Combinatorial Constructions of Ordered Orthogonal Arrays & Ordered Covering Arrays

by

Tamar Krikorian

Bachelor of Mathematics, University of Waterloo, 2006

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Program of

Applied Mathematics

Toronto, Ontario, Canada, 2011

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Combinatorial Constructions of Ordered Orthogonal Arrays & Ordered Covering Arrays

Master of Science 2011

Tamar Krikorian

Applied Mathematics

Ryerson University

In this thesis, we consider combinatorial objects called *ordered orthogonal arrays*, which are related to *orthogonal arrays* and *Latin squares*. We also introduce a new combinatorial object called *ordered covering arrays*, which generalize *covering arrays*. We adapt existing combinatorial methods to the construction of these objects, as well as developing new ones. We discuss the applications of ordered orthogonal arrays and ordered covering arrays to quasi-Monte Carlo integration through the construction of point sets called (t, m, s)-nets and a new object we call (t, m, s)-covering nets.

Acknowledgements

First are foremost, I would like to like to express my sincere gratitude to my supervisor, Dr. Peter Danziger, for his guidance, critical feedback, great ideas and continuous support throughout my thesis. I would also like to thank Dr. Brett Stevens and Dr. Katrin Rohlf, for their assistance with certain areas in my thesis. For their time, suggestions and positive feedback, I would like to thank the examining committee; Dr. Peter Danziger, Dr. Dejan Delic, Dr. Anthony Bonato and Dr. Jean-Paul Pascal. To my parents, Gretta Ayoub and Harout Krikorian, I cannot thank you enough. You have always stood by me in everything that I do, always providing me with your encouragement and support. This thesis would never have existed without you, and I dedicate my thesis to you. To my love, Baker Saleh, thank you for your continuous love, support and patience, and for always believing in me. And finally, I would like to thank Ryerson University, the Department of Mathematics, and the Yeates School of Graduate Studies, for giving me the opportunity to write this thesis, and to work with my supervisor, along with providing me with financial support.

Contents

Chapter 1

Introduction

Latin squares, orthogonal arrays, ordered orthogonal arrays and *covering arrays* are all related combinatorial objects that have been studied extensively [17]. They each have different applications, many in the fields of statistics as test suites and in certain areas within mathematics and computer science, such as coding theory and cryptography.

The main contributions of this thesis are the development of new constructions for ordered orthogonal arrays and the introduction of a new combinatorial object called *ordered covering arrays*, along with related constructions. We also discuss the application of both these objects to quasi-Monte Carlo methods of numerical integration through the construction of point sets called *(t, m, s)-nets* and a new object we call *(t, m, s)-covering nets*.

In Section 1.1, we begin by introducing the main combinatorial objects that will be discussed in this thesis, formal definitions will follow in Chapter 2. We also briefly discuss some applications. In Section 1.2, we give an overview of the thesis.

1.1 Latin Squares, Orthogonal Arrays, Covering Arrays and Ordered Orthogonal Arrays

A *Latin square* of order n is an $n \times n$ array filled with n different symbols, in such a way that each symbol occurs exactly once in each row and exactly once in each column. Latin squares were first introduced by Euler [20] in 1782. These objects are well studied and have applications to experimental designs, scheduling, graph theory, error correcting codes and cryptography [25].

Orthogonal arrays are combinatorial structures which are closely related to Latin squares and are essential in statistics in designing experiments and are also very useful in computer science and cryptography [23]. For positive integers λ, t, k and v, an *orthogonal array*, denoted $OA_\lambda(t, k, v)$ is a $\lambda v^t \times k$ array with entries from an alphabet of size v, such that every t-tuple appears exactly λ times within every set of t columns. Orthogonal arrays do not exist for all parameters λ, t, k and v, which brings us to our next object.

A *covering array*, which has applications in areas such as software [12, 22], hardware [22], circuits [36], new materials [10] and genetics [37] testing, extends the notion of an orthogonal array. A covering array, denoted $CA_\lambda(t, k, v)$, is similar to an orthogonal array, the only difference is that in a covering array each t-tuple occurs *at least* λ times, whereas in an orthogonal array, each t-tuple occurs *exactly* λ times. By relaxing the condition on the number of occurrences of each t-tuple from exactly λ times to at least λ times, allows covering arrays to exist for every integer set of parameter values. The problem now becomes how close we can get to an optimal covering array.

Ordered orthogonal arrays which were introduced independently by Lawrence [24] and Mullen and Schmid [30], generalize orthogonal arrays by imposing balance conditions on certain sets of columns. Point sets called *(t, m, s)-nets*, which were first introduced by Niederreiter [31] in 1987, are equivalent to a parametric subclass of ordered orthogonal arrays as shown in [24, 30]. A (t, m, s)-net is a set of points which satisfies strong uniformity properties with regards to their distribution in the s-dimensional unit cube and are useful in approximating the value of multi-dimensional integrals through quasi-Monte Carlo integration [32].

1.2 Overview of Thesis

In Chapter 2, we begin by providing the necessary background on *Latin squares, orthogonal arrays* and *covering arrays*. We describe existing recursive and Roux constructions for covering arrays. We then discuss another object called *ordered orthogonal arrays* (OOA), and describe their equivalence to point sets called *(t, m, s)-nets*. We also briefly discuss their application to quasi-Monte Carlo integration. In Chapter 3, we introduce a new homomorphism mapping ordered orthogonal arrays to orthogonal arrays, and also develop new theorems and constructions for OOAs. In Chapter 4, we introduce a new combinatorial object called an *ordered covering array* (OCA), and adapt existing recursive and Roux constructions for covering arrays to construct OCAs, as well as developing new theorems and constructions for this new object. We also introduce another new object called a *(t, m, s)-covering net*, which is similar to a (t, m, s)-net, and we discuss their equivalence to OCAs. In Chapter 5, we describe a method of constructing distinct (t, m, s)-nets from a single OOA, and we also discuss other useful applications of OOAs and OCAs in creating point sets used to numerically approximate the values of certain types of multi-dimensional integrals. Finally, in Chapter 6, we conclude by summarizing the main results presented in this thesis, and highlighting some possibilities for future work in this area.

Chapter 2

Background on Related Combinatorial Objects

In this chapter, we define Latin squares, orthogonal arrays, covering arrays, ordered orthogonal arrays and (t, m, s)-nets, and provide some background on these objects, as well as discussing some basic constructions.

2.1 Latin Squares and Orthogonal Arrays

Definition 2.1.1. *A **Latin square** of side n is an $n \times n$ array in which each cell contains a single element from an n-set S, such that each element of S occurs exactly once in each row and each column. The **order** of the Latin square is the value n.*

Definition 2.1.2. *Two Latin squares of order n, $L_1 = [a_{ij}]$ and $L_2 = [b_{ij}]$, on the same symbol set S are orthogonal if every element in $S \times S$ occurs exactly once among the n^2 pairs (a_{ij}, b_{ij}) , where $1 \leq i, j \leq n$. We say that L_1 and L_2 are two **mutually orthogonal latin squares** (MOLS) of order n.*

Example 2.1.3. *The following arrays are 2 MOLS of order 3:*

1	2	3
2	3	1
3	1	2

1	2	3
3	1	2
2	3	1

By superimposing one Latin square on the other, all 9 possible ordered pairs occur exactly once.

A set of s Latin squares of order n, $L_1, ..., L_s$, are mutually orthogonal Latin squares if L_i and L_j are orthogonal for all $1 \leq i < j \leq s$. There exists at most $n - 1$ mutually orthogonal Latin squares of order n. At present, no set of $n - 1$ MOLS of order n is known for integers other than prime powers [5]. In 1782, Euler conjectured that no pair of MOLS exist for $n \equiv 2 \,(\mathrm{mod} 4)$ [20]. In 1900, Tarry proved that

no pair of MOLS of order 6 exist [39]. In 1960, Bose, Shrikhande and Parker proved Euler's conjecture to be false for all $n \geq 10$ [6].

A set of mutually orthogonal Latin squares is equivalent to a parametric subclass of another combinatorial structure called an *orthogonal array*.

Definition 2.1.4. *For positive integers λ, t, k and v, where $2 \leq t \leq k$, an **orthogonal array** $OA_\lambda(N; t, k, v)$ is an $N \times k$ array A with entries from an alphabet of v symbols, where $N = \lambda v^t$, such that in every $N \times t$ subarray, every t-tuple occurs exactly λ times as a row.*

In an $OA_\lambda(N; t, k, v)$, t denotes the *strength* of the array, k is the number of columns and v is the alphabet size. When λ is not specified, its value is one. Note that the parameter N is not required and can be dropped, hence, it is followed by a semi-colon rather than a comma.

Example 2.1.5. *An example of an orthogonal array with strength 2, 4 columns and alphabet size 3; $OA(9; 2, 4, 3)$:*

$OA(9; 2, 4, 3)$

0	0	0	0
0	1	1	1
0	2	2	2
1	0	1	2
1	1	2	0
1	2	0	1
2	0	2	1
2	1	0	2
2	2	1	0

Every pair (0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2) is covered exactly once within every 2 columns of the array.

The existence of k MOLS of order n is equivalent to an orthogonal array of strength two, $k + 2$ columns and alphabet size n, namely $OA(2, k + 2, n)$. In the construction, each row of the OA corresponds to a particular cell in each of the Latin squares [1].

Orthogonal arrays with equal strength and number of columns always exist, as stated in the following theorem.

Theorem 2.1.6. *For all positive integers λ, t and v, there exists an $OA_\lambda(t, t, v)$.*

Proof. List each of the v^t possible t-tuples λ times. (There are v^t permutations of a row of length t, where each of the t positions has v choices.) Therefore each t-tuple appears λ times in this $v^t \times t$ array, which is thus an $OA_\lambda(t, t, v)$. □

In the following theorem, a multiplicative construction is used to create strength two orthogonal arrays with a larger alphabet size.

Theorem 2.1.7. *(MacNeish, [17]) If there exists an $OA(2, k_1, v_1)$ and an $OA(2, k_2, v_2)$, then there exists an $OA(2, min(k_1, k_2), v_1 v_2)$.*

Proof. Let $A = (a_{ij})$ be an $OA(N_1;\, 2, k_1, v_1)$ and $B = (b_{ij})$ be an $OA(N_2;\, 2, k_2, v_2)$. Let $k = min(k_1, k_2)$. For $1 \leq i \leq N_1 N_2$ and $1 \leq j \leq k$, form an $N_1 N_2 \times k$ array C by setting:

$$C_{i,j} = \begin{cases} (a_{\lceil \frac{i}{N_2} \rceil, j}, b_{i \bmod N_2, j}) & \text{for } i \ (\mathrm{mod} N_2) \neq 0, \\ (a_{\lceil \frac{i}{N_2} \rceil, j}, b_{N_2, j}) & \text{for } i \ (\mathrm{mod} N_2) = 0. \end{cases}$$

The result is an $OA(2, k, v_1 v_2)$. For an array A, with entry set V_1, where $v_1 = |V_1|$ and array B with entry set V_2, where $v_2 = |V_2|$, we construct an $OA(N_1 N_2;\, 2, min(k_1, k_2), v_1 v_2)$ with entry set $V_1 \times V_2$ by taking each entry from $\min(k_1, k_2)$ columns of array A, and pairing them with the entries from the corresponding $\min(k_1, k_2)$ columns of array B. The resulting array is an $OA(N_1 N_2;\, 2, min(k_1, k_2), v_1 v_2)$. This construction is shown in the figure below:

(a_{11}, b_{11})	(a_{12}, b_{12})	$\cdots$	(a_{1k}, b_{1k})
(a_{11}, b_{21})	(a_{12}, b_{22})	$\cdots$	(a_{1k}, b_{2k})
$\vdots$			
$(a_{11}, b_{N_2 1})$	$(a_{12}, b_{N_2 2})$	$\cdots$	$(a_{1k}, b_{N_2 k})$
(a_{21}, b_{11})	(a_{22}, b_{12})	$\cdots$	(a_{2k}, b_{1k})
(a_{21}, b_{21})	(a_{22}, b_{22})	$\cdots$	(a_{2k}, b_{2k})
$\vdots$			
$(a_{21}, b_{N_2 1})$	$(a_{22}, b_{N_2 2})$	$\cdots$	$(a_{2k}, b_{N_2 k})$
$\vdots$			
$(a_{N_1 1}, b_{11})$	$(a_{N_1 2}, b_{12})$	$\cdots$	$(a_{N_1 k}, b_{1k})$
$(a_{N_1 1}, b_{21})$	$(a_{N_1 2}, b_{22})$	$\cdots$	$(a_{N_1 k}, b_{2k})$
$\vdots$			
$(a_{N_1 1}, b_{N_2 1})$	$(a_{N_1 2}, b_{N_2 2})$	$\cdots$	$(a_{N_1 k}, b_{N_2 k})$

□

Example 2.1.8. *We will construct an $OA(36;\, 2, 2, 6)$ from $OA(4;\, 2, 3, 2)$ and $OA(9;\, 2, 2, 3)$.*

Let A be an $OA(4;\, 2, 3, 2)$ and B be an $OA(9;\, 2, 2, 3)$:

$OA(4;\, 2, 3, 2)$

0	0	1
0	1	0
1	0	0
1	1	1

$OA(9;\, 2, 2, 3)$

0	0
0	1
0	2
1	0
1	1
1	2
2	0
2	1
2	2

We take each entry in the first min(2,3) = 2 columns of each row of A, and pair them with the corresponding entries of each row of B. We get the following 72 pairs, where the first value from each pair has alphabet size 2 and the second value from each pair has alphabet size 3:

$(0,0)$	$(0,0)$
$(0,0)$	$(0,1)$
$(0,0)$	$(0,2)$
$(0,1)$	$(0,0)$
$(0,1)$	$(0,1)$
$(0,1)$	$(0,2)$
$(0,2)$	$(0,0)$
$(0,2)$	$(0,1)$
$(0,2)$	$(0,2)$
$(0,0)$	$(1,0)$
$(0,0)$	$(1,1)$
$(0,0)$	$(1,2)$
$(0,1)$	$(1,0)$
$(0,1)$	$(1,1)$
$(0,1)$	$(1,2)$
$(0,2)$	$(1,0)$
$(0,2)$	$(1,1)$
$(0,2)$	$(1,2)$
$(1,0)$	$(0,0)$
$(1,0)$	$(0,1)$
$(1,0)$	$(0,2)$
$(1,1)$	$(0,0)$
$(1,1)$	$(0,1)$
$(1,1)$	$(0,2)$
$(1,2)$	$(0,0)$
$(1,2)$	$(0,1)$
$(1,2)$	$(0,2)$
$(1,0)$	$(1,0)$
$(1,0)$	$(1,1)$
$(1,0)$	$(1,2)$
$(1,1)$	$(1,0)$
$(1,1)$	$(1,1)$
$(1,1)$	$(1,2)$
$(1,2)$	$(1,0)$
$(1,2)$	$(1,1)$
$(1,2)$	$(1,2)$

We can always translate the above pairs from alphabet size 2×3 to alphabet size 6, using the following translation:

$$(0,0) \to 0$$
$$(0,1) \to 1$$
$$(0,2) \to 2$$
$$(1,0) \to 3$$
$$(1,1) \to 4$$
$$(1,2) \to 5$$

Substituting our new alphabet into the pairs of the previous array gives us an $OA(36;\, 2, 2, 6)$:

$OA(36;\ 2, 2, 6)$

0	0
0	1
0	2
1	0
1	1
1	2
2	0
2	1
2	2
0	3
0	4
0	5
1	3
1	4
1	5
2	3
2	4
2	5
3	0
3	1
3	2
4	0
4	1
4	2
5	0
5	1
5	2
3	3
3	4
3	5
4	3
4	4
4	5
5	3
5	4
5	5

2.2 Covering Arrays

2.2.1 Introduction to Covering Arrays

A combinatorial object called a *covering array* extends the notion of an orthogonal array, and has applications in areas such as software [12, 22], hardware [22], circuits [36], new materials [10] and genetics [37] testing. A covering array is very similar to an orthogonal array, the only difference is that in a covering array each t-tuple occurs *at least* λ times, whereas in an orthogonal array, each t-tuple occurs *exactly* λ times. For more information on covering arrays, see the surveys [16, 21].

It has been estimated that at least 50% of the cost of developing a new piece of software is attributed

to testing, and the testing costs for hardware systems are even higher [22]. Interaction test suites is the main application of covering arrays [12, 16, 22]. In software testing, software systems are built using components. Therefore, unexpected interactions among these components can lead to system faults. If we were to apply exhaustive testing, then for a system with 5 components, each with 20 potential configurations, we have a total of $20^5 = 3{,}200{,}000$ potential interactions, and we therefore would require a total of 3,200,000 tests. So exhaustive testing becomes infeasible for larger systems. Instead, we can use interaction testing to reduce the number of tests. Covering arrays can be used to generate interaction test suites. By using a covering array as our test suite, we guarantee that we have tested all t-way interactions at least once (or if $\lambda > 1$, then at least λ times). Each component in the system is called a *factor*, and the possible configurations of the component are called *values* or *levels*. In this context, we are concerned with minimizing the number of rows of a covering array which translates to the number of test cases in our test suite.

Definition 2.2.1. *For positive integers λ, t, k and v, a **covering array** $CA_\lambda(N; t, k, v)$ is an $N \times k$ array, A, with entries from an alphabet with v symbols, such that in every $N \times t$ subarray, consisting of t columns of A, every t-tuple occurs at least λ times as a row. The parameter N is the size of the array; t is the strength of coverage of interactions; k is the number of factors; v is the number of values for each factor and λ is the index. The size of a covering array is the **covering array number** $CAN_\lambda(t, k, v)$ and it is the minimum integer N for which a $CA_\lambda(N; t, k, v)$ exists. The covering array is **optimal** if it contains the minimum possible number of rows.*

Remark 2.2.2. *Although the covering array number is defined as the minimum integer N for which a $CA(N; t, k, v)$ exists, we do not always know the value of this parameter N which gives us an optimal covering array for parameters λ, t, k and v.*

Note that the parameter N, from a $CA_\lambda(N; t, k, v)$, is not required and can be dropped. Therefore it is followed by a semi-colon, rather than a comma.

Example 2.2.3. *A covering array with pairwise coverage, 5 factors and alphabet size 2 with values 0 and 1; $CA(6; 2, 5, 2)$:*

$CA(6; 2, 5, 2)$

1	1	1	1	1
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Within every 2 columns of the array, all 4 pairs (0,0), (0,1), (1,0) and (1,1) are covered at least once. For example, the pair (0,0) is covered 3 times within every 2 columns, while the other 3 pairs (0,1), (1,0) and (1,1) are covered exactly once. The minimum number of rows required to cover all 4 pairs within every 2 columns of the 5 factors of the array is size $N = 6$.

Mixed covering arrays are a generalization of covering arrays that allows for different alphabet sizes in different columns of the array. This is a more realistic approach to interaction testing, since it allows different factors to have a different number of values. These objects are discussed in [29].

Definition 2.2.4. *For positive integers λ, t, k and $v_1, v_2, ..., v_k$, a **mixed covering array** $MCA_\lambda(N; t, k, (v_1, v_2, ..., v_k))$ is an $N \times k$ array in which the entries of the i^{th} column arise from an alphabet of size v_i, and the rows of each $N \times t$ sub-array cover all t-tuples of the t columns at least λ times.*

Example 2.2.5. *A mixed covering array with strength 2, 3 factors and alphabet sizes 2, 3 and 2 for each factor, respectively; $MCA(2, 3, (2, 3, 2))$:*

$MCA(2, 3, (2, 3, 2))$

0	0	0
0	1	1
0	2	0
1	0	1
1	1	0
1	2	1

All 6 pairs formed from two factors with values of alphabet sizes 2 and 3 are covered between factors 1 and 2, and 2 and 3. All 4 pairs formed from two factors, each with values of alphabet size 2, are covered at least once between factors 1 and 3.

The *excess graph* which is an important concept for covering designs [17], can also be applied to covering arrays [18]. The graph shows those values that appear in the array more than λ times.

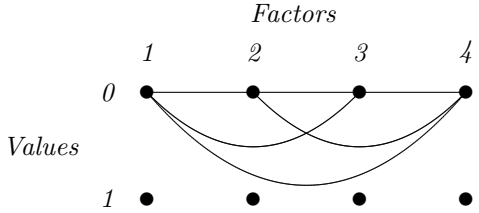
Definition 2.2.6. *The **excess graph** of a $CA_\lambda(2, k, v)$, is a graph on kv vertices, where each value of each of the k factors is represented by a vertex, such that two vertices are connected by an edge if the pair it represents appears more than λ times in the array.*

Remark 2.2.7. *In the above definition, we defined the excess graph only for strength two covering arrays. This can be generalized to any strength t, but for any $t > 2$, we will have a hypergraph.*

Example 2.2.8. *Below is a $CA(2, 4, 2)$ and its excess graph:*

$CA(2, 4, 2)$

1	1	1	1
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1



In the $CA(2, 4, 2)$, the pair (0,0) appears twice within every 2 columns of the array, while all other pairs appear exactly once within every 2 columns. Therefore, the corresponding excess graph has an edge connecting every two factors with value 0, since these pairs appear twice in the array.

The classical covering array problem of minimizing the size of the array as well as other generalizations of the problem (such as extending/reducing covering arrays and forbidden configurations) have been studied extensively, and many constructions and algorithms have been developed in order to construct these arrays efficiently [8, 9, 13]. We review some standard CA constructions which we will adapt to construct OCAs.

2.2.2 Recursive and Roux Constructions

For covering arrays of strength 2, a blocksize recursive construction can be used to create covering arrays with a larger number of factors [16]:

Theorem 2.2.9. *(Colbourn, [16]) If a $CA(N; 2, k, v)$ and a $CA(M; 2, l, v)$ both exist, then a $CA(N + M; 2, kl, v)$ exists.*

Proof. Let $A = (a_{ij})$ be a $CA(N; 2, k, v)$ and let $B = (b_{ij})$ be a $CA(M; 2, l, v)$. For $1 \leq f \leq l$ and $1 \leq g \leq k$, form an $(N + M) \times kl$ array $C = (c_{i,j}) = A \otimes B$ by setting:

$$c_{i,(f-1)k+g} = \begin{cases} a_{i,g} & \text{for } 1 \leq i \leq N, \\ b_{i-N,f} & \text{for } N+1 \leq i \leq N+M. \end{cases}$$

The result is a $CA(N + M; 2, kl, v)$. The construction of $A \otimes B$ is shown in the figure below where there are k copies of B being added to l copies of A. Any two columns of the $CA(N + M; 2, kl, v)$ with different indices mod k, will have all pairs covered from A. For any two columns with the same index mod k, all pairs formed from two columns of A will be of the form xx for $x \in V$, where V represents the alphabet set of size v. The remaining pairs are covered from the columns of B which were added below.

a_{11}	a_{12}	$\cdots$	a_{1k}	a_{11}	a_{12}	$\cdots$	a_{1k}	$\cdots$	a_{11}	a_{12}	$\cdots$	a_{1k}
a_{21}	a_{22}	$\cdots$	a_{2k}	a_{21}	a_{22}	$\cdots$	a_{2k}	$\cdots$	a_{21}	a_{22}	$\cdots$	a_{2k}
$\vdots$				$\vdots$				$\cdots$	$\vdots$			
a_{N1}	a_{N2}	$\cdots$	a_{Nk}	a_{N1}	a_{N2}	$\cdots$	a_{Nk}	$\cdots$	a_{N1}	a_{N2}	$\cdots$	a_{Nk}
b_{11}	b_{11}	$\cdots$	b_{11}	b_{12}	b_{12}	$\cdots$	b_{12}	$\cdots$	b_{1l}	b_{1l}	$\cdots$	b_{1l}
b_{21}	b_{21}	$\cdots$	b_{21}	b_{22}	b_{22}	$\cdots$	b_{22}	$\cdots$	b_{2l}	b_{2l}	$\cdots$	b_{2l}
$\vdots$				$\vdots$				$\cdots$	$\vdots$			
b_{M1}	b_{M1}	$\cdots$	b_{M1}	b_{M2}	b_{M2}	$\cdots$	b_{M2}	$\cdots$	b_{Ml}	b_{Ml}	$\cdots$	b_{Ml}

□

Remark 2.2.10. *If an array has r constant rows, then the resulting array will have r duplicated rows. So before the construction, the constant rows can be removed in order for the resulting array to be optimal.*

Example 2.2.11. *We will construct a $CA(11; 2, 20, 2)$ from a $CA(5; 2, 4, 2)$ and a $CA(6; 2, 5, 2)$.*

$CA(2,4,2)$

1	1	1	1
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

$CA(2,5,2)$

1	1	1	1	1
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

We place 5 copies of a $CA(2,4,2)$ beside each other, and under the first copy of the $CA(2,4,2)$, we place 4 copies of column 1 of a $CA(2,5,2)$ beside each other, and under the second copy of the $CA(2,4,2)$, we place 4 copies of column 2 of the $CA(2,5,2)$, and we do this till the last (5th) copy of the $CA(2,4,2)$, under which we place 4 copies of column 5 of the $CA(2,5,2)$. The resulting array has 4 $\times$ 5 = 20 columns, and a total of 5 + 6 = 11 rows. The result is a $CA(11;\, 2, 20, 2)$ since each pair of columns whose indices are the same mod 4 will have all pairs covered by the $CA(11;\, 2, 20, 2)$ and those that are not indexed by the same value mod 4 will have all pairs covered by the $CA(5;\, 2, 4, 2)$. The construction is shown in the figure below:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

The above $CA(11;\, 2, 20, 2)$ is not optimal. By removing the constant row of ones from the $CA(2,5,2)$ before applying the construction, results in a $CA(10;\, 2, 20, 2)$, which is optimal:

$CA(10;\, 2, 20, 2)$

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

For covering arrays of strength 3, Roux-type constructions can be used to create covering arrays with a larger number of factors. In [38], a theorem from Roux's Ph.D. dissertation [35] is presented, we provide a sketch of the proof:

Theorem 2.2.12. *(Roux, [35, 38])* $CAN(3, 2k, 2) \leq CAN(3, k, 2) + CAN(2, k, 2)$.

Proof. To construct a $CA(3, 2k, 2)$, we begin by placing two $CA(N_1; 3, k, 2)$'s side by side. We now have an $N_1 \times 2k$ array. For any three columns whose indices are all different modulo k, all triples are covered. But for a selection of two columns that have the same value modulo k, and a third column with a different value, all triples will not be covered. Only triples where the two same columns have the same values are covered, which are triples of the form aaa, aab, bba and bbb, for an alphabet set $\{a, b\}$. To the bottom of the new array, we add a $CA(N_2; 2, k, 2)$ and its bit complement, side by side. Now, for any three columns, which contain two columns that have the same value modulo k, we will get the remaining triplets aba, abb, baa and bab, from the $CA(2, k, 2)$ and its bit complement. The resulting array is a $CA(N_1 + N_2; 3, 2k, 2)$. This construction is shown in the figure below:

$CA(3, k, 2)$	$CA(3, k, 2)$
$CA(2, k, 2)$	Bit Complement of $CA(2, k, 2)$

□

Example 2.2.13. *We will construct a* $CA(3, 6, 2)$ *from a* $CA(3, 3, 2)$. *We begin by placing two* $CA(3, 3, 2)$*'s side by side:*

0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	1	1	1

We construct a $CA(2, 3, 2)$ *and its bit complement:*

$CA(2, 3, 2)$

1	1	1
1	0	0
0	1	0
0	0	1

Complement

0	0	0
0	1	1
1	0	1
1	1	0

We add the $CA(2,3,2)$ and its bit compliment side by side, to the bottom of the two $CA(3,3,2)$'s that were placed side by side, in order to get a $CA(3,6,2)$:

0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	0	0	0
1	0	0	0	1	1
0	1	0	1	0	1
0	0	1	1	1	0

$\longrightarrow$

$CA(3,6,2)$

0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	1	1	1
1	1	1	0	0	0
1	0	0	0	1	1
0	1	0	1	0	1
0	0	1	1	1	0

Chateauneuf and Kreher [11] have generalized the above theorem for all alphabet size v:

Theorem 2.2.14. *(Chateauneuf and Kreher, [11])* $CAN(3,2k,v) \leq CAN(3,k,v)+(v-1)CAN(2,k,v)$.

Proof. As per Theorem 2.2.12, we begin by placing two $CA(N_1;\,3,k,v)$'s side by side. For any three columns whose indices are all different modulo k, all triples are covered. But for a selection of two columns that have the same value modulo k, and a third column with a different value, all triples will not be covered, only triples of the form aax will be covered for $a,x \in V$, where V represents the alphabet set of size v. Let C be a $CA(N_2;\,2,k,v)$ and let π be a cyclic permutation of the v symbols. Then for all $1 \leq i \leq v-1$, we append N_2 rows consisting of C and $\pi^i(C)$ placed side by side, below the $CA(3,k,v)$'s. Now, for any three columns that contain two columns with the same value modulo k, the remaining triplets will be covered from the columns in arrays C and $\pi^i(C)$. The construction is shown in the figure below:

$CA(3,k,v)$	$CA(3,k,v)$
$C = CA(2,k,v)$	$\pi^1(C)$
C	$\pi^2(C)$
$\vdots$	$\vdots$
C	$\pi^{v-1}(C)$

□

Example 2.2.15. *We will construct a $CA(3,6,3)$ from a $CA(3,3,3)$. We begin by placing two $CA(3,3,3)$'s side by side:*

0	0	0	0	0	0
0	0	1	0	0	1
0	0	2	0	0	2
0	1	0	0	1	0
0	1	1	0	1	1
0	1	2	0	1	2
0	2	0	0	2	0
0	2	1	0	2	1
0	2	2	0	2	2
1	0	0	1	0	0
1	0	1	1	0	1
1	0	2	1	0	2
1	1	0	1	1	0
1	1	1	1	1	1
1	1	2	1	1	2
1	2	0	1	2	0
1	2	1	1	2	1
1	2	2	1	2	2
2	0	0	2	0	0
2	0	1	2	0	1
2	0	2	2	0	2
2	1	0	2	1	0
2	1	1	2	1	1
2	1	2	2	1	2
2	2	0	2	2	0
2	2	1	2	2	1
2	2	2	2	2	2

We construct a $CA(2,3,3)$, call it C, and $v-1=3-1=2$ cyclic permutations of it:

C

0	0	0
0	1	1
0	2	2
1	0	1
1	1	2
1	2	0
2	0	2
2	1	0
2	2	1

$\pi^1(C) = (012)$

1	1	1
1	2	2
1	0	0
2	1	2
2	2	0
2	0	1
0	1	0
0	2	1
0	0	2

$\pi^2(C) = (021)$

2	2	2
2	0	0
2	1	1
0	2	0
0	0	1
0	1	2
1	2	1
1	0	2
1	1	0

To the bottom of the two $CA(3,3,3)$'s that were placed side by side, we add two copies of the original $CA(2,3,3)$, one under the other, and beside each of them, we place one of the permuted $CA(2,3,3)$'s. This results in a $CA(3,6,3)$:

0	0	0	0	0	0
0	0	1	0	0	1
0	0	2	0	0	2
0	1	0	0	1	0
0	1	1	0	1	1
0	1	2	0	1	2
0	2	0	0	2	0
0	2	1	0	2	1
0	2	2	0	2	2
1	0	0	1	0	0
1	0	1	1	0	1
1	0	2	1	0	2
1	1	0	1	1	0
1	1	1	1	1	1
1	1	2	1	1	2
1	2	0	1	2	0
1	2	1	1	2	1
1	2	2	1	2	2
2	0	0	2	0	0
2	0	1	2	0	1
2	0	2	2	0	2
2	1	0	2	1	0
2	1	1	2	1	1
2	1	2	2	1	2
2	2	0	2	2	0
2	2	1	2	2	1
2	2	2	2	2	2
0	0	0	1	1	1
0	1	1	1	2	2
0	2	2	1	0	0
1	0	1	2	1	2
1	1	2	2	2	0
1	2	0	2	0	1
2	0	2	0	1	0
2	1	0	0	2	1
2	2	1	0	0	2
0	0	0	2	2	2
0	1	1	2	0	0
0	2	2	2	1	1
1	0	1	0	2	0
1	1	2	0	0	1
1	2	0	0	1	2
2	0	2	1	2	1
2	1	0	1	0	2
2	2	1	1	1	0

$\longrightarrow$

$CA(3, 6, 3)$

0	0	0	0	0	0
0	0	1	0	0	1
0	0	2	0	0	2
0	1	0	0	1	0
0	1	1	0	1	1
0	1	2	0	1	2
0	2	0	0	2	0
0	2	1	0	2	1
0	2	2	0	2	2
1	0	0	1	0	0
1	0	1	1	0	1
1	0	2	1	0	2
1	1	0	1	1	0
1	1	1	1	1	1
1	1	2	1	1	2
1	2	0	1	2	0
1	2	1	1	2	1
1	2	2	1	2	2
2	0	0	2	0	0
2	0	1	2	0	1
2	0	2	2	0	2
2	1	0	2	1	0
2	1	1	2	1	1
2	1	2	2	1	2
2	2	0	2	2	0
2	2	1	2	2	1
2	2	2	2	2	2
0	0	0	1	1	1
0	1	1	1	2	2
0	2	2	1	0	0
1	0	1	2	1	2
1	1	2	2	2	0
1	2	0	2	0	1
2	0	2	0	1	0
2	1	0	0	2	1
2	2	1	0	0	2
0	0	0	2	2	2
0	1	1	2	0	0
0	2	2	2	1	1
1	0	1	0	2	0
1	1	2	0	0	1
1	2	0	0	1	2
2	0	2	1	2	1
2	1	0	1	0	2
2	2	1	1	1	0

Cohen, Colbourn and Ling [15] generalized this further to allow the number of factors to be multiplied by $l \geq 2$ rather than $l = 2$. This is called the *k-ary Roux construction*.

Hartman [21, 22] showed that a similar Roux type construction for $2k$ factors holds for covering arrays of strength 4:

Theorem 2.2.16. *(Hartman, [21, 22])*

$$CAN(4, 2k, v) \leq CAN(4, k, v) + (v - 1)CAN(3, k, v) + CAN(2, k, v^2).$$

2.2.3 Algorithms

Much effort has gone into refining algorithms to construct covering arrays. Some objectives of these algorithms include the size of the test suite, execution time to generate the test suite, consistency of the test suites generated and accommodation of seeds and constraints imposed by the tester [13]. Completeness of coverage is a primary goal. The objective of creating the smallest possible test suite which covers all t-way interactions is important since every additional test case adds to the cost of testing, but no single algorithm provides the most optimal test suite for every possible input. Good performance in the average case and worst-case guarantee on test suite size are also important. The efficiency of the construction of the test suite is essential in order to reduce the time spent constructing it. Also, predictability of the test suites size and structure is important, because a method might create different test suite sizes at different execution times. There are many different approaches to these algorithms, including algebraic and combinatorial constructions, computational methods and greedy algorithms.

A recursive construction based on orthogonal arrays called TConfig was developed by Williams and Probert [33, 34]. Their method provides a worst-case guarantee on test suite size that is optimal up to a constant factor. But it does not provide for seeds or avoids, and often generates test suites that are much larger than required (especially for mixed-level covering arrays).

Computational search techniques such as tabu search, simulated annealing and hill climbing take a covering array through a series of transformations, computing the cost of a change and accepting the change according to an acceptance criterion. Hill climbing and tabu search produce the smallest test suites, and they provide for seeds. But they fail to produce a worst-case guarantee on test suite size, and they take longer to construct. Simulated annealing currently appears to be the best method available for minimizing the size of the test suite [14].

Greedy algorithms address seeding and constraints, and are fast and relatively accurate at generating test suites. For pairwise testing, greedy algorithms have been well studied [7, 8, 9, 12]. The AETG, Test Case Generator (TCG), In-Parameter-Order (IPO) algorithms and Deterministic Density Algorithm (DDA) use greedy techniques. For AETG and TCG algorithms, each test suite is built one test at a time, and for each subsequent test to be added, many are created and the best is then chosen [12]. The algorithms are greedy in determining which value to add to each factor of each test based on the local optimum. Unlike the other greedy algorithms, the IPO algorithm does not generate a test suite one row at a time, instead it generates one factor at a time using a greedy strategy. They do not usually produce the smallest test suites, but they are able to treat a wide variety of parameters and incorporate additional requirements such as the addition of levels to factors. The DDA constructs one row of a covering array at a time based on a steepest ascent approach. Factors are dynamically fixed one at a time in an order based on density. New rows are continually generated until all t-tuples have been covered. The DDA is particularly important, as it generates covering arrays with a logarithmic guarantee on the size of the solution as a function of the number of factors k, thus providing bounds on N in terms of k. In addition, computational results show that DDA often provides better solutions than other algorithms in its class [13].

2.3 Ordered Orthogonal Arrays and (t,m,s)-Nets

2.3.1 Introduction to Ordered Orthogonal Arrays

A combinatorial object called an *ordered orthogonal array* (OOA) generalizes orthogonal arrays by imposing balance conditions on certain sets of columns. In order to define OOAs, we first need to define a *left-justified* set:

Definition 2.3.1. *Given positive integers s and l, and an array with sl columns, labelled $\{(i,j) : 1 \leq i \leq s, 1 \leq j \leq l\}$, a set T of columns is **left-justified** if $(i,j) \in T$ with $j > 1$, then $(i, j-1) \in T$.*

Definition 2.3.2. *Given positive integers λ, t, s, l and v, where $l \leq t \leq sl$, an **ordered orthogonal array** $OOA_\lambda(t,s,l,v)$ is a $\lambda v^t \times sl$ array A, with columns indexed by ordered pairs (i,j) and with elements from an alphabet V of size v, with the property that every left-justified set T of t columns has the OA property: in the subarray obtained by restricting to columns lying in T, each t-tuple over V occurs exactly λ times as a row.*

In an $OOA_\lambda(t,s,l,v)$, t denotes the strength of the array, s is the number of *parts* it contains, l is the number of columns in each part, and v is the alphabet size. In the above definition, we restrict the value of the parameter l, such that $l \leq t$. We note that we can have an $OOA_\lambda(t,s,l,v)$ with $l > t$, but as a result, columns $(t+1)$ to l of each of the s parts will never interact with any other columns in the array, since they can never belong to a left-justified set of t columns.

Given integers (t,s,l,v), we call them *admissible* if they satisfy the necessary conditions for the existence of an $OOA_\lambda(t,s,l,v)$. If (t,s,l,v) is an admissible set then $l \leq t \leq sl$.

Remark 2.3.3. *When $l = 1$, an $OOA_\lambda(t,s,1,v)$ is just an $OA_\lambda(t,s,v)$.*

Example 2.3.4. *The following is an $OOA(3,3,3,2)$:*

$OOA(3,3,3,2)$

0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0	0
0	1	0	0	1	0	1	1	0
0	1	1	1	1	0	0	1	0
1	0	0	0	0	1	1	0	1
1	0	1	1	0	1	0	0	1
1	1	0	0	1	1	0	1	1
1	1	1	1	1	1	1	1	1

This OOA of strength $t = 3$ has a total of $sl = 9$ columns, and it is divided into 3 parts, where each part contains 3 columns. Within each part, all 3-tuples formed from the values of 0 and 1 are covered exactly once, since the 3 columns within each part form a left-justified set. The first columns of each part also form a left-justified set, which covers all 3-tuples. The first two columns of one part along with the first column of another part form a left-justified set of 3 columns which covers all 3-tuples.

The following simple theorems show some important results for ordered orthogonal arrays:

Theorem 2.3.5. *An $OOA_\lambda(sl, s, l, v)$ is equivalent to an $OA_\lambda(sl, sl, v)$.*

Proof. Since an $OOA_\lambda(sl, s, l, v)$ has strength sl, and the OOA has a total of sl columns, there exists only one left-justified set of sl columns which is the set of all columns of the OOA. An array with sl columns and strength, in which all sl-tuples are covered λ times is just an $OA_\lambda(sl, sl, v)$. □

Corollary 2.3.6. *For all positive integers λ, s, l and v, there exists an $OOA_\lambda(sl, s, l, v)$.*

Proof. From Theorem 2.3.5, we know that an $OOA_\lambda(sl, s, l, v)$ is equivalent to an $OA_\lambda(sl, sl, v)$. From Theorem 2.1.6, we know that that an $OA_\lambda(sl, sl, v)$ always exists for positive integers λ, s, l and v. □

Corollary 2.3.7. *For all positive integers λ, t and v, there exists an $OOA_\lambda(t, 2, t, v)$.*

Proof. From Theorem 2.1.6, we know that there exists an $OA_\lambda(t, t, v)$ for all positive integers λ, t and v. We form an $OOA_\lambda(t, 2, t, v)$ by placing $OA_\lambda(t, t, v)$ in the first part of the OOA, and then reversing the order of each of the entries of each row of $OA_\lambda(t, t, v)$ for the second part of the OOA. So for $1 \leq j \leq t$, the j^{th} column of the first part of the OOA, will be the $(t - j + 1)$ column of the second part. This construction works since $j + t - j + 1 = t + 1 > t$, thus every left-justified set of t columns of the $OOA_\lambda(t, 2, t, v)$ is constructed from a distinct set of t columns which satisfy the OA property. □

2.3.2 (t,m,s)-Nets and their Application

Point sets called *(t, m, s)-nets* are related to OOAs. A *(t, m, s)-net* is a set of points in $\mathbb{R}^s$, which satisfy strong uniformity properties with regard to their distribution in the s-dimensional unit cube. They were first introduced by Niederreiter [31] in 1987.

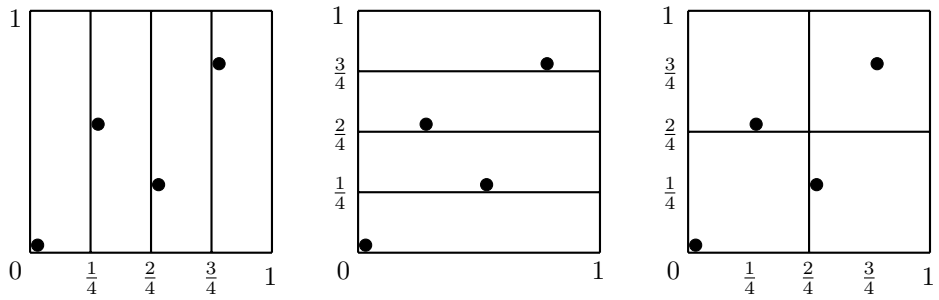
Definition 2.3.8. *[31] For integers $s \geq 1$ and $b \geq 2$, an **elementary interval in base b** is an interval of the form $E = \prod_{i=1}^{s} [\frac{a_i}{b^{d_i}}, \frac{a_i+1}{b^{d_i}})$, where a_i and d_i are non-negative integers such that $0 \leq a_i < b^{d_i}$ for $1 \leq i \leq s$.*

Definition 2.3.9. *[31] For integers $0 \leq t \leq m$, $s \geq 1$ and $b \geq 2$, a **(t,m,s)-net in base b** is a point set of b^m points in the s-dimensional hypercube $[0, 1)^s$ such that every elementary interval in base b having volume b^{t-m} contains exactly b^t points.*

Usually the points of a (t, m, s)-net are shifted by a small positive value, $\epsilon \in \mathbb{R}$, in all s-dimensions, in order for them to lie in the intervals and not on the boundaries of the intervals. For example, a point $(0, 0)$ that lies in a $(t, m, 2)$-net is actually $(0 + \epsilon, 0 + \epsilon)$.

Example 2.3.10. *A (0,2,2)-net in base 2 has a total of $b^m = 2^2 = 4$ points, each point is 2-dimensional, and there is exactly $b^t = 2^0 = 1$ point in every elementary interval of volume $b^{t-m} = 2^{0-2} = \frac{1}{4}$. A set of points of a (0,2,2)-net in base 2 are: $\{(0, 0), (\frac{1}{4}, \frac{1}{2}), (\frac{1}{2}, \frac{1}{4}), (\frac{3}{4}, \frac{3}{4})\} + (\epsilon, \epsilon)$*

These 4 points are shown in the figure below, together with all 12 possible elementary intervals of volume $\frac{1}{4}$:



For a (t, m, s)-net in base b, if s and b are fixed, then a larger value for $m - t$ will result in a smaller volume for the elementary interval b^{t-m}. Decreasing the volume of the elementary interval results in better uniformity properties for the net. A larger value for $m-t$ can be achieved by decreasing the value of t. Constructing (t, m, s)-nets with parameter t as close to zero as possible is preferred [27], since this results in better equidistribution properties for the net. In particular, if the value of the parameter t is zero, then there will be exactly $b^0 = 1$ point in every elementary interval of volume b^{t-m}.

There are many methods that have been developed for the construction of (t, m, s)-nets. The most common methods include direct constructions using properties of finite fields [31, 32], coding theoretic constructions [28] and one which characterizes them using orthogonal arrays [31] and ordered orthogonal arrays [24, 30]. This thesis deals with the latter construction.

The following theorem states the equivalence between a $(0, 2, s)$-net in base b and mutually orthogonal Latin squares:

Theorem 2.3.11. *[31] Let $s \geq 2$ and $b \geq 2$ be integers. There there exists a $(0, 2, s)$-net in base b if and only if there exists $s - 2$ MOLS of order b. (More generally, a $(t, t + 2, s)$-net in base b is equivalent to an $OA_{b^t}(2, s, b)$.)*

The following is a corollary of Theorem 2.1.7, used to construct larger nets:

Corollary 2.3.12. *[27] If there exists a $(0, 2, s_1)$-net in base b_1 and a $(0, 2, s_2)$-net in base b_2, then there exists a $(0, 2, min(s_1, s_2))$-net in base $b_1 b_2$.*

Proof. If a $(0, 2, s_1)$-net in base b_1 and a $(0, 2, s_2)$-net in base b_2 exist, then an $OA(2, s_1, b_1)$ and an $OA(2, s_2, b_2)$ exist. From Theorem 2.1.7, since both OAs exist, then an $OA(2, min(s_1, s_2), b_1 b_2)$ exists, which is equivalent to a $(0, 2, min(s_1, s_2))$-net in base $(b_1 b_2)$. □

A more general and different equivalence was proposed by Lawrence [24] and Mullen and Schmid [30] independently, which equate (t, m, s)-nets to a parametric subclass of ordered orthogonal arrays:

Theorem 2.3.13. *[24, 30] There exists a (t, m, s)-net in base b if and only if there exists an $OOA_{b^t}(m - t, s, m - t, b)$.*

The left-justified sets of an OOA are equivalent to the elementary intervals of its corresponding (t, m, s)-net.

Remark 2.3.14. *The parameter t from (t, m, s)-nets is not that of an $OOA(t, s, l, v)$. Both are standard usages within their domain. In cases of possible confusion, when using both objects at the same time, we will refer to the strength of the OOA by the parameter u.*

A (t, m, s)-net in base b can be constructed from an OOA by translating each row in the OOA to a point in the (t, m, s)-net. This is done by first placing a decimal point before each part of each row of the OOA, so that each point has s coordinates corresponding to the s parts of the OOA. We then take the decimal expansion in base b for each point, in order to translate them to points in an s-dimensional hypercube, $[0, 1)^s$. We give the following simple example to demonstrate this.

Example 2.3.15. *[27] We will construct a (0,2,2)-net in base 2 from an $OOA(2, 2, 2, 2)$.*

$OOA(2, 2, 2, 2)$

0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	1

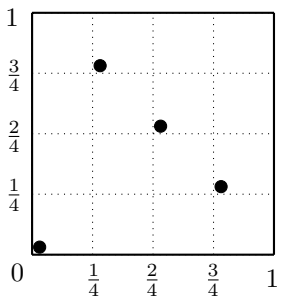
We take the values of each row in each part, and put a decimal point to their left, to form 4 points of 2-dimensions each. For example, the second row of the OOA is 0 1 | 1 1, where the first 2 numbers are from the first part, and the next 2 numbers are from the second part. By placing decimal points before the entries for each part, and interpreting as binary real numbers, we get .01 and .11. Therefore the point corresponding to the second row of the OOA is (.01,.11). By following the same process, we get the following 4 points for all 4 rows: (.00,.00), (.01,.11), (.10,.10), (.11,.01).

We can translate the values .00, .01, .10 and .11 from binary to their decimal expansion in base 2:

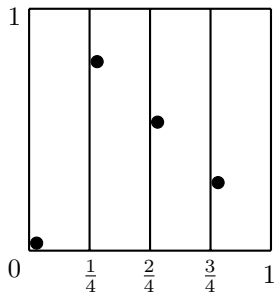
Binary	Decimal
.00	$(0 * 2^{-1}) + (0 * 2^{-2}) = 0$
.01	$(0 * 2^{-1}) + (1 * 2^{-2}) = \frac{1}{4}$
.10	$(1 * 2^{-1}) + (0 * 2^{-2}) = \frac{1}{2}$
.11	$(1 * 2^{-1}) + (1 * 2^{-2}) = \frac{3}{4}$

And by substituting the above points into our original 4 points, yields the following (0,2,2)-net in base 2 (after adding a small positive value ϵ to each coordinate): $\{(0, 0), (\frac{1}{4}, \frac{3}{4}), (\frac{2}{4}, \frac{2}{4}), (\frac{3}{4}, \frac{1}{4})\} + (\epsilon, \epsilon)$

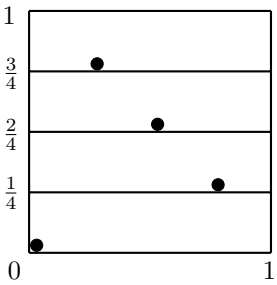
This is shown in the graph below:



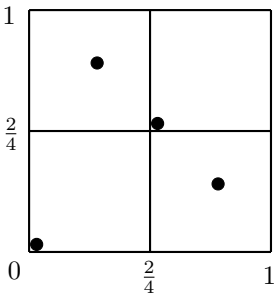
In this example, the two columns of the first part of the $OOA(2,2,2,2)$ correspond to the following elementary intervals in the (0,2,2)-net base 2:



The two columns of the second part of the $OOA(2,2,2,2)$ correspond to the following elementary intervals in the (0,2,2)-net base 2:



And the first columns of each part of the $OOA(2,2,2,2)$ correspond to the following elementary intervals in the (0,2,2)-net base 2:



(t,m,s)-nets were defined by Niederreiter [31] in the context of quasi-Monte Carlo methods of numerical integration. Numeric quadrature methods such as Simpson's rule and trapezoidal rule are highly effective in approximating the value of a low dimensional integral. However, these classical methods are impractical for multi-dimensional integrals since the number of function evaluations grow exponentially as the number of dimensions increase. The Monte Carlo (MC) method is commonly used in computing integrals of multiple dimensions. This method uses randomly distributed points in order to approximate the integral. However, truly random, or pseudorandom, sequences tend to cluster together, which means certain areas are over represented while others are under represented. The amount of clustering in a

sequence can be measured by the *discrepancy*. A *low discrepancy* sequence is a sequence of points that fills the area more uniformly than uncorrelated random points. Quasi-Monte Carlo (QMC) integration operates the same way as Monte Carlo integration but it uses a sequence of low discrepancy points called quasi-random points as opposed to pseudorandom points to approximate the value of the integral. The efficiency of the quasi-Monte Carlo integration method depends on the discrepancy or lack of clustering of the quasi-random points that are used. There are different methods that can be used to construct such a set of quasi-random points, with (t, m, s)-nets [32] being one of the most powerful and effective known constructions for these low discrepancy point sets in the s-dimensional unit cube. For more information on low discrepancy, see [31].

The difference in using quasi-random versus pseudorandom numbers to approximate the value of an integral affects the convergence rate of the integral. The convergence rate measures how quickly the error decreases with the number of simulations. For MC integration, the convergence rate is $O(\frac{1}{\sqrt{N}})$, where N is the number of simulations [32]. While for QMC integration, the rate of convergence can be a lot faster, with a rate of $O(\frac{1}{N})$ for optimal cases [4, 27].

Example 2.3.16. *We will numerically approximate the value of the integral* $\int_0^\pi (\int_0^\pi sin(xy)dy)dx$*, using Monte Carlo and quasi-Monte Carlo integration, and compare the results to the exact value of the integral.*

The exact value of $\int_0^\pi (\int_0^\pi sin(xy)dy)dx$ *is 2.90068.*

For the quasi-Monte Carlo method, we use an $OOA(2, 2, 2, 10)$ *to construct a (0,2,2)-net in base 10, to numerically approximate the value of the integral. Using the hundred points from a (0,2,2)-net in base 10, gives us an approximation of 2.86097, which differs from the exact value by 1.34%.*

For the Monte Carlo method, keeping the number of points used the same as in the QMC evaluation above, we use one hundred random numbers generated through MATLAB, to numerically approximate the value of the integral. This gives us a different approximation each time, depending on the value of the random numbers generated. Using one hundred points each time, some of the different values approximated for the integral were 2.4171, 2.3379 and 3.3159, which differ from the exact value by 16.67%, 19.4% and 14.31%, respectively. Therefore, by using the same number of points for both methods, each approximation in the MC method is a lot worse than that computed through the QMC method.

Chapter 3

Constructions for Ordered Orthogonal Arrays

Most known constructions for ordered orthogonal arrays involve methods from areas such as coding theory and finite projective geometry [2, 3]. In this chapter, we construct ordered orthogonal arrays from orthogonal arrays by developing a homomorphism between the two objects, and we also provide new combinatorial constructions based on recursive and multiplicative methods.

3.1 The Interaction Graph

We can construct an OOA from the columns of an OA using the following theorem:

Theorem 3.1.1. *If there exists an $OA(t, k, v)$, and integers $s \geq 2$, $l \leq t$, such that:*

$$k \geq \begin{cases} \frac{st}{2} & \text{if } t \text{ is even and } l > \frac{t}{2} \\ \lfloor \frac{t}{2} \rfloor s + 1 & \text{if } t \text{ is odd and } l > \lfloor \frac{t}{2} \rfloor \\ sl & \text{if } l \leq \lfloor \frac{t}{2} \rfloor, \end{cases}$$

then there exists an $OOA(t, s, l, v)$.

Proof. For the first case where t is even and $l > \frac{t}{2}$: We place $\frac{t}{2}$ distinct columns from the OA to the beginning of each part of the OOA, this uses $\frac{t}{2}s$ columns from the OA. We complete all remaining columns of the new OOA by putting the column that was placed in the $\frac{t}{2} - (i-1)^{\text{th}}$ position of the $(j+1) \bmod s$ part in the $\frac{t}{2} + i^{\text{th}}$ position of the j^{th} part, where $1 \leq i \leq \frac{t}{2}$ and $1 \leq j \leq s$. We note that $\frac{t}{2} - (i-1) + \frac{t}{2} + i = t + 1 > t$, so these columns never interact.

For the second case where t is odd and $l > \lfloor \frac{t}{2} \rfloor$: We place $\lfloor \frac{t}{2} \rfloor$ distinct columns from the OA to the beginning of each part of the OOA, this uses $\lfloor \frac{t}{2} \rfloor s$ columns from the OA. Then we fix a new column from the OA to the $(\lfloor \frac{t}{2} \rfloor + 1)^{\text{th}}$ position of each part of the OOA. We complete all remaining columns of

the new OOA by putting the column that was placed in the $\lfloor \frac{t}{2} \rfloor - (i-1)^{\text{th}}$ position of the $(j+1)$ mod s part in the $\lfloor \frac{t}{2} \rfloor + (i+1)^{\text{th}}$ position of the j^{th} part, where $1 \leq i \leq \lfloor \frac{t}{2} \rfloor$ and $1 \leq j \leq s$. We note that $\lfloor \frac{t}{2} \rfloor - (i-1) + \lfloor \frac{t}{2} \rfloor + i + 1 = \lfloor \frac{t}{2} \rfloor + \lfloor \frac{t}{2} \rfloor + 2 = t - 1 + 2 > t$, so these columns never interact.

For the third case where $l \leq \lfloor \frac{t}{2} \rfloor$: We place l distinct columns from the OA in each part of the OOA, this uses sl columns from the OA. In the resulting OOA, all columns are distinct, therefore any left-justified set of t columns will interact. □

Example 3.1.2. *We will use an $OA(4, 8, v)$ to construct an $OOA(4, 4, 4, v)$, using the construction given in the above theorem.*

Our OOA has $l = 4 \Rightarrow l > \lfloor \frac{t}{2} \rfloor = 2$. Following Theorem 3.1.1, since our strength $t = 4$ is even, the minimum number of columns required for our OA is: $k = \frac{st}{2} = \frac{(4)(4)}{2} = 8$. So the smallest OA we can use to construct an OOA(4,4,4,v) is an OA(4,8,v).

Let the 8 columns of an $OA(4, 8, v)$ be labelled 1 to 8: [1 2 3 4 5 6 7 8]

We fix 2 distinct columns from the OA to the beginning of each part of the OOA:

*[1 2 * * | 3 4 * * | 5 6 * * | 7 8 * *]*

We fill the remaining two columns of the first part of the OOA using the first two columns of the second part of the OOA. We place the $2-(i-1)^{th}$ column from the second part in the $2+i^{th}$ position of the first part, where $1 \leq i \leq 2$. Therefore, we place the second column from the second part of the OOA in the third position of the first part of the OOA. We place the first column from the second part of the OOA in the fourth position of the the first part of the OOA. We get the following array:

*[1 2 4 3 | 3 4 * * | 5 6 * * | 7 8 * *]*

We repeat this for the remaining six columns of the OOA to get the following OOA(4,4,4,v):

[1 2 4 3 | 3 4 6 5 | 5 6 8 7 | 7 8 2 1].

Note that the mapping of the columns described in the construction of Theorem 3.1.1 above is actually a graph homomorphism, we now generalize this notion. We will describe a graph homomorphism mapping the columns of an OOA, which preserves the OOA property. We begin by defining a graph homomorphism:

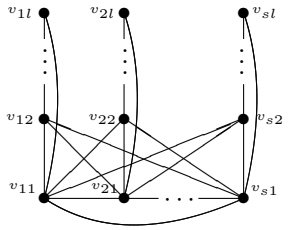
Definition 3.1.3. *A **graph homomorphism** from a graph $G = (V, E)$ to a graph $G' = (V', E')$, is a mapping f from V to V' such that if $uv \in E$ then $f(u)f(v) \in E'$.*

The idea of using a graph to model interaction patterns has been studied in the context of covering arrays [19]. We apply this notion to ordered orthogonal arrays by introducing a new graph called an *interaction graph* which characterizes the interaction properties of an OOA. In an $OOA(t, s, l, v)$, we say that two columns *can interact* with each other if it is possible to form a left-justified set of t columns containing both columns.

Definition 3.1.4. *The **interaction graph** of an $OOA(t, s, l, v)$ is a graph which has a vertex for each column of the OOA and an edge between two columns if they can interact with each other to form a t-tuple as part of a left-justified set.*

It is easy to see that the interaction graph characterizes the left-justified property: any set of t columns of the OOA form a left-justified set if and only if the corresponding nodes in the graph form a clique. Note that a graph homomorphism can never map an interacting set of vertices together as this would form a loop and the original graph has no loops.

The interaction graph of an OOA has a particular form. To represent this, we place the columns of each part vertically above one another, with the bottom node representing the first column of that part, and the top node representing the last column of that part. An interaction graph of an $OOA(t, s, l, v)$ is shown in the figure below. The node representing column j of part i of the OOA is represented by v_{ij}.



The interaction graph of an OOA is completely determined by the parameters of the OOA and exists for all admissible values of the parameters independently of the existence of the OOA. We denote such an interaction graph by $I(t, s, l)$.

The following theorem describes the homomorphism:

Theorem 3.1.5. *If there exists an $OOA(t, s, l, v)$ and a graph homomorphism from $I(t, s', l')$ to the interaction graph of the $OOA(t, s, l, v)$, then the $OOA(t, s', l', v)$ exists.*

Proof. Let A be an $OOA(t, s, l, v)$ and H be the interaction graph of A. To create an $OOA(t, s', l', v)$, say B, we let G be an $I(t, s', l')$. Let $f : G \longrightarrow H$ be a graph homomorphism. For $1 \leq i \leq s'l'$, we place column $f(i)$ from A as the i^{th} column of B. Suppose $c_1, \ldots, c_t$ is a left-justified set of columns in B, so the nodes corresponding to them form a clique in G. Since f is a homomorphism, the nodes corresponding to $f(c_1), \ldots, f(c_t)$ form a clique in H, and since A is an OOA, all pairs from this set of columns are covered exactly once in the rows of A, which are now also the rows of B. □

The interaction graph of an $OA(t, k, v)$ is K_k, which leads to the following corollary:

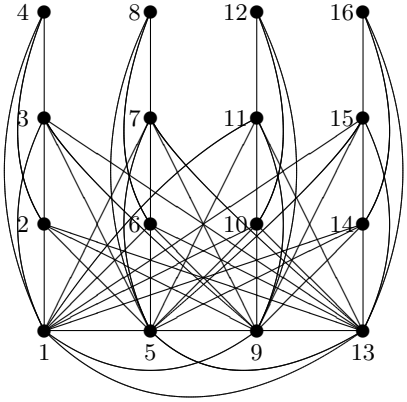
Corollary 3.1.6. *If there exists an $OA(t, k, v)$ and a graph homomorphism mapping $I(t, s, l)$ to K_k, then an $OOA(t, s, l, v)$ exists.*

Note that Theorem 3.1.1 is an example of the above corollary.

Example 3.1.7. *This example maps the 16 columns of an $OOA(4, 4, 4, v)$ to the 8 columns of an $OA(4, 8, v)$. Let the 16 columns of an $OOA(4, 4, 4, v)$ be labelled 1 to 16.*

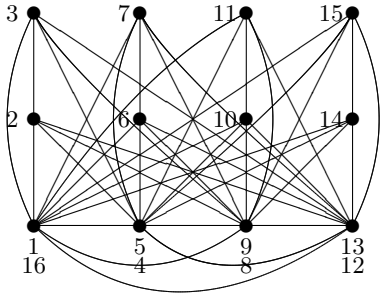
$OOA(4, 4, 4, v)$: [1 2 3 4 | 5 6 7 8 | 9 10 11 12 | 13 14 15 16]

Below is the corresponding $OOA(4, 4, 4, v)$ graph:



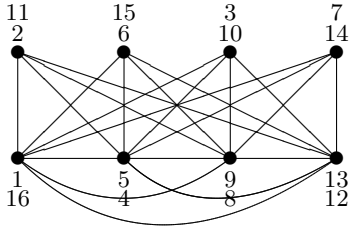
We start by mapping the fourth columns of each part (the top row of nodes) to the first columns of a different part (the bottom row of nodes), since the fourth columns will never interact with the first columns of a different part (since strength t=4):

Mapping:
$4 \to 5$
$8 \to 9$
$12 \to 13$
$16 \to 1$



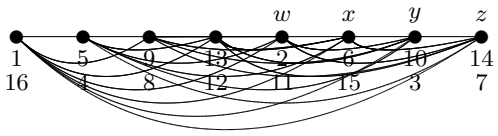
We map the third columns of each part to the second columns of a different part (but it cannot get mapped to the same part its fourth column was mapped to), since the third columns will never interact with the second columns of another part (since strength t=4):

Mapping:
$3 \to 10$
$7 \to 14$
$11 \to 2$
$15 \to 6$



We need to map the second columns of each part to a column, but they cannot be mapped to any of the first columns of the parts because they interact with them. So we need to introduce four new nodes to the graph, which represent columns w, x, y, z. Each of the second columns gets mapped to one of w, x, y, z:

Mapping:
2, 11 $\to w$
6, 15 $\to x$
3, 10 $\to y$
7, 14 $\to z$

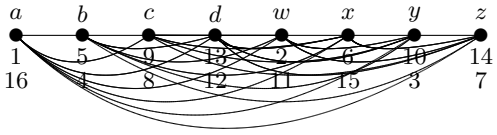


The above graph is a K_8, and it also represents an $OA(4, 8, v)$. Each node in the graph represents a column of the OA, so we require a total of 8 columns to construct our $OOA(4, 4, 4, v)$. If we label the first 4 nodes of the OA with the letters a, b, c, d, then the columns of our OA are: a, b, c, d, w, x, y, z.

The 16 columns of the OOA (labelled 1 to 16) are mapped as follows to the 8 columns of the OA (labelled a, b, c, d, w, x, y, z):

Mapping:

$1 \to a$	$9 \to c$
$2 \to w$	$10 \to y$
$3 \to y$	$11 \to w$
$4 \to b$	$12 \to d$
$5 \to b$	$13 \to d$
$6 \to x$	$14 \to z$
$7 \to z$	$15 \to x$
$8 \to c$	$16 \to a$



So the columns of the $OOA(4, 4, 4, v)$ are: $[a\ w\ y\ b \mid b\ x\ z\ c \mid c\ y\ w\ d \mid d\ z\ x\ a]$.

We can generalize this example in the following theorem:

Theorem 3.1.8. *For $s > \frac{t}{2}$, if t is even and there exists an $OA(t, \frac{t}{2}s, v)$, then there exists an $OOA(t, s, t, v)$, and if t is odd and there exists an $OA(t, \lfloor \frac{t}{2} \rfloor s+1, v)$, then there exists an $OOA(t, s, t, v)$.*

Proof. We first consider the case when t is even:

Let A be an $OA(t, \frac{t}{2}s, v)$ and H be the interaction graph of A, which is just a $K_{\frac{t}{2}s}$. To create an $OOA(t, s, t, v)$, say B, we let G be an $I(t, s, t)$. By Theorem 3.1.5, if there exists a graph homomorphism from G to H, then B exists. Hence, we need to give a graph homomorphism mapping the vertices from G to the vertices in H. For $1 \leq i \leq s$, $1 \leq j \leq t$, let $w_{i,j}$ be the vertices of G and let $U = \{u_1, ..., u_{\lfloor \frac{t}{2} \rfloor s}\}$ be the vertices of H. For $1 \leq i \leq s$, we define the following homomorphism f mapping the vertices of each column of G to the vertices in H, where all arithmetic is mod $\left(\lfloor \frac{t}{2} \rfloor s\right)$:

$$f(w_{i,j}) = \begin{cases} u_{(i+1)+(s+1)(t-j)} & \text{for } \lceil \frac{t}{2} \rceil < j \leq t, \\ u_{i+s(j-1)} & \text{for } 1 \leq j \leq \lfloor \frac{t}{2} \rfloor. \end{cases}$$

We now consider the case when t is odd:

This case is similar to the even case, except that we modify the mapping f to add the case where $j = \lceil \frac{t}{2} \rceil$. Let A be an $OA(t, \lfloor \frac{t}{2} \rfloor s + 1, v)$, with interaction graph H with vertices $U \cup \{u_\infty\}$, where U is the same as in the even case. The same mapping f, as in the even case above, still applies, we add the extra case, $f(w_{i,\lceil \frac{t}{2} \rceil}) = u_\infty$ for $1 \leq i \leq s$.

□

3.2 Further Constructions for OOAs

In this section, we will be constructing an $OOA(sl + t', s, l + l', v)$ from an $OOA(sl, s, l, v)$ and an $OOA(t', s, l', v)$. For admissible parameter sets (sl, s, l, v), (t', s, l', v) and $(sl + t', s, l + l', v)$, we first explain some of the terms which will be using in the following lemma and theorem.

- $\lfloor \frac{sl+t'}{l+l'} \rfloor$ is the maximum number of full parts, each of $(l+l')$ columns, formed from an $(sl+t')$-tuple in an $OOA(sl + t', s, l + l', v)$.
- $sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor (l + l')$ is the number of columns remaining in an $(sl + t')$-tuple, after removing all complete parts of $(l + l')$ columns.
- $\lfloor \frac{sl+t'}{l+l'} \rfloor l'$ is the number of sl' columns that is used to form all complete parts of $(l + l')$ columns of an $(sl + t')$-tuple in an $OOA(sl + t', s, l + l', v)$. The strength t' must be at least as large as this number, otherwise certain interactions will not be covered in the the corresponding $(sl + t')$-tuple. Therefore, the condition $t' \geq \lfloor \frac{sl+t'}{l+l'} \rfloor l'$ must hold.

We now proceed with the lemma:

Lemma 3.2.1. *Let (sl, s, l, v), (t', s, l', v) and $(sl + t', s, l + l', v)$ be admissible parameter sets, and $t' < sl'$, then the following three conditions must hold:*

1. $sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor (l + l') \geq l$
2. $\frac{sl+t'}{l+l'} \notin \mathbb{Z}$
3. $s \leq \lfloor \frac{sl+t'}{l+l'} \rfloor + 1$

Proof. 1. $sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor (l + l') \geq l$

If an $OOA(sl + t', s, l + l', v)$ exists, then the condition $t' \geq \lfloor \frac{sl+t'}{l+l'} \rfloor l'$ must be satisfied. Assume by contradiction that $sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor (l + l') < l$ and $t' \geq \lfloor \frac{sl+t'}{l+l'} \rfloor l'$. We take two cases:

Case 1: $t' = \lfloor \frac{sl+t'}{l+l'} \rfloor l'$

$sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor (l + l') < l \Rightarrow sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor l - \lfloor \frac{sl+t'}{l+l'} \rfloor l' - l < 0.$

Substituting t' in place of $\lfloor \frac{sl+t'}{l+l'} \rfloor l'$ gives $sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor l - t' - l < 0$

$\Rightarrow sl - \lfloor \frac{sl+t'}{l+l'} \rfloor l - l < 0 \Rightarrow s - \lfloor \frac{sl+t'}{l+l'} \rfloor - 1 < 0 \Rightarrow s < \lfloor \frac{sl+t'}{l+l'} \rfloor + 1 \Rightarrow s \leq \lfloor \frac{sl+t'}{l+l'} \rfloor.$

$s = \lfloor \frac{sl+t'}{l+l'} \rfloor \Rightarrow t' = sl'$, which is not possible since $t' < sl'$.

So $s < \lfloor \frac{sl+t'}{l+l'} \rfloor \Rightarrow sl' < \lfloor \frac{sl+t'}{l+l'} \rfloor l'$, and since $t' = \lfloor \frac{sl+t'}{l+l'} \rfloor l' \Rightarrow sl' < t'$, which can never be true since in general $t' \leq sl'$.

Case 2: $t' > \lfloor \frac{sl+t'}{l+l'} \rfloor l'$

$\Rightarrow t' - \lfloor \frac{sl+t'}{l+l'} \rfloor l' > 0$, adding $sl - \lfloor \frac{sl+t'}{l+l'} \rfloor l - l$ to both sides:

$sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor l - \lfloor \frac{sl+t'}{l+l'} \rfloor l' - l > sl - \lfloor \frac{sl+t'}{l+l'} \rfloor l - l,$

and from our assumption, we have:

$0 > sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor (l + l') - l > sl - \lfloor \frac{sl+t'}{l+l'} \rfloor l - l,$

$\Rightarrow 0 > sl - \lfloor \frac{sl+t'}{l+l'} \rfloor l - l \Rightarrow 0 > s - \lfloor \frac{sl+t'}{l+l'} \rfloor - 1 \Rightarrow s < \lfloor \frac{sl+t'}{l+l'} \rfloor + 1 \Rightarrow s \leq \lfloor \frac{sl+t'}{l+l'} \rfloor.$

If $s = \lfloor \frac{sl+t'}{l+l'} \rfloor \Rightarrow t' > sl'$, which can never be true since in general $t' \leq sl'$.

So $s < \lfloor \frac{sl+t'}{l+l'} \rfloor \Rightarrow sl' < \lfloor \frac{sl+t'}{l+l'} \rfloor l' < t' \Rightarrow sl' < t'$, which can never be true since $t' \leq sl'$.

So our initial assumption is false and $sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor (l + l') \geq l$.

2. $\frac{sl+t'}{l+l'} \notin \mathbb{Z}$

Assume by contradiction that $\frac{sl+t'}{l+l'} \in \mathbb{Z}$.

$\Rightarrow \lfloor \frac{sl+t'}{l+l'} \rfloor = \frac{sl+t'}{l+l'}$

Substituting this into condition 1: $sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor (l + l') \geq l$, yields:

$sl + t' - \frac{sl+t'}{l+l'}(l + l') \geq l$

$\Rightarrow sl + t' - (sl + t') \geq l \Rightarrow 0 \geq l$, which cannot be true.

So our initial assumption is false and $\frac{sl+t'}{l+l'} \notin \mathbb{Z}$.

3. $s \leq \lfloor \frac{sl+t'}{l+l'} \rfloor + 1$

Assume by contradiction that $s > \lfloor \frac{sl+t'}{l+l'} \rfloor + 1$.

From condition 1, we have $sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor (l + l') \geq l$. We take 2 cases:

Case 1: $sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor (l + l') = l$

$s > \lfloor \frac{sl+t'}{l+l'} \rfloor + 1 \Rightarrow sl > \lfloor \frac{sl+t'}{l+l'} \rfloor l + l \Rightarrow sl - \lfloor \frac{sl+t'}{l+l'} \rfloor l - l > 0.$

Add $t' - \lfloor \frac{sl+t'}{l+l'} \rfloor l'$ to both sides:

$sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor l - \lfloor \frac{sl+t'}{l+l'} \rfloor l' - l > t' - \lfloor \frac{sl+t'}{l+l'} \rfloor l'$

From our assumption, we know that $sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor (l + l') - l = 0$, substituting this into the left hand side of the above inequality yields:

$0 > t' - \lfloor \frac{sl+t'}{l+l'} \rfloor l' \Rightarrow t' < \lfloor \frac{sl+t'}{l+l'} \rfloor l'.$

If an $OOA(sl+t', s, l+l', v)$ exists, then the condition $t' \geq \lfloor \frac{sl+t'}{l+l'} \rfloor l'$ must be satisfied, but we just showed that $t' < \lfloor \frac{sl+t'}{l+l'} \rfloor l'$.

Case 2: $sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor (l+l') > l$

$s > \lfloor \frac{sl+t'}{l+l'} \rfloor + 1 \Rightarrow sl > \lfloor \frac{sl+t'}{l+l'} \rfloor l + l \Rightarrow sl - \lfloor \frac{sl+t'}{l+l'} \rfloor l - l > 0.$

Add $t' - \lfloor \frac{sl+t'}{l+l'} \rfloor l'$ to both sides:

$sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor l - \lfloor \frac{sl+t'}{l+l'} \rfloor l' - l > t' - \lfloor \frac{sl+t'}{l+l'} \rfloor l'$

Rearranging the above equation, we get:

$\lfloor \frac{sl+t'}{l+l'} \rfloor l' + (sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor l - \lfloor \frac{sl+t'}{l+l'} \rfloor l' - l) > t'.$

This means that all the columns from l' from all the parts that were used to make an $(sl+t')$-tuple has strength larger than t', therefore they cannot form an $OOA(sl+t', s, l+l', v)$ since there are left-justified sets which do not have the OA property.

So our initial assumption is false and $s \leq \lfloor \frac{sl+t'}{l+l'} \rfloor + 1$.

□

Theorem 3.2.2. *For positive integers s, l and v, if there exists an $OOA(t', s, l', v)$, then there exists an $OOA(sl+t', s, l+l', v)$.*

Proof. By Theorem 2.3.6, there exists an $OOA(sl, s, l, v)$. We first give an outline for the construction of the $OOA(sl+t', s, l+l', v)$: For $1 \leq i \leq s$, $1 \leq j \leq l$, and $1 \leq k \leq N$, let $A = (a_{ijk})$ be an $OOA(N;\, sl, s, l, v)$ where i represents the part, j represents the column within a part, and k represents the row of the OOA. Similarly, for $1 \leq i \leq s$, $1 \leq j \leq l'$, and $1 \leq k \leq M$, let $B = (b_{ijk})$ be an $OOA(M;\, t', s, l', v)$. For $1 \leq i \leq s$ and $1 \leq k \leq NM$, form an $NM \times (sl + sl')$ array $C = (c_{ijk})$ by setting:

$$c_{i,j,k} = \begin{cases} a_{i,j,\lceil \frac{k}{M} \rceil} & \text{for } 1 \leq j \leq l, \\ b_{i,j-l,k \bmod M} & \text{for } l+1 \leq j \leq l+l',\ k \,(\mathrm{mod} M) \neq 0, \\ b_{i,j-l,M} & \text{for } l+1 \leq j \leq l+l',\ k \,(\mathrm{mod} M) = 0. \end{cases}$$

Each part of the final OOA is created by concatenating a left hand piece from A and a right hand piece from B. We are adding all M rows of array B to M copies of each row of A, for each part i, where $1 \leq i \leq s$. Note that the construction of an $OOA(sl+t', s, l+l', v)$ must always begin with an $OOA(sl, s, l, v)$, which is equivalent to an orthogonal array $OA(sl, sl, v)$ by Theorem 2.3.5. The construction for an $OOA(sl+t', s, l+l', v)$ is shown in the figure below:

a_{111}	a_{121}	$\cdots$	a_{1l1}	b_{111}	b_{121}	$\cdots$	$b_{1l'1}$	$\cdots$	a_{s11}	a_{s21}	$\cdots$	a_{sl1}	b_{s11}	b_{s21}	$\cdots$	$b_{sl'1}$
a_{111}	a_{121}	$\cdots$	a_{1l1}	b_{112}	b_{122}	$\cdots$	$b_{1l'2}$	$\cdots$	a_{s11}	a_{s21}	$\cdots$	a_{sl1}	b_{s12}	b_{s22}	$\cdots$	$b_{sl'2}$
$\vdots$								$\cdots$	$\vdots$							
a_{111}	a_{121}	$\cdots$	a_{1l1}	b_{11M}	b_{12M}	$\cdots$	$b_{1l'M}$	$\cdots$	a_{s11}	a_{s21}	$\cdots$	a_{sl1}	b_{s1M}	b_{s2M}	$\cdots$	$b_{sl'M}$
a_{112}	a_{122}	$\cdots$	a_{1l2}	b_{111}	b_{121}	$\cdots$	$b_{1l'1}$	$\cdots$	a_{s12}	a_{s22}	$\cdots$	a_{sl2}	b_{s11}	b_{s21}	$\cdots$	$b_{sl'1}$
a_{112}	a_{122}	$\cdots$	a_{1l2}	b_{112}	b_{122}	$\cdots$	$b_{1l'2}$	$\cdots$	a_{s12}	a_{s22}	$\cdots$	a_{sl2}	b_{s12}	b_{s22}	$\cdots$	$b_{sl'2}$
$\vdots$								$\cdots$	$\vdots$							
a_{112}	a_{122}	$\cdots$	a_{1l2}	b_{11M}	b_{12M}	$\cdots$	$b_{1l'M}$	$\cdots$	a_{s12}	a_{s22}	$\cdots$	a_{sl2}	b_{s1M}	b_{s2M}	$\cdots$	$b_{sl'M}$
$\vdots$								$\vdots$	$\vdots$							
a_{11N}	a_{12N}	$\cdots$	a_{1lN}	b_{111}	b_{121}	$\cdots$	$b_{1l'1}$	$\cdots$	a_{s1N}	a_{s2N}	$\cdots$	a_{slN}	b_{s11}	b_{s21}	$\cdots$	$b_{sl'1}$
a_{11N}	a_{12N}	$\cdots$	a_{1lN}	b_{112}	b_{122}	$\cdots$	$b_{1l'2}$	$\cdots$	a_{s1N}	a_{s2N}	$\cdots$	a_{slN}	b_{s12}	b_{s22}	$\cdots$	$b_{sl'2}$
$\vdots$								$\cdots$	$\vdots$							
a_{11N}	a_{12N}	$\cdots$	a_{1lN}	b_{11M}	b_{12M}	$\cdots$	$b_{1l'M}$	$\cdots$	a_{s1N}	a_{s2N}	$\cdots$	a_{slN}	b_{s1M}	b_{s2M}	$\cdots$	$b_{sl'M}$

In choosing an $(sl+t')$-tuple, the first l columns in each part come from A, and we thus know that the first l columns in a part are covered properly by the OA property within each left-justified set in A. For each l-tuple on the left of a part, from A, we have a complete copy of B, thus every $(l+l')$-tuple is covered within a part. It remains to show that every $(sl+t')$-tuple is covered across left-justified sets. We take two cases:

1. $t' = sl'$:
   If $t' = sl'$, then the $OOA(sl, s, l, v)$ and the $OOA(sl', s, l', v)$ are both equivalent to orthogonal arrays, which cover all sl and sl' tuples within every set of l and l' columns respectively. From Theorem 2.3.5, we know that an $OOA(sl, s, l, v)$ and an $OOA(sl', s, l', v)$ are equivalent to an $OA(sl, sl, v)$ and an $OA(sl', sl', v)$ respectively. This construction creates every $(sl + sl')$-tuple, which is equivalent to an $OA(sl + sl', sl + sl', v)$.

2. $t' < sl'$:
   When $t' < sl'$, the conditions of Lemma 3.2.1 will hold, in particular the three properties. In general, we must have $\lfloor \frac{sl+t'}{l+l'} \rfloor \leq s$, since $\lfloor \frac{sl+t'}{l+l'} \rfloor$ tells us how many complete parts we have, each filled with $l + l'$ columns, and we can never have more than s parts. From the third property of Lemma 3.2.1, we know that $\lfloor \frac{sl+t'}{l+l'} \rfloor \geq s - 1$. Combining these two conditions, gives us $s - 1 \leq \lfloor \frac{sl+t'}{l+l'} \rfloor \leq s$. Therefore we have two cases, either $\lfloor \frac{sl+t'}{l+l'} \rfloor = s$ or $\lfloor \frac{sl+t'}{l+l'} \rfloor = s - 1$. Thus, every left-justified set of $sl + t'$ columns either has entries from all parts, or all but one part. We need to show that for both cases, every left-justified set of $sl + t'$ columns covers all $(sl + t')$-tuples:

   (a) If $\lfloor \frac{sl+t'}{l+l'} \rfloor = s$, then an $(sl+t')$-tuple is formed using all s parts of the OOA. The $(sl+t')$-tuple must consist of at least l columns from each part, since $sl + t' > sl$ for $t' \geq 1$. Therefore, we are left with choosing t' columns from the right hand side of sl' columns. We know that any left-justified set of t' columns will interact since these are the left-justified sets of the OOA(t', s, l', v).

(b) If $\lfloor \frac{sl+t'}{l+l'} \rfloor = s-1$, we can only have one leftover part since $\lfloor \frac{sl+t'}{l+l'} \rfloor + 1 = s$. From the first property of Lemma 3.2.1, we know that $sl + t' - \lfloor \frac{sl+t'}{l+l'} \rfloor (l+l') \geq l$, which means that from our leftover part, we must still use at least l columns to form our $(sl+t')$-tuple. So from the $OOA(sl, s, l, v)$, we are always using all sl columns in our $(sl+t')$-tuple since we can never have an $(sl+t')$-tuple with less than l columns in one part. We are left with choosing t' columns from the right hand side of sl' columns. We know that any left-justified set of t' columns will interact since these are the left-justified sets of the OOA(t', s, l', v).

□

Example 3.2.3. *We will construct an* $OOA(7,3,3,2)$ *from an* $OOA(3,3,1,2)$ *and an* $OOA(4,3,2,2)$.

$OOA(3,3,1,2)$

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

$OOA(4,3,2,2)$

0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	1	1	1
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	1	1
0	1	1	1	0	0
1	0	0	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	1	1	0
1	1	0	0	1	1
1	1	0	1	0	0
1	1	1	0	1	0
1	1	1	1	0	1

We start with the $OOA(sl, s, l, v) = OOA(3,3,1,2)$. For all 3 parts of the OOA, we place 16 copies of each value in every row under each other, and besides them we place all the values of the 16 rows of the corresponding part of the $OOA(4,3,2,2)$. This gives us a total of $8 \times 16 = 128$ rows. The resulting array is an $OOA(7,3,3,2)$:

$OOA(7, 3, 3, 2)$

0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0
0	0	0	0	1	0	0	0	1
0	0	0	0	1	1	0	1	1
0	0	1	0	0	0	0	1	0
0	0	1	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	0	1	0	1	1	0	0	0
0	1	0	0	0	0	0	0	1
0	1	0	0	0	1	0	1	1
0	1	0	0	1	0	0	0	0
0	1	0	0	1	1	0	1	0
0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	0	0	0
0	1	1	0	1	0	0	1	0
0	1	1	0	1	1	0	0	1
0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	1	1	0
0	0	0	0	1	0	1	0	1
0	0	0	0	1	1	1	1	1
0	0	1	0	0	0	1	1	0
0	0	1	0	0	1	1	0	1
0	0	1	0	1	0	1	1	1
0	0	1	0	1	1	1	0	0
0	1	0	0	0	0	1	0	1
0	1	0	0	0	1	1	1	1
0	1	0	0	1	0	1	0	0
0	1	0	0	1	1	1	1	0
0	1	1	0	0	0	1	1	1
0	1	1	0	0	1	1	0	0
0	1	1	0	1	0	1	1	0
0	1	1	0	1	1	1	0	1
1	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	1	0
1	0	0	0	1	0	0	0	1
1	0	0	0	1	1	0	1	1
1	0	1	0	0	0	0	1	0
1	0	1	0	0	1	0	0	1
1	0	1	0	1	0	0	1	1
1	0	1	0	1	1	0	0	0
1	1	0	0	0	0	0	0	1
1	1	0	0	0	1	0	1	1
1	1	0	0	1	0	0	0	0
1	1	0	0	1	1	0	1	0
1	1	1	0	0	0	0	1	1
1	1	1	0	0	1	0	0	0
1	1	1	0	1	0	0	1	0
1	1	1	0	1	1	0	0	1
1	0	0	0	0	0	1	0	0
1	0	0	0	0	1	1	1	0
1	0	0	0	1	0	1	0	1
1	0	0	0	1	1	1	1	1
1	0	1	0	0	0	1	1	0
1	0	1	0	0	1	1	0	1
0	0	0	1	0	0	0	0	0
0	0	0	1	0	1	0	1	0
0	0	0	1	1	0	0	0	1

$OOA(7, 3, 3, 2)$ continued:

0	0	0	1	1	1	0	1	1
0	0	1	1	0	0	0	1	0
0	0	1	1	0	1	0	0	1
0	0	1	1	1	0	0	1	1
0	0	1	1	1	1	0	0	0
0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	0	1	1
0	1	0	1	1	0	0	0	0
0	1	0	1	1	1	0	1	0
0	1	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0	0
0	1	1	1	1	0	0	1	0
0	1	1	1	1	1	0	0	1
0	0	0	1	0	0	1	0	0
0	0	0	1	0	1	1	1	0
0	0	0	1	1	0	1	0	1
0	0	0	1	1	1	1	1	1
0	0	1	1	0	0	1	1	0
0	0	1	1	0	1	1	0	1
0	0	1	1	1	0	1	1	1
0	0	1	1	1	1	1	0	0
0	1	0	1	0	0	1	0	1
0	1	0	1	0	1	1	1	1
0	1	0	1	1	0	1	0	0
0	1	0	1	1	1	1	1	0
0	1	1	1	0	0	1	1	1
0	1	1	1	0	1	1	0	0
0	1	1	1	1	0	1	1	0
0	1	1	1	1	1	1	0	1
1	0	1	0	1	0	1	1	1
1	0	1	0	1	1	1	0	0
1	1	0	0	0	0	1	0	1
1	1	0	0	0	1	1	1	1
1	1	0	0	1	0	1	0	0
1	1	0	0	1	1	1	1	0
1	1	1	0	0	0	1	1	1
1	1	1	0	0	1	1	0	0
1	1	1	0	1	0	1	1	0
1	1	1	0	1	1	1	0	1
1	0	0	1	0	0	0	0	0
1	0	0	1	0	1	0	1	0
1	0	0	1	1	0	0	0	1
1	0	0	1	1	1	0	1	1
1	0	1	1	0	0	0	1	0
1	0	1	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	0	1	1	1	1	0	0	0
1	1	0	1	0	0	0	0	1
1	1	0	1	0	1	0	1	1
1	1	0	1	1	0	0	0	0
1	1	0	1	1	1	0	1	0
1	1	1	1	0	0	0	1	1
1	1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	1	0
1	1	1	1	1	1	0	0	1
1	0	0	1	0	0	1	0	0
1	0	0	1	0	1	1	1	0

$OOA(7, 3, 3, 2)$ continued:

1	0	0	1	1	0	1	0	1
1	0	0	1	1	1	1	1	1
1	0	1	1	0	0	1	1	0
1	0	1	1	0	1	1	0	1
1	0	1	1	1	0	1	1	1
1	0	1	1	1	1	1	0	0
1	1	0	1	0	0	1	0	1
1	1	0	1	0	1	1	1	1
1	1	0	1	1	0	1	0	0
1	1	0	1	1	1	1	1	0
1	1	1	1	0	0	1	1	1
1	1	1	1	0	1	1	0	0
1	1	1	1	1	0	1	1	0
1	1	1	1	1	1	1	0	1

We can use the same basic construction to obtain different sets of parameters. Instead of having strength $sl + t'$, our OOA will have a smaller strength of just $l + l'$. But, since the same number of rows will be constructed, now with a smaller tuple to cover, our OOA will have an index $\lambda > 1$. So, we are essentially giving up some of the strength of the OOA in exchange for a larger number of occurrences of each of the smaller tuples.

Theorem 3.2.4. *For positive integers s, l and v, if there exists an $OOA(t', s, l', v)$, then there exists an $OOA_\lambda(l + l', s, l + l', v)$, where $\lambda = v^{sl+t'-l-l'}$.*

Proof. We follow the same construction outlined in the proof of Theorem 3.2.2. By Theorem 2.3.6, there exists an $OOA(sl, s, l, v)$. Let A be an $OOA(sl, s, l, v)$ and B an $OOA(t', s, l', v)$. Each part of the final OOA is created by concatenating a left hand piece from A and a right hand piece from B. The first l columns in each part come from A, and we thus know that the first l columns in a part are covered properly by the orthogonal array property of the left-justified sets in A. For each l-tuple on the left of a part, from A, we have a complete copy of B, thus every $(l + l')$-tuple is covered within a part. To find out the number of occurrences, λ, for each $(l + l')$-tuple, we count the number of rows in each array, A and B, multiply them together, and equate it to the number of rows in an $OOA_\lambda(l + l', s, l + l', v)$. We then solve for λ, to get $v^{sl}v^{t'} = \lambda v^{l+l'} \Rightarrow \lambda = v^{sl+t'-l-l'}$.

It remains to show that every $(l+l')$-tuple is covered λ times across all left-justified sets formed from more than one part of the OOA. To form an $(l + l')$-tuple from the new OOA, there will always be at least l columns from A in the $(l+l')$-tuple. So we take two cases for the choice of the number of columns from A.

1. If there is exactly l columns from A in the $(l + l')$-tuple, then there must be exactly l' columns from B, in the $(l + l')$-tuple. Since arrays A and B both satisfy the OA property within every left-justified set of l and l' columns respectively, then every $(l+l')$-tuple will be covered within the new OOA.

2. If there is more than l columns from A in the $(l+l')$-tuple, then there will be less than l' columns from B, in the $(l + l')$-tuple. We can pick at most sl columns from A, since A has a total of sl

columns. Since array A has strength sl, we know that we can choose up to sl columns from A and they will always satisfy the OA property. We are now left with less than l' columns to be chosen from B. These must also satisfy the OA property, since every left-justified set of $t' \geq l'$ columns of B satisfy the OA property. So every $(l + l')$-tuple will be covered within the new OOA.

The number of times each $(l + l')$-tuple appears follows exactly from the calculation of λ above. □

The following corollary is a direct result of Theorem 3.2.4, taking $l = 1$ and allowing the array to have a positive index λ.

Corollary 3.2.5. *For positive integers s and v, if there exists an $OOA_{\lambda_1}(t, s, l, v)$, then there exists an $OOA_{\lambda_2}(l + 1, s, l + 1, v)$, where $\lambda_2 = \lambda_1 v^{s+t-l-1}$.*

Proof. By Theorem 2.1.6, there exists an $OA(s, s, v)$. The multiplicative construction of Theorem 3.2.4, using an $OA(s, s, v)$ ($\equiv OOA(s, s, 1, v)$) and an $OOA_{\lambda_1}(t, s, l, v)$ gives us an $OOA_{\lambda_2}(l + 1, s, l + 1, v)$. Since the OOA used in the construction has a subscript λ_1, it affects the number of times each $(l + 1)$-tuple is repeated in the $OOA_{\lambda_2}(l + 1, s, l + 1, v)$. To find the value of λ_2, we count the number of rows in the $OA(s, s, v)$ and the $OOA_{\lambda_1}(t, s, l, v)$, multiply them together, and equate it the number of rows in the $OOA_{\lambda_2}(l + 1, s, l + 1, v)$. We then solve for λ_2 in $v^s \lambda_1 v^t = \lambda_2 v^{l+1} \Rightarrow \lambda_2 = \lambda_1 v^{s+t-l-1}$. □

Corollary 3.2.5 can be interpreted in terms of (t, m, s)-nets, where b^s copies of (t_1, m_1, s)-net base b are joined together to create a larger (t, m, s)-net in base b in an s-dimensional hypercube:

Corollary 3.2.6. *If there exists a (t, m, s)-net in base b, then there exists a $(t + s - 1, m + s, s)$-net in base b.*

Proof. By Theorem 2.3.13, a (t_1, m_1, s)-net base b is equivalent to an $OOA_{b^{t_1}}(m_1 - t_1, s, m_1 - t_1, b)$. By Theorem 3.2.5, if there exists an $OOA_{b^{t_1}}(m_1 - t_1, s, m_1 - t_1, b)$, then there exists an $OOA_{\lambda}(m_1 - t_1 + 1, s, m_1 - t_1 + 1, b)$, where $\lambda = b^{t_1+s-1}$. We now translate the last OOA back to a (t, m, s)-net in base b, say (t_2, m_2, s)-net base b. From Theorem 2.3.13, we get the two equations $b^{t_2} = b^{t_1+s-1}$ and $m_2 - t_2 = m_1 - t_1 + 1$. Solving for t_2 and m_2, gives us $t_2 = t_1 + s - 1$ and $m_2 = m_1 + s$. Substituting the values for t_2 and m_2 back into the (t_2, m_2, s)-net base b, gives us a $(t_1 + s - 1, m_1 + s, s)$-net base b. □

Example 3.2.7. *We will construct a (2,5,3)-net base 2 from a (0,2,3)-net base 2. We start with an $OOA(2, 3, 2, 2)$ which is equivalent to a (0,2,3)-net base 2 . We will construct an $OOA_4(3, 3, 3, 2)$ (which is equivalent to a (2,5,3)-net base 2), using an $OA(3, 3, 2)$ and an $OOA(2, 3, 2, 2)$.*

$OA(3,3,2)$

0	0	0
1	0	0
0	1	0
1	1	0
0	0	1
1	0	1
0	1	1
1	1	1

$OOA(2,3,2,2)$

0	0	0	0	1	0
0	1	1	0	0	1
1	0	0	1	0	0
1	1	1	1	1	1

$OOA_4(3,3,3,2)$

0	0	0	0	0	0	0	1	0
0	0	1	0	1	0	0	0	1
0	1	0	0	0	1	0	0	0
0	1	1	0	1	1	0	1	1
1	0	0	0	0	0	0	1	0
1	0	1	0	1	0	0	0	1
1	1	0	0	0	1	0	0	0
1	1	1	0	1	1	0	1	1
0	0	0	1	0	0	0	1	0
0	0	1	1	1	0	0	0	1
0	1	0	1	0	1	0	0	0
0	1	1	1	1	1	0	1	1
1	0	0	1	0	0	0	1	0
1	0	1	1	1	0	0	0	1
1	1	0	1	0	1	0	0	0
1	1	1	1	1	1	0	1	1
0	0	0	0	0	0	1	1	0
0	0	1	0	1	0	1	0	1
0	1	0	0	0	1	1	0	0
0	1	1	0	1	1	1	1	1
1	0	0	0	0	0	1	1	0
1	0	1	0	1	0	1	0	1
1	1	0	0	0	1	1	0	0
1	1	1	0	1	1	1	1	1
0	0	0	1	0	0	1	1	0
0	0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0	0
0	1	1	1	1	1	1	1	1
1	0	0	1	0	0	1	1	0
1	0	1	1	1	0	1	0	1
1	1	0	1	0	1	1	0	0
1	1	1	1	1	1	1	1	1

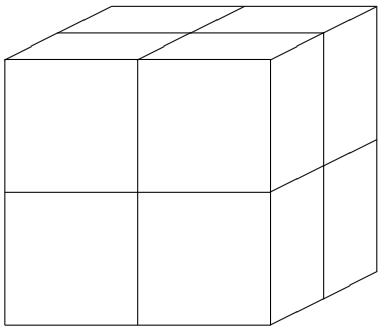
From the above $OOA(2,3,2,2)$, the corresponding (0,2,3)-net base 2 consists of the following 4 points: $\{(0,0,\frac{2}{4}),(\frac{1}{4},\frac{2}{4},\frac{1}{4}),(\frac{2}{4},\frac{1}{4},0),(\frac{3}{4},\frac{3}{4},\frac{3}{4})\}$

From our new $OOA_4(3,3,3,2)$, the corresponding (2,5,3)-net base 2 consists of the following 32 points:

$\{(0,0,\frac{2}{8}),(\frac{1}{8},\frac{2}{8},\frac{1}{8}),(\frac{2}{8},\frac{1}{8},0),(\frac{3}{8},\frac{3}{8},\frac{3}{8}),(\frac{4}{8},0,\frac{2}{8}),(\frac{5}{8},\frac{2}{8},\frac{1}{8}),(\frac{6}{8},\frac{1}{8},0),(\frac{7}{8},\frac{3}{8},\frac{3}{8}),(0,\frac{4}{8},\frac{2}{8}),(\frac{1}{8},\frac{6}{8},\frac{1}{8}),(\frac{2}{8},\frac{5}{8},0),$
$(\frac{3}{8},\frac{7}{8},\frac{3}{8}),(\frac{4}{8},\frac{4}{8},\frac{2}{8}),(\frac{5}{8},\frac{6}{8},\frac{1}{8}),(\frac{6}{8},\frac{5}{8},0),(\frac{7}{8},\frac{7}{8},\frac{3}{8}),(0,0,\frac{6}{8}),(\frac{1}{8},\frac{2}{8},\frac{5}{8}),(\frac{2}{8},\frac{1}{8},\frac{4}{8}),(\frac{3}{8},\frac{3}{8},\frac{7}{8}),(\frac{4}{8},0,\frac{6}{8}),(\frac{5}{8},\frac{2}{8},\frac{5}{8}),$
$(\frac{6}{8},\frac{1}{8},\frac{4}{8}),(\frac{7}{8},\frac{3}{8},\frac{7}{8}),(0,\frac{4}{8},\frac{6}{8}),(\frac{1}{8},\frac{6}{8},\frac{5}{8}),(\frac{2}{8},\frac{5}{8},\frac{4}{8}),(\frac{3}{8},\frac{7}{8},\frac{7}{8}),(\frac{4}{8},\frac{4}{8},\frac{6}{8}),(\frac{5}{8},\frac{6}{8},\frac{5}{8}),(\frac{6}{8},\frac{5}{8},\frac{4}{8}),(\frac{7}{8},\frac{7}{8},\frac{7}{8})\}$

In terms of (t,m,s)-net point sets, through this OOA construction, we have joined eight copies of a (0,2,3)-net base 2 to form a (2,5,3)-net base 2. The corresponding (t,m,s)-net construction is as follows: We add one cube to the right of the original, one directly above the original and one directly above the new cube that was added to the right. We now have four cubes, we make this eight cubes by adding one cube directly behind each of the four cubes. We now have a total of 32 points and 8 intervals, with 4

points/interval, which translates to a (2,5,3)-net base 2. The manner in which the eight cubes are placed altogether is shown in the figure below:



Eight cubes, each representing a (0,2,3)-net base 2, joined to form a (2,5,3)-net base 2.

Chapter 4

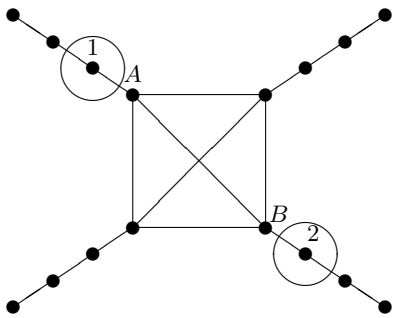
Ordered Covering Arrays

In this chapter, we introduce a new combinatorial object called an *ordered covering array* (OCA), which generalizes covering arrays by imposing balance conditions on certain sets of columns. We discuss several constructions for these objects, using similar recursive and Roux constructions described for covering arrays in Section 2.2.2. We will also look at other theorems, homomorphisms, and multiplicative constructions for OCAs.

4.1 Introduction to Ordered Covering Arrays

Ordered covering arrays have the same relationship to ordered orthogonal arrays as covering arrays have to orthogonal arrays. OCAs relax the exact condition for coverage, just as covering arrays do, so rather than covering t-tuples exactly λ times, we are covering them at least λ times.

In practice, there may exist factors in a test suite that will never interact with each other [19]. So the covering array for such a problem should avoid the t-wise interactions of those factors, but should still cover all the allowable interactions of the test suite. For example, when testing a series of servers represented by network nodes, there is a main hub of servers in the centre, connecting some network nodes to each other, so all nodes within the hub interact with each other. Each network node in the hub, is then connected to its own sub-network of servers, which interact with each other. One way to test such a network is to choose servers within each sub-network and test that they are interconnected (through the hub). For example, from the figure below, we may require Server 1 under hub Node A to interact with Server 2 under hub Node B. This pattern of interconnections between the network nodes/servers can result in the testing of left-justified sets, by allowing certain interactions and restricting others, therefore the interaction test suite can be represented using an ordered covering array.



Definition 4.1.1. *For positive integers λ, t, s, l and v, where $l \leq t \leq sl$, an **ordered covering array** $OCA_\lambda(N; t, s, l, v)$ is an $N \times sl$ array, A, with entries from an alphabet of size v, such that in every left-justified set of t columns, every t-tuple occurs at least λ times as a row. The **ordered covering array number** $OCAN_\lambda(t, s, l, v)$ is the minimum integer N for which an $OCA_\lambda(N; t, s, l, v)$ exists.*

In the above definition, we restrict the value of the parameter l, such that $l \leq t$. We note that we can have an $OCA_\lambda(t, s, l, v)$ with $l > t$, but as a result, columns $(t + 1)$ to l of each of the s parts will never interact with any other columns in the array, since they can never belong to a left-justified set of t columns.

In Section 2.3.1, we defined admissible parameter sets for OOAs. They are defined in the same way for OCAs: Given integers (t, s, l, v), we call them *admissible* if they satisfy the necessary conditions for the existence of an $OCA(t, s, l, v)$. If (t, s, l, v) is an admissible set, then $l \leq t \leq sl$.

Example 4.1.2. *The following is an $OCA(5; 2, 4, 2, 2)$:*

1	0	1	0	1	0	1	0
1	1	0	1	0	0	0	0
0	0	1	1	0	1	0	1
0	1	0	0	1	1	0	0
0	0	0	0	0	0	1	1

Each pair (0,0), (0,1), (1,0), (1,1) is covered at least once within every left-justified set of 2 columns. For example, columns 1 of parts 3 and 4 form a left-justified set, so each pair must be covered at least once. The pairs (0,1), (1,0) and (1,1) are covered exactly once, while (0,0) is covered twice.

Mixed covering arrays allow different alphabet sizes in different columns of the array, as described in Section 2.2.1. Similarly, we can define a *mixed ordered covering array* to vary the alphabet size of different columns of the array:

Definition 4.1.3. *For positive integers λ, t, s, l and $v_{1,1}, ..., v_{s,l}$, a **mixed ordered covering array** $MOCA_\lambda(N; t, s, l, (v_{1,1}, ..., v_{1,l}, ..., v_{s,1}, ..., v_{s,l}))$ is an $N \times sl$ array, where $v_{i,j}$ is the alphabet size for the j^{th} column of the i^{th} part of the OCA for $1 \leq i \leq s, 1 \leq j \leq l$, and every left-justified set of t columns covers all t-tuples at least λ times.*

Example 4.1.4. *The following array is a* $MOCA(6; 2, 2, 2, (2, 2, 3, 2))$*:*

0	0	0	0
0	1	1	0
0	∗	2	0
1	0	0	1
1	1	1	1
1	∗	2	1

*The first, second and fourth columns have alphabet size two, while the third column has alphabet size three. All left-justified sets of two columns cover all pairs at least once. For example, the first columns of each part cover all six pairs formed from the alphabets of size two and three, while the two columns in the first part of the array cover all four pairs (at least once) formed from the alphabets of size two. There are two "do not care" positions which are denoted by the symbol *.*

Note that we cannot have mixed ordered orthogonal arrays because by allowing the alphabet size of different columns to vary, we will force repeated t-tuples for some other columns.

4.2 Recursive and Roux Constructions

For ordered covering arrays of strength two, the same recursive construction for strength two covering arrays (as discussed in Section 2.2.2), can be applied to an OCA and a CA to create an array with a larger number of parts.

Theorem 4.2.1. *If an* $OCA(N; 2, s, l, v)$ *and a* $CA(M; 2, k, v)$ *both exist, then an* $OCA(N + M; 2, sk, l, v)$ *exists.*

Proof. We use a similar method as in the proof of Theorem 2.2.9 [16]. We place k copies of the $OCA(2, s, l, v)$ side by side. Under the first copy of the OCA, we place the first column of the CA under the first columns of each part. We repeat this for all k columns, so that under the first columns of the parts of the i^{th} copy of the OCA, we place the i^{th} column of the CA, for $1 \leq i \leq k$. The result is an $OCA(N + M; 2, sk, l, v)$. For any left-justified set of two columns of the $OCA(N + M; 2, sk, l, v)$, that are either from the same part, or from different parts with different indices mod s, all pairs will be covered from the $OCA(N; 2, s, l, v)$. For any left-justified set of two columns of the $OCA(N + M; 2, sk, l, v)$, that are from different parts, with the same index mod s, all pairs from the two columns of the $OCA(N; 2, s, l, v)$ will be of the form xx for $x \in V$, where V represents the alphabet set of size v. The remaining pairs are covered from the columns of the $CA(M; 2, k, v)$ which were added below. If $l = 2$, in the $(N + 1)$ to $(N + M)$ rows of the resulting OCA, the second columns of each part can be considered "don not care" positions, because the second columns do not interact with each other (or with the first columns of different parts). Their interactions with the first columns of the same part have already been covered in the first N rows (since we placed k copies of the OCA side by side). □

We note that in the above theorem, even if an OOA and an OA are used in the construction, in place of an OCA and a CA, this construction will always result in some pairs being repeated within every left-justified set of two columns, so the resulting array will always be an OCA.

Example 4.2.2. *We will construct an* $OCA(2,6,2,2)$ *from an* $OOA(2,2,2,2)$ *and an* $OA(2,3,2)$*:*

$OOA(2,2,2,2)$

0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1

$OA(2,3,2)$

0	0	0
0	1	1
1	0	1
1	1	0

We place 3 copies of the $OOA(2,2,2,2)$ *side by side. Under the first columns of the two parts of the first copy, we place the first column of the OA. We repeat this for the second and third columns of the OA:*

0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	1	1	0	0	1	1	0
1	0	0	1	1	0	0	1	1	0	0	1
1	1	1	1	1	1	1	1	1	1	1	1
0		0		0		0		0		0	
0		0		1		1		1		1	
1		1		0		0		1		1	
1		1		1		1		0		0	

*Note that the second columns of the 5th to 8th rows of the above array are "do not care" positions, which we will denote by the symbol *. The resulting array is an* $OCA(2,6,2,2)$*:*

0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	1	1	0	0	1	1	0
1	0	0	1	1	0	0	1	1	0	0	1
1	1	1	1	1	1	1	1	1	1	1	1
0	*	0	*	0	*	0	*	0	*	0	*
0	*	0	*	1	*	1	*	1	*	1	*
1	*	1	*	0	*	0	*	1	*	1	*
1	*	1	*	1	*	1	*	0	*	0	*

*By choosing zeros for the * positions of the fifth row, we will get a repeated constant row of zeros. So we can remove the fifth row from the OCA, which gives us the following optimal* $OCA(2,6,2,2)$*:*

0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	1	1	0	0	1	1	0
1	0	0	1	1	0	0	1	1	0	0	1
1	1	1	1	1	1	1	1	1	1	1	1
0	∗	0	∗	1	∗	1	∗	1	∗	1	∗
1	∗	1	∗	0	∗	0	∗	1	∗	1	∗
1	∗	1	∗	1	∗	1	∗	0	∗	0	∗

For ordered covering arrays of strength 3, a Roux-type construction, similar to Theorem 2.2.12, can be applied to OCAs to construct an OCA with a larger number of parts.

Theorem 4.2.3. *$OCAN(3, 2s, l, 2) \leq OCAN(3, s, l, 2) + OCAN(2, s, m, 2)$, where $m = 2$ for $2 \leq l \leq 3$ and $m = 1$ for $l = 1$.*

Proof. We use a similar method as in the Roux construction of Theorem 2.2.12 [35, 38]. To construct an $OCA(3, 2s, l, 2)$, we place two $OCA(N_1; 3, s, l, 2)$'s side by side. For any three columns that form a left-justified set, that either have all three columns from one copy of the OCA, or have columns from both copies, but with no two columns that have the same column and part index, all triples are covered. But for a selection which contains two columns that have the same column and part index, all triples will not be covered, only triples of the form aaa, aab, bba and bbb will be covered for an alphabet set $\{a, b\}$. Therefore, to the bottom of the new array, we add an $OCA(N_2; 2, s, m, 2)$, and its bit complement, side by side, left aligning the parts with the parts of the above array, where $m = 2$ for $l = 2$ or $l = 3$ and $m = 1$ for $l = 1$. Now, for any three columns that form a left-justified set which contains two columns that have the same column and part index, we will get the remaining triplets aba, abb, baa and bab, from the $OCA(N_2; 2, s, m, 2)$ and bit complement that we added. For the $OCA(3, s, l, 2)$, if $l = 3$, then in the $(N_1 + 1)$ to $(N_1 + N_2)$ rows of the resulting OCA, the third columns of each part will be "do not care" positions, since the interaction for all 3 columns within the same part have already been covered in the first N_1 rows of that part, and the third columns of any part of the OCA will not interact with any columns from a different part.

□

Example 4.2.4. *We will construct an $OCA(3, 4, 3, 2)$ from an $OOA(3, 2, 3, 2)$ and an $OOA(2, 2, 2, 2)$. We begin by placing two $OOA(3, 2, 3, 2)$'s side by side:*

0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	1	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0
0	1	1	1	0	0	0	1	1	1	0	0
1	0	0	0	0	1	1	0	0	0	0	1
1	0	1	1	0	1	1	0	1	1	0	1
1	1	0	0	1	1	1	1	0	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1

We construct an $OOA(2,2,2,2)$ *and its bit complement:*

$OOA(2,2,2,2)$

0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1

Complement

1	1	1	1
1	0	0	1
0	1	1	0
0	0	0	0

We add the $OOA(2,2,2,2)$ *and its bit compliment side by side, to the bottom of the two* $OOA(3,2,3,2)$*'s that were placed side by side, in order to get an* $OCA(3,4,3,2)$*:*

0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	1	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0
0	1	1	1	0	0	0	1	1	1	0	0
1	0	0	0	0	1	1	0	0	0	0	1
1	0	1	1	0	1	1	0	1	1	0	1
1	1	0	0	1	1	1	1	0	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1
0	0		0	0		1	1		1	1	
0	1		1	0		1	0		0	1	
1	0		0	1		0	1		1	0	
1	1		1	1		0	0		0	0	

$\longrightarrow$

$OCA(3,4,3,2)$

0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	1	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0
0	1	1	1	0	0	0	1	1	1	0	0
1	0	0	0	0	1	1	0	0	0	0	1
1	0	1	1	0	1	1	0	1	1	0	1
1	1	0	0	1	1	1	1	0	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1
0	0	$*$	0	0	$*$	1	1	$*$	1	1	$*$
0	1	$*$	1	0	$*$	1	0	$*$	0	1	$*$
1	0	$*$	0	1	$*$	0	1	$*$	1	0	$*$
1	1	$*$	1	1	$*$	0	0	$*$	0	0	$*$

As per covering arrays, this can be generalised for all v:

Theorem 4.2.5. $OCAN(3,2s,l,v) \leq OCAN(3,s,l,v) + (v-1)OCAN(2,s,m,v)$*, where* $m = 2$ *for* $2 \leq l \leq 3$ *and* $m = 1$ *for* $l = 1$*.*

Proof. We follow a similar construction as in the proof of Theorem 2.2.14 [11]. We begin by placing two $OCA(N_1;3,s,l,v)$'s side by side. For any three columns that form a left-justified set, that either have all three columns from one copy of the OCA, or have columns from both copies, but with no two columns that have the same column and part index, all triples are covered. But for a selection which contains two columns that have the same column and part index, all triples will not be covered, only triples of the form aax will be covered for $a, x \in V$, where V represents the alphabet set of size v. Let C be an $OCA(N_2;2,s,m,v)$ where $m = 2$ for $2 \leq l \leq 3$ and $m = 1$ for $l = 1$. Let π be a cyclic permutation of the v symbols. For each $1 \leq i \leq v-1$, we append N_2 rows consisting of C and $\pi^i(C)$ placed side by side, below the $OCA(3,s,l,v)$'s, left aligning the parts of C and $\pi^i(C)$ with the corresponding parts of the first and second copies of the $OCA(3,s,l,v)$ respectively. Now, for any three columns that form a left-justified set which contains two columns that have the same column and part index, we will get the remaining triplets which have different alphabets in the first 2 columns, from the columns in arrays C and $\pi^i(C)$ that we added. The construction is shown in the figure below:

$OCA(3, s, l, v)$	$OCA(3, s, l, v)$
$C = OCA(2, s, m, v)$	$\pi^1(C)$
C	$\pi^2(C)$
$\vdots$	$\vdots$
C	$\pi^{v-1}(C)$

For an $OCA(3, s, l, v)$, if $l = 3$, then in the $(N_1 + 1)$ to $(N_1 + N_2)$ rows of the resulting OCA, the third columns of each part will be "do not care" positions. □

See Example 2.2.15 for a similar Roux construction to the one described above, being applied to covering arrays.

4.3 Further Constructions for OCAs

The same theorems and proofs we used to construct ordered orthogonal arrays in Sections 3.1 and 3.2 can be applied to ordered covering arrays.

OCAs can be constructed using the columns of a CA:

Theorem 4.3.1. *If there exists a $CA(t, k, v)$, and integers $s \geq 2$, $l \leq t$, such that:*

$$k \geq \begin{cases} \frac{st}{2} & \text{if } t \text{ is even and } l > \frac{t}{2}, \\ \lfloor \frac{t}{2} \rfloor s + 1 & \text{if } t \text{ is odd and } l > \lfloor \frac{t}{2} \rfloor, \\ sl & \text{if } l \leq \lfloor \frac{t}{2} \rfloor, \end{cases}$$

then there exists an $OCA(t, s, l, v)$.

Proof. This follows from the proof of Theorem 3.1.1. □

The *interaction graph* (Definition 3.1.4) also applies to OCAs. The homomorphisms described in Theorem 3.1.5 and Corollary 3.1.6 still hold, but they generally will not result in an optimal OCA because the number of edges in the excess graph blows up through the construction. For example, if we use the columns of a $CA(6; 2, 4, 2)$, which has twelve edges in its corresponding excess graph, to construct an $OCA(6; 2, 4, 2, 2)$, by mapping the columns of the OCA to the CA, then the resulting OCA will have a total of twenty edges in its excess graph. By other methods, we can construct an $OCA(5; 2, 4, 2, 2)$ which has only ten edges in its excess graph (and one less row). However, OCAs can be constructed from other OCAs following the same construction as in Theorem 3.2.2:

Theorem 4.3.2. *For positive integers s, l and v, if there exists an $OCA(t', s, l', v)$, then there exists an $OCA(sl + t', s, l + l', v)$.*

As in Theorem 3.2.4, we can use the same construction to obtain different sets of parameters. Instead of having strength $sl+t'$, the strength of the OCA will be $l+l'$, and the array will have an index $\lambda > 1$.

Theorem 4.3.3. *For positive integers s, l and v, if there exists an $OCA(t', s, l', v)$, then there exists an $OCA_\lambda(l+l', s, l+l', v)$, where $\lambda = v^{sl+t'-l-l'}$.*

As in Corollary 4.3.4, the following is a direct result of Theorem 4.3.3, taking $l = 1$ and allowing the array to have a positive index λ:

Corollary 4.3.4. *For positive integers s and v, if there exists an $OCA_{\lambda_1}(t, s, l, v)$, then there exists an $OCA_{\lambda_2}(l+1, s, l+1, v)$, where $\lambda_2 = \lambda_1 v^{s+t-l-1}$.*

4.4 (t,m,s)-Covering Nets

We have seen how to construct (t, m, s)-nets from orthogonal arrays and ordered orthogonal arrays. Theorem 2.3.13 describes the equivalence between (t, m, s)-nets and parametric subclasses of ordered orthogonal arrays by stating that a (t, m, s)-net in base b exists if and only if an $OOA_{b^t}(m-t, s, m-t, b)$ exists. But for certain parameters where such an OOA does not exist, we can instead use an OCA to construct an object similar to a (t, m, s)-net which we will introduce as a *(t, m, s)-covering net*.

For example, from Theorem 2.3.13, a (0,2,4)-net in base 2 is equivalent to an $OOA(2, 4, 2, 2)$. From Theorem 3.1.1, we know that in order for an $OOA(2, 4, 2, 2)$ to exist, we need an $OA(2, 4, 2)$, which we know does not exist since there do not exist two MOLS of order 2 [20]. So instead of using an $OOA(2, 4, 2, 2)$ to construct the (0,2,4)-net in base 2, we can use an $OCA(2, 4, 2, 2)$, since an $OCA(2, 4, 2, 2)$ exists, with the most optimal one having 5 rows. We follow the exact same construction as if it were an OOA, and instead of having 4 points in our net, we will have a total of 5 points. Also, instead of having exactly one point in every elementary interval of volume $\frac{1}{4}$, we now have at least one point in every such interval. In this example, in one 4-dimensional unit hypercube, there will be one elementary interval of volume $\frac{1}{4}$, which has two points in it, while the other three elementary intervals will have exactly one point in them. So four out of five points will still be uniformly distributed, and therefore they will still have a low discrepancy.

We now give a formal definition of a (t, m, s)-covering net in base b:

Definition 4.4.1. *For integers $0 \leq t \leq m$, $s \geq 1$ and $b \geq 2$, a **(t,m,s)-covering net in base b** is a point set of at least b^m points in the s-dimensional hypercube $[0, 1)^s$ such that every elementary interval in base b having volume b^{t-m} contains at least b^t points.*

Just as the points of a (t, m, s)-net are shifted by a small positive value, $\epsilon \in \mathbb{R}$, in all s-dimensions, in order for them to lie in the intervals and not on the boundaries of the intervals, the same applies for (t, m, s)-covering nets.

A (t, m, s)-covering net has the same relationship to an OCA as a (t, m, s)-net has to an OOA. We can use the same construction described in section 2.3.2, to construct (t, m, s)-covering nets from OCAs. A (t, m, s)-covering net in base b can be constructed from an OCA by translating each row in

the OCA to a point in the (t, m, s)-covering net. This is done by first placing a decimal point before each part of each row of the OCA, so that each point has s coordinates corresponding to the s parts of the OCA. We then take the decimal expansion in base b for each point, in order to translate them to points in an s-dimensional hypercube, $[0, 1)^s$, which results in the set of points for the corresponding (t, m, s)-covering net. This process can be reversed to construct an OCA from a (t, m, s)-covering net. Therefore the left-justified sets of an OCA are equivalent to the elementary intervals of its corresponding (t, m, s)-covering net, in the same way as in Theorem 2.3.13, and as a result, a (t, m, s)-covering net in base b is equivalent to an $OCA_\lambda(m - t, s, m - t, b)$.

Example 4.4.2. *A (0,2,4)-covering net in base 2 has at least $b^m = 2^2 = 4$ points, each point is 4-dimensional, and there is at least $b^t = 2^0 = 1$ point in every elementary interval of volume $b^{t-m} = 2^{0-2} = \frac{1}{4}$. We will construct a (0,2,4)-covering net in base 2 from an OCA(2,4,2,2):*

$OCA(2, 4, 2, 2)$

1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1
0	1	1	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	0	0	0	1	1	0

We use the same construction described in Section 2.3.2. The points corresponding to the OCA in binary form are: $(.11, .11, .11, .11), (.10, .00, .00, .01), (.01, .10, .00, .00), (.00, .01, .10, .00), (.00, .00, .01, .10)$.

The following table translates these values from binary to decimal:

Binary	*Decimal*
.00	$(0 * 2^{-1}) + (0 * 2^{-2}) = 0$
.01	$(0 * 2^{-1}) + (1 * 2^{-2}) = \frac{1}{4}$
.10	$(1 * 2^{-1}) + (0 * 2^{-2}) = \frac{1}{2}$
.11	$(1 * 2^{-1}) + (1 * 2^{-2}) = \frac{3}{4}$

And substituting the decimal values back into our binary points, gives us the points of our (0,2,4)-covering net in base 2: $\{(\frac{3}{4}, \frac{3}{4}, \frac{3}{4}, \frac{3}{4}), (\frac{2}{4}, 0, 0, \frac{1}{4}), (\frac{1}{4}, \frac{2}{4}, 0, 0), (0, \frac{1}{4}, \frac{2}{4}, 0), (0, 0, \frac{1}{4}, \frac{2}{4})\} + (\epsilon, \epsilon)$.

If an $OOA(t, s, l, v)$ does not exist, then by keeping the values of t, s and l the same, the closest OOA would be an $OOA(t, s, l, v')$, where v' is the smallest integer, larger than v, such that an $OOA(t, s, l, v')$ exists. This would introduce at least $(v')^t - v^t$ more rows. So instead of using an OOA with a larger alphabet size, we can use an $OCA(t, s, l, v)$, which always exists, and will result in a smaller number of extra rows. For example, an $OOA(2, 2, 4, 2)$ does not exist, and by using an $OOA(2, 2, 4, 3)$, which exists, we will have five rows more than the number of rows in an $OOA(2, 2, 4, 2)$ (if it existed). So instead, we can use an $OCA(2, 2, 4, 2)$, where an optimal $OCA(2, 2, 4, 2)$ has five rows, which is only one row more than an $OOA(2, 2, 4, 2)$ would have if it existed.

Example 4.4.3. *We will use a (0,2,4)-covering net in base 6, which is equivalent to an* $OCA(2,4,2,6)$*, to numerically approximate the value of the integral* $\int_0^1\int_0^1\int_0^1\int_0^1 wxyz\,dzdydxdw$*. (An* $OOA(2,4,2,6)$ *does not exist, therefore the corresponding (0,2,4)-net in base 6 does not exist.)*

First, we compute the exact value of the integral:

$$\int_0^1\int_0^1\int_0^1\int_0^1 wxyz\,dzdydxdw \;=\; \int_0^1\int_0^1\int_0^1 \frac{wxy}{2}\,dydxdw = \int_0^1\int_0^1 \frac{wx}{4}\,dxdw = \int_0^1 \frac{w}{8}\,dw = \frac{1}{16}.$$

The most optimal $OCA(2,4,2,6)$ *has 37 rows, we use the following* $OCA(37;2,4,2,6)$ *for our (0,2,4)-covering net in base 6:*

$OOA(37;2,4,2,6)$

0	0	0	0	4	0	0	4
0	1	1	0	5	1	1	5
0	4	2	0	2	2	4	2
0	5	3	0	3	3	5	3
0	2	4	0	0	4	2	0
0	3	5	0	1	5	3	1
1	5	0	1	1	0	5	1
1	4	1	1	0	1	4	0
1	0	2	1	5	2	0	5
1	1	3	1	4	3	1	4
1	3	4	1	2	4	3	2
1	2	5	1	3	5	2	3
2	3	0	2	5	0	3	5
2	2	1	2	4	1	2	4
2	5	2	2	0	2	5	0
2	4	3	2	1	3	4	1
2	0	4	2	3	4	0	3
2	1	5	2	2	5	1	2
3	4	0	3	3	0	4	3
3	5	1	3	2	1	5	2
3	3	2	3	4	2	3	4
3	2	3	3	5	3	2	5
3	1	4	3	1	4	1	1
3	0	5	3	0	5	0	0
4	1	0	4	0	0	1	0
4	3	1	4	3	1	3	3
4	2	2	4	1	2	2	1
4	0	3	4	2	3	0	2
4	4	4	4	4	4	4	4
4	5	5	4	5	5	5	5
5	2	0	5	2	0	2	2
5	0	1	5	1	1	0	1
5	1	2	5	3	2	1	3
5	3	3	5	0	3	3	0
5	5	4	5	5	4	5	5
5	5	5	5	4	5	5	4

The corresponding points in the (0,2,4)-covering net in base 6 are:

$\{(0,0,\frac{24}{36},\frac{4}{36}),(\frac{1}{36},\frac{6}{36},\frac{31}{36},\frac{11}{36}),(\frac{4}{36},\frac{12}{36},\frac{14}{36},\frac{26}{36}),(\frac{5}{36},\frac{18}{36},\frac{21}{36},\frac{33}{36}),(\frac{2}{36},\frac{24}{36},\frac{4}{36},\frac{12}{36}),(\frac{3}{36},\frac{30}{36},\frac{11}{36},\frac{19}{36}),$
$(\frac{11}{36},\frac{1}{36},\frac{6}{36},\frac{31}{36}),(\frac{10}{36},\frac{7}{36},\frac{1}{36},\frac{24}{36}),(\frac{6}{36},\frac{13}{36},\frac{32}{36},\frac{5}{36}),(\frac{7}{36},\frac{19}{36},\frac{27}{36},\frac{10}{36}),(\frac{9}{36},\frac{25}{36},\frac{16}{36},\frac{20}{36}),(\frac{8}{36},\frac{31}{36},\frac{23}{36},\frac{15}{36}),$
$(\frac{15}{36},\frac{2}{36},\frac{30}{36},\frac{23}{36}),(\frac{14}{36},\frac{8}{36},\frac{25}{36},\frac{16}{36}),(\frac{17}{36},\frac{14}{36},\frac{2}{36},\frac{30}{36}),(\frac{16}{36},\frac{20}{36},\frac{9}{36},\frac{25}{36}),(\frac{12}{36},\frac{26}{36},\frac{22}{36},\frac{3}{36}),(\frac{13}{36},\frac{32}{36},\frac{17}{36},\frac{8}{36}),$

$(\frac{22}{36}, \frac{3}{36}, \frac{18}{36}, \frac{27}{36}), (\frac{23}{36}, \frac{9}{36}, \frac{13}{36}, \frac{32}{36}), (\frac{21}{36}, \frac{15}{36}, \frac{26}{36}, \frac{22}{36}), (\frac{20}{36}, \frac{21}{36}, \frac{33}{36}, \frac{17}{36}), (\frac{19}{36}, \frac{27}{36}, \frac{10}{36}, \frac{7}{36}), (\frac{18}{36}, \frac{33}{36}, \frac{5}{36}, 0),$
$(\frac{25}{36}, \frac{4}{36}, 0, \frac{6}{36}), (\frac{27}{36}, \frac{10}{36}, \frac{19}{36}, \frac{21}{36}), (\frac{26}{36}, \frac{16}{36}, \frac{8}{36}, \frac{13}{36}), (\frac{24}{36}, \frac{22}{36}, \frac{15}{36}, \frac{2}{36}), (\frac{28}{36}, \frac{28}{36}, \frac{28}{36}, \frac{28}{36}), (\frac{29}{36}, \frac{34}{36}, \frac{35}{36}, \frac{35}{36}),$
$(\frac{32}{36}, \frac{5}{36}, \frac{12}{36}, \frac{14}{36}), (\frac{30}{36}, \frac{11}{36}, \frac{7}{36}, \frac{1}{36}), (\frac{31}{36}, \frac{17}{36}, \frac{20}{36}, \frac{9}{36}), (\frac{33}{36}, \frac{23}{36}, \frac{3}{36}, \frac{18}{36}), (\frac{35}{36}, \frac{29}{36}, \frac{34}{36}, \frac{35}{36}), (\frac{35}{36}, \frac{35}{36}, \frac{29}{36}, \frac{34}{36}),$
$(\frac{34}{36}, \frac{35}{36}, \frac{35}{36}, \frac{28}{36})\} + (\epsilon, \epsilon)$.

Using the above (0,2,4)-covering net in base 6, to approximate the value of the integral using QMC integration, gives us an approximation of 0.1102.

In the next example, we compare the values obtained from approximating an integral using a (t, m, s)-net constructed from an OOA versus (t, m, s)-covering nets constructed from OCAs, where the OOAs and OCAs have the same parameter values for t, s and l.

Example 4.4.4. *We will numerically approximate the value of the integral* $\int_0^\pi (\int_0^1 ysin(200x)dy)dx$, *using a (0,2,2)-net in base 4 and using a (0,2,2)-covering net in base 4.*

The exact value of the integral is computed as follows:

$$\begin{aligned}\int_0^\pi (\int_0^1 ysin(200x)dy)dx &= \int_0^\pi \frac{y^2}{2}sin(200x)|_0^1 dx = \int_0^\pi \frac{1}{2}sin(200x)dx = -\frac{1}{400}cos(200x)|_0^\pi \\ &= -\frac{1}{400} + \frac{1}{400} = 0.\end{aligned}$$

For our (0,2,2)-net in base 4, we use the following $OOA(16; 2, 2, 2, 4)$*:*

$OOA(16; 2, 2, 2, 4)$

0	0	0	0
0	1	1	0
0	2	2	0
0	3	3	0
1	0	0	1
1	1	1	1
1	2	2	1
1	3	3	1
2	0	0	2
2	1	1	2
2	2	2	2
2	3	3	2
3	0	0	3
3	1	1	3
3	2	2	3
3	3	3	3

Using the corresponding (0,2,2)-net in base 4 to approximate the value of the integral using QMC integration gives us an approximation of -0.3928.

For our (0,2,2)-covering net in base 4, we use an OCA(2,2,2,4). The most optimal OCA is just an $OOA(16; 2, 2, 2, 4)$*, therefore we use an* $OCA(17; 2, 2, 2, 4)$ *to construct the (0,2,2)-covering net in base 4. Such an OCA has one repeated row, and depending on which row is repeated, the value of the*

approximation will be different. The table below shows the value of the approximation obtained when using the $OOA(16; 2, 2, 2, 4)$ *above, along with the extra row specified.*

Extra Row	*Integral Approximation using Extra Row*
00|00	*-0.3695*
01|10	*-0.3233*
02|20	*-0.3695*
03|30	*-0.5081*
10|01	*-0.3695*
11|11	*-0.3117*
12|21	*-0.3695*
13|31	*-0.5196*
20|02	*-0.3695*
21|12	*-0.3002*
22|22	*-0.3695*
23|23	*-0.5312*
30|03	*-0.3695*
31|13	*-0.2886*
32|23	*-0.3694*
33|33	*-0.5427*

The numerical approximations obtained using an $OCA(17; 2, 2, 2, 4)$ *to construct a (0,2,2)-covering net in base 4, range from -0.2886 to -0.5427, whereas the approximation obtained using an* $OOA(16; 2, 2, 2, 4)$ *is -0.3928. Taking an average of all sixteen different approximations obtained in the table above gives us -0.3926.*

In Corollary 4.3.4, we are constructing one OCA from another one. This can now be interpreted in terms of (t, m, s)-covering nets, since we have an equivalence between (t, m, s)-covering nets and OCAs.

Corollary 4.4.5. *If there exists a* (t, m, s)*-covering net in base* b*, then there exists a* $(t+s-1, m+s, s)$*-covering net in base* b*.*

Proof. A (t_1, m_1, s)-covering net base b is equivalent to an $OCA_{b^{t_1}}(m_1 - t_1, s, m_1 - t_1, b)$. By Theorem 4.3.4, if there exists an $OCA_{b^{t_1}}(m_1-t_1, s, m_1-t_1, b)$, then there exists an $OCA_\lambda(m_1-t_1+1, s, m_1-t_1+1, b)$, where $\lambda = b^{t_1+s-1}$. By translating the $OCA_\lambda(m_1-t_1+1, s, m_1-t_1+1, b)$ back to a (t, m, s)-covering net, gives us a $(t_1 + s - 1, m_1 + s, s)$-covering net base b. □

By relaxing the condition for the number of points in every elementary interval of a (t, m, s)-covering net, from *exactly* to *at least* such a number of points, allows us to introduce a mixed alphabet size for the base of the net. Instead of constructing such a net using an OCA, we can instead use a MOCA, as we will see in Section 5.2.

Chapter 5

Applications of OOAs and OCAs to Numerical Methods

This chapter, outlining the applications of OOAs and OCAs to numerical integration, is not the main focus of our thesis, but we would like to demonstrate how these objects may be useful in evaluating certain multi-dimensional integrals. We have seen how OOAs and OCAs can be used to construct (t, m, s)-nets and (t, m, s)-covering nets respectively, which are useful as low discrepancy points for quasi-Monte Carlo integration. In this chapter, we consider further useful applications of these combinatorial objects.

5.1 Distinct (t,m,s)-Nets from a Single OOA

In Monte Carlo integration, randomly distributed points are used to numerically approximate the value of an integral. In standard MC integration, one common technique is to evaluate the integral at N pseudorandom points, repeat this k times, and then take an average of the k approximations, resulting in kN function evaluations in total. While quasi-random sequences generally have much better convergence than pseudorandom sequences, they do not have the flexibility of repetition. In this section, we will describe a method of generating multiple distinct (t, m, s)-nets from an OOA, in order to have a larger set of points to numerically approximate the value of the integral. Taking an average of a larger number of function evaluations can improve the approximation of the integral.

We have shown an equivalence between (t, m, s)-nets in base b and parametric subclasses of OOAs. But following the construction described in Section 2.3.2, a single OOA will always give us the same set of points for our (t, m, s)-net. A new set of points would require a distinct OOA, which translates to a distinct corresponding (t, m, s)-net. By using two different sets of quasi-random points with the same parameter values for t, m, s and b, we may get different values when approximating an integral depending on the value of these points and the integral we are approximating. Therefore, being able to generate different sets of random points from the same OOA using the same method of construction would be useful in numerically approximating the value of an integral. One way of achieving this is by permuting

the values of the alphabet of the OOA to form another OOA which is isomorphic to the original one. This also applies to OCAs.

Example 5.1.1. *We will use an* $OOA(2,2,2,4)$ *to construct (0,2,2)-nets in base 4 to numerically approximate the value of the following integral:* $\int_0^\pi (\int_0^1 ysin(200x)dy)dx$.

First, we will compute the exact value of the integral, so we know what to compare our result to:

$$\begin{aligned}\int_0^\pi (\int_0^1 ysin(200x)dy)dx &= \int_0^\pi \frac{y^2}{2}sin(200x)|_0^1 dx = \int_0^\pi \frac{1}{2}sin(200x)dx = -\frac{1}{400}cos(200x)|_0^\pi \\ &= -\frac{1}{400}+\frac{1}{400} = 0.\end{aligned}$$

We start with an $OOA(2,2,2,4)$*, call it A:*

		A	
0	0	0	0
0	1	1	0
0	2	2	0
0	3	3	0
1	0	0	1
1	1	1	1
1	2	2	1
1	3	3	1
2	0	0	2
2	1	1	2
2	2	2	2
2	3	3	2
3	0	0	3
3	1	1	3
3	2	2	3
3	3	3	3

Constructing a (0,2,2)-net in base 4 from the above OOA gives us the following set of 16 points, using the construction described in Section 2.3.2:

$$\{(0,0),(\tfrac{1}{16},\tfrac{4}{16}),(\tfrac{2}{16},\tfrac{8}{16}),(\tfrac{3}{16},\tfrac{12}{16}),(\tfrac{4}{16},\tfrac{1}{16}),(\tfrac{5}{16},\tfrac{5}{16}),(\tfrac{6}{16},\tfrac{9}{16}),(\tfrac{7}{16},\tfrac{13}{16}),$$
$$(\tfrac{8}{16},\tfrac{2}{16}),(\tfrac{9}{16},\tfrac{6}{16}),(\tfrac{10}{16},\tfrac{10}{16}),(\tfrac{11}{16},\tfrac{14}{16}),(\tfrac{12}{16},\tfrac{3}{16}),(\tfrac{13}{16},\tfrac{7}{16}),(\tfrac{14}{16},\tfrac{11}{16}),(\tfrac{15}{16},\tfrac{15}{16})\}$$

Using these points to approximate the value of the integral using QMC integration gives us an approximation of -0.3928.

Now, we can perform the permutation $\sigma_1 = (0123)$ *for each value in the first part of the original OOA, and the permutation* $\sigma_2 = (02)(13)$ *for each value in the second part of the original OOA. So our new* $OOA(2,2,2,4)$*, call it B, which is isomorphic to the original is:*

B

1	1	2	2
1	2	3	2
1	3	0	2
1	0	1	2
2	1	2	3
2	2	3	3
2	3	0	3
2	0	1	3
3	1	2	0
3	2	3	0
3	3	0	0
3	0	1	0
0	1	2	1
0	2	3	1
0	3	0	1
0	0	1	1

Constructing a (0,2,2)-net in base 4 from the above OOA gives us the following set of 16 points:

$$\{(\tfrac{5}{16},\tfrac{10}{16}),(\tfrac{6}{16},\tfrac{14}{16}),(\tfrac{7}{16},\tfrac{2}{16}),(\tfrac{4}{16},\tfrac{6}{16}),(\tfrac{9}{16},\tfrac{11}{16}),(\tfrac{10}{16},\tfrac{15}{16}),(\tfrac{11}{16},\tfrac{3}{16}),(\tfrac{8}{16},\tfrac{7}{16}),$$
$$(\tfrac{13}{16},\tfrac{8}{16}),(\tfrac{14}{16},\tfrac{12}{16}),(\tfrac{15}{16},0),(\tfrac{12}{16},\tfrac{4}{16}),(\tfrac{1}{16},\tfrac{9}{16}),(\tfrac{2}{16},\tfrac{13}{16}),(\tfrac{3}{16},\tfrac{1}{16}),(0,\tfrac{5}{16})\}$$

Using these points to approximate the value of the integral using QMC integration gives us an approximation of 0.3928.

For this example, if we take the average of both these approximations, we get: $\frac{0.3928-0.3928}{2} = 0$*, which is obviously much closer to the exact value which we calculated to be 0.*

We tried other permutations for the original $OOA(2,2,2,4)$, A, given above. The following table summarizes the permutations for each part of the OOA, σ_1 and σ_2, based on the original OOA, A, and the corresponding values for the numerical approximations obtained for $\int_0^{\pi}(\int_0^1 ysin(200x)dy)dx$ using the corresponding (0,2,2)-nets base 4:

σ_1	σ_2	Numerical Approximation
(021)	(01)(23)	0.1964
(0)(1)(2)(3)	(0123)	0.3928
(03)	(032)	-0.3928

Therefore different permutations of the alphabet of an OOA will result in different OOAs and therefore distinct corresponding (t,m,s)-net point sets which in turn may give us different approximations for the integral.

5.2 Integrals that Fluctuate in Certain Dimensions

An integral might fluctuate more in a certain dimension depending on the function itself, or because it has a larger interval to cover in that dimension. When solving such integrals using quasi-Monte Carlo integration, it would be advisable to use more points for a better numerical approximation. But instead of increasing the total number of points in each dimension, we can instead increase the number of points only for specific dimensions that are required. This would require a smaller number of points than that required for increasing the number of points in each dimension.

In order to construct a (t, m, s)-net with a different number of points in different dimensions, while keeping the number of points in each elementary interval the same, we require a different number of elementary intervals in different dimensions. In terms of the OCA, this translates to a different number of u-tuples within different left-justified sets. Without changing the value of u, we can increase the number of u-tuples within a left-justified set by increasing the alphabet for at least one of the u columns. This corresponds to a mixed ordered covering array (Definition 4.1.3). In Section 4.4, we saw how OCAs were used to construct (t, m, s)-covering nets. Since MOCAs are just OCAs with mixed alphabet size, we will follow the same construction, to create a corresponding (t, m, s)-covering net in a mixed base.

Example 5.2.1. *We will construct a (t,m,s)-covering net from a* $MOCA(6; 2, 2, 2, (2, 2, 3, 2))$.

$MOCA(2, 2, 2, (2, 2, 3, 2))$

0	0	0	0
0	1	1	0
0	$*$	2	0
1	0	0	1
1	1	1	1
1	$*$	2	1

The alphabet size of the OCA relates to the base of the (t, m, s)*-covering net. Since this is a MOCA, the alphabet size of the array is not constant, therefore we end up with a (0,2,2)-covering net in a mixed base. The first part of the MOCA has alphabet size two, and four different pairs, so we will translate it to points in the (0,2,2)-net using a binary to decimal conversion:*

Binary	*Decimal*
.00	$(0 * 2^{-1}) + (0 * 2^{-2}) = 0$
.01	$(0 * 2^{-1}) + (1 * 2^{-2}) = \frac{1}{4}$
.10	$(1 * 2^{-1}) + (0 * 2^{-2}) = \frac{1}{2}$
.11	$(1 * 2^{-1}) + (1 * 2^{-2}) = \frac{3}{4}$

For the second part of the MOCA which has mixed alphabet size, we will translate it to 6 distinct points in the (0,2,2)-net using a mixed base to decimal conversion. For the conversion, we will use base 3 for the first digit after the decimal point (since the alphabet size for the corresponding column is 3) and

base $3 \times 2 = 6$ for the second digit after the decimal point (since the alphabet size for the corresponding two columns are 3 and 2):

Mixed Base	*Decimal*
.00	$(0 * 3^{-1}) + (0 * 6^{-1}) = 0$
.01	$(0 * 3^{-1}) + (1 * 6^{-1}) = \frac{1}{6}$
.10	$(1 * 3^{-1}) + (0 * 6^{-1}) = \frac{2}{6}$
.11	$(1 * 3^{-1}) + (1 * 6^{-1}) = \frac{3}{6}$
.20	$(2 * 3^{-1}) + (0 * 6^{-1}) = \frac{4}{6}$
.21	$(2 * 3^{-1}) + (1 * 6^{-1}) = \frac{5}{6}$

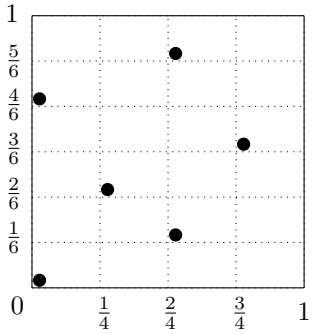
In mixed base form, our six points are:

$$(.00, .00), (.01, .10), (.00, .20), (.10, .01), (.11, .11), (.10, .21).$$

Translating these points to decimal using the corresponding conversions from the tables above for the first and second coordinates of each point, gives us the following points for our covering net:

$$\{(0,0), (\tfrac{1}{4}, \tfrac{2}{6}), (0, \tfrac{4}{6}), (\tfrac{2}{4}, \tfrac{1}{6}), (\tfrac{3}{4}, \tfrac{3}{6}), (\tfrac{2}{4}, \tfrac{5}{6})\} + (\epsilon, \epsilon)$$

In our first dimension, we divided our area into quarters. We have a total of four points with distinct x-coordinates, with at least one point per elementary interval of volume $\frac{1}{4}$. In our second dimension, we divided our area into sixths. We have a total of six points with distinct y-coordinates, with at least one point per elementary interval of volume $\frac{1}{6}$.

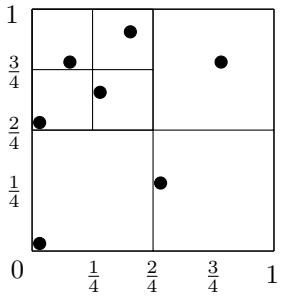


5.3 Integrals that Fluctuate in Specific Intervals

If we know that an integral fluctuates within a certain region for all dimensions, we can increase the number of points required for that specific region only, while keeping the number of points in other regions the same. In terms of (t, m, s)-nets and elementary intervals, this can be achieved by placing a (t, m, s)-net within an elementary interval of the (t, m, s)-net (in which the function fluctuates).

For example, placing a (0,2,2)-net base 2 into the elementary interval $[0, \frac{1}{2}) \times [\frac{1}{2}, 1)$ of a (0,2,2)-net base 2 yields the following points in a $[0, 1) \times [0, 1)$ square:

$$\{(0,0),(0,\tfrac{2}{4}),(\tfrac{1}{8},\tfrac{6}{8}),(\tfrac{2}{8},\tfrac{5}{8}),(\tfrac{3}{8},\tfrac{7}{8}),(\tfrac{2}{4},\tfrac{1}{4}),(\tfrac{3}{4},\tfrac{3}{4})\}+(\epsilon,\epsilon)$$



Combinatorially, this can be constructed in the following way:

Example 5.3.1. *We will construct the above 7 points by placing a (0,2,2)-net base 2 in the elementary interval* $[0,\frac{1}{2})\times[\frac{1}{2},1)$ *of a (0,2,2)-net base 2:*

A (0,2,2)-net base 2 is equivalent to an $OOA(2,2,2,2)$*. We will use the following* $OOA(2,2,2,2)$ *for our construction:*

0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1

Using the (t,m,s)-net construction discussed earlier, we place a decimal point before each value of each part of the OOA, which yields the following corresponding points:

$$(.00,.00)$$
$$(.01,.10)$$
$$(.10,.01)$$
$$(.11,.11)$$

We translate these decimal points to points in a (0,2,2)-net base 2 using the binary to decimal conversion:

Binary	*Decimal*
.00	$(0*2^{-1})+(0*2^{-2})=0$
.01	$(0*2^{-1})+(1*2^{-2})=\frac{1}{4}$
.10	$(1*2^{-1})+(0*2^{-2})=\frac{1}{2}$
.11	$(1*2^{-1})+(1*2^{-2})=\frac{3}{4}$

Using this translation, we can see that the second point, $(.01,.10)$*, which translates to* $(\frac{1}{4},\frac{1}{2})$*, lies in the elementary interval* $[0,\frac{1}{2})\times[\frac{1}{2},1)$*, to which we will be applying another (0,2,2)-net base 2. In order to do this, we first scale by multiplying each value of the four binary points by 0.1, in order to get the following points:* $(.000,.000),(.001,.010),(.010,.001),(.011,.011)$*.*

We now add $(0, .1)$ *to each of the four points to move all the points into the top left quarter of the square. This gives us the following four points:* $(.000, .100), (.001, .110), (.010, .101), (.011, .111)$.

We now translate these points from binary to decimal to get four points that lie in the elementary interval $[0, \frac{1}{2}) \times [\frac{1}{2}, 1)$*:*

Binary	*Decimal*
.000	$(0 * 2^{-1}) + (0 * 2^{-2}) + (0 * 2^{-3}) = 0$
.001	$(0 * 2^{-1}) + (0 * 2^{-2}) + (1 * 2^{-3}) = \frac{1}{8}$
.010	$(0 * 2^{-1}) + (1 * 2^{-2}) + (0 * 2^{-3}) = \frac{2}{8}$
.011	$(0 * 2^{-1}) + (1 * 2^{-2}) + (1 * 2^{-3}) = \frac{3}{8}$
.100	$(1 * 2^{-1}) + (0 * 2^{-2}) + (0 * 2^{-3}) = \frac{4}{8}$
.101	$(1 * 2^{-1}) + (0 * 2^{-2}) + (1 * 2^{-3}) = \frac{5}{8}$
.110	$(1 * 2^{-1}) + (1 * 2^{-2}) + (0 * 2^{-3}) = \frac{6}{8}$
.111	$(1 * 2^{-1}) + (1 * 2^{-2}) + (1 * 2^{-3}) = \frac{7}{8}$

Using the above translation, the four points in the elementary interval $[0, \frac{1}{2}) \times [\frac{1}{2}, 1)$ *are:*

$$\{(0, \tfrac{2}{4}), (\tfrac{1}{8}, \tfrac{6}{8}), (\tfrac{2}{8}, \tfrac{5}{8}), (\tfrac{3}{8}, \tfrac{7}{8})\} + (\epsilon, \epsilon)$$

Combining these four points with the other three points gives us all seven points in the $[0, 1) \times [0, 1)$ *square:*

$$\{(0, 0), (0, \tfrac{2}{4}), (\tfrac{1}{8}, \tfrac{6}{8}), (\tfrac{2}{8}, \tfrac{5}{8}), (\tfrac{3}{8}, \tfrac{7}{8}), (\tfrac{2}{4}, \tfrac{1}{4}), (\tfrac{3}{4}, \tfrac{3}{4})\} + (\epsilon, \epsilon).$$

Chapter 6

Conclusion

In this thesis, we focused on two combinatorial objects, ordered orthogonal arrays and a new object that we introduced called ordered covering arrays. We now summarize all the main contributions of this thesis and discuss some future work in this area.

We considered new constructions for ordered orthogonal arrays, using simple recursive and homomorphic constructions. We introduced a new graph called the interaction graph (see Definition 3.1.4), which models the interaction patterns between the columns of an OOA. We used this graph to define a graph homomorphism mapping the columns of an ordered orthogonal array to those of an orthogonal array. We then developed some theorems and corollaries for constructions of OOAs using recursive methods. The first main theorem involves using an OA and an OOA to construct an OOA with strength equal to the sum of the strengths of the first two arrays (see Theorem 3.2.2). The second main theorem involves using an OA and an OOA to construct an OOA with strength equal to the sum of the columns of the first two arrays (see Theorem 3.2.4). A corollary from the latter construction for OOAs (Theorem 3.2.2) is also translated in terms of (t, m, s)-nets, and therefore a recursive method is developed for creating (t, m, s)-nets.

The new theorems developed in this thesis for the construction of ordered orthogonal arrays can be studied further. In these theorems, we have kept the alphabet size for both the original and final OOA the same. One idea is to extend these theorems in order to construct an OOA with a different alphabet size than the original one.

For future work in the area of (t, m, s)-nets, it would be useful to construct a specific family of (t, m, s)-nets with certain parameters, such as $(0, 2, s)$-nets in base b, using a similar recursive method developed for ordered orthogonal arrays with certain parameters.

In Section 5.1, we discussed how distinct (t, m, s)-nets can be constructed from a single OOA using permutations of the alphabet of the ordered orthogonal array. In Section 5.3, we discussed how a (t, m, s)-net can be placed within an elementary interval of that (t, m, s)-net, to yield more points in a specific area of the s-dimensional hypercube, suitable for multi-dimensional integrals that fluctuate more within a specific interval. Therefore, we can combine these two concepts to create distinct (t, m, s)-nets that are more concentrated in a specific interval.

Ordered covering arrays, introduced in Chapter 4, have the same relationship to ordered orthogonal arrays as covering arrays have to orthogonal arrays. They relax the *exact* coverage condition just as covering arrays do. In testing areas such as networks using interaction test suites, there may exist factors that will never interact with each other. By modelling the test suite using ordered covering arrays, we avoid certain interactions of the factors while allowing other interactions between factors. Constructions for ordered covering arrays were studied in Sections 4.2 and 4.3. Covering array constructions such as the block recursive method and Roux-type construction were adapted for the construction of ordered covering arrays. The recursive and multiplicative methods developed for the construction of ordered orthogonal arrays are also applicable to ordered covering arrays, as studied in Section 4.3. The concept of ordered covering arrays is also applicable in constructing point sets similar to (t, m, s)-nets. As discussed in Section 2.3, there exists an equivalence between (t, m, s)-nets and a parametric subclass of ordered orthogonal arrays. But as we know, ordered orthogonal arrays do not exist for all parameter values. Therefore, by relaxing the *exact* coverage condition required for ordered orthogonal arrays, allows such arrays to exist for all parameters. We can use these ordered covering arrays to construct a new object called a (t, m, s)-covering net, which was introduced in Section 4.4. This object is similar to a (t, m, s)-net, except that in a (t, m, s)-covering net, there is *at least* b^t points in every elementary interval of volume b^{m-t}, while for a (t, m, s)-net, there is *exactly* b^t such points.

We conclude this chapter with suggestions for future work within the area of ordered covering arrays. Since this is a new concept introduced in this thesis, there are many areas to explore and expand on with regards to ordered covering arrays.

One such area is algorithms used to generate such arrays for any given parameter. As discussed in Section 2.2.3, many algorithms have been developed for covering arrays, with different algorithms targeting different objectives. Some of these algorithms may be adapted in order to construct ordered covering arrays. In Section 4.4, we discussed how ordered covering arrays can be used to construct (t, m, s)-covering nets. Therefore, these algorithms can be beneficial in creating large (t, m, s)-covering nets for parameter values where their corresponding ordered orthogonal arrays do not exist.

In Section 5.2, we briefly discussed the application of having a mixed base for (t, m, s)-covering nets, in order to vary the number of intervals in different dimensions, suitable for approximating the value of multi-dimensional integrals which fluctuate more in a specific dimension. We just touched on the concept of mixed bases for covering nets, such an object and its applications is another area that can be studied further.

Another area for future work is upper bounds on the number of rows for ordered covering arrays. Bounds for covering arrays have been studied extensively [16, 22]. It would be both interesting and useful to see how these relate to ordered covering arrays.

And finally bounds for (t, m, s)-nets and ordered orthogonal arrays have also been studied [26, 31]. Therefore studying the bounds on the discrepancy of (t, m, s)-covering nets constructed using ordered covering arrays is essential, in order to conclude that the set of points have a low discrepancy. In this thesis, we just assume this to be true, but it is an important step in order to further discuss and formalize applications of (t, m, s)-covering nets.

References

[1] R. J. Abel, C. J. Colbourn and J. H. Dinitz, *Mutually orthogonal Latin squares (MOLS), The CRC handbook of combinatorial designs, 2nd ed.*, C. J. Colbourn and J. H. Dinitz, 160-171, CRC Press, 2007.

[2] M. J. Adams, *Generalized orthogonal arrays: Construction and related graphs*, J. Combin. Des. **7** (1999), 31-39.

[3] J. Bierbrauer, Y. Edel and W. Ch. Schmid, *Coding-theoretic constructions for (t,m,s)-nets and ordered orthogonal arrays*, J. Combin. Des. **10** (2002), 403-418.

[4] J. R. Birge, *Quasi-Monte Carlo approaches to option pricing*, Tech. Report 94-19, Dept. of Ind. and Oper. Eng., Uni. of Michigan, 1995.

[5] R. C. Bose, *On the application of the properties of Galois fields to the problem of constructions of hyper-Graeco-Latin squares*, Sankhya: The Indian Journal of Statistics **3** (1938), 323-338.

[6] R. C. Bose, S. S. Shrikhande, and E. T. Parker, *Further results on the construction of mutually orthogonal latin squares and the falsity of Euler's conjecture*, Canad. J. Math. **12** (1960), 189-203.

[7] R. C. Bryce, C. J. Colbourn and M. B. Cohen, *A framework of greedy methods for constructing interaction tests*, In Proc. of the 27th Int. Conference on Software Eng. (2005), 146-155, IEEE, Los Alamitos, CA.

[8] R. C. Bryce and C. J. Colbourn, *Prioritized interaction testing for pairwise coverage with seeding and avoids*, Info. and Software Tech. J. **48**, (2006), no. 10, 960-970.

[9] R. C. Bryce and C. J. Colbourn, *The density algorithm for pairwise interaction testing: Research articles*, Software Test. Verif. Reliab. **17**, (2007), no. 3, 159-182.

[10] J. N. Cawse (ed.), *Experimental design for combinatorial and high throughput materials development*, John Wiley & Sons, New York, 2003.

[11] M. A. Chateauneuf and D. L. Kreher, *On the state of strength-three covering arrays*, J. Combin. Des. **10** (2002), 217-238.

[12] D. M. Cohen, S. R. Dalal and G. C. Patton, *The AETG system: An approach to testing based on combinatorial design*, IEEE Transaction on Software Engineering **23** (1997), no. 7, 437-444.

[13] M. B. Cohen and C. J. Colbourn, *A deterministic density algorithm for pairwise interaction coverage*, In Proc. of the IASTED Intl. Conference on Software Engineering (2004), 242-252.

[14] M. B. Cohen, C. J. Colbourn, P. B. Gibbons and W. B. Mugridge, *Constructing test suites for interaction testing*, In Proc. of the Intl. Conf. on Software Engineering (2003), 38-48.

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

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

[17] C. J. Colbourn and J. H. Dinitz, *The CRC handbook of combinatorial designs, 2nd ed.*, CRC Press, 2007.

[18] P. Danziger, N. Francetić and E. Mendelsohn, *Covering arrays with row limit w=4*, Submitted to the J. Combin. Des., 2011.

[19] P. Danziger, E. Mendelsohn, L. Moura and B. Stevens, *Covering arrays avoiding forbidden edges*, Theoretical Comp. Sci. **410** (2009), 5403-5414.

[20] L. Euler, *Recherches sur une nouvelle espèce de quarrés magiques.* Opera Omnia, Ser. I, Vol 7, 291-392, Verhandelingen utigegeven door het zeeuwsch Genootschap det Wetenschappen te Vlissingen **9** (1782), 85-239.

[21] A. Hartman and L. Raskin, *Problems and algorithms for covering arrays*, Discrete Math. **284**, no. 1-3 (2004), 149-156.3

[22] A. Hartman, *Software and hardware testing using combinatorial covering suites, in: graph theory, combinatorics and algorithms: interdisciplinary applications*, Springer, Berlin (2005), 237-266.

[23] A. S. Hedayat, N. J. A. Sloane and J. Stufken, *Orthogonal arrays: theory and applications*, Springer, Berlin (1999)

[24] K. M. Lawrence, *A combinatorial characterization of (t,m,s)-nets in base b*, J. Combin. Des. **4** (1996), 275-293.

[25] C. F. Laywine and G. L. Mullen, *Discrete mathematics using Latin squares*, Wiley-Interscience, New York, 1998.

[26] W. J. Martin, *Linear programming bounds for ordered orthogonal arrays and (t,m,s)-nets, Monte Carlo and quasi-Monte Carlo methods 1998*, Springer, Berlin (2000), 368-376.

[27] W. J. Martin, *(t,m,s)-Nets, The CRC handbook of combinatorial designs, 2nd ed.*, C. J. Colbourn and J. H. Dinitz, 639-643, CRC Press, 2007.

[28] W. J. Martin and D. R. Stinson, *Association schemes for ordered orthogonal arrays and (t,m,s)-nets*, Canad. J. Math. **51** (1999), 326-346.

[29] L. Moura, J. Stardom, B. Stevens and A. Williams, *Covering arrays with mixed alphabet sizes*, J. Combin. Des. **11** (2003), 413-432.

[30] G. L. Mullen and W. Ch. Schmid, *An equivalence between (t,m,s)-nets and strongly orthogonal hypercubes*, J. Combin. Theory A **76** (1996), 164-174.

[31] H. Niederreiter, *Point sets and sequences with small discrepancy*, Monatsh. Math. **104** (1987), 273-337.

[32] H. Niederreiter, *Random number generation and quasi-Monte Carlo methods*, CBMS-NSF Series in Applied Math. **63**, SIAM, Philadelphia, 1992.

[33] R. L. Probert and A. W. Williams, *A practical strategy for testing pair-wise coverage of network interfaces*, Proc. Seventh Intl. Symp. on Software Reliability Engineering (1996), 246-254.

[34] R. L. Probert and A. W. Williams, *A measure for component interaction test coverage*, Proc. ACS/IEEE Intl. Conf. on Computer Systems and Applications (2001), 301-311.

[35] G. Roux, *k-Propriétés dans les tableaux de n colonnes: cas particulier de la k-surjectivité et de la k-permutivité*, Ph.D. Thesis, Université de Paris, 1987.

[36] G. Serourri and N. H. Bshouty, *Vector sets for exhaustive testing of logic circuits*, IEEE Trans. on Infor. Theory **34** (1988), 513-522.

[37] D. E. Shasha, A. Y. Kouranov, L. V. Lejay, M. F. Chou and G. M. Coruzzi, *Using combinatorial design to study regulation by multiple input signals: a tool for parsimony in the post-genomics era*, Plant Physiol. **127** (2001), no. 4, 1590-1594.

[38] N. J. A. Sloane, *Covering arrays and intersecting codes*, J. Combin. Des. **1** (1993), 51-63.

[39] G. Tarry, *Le problème des 36 officiers*, C. R. Assoc. Fran. Av. Sci. **1** (1900), 122-123.