is sufficient, as
given in Theorem 6.1.2.

\textbf{Theorem 6.1.2} (Lovász Local Lemma - Shearer Symmetric Case [81]). If, in The-
orem 6.1.1, we have one of the following cases:

1. $d = 1$ and $p < 1/2$, or

2. $d > 1$ and $p < \frac{(d-1)(d-1)}{d^2}$

then the probability that none of the events occur is nonzero.
Theorem 6.1.3 (Lovász Local Lemma - General Case [43]). Let \( \mathcal{A} = \{A_0, \ldots, A_{m-1}\} \) be a finite set of events in a probability space \( \Omega \). Define a function \( \Gamma : \mathcal{A} \rightarrow \mathcal{P}(\mathcal{A}) \) such that for \( A \in \mathcal{A} \), \( A \) is independent from all events in \( \mathcal{A} \setminus (\Gamma(A) \cup \{A\}) \). If there is a map \( x : \mathcal{A} \rightarrow (0,1) \) such that for all \( A \in \mathcal{A} \):

\[
\Pr(A) \leq x(A) \prod_{B \in \Gamma(A)} (1 - x(B))
\]

then the probability that none of the events occur is nonzero, and is:

\[
\Pr(A_0 \land \ldots \land A_{m-1}) \geq \prod_{A \in \mathcal{A}} (1 - x(A)).
\]

The general Local Lemma, as expected, is a generalization of the symmetric Local Lemma, and the symmetric Local Lemma can be obtained from the general Local Lemma by setting \( x(A) = \frac{1}{d+1} \) for all \( A \in \mathcal{A} \), which gives the required result as:

\[
\left(1 - \frac{1}{d+1}\right)^d \geq \frac{1}{e}.
\]

Theorem 6.1.4 (Lovász Local Lemma - Asymmetric Case [48]). Let \( \mathcal{A} = \{A_0, \ldots, A_{m-1}\} \) be a finite set of events in a probability space \( \Omega \). Define a function \( \Gamma : \mathcal{A} \rightarrow \mathcal{P}(\mathcal{A}) \) such that for \( A \in \mathcal{A} \), \( A \) is independent from all events in \( \mathcal{A} \setminus (\Gamma(A) \cup \{A\}) \). If, for each \( 0 \leq i < m \), we have that both:

1. \( \Pr(A_i) \leq \frac{1}{8} \)
2. \( \sum_{A_j \in \Gamma(A_i)} \Pr(A_j) \leq \frac{1}{4} \)

then the probability that none of the events occur is positive.

Theorem 6.1.5 (Lovász Local Lemma - Weighted Case [48]). Let \( \mathcal{A} = \{A_0, \ldots, A_{m-1}\} \) be a finite set of events in a probability space \( \Omega \). Define a function \( \Gamma : \mathcal{A} \rightarrow \mathcal{P}(\mathcal{A}) \) such that for \( A \in \mathcal{A} \), \( A \) is independent from all events in \( \mathcal{A} \setminus (\Gamma(A) \cup \{A\}) \). If, for each \( 0 \leq i < m \), there exists a positive integer \( t_i \geq 1 \), and there exists \( 0 \leq p < \frac{1}{8} \) such that the following hold:

1. \( \Pr(A_i) \leq p^{t_i} \)
2. \( \sum_{A_j \in \Gamma(A_i)} (2p)^{t_j} \leq \frac{4}{t_i} \)

then the probability that none of the events occur is positive.
The asymmetric Local Lemma in Theorem 6.1.4 follows directly from the general Local Lemma in Theorem 6.1.3 by setting $x_i = 2 \Pr(A_i)$ for each $i$. Since, for all $i$, $\Pr(A_i) \leq \frac{1}{8}$ by assumption, then $x_i \leq \frac{1}{4}$, which gives that $(1 - x_i) \geq e^{-1.2x_i}$. The proof that the weighted Local Lemma in Theorem 6.1.5 holds given the general Local Lemma in Theorem 6.1.3 is similar, and arises from setting $x_i = (2p)^{t_i}$.

Godbole, Skipper, and Sunley [46] use the symmetric Local Lemma to derive an upper bound on the size of $\text{CAN}(t, k, g)$. Their strategy is to find a lower bound on a positive integer $n$ such that if they generate a random $n \times k$ array with entries chosen with uniform probability from $\mathbb{Z}_g$, then the probability that no $n \times t$ subarray is missing a $t$-tuple from $\mathbb{Z}_n^t$ is nonzero. This then translates into an upper bound on the size of $\text{CAN}(t, k, g)$, which is given in Theorem 6.1.6.

**Theorem 6.1.6 (Covering array bound - Local Lemma [46]).**

$$\text{CAN}(t, k, g) \leq \frac{(t - 1) \log k}{\log \frac{g^t}{g^t - 1}} + O(1).$$

Our goal is to use a similar technique to derive bounds on the size of VCA over various families of hypergraphs and compare the results to the bound derived from $\text{VARDENS}$ in Theorem 5.2.3. This will also afford us the opportunity, in the case of certain families of hypergraphs, to make conclusions regarding the predictive power of the Local Lemma versus $\text{VARDENS}$. Since the general framework of the technique used in the proof of the derivation of each bound is similar, we give an intermediate result in Lemma 6.1.9 that encapsulates most of the details of the calculation and allows us to directly state most of the Local Lemma bounds as a corollary. Additionally, this lemma uses the result of Lemma 5.2.2, which describes a natural logarithm approximation that allows us to simplify many of the derived bounds.

Our strategy, like that of Godbole, Skipper, and Sunley, involves finding a value $n$ such that given a hypergraph $H$ over $k$ points and a value $g > 0$, randomly filling in an $n \times k$ array $M$ with entries chosen with uniform probability from $\mathbb{Z}_g$ yields an array that is, with some nonzero probability, a $\text{VCA}(n; H, g)$. In this case, we define one bad event for each edge $e \in E(H)$, say $A_e$, which is the event that the $n \times |e|$ subarray $M'$ over the columns corresponding to the vertices in $e$ is missing some $|e|$-tuple from $\mathbb{Z}_g^{|e|}$. Thus, we require the probability that the event $A_e$ occurs in generating $M$. While we will typically use a rough upper bound on this probability, we show how to calculate it precisely. To do so, we must consider the exclusion of each of the $|e|$-tuples as distinct but possibly dependent events and calculate the joint probability that they are missing.
The Lovász Local Lemma and Probabilistic Bounds

from the subarray $M'$. Lemma 6.1.7 demonstrates how to calculate the joint probability of dependent events, and the final result, with details on how to truncate the expression to obtain upper bounds on the probability, is given in Corollary 6.1.8. We note that in Theorem 6.1.6, the probability upper bound is truncated after the first term, which is similar to the strategy that we employ in our nonconstructive upper bound derivations.

**Lemma 6.1.7.** If $A = \{A_0, \ldots, A_{m-1}\}$ is a set of events in a probability space, then:

$$
\Pr(A_0 \lor \ldots \lor A_{m-1}) = \sum_{i=1}^{m} (-1)^{i+1} \sum_{S \in \binom{A_i}{i}} \Pr \left( \bigwedge_{j \in S} A_j \right).
$$

Then for any $1 \leq s \leq m$, $s$ odd, we can truncate after $s$ terms to derive an upper bound:

$$
\Pr(A_0 \lor \ldots \lor A_{m-1}) \leq \sum_{i=1}^{s} (-1)^{i+1} \sum_{S \in \binom{A_i}{i}} \Pr \left( \bigwedge_{j \in S} A_j \right).
$$

**Proof.** This is simply the Inclusion-Exclusion principle applied to probability spaces. The second inequality follows by truncating the formula after an inclusion step has been performed, which gives an upper bound on the probability. \hfill \Box

We now have the following result about coverage in a random $n \times t$ array, which will serve to compute the probability of a bad event in our application of the Local Lemma.

**Corollary 6.1.8.** If we randomly fill an $n \times t$ array by selecting the value in each position uniformly from $\mathbb{Z}_g$, the probability that the array will miss at least one of the $g^t$ t-tuples is:

$$
\sum_{i=1}^{g^t} (-1)^{i+1} \left( \frac{g^t}{i} \right) \left( \frac{g^t - i}{g^t} \right)^n \leq g^t \left( \frac{g^t - 1}{g^t} \right)^n.
$$

**Proof.** The inequality comes directly from Lemma 6.1.7 by truncating after the first term of the summation. We define $g^t$ events, with event $A_i$ being the event that, according to any ordering on $\mathbb{Z}_g^t$, the $i$th t-tuple of $\mathbb{Z}_g^t$ does not appear amongst the rows of the array. \hfill \Box

We now have all the tools in place to create a general framework for our symmetric Local Lemma-based proofs.

**Lemma 6.1.9.** Let $H = (V, E)$ be a hypergraph with rank($H$) = $t$, and let $d$ be an integer such that no edge of $H$ intersects more than $d$ other edges of $H$:

$$
\max\{|\{e' \in E \setminus \{e\} : e \cap e' \neq \emptyset\}| : e \in E\} \leq d.
$$
Then, for any $g \geq 1$, we have:

$$VCAN(H, g) \leq \frac{\ln(d + 1) + t \ln g + 1}{\ln \frac{g^t}{g^t - 1}}.$$  

Additionally, we have:

$$VCAN(H, g) \leq g^t (\ln(d + 1) + t \ln g + 1).$$

**Proof.** Let $k = |V|$. For a positive integer $n$, consider a randomly generated $n \times k$ array $M$ with entries chosen from $\mathbb{Z}_g$ with uniform probability. For each edge $e \in E$, write $s = |e|$, and associate an event $A_e$ that the $n \times s$ subarray of $M$ over the columns of $e$ is missing, as rows, one or more of the $g^s$ $s$-tuples of $\mathbb{Z}_g^s$. Define:

$$p = g^t \left( \frac{g^t - 1}{g^t} \right)^n.$$  

From Corollary 6.1.8 and the fact that $s \leq t$, we have:

$$\Pr(A_e) \leq g^s \left( \frac{g^s - 1}{g^s} \right)^n \leq g^t \left( \frac{g^t - 1}{g^t} \right)^n = p.$$  

We apply the symmetric Local Lemma as in Theorem 6.1.1, which states that if $ep(d + 1) \leq 1$, the probability that none of the bad events occur is positive, and hence there is some $n \times k$ array that is a VCA($n; H, g$). This happens when:

$$ep(d + 1) \leq 1$$

$$e \left( g^t \left( \frac{g^t - 1}{g^t} \right)^n \right) (d + 1) \leq 1$$

$$n \geq \frac{\ln(d + 1) + t \ln g + 1}{\ln \frac{g^t}{g^t - 1}}.$$  

Thus, we have:

$$VCAN(H, g) \leq \frac{\ln(d + 1) + t \ln g + 1}{\ln \frac{g^t}{g^t - 1}}.$$  

If we use the approximation from Lemma 5.2.2, we can rewrite the inequality as follows:

$$VCAN(H, g) \leq g^t (\ln(d + 1) + t \ln g + 1).$$

With this framework in place, we now examine upper bounds for VCA over individual families of hypergraphs and compare the bounds to those from VarDens.
6.1.1 VCA over Combinatorial Designs

We first use the Local Lemma and VarDens to determine some upper bounds for VCA over combinatorial designs. In these arguments, the bad events correspond to the failure of coverage in the subarrays over the columns corresponding to points in a block. Additionally, two events are independent if and only if their corresponding blocks do not intersect, so our goal will be to count the block intersections in these designs.

The simplest cases that we can analyze with the highest degree of accuracy are the \((t - 1) -(k, t, \lambda)\) simple designs. If \(B\) is a \((t - 1) -(k, t, \lambda)\) design, then the ASC \(\Delta(B)\) contains every \((t - 1)-s\et\) over the base set since by nature of the design, every \((t - 1)-s\et\) appears in exactly \(\lambda\) blocks of the design. This regularity, along with the simplicity of the design, guarantees that the block intersection number (i.e. the number of blocks that a given block intersects) is uniform across all choices of block. The block intersection number for these designs is determined in Lemma 6.1.10; we can then immediately use the symmetric Local Lemma as in the framework in Theorem 6.1.9 to derive an upper bound on the size of a covering array for any given number of levels.

Lemma 6.1.10. Let \(B\) be a \((t - 1) -(k, t, \lambda)\) simple design. Then for any \(B \in B\):

\[
d = |\{B' \in B \setminus \{B\} : B \cap B' \neq \emptyset\}| = \sum_{i=1}^{t-1} (-1)^i+1 \binom{t}{i} \lambda \binom{k-i}{(t-1)-i} \frac{(t-i-1)}{t-i} - 1.
\]

Furthermore, for any \(1 \leq m \leq t - 1, m \text{ odd}\), we can truncate the summation after \(m\) terms to derive an upper bound on \(d\).

Proof. We first note that if we consider the ASC of the design, \(\Delta(B)\), then the facets of this ASC are precisely the sets of size \(t\) in \(B\): any set \(S\) of size \(t - 1\) is, by the strength of the design, contained in a block, so neither \(S\) nor any subset of \(S\) can be a facet.

Consider any point \(a \in B\). Then \(a\) must appear in \(\binom{k-1}{(t-1)-1}\) sets of size \(t - 1\) and each \((t - 1)-s\et\) must appear in \(\lambda\) blocks. Each block of \(B\) covers exactly \(\binom{t-1}{(t-1)-1}\) \(t - 1\) sets of size \(t - 1\) containing \(a\), so \(a\) then appears in \(\frac{\lambda \binom{k-i}{(t-1)-i}}{t-i} \) blocks of \(B\). We do not, however, want to include \(B\) in this count, so we subtract 1. There are \(\binom{t}{1}\) possible choices for \(a\).

By taking this sum for all the possible choices of points \(a \in B\), we have counted multiple times the blocks that contain multiple points of \(B\). Hence, we must use the Inclusion-Exclusion principle to account for overcounting, and thus derive a general term: for any \(i\)-subset of \(S \subset B, 1 \leq i \leq t - 1\), we have that \(S\) must appear in \(\binom{k-1}{(t-1)-i}\) sets of size \(t - 1\), and each of these sets of size \(t - 1\) must appear in \(\lambda\) blocks. Each block
of $B$ covers precisely $\binom{t-i}{(t-1)-i} = t-i$ sets of size $t-1$ containing $S$, so $S$ appears in $\lambda \binom{k-i}{(t-1)-i}$ blocks of $B$. Again, we do not want to include $B$ in this count, so we must subtract 1. There are $\binom{t}{i}$ possible sets $S$. Applying this iteratively to remove and add back overcounted elements by the Inclusion-Exclusion principle gives the equation.

We can truncate the summation after an addition (inclusion) in order to derive an upper bound which has included some overcounted blocks.

We begin by examining bounds for $S(t-1,t,k)$ systems. For $t=3$ these are exactly the STS$(k)$, and for $t=4$ these are the SQS$(k)$, which, from Theorems 2.3.11 and 2.3.13, exist for infinitely many values of $k$.

For $t=6$, $S(5,6,k)$ are only known for a finite number of values of $k$, and all known $S(4,5,k)$ are derived from $S(5,6,k)$. There are no known systems for $t > 6$. Thus, using Lemma 6.1.10, we calculate the precise block intersection numbers for $3 \leq t \leq 6$.

**Corollary 6.1.11.** For an $S(t-1,t,k)$ system, given any block, the exact number of blocks it intersects, say $d$, is given in the following table:

<table>
<thead>
<tr>
<th>$t$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>$\frac{3}{2}k - \frac{9}{2}$</td>
</tr>
<tr>
<td>4</td>
<td>$\frac{3}{4}k^2 - \frac{11}{3}k + 8$</td>
</tr>
<tr>
<td>5</td>
<td>$\frac{5}{24}k^3 - \frac{35}{12}k^2 + \frac{125}{8}k - \frac{125}{4}$</td>
</tr>
<tr>
<td>6</td>
<td>$\frac{1}{20}k^4 - \frac{2}{8}k^3 + \frac{257}{24}k^2 - \frac{595}{12}k + \frac{451}{5}$</td>
</tr>
</tbody>
</table>

Now, we can use the probability upper bounds from Corollary 6.1.8 and the block intersection numbers from Corollary 6.1.11 with the framework of Lemma 6.1.9 to find upper bounds on the sizes of VCA over these designs.

**Theorem 6.1.12** (Local Lemma bound for VCA over $S(t-1,t,k)$). We have the following upper bounds and asymptotics:

1. If $B$ is an STS$(k)$:

   $$VCAN(B,g) \leq \frac{\ln \left( \frac{3}{2}k - \frac{7}{2} \right) + 3 \ln g + 1}{\ln \frac{g^3}{g^3-1}}.$$

   For fixed $g$, as $k \to \infty$:

   $$VCAN(B,g) \leq g^3 \ln k + O(1).$$
2. If $\mathcal{B}$ is an SQS($k$):

$$VCAN(\mathcal{B}, g) \leq \frac{\ln \left( \frac{2}{3} k^2 - \frac{11}{3} k + 9 \right) + 4 \ln g + 1}{\ln \frac{g^2}{g^2-1}}.$$  
For fixed $g$, as $k \to \infty$:

$$VCAN(\mathcal{B}, g) \leq 2g^4 \ln k + O(1).$$

3. If $\mathcal{B}$ is an $S(4, 5, k)$:

$$VCAN(\mathcal{B}, g) \leq \frac{\ln \left( \frac{5}{24} k^3 - \frac{35}{12} k^2 + \frac{125}{8} k - \frac{121}{4} \right) + 5 \ln g + 1}{\ln \frac{g^5}{g^5-1}}.$$  
For fixed $g$, as $k \to \infty$:

$$VCAN(\mathcal{B}, g) \leq 3g^5 \ln k + O(1).$$

4. If $\mathcal{B}$ is an $S(5, 6, k)$:

$$VCAN(\mathcal{B}, g) \leq \frac{\ln \left( \frac{1}{20} k^4 - \frac{9}{8} k^3 + \frac{257}{24} k^2 - \frac{595}{12} k + \frac{456}{5} \right) + 6 \ln g + 1}{\ln \frac{g^6}{g^6-1}}.$$  
For fixed $g$, as $k \to \infty$:

$$VCAN(\mathcal{B}, g) \leq 4g^6 \ln k + O(1).$$

5. There are currently no known $S(t-1, t, k)$ for $t > 6$; however, if such a system $\mathcal{B}$ were to exist, for fixed $g$, as $k \to \infty$:

$$VCAN(\mathcal{B}, g) \leq (t-2)g^t \ln k + O(1).$$

Proof. The asymptotics immediately follow from the second formula of Lemma 6.1.9 and observing that $\ln(a_n k^n + \ldots + a_0) \to \ln k^n = n \ln k$ as $k \to \infty$. 

We now compare with the upper bound as given by VARDENS. From Corollary 2.3.5, we can calculate the number of blocks in each of the families of designs of interest.

Corollary 6.1.13. For a $S(t-1, t, k)$ system, we have the following number $b$ of blocks:
We then substitute the values into Theorem 5.2.3 to give us the upper bound and asymptotics as derived from \textsc{VarDens}.

**Theorem 6.1.14** (\textsc{VarDens} bound for VCA over $S(t - 1, t, k)$). We have the following upper bounds and asymptotics:

1. If $\mathcal{B}$ is an STS($k$):

   \[
   VCAN(\mathcal{B}, g) \leq \frac{\ln \left( \frac{1}{6} k^2 - \frac{1}{6} k \right) + 3 \ln g}{\ln \frac{g^3}{g^3 - 1}}.
   \]

   For fixed $g$, as $k \to \infty$:

   \[
   VCAN(\mathcal{B}, g) \leq 2g^3 \ln k + O(1).
   \]

2. If $\mathcal{B}$ is an SQS($k$):

   \[
   VCAN(\mathcal{B}, g) \leq \frac{\ln \left( \frac{1}{24} k^3 - \frac{1}{8} k^2 + \frac{1}{12} k \right) + 4 \ln g}{\ln \frac{g^4}{g^4 - 1}}.
   \]

   For fixed $g$, as $k \to \infty$:

   \[
   VCAN(\mathcal{B}, g) \leq 3g^4 \ln k + O(1).
   \]

3. If $\mathcal{B}$ is an $S(4, 5, k)$:

   \[
   VCAN(\mathcal{B}, g) \leq \frac{\ln \left( \frac{1}{120} k^4 - \frac{1}{20} k^3 + \frac{11}{120} k^2 - \frac{1}{20} k \right) + 5 \ln g}{\ln \frac{g^5}{g^5 - 1}}.
   \]

   For fixed $g$, as $k \to \infty$:

   \[
   VCAN(\mathcal{B}, g) \leq 4g^5 \ln k + O(1).
   \]
4. If $\mathcal{B}$ is an $S(5, 6, k)$:

$$VCAN(\mathcal{B}, g) \leq \frac{\ln \left( \frac{1}{720} k^5 - \frac{1}{72} k^4 + \frac{7}{144} k^3 - \frac{5}{72} k^2 + \frac{1}{30} \right) + 6 \ln g}{\ln \left( \frac{g^6}{g^6 - 1} \right)}.$$

For fixed $g$, as $k \to \infty$:

$$VCAN(\mathcal{B}, g) \leq 5g^6 \ln k + O(1).$$

Hence, for all $t$, the upper bound derived from the Local Lemma is asymptotically $(t - 2)/(t - 1) = (s - 1)/s$ times the bound derived from VARDENS. Some experimental results comparing the actual values for STS and SQS are enumerated in Tables 6.1 and 6.2 respectively.

We now turn our attention to general $s-(k, t, \lambda)$ designs and consider the following cases:

1. $(t - 1)-(k, t, \lambda)$ simple designs, $\lambda > 1$. Then the block intersection count is uniform across all blocks, and is given by Lemma 6.1.10.

2. $(t - 1)-(k, t, \lambda)$ nonsimple designs, $\lambda > 1$. Then the block intersection count may not be uniform due to repeated blocks, and Lemma 6.1.15 provides a general upper bound across all blocks.

3. $s-(k, t, \lambda)$ designs with $s < t - 1$. If $\mathcal{B}$ is such a design, then it is possible that $\Delta(\mathcal{B})$ does not contain every set of size $s + 1, \ldots, t - 1$; thus, the Inclusion-Exclusion principle cannot be used in general to fully eliminate double-counting when calculating block intersection counts, and indeed, the block intersection counts may be variable, thus resulting in a variable number of dependencies between the events in our symmetric Local Lemma framework. An example of a design which has variable block intersection counts is the $1-(9, 3, 2)$ design in Figure 6.1, which has blocks:

$$\mathcal{B} = \{012, 013, 245, 367, 578, 468\}.$$

In this design, the blocks 012 and 013 intersect two other blocks each, whereas the remaining blocks intersect three other blocks each. As in the previous case, we can use Lemma 6.1.15 to derive an upper bound on the block intersection count across all blocks.
Table 6.1: Comparison on the bounds of $\text{VCAN}(B, g)$, where $B$ are the blocks of an STS($k$). The value in the table is $\frac{n_{\text{LL}} - n_{\text{VARDENS}}}{n_{\text{VARDENS}}}$, where $n_{\text{LL}}$ is the Local Lemma upper bound, and $n_{\text{VARDENS}}$ is the VARDENS upper bound, so negative values indicate parameters where the Local Lemma bound is smaller than the VARDENS bound, and positive values indicate parameters were the VARDENS bound is smaller than the Local Lemma bound.
The Lovász Local Lemma and Probabilistic Bounds

<table>
<thead>
<tr>
<th>$k$</th>
<th>$g = 2$</th>
<th>$g = 3$</th>
<th>$g = 4$</th>
<th>$g = 5$</th>
<th>$g = 6$</th>
<th>$g = 7$</th>
<th>$g = 8$</th>
<th>$g = 9$</th>
<th>$g = 10$</th>
<th>$g = 100$</th>
<th>$g = 1000$</th>
<th>$g = 10000$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>+0.941</td>
<td>+0.594</td>
<td>+0.471</td>
<td>+0.405</td>
<td>+0.364</td>
<td>+0.335</td>
<td>+0.314</td>
<td>+0.297</td>
<td>+0.283</td>
<td>+0.142</td>
<td>+0.094</td>
<td>+0.071</td>
</tr>
<tr>
<td>8</td>
<td>+0.271</td>
<td>+0.209</td>
<td>+0.179</td>
<td>+0.162</td>
<td>+0.150</td>
<td>+0.141</td>
<td>+0.134</td>
<td>+0.128</td>
<td>+0.124</td>
<td>+0.070</td>
<td>+0.048</td>
<td>+0.037</td>
</tr>
<tr>
<td>10</td>
<td>+0.204</td>
<td>+0.162</td>
<td>+0.141</td>
<td>+0.128</td>
<td>+0.119</td>
<td>+0.113</td>
<td>+0.108</td>
<td>+0.104</td>
<td>+0.100</td>
<td>+0.058</td>
<td>+0.041</td>
<td>+0.031</td>
</tr>
<tr>
<td>14</td>
<td>+0.133</td>
<td>+0.109</td>
<td>+0.096</td>
<td>+0.089</td>
<td>+0.083</td>
<td>+0.079</td>
<td>+0.076</td>
<td>+0.073</td>
<td>+0.071</td>
<td>+0.042</td>
<td>+0.030</td>
<td>+0.023</td>
</tr>
<tr>
<td>16</td>
<td>+0.111</td>
<td>+0.091</td>
<td>+0.081</td>
<td>+0.075</td>
<td>+0.071</td>
<td>+0.067</td>
<td>+0.064</td>
<td>+0.062</td>
<td>+0.060</td>
<td>+0.037</td>
<td>+0.026</td>
<td>+0.020</td>
</tr>
<tr>
<td>20</td>
<td>+0.078</td>
<td>+0.065</td>
<td>+0.059</td>
<td>+0.054</td>
<td>+0.051</td>
<td>+0.049</td>
<td>+0.047</td>
<td>+0.046</td>
<td>+0.044</td>
<td>+0.027</td>
<td>+0.020</td>
<td>+0.015</td>
</tr>
<tr>
<td>22</td>
<td>+0.066</td>
<td>+0.055</td>
<td>+0.050</td>
<td>+0.046</td>
<td>+0.044</td>
<td>+0.042</td>
<td>+0.040</td>
<td>+0.039</td>
<td>+0.038</td>
<td>+0.023</td>
<td>+0.017</td>
<td>+0.013</td>
</tr>
<tr>
<td>26</td>
<td>+0.046</td>
<td>+0.039</td>
<td>+0.035</td>
<td>+0.033</td>
<td>+0.031</td>
<td>+0.030</td>
<td>+0.028</td>
<td>+0.027</td>
<td>+0.026</td>
<td>+0.017</td>
<td>+0.012</td>
<td>+0.010</td>
</tr>
<tr>
<td>28</td>
<td>+0.037</td>
<td>+0.032</td>
<td>+0.029</td>
<td>+0.027</td>
<td>+0.025</td>
<td>+0.024</td>
<td>+0.024</td>
<td>+0.023</td>
<td>+0.022</td>
<td>+0.014</td>
<td>+0.010</td>
<td>+0.008</td>
</tr>
<tr>
<td>32</td>
<td>+0.023</td>
<td>+0.020</td>
<td>+0.018</td>
<td>+0.017</td>
<td>+0.016</td>
<td>+0.015</td>
<td>+0.015</td>
<td>+0.014</td>
<td>+0.014</td>
<td>+0.009</td>
<td>+0.007</td>
<td>+0.005</td>
</tr>
<tr>
<td>34</td>
<td>+0.017</td>
<td>+0.015</td>
<td>+0.014</td>
<td>+0.013</td>
<td>+0.012</td>
<td>+0.012</td>
<td>+0.011</td>
<td>+0.011</td>
<td>+0.011</td>
<td>+0.007</td>
<td>+0.005</td>
<td>+0.004</td>
</tr>
<tr>
<td>38</td>
<td>+0.007</td>
<td>+0.006</td>
<td>+0.005</td>
<td>+0.005</td>
<td>+0.005</td>
<td>+0.005</td>
<td>+0.004</td>
<td>+0.004</td>
<td>+0.004</td>
<td>+0.003</td>
<td>+0.002</td>
<td>+0.002</td>
</tr>
<tr>
<td>40</td>
<td>+0.002</td>
<td>+0.002</td>
<td>+0.002</td>
<td>+0.002</td>
<td>+0.001</td>
<td>+0.001</td>
<td>+0.001</td>
<td>+0.001</td>
<td>+0.001</td>
<td>+0.001</td>
<td>+0.000</td>
<td>+0.000</td>
</tr>
<tr>
<td>44</td>
<td>−0.006</td>
<td>−0.005</td>
<td>−0.005</td>
<td>−0.005</td>
<td>−0.004</td>
<td>−0.004</td>
<td>−0.004</td>
<td>−0.004</td>
<td>−0.003</td>
<td>−0.002</td>
<td>−0.002</td>
<td>−0.002</td>
</tr>
<tr>
<td>46</td>
<td>−0.010</td>
<td>−0.009</td>
<td>−0.008</td>
<td>−0.007</td>
<td>−0.007</td>
<td>−0.007</td>
<td>−0.006</td>
<td>−0.006</td>
<td>−0.004</td>
<td>−0.003</td>
<td>−0.002</td>
<td>−0.002</td>
</tr>
<tr>
<td>50</td>
<td>−0.017</td>
<td>−0.015</td>
<td>−0.013</td>
<td>−0.013</td>
<td>−0.012</td>
<td>−0.012</td>
<td>−0.011</td>
<td>−0.011</td>
<td>−0.011</td>
<td>−0.007</td>
<td>−0.005</td>
<td>−0.004</td>
</tr>
</tbody>
</table>

Table 6.2: Comparison on the bounds of $\text{VCAN}(\mathcal{B}, g)$, where $\mathcal{B}$ are the blocks of an SQS($k$). The value in the table is $\frac{n_{\text{LL}} - n_{\text{VARDENS}}}{n_{\text{VARDENS}}}$, where $n_{\text{LL}}$ is the Local Lemma upper bound, and $n_{\text{VARDENS}}$ is the VARDENS upper bound, so negative values indicate parameters where the Local Lemma bound is smaller than the VARDENS bound, and positive values indicate parameters where the VARDENS bound is smaller than the Local Lemma bound.
If $B$ is a design, we note that when we discuss a VCA$(n; B, g)$, we are actually considering a covering array over the ASC induced by the design, i.e. $\Delta(B)$. An ASC is a set, and thus repeated blocks in the design actually weaken the requirements of the covering array since the facet set $\Lambda$ over a nonsimple design actually contains fewer facets than over a simple design.

![Diagram](image)

**Figure 6.1:** A 1-(9, 3, 2) design with facets whose dependencies vary.

**Lemma 6.1.15.** Let $B$ be a $s$-$(k, t, \lambda)$ design. Then for any $B \in B$:

$$d = |\{B' \in B \setminus \{B\} : B \cap B' \neq \emptyset\}| \leq \sum_{i=1}^{2|s-1|+1} (-1)^{i+1} \binom{t}{i} \left( \frac{\lambda^{(k-i)}(k-s-i)}{(t-i)(s-i)} - 1 \right).$$

Furthermore, for any $1 \leq m \leq s$, $m$ odd, we can truncate the summation after $m$ terms to derive an upper bound on the summation.

**Proof.** We note that the upper bound on the summation index $i$, $s - 1 \leq 2\lfloor \frac{s-1}{2} \rfloor + 1 \leq s$ is simply the odd integer in this range. Consider any block $B \in B$, and consider the number of blocks it intersects in exactly one point. For each of the possible $\binom{t}{i}$ points $u \in B$, we determine the number of blocks containing $u$. We have that $u$ occurs in exactly $\binom{k-1}{s-1}$ $s$-sets, and each $s$-set must appear $\lambda$ times in the blocks of $B$. Each block of $B$ covers exactly $\binom{t-1}{s-1}$ of the $s$-sets containing $u$, and hence $u$ is in $\lambda \frac{(k-1)}{(t-1)(s-1)}$ blocks of $B$. We subtract 1 to exclude $B$ from this count, giving the term for $i = 1$.

This, however, has double-counted the blocks that $B$ intersects in a pair, so we must add these back, and so on, using the Inclusion-Exclusion principle. However, as we
cannot predict the intersections between blocks for sets of size \( s+1, \ldots, t-1 \) since the sets of these sizes that appear as facets in \( \Delta(\mathcal{B}) \) is dependent on the structure of \( \mathcal{B} \), we can calculate only an upper bound on the number of dependencies: to do so, by the Inclusion-Exclusion principle, we must truncate after an odd number of terms, i.e. after an addition, so that we ensure only overcounting, and no undercounting has occurred. Thus, we stop at the odd integer at most \( s \).

Furthermore, as simplicity is not assumed, the design could contain repeated blocks; thus, even when \( s = t - 1 \), if we allow for repeated blocks, we must still allow for \( d \) to merely be an upper bound.

In order to use the symmetric Local Lemma to derive upper bounds on VCA for designs where \( s \) may be less than \( t-1 \) or we do not assume simplicity, we calculate a bound on \( d \) as above, and then we can apply the Local Lemma as used in Theorem 6.1.16; however, since the upper bound \( d \) on block intersection counts may differ from one design to the next, except for \((t-1)-(k, t, \lambda)\) simple designs, there may be room for improvement in the bound. We note that this approach generalizes and captures the behaviour of Theorem 6.1.12.

**Theorem 6.1.16** (Local Lemma bound for VCA over \( s-(k, t, \lambda) \) designs). Let \( \mathcal{B} \) be an \( s-(k, t, \lambda) \) design, and let \( d \) be an upper bound on the block intersection count of \( \mathcal{B} \). Then:

\[
\text{VCAN}(\mathcal{B}, g) \leq \frac{\ln(d+1) + t \ln g + 1}{\ln g^{s-1}}.
\]

For fixed \( s, t, g, \) and \( \lambda \), as \( k \to \infty \), we have that:

\[
\text{VCAN}(\mathcal{B}, g) \leq (s-1) g^{t} \ln k + O(1).
\]

**Proof.** The first equation is a direct consequence of the first equation of Lemma 6.1.9. The second equation follows from the second equation of Lemma 6.1.9, combined with either Lemma 6.1.10 or Lemma 6.1.15, depending on which case we are in from above; using the formulas in these lemmas to obtain an upper bound on \( d \), we have that the expansion of the summation in these formulas, regardless of number of terms considered, is a polynomial in \( k \) of degree \( s-1 \). Hence, as \( k \to \infty \), \( \ln(d+1) \to \ln k^{s-1} = (s-1) \ln k \).

Another possibility, when there is variance in the block intersection count, would be to use one of the forms of the general Local Lemma to possibly get a better bound, which allows for variations in the event dependencies and probabilities.
Open Question 6.1.1. For nonsimple $s$-$(k, t, \lambda)$ designs or simple $s$-$(k, t, \lambda)$ designs with $s < t - 1$, is it possible to use one of the forms of the general Local Lemma to derive a better upper bound on VCA over these designs?

We now use \textsc{VarDens} to determine a second bound for VCA over general $s$-$(k, t, \lambda)$ designs, and then compare with the symmetric Local Lemma bound.

\textbf{Theorem 6.1.17 (\textsc{VarDens} bound for VCA over $s$-$(k, t, \lambda)$ designs).} Let $\mathcal{B}$ be an $s$-$(k, t, \lambda)$ design and let:

$$b = \frac{\lambda \binom{k}{s}}{\binom{t}{s}}.$$

We then have:

$$\text{VCAN}(\mathcal{B}, g) \leq \frac{\ln b + t \ln g}{\ln \frac{g^t}{g^s - 1}}.$$

Furthermore, for $s, t, \lambda,$ and $g$ fixed, as $k \to \infty$, we have:

$$\text{VCAN}(\mathcal{B}, g) \leq sg^t \ln k + O(1).$$

\textbf{Proof.} We have, from Corollary 2.3.5, that $|\mathcal{B}| = b$. Consider the ASC $\Delta(\mathcal{B})$ with set of facets $\Lambda$, and invoke \textsc{VarDens} on $\Lambda$. If $\mathcal{B}$ is simple, we have that $|\Lambda| = b$, and if not, then $|\Lambda| < b$. In either case, $b$ is an upper bound, and $\ln |\Lambda| \leq \ln b$, so we can substitute this into the \textsc{VarDens} bound.

Considering $s, t,$ and $\lambda$ fixed, the expression for $b$ is a polynomial in $k$ of degree $s$; thus, as $k \to \infty$, $\ln b \to \ln k^s = s \ln k$. Using the approximation in Lemma 5.2.2 eliminates the denominator and gives the second equation.

Thus, as with the limited case where $s = t - 1$ and $\lambda = 1$, in the general case of $s$-$(k, t, \lambda)$ designs, it is still the case that the symmetric Local Lemma in Theorem 6.1.16 provides the better bound, which is asymptotically $(s - 1)/s$ of the bound derived from \textsc{VarDens} in Theorem 6.1.17.

As discussed in Section 3.4.2, one type of covering arrays of particular interest are uniform covering arrays of size $n$ over $g$ levels, which are covering arrays in which every column has all $g$ levels appearing precisely $n/g$ times. In [39], Donders and Godbole examine a technique to randomly generate uniform covering arrays randomly, and then analyze the problem using the symmetric Local Lemma. Instead of randomly filling an $n \times k$ array entry-by-entry with symbols chosen with uniform probability from $\mathbb{Z}_g$, for any integer $\alpha \geq 1$ such that $(\alpha g)|n$, they consider the set $T$ of all possible $(\alpha g) \times 1$ tiles
that contain each symbol from $\mathbb{Z}_g$ precisely $\alpha$ times, and then they randomly tile each of the $k$ columns of the array with $n/(\alpha g)$ such tiles, chosen with uniform probability from $T$. This guarantees that the resultant array is uniform, and they are then able to apply the symmetric Local Lemma to determine an upper bound on the size of the required array.

**Open Question 6.1.2.** For an STS($k$) or SQS($k$) $\mathcal{B}$, use the symmetric Local Lemma to find a lower bound on $m$ such that a randomly generated $(mg) \times k$ array $M$ is a uniform VCA($mg; \mathcal{B}, g$) with nonzero probability. Then this bound translates to an upper bound on the size of a uniform VCA($mg; \mathcal{B}, g$).

### 6.1.2 VCA over Consecutive Covering Arrays

We now turn our attention to another family of hypergraphs which have a highly regular structure that illustrates the ability of the symmetric Local Lemma to make some predictive measures regarding the size of covering arrays. The result is an upper bound obtained by the symmetric Local Lemma that is better than the $\text{VarDens}$ bound.

In Section 2.2.7, we defined consecutive covering arrays (CCA) and cyclic consecutive covering arrays (cCCA), and then in Section 3.4.3, we investigated direct constructions for both. We now examine the block intersection counts for these two types of arrays.

**Lemma 6.1.18.** Let $k \geq 2t - 1$. The hypergraph $H_{c}^{t,k}$ has $k$ edges, and each edge intersects exactly $2t - 2$ other edges.

*Proof.* For $0 \leq i \leq k-1$, let $e_i = \{i \mod k, (i+1) \mod k, \ldots, (i+t-1) \mod k\} \in E(H_{c}^{k,t})$. Then for each possible value of $i$, these edges are distinct. The edges that intersect $e_i$ are precisely the edges $e_j$ such that:

$$j \in \{(i - t + 1) \mod k, (i - t + 2) \mod k, \ldots, (i + t - 1) \mod k\} \setminus \{i\}.$$  

The interval contains $(i + t - 1) - (i - t + 1) + 1 = 2t - 1$ values, but removing $i$ as a possible starting position to disinclude $e_i$ itself from this list gives that there are $2t - 2$ distinct edges that intersect $e_i$.  

**Lemma 6.1.19.** Assume $k \geq 2t - 1$, and let $H_{c}^{t,k}$ be the hypergraph in Definition 2.2.34. For $0 \leq i \leq k - t$, write $B_i$ to be the edge $\{i, \ldots, i + t - 1\} \in E(H_{c}^{k,t})$. Thus, $H_{c}^{t,k}$ has
The Lovász Local Lemma and Probabilistic Bounds

\(k - t + 1\) edges, and the block intersection count for edge \(B_i\) is:

\[
d = \begin{cases} 
  t + i - 1 & \text{if } 0 \leq i \leq t - 2 \\
  2t - 2 & \text{if } t - 1 \leq i \leq k - 2t + 1 \\
  k - t - i & \text{if } k - 2t + 1 < i \leq k - t.
\end{cases}
\]

**Proof.** For each of the \(k - t + 1\) values \(0 \leq i \leq k - t\), the unique edge \(B_i\) is in the hypergraph.

To count the block intersections, there are three distinct types of blocks \(B_i\) to consider:

- \(t - 1 \leq i \leq k - 2t + 1\). Then, as in Lemma 6.1.18, the number of blocks that intersect \(B_i\) are \(2t - 2\), and are precisely the blocks \(B_{i-(t-1)}, \ldots, B_{i-1}\), and \(B_{i+1}, \ldots, B_{i+(t-1)}\). We note, then, that:
  \[i - (t - 1) \geq 0 \Rightarrow i \geq t - 1\]
  \[i + (t - 1) \leq k - t \Rightarrow i \leq k - 2t + 1.\]

- \(0 \leq i \leq t - 2\). Then the \(i\) blocks \(B_0, \ldots, B_{i-1}\) intersect \(B_i\) on the left, and the \(t - 1\) blocks \(B_{i+1}, \ldots, B_{i+(t-1)}\) intersect \(B_i\) on the right, giving a total of \(i + (t - 1) = t + i - 1\).

- \(k - 2t + 1 < i \leq k - t\). Then the \(t - 1\) blocks \(B_{i-(t-1)}, \ldots, B_{i-1}\) intersect \(B_i\) on the left, and the \(k - 2t - i + 1\) blocks \(B_{i+1}, \ldots, B_{k-2t+1}\) intersect \(B_i\) on the right, giving a total of \((t - 1) + (k - 2t - i + 1) = k - t - i.\)

Because of the regularity of the structure of \(H_{t,k}^c\), it is an ideal candidate for the symmetric Local Lemma, so we apply it to derive an upper bound on \(c_{CCAN}(t,k,g)\) which follows directly from our framework of Lemma 6.1.9. In both of the following theorems, we use Lemma 5.2.2 to eliminate the denominator in the expressions. The \textsc{VarDens} bound comes from Theorem 5.2.3.

**Theorem 6.1.20** (Local Lemma bound for \(c_{CCAN}(t,k,g)\)).

\[
c_{CCAN}(t,k,g) \leq \frac{t \ln g + \ln(2t - 1) + 1}{\ln \frac{g^t}{g^t - 1}} \leq g'(t \ln g + \ln(2t - 1) + 1).
\]

Furthermore, as this is independent of \(k\), for fixed \(g\) and \(t\), it is \(O(1)\).
Theorem 6.1.21 (VARDENS bound for cCCAN\( (t, k, g) \)).

\[
c_{\text{CCAN}}(t, k, g) \leq \frac{\ln k + t \ln g}{\ln g^{t-1}} \leq g^t (\ln k + t \ln g).
\]

For fixed \( g \) and \( t \), as \( k \to \infty \):

\[
c_{\text{CCAN}}(t, k, g) \leq g^t \ln k + O(1).
\]

For fixed \( g \) and \( t \), the Local Lemma bound is constant, so the size of the array is determined completely by \( g \) and \( t \). On the other hand, the VARDENS bound is still logarithmic in \( k \); thus, we have an example of a family of hypergraphs where the Local Lemma bound is substantially better than the VARDENS bound.

We note that, from Theorem 3.4.18, if \( t \mid k \), then \( c_{\text{CCAN}}(t, k, g) = g^t \). If not, by Corollary 2.2.37 we have a homomorphism \( H^t_c \to H^t_{c + (k \mod t)} \), i.e. to a covering array over \( k' = t + (k \mod t) \) factors, where \( 0 \leq k' \leq 2t - 1 \). The appearance of the term \( \ln(2t - 1) \) in the bound derived from the Local Lemma indicates its ability to recognize this property from Corollary 2.2.37.

The bound derived from VARDENS is unable to see this, and the guaranteed array size naively still varies based on the number of factors despite the fact that the structure of the hypergraph is, locally, completely independent from the number of factors. We do note, however, that actually running the VARDENS algorithm for \( c_{\text{CCA}}(N; t, k, g) \) does seem to generate arrays where the array size is independent of \( k \). We illustrate that this is the case for \( c_{\text{CCA}}(N; 3, k, 3) \) in Figure 6.2, which shows that as \( k \) grows, \( N \) remains constant.

6.2 VCA over Triangulations of the Sphere

In Section 2.2.9, we defined the triangulation hypergraphs of the sphere, and in Theorem 3.5.2, we showed that for any triangulation of the sphere \( \Delta \) and \( g \geq 1 \), \( \text{VCAN}(\Delta, g) = g^3 \).

We have the following theorem regarding intersections of facets.

Theorem 6.2.1. Let \( \Delta \) be a triangulation hypergraph of the sphere with facet set \( \Lambda \). If \( e = \{x, y, z\} \in \Lambda \):

\[
|\{e' \in \Lambda \setminus \{e\} : e \cap e' \neq \emptyset\}| = d^3_\Delta(x) + d^3_\Delta(y) + d^3_\Delta(z) - 6.
\]
Proof. For vertex $x$, there are $d_\Delta^3(x)$ facets containing $x$. One of these is $e$, so $e$ intersects $d_\Delta^3(x) - 1$ facets through $x$ (similarly for $y$ and $z$). However, one of these facets shares the dimension-1 face $\{x, y\}$ with $e$, so we have counted it twice: once in the facets intersected through $x$, and once in the facets intersected through $y$. This is also the case for $\{x, z\}$ and $\{y, z\}$, so we counted three facets exactly twice. Thus, we have that the total number of facets intersecting $e$ is precisely:

$$(d_\Delta^3(x) - 1) + (d_\Delta^3(y) - 1) + (d_\Delta^3(z) - 1) - 3 = d_\Delta^3(x) + d_\Delta^3(y) + d_\Delta^3(z) - 6.$$  

Consider a triangulation hypergraph of the sphere $\Delta$ with facet set $\Lambda$ over $v$ vertices. We note that the degrees of vertices of $\Delta$ are unlikely to be regular, and as a result, the intersection counts amongst the facets of $\Delta$ are unlikely to be equal. Thus, analyzing the triangulation using the symmetric Local Lemma will result in a bound that is not
tight and will certainly depend on the size of the triangulation. Indeed, even if we use
the more general forms of the Local Lemma, the bound will depend on the intersection
sets of each facet, and thus will grow with the number of facets. Hence, the triangulation
hypergraphs represent a case where, unlike the case for consecutive covering arrays, the
Local Lemma is unable to capture the behaviour of the homomorphism \( \Lambda \to K_3^3 \) which
shows that the covering array number is fully independent of the triangulation itself.
This is also the case with the bound from \textsc{VarDens}:

**Theorem 6.2.2** (\textsc{VarDens} bound for triangulation hypergraphs of the sphere). If \( \Delta \)
is a triangulation of the sphere on \( v \) vertices and \( g \geq 1 \), then:

\[
VCAN(\Delta, g) \leq \frac{\ln(2v - 4) + 3\ln g}{\ln \frac{g^3}{g^3 - 1}} \leq g^3(\ln(2v - 4) + 3\ln g).
\]

*Proof.* This follows immediately from the \textsc{VarDens} bound of Theorem 5.2.3 and the
structural theorem of triangulations in Theorem 2.2.45. \( \square \)

In the case of consecutive covering arrays, despite the pessimistic upper bound guar-
anteed by \textsc{VarDens}, the results showed that the array size was not dependent on the
size of the hypergraph. This is not the case with triangulation hypergraphs of the sphere:
we generate random triangulations by using the recursive construction procedure of The-
orem 2.2.43 randomly performing one of O1, O2, or O3 whenever possible. We then ran
30 trials of \textsc{VarDens} for \( g \in \{2, 3, 5\} \), giving the results in Table 6.3. Our experiments
suggest that the size of the arrays increase in some function of \( g \).

<table>
<thead>
<tr>
<th>( g )</th>
<th>( N_{opt} )</th>
<th>( v = 4 )</th>
<th>( v = 11 )</th>
<th>( v = 100 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N_m, N_M, N_a )</td>
<td>( N_m, N_M, N_a )</td>
<td>( N_m, N_M, N_a )</td>
<td>( N_m, N_M, N_a )</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>27</td>
<td>27</td>
<td>34</td>
<td>31.17</td>
</tr>
<tr>
<td>5</td>
<td>125</td>
<td>142</td>
<td>152</td>
<td>146.30</td>
</tr>
</tbody>
</table>

Table 6.3: Results for running 30 trials of \textsc{VarDens} on random triangulations with \( v \)
vertices. \( N_{opt} \) is the known optimal value. \( N_m, N_M, \) and \( N_a \) are the minimum, maximum,
and mean array sizes, respectively, as found by \textsc{VarDens}. 
6.3 VarDens and the Symmetric Local Lemma

We now show that the bound determined by VarDens in fact corresponds almost exactly with the best possible application of the Local Lemma when we have no knowledge regarding the structure of the ASC.

Lemma 6.3.1. Let \( \Delta \) be an ASC with facet set \( \Lambda \). Then we have that, for \( e \in \Lambda \):

\[
|\{e' \in \Lambda \setminus \{e\} : e' \cap e \neq \emptyset\}| \leq |\Lambda| - 1.
\]

Proof. The result is obvious, as \( |\Lambda \setminus \{e\}| = |\Lambda| - 1 \). \( \square \)

If we have an ASC \( \Delta \), and our only knowledge of \( \Delta \) is the number of facets \( |\Lambda| \), then the best we can do with the Local Lemma is to use the symmetric Local Lemma, taking the upper bound \( d \leq |\Lambda| - 1 \).

Theorem 6.3.2. Let \( \Delta \) be any ASC with \( |\Lambda| \) facets, and let \( g \geq 1 \). If \( b_V \) is the bound on \( VCAN(\Delta, g) \) guaranteed by VarDens, and \( b_L \) is the bound on \( VCAN(\Delta, g) \) from the symmetric Local Lemma assuming no additional knowledge of the hypergraph, then we have that \( b_V - 1 < b_L < b_V \).

Proof. Let \( s = \max\{|e| : e \in \Lambda\} \). Then for \( n \geq 1 \), the probability that a randomly generated \( n \times s \) array \( M \) over \( \{0, \ldots, g-1\} \) is not covered is:

\[
p \leq g^s \left( \frac{g^s - 1}{g^s} \right)^n.
\]

Using Lemma 6.3.1, we have that the dependency count \( d \leq |\Lambda| - 1 \). Thus, from the symmetric Local Lemma, we have that the probability that \( M \) is a \( VCA(\Delta, g) \) is positive when:

\[
ep(d + 1) \leq 1
\]

\[
n \leq \frac{\ln |\Lambda| + s \ln g + 1}{\ln \frac{g^s}{g^s - 1}}
\]

From Theorem 5.2.3, we have that the VarDens bound is:

\[
n \leq \frac{\ln |\Lambda| + s \ln g}{\ln \frac{g^s}{g^s - 1}} + 1.
\]

The inequality immediately follows. \( \square \)
6.4 VCA over Hypergraphs with Cliques

Cohen et al. [22] use simulated annealing to find upper bounds on the sizes of variable strength covering arrays over hypergraphs that have certain restrictions. The hypergraphs that they examine have a base strength, and then disjoint cliques of factors of higher strengths. This disjoint property makes it a simple exercise to determine the exact block intersection counts for these particular hypergraphs, and identify upper bounds on probabilities for edges over the base strength and within the higher-strength cliques. Since, using this model, we can construct hypergraphs where the difference in the probability of bad events and the block intersection counts between areas of base strength and maximum strength can be made to be arbitrarily large, the symmetric Local Lemma can be made to be an arbitrarily poor approximation. Thus, these hypergraphs are ideal candidates to be analyzed using the more general forms of the Local Lemma.

In order to investigate the effects of using the nonsymmetric forms of the Local Lemma, we chose one particular problem on which to focus our efforts: let \( \{K_2, K_3\} \) be a partition of a set of factors \( \{0, \ldots, k - 1\} \) with sizes \( |K_2| = k_2 \) and \( |K_3| = k_3 \), and define the ASC \( \Delta \) over vertices \( K_2 \cup K_3 \), with strength 2 across all vertices, and strength 3 across the vertices in \( K_3 \):

\[
\Delta = \left( \begin{array}{c} K_2 \\ 2 \end{array} \right) \cup \left( \begin{array}{c} K_3 \\ 3 \end{array} \right) \cup \{ \{i, j\} : i \in K_2, j \in K_3 \}.
\]

Thus, we have that \( \Delta \) has three different types of edges, namely:

- \( e_1 \): size-2 edges fully within \( K_2 \), of which there are \( \binom{k_2}{2} \),
- \( e_2 \): size-2 edges with one end in \( K_2 \) and one end in \( K_3 \), of which there are \( k_2k_3 \),
- \( e_3 \): size-3 edges fully within \( K_3 \), of which there are \( \binom{k_3}{3} \).

Let \( g \) be the number of levels, taken to be uniform across all factors, and write \( k = k_2 + k_3 \) to represent the total number of factors. For a given \( n \), our strategy is to fill an \( n \times k \) array with entries taken with uniform probability from \( \mathbb{Z}_g \), and then we define one bad event for each facet of \( \Delta \), representing one or more interactions over the factors in the face not being covered in the array. Let \( E_i \) be the set of bad events defined on the edges of type \( e_i \). In order to apply the different forms of the Local Lemma, we first calculate the probability of each bad event. From Corollary 6.1.8, we have:
1. $A \in E_1$: $p(A) \leq g^2 \left( \frac{g^2 - 1}{g^2} \right)^n$.

2. $A \in E_2$: $p(A) \leq g^2 \left( \frac{g^2 - 1}{g^2} \right)^n$.

3. $A \in E_3$: $p(A) \leq g^3 \left( \frac{g^3 - 1}{g^3} \right)^n$.

We also need to calculate the dependencies between the bad events. As with probabilities, these depend on the type of bad event, and for each type of bad event, we determine the number of bad events of each type with which it is dependent.

1. $A \in E_1$ is represented by an edge $e = \{u, v\} \subseteq K_2$. In order to count the number of $E_1$ events dependent with $A$, we count the number of edges of type $e_1$ that intersect $e$. These are all edges of the form:

$$\{\{u, w\} : w \in K_2 \setminus \{u, v\}\} \cup \{\{x, v\} : x \in K_2 \setminus \{u, v\}\}.$$

There are $2(k_2 - 2)$ such edges.

In order to count the number of $E_2$ events dependent with $A$, we count the number of edges of type $e_2$ that intersect $e$:

$$\{\{u, w\} : w \in K_3\} \cup \{\{v, w\} : w \in K_3\}.$$

There are $2k_3$ such edges.

Finally, there are no edges of type $e_3$ that intersect $e$, since all $e_3$ edges are in $K_3$, which is disjoint from $K_2$.

2. $A \in E_2$ is represented by an edge $e = \{u, v\}$, $u \in K_2$, $v \in K_3$. In order to count the number of $E_1$ events dependent with $A$, we count the number of edges of type $e_1$ that intersect $e$. These have the form:

$$\{\{u, w\} : w \in K_2 \setminus \{u\}\}.$$

There are $k_2 - 1$ such edges.

In order to count the number of $E_2$ events dependent with $A$, we count the number of edges of type $e_2$ that intersect $e$:

$$\{\{u, w\} : w \in K_3 \setminus \{v\}\} \cup \{\{x, v\} : x \in K_2 \setminus \{u\}\}.$$

There are $k_2 + k_3 - 2$ such edges.
In order to count the number of \( E_3 \) events dependent with \( A \), we count the number of edges of type \( e_3 \) that intersect \( e \):

\[
\{ \{ v, w, x \} : \{ w, x \} \subseteq K_3 \setminus \{ v \} \}.
\]

There are \( \binom{k_3 - 1}{2} = (k_3 - 1)(k_3 - 2)/2 = (k_3^2 - 3k_3 + 2)/2 \) such edges.

3. \( A \in E_3 \) is represented by an edge \( e = \{ u, v, w \} \subseteq K_3 \). There are no edges of type \( e_1 \) intersecting \( e \) since all \( e_1 \) edges are exclusively in \( K_2 \).

In order to count the number of \( E_2 \) events dependent with \( A \), we count the number of edges of type \( e_2 \) that intersect \( e \). These have the form:

\[
\{ \{ x, u \} : x \in K_2 \} \cup \{ \{ x, v \} : x \in K_2 \} \cup \{ \{ x, w \} : x \in K_2 \}.
\]

Thus, there are \( 3k_2 \) such edges.

In order to count the number of \( E_3 \) events dependent with \( A \), we count the number of edges of type \( e_3 \) that intersect \( e \). These have the form:

\[
\{ \{ u, x, y \} : \{ x, y \} \subseteq K_3 \setminus \{ u, v, w \} \} \cup \{ \{ v, x, y \} : \{ x, y \} \subseteq K_3 \setminus \{ u, v, w \} \} \\
\cup \{ \{ w, x, y \} : \{ x, y \} \subseteq K_3 \setminus \{ u, v, w \} \} \cup \{ \{ u, x \} : x \in K_3 \setminus \{ u, v, w \} \} \\
\cup \{ \{ u, w \} : x \in K_3 \setminus \{ u, v, w \} \} \cup \{ \{ v, w \} : x \in K_3 \setminus \{ u, v, w \} \}.
\]

The size of this set is thus:

\[
\binom{3}{1} \binom{k_3 - 3}{2} + \binom{3}{2} \binom{k_3 - 3}{1} = \frac{3k_3^2 - 15k_3 + 18}{2}.
\]

The results are listed in Table 6.4

<table>
<thead>
<tr>
<th></th>
<th>( e_1 )</th>
<th>( e_2 )</th>
<th>( e_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e_1 )</td>
<td>( 2k_2 - 4 )</td>
<td>( 2k_3 )</td>
<td>( 0 )</td>
</tr>
<tr>
<td>( e_2 )</td>
<td>( k_2 - 1 )</td>
<td>( k_2 + k_3 - 2 )</td>
<td>( \frac{k_3^2 - 3k_3 + 2}{2} )</td>
</tr>
<tr>
<td>( e_3 )</td>
<td>( 0 )</td>
<td>( 3k_2 )</td>
<td>( \frac{3k_3^2 - 15k_3 + 18}{2} )</td>
</tr>
</tbody>
</table>

Table 6.4: Bad event dependencies for \( k_2 + k_3 \) factors, strength 2 across all factors, and strength 3 across \( k_3 \) factors.

We chose to begin by focusing on the case \( k_2 = 11, k_3 = 4 \), which was studied via simulated annealing in [22], with an array size of 27 found. Let \( A \) be our set of bad
events, and for each \( A \in \mathcal{A} \), let \( \Gamma(A) \subset \mathcal{A} \) be the set of bad events dependent on \( A \).

To apply the symmetric Local Lemma to this problem, we must find an upper bound on both the number of dependencies and the probability of bad events. We have, as calculated from Table 6.4, that for \( A_1 \in E_1 \), \(|\Gamma(A_1)| = 26\); for \( A_2 \in E_2 \), \(|\Gamma(A_2)| = 26\); and for \( A_3 \in E_3 \), \(|\Gamma(A_3)| = 36\). Hence, we use the upper bound \( d \leq \max(26, 26, 36) = 36 \).

Thus, all bad events are avoided when:

\[
ep(d + 1) \leq 1
\]

\[
37eg^3 \left( \frac{g^3 - 1}{g^3} \right)^n \leq 1
\]

\[
n \geq \frac{3\ln g + \ln 37 + 1}{3\ln g - \ln(g^3 - 1)}.
\]

(6.1)

We thus have that coverage is guaranteed for any \( n \) such that Equation 6.1 holds.

The general form of the Local Lemma, as in Theorem 6.1.3, requires us to find a function \( x : \mathcal{A} \to (0, 1) \) such that, for each \( A \in \mathcal{A} \):

\[
\Pr(A) \leq x(A) \prod_{B \in \Gamma(A)} (1 - x(B)).
\]

If such a function \( x \) exists, then the Local Lemma guarantees that the probability that all bad events can be avoided is nonzero. Additionally, we work under the assumption that for \( A, A' \in E_i \), \( x(A) = x(A') = x_i \) for some \( x_i \in (0, 1) \), i.e. two bad events of the same type had the same value with respect to \( x \). This assumption gives us the following system of inequalities:

\[
g^2 \left( \frac{g^2 - 1}{g^2} \right)^n \leq x_1(1 - x_1)^{18}(1 - x_2)^8
\]

\[
g^2 \left( \frac{g^2 - 1}{g^2} \right)^n \leq x_2(1 - x_1)^{10}(1 - x_2)^{13}(1 - x_3)^3
\]

\[
g^3 \left( \frac{g^3 - 1}{g^3} \right)^n \leq x_3(1 - x_2)^{33}(1 - x_3)^3
\]

(6.2)

There does not seem to be any immediate way to solve the system using symbolic methods, so we attempted to do so computationally using OpenOpt \( \Pi \) on a system of four variables \( (x_1, x_2, x_3, n) \), specifying the above constraints and \( 0 < x_i < 1 \) for \( i = 1, 2, 3 \), with the goal of minimizing \( n \). We provide the results obtained for \( g = 2, \ldots, 10 \) in Table 6.5, comparing them to the results that would be obtained from applying the symmetric Local Lemma to the problem as in Equation 6.1. As expected, the results
obtained from the general Local Lemma showed significant improvement over those from 
the symmetric Local Lemma, with an average improvement of 24.71% and a median 
improvement of 23.65%. We note, however, that considerable work went into optimizing: 
due to the fact that the equations are highly nonlinear, and the search space likely con-
tains many local minima for $n$, just to obtain feasible solutions, we needed to considerably 
experiment with the optimizer with respect to starting position and add constraints to 
nudge the optimizer in the right direction at times during the optimization. Thus, the 
process could not be automated, and was highly intensive, requiring significant manual 
experimentation and interaction.

During optimization, the values obtained for $x_1, x_2, x_3, n$ resulted, in all cases, in 
a bound meeting Equation 6.2. Thus, since this equation imposed restrictions on the 
solutions of the system, we try maximizing the right hand side in order to give the 
equation the most room to grow. We select $x_3$ to maximize the right hand side by taking 
the derivative with respect to $x_3$:

$$
\frac{d}{dx_3} x_3(1-x_2)^3(1-x_3)^3 = (1-x_2)^3((1-x_3)^3-3x_3(1-x_3)^2) = (1-x_2)^3(1-x_3)^2(1-4x_3).
$$

Since $0 < x_3 < 1$, setting this equation to 0 to maximize $x_4$ gives $1 - 4x_3 = 0$, or 
x_3 = 0.25. This is supported by the values returned through optimization in Table 6.5, 
where the median value for $x_3$ is 0.24498101981, and the mean is 0.238913295055554, 
which are within 2.02% and 4.44% of 0.25. We then optimized on the system of equations 
with $x_3 = 0.25$, with the results as in Table 6.6. Despite the solver reporting that the 
solution was feasible, the results reported here, for $g \in \{3, 4, 5, 6, 8, 9, 10\}$, violated the 
third constraint, with the left hand side being less than 0.1% greater than the right hand 
side. This was likely due to floating point number estimation.

Additionally, as $x_1$ and $x_2$ approach 0 as $g$ increases, by looking at the third constraint 
and taking $1 - x_2 = 1$, we have that the value of $n$ behaves as follows:

$$
n \approx \frac{\ln 0.25 + 3 \ln 0.75 - 3 \ln g}{\ln (g^3 - 1) - 3 \ln g}.
$$

In general, the $x$ function for the General Local Lemma can be quite difficult to 
produce [48], so as an alternative, we turn to the asymmetric Local Lemma, as in Theo-
rem 6.1.4, which does not require us to explicitly do so. This gives the following system
<table>
<thead>
<tr>
<th>$g$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$n_g$</th>
<th>$n_s$</th>
<th>$p_{s,g}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.14284942633</td>
<td>0.00446196204405</td>
<td>0.310907856123</td>
<td>33.7942070729</td>
<td>50.103270271</td>
<td>32.5508956000019</td>
</tr>
<tr>
<td>3</td>
<td>1.33733174342e-06</td>
<td>0.00013350888259</td>
<td>0.187895413312</td>
<td>148.302981958</td>
<td>209.504135265</td>
<td>29.212384390211287</td>
</tr>
<tr>
<td>4</td>
<td>0.716159052016</td>
<td>4.30427533239e-05</td>
<td>0.244948101981</td>
<td>407.024758097</td>
<td>556.870854131</td>
<td>26.90859019136577</td>
</tr>
<tr>
<td>5</td>
<td>0.773241845215</td>
<td>8.04437280583e-08</td>
<td>0.234016578575</td>
<td>881.513683589</td>
<td>1175.1780223</td>
<td>24.98892364718111</td>
</tr>
<tr>
<td>6</td>
<td>0.891532751222</td>
<td>5.72719367592e-07</td>
<td>0.250637004412</td>
<td>1643.10293857</td>
<td>2152.02144539</td>
<td>23.648393835023843</td>
</tr>
<tr>
<td>7</td>
<td>6.3504223899e-15</td>
<td>2.52218931347e-15</td>
<td>0.277122690176</td>
<td>2777.33622928</td>
<td>3578.65952098</td>
<td>22.39171642348813</td>
</tr>
<tr>
<td>8</td>
<td>2.06633383427e-15</td>
<td>3.41234024849e-10</td>
<td>0.185287026146</td>
<td>4367.67557995</td>
<td>5549.38579047</td>
<td>21.29443248565201</td>
</tr>
<tr>
<td>9</td>
<td>2.31909327559e-26</td>
<td>4.84898670973e-26</td>
<td>0.249822207931</td>
<td>6440.6824068</td>
<td>8161.08673166</td>
<td>21.08057886685469</td>
</tr>
<tr>
<td>10</td>
<td>0.991317327272</td>
<td>2.7684395883e-17</td>
<td>0.209475086894</td>
<td>9171.6416374</td>
<td>11512.9128947</td>
<td>20.336045957385913</td>
</tr>
</tbody>
</table>

Table 6.5: Comparison of symmetric versus general Local Lemma, where $n_g$ is the upper bound on $n$ from the general Local Lemma, $n_s$ is the upper bound on $n$ from the symmetric Local Lemma, and $p_{s,g}$ is the percentage improvement seen in using the general Local Lemma over the symmetric one.
The Lovász Local Lemma and Probabilistic Bounds

Table 6.6: Comparison of the effects of setting $x_3 = 0.25$ in the constraints from the general Local Lemma and then optimizing versus performing pure optimization. We note that for $g \in \{3, 4, 5, 6, 8, 9, 10\}$, even through constraints, the optimizer returned values for $x_1, x_2$ such that the third equation was not satisfied, with the left-hand side being less than 0.1% greater than the right-hand side. The upper bound on $n$ from the modified constraints is $n_f$, whereas $n_g$ is the upper bound on $n$ from the general Local Lemma. The column $p_{g,f}$ gives the improvement of $n_f$ over $n_g$ as a percentage of $n_g$.

<table>
<thead>
<tr>
<th>$g$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$n_f$</th>
<th>$n_g$</th>
<th>$p_{g,f}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.00120148574882</td>
<td>0.00103686860573</td>
<td>32.67430366</td>
<td>33.7942070729</td>
<td>3.31389166932</td>
</tr>
<tr>
<td>3</td>
<td>0.496760979291</td>
<td>0.00059222982587</td>
<td>147.447453469</td>
<td>148.302981958</td>
<td>0.576878817745</td>
</tr>
<tr>
<td>4</td>
<td>0.499129529052</td>
<td>0.0001489212413</td>
<td>32.67430366</td>
<td>33.7942070729</td>
<td>3.31389166932</td>
</tr>
<tr>
<td>5</td>
<td>0.499129529052</td>
<td>0.0001489212413</td>
<td>147.447453469</td>
<td>148.302981958</td>
<td>0.576878817745</td>
</tr>
<tr>
<td>6</td>
<td>0.499129529052</td>
<td>0.0001489212413</td>
<td>32.67430366</td>
<td>33.7942070729</td>
<td>3.31389166932</td>
</tr>
<tr>
<td>7</td>
<td>0.499129529052</td>
<td>0.0001489212413</td>
<td>147.447453469</td>
<td>148.302981958</td>
<td>0.576878817745</td>
</tr>
<tr>
<td>8</td>
<td>0.499129529052</td>
<td>0.0001489212413</td>
<td>32.67430366</td>
<td>33.7942070729</td>
<td>3.31389166932</td>
</tr>
<tr>
<td>9</td>
<td>0.499129529052</td>
<td>0.0001489212413</td>
<td>147.447453469</td>
<td>148.302981958</td>
<td>0.576878817745</td>
</tr>
<tr>
<td>10</td>
<td>0.499129529052</td>
<td>0.0001489212413</td>
<td>32.67430366</td>
<td>33.7942070729</td>
<td>3.31389166932</td>
</tr>
</tbody>
</table>

Equation 6.3 dictates the lower bound:

$$n \geq \frac{3 \ln g + 2 \ln 2}{3 \ln g - \ln (g^3 - 1)}.$$ 

Taking the upper bound as given by the symmetric Local Lemma in Equation 6.1, we performed a binary search to find the smallest value $n'$ satisfying the system of inequalities. The results are given in Table 6.7. The asymmetric Local Lemma gives results that are slightly worse than those given by optimizing the general Local Lemma constraints: on average, they are within 2.76% (median: 2.66%) of those given by the general Local
Lemma. While this difference is small, the improvement that the use of the asymmetric Local Lemma gives over the use of the symmetric Local Lemma is considerably more significant: the asymmetric Local Lemma gives, on average, an improvement of 22.63% (median: 21.29%) over the symmetric Local Lemma.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>34.3846263617</td>
<td>33.7942070729</td>
<td>1.74710206257</td>
<td>50.103270271</td>
</tr>
<tr>
<td>3</td>
<td>153.172068422</td>
<td>148.302981958</td>
<td>3.28320199615</td>
<td>209.504135265</td>
</tr>
<tr>
<td>4</td>
<td>421.871910683</td>
<td>407.024758097</td>
<td>3.64772714451</td>
<td>556.87085413</td>
</tr>
<tr>
<td>5</td>
<td>910.491167915</td>
<td>881.513683589</td>
<td>3.28724157837</td>
<td>1175.1780223</td>
</tr>
<tr>
<td>6</td>
<td>1693.86682186</td>
<td>1643.102981958</td>
<td>3.28724157837</td>
<td>2152.02144539</td>
</tr>
<tr>
<td>7</td>
<td>2850.50114291</td>
<td>2777.33622928</td>
<td>2.63435564116</td>
<td>3578.65952098</td>
</tr>
<tr>
<td>8</td>
<td>4461.93147599</td>
<td>4367.67557995</td>
<td>2.15803335927</td>
<td>5549.38579047</td>
</tr>
<tr>
<td>9</td>
<td>6612.2879016</td>
<td>6440.6824068</td>
<td>2.66439926643</td>
<td>8161.08673166</td>
</tr>
<tr>
<td>10</td>
<td>9387.96495097</td>
<td>9171.6416374</td>
<td>2.35861061871</td>
<td>11512.9128947</td>
</tr>
</tbody>
</table>

Table 6.7: Binary searching with the asymmetric Local Lemma constraints, where \( n_a \) is the value obtained by binary search over the asymmetric Local Lemma constraints, \( n_g \) is the value from optimizing the General Local Lemma constraints, and \( n_s \) is the value obtained from applying the symmetric Local Lemma. The columns \( p_{a,g} \) and \( p_{s,a} \) represents the percentage of how much worse \( n_a \) is than \( n_g \), and how much better \( n_a \) is than \( n_s \) respectively.

The bounds given by the general Local Lemma, which are valid, unlike those when manual optimization was performed, are better than those as given by VARDENS. However, actually running VARDENS on the hypergraph in this section generates significantly smaller arrays than the Local Lemma bound. In Figure 6.3, we show the behaviour of the VARDENS bound, the general Local Lemma bounds, and the results from actually running the VARDENS algorithm, with linear regression analysis on the general Local Lemma bound to show its expected behaviour.

### 6.5 Analysis of Results

The symmetric Local Lemma of Theorem 6.1.1 for problems that demonstrate symmetry with respect to bad events in terms of their probability of occurring and their number.
of dependencies with other bad events, provides us with a nonconstructive technique to symbolically calculate upper bounds for a variety of problems. In the study of variable strength covering arrays, this is particularly effective when there is some regularity to the structure of the underlying hypergraph, such as is the case with combinatorial designs.

If, on the other hand, the problem lacks symmetry, as is the case with the types of variable strength covering arrays studied by Cohen et al. in [22], there is benefit in using a more general form of the Local Lemma; the upper bounds on the size of the arrays will be considerably lower, especially when the level of asymmetry is high, e.g. when the probabilities of bad events differ considerably, or the sizes of the dependency sets vary significantly. The general Local Lemma gave the best results, but can be difficult to use for these types of problems, as we saw in our experiments, and required the use of nonlinear optimization. The asymmetric Local Lemma, on the other hand, can be optimized by simply binary searching on $n$, the size of the array, which is considerably easier, but the upper bounds found were not as good as with the general Local Lemma; however, this decrease in quality was small. Due to the ease of use of this technique, it is a worthwhile tool to investigate. Thus, when a problem lacks symmetry with regards to probability bounds of bad events or dependency set sizes, we recommend the use of the asymmetric Local Lemma.
Figure 6.3: A comparison of the VARDENS bound (upper line), the general Local Lemma bounds (middle line), and the actual results from running VARDENS on a hypergraph with \( k = 15 \) vertices, strength 2 throughout, and full strength 3 on four of the vertices. The horizontal axis is the number of levels and the vertical axis represents the natural logarithm of the size of the arrays.
Chapter 7

Conclusion and Open Questions

We now examine the major contributions of the thesis and then enumerate the open questions that arose from this work.

7.1 Major Contributions and Summary

This research comprised the first comprehensive study of variable strength covering arrays. We defined variable strength covering arrays and orthogonal arrays, explored the theory behind them and motivated their importance by illustrating their use in combinatorial testing. The thesis consists of three main components, each comprising a technique for working with VCA problems. The first of these encompasses the theory of VCA and direct and recursive constructions for creating VCA (Chapters 3 and 4); the second is a greedy algorithm called VarDens for building VCA with an upper bound guarantee on the size of the resultant arrays, which serves as a general upper bound for VCA (Chapter 5); and the third entails nonconstructive probabilistic arguments based on the Local Lemma for determining upper bounds on the size of VCA (Chapter 6). We outline the results of our research in each of these classes below.

7.1.1 Theory of VCA and Constructions

In Chapter 3, we delved into the theory of VCA and discussed some recursive and direct constructions. We began by demonstrating how homomorphisms and strong hypergraph colourings serve as an invaluable tool for finding VCA: in Theorem 3.1.1 we show that if we have two hypergraphs \( G, H \) such that \( G \rightarrow H \), then for any value \( g > 0 \),...
VCAN\((G, g)\) \leq \text{VCAN}(H, g)\) as the columns of the VCA over \(G\) can be pulled back through the homomorphism.

The connection between hypergraphs and covering arrays motivates the introduction and investigation of a family of ASC that naturally arise in the study of covering arrays, namely the qualitative independence hypergraphs VQIH\((n, g)\) and PQIH\((n, g)\), as presented for the first time in Section 3.2. The hypergraph PQIH\((n, g)\) consists of all partitions of an \(n\)-set into at most \(g\) nonempty subsets, with hyperedges dictated by qualitative independence of partitions, which is exactly the property required by the columns of VCA. We then established the precise equivalence between homomorphisms \(H \to \text{PQIH}(n, g)\) and VCA\((n; H, g)\) as shown in Theorem 3.2.9. Furthermore, for covering arrays, there is an exact correspondence between CA\((n; t, k, g)\) and the \(t\)-cliques of size \(k\) in PQIH\((n, g)\), as shown in Corollary 3.2.10.

Given their importance, we determined many important hypergraph-theoretic properties of these graphs, which provide insight into their structure, including the number of vertices (Proposition 3.2.11); the spectrum of hyperedge size (Theorem 3.2.14), and the number of hyperedges of each size in the spectrum (Theorem 3.2.19); the \(t\)-degree of each vertex (Theorem 3.2.18); identification of isolated vertices and connected components (Theorems 3.2.17, 3.2.20); and the diameter of the hypergraphs (Corollary 3.2.24). We then discussed the importance of cores of hypergraphs, and uniformity of partitions and vectors. We then proposed that the uniform subASC of PQIH\((n, g)\) is actually its core.

We also gave several different tools for use in the construction of VCA. In Section 3.4.1, we explained two operations that can be used on VCA over hypergraphs. First, we have the split operation of Theorem 3.4.1 if \(H\) is a weighted hypergraph and we have a vertex \(v \in V(H)\) of weight \(ab\) for \(a, b > 1\), then we can split \(v\) into two vertices, \(v'\) and \(v''\) of weights \(a\) and \(b\) respectively, and the VCA over these two hypergraphs are equivalent. In Theorem 3.4.3, we showed that if, for a weighted hypergraph \(H\), we have two vertices, say \(v'\) and \(v''\) of weights \(a\) and \(b\) respectively, then we can join them into one vertex \(v\) of weight \(ab\), and the VCA over this new hypergraph translates to a VCA over the original hypergraph. Using these two operations, along with the level-dropping operation from covering arrays (where we can reduce a factor’s number of levels), provides a flexible framework for analyzing a variable strength covering array problem in a number of ways, and allows us to extend known results, as shown in three examples.

The second tool we detailed is a divide-and-conquer approach to building VCA that uses cut-vertices or cut-cliques, and is explained in Section 3.4.2. In Theorem 3.4.11, we
showed that if a hypergraph $H$ has a cut vertex $v$, we may build covering arrays individually on the components that are separated by $v$ (including $v$ in each component). If this can be done such that the column over $v$ is uniform or almost-uniform in each component, the components can then be assembled to provide the full VCA, thus guaranteeing that the size of the full VCA is bounded by the size of its components as separated by $v$. We further generalized the technique to cut cliques, which requires uniformity, but is otherwise the same in execution.

We then looked at direct constructions for VCA over several families of ASC. In Section 3.4.3, we provided a direct construction for optimal consecutive and cyclic consecutive covering arrays for many combinations of parameters, identifying the remaining cases where the covering array number has yet to be established. We also determined the strong chromatic number for the ASC $H_{c}^{kt}$, as given in Theorem 3.4.23. We fully solved the problem of determining the covering array number and constructing optimal covering arrays for triangulation hypergraphs of the sphere in Section 3.5 using homomorphisms.

The covering array number $\text{CAN}(G, (g_{0}, \ldots, g_{s}))$ has been established for any bipartite graph $G$ in \cite{17, 67} and is simply the maximum product of the number of levels over the edges of the graph. This motivated us to look at similar problems for 3-colourable hypergraphs. In particular, we examined one family of strongly 3-chromatic hypergraphs called 2-trees in Section 3.4.4, and provided an algorithm we used to construct VCA over these hypergraphs, which suggests that they analogously have size determined in the same way as in the bipartite graph problem.

In Chapter 4, we derived an important new construction for variable strength orthogonal arrays based on the use of linear feedback shift register sequences. By Theorem 4.2.1, we can construct a VOA by taking all $q^{d} - 1$ of the length-$\frac{q^{d} - 1}{q - 1}$ subwords from the linear feedback shift register, along with a row of all zeros. This VOA covers all 3-sets but a projective plane, so its hypergraph is precisely the complement of the projective plane, which is also a combinatorial design.

We then showed, in Theorem 4.2.10, that the result of appending to the VOA a copy of all of the mirror images of the rows from the linear feedback shift register sequence is a full strength-3 covering array with parameters $\text{CA}(2(q^{3} - 1) + 1; 3, q^{3} + q + 1, q)$. In \cite{25}, Colbourn provides a list of the best known covering array sizes for $2 \leq t \leq 6$ and $2 \leq g \leq 25$. For all values of $g \neq 2, 3, 6$, our technique results in substantial improvements on the best known covering array bounds, as shown in Table 4.1. We then apply the recursive covering array constructions of Section 2.2.3 along with fusion and factor increasing to extend these results to larger ranges of levels $g$ and factors $k$, again...
improving on the best known upper bounds for covering array sizes for many values of \( g \) and \( k \).

We described our attempts to generalize this construction to higher degree \( m > 3 \) primitive polynomials in the hope of constructing higher strength CA using this technique. Consider a degree-\( m \) primitive polynomial \( f \) over \( \mathbb{F}_q \), and take all \( q^m - 1 \) subwords of length \( \frac{q^m - 1}{q-1} \) from the linear feedback shift register sequence to obtain a \( (q^m - 1) \times \frac{q^m - 1}{q-1} \) array \( M \), as we originally did for \( m = 3 \). Let \( \alpha \) be a root of \( f \) in the extension field \( \mathbb{F}_{q^m} \). In Theorem 4.4.1, we proved the relationship between coverage and linear independence, identifying precisely the uncovered \( m \)-sets of columns in \( M \): we showed that an \( m \)-set of columns \( c_{i_0}, \ldots, c_{i_{m-1}} \) in \( M \), \( 0 \leq i_j < \frac{q^m - 1}{q-1} \) for all \( j \) is uncovered if and only if the \( m \)-set of vectors \( \{ \alpha_{i_0}, \ldots, \alpha_{i_{m-1}} \} \) is linearly dependent.

We then attempted to develop \( M \) into a full strength-\( m \) covering array, as we were able to do for \( m = 3 \). Our attempts, however, showed that the experimentally determined number of permuted copies of the original VOA that we needed to append does not scale linearly with \( t \), and unlike the case \( t = 3 \), may depend on \( q \). Furthermore, we proposed a conjecture on how the construction might be generalized in Conjecture 4.4.3.

### 7.1.2 Greedy Algorithm for VCA

In Chapter 5, we designed an algorithm to find VCA over arbitrary weighted hypergraphs. This algorithm, called \textsc{VarDens}, is of practical importance as it can be immediately and directly used to construct test suites for covering array testing. The algorithm builds the array one row at a time, selecting the factors one at a time and fixing them to levels. This is done by using a concept of density [10, 11, 32] to predict the number of uncovered interactions a given row will be able to cover. This allows us to make a guarantee on the size of the resultant arrays, given in Theorem 5.2.3 which provides the first general upper bound for VCA over arbitrary ASC. If \( \Lambda \) are the facets of an ASC over \( \{0, \ldots, k-1\} \), for number of levels \( (g_0, \ldots, g_{k-1}) \), let \( m = \max\{\prod_{i \in E} g_i : E \in \Lambda\} \). Then:

\[
\text{VCAN}(\Lambda, (g_0, \ldots, g_{k-1})) \leq \frac{m(\ln |\Lambda| + \ln m)}{\ln \frac{m}{m-1}}.
\]

We proposed a number of algorithmic variants, and performed experiments on a number of different families of ASC to determine which variant performs better; we also compared our results to known results to examine the performance of \textsc{VarDens}.

Furthermore, the description of this algorithm clarifies the details of density-based
greedy algorithms for building covering arrays [10, 11, 32], which should assist programmers to implement these algorithms.

### 7.1.3 Probabilistic Methods for VCA Bounds

In Chapter 6, we used variants of the Lovász Local Lemma to find upper bounds on the sizes of covering arrays. This technique differs considerably from the linear feedback register sequence construction, which is a direct construction, and VARDENS, which is a heuristic, in that it is nonconstructive, i.e. it guarantees an upper bound without building an actual VCA. For a hypergraph $H$ over $k$ vertices, and a $g \geq 2$, the main idea behind this technique is to find a size $n$ such that a randomly generated $n \times k$ array is a VCA($n; H, g$) with positive probability.

We derived a framework in Theorem 6.1.9 for applying the symmetric Local Lemma, which works best when the hypergraph is $t$-uniform and the number of edges that an edge intersects is invariant. We then applied this framework to determine upper bounds on the sizes of VCA over combinatorial designs: for Steiner systems of the form $S(t-1, t, k)$, we were able to determine more precise bounds, as given in Theorem 6.1.12, which we contrasted with the bounds that we obtained with VARDENS as given in Theorem 6.1.14 to show that the results obtained from the Local Lemma were always better than those obtained from VARDENS. We showed that similar results also apply to general $s$-($k, t, \lambda$) designs in Theorems 6.1.16 and 6.1.17; however, due to the fact that these designs do not necessarily enjoy the same properties as Steiner systems, the bounds are less precise.

We proceeded to look at the effects of using the Local Lemma on determining a bound for the size of consecutive and cyclic consecutive covering arrays in Section 6.1.2. From Theorem 3.4.18, by using homomorphisms to construct these arrays, we know that the size of the arrays is largely independent of the number of factors $k$: the Local Lemma, as in Theorem 6.1.20, is able to recognize this fact, and even more impressively, is able to recognize the homomorphic behaviour demonstrated in Theorem 3.4.18. The VARDENS guaranteed bound, however, is unable to do so, and the size of arrays depends on $k$. Hence, this illustrates a case where the bound given by the Local Lemma can, for a value of $k$, be arbitrarily smaller than the bound from VARDENS. In practice, however, the arrays generated by VARDENS do not depend on $k$, as shown in Figure 6.2. On the other hand, triangulation hypergraphs of the sphere, as examined in Section 6.2, have a fixed optimal size completely independent of the hypergraph, as proved in Corollary 3.5.2 and dependent only on the number of levels. Neither a bound guaranteed by VARDENS
or the Local Lemma is able to capture this behaviour, and indeed, our experiments, summarized in Table 6.3, indicate that the sizes of arrays generated by VARDENS grow in some function of the size of the hypergraph. This suggests that using homomorphisms provide the best bound of the three techniques when possible.

We also show in Section 6.3 that the bound from VARDENS is naive in the sense that it is almost equivalent to a bound generated by the symmetric Local Lemma when we have no knowledge of the hypergraph other than the number of facets.

Finally, we examined a family of hypergraphs inspired by the definition of variable strength covering arrays given by Cohen et al. in [22], where the hypergraph has a base strength, and there is one or more disjoint sets of vertices over which we have uniform higher strengths. In this model, for a given bad event, the difference between the size of its dependency set and the maximum dependency set can be made to be arbitrarily large, and thus the symmetric Local Lemma performed poorly through the need to upper bound dependencies and probabilities. We thus explored and developed techniques for applying the general and asymmetric Local Lemma, along with numerical methods, to obtain better upper bounds on the sizes of the arrays; in our example problem, these bounds were substantially better than those determined by the symmetric Local Lemma.

## 7.2 Open Questions

We now list some open questions inspired by this research.

**Open Question 7.2.1.** From a practical point of view, it would be interesting to investigate the structure of ASC arising in different software testing applications. Any special structures of interest might motivate the direction of algorithms over ASC and be exploited to generate minimal array sizes.

**Open Question 7.2.2.** If, instead of using ASC or hypergraphs to model the interactions that we wish to cover in our variable strength covering arrays, we use general relational systems as described in Section 2.1.1 what flexibility does this give us? Is it possible, for example, to use them to define variable strength covering arrays with forbidden edges?

**Open Question 7.2.3.** What other hypergraph theoretic properties can we learn about VQIH(n, g), PQIH(n, g), UVQIH(n, g), and UPQIH(n, g)?

**Open Question 7.2.4.** Show that $\text{diam}(\overline{VQIH}(n, g)) = 2$. 
Open Question 7.2.5. Is UPQIH(n, g) a core?

Open Question 7.2.6. Are there homomorphisms PQIH(n, g) → UPQIH(n, g)?

Open Question 7.2.7. Is UPQIH(n, 2) the core of PQIH(n, 2)? If not, what is PQIH(n, 2)? What about PQIH(n, g)?

Open Question 7.2.8. What can be determined about the efficacy of using different split / join strategies (Theorems 3.4.1 and 3.4.3) along with level dropping?

Open Question 7.2.9. Let \( k > t > 0, g > 1 \), and write \( k = qt + r \) for \( 0 \leq r < t \). Find cCCAN(t, k, g) for the cases:

1. \( t + 2 \leq k \leq 2t - 1 \).
2. \( q + 1 \leq r < t, 2t + 3 \leq k \leq t(t - 1) - 1 \), and one of the following:
   
   (a) \( g \) not a prime power, or
   
   (b) \( g < t + \left\lceil \frac{r}{q} \right\rceil - 1 \).

Open Question 7.2.10. Let \( H \) be a 2-tree hypergraph over \( s \) vertices, and \( g_0, \ldots, g_{s-1} \geq 1 \). Find an algorithm that serves as a proof of Conjecture 3.4.27. Then this gives that:

\[
\text{VCAN}(H, (g_0, \ldots, g_{s-1})) = \max \left\{ \prod_{e \in E(H)} g_x g_y g_z : \{x, y, z\} \in E(H) \right\}.
\]

Open Question 7.2.11. Identify other families of strongly 3-chromatic hypergraphs \( H = (V, E) \), \( |V| = k \) that have the property:

\[
\text{VCAN}(H, (g_0, \ldots, g_{k-1})) = \max \left\{ \prod_{i \in e} g_i : e \in E \right\}.
\]

Open Question 7.2.12. Find a strongly 3-chromatic hypergraph \( H = (V, E) \), \( |V| = k \), such that:

\[
\text{VCAN}(H, (g_0, \ldots, g_{k-1})) > \max \left\{ \prod_{i \in e} g_i : e \in E \right\}.
\]

Alternatively, show that no such hypergraph exists.

Open Question 7.2.13. Let \( H \) be a \( k \)-tree over \( s \) vertices for \( k \geq 3 \), and let \( g_0, \ldots, g_{s-1} \geq 1 \). Determine whether or not it the case that:

\[
\text{VCAN}(H, (g_0, \ldots, g_{s-1})) = \max \left\{ \prod_{i \in e} g_i : e \in E(H) \right\}.
\]
Open Question 7.2.14. Prove or disprove the claim of Conjecture 4.4.3. Let $q$ be a prime power, and $f$ be a primitive polynomial of degree $d > 3$ over $\mathbb{F}_q$. The structure of the VOA determined by taking all $q^d - 1$ of the unique length-$\frac{q^d-1}{q-1}$ subsequences of the LFSR sequence defined by $f$ is described in Theorem 4.4.1. What is the smallest integer $c$ such that this VOA can be extended to a CA($c(q^d - 1) + 1; d, \frac{q^d-1}{q-1}, q$) by taking permutations of the columns? Is $c$ a function of $d$ or a function of both $q$ and $d$?

Open Question 7.2.15. Let $S$ be the 80 nonisomorphic STS(15). Let $g > 0$ be any integer. For all $B_1, B_2 \in S$, is $\text{VCAN}(B_1, g) = \text{VCAN}(B_2, g)$? If not, determine the spectrum:

$$S_g = \{\text{VCAN}(B, g) : B \in S\}.$$  

Can any specific substructures be indentified that cause the covering array numbers to differ?

Open Question 7.2.16. When the maximum and minimum facet sizes of an ASC differ considerably, the VARDENS algorithm may make poor choices in its execution, especially when there are nonempty intersections between large and small facets. What strategies can be used to avoid this?

Open Question 7.2.17. For an STS($k$) or SQS($k$) $B$, $s \geq 1$, and $g \geq 2$, create a random uniform $(sg) \times k$ array $M$ using the tiling technique of Donders and Godbole [39]. Apply the symmetric Local Lemma to determine if the probability that $M$ is a VCA($B, g$) is positive, and find an upper bound on $s$ which guarantees a positive probability, thus translating to an upper bound on the size of a variable strength covering array.

Open Question 7.2.18. Explore uses of the general forms of the Local Lemma - namely the general Local Lemma (Theorem 6.1.3), the asymmetric Local Lemma (Theorem 6.1.4), and the weighted Local Lemma (Theorem 6.1.5) to determine more precise upper bounds for VCA over hypergraphs containing varying strengths and edge intersection numbers. Is there any way to avoid numerical methods and apply these strictly through symbolic computation? Are there any families of hypergraphs where we can generate simple frameworks based on these forms of the Local Lemma to derive upper bounds on array sizes?
Bibliography


[64] Karen Meagher. Personal communication.


Glossary

**k-tree** A graph $G$ that can be constructed by starting with a $k$-clique, and then iteratively selecting a $k$-clique $C$ in $G$ and adding a new vertex $v$ with the edges to make $C \cup \{v\}$ a $(k+1)$-clique.

**k-tree hypergraph** A hypergraph $H$ arising from a $k$-tree $G$ with at least $k+1$ vertices, where $V(H) = V(G)$ and if $C$ is a $(k+1)$-clique in $G$, then $C \in E(H)$.

**n-section** $[H]_n$ For a hypergraph $H = (V, E)$, the $n$-section of $H$, $[H]_n$, is the hypergraph $(V, E')$ such that, for all $e \in E$, if $|e| < n$, then $e \in E'$, and otherwise, for all $f \in \binom{e}{n}$, $f \in E'$.

**abstract simplicial complex** A set $\Delta$ of sets that are closed under subsets: if $\Delta$ is an ASC and $X \in \Delta$, $Y \subset X$, then $Y \in \Delta$.

**automorphism (hypergraph)** An isomorphism from a hypergraph to itself: a bijective endomorphism.

**balanced incomplete block design** A BIBD($k$, $t$, $\lambda$) balanced incomplete block design is a 2-($k$, $t$, $\lambda$) design.

**clique** For a hypergraph $H$, a $t$-clique of size $n$ is a set of vertices of size $n$ such that every $t$-set over the vertices appears in $E(H)$.

**clique** For a hypergraph $H$, a $t$-coclique of size $n$ is a set of vertices of size $n$ such that no two vertices appear together in any edge of size $t$.

**colouring (graph)** A $c$-colouring of a graph $G$ is a homomorphism $G \rightarrow K_c$.

**colouring (strong)** A strong $c$-colouring of a hypergraph $H$ is a map $f : (V) \rightarrow \mathbb{Z}_c$ such that for all $e \in E(H)$, $|e| = |f(e)|$. 
**colouring (weak)** A weak c-colouring of a hypergraph $H$ is a map $f : V(H) \rightarrow \mathbb{Z}_c$ such that for all $e \in E(H)$, if $|e| > 1$, then $|f(e)| > 1$.

**connectivity** In a hypergraph $H$, two vertices $u, v$ are connected if there is a path from $u$ to $v$. We say that $H$ is connected if every pair of vertices in $H$ are connected. The connected components of $H$ are the maximal connected subhypergraphs of $H$.

**consecutive covering array** Let $H^{k,t} = (V,E)$ be the hypergraph with $V = \{0, \ldots, k-1\}$ and $E = \{\{i, i+1, \ldots, i+t-1\} : 0 \leq i \leq k-t\}$. A $\text{CCA}(N; t, k, g)$ consecutive covering array is a $\text{VCA}(N; H^{k,t}, g)$.

**core** A hypergraph (graph) in which every endomorphism is an automorphism.

**covering array** An $N \times k$ array with entries from a set of size $g$ such that, for a given $t$, every $N \times t$ subarray contains as rows all of the $g^t$ tuples at least $\lambda$ times. The array is said to have size $N$, strength $t$, $k$ factors, $g$ levels, and index $\lambda$, and is denoted $\text{CA}_\lambda(N; t, k, g)$, or, for $\lambda = 1$, $\text{CA}(N; t, k, g)$.

**covering array on graph** Let $G = (\{0, \ldots, k-1\}, E)$ be a graph. A $\text{CA}_\lambda(N; G, g)$ covering array is an $N \times k$ array with entries from a set of size $g$ such that, for every $e \in E$, the $N \times 2$ subarray over the columns in $e$ contains as rows all of the $g^2$ pairs at least $\lambda$ times. If $\lambda = 1$, we write $\text{CA}(N; G, g)$.

**covering ordered design** A $\text{COD}(N; t, k, g)$ covering ordered design is an $N \times k$ array over $S = \{0, \ldots, g-1\}$ with the property that every $N \times t$ subarray must contain all $t$-tuples from $S^t$ except those of the form $(i, \ldots, i)$.

**cycle** A path where we allow and require the first and last vertices to be the same.

**cyclic consecutive covering array** Let $H^{k,t}_c = (V,E)$ be the hypergraph with $V = \{0, \ldots, k-1\}$ and $E = \{\{i, (i+1) \mod k, \ldots, (i+t-1) \mod k\} : 0 \leq i \leq k-1\}$. A $\text{cCCA}(N; t, k, g)$ cyclic consecutive covering array is a $\text{VCA}(N; H^{k,t}_c, g)$.

**degree (hypergraph)** The degree of a vertex $v$ in a hypergraph is the number of edges containing $v$. The $t$-degree of a vertex $v$ is the number of edges of size $t$ containing $v$.

**design** An $s$-$(k, t, \lambda)$ design is a set system $(V, B)$ such that $|V| = k$, and $B$ is a multiset with elements from $\binom{V}{s}$ such that for every $S \in \binom{V}{s}$, there are exactly $\lambda$ sets $B \in B$ such that $S \subset B$. 
diameter The maximum of the shortest distances between all pairs of vertices.

**endomorphism (hypergraph)** A homomorphism from a hypergraph to itself.

**homomorphism (hypergraph)** For two hypergraphs (graphs) $G$ and $H$, a hypergraph (graph) homomorphism is a map $f : V(G) \rightarrow V(H)$ such that if $e \in E(G)$, then $f(e) \in E(H)$.

**hypertree** A hypergraph with no cycles.

**interaction** An assignment of levels to factors.

**isomorphism (hypergraph)** For two hypergraphs (graphs) $G$ and $H$, a hypergraph (graph) homomorphism is an isomorphism if the homomorphism is bijective, and $|E(G)| = |E(H)|$.

**mixed-level covering array** An MCA$_\lambda(N; t, (g_0, \ldots, g_{k-1}))$ mixed-level covering array is an $N \times k$ array where column $i$ has entries over $G_i = \{0, \ldots, g_i - 1\}$ such that for every $B = \{b_0, \ldots, b_{t-1}\} \in \{\{0, \ldots, k-1\}\}_t$, the $N \times t$ subarray over the columns in $B$ contains every tuple from $G_{b_0} \times \ldots \times G_{b_{t-1}}$ at least $\lambda$ times. If $\lambda = 1$, we write MCA$(N; t, (g_0, \ldots, g_{k-1}))$.

**orthogonal array** An $N \times k$ array with entries from a set of size $g$ such that, for a given $t$, every $N \times t$ subarray contains as rows all of the $g^t$ tuples precisely $\lambda = N/g^t$ times. The array is said to have size $N$, strength $t$, $k$ factors, $g$ levels, and index $\lambda$, and is denoted OA$_\lambda(t, k, g)$, or, for $\lambda = 1$, OA$(t, k, g)$.

**path** A walk in which all vertices and edges are distinct.

**rank (hypergraph)** The size of a largest edge in a hypergraph.

**regularity (hypergraph)** A hypergraph is regular if every vertex has the same degree.

**resolution** If $C$ is an $M \times k$ array over $g$ symbols, for an integer $1 \leq N \leq M$ and integers $M_1, \ldots, M_N$ such that $M = \sum_{i=1}^{N} M_i$, a resolution of $C$ with $N$ classes is a partition of $C$ into $N$ subarrays $C_1, \ldots, C_N$ such that for all $1 \leq i \leq N$, we have that $C_i$ is an $M_i \times k$ array where every symbol occurs at most once in each column.
resolvable covering ordered design  A RCOD($N; t, k, g$) is a COD($M; t, k, g$) that has a resolution with $N$ classes.

simplicity (design)  A design $(V, B)$ is simple if $B$ is a set.

skeleton  An $n$-skeleton of an ASC is the sub-ASC consisting of all faces of dimension at most $n$ (size at most $n + 1$).

Steiner quadruple system  An SQS($k$) Steiner quadruple system of order $k$ is a 3-($k, 4, 1$) design.

Steiner system  Let $k > t > s > 0$. An S($s, t, k$) Steiner system is an $s$-($k, t, 1$) design.

Steiner triple system  An STS($k$) Steiner triple system of order $k$ is a 2-($k, 3, 1$) design.

triangulation of the sphere  A triangulation of the sphere is an ASC with facets all 3-sets such that for every pair of vertices in the triangulation, the pair appears in precisely 0 or 2 facets.

uniformity (array)  An $n \times k$ array $A$ is uniform if, for each column $c_i$ over $g_i$ symbols, every symbol appears exactly $n/g_i$ times in $c_i$. If, for each column $c_i$ over $g_i$ symbols, every symbol appears either $\lfloor n/g_i \rfloor$ or $\lceil n/g_i \rceil$ times, the array is almost-uniform.

uniformity (hypergraph)  A hypergraph is uniform if every edge has the same size.

uniformity (vector / partition)  Let $S$ be a set. A vector $v \in S^n$ is uniform if every symbol of $S$ appears exactly $n/|S|$ times in $v$, and almost-uniform if every symbol appears either $\lfloor n/|S| \rfloor$ or $\lceil n/|S| \rceil$ times. A partition of $\{0, \ldots, n - 1\}$ into $s$ classes is uniform if each class has size $n/s$, and almost-uniform if each class has size either $\lfloor n/s \rfloor$ or $\lceil n/s \rceil$.

variable strength covering array  Let $\Delta$ be an abstract simplicial complex (ASC) over $\{0, \ldots, k - 1\}$ with set of facet set $\Lambda$ and let $t = \text{rank}(\Delta)$. A VCA$_\Delta(N; \Lambda, g)$ variable strength covering array, where $\lambda = (\lambda_1, \ldots, \lambda_t)$, is an $N \times k$ array over $\{0, \ldots, g - 1\}$ with columns $0, \ldots, k - 1$ such that if $B = \{b_0, \ldots, b_{s-1}\} \in \Lambda$, then the $N \times s$ subarray over the columns of $B$ contains as rows all of the $g^s$ tuples at least $\lambda_s$ times. If $\lambda_i = 1$ for all $1 \leq i \leq t$, we write VCA($N; \Lambda, g$).
**variable strength orthogonal array** Let $\Delta$ be an ASC over $\{0, \ldots, k-1\}$ with facet set $\Lambda$. A VOA($N; \Lambda, g$) variable strength orthogonal array is an $N \times k$ array over $S = \{0, \ldots, g-1\}$ with columns $0, \ldots, k-1$ such that if $B = \{b_0, \ldots, b_{s-1}\} \in \Delta$, then every $g^s$ tuple over $S$ appears in the $N \times s$ subarray over the columns of $B$ exactly $r = N/g^s$ times, where $r$ must be an integer.

**walk** For a hypergraph $H$, a walk of length $k$ from $u_0$ to $u_k$ $P = u_0e_1u_1e_2\ldots u_{k-1}e_ku_k$ has the property that $u_0 \neq u_k$, and for all $j$, $e_j \in E(H)$ and $u_{j-1}, u_j \in e_j$. 
Acronyms

ASC abstract simplicial complex
BIBD balanced incomplete block design
CA covering array
CCA consecutive covering array
cCCA cyclic consecutive covering array
COD covering ordered design
MCA mixed-level covering array
OA orthogonal array
RCOD resolvable covering ordered design
SQS Steiner quadruple system
STS Steiner triple system
VCA variable strength covering array
## Index

- $E_i(H)$, 15
- $H^\bullet$, 21
- $H^{k,t}$, 44
- $H^{c,t}$, 44
- $K_n$, 14
- $K_n^t$, 18
- $S(n, k)$, 68
- $T(v, n)$, 40
- $V(\Delta)$, 20
- $[H]_n$, 15
- $\text{Aut}(H)$, 17
- $\text{CAN}_\lambda(t, k, g)$, 27
- $\text{CA}_\lambda(N; t, k, g)$, 27
- $\text{CCA}(N; t, k, g)$, 44
- $\text{CCAN}(t, k, g)$, 44
- $\text{PQIH}(n, g)$, see qualitative independence hypergraph, partition
- $\text{COD}(N; t, k, g)$, 39
- $\text{CODN}(t, k, g)$, 39
- $\text{DCA}(N; 2, k, g)$, 38
- $\Delta$, 20
- $\Delta(H)$, 20
- $\Delta_i$, 20
- $\text{LS}(n)$, 35
- $\Lambda$, 20
- $\text{MCAN}_\lambda(t, (g_0, \ldots, g_{k-1}))$, 28
- $\text{MCA}_\lambda(N; t, (g_0, \ldots, g_{k-1}))$, 28
- $\text{MOLS}(n)$, 35
- $\text{OA}_\lambda(t, k, g)$, 26
- $\text{OOA}_\lambda(t, k, l, g)$, 42
- $\text{VQIH}(n, g)$, see qualitative independence hypergraph, vector
- $\text{RCOD}(N; t, k, g)$, 40
- $\text{RCODN}(t, k, g)$, 40
- $\text{UVQIH}(n, g)$, 77
- $\text{VCAN}_\lambda(\Lambda, g)$, 29
- $\text{VCA}_\lambda(N; \Lambda, g)$, 29
- $\text{VOA}(N; \Lambda, g)$, 29
- $\alpha(H)$, 96
- $\alpha_t(H)$, 19
- $\text{cCCA}(N; t, k, g)$, 45
- $\text{cCCAN}(t, k, g)$, 45
- $\chi$ (graph), 14
- $\chi(H)$, 23
- $\text{diam}(H)$, 17
- $\Delta_\lambda$, 23
- $\Delta_\lambda(G)$, 23
- $\lambda$-covered, 26
- $\lambda$-uncovered, 26
- $\omega(G)$, 19
- $\omega_t(H)$, 18
- $\text{rank}(H)$, 15
- $\rightarrow^W$, 79
- $\text{skel}$, 21
- $\text{skel}_n(\Delta)$, 21
- $\mathcal{H}$, 17
- $d_H(v)$, 16
- $d'_H(v)$, 16

228
\textit{k-tree}, \textit{46}

- hypergraph, \textit{47}

\textit{n-section}, \textit{15}

\textit{p}_H(u, v), \textit{17}

\textit{t-clique} of size \textit{n}, \textit{18}

\textit{t-degree}

- hypergraph, \textit{16}

2-tuple balance property, \textit{see} sequence, 2-tuple balance property

abstract simplicial complex, \textit{20}

- face, \textit{20}

- facet, \textit{20}

- facet hypergraph, \textit{20}

- induced by a hypergraph, \textit{20}

- simplex, \textit{20}

- skeleton, \textit{21}

- sub-ASC, \textit{20}

acyclic

- hypergraph, \textit{17}

ASC, \textit{see} abstract simplicial complex

- colouring, \textit{23}

automorphism

- hypergraph, \textit{17}

- set system, \textit{52}

automorphism group

- set system, \textit{52}

balance-covered, \textit{26}

balance-uncovered, \textit{26}

balanced incomplete block design, \textit{53}

- properties, \textit{53}

BIBD, \textit{see} balanced incomplete block design

bipartite, \textit{14}

Bush construction, \textit{36}

CA, \textit{see} covering array

chromatic number

- ASC, \textit{23}

- graph, \textit{14}

- hypergraph, strong, \textit{23}

- hypergraph, weak, \textit{23}

clique

- \textit{t-clique} of size \textit{n}, \textit{18}

- graph, \textit{19}

- hypergraph, \textit{18}

co-clique, \textit{19}

colouring

- ASC, \textit{23}

- graph, \textit{14}

- hypergraph, strong, \textit{23}

- hypergraph, weak, \textit{23}

combinatorial design, \textit{see} design

connected components, \textit{17}

connectedness

- hypergraph, \textit{17}

consecutive covering array, \textit{44}

core

- hypergraph, \textit{21}

coverage criteria, \textit{55}

covered, \textit{26}

- balance, \textit{26}

covering array, \textit{27}

- consecutive, \textit{44}

- cyclic consecutive, \textit{45}

factor, \textit{27}

forbidden edges, \textit{43}

index, \textit{27}

level, \textit{27}

mixed-level, \textit{28}

on graph, \textit{41}
optimal, 27
size, 27
strength, 27
uniform, 89
variable strength, 29
covering ordered design, 39
cut-clique, 93
cut-vertex, 90
cycle
hypergraph, 17
cyclic consecutive covering array, 45
DCA, see difference covering array
degree
hypergraph, 16
density
factor, 139
interaction, 139
set, 139
derived design, 55
design, 52
balance property, 53
index, 52
strength, 52
diameter, 17
difference covering array, 38
difference set, 109
endomorphism
hypergraph, 17
set system, 52
epimorphism
hypergraph, 16
set system, 52
extension design, 55
face, 20
facet, 20
factor
covering array, 27
fixed, 31
free, 31
mixed-level covering array, 28
orthogonal array, 26
fixed, 31
free, 31
Gaifman graph, 15
general relational system, 24
homomorphism, 25
pattern, 24
type, 24
graph, 14
bipartite, 14
chromatic number, 14
colouring, 14
complete, 14
Gaifman, 15
homomorphism, 14
isomorphism, 14
primal, 15
subgraph, 14
great deluge, 119
GRS, see general relational system
hill-climbing, 119
hole, 121
homomorphic equivalence
hypergraphs, 16
homomorphism
general relational system, 25
graph, 14
hypergraph, 16
set system, 52
hypergraph
  connected components, 17
hypergraph, 14
  n-section, 15
t-degree, 16
acylic, 17
chromatic number, strong, 23
chromatic number, weak, 23
colouring, strong, 23
colouring, weak, 23
connected, 17
cycle, 17
degree, 16
diameter, 17
facet hypergraph of ASC, 20
Gaifman graph, 15
homomorphism, 16
induced ASC, 20
loop, 14
path, 17
primal graph, 15
rank, 15
regular, 16
subhypergraph, 15
uniform, 15
walk, 17
hypertree, 17
input space partitioning, 56
interaction, 30
  extension, 31
  index function, 31
  subinteraction, 31
interaction testing, 27
irreducible polynomial, 103
isolated vertex, 72
isomorphism
  graph, 14
  hypergraph, 16
  set system, 52
latin square, 35
  orthogonal, 35
level
  covering array, 27
  mixed-level covering array, 28
  orthogonal array, 26
LFSR, see linear feedback shift register
linear feedback shift register, 103
Local Lemma, 9
  asymmetric, 172
  general, 172
  Shearer improvement, 171
  symmetric, 171
  weighted, 172
loop, 14
MCA, see mixed-level covering array
minimal polynomial, 103
mixed-level covering array, 28
  factor, 28
  index, 28
  level, 28
  optimal, 28
qualitative independence definition, 79
size, 28
strength, 28
MOLS, see mutually orthogonal latin squares
monic polynomial, 103
monomorphism
hypergraph, 16
set system, 52
mutually orthogonal latin squares, 35
OA, see orthogonal array
optimality, 28
covering array, 27
ordered orthogonal array, 42
orthogonal array
Bush construction, 36
factor, 26
finite field construction, 35
index, 26
level, 26
MOLS construction, 35
ordered, 42
strength, 26
variable strength, 29
Zero-Sum construction, 36
path
hypergraph, 17
period, see sequence, period
polynomial
irreducible, 103
minimal, 103
monic, 103
primitive, 103
primal graph, 15
primitive element, 103
primitive polynomial, 103
projective plane, 54
qualitative independence
mixed-level, 79
partitions, 66
vectors, 66
qualitative independence hypergraph
mixed-level, 80
partition, 68
uniform, 77
vector, 68
rank
hypergraph, 15
regular, see hypergraph, regular
set system, 51
resolution, 40
Roux-type construction
t = 2, 37
t = 3, 38
section, 15
sequence
2-tuple balance property, 105
period, 104
reversal, 110
set system, 51
automorphism, 52
automorphism group, 52
endomorphism, 52
epimorphism, 52
homomorphism, 52
isomorphism, 52
monomorphism, 52
regular, 51
replication number, 51
simple, 51
simple
set system, 51
simplex, 20
size
mixed-level covering array, 28
skeleton, 21
sphere
triangulation hypergraph, 47
Steiner quadruple system, 54
existence, 55
replication number, 55
Steiner system, 54
Steiner triple system, 54
existence, 54
replication number, 54
Stirling numbers
second kind, 68
strength
covering array, 27
design, 52
mixed-level covering array, 28
orthogonal array, 26
strong colouring, 23
subgraph, 14
subhypergraph, 15
subinteraction, 31
subsumption, 56
trace, 104
transitivity
vertex, 75
tree, 17
triangulation hypergraph
sphere, 47
Turán numbers, 40
uncovered, 26
balance, 26
uniform
higher strength, 92
hypergraph, 15
partition, 76
vector, 76
uniform qualitative independence hypergraph, 77
variable strength covering array, 29
qualitative independence definition, 66
variable strength orthogonal array, 29
VCA, see variable strength covering array vertex transitive, 75
VOA, see variable strength orthogonal array
walk
hypergraph, 17
weak colouring, 23
Zero-Sum construction, 36