Listing Combinatorial Objects

by

Victoria E. Horan

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

Approved September 2012 by the
Graduate Supervisory Committee:

G. Hurlbert, Chair
A. Czygrinow
S. Fishel
C.J. Colbourn
A. Sen

ARIZONA STATE UNIVERSITY

December 2012

ABSTRACT

Gray codes are perhaps the best known structures for listing sequences of combinatorial objects, such as binary strings. Simply defined as a minimal change listing, Gray codes vary greatly both in structure and in the types of objects that they list. More specific types of Gray codes are universal cycles and overlap sequences. Universal cycles are Gray codes on a set of strings of length n in which the first $n-1$ letters of one object are the same as the last $n-1$ letters of its predecessor in the listing. Overlap sequences allow this overlap to vary between 1 and $n-1$.

Some of our main contributions to the areas of Gray codes and universal cycles include a new Gray code algorithm for fixed weight m-ary words, and results on the existence of universal cycles for weak orders on $[n]$. Overlap cycles are a relatively new structure with very few published results. We prove the existence of s-overlap cycles for k-permutations of $[n]$, which has been an open research problem for several years, as well as constructing 1-overlap cycles for Steiner triple and quadruple systems of every order. Also included are various other results of a similar nature covering other structures such as binary strings, m-ary strings, subsets, permutations, weak orders, partitions, and designs.

These listing structures lend themselves readily to some classes of combinatorial objects, such as binary n-tuples and m-ary n-tuples. Others require more work to find an appropriate structure, such as k-subsets of an n-set, weak orders, and designs. Still more require a modification in the representation of the objects to fit these structures, such as partitions. Determining when and how we can fit these sets of objects into our three listing structures is the focus of this dissertation.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Dr. Glenn Hurlbert, for encouraging me to consider graduate school and for not only advising me on writing my dissertation but also on being a well-rounded mathematician. I will miss our weekly meetings and the many, many jokes about "fixed weight Mary".

Next I would like to thank my committee members for taking the time to review this document. In addition, they also deserve thanks individually for the following: Dr. Andrzej Czygrinow for his extra help in preparing for qualifying and comprehensive exams; Dr. Charles Colbourn for teaching me about combinatorial designs and answering my many emails; Dr. Arunabha Sen for welcoming me into his research group and helping me to think about applications for my research; and last but not least Dr. Susanna Fishel for teaching me enumerative combinatorics as well as for always being interested in whatever work I was currently involved in.

For guiding me through the rules of "the real world", I must recognize Dr. Warren Debany. Also important to my success are the many individuals at the School of Mathematical and Statistical Sciences at ASU, AFRL Rome, and the SMART Scholarship Program.

Finally, none of this would have been possible without the help of my family. I owe many thanks to my parents, John and Jeanette, and my sister, Jacqueline, for always encouraging me to work hard but enjoy life at the same time. You always knew I'd be a mathematician one day, before I even realized it myself!

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Consider the following cyclic string.

$$S = 0000111101100101$$

Notice that every binary 4-tuple $\{a, b, c, d\}$ appears equally often (in fact, exactly once) as a contiguous substring of S. It follows that every n-tuple with $n < 4$ appears equally often in S as well. Since these are some of the properties one might expect of a randomly generated binary string, one could use S as an approximation to a random string; that is, S is a pseudo-random generator. Such generators as these have been known for all values of n (as well as for k-ary in place of binary) since the work of de Bruijn [12] and, independently good, in the 1940s, who used them in number theoretic and cryptographic applications. These de Bruijn cycles, as they have come to be known, originated farther back, however, as discovered (and enumerated) by Flye-St Marie in the 1890s [39]. Because of the tremendous need for pseudo-random generators in math and science, de Bruijn cycles have held an important position in a wide range of applications. they continue to be objects of great study in mathematics for a number of properties useful for other endeavors. For the last two decades, structures other than k-ary n-tuples have been investigated similarly. Our thesis here pushes the envelope of structures used and of types of cycles allowed, in hopes that new applications to mathematics, computer science, and elsewhere might be found by knowing the results and techniques we develop.

In the celebrated Art of Computer Programming, Knuth states the following [36] (fasc. 1, p. 1).

> "Five basic types of questions typically arise when combinatorial problems are studied, some more difficult than others.
>
> 1. Existence: Are there any arrangements X that conform to the pattern?
> 2. Construction: If so, can such an X be found quickly?
> 3. Enumeration: How many different arrangements X exist?
> 4. Generation: Can all arrangements $X_1, X_2, \ldots$ be visited systematically?
> 5. Optimization: What arrangements maximize or minimize $f(X)$, given an objective function f?"

This dissertation will primarily discuss the fourth question: generation. For thousands of years, people have been looking for ways to efficiently list/generate combinatorial objects. While the motivations may be varied, they are linked together by the desire to compactly, yet completely, represent an entire set of objects. The three listing structures that this dissertation will focus on are Gray codes, universal cycles, and overlap sequences. The applications of these listings range from using Gray codes to solve children's puzzles or create efficient position encoders, to using universal cycles to efficiently discover a password.

In the 1940's, Frank Gray introduced what is now well-known as the reflected binary Gray code as part of a new color television system [21]. in the next few decades more work was done on expanding these minimal-change

listings over other types of objects, such as Johnson's work on generating permutations by adjacent transpositions [32]. Published in 1978 and then updated and republished in 1987, Wilf and Nijenhuis brought more attention to Gray codes as a research area in pure mathematics [50]. In 1993, Chung, Diaconis, and Graham built on the work of de Bruijn [12] and defined universal cycles, a more specific type of Gray code [8]. As a way to relax the somewhat restrictive universal cycle structure, Godbole, Knisley, and Norwood introduced the concept of overlap cycles in 2010 [19].

Gray codes have been used for many years in an extremely wide array of applications. Gray code orderings of binary strings are often used as a means of solving a synchronization problem. For example, rotary encoders utilize sequences of on-off switches, which may be thought of as binary strings. As we move from one binary string to the next, if more than one bit changes we must flip multiple switches at the same time. If the flips are not exactly simultaneous, then false intermediary states will be found. The use of Gray codes eliminates the synchronization problem.

The often quoted application of universal cycles is discovering an ATM password. Suppose that an ATM requires a 4-digit password, and at any given time considers the last four numbers entered as an entry attempt. Plugging in all possible 4-digit numbers would require typing in 4×10^4 different keystrokes. However, if we take advantage of the shifting-window type structure of the program input, we quickly realize that a universal cycle over all 4-digit numbers will solve the problem using only $10^4 + 3$ keystrokes. Thus, on average, universal cycles would allow us to break into the ATM in a more time- and energy-efficient manner by saving us from typing approximately 3×10^4 more numbers into the keypad.

As a natural extension of the "shifting window" view of universal cycles, overlap sequences are the newcomers in this area. Given the multitude of applications for Gray codes and the growing number for universal cycles, overlap sequences have the potential to be extremely useful structures.

While Gray codes have been well-studied for many years, universal cycles are relatively new to the scene. Gray codes and orderings in general have been useful in many applications, such as disk erasure codes [9]. Because of this, I believe that universal cycles and overlap sequences are not being utilized to their full extent. My work outside of the university tells me that Gray codes are well-known, but universal cycles are little known and little understood, let alone overlap sequences. However, these newer structures have the potential to be extremely useful - maybe more so than Gray codes in some situations.

Throughout this prospectus we will present many known theorems, as well as new results. Any claim that is called a result will be a new result, whereas all others are known and have citations to refer the reader to the source. Any claims labeled "fact" are trivial results, and will generally be presented without proof. All necessary background information for this prospectus is provided in Chapter 2, including the definitions of all combinatorial objects and structures discussed. Chapter 3 discusses Gray codes, Chapter 4 covers universal cycles, and Chapter 5 presents overlap sequences. In Chapter 6 we list our new results, and Chapter 7 provides a list of open problems.

Chapter 2

DEFINITIONS

2.1 Graph Theory

This dissertation will utilize many important definitions and theorems from graph theory. We define a **graph** $G = (V, E)$ to be a set V of **vertices** and a set E of **edges**. Unless otherwise stated, an edge is an unordered pair $\{i, j\}$ with $i, j \in V$. If we consider the edges to be **directed** then the edge (i, j) is directed from i to j. If all edges in a graph are directed, we will call it a **directed graph** or **digraph**. We will assume that unless otherwise noted, "graph" refers to an undirected graph with undirected edges.

Some useful graph structures that will be considered are now defined. Unless otherwise noted, all definitions correspond to both directed and undirected graphs. A **walk** is a sequence of vertices and edges $v_1 e_1 v_2 e_2 \dots e_{n-1} v_n$ with $v_i \in V$ and $e_j = \{v_j, v_{j+1}\} \in E$ for $i \in [n]$ and $j \in [n-1]$, where we use the notation $[n] = \{1, 2, \dots, n\}$. A **path** is a walk in which no vertices are repeated. A **closed walk** is a walk with $v_1 = v_n$. A closed walk may also be referred to as a **tour** or **circuit**. A closed walk in which no vertex is repeated is a **cycle**.

There are two graph structures that will be crucial to this dissertation: Euler tours and Hamilton cycles. An **Euler tour** is a closed circuit in which every edge of the graph appears exactly once. When the circuit is not closed, we call it an **Euler path**. If a graph has an Euler tour, we will call it **eulerian**.

We will characterize eulerian graphs, but first must define a few more terms. Define a vertex v in an undirected graph to be **even** if the number of edges having v as an endpoint is even. This number is called the **degree** of the vertex. If every vertex in a graph is even, then we call the graph **even**. Define an undirected graph G to be **connected** if for every pair of vertices (v, w) there exists a path from v to w in G.

Theorem 2.1.1. ([14], Theorem 1.8.1.) *An undirected graph G is eulerian if and only if it is both even and connected.*

This theorem has a corresponding theorem for directed graphs. In a directed graph, we call a vertex v **balanced** if we have

$$|\{(i, v) \mid i \in V\}| = |\{(v, i) \mid i \in V\}|.$$

In other words, the number of edges pointing to v, or the **indegree** of v is equal to the number of edges directed out of v, or the **outdegree**. We call a directed graph G **weakly connected** if for any pair of vertices (v, w) there exists an *undirected* path from v to w, i.e. the graph is connected when all edges are undirected. The graph G is **strongly connected** if for all pairs of vertices (v, w) there is a *directed* path from v to w. A **directed path** is a path that follows the direction of the edges.

Theorem 2.1.2. ([49], p. 60.) *A directed graph G is eulerian if and only if it is both balanced and weakly connected.*

The second crucial structure is a cycle in a graph that passes through every vertex exactly once, otherwise known as a **Hamilton cycle**. If a graph has a Hamilton cycle, we will call it **hamiltonian**. While proving a graph to be

eulerian is fairly simple given Theorem 2.1.2, it is not so easy to prove that a graph is hamiltonian. In general, the problem is very difficult (NP-complete in fact), but some cases can be solved easily. We will list an important theorem, and then introduce others as needed throughout the text.

Theorem 2.1.3. (Dirac's Theorem, [14], Theorem 10.1.1.) *Every graph* $G = (V, E)$ *with* $|V| \geq 3$ *and minimum degree at least* $|V|/2$ *has a Hamilton cycle.*

Because of the difference in the level of difficulty between finding Euler tours and Hamilton cycles, one of the main techniques in this work involves building transition graphs whose Euler tours correspond to the universal or overlap question.

2.2 Objects and Notation

This dissertation discusses methods for listing various sets of combinatorial objects. We now define completely the types of combinatorial objects discussed.

An **alphabet** is a set of symbols. A string of letters from some alphabet $\mathcal{A}$ is a **word**. A **subword** is a set of consecutive letters within a word. We denote the **prefix** of a word $w = w_1 w_2 \cdots w_k$ as $w^- = w_1 w_2 \ldots w_{k-1}$ and the **suffix** of w as $w^+ = w_2 w_3 \ldots w_k$. When $\mathcal{A} \subseteq \mathbb{N} = \{0, 1, \ldots\}$, we define the **weight** of the word w to be

$$\mathsf{wt}(w) = \sum_{i=1}^{k} w_i.$$

Define the operation $w_i \leftrightarrow w_j$ on a word $w_1 w_2 \ldots w_n$ to exchange letters in

positions i and j. Define another operation $w_i \leftarrow w_j$ to copy letter w_j into position i in the word.

A **binary word** of length n is an n-letter word over the alphabet $\{0, 1\}$. We will denote the set of all binary words of length n by $\mathcal{B}(n)$. When we consider the weight of some $w \in \mathcal{B}(n)$, we notice that $\mathsf{wt}(w) = |\{i \mid w_i = 1\}|$, i.e. the number of ones in the word. Define the set of fixed weight binary words as

$$\mathcal{B}_k(n) = \{w \in \mathcal{B}(n) \mid \mathsf{wt}(w) = k\}.$$

An m**-ary** word of length n is an n-letter word over an alphabet of cardinality m. Most often we will identify this alphabet with the set $\{0, 1, 2, \ldots, m-1\}$. Define $\mathcal{B}^m(n)$ to be the set of all m-ary words of length n, and $\mathcal{B}^m_k(n)$ to be the set of m-ary words of length n and weight k.

When we refer to a **partition**, we first specify what type of partition we are discussing. A **partition of an integer** n is a set of integers $\{i_1, i_2, \ldots i_k\}$ for some $k < n$ with $i_1, i_2, \ldots i_k \in \mathbb{Z}^+$ so that

$$i_1 + i_2 + \ldots + i_k = n.$$

Note that this definition does not consider the set to be ordered. That is, the partitions of 3 given by $1+2$ and $2+1$ are equivalent. When given a partition $\{i_1, i_2, \ldots i_k\}$ of an integer n, we may group together parts of the same size and write the number of equal parts as an exponent. For example, we may represent the partition $1+1+1+2+3+3$ as $1^3 2^1 3^2$. We will refer to this as the **exponential representation** for integer partitions.

A **partition of a set** A is a partition of A into disjoint subsets of A, say $\{A_1, A_2, \ldots, A_k\}$ for some k so that $A_i \cap A_j = \emptyset$ for all $i \neq j$. Unless otherwise specified, we will assume that none of the subsets A_i are empty. In this

dissertation, we will be considering both ordered and unordered partitions of sets. An **ordered** partition of a set A treats the set of subsets $\{A_1, A_2, \ldots, A_k\}$ as an ordered set, or sequence. An **unordered** partition of the set A treats the set of subsets as an ordinary, unordered set.

A **weak order** on $[n]$ is a relation $\preceq$ that is transitive and complete. We write $x \equiv y$ if $x \preceq y$ and $y \preceq x$ and $x \prec y$ if $x \preceq y$ but $y \not\preceq x$. A weak order on $[n]$ can be written as a permutation of $[n]$ with consecutive symbols separated by $\equiv$ or $\prec$ ([29], Problem 482.). For example, $1 \prec 5 \equiv 3 \prec 4 \prec 2$ is a weak order on [5].

Define $\mathcal{W}(n)$ to be the set of all weak orders on $[n]$. For each $a \in \mathcal{W}(n)$, we can define the **height** of an element $j \in [n]$ to be the number of symbols $\prec$ that precede it in the weak order. This gives an alternative representation for each $w \in \mathcal{W}(n)$ by a word $w_1 w_2 \ldots w_n$ where letter w_j is the height of element j. We will always utilize this representation of weak orders, and so we write $w = w_1 w_2 \ldots w_n$. In our previous example, the weak order $1 \prec 5 \equiv 3 \prec 4 \prec 2$ corresponds to the string 03121.

We may sometimes refer to the **height of a weak order**, denoted $\mathsf{ht}(w)$. This is defined to be the maximum letter height in the weak order. In our example, the string 03121 has height 3. It is often useful to consider only the subset of $\mathcal{W}(n)$ with a specific height. Let $0 \leq k < n$. Define

$$\mathcal{W}(n,k) = \{w \in \mathcal{W}(n) \mid \mathsf{ht}(w) = k\}.$$

Let $\mathcal{W}_k(n)$ denote the set of all weak orders on $[n]$ with weight k. When considering $\mathcal{W}_k(n)$, each weak order may be represented by the prefix of the n-tuple. Then the nth element must be $w_n = k - \sum_{i=1}^{n-1} w_i$. Call this the **prefix representation**. Later, we will generalize this definition to other classes of

combinatorial objects. Let $\mathcal{W}_k^-(n)$ denote the set of weak orders of weight k on $[n]$ using the prefix representation.

Define the **multiset** of a weak order $a = a_1 a_2 \dots a_n$ to be the unordered multiset of elements $\{a_1, a_2, \dots a_n\}$. Denote this by $\mathsf{ms}(a)$. For our example, the weak order 03121 has multiset $\{0, 1, 1, 2, 3\}$. Then we can partition $\mathcal{W}_k(n)$ by grouping together weak orders that have the same multiset. Call this the **multiset partition** $\mathcal{P} = \{P_1, \dots, P_r\}$. Define $\mathcal{W}_M(n) = \{w \in \mathcal{W}(n) \mid \mathsf{ms}(w) = M\}$ and $\mathcal{W}_M^-(n) = \{w^- \mid w \in \mathcal{W}_M(n)\}$. Define the **ground set** of $w = w_1 \dots w_n \in \mathcal{W}(n)$ to be $\mathsf{gs}(w) = \{i \mid i = w_j \text{ for some } j \in [n]\}$.

The last type of combinatorial objects considered are designs. A **design** is a pair $(X, \mathcal{B})$ where X is a finite set of elements, called **points**, and $\mathcal{B}$ is a (multi)set of subsets of X, called **blocks**. Designs may also be represented as **set systems** or **hypergraphs**. A **simple** design is a design in which the set $\mathcal{B}$ has no repeated blocks. We call a design k**-uniform** if for every $B \in \mathcal{B}$, $|B| = k$, and a design is r**-regular** if for every $x \in X$, $|\{B \mid x \in B \in \mathcal{B}\}| = r$. Given a design $(X, \mathcal{B})$, it is λ**-pairwise balanced** if for all $x, y \in X$ with $x \neq y$, we have

$$|\{B \mid \{x, y\} \subseteq B, B \in \mathcal{B}\}| = \lambda.$$

Lastly, we define a **balanced incomplete block design**, denoted $(v, k, \lambda) - BIBD$, to be a design $(X, \mathcal{B})$ where

1. $|X| = v$,
2. $(X, \mathcal{B})$ is k-uniform, and
3. $(X, \mathcal{B})$ is λ-pairwise balanced.

Three special types of balanced incomplete block designs that are considered are Steiner triple systems, denoted STS(v), Steiner quadruple systems, denoted SQS(v), and twofold triple systems, denoted TTS(v). An STS(v) is a $(v, 3, 1)$-BIBD, and a TTS(v) is a $(v, 3, 2)$-BIBD. An SQS(v) is a $3-(v, 4, 1)$-BIBD, which simply means that every 3-subset of points appears in a unique quadruple.

In constructing some types of designs, latin squares are often used. Let M be an $n \times n$ array filled with n distinct symbols. If for every row, every symbol appears exactly once, we say that M is **row latin**. If for every column, every symbol appears exactly once, we say that M is **column latin**. If M is both row and column latin, then M is a **latin square of side** n. Given two latin squares L_1 and L_2 of side n with symbols from some set X of size n, we call L_1 and L_2 **orthogonal** if for every $x, y \in X$ there are unique $i, j \in [n]$ such that $L_1(i, j) = x$ and $L_2(i, j) = y$. We call a set of latin squares **mutually orthogonal** if any two distinct latin squares from the set are orthogonal. A set of mutually orthogonal latin squares of side n is often abbreviated as a set of **MOLS**(n).

MOLS are equivalent to transversal designs. ([46], pg 144) A **transversal design**, $TD(k, n)$, is a triple $(X, \mathcal{G}, \mathcal{B})$ such that the following properties are satisfied:

1. X is a set of kn elements, called *points*,
2. $\mathcal{G}$ is a partition of X into k subsets of size n called *groups*,
3. $\mathcal{B}$ is a set of k-subsets of X called *blocks*,
4. any group and any block contain exactly one common point, and

5. every pair of points from distinct groups is contained in exactly one block.

Other variations of designs are common, such as a BIBD in which the number of blocks is equal to v, which is a **symmetric design**.

2.3 Types of Lists

This dissertation will consider three main listing structures of combinatorial objects: Gray codes, universal cycles, and overlap cycles.

2.3.1 Gray Codes

Gray codes were originally developed by Frank Gray [21] as a method of listing binary n-tuples so that successive words differ in only one position. The term Gray code has now come to mean a listing of a set $\mathcal{C}$ of combinatorial objects in which successive words differ in some predefined manner. For example, if we consider the set of binary words of length 3, or $\mathcal{B}(3)$, the original Gray code in which successive words differ in only one position is shown in Figure 2.1. Note that this Gray code is **cyclic**, that is, the minimal change property is preserved when transitioning from the last element back to the first. Gray codes will be considered in further detail in Chapter 3.

000
001
011
010
110
111
101
100

Figure 2.1: Gray code for $\mathcal{B}(3)$.

2.3.2 Universal Cycles

The topic of Chapter 4 is universal cycles. If $\mathcal{C}$ is a set of combinatorial objects each of length k, then a **universal string** is a word such that each $c \in \mathcal{C}$ appears exactly once as a subword, and so that these objects appear consecutively. If a universal string has the last $k-1$ letters matching the first $k-1$ letters, we may omit these letters and call the resulting word a **universal cycle** or **ucycle**. For example, a ucycle on the set of all 2-subsets of [5] is

$$1234531425.$$

2.3.3 Overlap Cycles

Lastly, we will discuss overlap cycles. Let $\mathcal{C}$ be a set of objects represented as words from an alphabet. An **s-overlap cycle** $O(\mathcal{C}, s)$ for $s \in \mathbb{N}$, or **s-ocycle**, is an ordered listing of the elements of $\mathcal{C}$ so that the last s letters

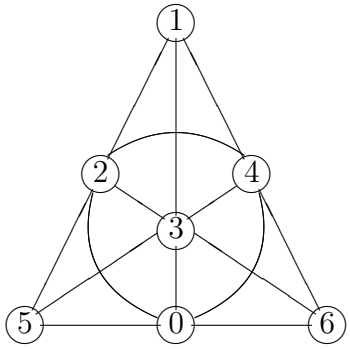

Figure 2.2: Fano Plane

of a word are the first s letters of its successor in the listing. For example, consider the design represented by the Fano plane, shown in Figure 2.2. The Fano plane represents a $(7, 3, 1)$-design, in which each block is given by any set of points that lie on the same line or circle. That is, the blocks are

$$\{1, 2, 5\}, \{1, 3, 0\}, \{1, 4, 6\}, \{2, 3, 6\}, \{4, 3, 5\}, \{5, 0, 6\} \text{ and } \{0, 2, 4\}.$$

We can represent this design using the 1-overlap cycle

$$\overline{52}\underline{10}\overline{35}\underline{41}\overline{63}\underline{24}\overline{06}\underline{5}.$$

The blocks are underlined or overlined in order to show the overlap of size one between consecutive blocks. Overlap cycles will be discussed in Chapter 5.

2.3.4 Variations

When a class of combinatorial objects does not lend itself well to one of the previously discussed structures, there are several variations that can be considered.

2.3.4.1 Packing Words

A **packing word** over a set of combinatorial objects is a string with all of the properties of a universal string, except not every object needs to appear. With packing words, we generally want to maximize the length of the string, which will simultaneously maximize the number of objects that appear.

In proving that no ucycle for $(n-2)$-subsets of $[n]$ can exist, Stevens, et al., also proved the following two theorems.

Theorem 2.3.1. [45] *The longest possible cyclic packing word of* $(n-2)$*-subsets of* $[n]$ *has length* n*, and a word achieving this bound always exists.*

The packing word that always achieves this bound is surprisingly simple:

$$1, 2, \ldots, (n-1), n.$$

If we no longer require the packing word to be cyclic, we can create a longer packing word on the same set of objects.

Theorem 2.3.2. [45] *The longest possible packing word of* $(n-2)$*-subsets of* $[n]$ *has length* $3n-6$ *and a word achieving this bound always exists.*

In [11], this notion of increasing a packing word to contain as many k-subsets of $[n]$ as possible is extended. Define a **near-ucycle packing** as a sequence S of n packing words so that as $n \to \infty$, asymptotically few k-subsets are omitted from S. Using this definition, they were able to prove the following result.

Theorem 2.3.3. [11] *For all* $0 < k < n$*, near-ucycle packings exist for* k*-subsets of* $[n]$*.*

While all of these results concern subsets, the techniques may be useful when consider other objects as well.

2.3.4.2 Covering Words

A **covering word** over a set of combinatorial objects is a string with all of the properties of a universal string, except that the words do not need to appear contiguously. That is, some consecutive objects in the string may have a large overlap (say $n-1$, as desired in a ucycle), while others may have a smaller overlap (say s, as desired in an s-overlap cycle). Note that any overlap cycle is a covering word.

2.3.4.3 Multicover Universal Cycles

Covering words are often extended to multicover universal cycles. Let $\mathcal{C}$ be a set of combinatorial objects represented by words of length k. Recall that a universal string over $\mathcal{C}$ is a string in which each $c \in \mathcal{C}$ appears exactly once, and each k-letter subword is an object from $\mathcal{C}$. A **t-multicover universal string** is a universal string, with the exception that each object appears exactly t-times. If a t-multicover string has the last $k-1$ letters matching the first $k-1$ letters we may omit these letters and call the resulting word a **t-multicover universal cycle**. Since each object appears exactly t times, it is clear that a 1-multicover universal cycle is the standard ucycle discussed in Chapter 3. From this, we easily get the following fact.

Fact 2.3.4. *Let $\mathcal{C}$ be a set of combinatorial objects. If a universal cycle over $\mathcal{C}$ exists, then a t-multicover universal cycle exists, and is simply the universal*

cycle taken t times.

For all of the classes of combinatorial objects that do not easily yield to the universal cycle structure, it would be interesting to determine the minimum value for t such that the class of objects allows a t-multicover universal cycle.

In [27], Hurlbert presents many results on multicover ucycles for k-subsets of $[n]$. Define $U(n,k)$ to be the minimal t for which a t-cover ucycle exists for k-subsets of $[n]$. The main results presented in [27] are as follows. Note the similarity between Theorem 2.3.5 and Conjecture 4.5.2 (and the discussion following the conjecture).

Theorem 2.3.5. [27] *If $k = 2, 3, 4$ or 6, then there exists $n_0(k)$ such that if $n \geq n_0(k)$ and $gcd(n,k) = 1$, then $U(n,k) = 1$.*

Theorem 2.3.6. [27] *$U(n,k) \leq k$ for all $n \geq k$.*

Much of the work on covering words and multicover universal cycles pertains only to k-subsets of $[n]$.

2.3.4.4 Rosaries

While packing and covering words have been discussed largely in terms of subsets, rosaries often involve permutations. A **rosary** for a set of combinatorial objects, each of size k, is a cyclic sequence $x_1 x_2 \ldots x_r$ in which every combinatorial object can be found as a subsequence of $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ for some $\{i_1, i_2, \ldots, i_k\} \subseteq [r]$. Note that this subsequence does not need to be consecutive. See Figure 2.3 for an example of a rosary for permutations of [4].

We define the minimum r such that this property holds to be the **rosary number** for the set of objects. A well-known conjecture on the rosary num-

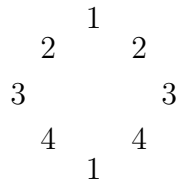

Figure 2.3: Example rosary for permutations of [4]

ber for permutations of $[n]$ is the following. Define Π_n to be the set of all permutations of $[n]$.

Conjecture 2.3.7. [22] *For all integers $n > 1$, the inequality $r(\Pi_n) \leq \frac{n^2}{2}$ is satisfied.*

Several results have been published non-cyclic sequences, such as the following.

Theorem 2.3.8. [37] *For $n \geq 3$, there exists a string of length $n^2 - 2n + 4$ such that every permutation in Π_n appears as a subsequence.*

2.4 Common Techniques / Methods

2.4.1 Graphical Representations

The most commonly used method for finding these listings of combinatorial objects is to create the **transition graph**. In this directed graph, we define the set of vertices to be the set of combinatorial objects $\mathcal{C}$. For any $c, d \in \mathcal{C}$, we draw an edge from c to d in the graph if the transition from object c to object d is allowed under the given type of structure. For example, the transition graph for a universal cycle on $\mathcal{B}(3)$ would have an edge from $(0, 0, 0)$ to $(0, 0, 1)$, but

not the reverse from $(0,0,1)$ to $(0,0,0)$, since the last two letters of $(0,0,1)$ do not agree with the first two letters of $(0,0,0)$.

Given this definition of the transition graph, a Hamilton cycle in the graph corresponds to the listing structure that we desire. This is clear, since following a Hamilton cycle we will pass each object from $\mathcal{C}$ exactly once, and the necessary transition properties will be satisfied by our definition of edges in the graph.

While this structure lays out a straightforward method to find the desired listing, it is not computationally easy. That is, the problem of finding a Hamilton cycle is a well-known NP-complete problem that cannot be solved easily in all cases. There are several classes of graphs that can be solved quickly, such as the n-cube Q_n (the transition graph for $\mathcal{B}(n)$ under the original Gray code). However, in general the problem is too difficult to be solved on large instances. For this reason, we would like to modify the graph to represent the combinatorial objects as edges instead of vertices. Then we are searching for an Euler tour, which is a computationally easy problem. The downside to this approach is that it is not always clear how to obtain this type of eulerian transition graph.

2.4.2 Euler Tours vs. Hamilton Cycles

Given a graph, the problem of finding an Euler tour and the problem of finding a Hamilton cycle seem to be structurally very similar, and yet computationally completely opposite. In this section, we will discuss methods and

approaches for dealing with both problems.

If we are searching for an Euler tour in a graph, we can determine whether such a tour exists using Theorem 2.1.1 for an undirected graph, or Theorem 2.1.2 for a directed graph. To show that a given graph satisfies one of these theorems, several standard approaches are used. First, to show that a graph is even or balanced, it is often useful to pair up edges incident to a particular vertex and show that this pairing defines a 2-regular partition of the incident edge set. To show that a graph is connected, there are two common techniques. One technique is to partition the graph into connected subgraphs, and then show that all of these subgraphs are connected. The other identifies a specific root vertex, usually referred to as the **minimum vertex**, and then shows that a path exists from every vertex to the root.

Once we have determined the existence of an Euler tour in a graph, we can produce the desired sequence by following the tour and listing the objects as we visit their corresponding edges in the order of the Euler tour. When considering overlap cycles, we may omit the non-overlapping portions of each object in order to more compactly represent the final result.

However, if we are unable to transform our problem into an eulerian-type problem, we are stuck attempting to determine the hamiltonicity of our transition graph. Most efficient algorithms for the Hamilton cycle problem deal only with specific classes of graphs, for example graphs with high minimum degree. For this reason, there is no standard approach to finding Hamilton cycles in a graph - different instances may require different methods. While algorithms for solving these problems will not be a focus in this dissertation, there are a few theorems that allow us to prove the existence of a Hamilton cycle easily in specific graphs, such as Dirac's Theorem (Theorem 2.1.3).

2.5 Alternative Word Representations

For some of the structures and sets of combinatorial objects, we may not be able to create the listing desired using the objects in their "standard" representation. The **standard representation** for a set of combinatorial objects must be defined, for most sets have multiple representations, all of which may be commonly used in the literature. Listed below, we briefly define the standard representation for each class of objects considered. Most classes have alternative representations that will be defined throughout.

Binary n-tuples: A binary n-tuple will be represented as a word of length n over the alphabet $\{0, 1\}$.

m-ary Words: An m-ary word of length n will be represented as a word of length n over the alphabet $\{0, 1, \ldots, m-1\}$.

Permutations: A permutation will be represented in functional notation as a word. For example, the permutation 312 corresponds to the function $f : [3] \to [3]$ given by $f(1) = 3$, $f(2) = 1$, and $f(3) = 2$.

Subsets: A subset will be represented as an unordered word. For example, the 2-subsets of [3] in standard representation are 12, 13, and 23. Note that 12 is equivalent to 21 since the word is unordered.

Partitions of Integers: A partition of an integer will be represented as a word. For example, the partition $5 = 1 + 2 + 2$ is written as 122. If the partition is unordered, then the word is unordered, i.e. 122 is equivalent to 212 and 221.

Partitions of Sets: A partition of a set will be represented by a word, representing the set, separated with vertical bars. For example, the partition of $\{1, 2, 3\}$ given by splitting the set as $\{1, 2\}$, $\{3\}$ is represented by 12|3. Note that the ordering within partition sets is irrelevant, as well as the position of the partition set as a whole. For example, 12|3 is equivalent to 21|3, 3|12, and 3|21.

Weak Orders: A weak order will be represented as a word of length n using the height of each element as defined in Section 2.2.

Designs: A design will be represented as a list of blocks. Each block in a design will be represented as an unordered word, just as with subsets.

2.5.1 Natural Representation Choices

Many of the classes of combinatorial objects discussed have several natural representation choices other than the standard representation that we have defined. For example, subsets of $[n]$ could also be represented as a binary string on length n, with a 1 in position i if and only if i is in the subset. We will refer to this as the **subset membership representation**. It is also often called the **incidence vector**.

Permutations also have many possible representations. For example, instead of the standard functional notation, we could write a permutation in **disjoint cycle representation**. Each permutation can be partitioned into disjoint cycles, for example the permutation 13254 in standard notation is represented as (1)(23)(45) using the disjoint cycle representation. For other

representations of permutations, see [44], Section 1.3. Depending on the type of listing desired and the subset of the combinatorial class being considered, we may need to utilize these other natural representations.

2.5.2 Dropping Symbols

One method of altering the standard representation is to drop unnecessary symbols. For example, the first $n-1$ letters of a permutation of $[n]$ completely determines the last. Thus we can utilize what we call the **prefix representation** in which we drop the last letter of each string in standard representation. The prefix representation can be used on any set $\mathcal{C}$ of combinatorial objects in which every object $c = c_1c_2 \ldots c_n \in \mathcal{C}$ has the last letter c_n completely determined by the first $n-1$ letters. We can use the same type of prefix representation for fixed weight binary strings and fixed weight weak orders. Every element in each of these classes of combinatorial objects has the last letter completely determined by the previous letters.

In these examples of the prefix representation, it is trivial in a word of length n to recover the nth letter using the first $n-1$ letters. When a given substring of the word completely defines the combinatorial object, we call that substring a **basis** for the object. If we represent every word by a basis, we are listing the words using a **basis representation**. For example, in a $(v, k, 1) - BIBD$, we may drop up to $k-2$ symbols, leaving at least 2 symbols to represent each block. Since $\lambda = 1$ in this case, each set of two symbols appears in exactly one block, making this a valid representation in which each

block has a unique representative. The string of size two is a basis for the entire block. However in this case, it is not such a trivial task to recover the original block containing a given pair of elements. In fact, the simplest way to recover the original block from the design is to have a lookup table to refer to. In this case, it is reasonable to ask whether the benefits of dropping unnecessary symbols outweigh the computational cost of referencing a lookup table to recover each block from a defining pair of elements. For this reason, overlap cycles could be a useful alternative to universal cycles. Some classes of objects cannot be conformed to the structure of a universal cycle without dropping symbols, but by considering an overlap cycle instead, we may be able to leave all symbols in place.

2.5.3 Adding Symbols

In some instances, we may have a class of combinatorial objects that does not conform to the desired listing structure. If dropping symbols is not an option and we are unwilling to change the kind of structure, it may be possible to add symbols to the given alphabet to obtain the listing that we are looking for.

For example, consider attempting to find a universal cycle for permutations. If we represent each permutation completely in the standard representation, it is not possible to construct a universal cycle. To illustrate this, consider the transition graph in this scenario. If we look at a specific permutation $w_1 w_2 \dots w_n$, the only outneighbor in the transition graph will be

$w_2 \dots w_n w_1$. In fact, the transition graph will consist of disjoint cycles that correspond to the rotation of one permutation, similar to the setup illustrated in Figure 4.1. Thus the transition graph is not connected, so it is not possible to find a Hamilton cycle. We will see in Section 4.4 that it is possible to find a ucycle for permutations using the prefix representation, but if we do not want to drop symbols we must find some alternative representation method.

One method is to utilize order isomorphic representations, and is discussed in [31]. In this paper, the size of the alphabet for permutations of $[n]$ is increased from n symbols to $n+1$ symbols. Over this new alphabet, two representations of permutations $a_1 a_2 \dots a_n$ and $b_1 b_2 \dots b_n$ are **order isomorphic** if $a_i < a_j$ if and only if $b_i < b_j$ for all $1 \leq i, j \leq n$. Using this idea, universal cycles for permutations can be constructed with order isomorphic representations.

Another alternative is to consider equivalence class representations [28]. In this method, the size of the alphabet for permutations of $[n]$ is again increased from n symbols to $n+1$ symbols. However, this time, we define a relationship $\sim$ between words $a = a_1 a_2 \dots a_n$ and $b = b_1 b_2 \dots b_n$ over this new larger alphabet by $a \sim b$ if and only if there exists some k such that for all $i \in [n]$ we have $a_i - b_i \equiv k \pmod{n+1}$. Then $\sim$ defines an equivalence relation in which each equivalence class represents a unique permutation. Using this representation, universal cycles for permutations can be constructed.

Chapter 3

GRAY CODES

3.1 Introduction and Definitions

Definition 3.1.1. *A **Gray code** for a set of combinatorial objects is an ordering of the set so that successive elements satisfy a given minimal change property.*

We call a Gray code **cyclic** if the transition from the last element to the first in the list also satisfies the minimal change property.

For example, consider the set of partitions of 4, shown below.

$$\left\{ \begin{array}{c} 4 \\ 3+1 \\ 2+2 \\ 2+1+1 \\ 1+1+1+1 \end{array} \right\}$$

If we consider each partition to have length 4 (adding zeros as needed), then we define our minimal change property so that to move from one partition to the next we decrease one part by 1 and increase another by 1. Then one type

of Gray code that utilizes this minimal change property is:

$$4+0+0+0$$
$$3+1+0+0$$
$$2+2+0+0$$
$$2+1+1+0$$
$$1+1+1+1$$

Note that this Gray code is not cyclic. In fact, when we examine the first partition in our list, $4+0+0+0$, we see that there is only one possible predecessor or successor, $3+1+0+0$, since we only have one part that can be decreased. Thus there is no cyclic Gray code possible using this minimal change property and this set of objects.

3.2 Binary Words of Length n

One of the first examples of a Gray code is the Chinese Rings puzzle, illustrated in Figure 3.1 ([36], fasc. 2, p. 5). While the origins of the puzzle are unclear, it is believed to have been first created around the end of the second century A.D. The puzzle consists of several interlocked rings that only allow two moves:

1. The last ring on the right can be taken off of or placed back on to the top rod.

2. Any other ring can be moved on or off of the top rod if and only if its neighbor ring to the right is on the bar, and all rings to the right of the neighbor ring are off of the rod.

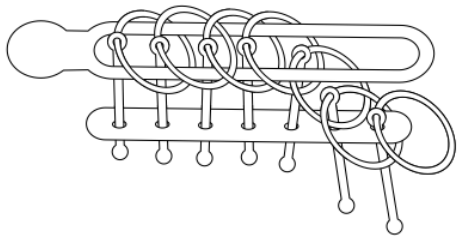

Figure 3.1: The Chinese Rings Puzzle

It is not immediately obvious how this puzzle relates to our initial discussion. However, if there are n rings, we may number the rings from 1 to n and define word $w = w_1 w_2 \dots w_n \in \{0,1\}^n$ to be defined where $w_i = 1$ if and only if ring i is on the top rod. Now each state of the puzzle is represented by a binary word of length n. The initial state in which all rings are on the top rod is represented by the word $11 \cdots 1$ and the final state in which all rings are off of the top rod is represented by the word $00 \cdots 0$. To move to the next state, we may change only one letter in the word that represents the current state, according to the rules outlined above. Thus the solution to the Chinese Rings puzzle is also an algorithm to generate a binary Gray code.

One of the major results that spurred research in this area was the development of the **reflected binary code**, described in Theorem 3.2.1. This cyclic Gray code is over the set $\mathcal{B}(n)$, the set of binary words of length n.

Theorem 3.2.1. [21] *There exists a cyclic Gray code listing of $\mathcal{B}(n)$ with successive elements differing in exactly one bit.*

Proof. We show Gray's original construction of the reflected binary code. This is done in a recursive manner in which all binary words of length n are generated using the code for binary words of length $n-1$.

Our initial case is for $n = 1$, and we have the code:

0

1

Now suppose that we have the reflected binary code for $\mathcal{B}(n-1)$. We construct the reflected binary code for $\mathcal{B}(n)$ as follows.

1. List the code for $\mathcal{B}(n-1)$.
2. Next list the code for $\mathcal{B}(n-1)$ backwards, i.e. the last element is now the first.
3. For each word from (1), add a 0 to the front.
4. For each word from (2), add a 1 to the front.

We now show that this gives a code in which successive elements only differ in one position. First note that the initial code for $\mathcal{B}(1)$ satisfies this property. Now given that $\mathcal{B}(n-1)$ is correct, we know that the lists generated from steps (1), (3), and (2), (4) are both incomplete codes in which successive elements differ in only one position. Thus we need only compare the words at the end of one list and the start of the next. Note that in both cases (the words in positions 1 and 2^n, and the words in positions 2^{n-1} and $2^{n-1}+1$) the words only differ in the first entry, and thus satisfy the necessary property. Thus the code generated has successive elements differing in only one position. $\square$

For example, we obtain $\mathcal{B}(2)$ from the initial given code for $\mathcal{B}(1)$ as shown below.

$$\begin{array}{ccccccc} & & 0 & & 00 & & 00 \\ 0 & & 1 & & 01 & & 01 \\ & \rightarrow & & \rightarrow & & \rightarrow & \\ 1 & & 1 & & 1 & & 11 \\ & & 0 & & 0 & & 10 \end{array}$$

This code has many applications. One particular benefit of the reflected binary code is that it avoids the problem of timing multiple bit flips so that they occur at exactly the same time. It requires only one bit flip at each stage. It also implies a useful result from graph theory. Recall that the graph Q_n is the graph whose vertex set is the set $\mathcal{B}(n)$ and two vertices have an edge between them if and only if they differ in exactly one bit. Applying the reflected binary code, we see that the graph Q_n has a hamiltonian cycle.

If we now restrict our attention the set $\mathcal{B}_k(n)$, we see that we cannot possibly hope to find a Gray code with the same property. For if we change a word from $\mathcal{B}_k(n)$ to move to another word that differs in only one bit, we are no longer considering a word of weight k. We have changed a bit from 0 to 1, or vice versa. This means increasing or decreasing the weight of the original word by one, respectively.

However, if we allow ourselves to consider allowing more than one bit to change at each step, we arrive at what are called **revolving door combinations**.

Theorem 3.2.2. (Revolving Door Combinations, [41], p. 28.) *There exists a Gray code listing for $\mathcal{B}_k(n)$ in which successive elements differ in exactly two positions.*

Proof. Let $L(n,k)$ denote the desired listing for $\mathcal{B}_k(n)$. Assume that $L(n,k)$ starts with the lexicographically smallest string and ends with the largest. We will show that

$$L(n,k) = (0) \oplus L(n-1,k), (1,0) \oplus \overline{L(n-2,k-1)}, (1,1) \oplus L(n-2,k-2)$$

where $\overline{L(n,k)}$ denotes the list $L(n,k)$ in reverse order and $\mathbf{v} \oplus L(n,k)$ denotes the word $\mathbf{v}$ appended to the start of each word in the list $L(n,k)$.

We will prove that this holds by induction. The base cases are displayed in the table below.

$n \setminus k$	0	1	2
1	0	1	
2	00	01, 10	11

First, from the structure of the list, every string from $\mathcal{B}_k(n)$ must appear at least once in $L(n,k)$. Every string that starts with 0 appears in $(0) \oplus L(n-1,k)$, every string that starts in 10 appears in $(1,0) \oplus \overline{L(n-2,k-1)}$, and every string that starts in 11 appears in $(1,1) \oplus L(n-2,k-2)$.

All that remains is to show that the minimal change property holds at the two points where one sublist meets another, and that our list begins with the minimum and ends with the maximum. Other than these two intersection points, we know that the minimal change property must hold by the induction hypothesis.

The first intersection we must check is between the last word of $(0) \oplus L(n-1,k)$ and the first word of $(1,0) \oplus \overline{L(n-2,k-1)}$. Since the last word of $L(n-1,k)$ is $1^k 0^{n-k-1}$ and the first word of $\overline{L(n-2,k-1)}$ is $1^{k-1}0^{n-k-1}$, we see that the intersection strings differ in the first two positions.

Next we must check the last word, say x, of $(1,0) \oplus \overline{L(n-2,k-1)}$ and the

first word, say y, of $(1,1) \oplus L(n-2,k-2)$. Again, we know exactly what x and y are. We must have $x = 100^{n-k-1}1^{k-1}$ and $y = 110^{n-k}1^{k-2}$, which differ in positions 2 and $n-k+2$.

Lastly we must check that this list begins with the minimum string and ends with the maximum string. Since the first string in $(0) \oplus L(n-1,k)$ must be $00^{n-k-1}1^k = 0^{n-k}1^k$, the list begins with the minimum string. Similarly, the last string in $(1,1) \oplus L(n-2,k-2)$ must be $111^{k-2}0^{n-k} = 1^k0^{n-k}$, and so the list ends with the maximum string. □

To make this result stronger, we may restrict the locations of the two changed bits.

Theorem 3.2.3. ([36] fasc. 3, p. 11.) *There exists a Gray code listing of $\mathcal{B}_k(n)$ allowing only the operations $w_i \leftrightarrow w_{i+1}$ and $w_i \leftrightarrow w_{i+2}$.*

In fact, this is as good as we can hope for. It is not possible to restrict the positions to be adjacent as before. For example, consider the set $\mathcal{B}_2(4)$. Shown in Figure 3.2 is the transition graph. It is immediately clear that this graph is not hamiltonian, and since it does not even contain a Hamilton path, it is impossible to find a Gray code with this minimal change property. The following theorem tells us when we can expect to find a Gray code for $\mathcal{B}_k(n)$ using adjacent transpositions.

Theorem 3.2.4. [33] *Let $n, k \in \mathbb{Z}^+$. There exists a Gray code for $\mathcal{B}_k(n)$ using adjacent transpositions when:*

- *$k = 3$ or 4 and n is odd, or*
- *$k = 5$ and $n \neq 7$.*

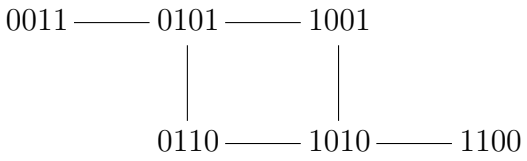

Figure 3.2: Transition graph for $\mathcal{B}_2(4)$ using adjacent transpositions

There does not exist a Gray code for $\mathcal{B}_k(n)$ using adjacent transpositions when:

- *both n and k are even, or*
- *$k = 2$ or $n - 2$ and n is odd.*

Instead of considering fixed weight, we could consider alternative weight restrictions on $\mathcal{B}(n)$.

Open Problem 3.2.5. *Is there a Gray code for the set $\mathcal{B}_o(n)$, the set of all odd-weight binary strings of length n? What about for $\mathcal{B}_e(n)$, the set of all even-weight binary strings of length n?*

Many of the open problems in this area fall into one of two categories: finding Gray codes under a different definition of minimal change, or finding Gray codes for a restricted subset of $\mathcal{B}(n)$ using some previously defined notion of minimal change.

Note that depending on the application, the definition of minimal change varies greatly. This can be seen in the variety of published results in this area. For example, we have the following variations on Gray codes for $\mathcal{B}(n)$.

We define an equivalence class $\sim$ in which for $a_1a_2 \dots a_n, b_1b_2 \dots b_n \in \mathcal{B}(n)$, we have

$$a_1a_2 \dots a_n \sim b_1b_2 \dots b_n \Leftrightarrow \exists j \in [n] \text{ s.t. } a_ja_{j+1} \dots a_na_1a_2 \dots a_{j-1} = b_1b_2 \dots b_n.$$

We define **necklaces** over $\mathcal{B}(n)$ to be these equivalence classes under $\sim$. It is standard practice to represent each necklace by the lexicographically largest string in the equivalence class.

Theorem 3.2.6. [47] *There is a Gray code listing of necklaces of 0's and 1's with fixed weight where adjacent elements differ by a single transposition of a 0 and 1.*

Another example is known as "balanced Gray codes". Given a Gray code for $\mathcal{B}(n)$, define the **delta sequence** $\delta_0, \ldots, \delta_{2^n-1}$ where δ_i indicates the position of the change between the $(i-1)$th and ith binary words in the list. Next define the **transition count** of a specific delta sequence to be $(c_0, \ldots, c_{n-1})$ where c_j is the number of times $\delta_k = j$.

Theorem 3.2.7. (Balanced Gray Codes, [36], fasc. 2, p. 14.) *For all $n \geq 1$, there is a Gray code for $\mathcal{B}(n)$ with transition counts $(c_0, c_1, \ldots c_{n-1})$ that satisfy the condition*

$$|c_j - c_k| \leq 2 \text{ for } 0 \leq j < k < n.$$

Another type of Gray code considers the **run lengths** of delta sequences. Given a delta sequence, we define the run length to be the minimum k such that there exists some i with $\delta_i = \delta_{i+k}$.

Open Problem 3.2.8. ([36], fasc. 2, p. 15.) *Let $r(n)$ be the maximum value r such that a Gray code for $\mathcal{B}(n)$ exists with run length at least r. What are the values of $r(n)$?*

The values for n up to 8 are known, however the values beyond that are yet to be determined.

As far as Gray codes for restricted subsets of $\mathcal{B}(n)$, we consider the subset of $\mathcal{B}(n)$ that consists of strings that do not contain k consecutive 0's.

Result 3.2.9. *For all $n \geq 0$ and $k \geq 1$, there exists a Gray code listing for the set*

$$\hat{\mathcal{B}}_k(n) = \{b \in \mathcal{B}(n) \mid \nexists i \in [n-k+1] \text{ with } b_i = b_{i+1} = \cdots = b_{i+k-1} = 0\}$$

in which successive elements differ in exactly one position.

Proof. Let $L(n)$ denote the desired listing for $\hat{\mathcal{B}}_k(n)$. We proceed by induction on n. Our base cases are as follows. For $n = 0$, our set is empty so the claim is vacuously true. When $n = 1$, the only choice for $k \leq n$ is $k = 1$. For this listing we have the sequence 1. When $k > n$, we choose the listing $1, 0$. When $n = 2$ and $k = 1$, we know the set: $\hat{\mathcal{B}}_1(2) = \{11\}$. Thus there is only one possible listing: 11. When $n = 2$ and $k = 2$, we can use the listing $01, 11, 10$. For $n = 2$ and $k > 2$, a possible listing is $01, 11, 10, 00$.

For the cases with $n > 2$, we want to list the elements of $\hat{\mathcal{B}}_k(n)$ so that the number of zeros at the end of each string is a non-decreasing function. We call this the **non-decreasing property**. Note that this property is maintained in our base cases.

For $n > 2$, we claim that one possible listing is

$$L(n) = \begin{bmatrix} \overline{L(n-1)} & \oplus & 1, \\ L(n-1)^* & \oplus & 0, \end{bmatrix}$$

where $L(n-1)^*$ is the listing for $\hat{\mathcal{B}}_k(n-1)$ with strings ending in 0^{k-1} omitted, and $\overline{L(n-1)}$ is the listing for $\hat{\mathcal{B}}_k(n-1)$ in reversed order. To show that this listing works, we must show three things: (1) the nondecreasing property is

maintained, (2) successive elements differ in exactly one position, and (3) the last string does not end in k zeros.

First, the beginning portion of $L(n)$ given by $\overline{L(n-1)} \oplus 1$ has every string ending in 1, so has zero 0's at the end. The last portion of $L(n)$, given by $L(n-1)^* \oplus 0$ has every string ending in 0. Since $L(n-1)$ satisfies the nondecreasing property, we know that $L(n-1)^* \oplus 0$ must also. Since every string in $L(n-1)^* \oplus 0$ has at least one 0 at the end, the nondecreasing property is satisfied over the entire listing $L(n)$.

Second, note that in both $\overline{L(n-1)}$ and $L(n-1)^*$, successive elements differ in exactly one position by the recursive construction. Thus $\overline{L(n-1)} \oplus 1$ and $L(n-1)^* \oplus 0$ also satisfy our Gray code property. So we only need to show that the last element of $\overline{L(n-1)} \oplus 1$ and the first element of $L(n-1)^* \oplus 0$ differ in exactly one position. By the nondecreasing property, we know that the last element of $\overline{L(n-1)} \oplus 1$ ends in 1, and so is the the same as the first element of $L(n-1)^*$, and hence they only differ in the last position.

Lastly, by the nondecreasing property, the last string in $L(n-1)^*$ ends in 10^{k-2}. Thus the last string in $L(n-1)^* \oplus 0$ ends in 10^{k-1}. Thus $L(n)$ is one possible listing for $\hat{\mathcal{B}}_k(n)$ that satisfies the desired minimal change property. □

Because $|\hat{B}_2(n)| = F_{n+1}$, where F_m denotes the mth Fibonacci number, the preceeding theorem can be used to construct Gray codes for any other set in bijection with $\hat{B}_2(n)$, such as the set of compositions of n into odd parts (see [44] for more examples).

Another variation on restricted subsets of $\mathcal{B}(n)$ is the set of strings of balanced parentheses. When we consider a word of $2n$ nested parentheses, we

can choose to associate it with a binary word of length $2n$ by assigning 1 to represent a left parenthesis and a 0 for a right parenthesis, or vice versa.

Theorem 3.2.10. ([36], fasc. 4, p. 8.) *It is possible to create a Gray code listing for the set of all possible words of length $2n$ of nested parentheses so that successive words differ by either of the following two operations:* $() \leftrightarrow)($ *or* $()) \leftrightarrow))($.

Since it is well-known that the number of strings of balanced parentheses of length $2n$ is C_n, where C_m is the mth Catalan number, this theorem can be used to find Gray codes for various other structures, such as Dyck words (see [44] for more examples).

3.3 m-ary Words of Length n

Recall that the set of m-ary words of length n are identified with words on the alphabet $\{0, 1, \ldots, m-1\}$. We will denote the set of all such m-ary words of length n by $\mathcal{B}^m(n)$. Using this notation, we see that the set of all binary words of length n is denoted by $\mathcal{B}^2(n)$. The first main theorem in this area is very similar to what we saw for binary words.

Theorem 3.3.1. [17] *For every $n, m \in \mathbb{Z}^+$, there exists a Gray code listing for $\mathcal{B}^m(n)$ so that each word differs from its successor in exactly one index position.*

Proof. We define the Gray code recursively. First, for our base case, when $n = 0$ we have an empty list of words. Next, suppose that we have the desired

listing for $\mathcal{B}^m(n-1)$, call it $L^m(n-1)$. Let $\overline{L^m(n-1)}$ denote the list written in reverse, with the last word first. Then we construct $L^m(n)$ as shown below:

$$L^m(n) = \left[\begin{array}{ll} 0 & L^m(n-1) \\ 1 & \overline{L^m(n-1)} \\ 2 & L^m(n-1) \\ \vdots & \\ m-1 & \begin{cases} L^m(n-1) & \text{if } m-1 \text{ is even,} \\ \overline{L^m(n-1)} & \text{if } m-1 \text{ is odd} \end{cases} \end{array}\right.$$

It is clear inductively that this has the desired minimal change property. Clearly the empty string has the desired property for our base case. Then, if $L^m(n-1)$ satisfies the minimal change property, then so does $\overline{L^m(n-1)}$. Note that the last element of $L^m(n-1)$ is the same as the first element of $\overline{L^m(n-1)}$ and that the first element of $L^m(n-1)$ is the same as the last element of $\overline{L^m(n-1)}$. So, whenever we change the value of the first letter, all other letters remain constant. Whenever the first letter remains constant, we know by the induction hypothesis that the rest of the word only changes in one position. □

Next, we consider fixed weight m-ary words of length n. Recall that the weight of a word is the sum of its letters. The following result is known and published in [48], however we provide an alternative algorithm.

Result 3.3.2. *There exists a Gray code listing for $\mathcal{B}_k^m(n)$ in which successive words differ in at most two positions.*

We will present an algorithm and then prove that it is correct. First, we define a few simple functions that will be used in the algorithm. Given a list

L, Rev(L) produces the list in reversed order. The second function needed is the exponent function, defined as:

$$\text{Expo}(L, e) = \begin{cases} L & \text{if } e \equiv 0 \pmod 2 \\ \text{Rev}(L) & \text{if } e \equiv 1 \pmod 2. \end{cases}$$

Finally, the function Pref(a, L) adds the prefix a to every string in list L. This operation may also be denoted $a \oplus L$. Now we can provide the following algorithm, borrowing the reflection idea from Gray's original algorithm for binary words [21]:

```
FWM(m, n, k):
    L = [ ];
    if (m-1)n >= k and k >= 0 then
        if (m-1)n = 0 then
            L = [ [ ] ];
        for i from 0 to min(m-1, k) do
            M = Pref(i, Expo(FWM(m, n-1, k-i), i));
            L = L, M;
    return L;
```

We will work through an example of the algorithm when we run `FWM(3,4,5)`. In the lists, double lines indicate changes in our outer loop, and single lines indicate changes in the secondary loop.

We begin with L, an empty list.

A

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

B

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

C

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

Figure 3.3: Various sublists for the FWM algorithm

Starting with $i = 0$, we need to determine

$$M = 0 \oplus \texttt{FWM(3,3,5)} = \text{Pref}(0, \text{Expo}(\texttt{FWM(3,3,5)}, 0)).$$

We start by determining this sublist, `FWM(3,3,5)`.

`FWM(3,3,5)`$= [0 \oplus$ `FWM(3,2,5)`$, 1 \oplus \text{Rev}($`FWM(3,2,4)`$), 2 \oplus$ `FWM(3,2,3)`$]$:

For these sublists, we have:

1	2	2
2	1	2
2	2	1

Returning to our original list, this sequence is preceded by a 0 to produce list **A** in Figure 3.3. When $i = 1$, our list continues with list **B** in the same figure. When $i = 2$, our list finishes with list **C** from the figure.

This gives us the final list:

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

Next, we describe more clearly what is happening in this algorithm. The algorithm is recursive, and the strings are organized so that every string beginning with i comes before every string beginning with $i+1$. However, within these subsets of our list the ordering is not so simple. When we consider a sublist of strings that all begin with the same prefix $w_1 w_2 \dots w_\ell$, we can determine whether the list is reversed or not by considering $\sum_{i=1}^{\ell} w_i$. If the sum is odd the list is reversed and if the sum is even then it is not. This immediately tells us the ordering of the $(\ell+1)$st elements in this sublist.

Lemma 3.3.3. *The first element of the list* `FWM(m,n,k)` *is*

$$0 \cdots 0r(m-1) \cdots (m-1),$$

where $k = q(m-1)+r$ *for some* q *and* r *where* $0 \leq r < m-1$. *To determine the last element of the list, we define* $u_1 = \min\{m-1, k\}$ *and* $k - u_1 = q'(m-1)+r'$ *for some* q' *and* $0 \leq r' < m-1$. *Then the last element of the list is*

$$\boldsymbol{u} = \begin{cases} u_1(m-1) \cdots (m-1)r'0 \cdots 0, & \text{if } u_1 \text{ is even, and} \\ u_1 0 \cdots 0r'(m-1) \cdots (m-1), & \text{if } u_1 \text{ is odd.} \end{cases}$$

Proof. It is clear from the algorithm that any list always starts with the minimum string in lexicographic order. Thus the list must start with $0 \cdots 0r(m-1) \cdots (m-1)$.

To find the last element of the list, we proceed by induction on n. For our base case, we consider $n = 1$. When $n = 1$, clearly there is only one string: k. Note that this agrees with our definition of $\mathbf{u}$.

Before considering the two cases of u_1 either odd or even, we note that if $u_1 = k$ then the string must be

$$u_1 0 \cdots 0$$

and our claim is satisfied. So we may now assume that $u_1 = m-1$.

When u_1 is even, we are searching for the last element of the list

$$u_1 \oplus \texttt{FWM(m,n-1,k-}u_1\texttt{)}.$$

Note that u_1 even implies that $m-1$ is even. In this case, we know that the second letter is $u_2 = \min\{m-1, k-(m-1)\}$. As before, if $u_2 = k-(m-1)$, then the only string possible is

$$(m-1)(k-m+1)0 \cdots 0,$$

which meets our requirements. So we assume that $u_2 = m-1$, which is even. So now we know that the last element of the list must be

$$u_1 \oplus (m-1)\cdots(m-1)r''0\cdots 0 = (m-1)\cdots(m-1)r'0\cdots 0,$$

where $0 \leq r'' < m-1$ and there is some q'' so that $k-2(m-1) = q''(m-1)+r''$.

When u_1 is odd, we are searching for the last element of the list

$$u_1 \oplus \mathrm{Rev}(\texttt{FWM(m,n-1,k-}u_1\texttt{)}),$$

which is the same as searching for the first element of the list

$$u_1 \oplus \texttt{FWM(m,n-1,k-}u_1\texttt{)}.$$

By the first part of the claim, this is

$$u_1 0\cdots 0r(m-1)\cdots(m-1),$$

where $k-u_1 = q(m-1)+r$ for $0 \leq r < m-1$. □

Corollary 3.3.4. *For all m, n, k, the first element of* `FWM(m,n,k)` *and the first element of* `FWM(m,n,k-1)` *differ in exactly one position.*

Proof. By Lemma 3.3.3, we have the first element of `FWM(m,n,k-1)` is

$$0\cdots 0r(m-1)\cdots(m-1),$$

with $0 \leq r < m-1$. Then the first element of `FWM(m,n,k)` must be

$$0\cdots 0(r+1)(m-1)\cdots(m-1),$$

where $1 \leq r+1 \leq m-1$. These only differ in one position. □

Corollary 3.3.5. *For all m, n, k, the last element of* `FWM(m,n,k)` *and the last element of* `FWM(m,n,k-1)` *differ in exactly one position.*

Proof. We proceed by induction on n. When $n = 1$, then the two strings must differ in exactly one position. We now assume that $n > 1$. Define $u_1 = \min\{m-1, k\}$. By Lemma 3.3.3, the last element of `FWM(m,n,k)` is

$$\mathbf{u} = \begin{cases} u_1(m-1)\cdots(m-1)r'0\cdots 0, & \text{if } u_1 \text{ is even, and} \\ u_1 0\cdots 0r'(m-1)\cdots(m-1), & \text{if } u_1 \text{ is odd.} \end{cases}$$

As before, if $u_1 = k$, then this string is $\mathbf{u} = k0\cdots 0$ and the last element of `FWM(m,n,k-1)` is $(k-1)0\cdots 0$, which clearly only differs in one position. If $u_1 = (m-1) \neq k$, then we consider the last element of `FWM(m,n,k-1)`, which we will call $\mathbf{v}$. In this case we must have $u_1 = v_1$, and so we consider the two substrings $u_2u_3\ldots u_n$ and $v_2v_3\ldots v_n$. If u_1 is even, then these are the last elements of the lists `FWM(m,n-1,k-`u_1`)` and `FWM(m,n-1,k-`u_1`-1)`, respectively. By the induction hypothesis these must differ in exactly one position, and we are done. If u_1 is odd, then our two substrings are the first elements of the lists `FWM(m,n-1,k-`u_1`)` and `FWM(m,n-1,k-`u_1`-1)`, which by Corollary 3.3.4 differ in exactly one position. □

We are now ready to prove that our algorithm is correct.

Proof of Result 3.3.2. We will show by induction that the algorithm provided produces the desired Gray code. For our base cases, when $n = 1$ the lists are easily constructed. If $m - 1 \geq k$, then we get the list $[k]$, otherwise we have an empty list.

For $n > 1$, suppose we want to construct the desired Gray code for $\mathcal{B}_k^m(n)$. We will show that `FWM(m,n,k)` is correct. By our induction hypothesis, for each i from 0 to $\min(m-1, k)$, our sublist M has adjacent elements differing in exactly two positions. All that remains is to check that this minimal change property is maintained as i increases.

First, when i increases from an odd to an even number, we have the transition:

$$\begin{aligned} i &\oplus \text{Rev}(\texttt{FWM(m,n-1,k-i)}) \\ (i+1) &\oplus \texttt{FWM(m,n-1,k-i-1)} \end{aligned}$$

By Corollary 3.3.4, the adjacent elements of these two sublists differ in exactly one position, which together with the leftmost position gives exactly two positions.

When i increases from an even number to an odd number, we have the transition:

$$\begin{aligned} i &\oplus \texttt{FWM(m,n-1,k-i)} \\ (i+1) &\oplus \text{Rev}(\texttt{FWM(m,n-1,k-i-1)}) \end{aligned}$$

Clearly the adjacent elements of these two sublists differ in the first coordinate, so we must check that they only differ in one other position at their meeting point - that is, that the last element of `FWM(m,n-1,k-i-1)` differs from the last element of `FWM(m,n-1,k-i)` in at most one position. To prove this, we use Corollary 3.3.5. □

Another restriction that we can consider for $\mathcal{B}^m(n)$ is when we restrict the multiset. In comparison, note that $\mathcal{B}_k^2(n)$ consists of all words with multiset $\{0^{n-k}, 1^k\}$. Thus from the example illustrated in Section 3.2 (Figure 3.2), we see that it is not always possible to find a Gray code listing for words from $\mathcal{B}^m(n)$ with fixed multiset when we restrict our listing to allow only adjacent transpositions. However, there does exist a Gray code for $\mathcal{B}_M(n)$ when we transpositions are not required to be adjacent.

Theorem 3.3.6. ([36], fasc. 2, p. 39) *Given a multiset M of size n with elements from $\{0, 1, \ldots, m-1\}$, there is a Gray code listing for the subset of $\mathcal{B}^m(n)$ with multiset M in which each word differs from its successor by the transposition of two elements.*

3.4 Permutations of $[n]$

Under the standard representation, we note that permutations of $[n]$ can be identified with n-ary words of length n with multiset $\{0, 1, \ldots, n-1\}$. From Figure 3.2, it may seem unreasonable to hope for a Gray code listing allowing transposition of adjacent elements. However, the following well-known result proves that this is possible.

Theorem 3.4.1. [32] *Gray codes for permutations exist when we allow transposition of adjacent elements.*

Theorem 3.4.1 is a direct corollary to the following well-known result.

Theorem 3.4.2. [32] *The set of all permutations of n distinct objects can be generated by allowing only transposition of adjacent elements.*

Proof. We prove this result using a recursive construction. When $n = 1$, there is only one permutation, and hence only one way to list it. For $n > 1$, suppose that we have a listing $L(n-1)$ of all permutations of $[n-1]$ so that successive elements differ by an adjacent transposition. For each permutation in $L(n-1)$, we will create n new permutations of $[n]$ using the following algorithm to create the new list $L(n)$.

```
For each permutation w in L(n-1) do
    If position of w on L(n-1) is odd then
        For i from n down to 1 do
            Insert n into w in position i
    If position of w on L(n-1) is even then
        For i from 1 up to n do
            Insert n into w in position i
```

First note that this produces every permutation of $[n]$. This is because for each permutation of $[n-1]$, the algorithm adds the letter n in every position possible.

To see that this only transposes adjacent elements, first consider the part of $L(n)$ that is created from one word w in $L(n-1)$. At each step in the given algorithm, we are either moving the letter n one position to the right or to the left, both of which can be done using an adjacent transposition. Next, consider a point where we shift from a word w to a word v in $L(n-1)$. The position of the letter n is the same in the last word created from w, say w^*, and in the first created from v, say v^*, and it is either in the first position or the last position. Thus since w and v differ by an adjacent transposition, so must w^* and v^*.

□

In [41], Nijenhuis and Wilf have several exercises asking for various Gray codes for different objects. The following theorem is the solution to one.

Theorem 3.4.3. [51] *There exists a Gray code for the set of permutations of* $[n]$ *in which an element is obtained from its predecessor by applying either* $\sigma = (1\,2\ldots n)$ *or* $\tau = (1\,2)$.

Beyond this, the other results in this area focus on restricted subsets of the set of all permutations. For example, we have the following theorems, which we will include without proof.

Theorem 3.4.4. [3] *There exists a cyclic Gray code listing for the set of derangements of* $[n]$ *so that successive elements differ in at most four positions.*

Theorem 3.4.5. [2] *There exists a cyclic Gray code listing for the set of permutations of* $[n]$ *with* k *cycles so that successive elements differ by a product with a three-cycle.*

Another way to restrict the class of permutations of $[n]$ is to consider only permutations that avoid some specified pattern. For example, a permutation that avoids the pattern 123 cannot have any increasing subsequence of length three. In [15], many results are given on various classes of pattern-avoiding permutations, such as permutations of $[n]$ that avoid the pattern 231.

One problem that does not appear to have been considered yet concerns permutations with fixed rank. The **rank** of a permutation is defined as the distance from the identity permutation in the Cayley graph with adjacent transpositions as generators. This subset of permutations leads us to several ope problems.

Open Problem 3.4.6. *What is a good minimal change property for the set of permutations with fixed rank?*

Open Problem 3.4.7. *Does there exist a Gray code listing for the set of permutations with fixed rank?*

In considering these open problems, another way to state the definition is that a permutation of rank k can be written as an ordered product of k generators. So perhaps a solution to the following problem could be helpful.

Open Problem 3.4.8. *Does there exist a Gray code listing for the set of permutations with fixed rank k when these permutations are written as a product of k generators?*

This leads us to consider alternate representations for permutations. We can represent a permutation in terms of its cycles. This is done by first representing a permutation as a product of disjoint cycles. Then we rotate each cycle to start with the largest element, and then order the cycles by the starting element, from smallest to largest. In this manner, we can write the permutation as a word. Wilf asks the following:

Open Problem 3.4.9. [50] *If we choose to represent permutations in cycle form, is there an interesting Gray code listing?*

We find the following Gray code using a straightforward bijection.

Result 3.4.10. *There exists a Gray code for permutations using the disjoint cycle representation in which consecutive strings differ by a transposition of adjacent elements.*

Proof. From Stanley [44], we know that there is a bijection between permutations in the functional notation and permutations in the disjoint cycle notation. This is obtained by mapping a permutation in disjoint cycle notation to permutation written as a word in functional notation as follows. See Figure 3.4 for an example of this mapping.

(123)	$\longrightarrow$	123
(132)	$\longrightarrow$	132
(2)(13)	$\longrightarrow$	213
(23)(1)	$\longrightarrow$	231
(3)(12)	$\longrightarrow$	312
(3)(2)(1)	$\longrightarrow$	321

Figure 3.4: Permutations of [3] using disjoint cycle representation and their corresponding strings

1. Rotate each cycle so that it starts with the least element.

2. Order the cycles in decreasing order by starting element.

3. Remove the parentheses.

This process can be easily reversed. By Theorem 3.4.1, we know that a Gray code for permutations using the functional notation exists when we allow transposition of adjacent elements. Thus using the same algorithm, we obtain a Gray code for permutations in disjoint cycle notation (written as a string, as shown above).

□

As an example, consider the permutations of [3]. In Figure 3.4, we list each permutation in disjoint cycle notation, and their corresponding strings.

3.5 Subsets

Consider the set of all subsets of $[n]$. We must first decide how we want to represent these subsets. If we are looking for a Gray code listing, it may be difficult to define minimal change if we list them directly, since the subsets

have cardinalities ranging from 0 (the empty set) to n. An alternative choice is to use the subset membership representation: Each subset is a binary word of length n with a 1 in position i if and only if i is a member of the subset. From this definition, we clearly have the following fact.

Fact 3.5.1. [50] *There is a bijection between the set of all binary words of length n and the set of all subsets of $[n]$ using the subset membership representation.*

Now that we have this fact, we consider the Gray codes for $\mathcal{B}^2(n)$ that were determined in Section 3.2. The first one discussed was the reflected binary code. In this cyclic Gray code the minimal change property ensures that each word differs from its successor in one position. Translating this back to subsets, this is equivalent to a cyclic Gray code for subsets in which each word is modified by either adding an element to or deleting an element from the current subset to obtain the next one in the list. This is summarized in the following corollary.

Corollary 3.5.2. *There is a cyclic Gray code for the set of subsets of $[n]$ in which a subset of $[n]$ is obtained from its predecessor by adding or removing an element.*

Next if we consider the revolving door code from Section 3.2, we focus on the set $\mathcal{B}_k^2(n)$. This set is the set of all k-subsets of $[n]$ under the correspondence discussed in Fact 3.5.1. Using this representation, the revolving door code gives us a cyclic Gray code for the set of k-subsets of $[n]$ in which each word is obtained from its predecessor by swapping out one set member, that is,

removing an element from the set and adding a new one to obtain a different k-subset of $[n]$. This is stated in the corollary below.

Corollary 3.5.3. *There is a cyclic Gray code for the set of k-subsets of $[n]$ in which a subset differs from its predecessor in exactly one element.*

We can also restrict the minimal change property even more. Suppose that we write all k-subsets of $[n]$ as strictly increasing vectors, that is we order each subset from smallest element to largest.

Theorem 3.5.4. [16] *If we write the k-subsets of $[n]$ as a strictly increasing vector, we can find a Gray code listing so that successive elements differ only in a single component.*

An alternative restriction is to limit what values are allowed to be interchanged.

Theorem 3.5.5. [50] *Suppose that the minimal change properties allowed for k-subsets of $[n]$ are for $i \in [n]$:*

1. *delete i and adjoin $i+1$, or*

2. *delete $i+1$ and adjoin i.*

There is a Gray code listing using this property if and only if either $k \in \{0, 1, n-1, n\}$, or if n is even and k is odd.

3.6 Weak Orders on $[n]$

Knuth proves the following theorem, which leads to Open Problem 3.6.2.

Theorem 3.6.1. ([36], fasc. 2, Problem 106, p. 119.) *For every n, there exists a Gray code for the weak orders on $[n]$ so that two consecutive objects differ by one of the two elementary operations $w_i \leftrightarrow w_j$ or $w_i \leftarrow w_j$.*

Open Problem 3.6.2. ([29], Problem 482.) *For which positive integers n does there exist a Gray code for the weak orders on $[n]$ allowing only elementary operations of the form $w_i \leftrightarrow w_{i+1}$ and $w_i \leftarrow w_{i+1}$?*

In [29], Knuth gives the following example when $n = 3$:

$$201, 210, 120, 102, 012, 021, 011, 101, 001, 010, 110, 100, 000.$$

When we consider only the elements of $\mathcal{W}(n)$ with maximum height, we are actually considering the set of all permutations of $[n]$. In this case, Theorem 3.4.1 from Section 3.4 tells us that we can in fact find a Gray code on the set of permutations so that consecutive objects differ only by transposition of adjacent elements.

A Gray code as in Open Problem 3.6.2 must list all elements of $\mathcal{W}(n, n-1)$ before any elements from $\mathcal{W}(n, n-2)$, which then must be before elements from $\mathcal{W}(n, n-3)$, and so on until the last element is from $\mathcal{W}(n, 0)$. To transition from a word $w \in \mathcal{W}(n, h)$ to some $u \in \mathcal{W}(n, h-1)$ with $0 \leq h < n$, we must copy over the letters with maximum height. After this copy operation, there is no way to recover the lost letter and return to listing weak orders containing letters of height h. This gives us the following result.

Result 3.6.3. *In a Gray code for weak orders using the operations $w_i \leftrightarrow w_{i+1}$ and $w_i \leftarrow w_{i+1}$, all words from $\mathcal{W}(n, h)$ must appear before all words from $\mathcal{W}(n, h-1)$, where $0 \leq h \leq n-1$ and $1 \leq i \leq n-1$.*

Because of this, if we can answer the following problem, then we might have an answer to Open Problem 3.6.2.

Open Problem 3.6.4. *For which positive integers n and h with $0 \leq h < n$ does there exist a Gray code for $\mathcal{W}(n,h)$ using the operations $w_i \leftrightarrow w_{i+1}$ and $w_i \leftarrow w_{i+1}$?*

If we can answer Open Problem 3.6.4 successfully, then we can hope to find some way to connect the lists that correspond to each set $\mathcal{W}(n,h)$ to answer Open Problem 3.6.2. For a few specific values of h, we know that we can find the Gray code desired in Open Problem 3.6.4. For example, when we look at the set $\mathcal{W}(n,1)$, we are examining the set of binary words minus the word 1^n. We define

$$\mathcal{B}^*(n) = \mathcal{B}(n) \setminus 1^n.$$

Open Problem 3.6.5. *Can we find a Gray code for the set $\mathcal{B}^*(n)$ using the operations $w_i \leftrightarrow w_{i+1}$ and $w_i \leftarrow w_{i+1}$?*

In working towards a solution to Open Problem 3.6.5, we have the following result.

Result 3.6.6. *There is a Gray code for $\mathcal{B}^*(n)$ using only the following operations:*

$$w_i \leftrightarrow w_{i+1},$$

$$w_i \leftrightarrow w_{i+2}, \text{ and}$$

$$w_i \leftarrow w_{i+1}.$$

To prove this, we use the following theorem.

Theorem 3.6.7. ([36], fasc. 3, p. 11.) *If $0 < k < n$, there is a Gray code listing for $\mathcal{B}_k(n)$ using only the operations $w_i \leftrightarrow w_{i+1}$ and $w_i \leftrightarrow w_{i+2}$. Furthermore, this listing can be chosen to begin with 1^k0^{n-k} and end with 01^k0^{n-k-1}.*

Proof of 3.6.6 . For each k, we define the listing from Theorem 3.6.7 as $L_k(n)$. Let $\overline{L_k(n)}$ denote this listing in reverse order. Then we claim that the following sequence produces the desired Gray code:

$$\begin{array}{c} \overline{L_{n-1}(n)} \\ L_{n-2}(n) \\ \overline{L_{n-3}(n)} \\ L_{n-4}(n) \\ \vdots \end{array}$$

This sequence looks like:

$$\begin{array}{ll} \overline{L_{n-1}(n)}: & 01^{n-1} \\ & \vdots \\ & 1^{n-1}0 \\ L_{n-2}(n): & 1^{n-2}0^2 \\ & \vdots \\ & 01^{n-2}0 \\ \overline{L_{n-3}(n)}: & 01^{n-3}0^2 \\ \vdots & \vdots \end{array}$$

It is clear that at each transition from one sublist to the next, only the copy operation $w_i \leftarrow w_{i+1}$ is used. All that remains is to check that we can use the copy operation to append 0^n to the end of our list. Depending on whether n

is odd or even, our listing will end with either $L_1(n)$ or $\overline{L_1(n)}$. If it ends with $L_1(n)$, then our list ends with 010^{n-2} and so we may copy over the sole 1 to obtain 0^n. If the listing ends with $\overline{L_1(n)}$, then it ends with 10^{n-1} and so again we may copy over the only 1 to obtain 0^n. Either way, we obtain a Gray code for $\mathcal{B}^*(n)$ using only the given operations. □

Result 3.6.8. *Let $M = \{0, 1, \ldots, n-1\}$. There exists a Gray code for the set $\mathcal{W}_M(n)$ using the operation $w_i \leftrightarrow w_{i+1}$.*

Proof. The set of weak orders on $[n]$ with multiset $\{0, 1, \ldots, n-1\}$ is the set of permutations of $\{0, 1, \ldots, n-1\}$, and so this result follows from Theorem 3.4.2. □

If we next consider the set of weak orders with fixed multiset, we note that it is not possible to find a Gray code using the operation $w_i \leftrightarrow w_{i+1}$. This is clear from the example in Figure 3.2. Note that in the case of fixed multiset, the operation $w_i \leftarrow w_{i+1}$ cannot be used.

Another option is to consider alternative representations. We may be able to make use of order isomorphic representations or equivalence class representations that will allow us more freedom with our elementary operations. With these alternative representations, we may not need to list all elements with bounded height together.

With Open Problem 3.6.2, Knuth also asked the following.

Open Problem 3.6.9. ([29], Problem 482.) *For which positive integers n does there exist a Gray code for the weak orders on $[n]$ where two consecutive objects always differ in exactly one position?*

3.7 Partitions

3.7.1 Partitions of an Integer

When we consider partitions of an integer n, we must first determine what minimal change property to utilize in creating a Gray code. Note that we cannot only change one part to move up in the list, as this would change the sum of the parts from n. Thus we first consider changing two parts as little as possible to obtain the following theorem.

Theorem 3.7.1. [43] *Fix $n \in \mathbb{Z}^+$. There is a Gray code listing for the partitions of n in which a partition differs from its predecessor in two parts: one part increased by one and one part decreased by one.*

One way to restrict the set of partitions of an integer is to set an upper bound on the size of the parts.

Theorem 3.7.2. [43] *For all $n \geq k \geq 1$, there is a way to list all partitions of n into integers of size at most k so that each partition differs from its predecssor in two parts: one part increased by one and one part decreased by one. One can also create this Gray code so that, unless $n = 6$ and $k = 4$, it starts with the lexicographically smallest word and ends with the largest.*

The proof of this result includes the recursive construction of the Gray code broken down into many cases, with several cases written out explicitly. Note that this result also proves the existence claimed in Theorem 3.7.1 by considering the case when we set $k = n$.

Another restriction to consider is the set of **Fibonacci sequences**. These are the ordered partitions of an integer n into parts of size one, two, and zero. This set is closely related to the set $\hat{B}_2(n)$, as discussed following Result 3.2.9.

Result 3.7.3. *There is a Gray code listing for the set of Fibonacci sequences in which a partition differs from its predecessor in two parts: one part increased by one and one part decreased by one.*

Proof. This is direct from Theorem 3.7.2 with $k = 2$. □

For example, we may consider $n = 4$, in which case both $1+1+1+1$ and $2+2$ are partitions. However, we must view the latter sequence as $2+2+0+0$. Then we can move from $2+2$ to $1+1+1+1$ via the following listing when we allow two parts to change: one increasing by one and the other decreasing by one.

$$1+1+1+1$$

$$2+1+1+0$$

$$2+2+0+0$$

3.7.2 Ordered Partitions of a Set

An ordered partition $\mathcal{P} = \{P_0, P_1, \ldots, P_k\}$ of an n-set can be defined as follows. First assume that none of the P_i's are empty. Identify the n-set with the set $[n]$. Label the partitions by their index. Then we define a word

$w = w_1 w_2 \dots w_n$ to correspond to the ordered partition by setting $w_i = j$ if and only if $i \in P_j$.

Theorem 3.7.4. ([29], Problem 482.) *There is a bijection between the set of ordered partitions of* $[n]$ *and the set of weak orders on* $[n]$.

Proof. Using the word representation given above for an ordered partition, we see that every ordered partition corresponds to exactly one weak order, so the map is well-defined. Note that the assumption of no empty parts ensures that every integer from 0 to k appears in the word, and hence the word is a weak order.

Next, given a weak order we can reverse the map to define a partition of $[n]$ that maps to the weak order. Note that there is one and only one partition that maps to each weak order, and so the map is one-to-one and onto. □

Given the previous result, we can find Gray codes for ordered partitions whenever we can find Gray codes for the corresponding weak orders. Thus, we get the following corollary to a theorem from Section 3.6.

Result 3.7.5. *For every* n*, one can list the ordered partitions of* $[n]$ *so that two consecutive words differ by one of the two elementary operations* $w_i \leftrightarrow w_j$ *or* $w_i \leftarrow w_j$.

Proof. This is a direct corollary to Theorem 3.6.1. □

3.7.3 Unordered Partitions of a Set

We now consider when the partition is unordered. Some identify this set of partitions as the set of distinct ways to place n labeled balls into unlabeled boxes.

Theorem 3.7.6. [35] *It is possible to arrange the set of all partitions of an n-set in a list so that each partition is obtained from its immediate predecessor by changing the class of exactly one element.*

This theorem has several proofs by construction. We will include one below, given by Knuth but unpublished, and summarized in [35].

Proof. The construction is recursive. Suppose that we have the correct Gray code listing for the set of all partitions of $[n-1]$, call it $L(n-1) = (L_1, L_2, \ldots)$. For any partition L_i of $[n-1]$, define the **children** of L_i to be any partition of $[n]$ obtained from L_i by either adding n to an existing part, or adding n as a singleton part to L_i. Order the children of L_i in a list $L_i(n) = (L_i^1, L_i^2, \ldots)$ where L_i^j is obtained from L_i by adding n to the jth part of L_i (putting the partition with n as a singleton last). Let $\overline{L_i(n)}$ denote the list of children from L_i reversed. Then we obtain the Gray code listing for partitions of $[n]$ as follows:

$$L(n) = \begin{bmatrix} L_1(n) \\ \overline{L_2(n)} \\ L_3(n) \\ \overline{L_4(n)} \\ \vdots \end{bmatrix}$$

It is clear that if $L(n-1)$ is correct, then so is this construction of $L(n)$. This is because, as in the proof of Theorem 3.2.1, the only points to check are where the sublists meet. However, at these points, the position of n is held constant since the direction of the sublists alternate depending on parity. Thus the desired property is preserved because the remaining positions of $[n-1]$ are determined by $L(n-1)$.

All that remains is to provide a base case. It is clear that when $n = 1$, there is only one possible partition, $\{1\}$. □

For an example, consider when $n = 4$. A valid ordering for $n = 3$ is as follows:

$$(123)$$

$$(12)(3)$$

$$(1)(2)(3)$$

$$(1)(23)$$

$$(13)(2)$$

First we must consider the children of each partition listed. These are given in the following chart.

	Children
(123)	$(1234), (123)(4)$
$(12)(3)$	$(124)(3), (12)(34), (12)(3)(4)$
$(1)(2)(3)$	$(14)(2)(3), (1)(24)(3), (1)(2)(34), (1)(2)(3)(4)$
$(1)(23)$	$(14)(23), (1)(234), (1)(23)(4)$
$(13)(2)$	$(134)(2), (13)(24), (13)(2)(4)$

(1234)
(123)(4)
(12)(3)(4)
(12)(34)
(124)(3)
(14)(2)(3)
(1)(24)(3)
(1)(2)(34)
(1)(2)(3)(4)
(1)(23)(4)
(1)(234)
(14)(23)
(134)(2)
(13)(24)
(13)(2)(4)

Figure 3.5: Gray code for unordered partitions of $\{1, 2, 3, 4\}$

Alternating lists of children forwards and backwards, we end up with the Gray code in Figure 3.5.

Open Problem 3.7.7. *Is there a Gray code listing for the set of all partitions of an n-set with part size at most k?*

3.8 Designs

Given the definition of a transversal design in Section 2.2, we know that in any TD(k, n), every pair of points from distinct groups is contained in exactly one block. Thus we can represent each block by a pair of points.

Fact 3.8.1. *There is a basis representation for any TD(k, n) such that each block is represented by a string of length 2.*

In order to more specifically select which points we will use to represent each block, we use the following theorem.

Theorem 3.8.2. ([46], p. 146.) *Suppose that $n \geq 2$ and $k \geq 3$. Then there exists a set of $k-2$ MOLS of side n if and only if there exists a TD(k, n).*

While the proof of this result is not difficult, we will be mostly interested in the construction of a TD(k, n) from a set of $k-2$ MOLS of side n, labeled arbitrarily as $L_1, L_2, \ldots, L_{k-2}$. The TD$(k, n)$ blocks are defined as follows. Fix $(i, j) \in [n] \times [n]$. Let $L_h(i, j)$ denote the entry in row i, column j, in L_h. Define the blocks of our TD(k, n) to be $\{i, j, L_1(i, j), L_2(i, j), \ldots, L_{k-2}(i, j)\}$, with the first element in group G_1, the second in group G_2, and so on until the last element is in group G_k.

From this construction, we see that each block is completely defined by the two elements, i and j in groups G_1 and G_2. If we use this as our basis representation, it is clear that every pair (i, j) must appear, i.e. the set of blocks using the basis representation corresponds to the set of all n-ary words of length 2. This is summarized in the following fact.

Fact 3.8.3. *There is a direct correspondence between the blocks of a TD(k, n) and the set of all n-ary words of length 2.*

Using this representation we can find a Gray code for any TD(k, n) whenever there exists a Gray code for n-ary words of length 2. Thus we get the following corollary.

Result 3.8.4. *For every $n \in \mathbb{Z}^+$, there exists a Gray code listing for the blocks of any TD(k, n) using the basis representation so that each word differs from its successor in exactly one index position.*

Proof. Under the correspondence from Fact 3.8.3, this is a direct corollary to Theorem 3.3.1. □

It is important to note that this representation of a TD(k, n) is not unique, and so several transversal designs could have the same representation, and hence the same Gray code.

Similar to Theorem 3.8.2, we can relate MOLS and BIBDs.

Theorem 3.8.5. [46], p. 139. *Let $n \geq 2$. Then the existence of any one of the following designs implies the existence of the other two designs.*

- *$n - 1$ MOLS of side n.*
- *a projective plane of order n, or an $(n^2 + n + 1, n + 1, 1)$-BIBD.*
- *an affine plane of order n, or an $(n^2, n, 1)$-BIBD.*

In particular, this theorem together with Theorem 3.8.2 implies that from either a $(n^2 + n + 1, n + 1, 1)$-BIBD or a $(n^2, n, 1)$-BIBD, we can construct a TD$(n + 1, n)$, and vice versa. This suggests that there may be some way to construct a Gray code for projective and affine planes from the corresponding transversal designs.

Open Problem 3.8.6. *Is there a Gray code for projective planes or affine planes?*

To consider more general BIBDs, for any (v, k, λ)-BIBD, define a **κ-intersecting Gray code** for $0 < \kappa < k$ as a listing of the blocks of the design such

that consecutive blocks intersect in exactly κ points. For example, the Gray code that we constructed in Result 3.8.4 is a 1-intersecting Gray code. Finding κ-intersecting Gray codes is similar to (and in some cases exactly the same as) finding Hamilton cycles in the block intersection graph of a design. The **block intersection graph** of a design represents blocks as vertices and draws an edge between blocks if and only if the blocks share a common element. In the case of $(v, k, 1)$-BIBDs, a Hamilton cycle in the block intersection corresponds to a 1-intersecting Gray code. For more on block intersection graphs, see [1, 24, 42], among others.

Dewar proves the following theorem in her dissertation, and poses the question that follows.

Theorem 3.8.7. [13] *With "sporadic" exceptions, for* $v \equiv 1, 3, 4, 7 \pmod{12}$*, there is a TTS*$(v)$ *that admits a 2-intersecting Gray cycle for its blocks.*

Open Problem 3.8.8. [13] *Does there exist a TTS*(v) *with* $v \equiv 0, 6, 9 \pmod{12}$ *that admits a 2-intersecting Gray cycle?*

In her dissertation, Dewar uses a construction that involves cyclic designs and difference sets, which we will not discuss. We would like to consider Open Problem 3.8.10 in the hopes of a simpler, or at least different, construction. First, we define the direct product for Steiner triple systems.

Construction 3.8.9. (Direct Product, [10], p. 39.) *Let* $v, w \in \mathbb{Z}^+$*. If there exists an STS*(v) *and an STS*(w)*, then there exists an STS*(vw)*.*

Construction. Suppose that $(X, \mathcal{A})$ is an STS(u) and $(Y, \mathcal{B})$ is an STS(v). We identify X with $\mathbb{Z}_u$ and Y with $\mathbb{Z}_v$. Then we construct a new STS$(uv) = (Z, \mathcal{C})$

with points:

$$Z = \mathbb{Z}_u \times \mathbb{Z}_v$$

and blocks:

1. $\{(i,a),(i,b),(i,c)\}$ with $i \in \mathbb{Z}_u$ and $\{a,b,c\} \in \mathcal{B}$,

2. $\{(i,a),(j,a),(k,a)\}$ with $\{i,j,k\} \in \mathcal{A}$ and $a \in \mathbb{Z}_v$, and

3. $\{(i,a),(j,b),(k,c)\}$ with $\{i,j,k\} \in \mathcal{A}$ and $\{a,b,c\} \in \mathcal{B}$.

□

Open Problem 3.8.10. *Suppose that the design $(X, \mathcal{B})$ is the direct product of two Steiner triple systems that admit Gray codes. Does $(X, \mathcal{B})$ also admit a Gray code under the same (or a similar) minimal change property?*

Define a ***t*-swap Gray code** on a design $(X, \mathcal{B})$ to be an ordering of the blocks in $\mathcal{B}$ as $B_1, B_2, \ldots, B_n$ in which $|B_i \cap \overline{B_{i+1}}| \geq t$.

Lemma 3.8.11. *Given a design $(X, \mathcal{B})$ with block size k, if an s-overlap cycle exists for $(X, \mathcal{B})$, then $(X, \mathcal{B})$ has a $(k-s)$-swap cyclic Gray code.*

Proof. An s-overlap cycle on $(X, \mathcal{B})$ requires consecutive blocks to share s elements. Since each block contains k points, and any two adjacent blocks in the overlap cycle share at least s points, they can differ by at most $k - s$ points. Thus, listing the blocks from $\mathcal{B}$ in the order that they appear in the s-overlap cycle gives us a cyclic Gray code on $(X, \mathcal{B})$. □

Chapter 4

UNIVERSAL CYCLES

4.1 Introduction and Definitions

A **universal cycle** or **ucycle** over a set of combinatorial objects $\mathcal{C}$ is defined as a cyclic sequence of letters in which every object in $\mathcal{C}$ is represented once and only once in a consecutive manner. For example, consider the set of weak orders of [3]. This set consists of the following objects:

$$\mathcal{W}(3) = \left\{ \begin{array}{llllllll} 000, & 001, & 011, & 010, & 100, & 101, & 110, \\ 021, & 102, & 012, & 120, & 201, & 210 & \end{array} \right\}$$

A ucycle on $\mathcal{W}(3)$ is 0001101201021.

4.2 Binary Words of Length n

Universal cycles on $\mathcal{B}(n)$ are also referred to as **binary de Bruijn cycles**. The first main result in this area has been proven by many.

Theorem 4.2.1. [12, 20, 39] *For all $n \in \mathbb{Z}^+$ there exists a universal cycle on $\mathcal{B}(n)$.*

Proof. We construct the transition graph $G(n)$ with the elements of $\mathcal{B}(n)$ represented as edges, and show an Euler tour in the graph. To do this, we let the vertex set of $G(n)$ equal $\mathcal{B}(n-1)$. Then we add a directed edge from one

vertex v to another w if the last $n-2$ letters of v are equal to the first $n-2$ letters of w.

From the definition of $G(n)$, at each vertex $w_1w_2\ldots w_{n-1}$ we can pair each outgoing edge $w_1w_2\ldots w_n$ with an incoming edge $w_nw_1w_2\ldots w_{n-1}$. From this, it is clear that the indegree of each vertex is equal to its outdegree.

Next, we show a path from any arbitrary vertex to the vertex $00\cdots0$. This proves that the graph is weakly connected. Consider an arbitrary vertex $w_1w_2\ldots w_{n-1}$. There is a path from this vertex to $00\cdots0$ as shown below:

$$w_1\ldots w_{n-1}\to w_2\ldots w_{n-1}0\to w_3\ldots w_{n-1}00\to\cdots\to w_{n-1}0\cdots0\to00\cdots0.$$

Since the graph is balanced and weakly connected, it is eulerian by Theorem 2.1.2. The Euler tour corresponds to a universal cycle on $\mathcal{B}(n)$. □

We now restrict our attention to fixed weight binary words. Note that if we use the standard word representation for elements of $\mathcal{B}_k(n)$, the transition graph will consist of disconnected cycles. The only words that can be reached from a given word are rotations, i.e. we find cycles of the form shown in Figure 4.1. To cope with this, we must consider some alternative representation, so we begin with the following result.

Result 4.2.2. *Let $n\in\mathbb{Z}^+$, and let M be some fixed multiset of size n. Define the set A to be the set of all permutations of M. Then there exists a ucycle for A using the prefix representation.*

Proof. Construct a graph G_M with

$$V(G_M)=\{v=a_1a_2\ldots a_{n-2}\mid v=a^-\text{ for some }a\in A^-\}$$

and

$$E(G_M) = \{(u, v) \mid u = a^- \text{ and } v = a^+ \text{ for some } a \in A^-\}.$$

Note that the edges in this graph correspond to the elements of A^-, so we would like to find an Euler tour in G_M, which will produce a ucycle as desired.

For any vertex $w_1 \dots w_{n-4} w_{n-3} w_{n-2}$ corresponding to word $w = w_1 \dots w_n$, there is an undirected path to the vertex $w_1 \dots w_{n-4} w_{n-2} w_{n-3}$. This path is:

$$\begin{aligned} w_1 \dots w_{n-4} w_{n-3} w_{n-2} &\leftarrow w_n w_1 \dots w_{n-4} w_{n-3} \\ &\leftarrow w_{n-1} w_n w_1 \dots w_{n-4} \\ &\rightarrow w_n w_1 \dots w_{n-4} w_{n-2} \\ &\rightarrow w_1 \dots w_{n-4} w_{n-2} w_{n-3}. \end{aligned}$$

Thus we can always find an undirected path from one vertex to another whose difference is the transposition of elements $n-1$ and $n-2$. Since we also have paths from a vertex $v = w^{--}$ to $\rho(v) = \rho(w)^{--}$ (where ρ is the previously defined rotation function), we can find an undirected path between any two vertices that differ by adjacent transpositions. Then, since the set of all adjacent transpositions generate all permutations of a set (see [32]), there exists an undirected path between any vertices with the same multiset, so the graph G_M is weakly connected.

Note that the graph G_M must also be balanced, since having fixed multiset ensures that any edge

$$w_1 \dots w_{n-2} \rightarrow w_2 \dots w_{n-1}$$

can be balanced by the edge

$$w_{n-1} w_1 \dots w_{n-3} \rightarrow w_1 \dots w_{n-2}.$$

Thus the graph is balanced and connected, and so is eulerian by Theorem 2.1.2.

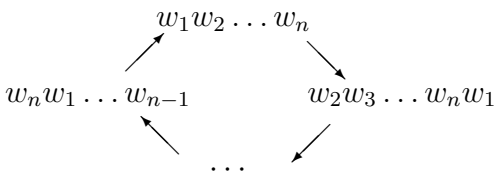

Figure 4.1: Cycles found in the transition graph for $\mathcal{B}_k(n)$

□

Corollary 4.2.3. *There is a universal cycle on $\mathcal{B}_k(n)$ using the prefix representation.*

Proof. The set $\mathcal{B}_k(n)$ can be viewed as the set of all words with multiset $\{0^{n-k}, 1^k\}$. Applying Result 4.2.2, we know that a universal cycle on this set exists using the prefix representation. □

Another alternative is to consider binary n-tuples with weights having the same parity. Let $\mathcal{B}_o(n)$ denote the subset of $\mathcal{B}(n)$ in which all words have odd weight, and let $\mathcal{B}_e(n)$ denote the subset of $\mathcal{B}(n)$ in which all words have even weight.

Result 4.2.4. *The sets $\mathcal{B}_o(n)$ and $\mathcal{B}_e(n)$ admit universal cycles using the prefix representation.*

Proof. First, note that it is valid to apply the prefix representation to these sets. Depending on the parity of the prefix of the n-tuple, the nth bit will be either 0 or 1 if the prefix parity either does or does not match the set considered, respectively.

Next, note that we must have $\mathcal{B}_o^-(n) = \mathcal{B}(n-1) = \mathcal{B}_e^-(n)$. This is straightforward, for if we consider any word in $\mathcal{B}(n-1)$, it either has the correct

weight parity (and so we add 0 to the end of the word), or it does not (and so we add 1 to the end of the word).

Finally, since we have already seen that $\mathcal{B}(n-1)$ must allow a ucycle in Theorem 4.2.1, both $\mathcal{B}_o^-(n)$ and $\mathcal{B}_e^-(n)$ also admit ucycles. □

As another possible restriction to the set of all binary strings, we would like to consider the following question. Note that we saw the Gray code version in Theorem 3.2.10.

Open Problem 4.2.5. *Is there a universal cycle on the set of words of length $2n$ of balanced parentheses using some representation?*

Note that we must use an alternative representation, as otherwise we would arrive at a word starting with a right parenthesis, which is invalid. One choice for an alternative representation is to assign 0 to a left parenthesis and 1 to a right parenthesis, or vice versa. Note that if this assignment is constant over the entire set of words, we will have the same problem with words starting with the representative for a right parenthesis. Thus we consider an assignment to be made for each word individually. However even with this representation we run in to trouble, as now we are considering a specific subset of fixed weight binary strings, which we have shown in Figure 3.2 do not allow ucycles without changing the representation. For this problem of balanced parentheses represented as binary strings, we have yet to find a representation that works. Fortunately, there are many representations for strings of balanced parentheses (see [44]).

4.3 m-ary Words of Length n

When we extend our alphabet from size two to size m, have a similar theorem to Theorem 4.2.1. We define an **m-ary de Bruijn cycle of order n** to be a universal cycle over the set of m-ary strings of length n.

Theorem 4.3.1. [12, 20, 39] *De Bruijn cycles exist for any m and any n.*

Proof. We can prove this result in much the same way as Theorem 4.2.1 in Section 4.2. We construct the transition graph $G(n)$ with the elements of $\mathcal{B}^m(n)$ represented as edges and show that an Euler tour must exist in the graph.

Define the transition graph $G(n)$ with vertex set equal to $\mathcal{B}^m(n-1)$, and draw an edge from vertex $w_1w_2\dots w_{n-1}$ to $v_1v_2\dots v_{n-1}$ if $w_{i+1} = v_i$ for all $i \in [n-2]$. Then an Euler tour in this graph will correspond to a universal cycle on the set $\mathcal{B}^m(n)$.

First we must show that the graph is balanced. This is clear, since at vertex $w_1w_2\dots w_{n-1}$, we can pair together every outgoing edge $w_1w_2\dots w_{n-1}w_n$ with incoming edge $w_nw_1w_2\dots w_{n-1}$.

Next we show that the graph is weakly connected. This again is clear, since there must be a path from any vertex $w_1w_2\dots w_{n-1}$ to the vertex 0^{n-1}

by following the path

$$\begin{aligned} w_1w_2\ldots w_{n-1} &\rightarrow w_2w_3\ldots w_{n-1}0 \\ &\rightarrow w_3w_4\ldots w_{n-1}0^2 \\ &\vdots \\ &\rightarrow w_{n-1}0^{n-2} \\ &\rightarrow 0^{n-1}. \end{aligned}$$

Thus since the graph is balanced and weakly connected, we may apply Theorem 2.1.2 to see that $G(n)$ must be eulerian. □

There are also many interesting algorithms for generating universal cycles of m-ary words of length k. For example, in [18], Fredricksen and Maiorana were able to create an algorithm that generated a list of necklaces in decreasing order, and then use this list to generate the m-ary de Bruijn sequence for words of length n. For the definition of necklaces, see Section 3.2.

When we consider restricting the set of m-ary words, it is a natural extension from the work on binary words to consider m-ary words of fixed weight.

Open Problem 4.3.2. *Is there a universal cycle on the set of fixed weight words?*

In Section 4.6 we consider a similar question for fixed weight weak orders. It may be possible that some of the results in Section 4.6 can be transformed to help solve Open Problem 4.3.2. There may be some way to apply the ideas and results that we have found in Section 4.6 to help us answer this question. One other idea is to create equivalence classes on the set $\mathcal{B}_k^m(n)$ that can each be represented by a unique weak order. For example, on $\mathcal{B}_8^4(3)$ we might have that 044 and 233 both correspond to the weak order 011.

4.4 Permutations

Universal cycles for permutations were first introduced by Chung, Diaconis, and Graham, in [7]. They unsuccessfully searched for a method to find such a ucycle for permutations using order isomorphic representations and allowing more than n symbols in the alphabet used for the words representing permutations of $[n]$. Note that permutations listed in our standard representation give the same complications as fixed weight binary words - only rotations can be obtained, leading to a transition graph consisting of disconnected cycles. The following conjecture was put forth.

Conjecture 4.4.1. [7] *For $n \geq 3$, only $n+1$ symbols are needed to create a universal cycle on the set of all permutations of $[n]$.*

In 1996, Hurlbert and Isaak answered this conjecture using an equivalence class representation.

Theorem 4.4.2. [28] *There exists a complete family of equivalence class universal cycles for permutations of $\{1, 2, \ldots, n\}$ using the symbols $\{0, 1, 2, \ldots, n\}$.*

Hurlbert and Isaak proved the previous result by examining the equivalence classes under a specific relation on the set of permutations of $\{0, 1, \ldots, n\}$. They first showed that a universal cycle exists for a carefully identified set of representatives for the equivalence classes, and then showed that this ucycle can be "lifted" back to a family of universal cycles in which each equivalence class is represented exactly once. Each class member is used as an equivalence class representative in exactly one of the cycles in the family.

Some of the techniques used to prove Theorem 4.4.2 may be useful in other unanswered questions. For example, in Open Problem 4.3.2 we will need to utilize some sort of equivalence class representation.

In 2009, Johnson used a different technique to answer Conjecture 4.4.1 using order isomorphic representations. Again, the ideas developed in this paper may be useful in solving some of our questions that cannot be solved in a more straightforward manner.

Theorem 4.4.3. [31] *Universal cycles exist for the set of permutations of* $\{1, 2, \ldots, n\}$ *using order isomorphic representations with an alphabet of* $n + 1$ *characters.*

Johnson proved his result inductively. He began by partitioning the transition graph containing vertices corresponding to n-permutations of $\{0, 1, \ldots, n\}$ into many short cycles that are simply cycles created by applying the rotation function to one string repeatedly. He was able to increase these short cycles by linking together pairs that exhibit similar strings. He continued to enlarge the cycles until he finds one of length $n!$ and each permutation of $[n]$ is order isomorphic to some vertex in the cycle. At this point, he found a universal cycle.

Note that given our results in Section 4.6 on weak orders on $[n]$, we can prove the existence of universal cycles for permutations of $[n]$ easily using the prefix representation.

Theorem 4.4.4. *Universal cycles exist for the set of permutations of* $[n]$ *using prefix representations.*

Proof. Note that we can consider permutations on the set $\{0, 1, \ldots, n-1\}$

instead of $[n]$. Then each permutation can be viewed as a weak order of $[n]$ with multiset $\{0, 1, \ldots, n-1\}$. It is clear then that the set of weak orders with fixed multiset $\{0, 1, \ldots, n-1\}$ is the same as the set of permutations on $\{0, 1, \ldots, n-1\}$. Then by Result 4.2.2 in Section 4.6, we know that this set allows a universal cycle using the prefix representation. □

So far, the only result found that restricts the set of permutations considers the set of k-permutations of $[n]$.

Theorem 4.4.5. [30] *For every $k \geq 3$ and $n \geq k+1$, there is a universal cycle on the set of k-permutations of $[n]$.*

Since no other results have been found, we pose the following question.

Open Problem 4.4.6. *What sets of restricted permutations of $[n]$ allow universal cycles?*

4.5 Subsets

Universal cycles for k-subsets of $[n]$ appear to be more difficult to find than universal cycles for other combinatorial structures. One reason for this is the idea that subsets are unordered sets, whereas all other objects that we have considered are ordered sets. If we assume that our vertices represent strings (i.e. the letters are ordered), each subset of size k would have $k!$ different vertices associated with it. Then a Hamilton cycle in the transition graph does not correspond to a universal cycle. We must determine which ordering of the k-subset to use in the universal cycle.

Theorem 4.5.1. [7] *If a ucycle for k-subsets of $[n]$ exists for $n, k \in \mathbb{Z}^+$, then k must divide $\binom{n-1}{k-1}$.*

Proof. Consider an arbitrary element $x \in [n]$. If a universal cycle for k-subsets of $[n]$ exists, then each occurrence of x in the ucycle is used to represent exactly k of the $\binom{n}{k}$ subsets. Note that there are exactly $\binom{n-1}{k-1}$ subsets that contain x. Thus the total number of occurrences of x in the ucycle is $\binom{n-1}{k-1}$ divided by k. Since we are assuming the ucycle to exist, this number must be an integer. Thus, we have $k \mid \binom{n-1}{k-1}$. □

From this necessary condition, the following conjecture was formed.

Conjecture 4.5.2. [7] *Given $k \in \mathbb{Z}^+$, there is some $n_0(k) \in \mathbb{Z}^+$ so that for all $n \geq n_0(k)$, a universal cycle for k-subsets of $[n]$ exists, provided k divides $\binom{n-1}{k-1}$.*

Progress on this conjecture has been slow. In 1993, Jackson proved the conjecture for $k = 3$ or 4 with n relatively prime to k [30]. In 1994, Hurlbert was able to prove that it holds for $k = 3, 4$ or 6 with n relatively prime to k [26]. However, beyond this, the conjecture remains open.

Nonexistence results in this area are scarce. One of the few results is as follows.

Theorem 4.5.3. [45] *For $n \geq 4$, no universal cycles for $(n-2)$-subsets of $[n]$ exist.*

If we consider the subset membership representation instead of the standard subset representation, then the set of all k-subsets of $[n]$ corresponds to the set $\mathcal{B}_k(n)$, the set of all binary words of length n and weight k.

Corollary 4.5.4. *There is a universal cycle for the set of k-subsets of $[n]$ using the prefix representation of the subset membership representation.*

Proof. This is direct from Corollary 4.2.3 in Section 4.2. □

If we also consider the set of *all* subsets of $[n]$ using the subset membership representation, we are now looking at the set $\mathcal{B}(n)$. We saw in Section 4.2 that this too admits a universal cycle for all $n \in \mathbb{Z}^+$.

Corollary 4.5.5. *There is a universal cycle for the set of subsets of $[n]$ using the subset membership representation.*

Proof. This is a direct corollary from Theorem 4.2.1 in Section 4.2. □

Recently, Blanca and Godbole in [5] have produced some new results in this area, one of which is the following.

Theorem 4.5.6. [5] *Let A be the set of all subsets of $[n]$ with size between p and q for $p < q$. Let A' be the set of strings representing A using the subset membership representation. There exists a univeral cycle for A'.*

4.6 Weak Orders

As no work appears to have been done in this area, we begin with a straightforward question.

Question 4.6.1. *Is there a universal cycle for $\mathcal{W}(n)$?*

For example, one ucycle for $\mathcal{W}(3)$ is:

0001101201021.

Let $w = w_1 w_2 \dots w_n \in \mathcal{W}(n)$. Define the **rotation function** ρ from $\mathcal{W}(n)$ to itself by $\rho(w) = w_2 w_3 \dots w_n w_1$.

Fix $n \in \mathbb{Z}^+$ and define the graph $G(n)$ as follows:

$$V(G(n)) = \{v = v_1 \dots v_{n-1} \mid v = w^- \text{ for some } w \in \mathcal{W}(n)\}$$

and

$$E(G(n)) = \{(v_1, v_2) \mid v_1 = w^- \text{ and } v_2 = w^+ \text{ for some } w \in \mathcal{W}(n)\}$$

Note that the edge set is well defined, since any prefix w^- of some $w \in \mathcal{W}(n)$ is also a suffix of the word $w_n w_1 w_2 \dots w_{n-1} \in \mathcal{W}(n)$.

The first result answers Question 4.6.1.

Result 4.6.2. *For all $n \in \mathbb{Z}^+$, there exists a ucycle on $\mathcal{W}(n)$.*

Proof. For $n = 1$, we get the universal cycle 0. For $n \geq 2$, we show a method to explicitly construct a ucycle on $\mathcal{W}(n)$ using the graph $G(n)$.

By the construction of $G(n)$, it is clear that an Euler tour will correspond to a ucycle. Using Theorem 2.1.2 from Section 2.1, we need only show that the graph is weakly connected and balanced and we are done. The graph is clearly balanced, since any incoming edge $(w_1 \dots w_{n-1}, w_2 \dots w_n)$ can be paired with the outgoing edge $(w_2 \dots w_n, w_3 \dots w_n w_1)$ at the vertex $w_2 \dots w_n$.

Next, we will show a path from any vertex $w_1 \dots w_{n-1}$ to the vertex $00 \dots 0$. We may apply the rotation function ρ to w, where $w = w_1 \dots w_n \in \mathcal{W}(n)$, as many times as necessary until we arrive at some $u = u_1 u_2 \dots u_n \in \mathcal{W}(n)$ with u_n equal to the height of w. Then u is represented in $G(n)$ as the edge $(u_1 \dots u_{n-1}, u_2 \dots u_n)$. Since u_n is a maximum letter in w, we must also have $u_1 \dots u_{n-1} 0 \in \mathcal{W}(n)$, and so we have $(u_1 \dots u_{n-1}, u_2 \dots u_{n-1} 0) \in E(G(n))$.

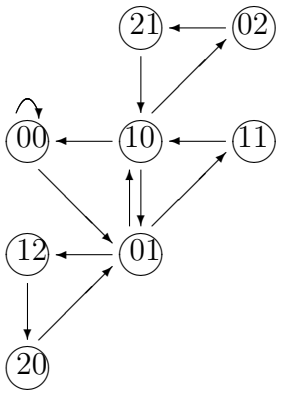

Figure 4.2: $G(3)$: The transition graph for ucycles for weak orders on [3]

Note that $u_2 \dots u_{n-1}0$ has more zeros than $w_1 \dots w_{n-1}$. Repeating this process, we add more zeros at every step, which eventually must terminate when we arrive at the vertex with $n-1$ zeros. Thus, we have a path from $w_1 \dots w_{n-1}$ to $00\dots0$, and so $G(n)$ is weakly connected.

□

Figure 4.2 shows an example of the graph $G(3)$. It produces the universal cycle 000120110102 for $n = 3$ (among others).

Next we consider some classes of restricted weak orders.

Question 4.6.3. *Is there a ucycle for $\mathcal{W}_k(n)$?*

To answer Question 4.6.3, we will construct a new transition graph, $G_k(n)$. Define

$$V(G_k(n)) = \{v = w_1 \dots w_{n-2} \mid v = w^- \text{ for some } w \in \mathcal{W}_k^-(n)\}$$

and

$$E(G_k(n)) = \{(v_1, v_2) \mid v_1 = w^- \text{ and } v_2 = w^+ \text{ for some } w \in \mathcal{W}_k^-(n)\}.$$

As before, this definition of edges is well-defined, for if $w_1 w_2 \dots w_{n-1} \in \mathcal{W}_k^-(n)$, then $w_{n-1} w_1 w_2 \dots w_{n-2} \in \mathcal{W}_k^-(n)$. Using this graph, we obtain the following result.

Result 4.6.4. *For all $n, k \in \mathbb{Z}^+$ with $n \geq 3$ and $k \leq \binom{n}{2}$, there exists a ucycle for $\mathcal{W}_k^-(n)$.*

Since the vertices in our transition graph $G_k(n)$ are words of length $n-2$, we identify the vertex v that is the minimum word in lexicographic order as the **minimum vertex**. It will be useful to determine exactly what word v is, and so we have the following fact, together with a lemma.

Fact 4.6.5. ([36], fasc. 3, p. 19.) *For every $k \in \mathbb{Z}^+$, there are unique $a, b \in \mathbb{N}$ with $a \geq b$ so that $k = \binom{a}{2} + \binom{b}{1}$.*

Lemma 4.6.6. *Fix $n, k \in \mathbb{Z}^+$. Define*

$$w = [0, b-1] b^2 [b+1, a]$$

with $k = \binom{a}{2} + \binom{b}{1}$ and $a \geq b$, and adding additional 0's to the front to create a word of length n. Then the minimum vertex v in $G_k(n)$ is $v = w^{--}$.

Proof. If we are looking for the element $w \in \mathcal{W}_k(n)$ with w^{--} minimum in lex order, we may consider only the elements of $\mathcal{W}_k(n)$ with the largest two elements in positions $n-1$ and n. These elements must be as large as possible in order to get the smallest elements possible in positions 1 to $n-2$, so the string we desire will have letters of the largest height possible. Note that if $\mathsf{ht}(w) = h$ and $w \in \mathcal{W}_k(n)$, then we must have $k \geq \binom{h+1}{2}$. Thus we choose w_n as large as possible so that $\binom{w_n+1}{2} \leq k < \binom{w_n+2}{2} = \binom{w_n+1}{2} + (w_n + 1)$.

Now, by our choice of w_n, the remaining weight after removing $0, 1, 2, \ldots, w_n$ is $0 \leq k - \binom{w_n+1}{2} < w_n + 1$. Thus $k - \binom{w_n+1}{2} \in \{0, 1, \ldots, w_n\}$, and so $k - \binom{w_n+1}{2}$ is equal to some element $b \leq w_n$. Then we can add another letter b to the string to obtain a word with a multiset of cardinality at most n. If the multiset contains less than n elements, we add 0's until it has size n. Sorting this multiset from smallest element to largest we obtain the desired weak order w. □

Before we begin the proof of Result 4.6.4, we define one more term. We say that there is a **duplicate at index i** if $w_i = w_{i+1} > 0$ for some string $w_1 w_2 \ldots w_n \in \mathcal{W}(n)$. Now we are finally ready to prove Result 4.6.4.

Proof of Result 4.6.4. First, note that if $k > \binom{n}{2}$, then it is not possible to construct a weak order on $[n]$ with weight k. The maximum weight weak order possible is one with multiset $\{0, 1, \ldots, n-1\}$, which must have weight

$$\sum_{i=0}^{n-1} i = \binom{n}{2}.$$

We would like to show an undirected path from any vertex $x^{--} = x_1 x_2 \ldots x_{n-2}$ to the minimum vertex $v^{--} = v_1 v_2 \ldots v_{n-2}$ in $G_k(n)$. Since $x \in \mathcal{W}_k(n)^{--}$, we may define x_{n-1}, x_n accordingly so that $x \in \mathcal{W}_k(n)$. By Result 4.2.2, we may assume that $x_1 \leq x_2 \leq \ldots \leq x_{n-2} \leq x_{n-1} \leq x_n$. Our first observation is that if x has either no duplicates or one duplicate then we must have $x = v$, and hence $x^{--} = v^{--}$. This follows from the fact that the minimum vertex writes weight k in the most compact way possible, i.e. with either zero or one duplicate. Now we assume that x has at least two duplicates, and we have several cases depending on the relationship between $\mathsf{ht}(v)$ and $\mathsf{ht}(x)$.

If $\mathsf{ht}(x) = \mathsf{ht}(v) = h$, then using Result 4.2.2 we may rewrite both weak orders as

$$x = [0, h]x_{h+2}x_{h+3}\dots x_n \text{ and } v = [0, h]v_{h+2}v_{h+3}\dots v_n$$

where $x_{h+2} \leq x_{h+3} \leq \cdots \leq x_n$ and $v_{h+2} \leq v_{h+3} \leq \cdots \leq v_n$. Now since $x \neq v$ there must be indices $i < j$ with $i, j \in \{h+2, h+3, \dots, n\}$ so that $x_i > v_i$ and $x_j < v_j$. Using Result 4.2.2 again, we can reorder the letters of x so that we have

$$x^{--} = x_i x_j [0, h]x_{h+2}x_{h+3}\dots x_{n-2} \text{ (with } x_i, x_j \text{ missing from the end).}$$

Then we have the undirected path:

$$\begin{aligned} x_i x_j [0, h]x_{h+2}x_{h+3}\dots x_{n-2} &\rightarrow x_j [0, h]x_{h+2}x_{h+3}\dots x_{n-2}x_{n-1} \\ &\rightarrow [0, h]x_{h+2}x_{h+3}\dots x_n \text{ (with } x_i, x_j \text{ missing)} \\ &\leftarrow (x_j + 1)[0, h]x_{h+2}x_{h+3}\dots x_{n-1} \\ &\leftarrow (x_i - 1)(x_j + 1)[0, h]x_{h+2}x_{h+3}\dots x_{n-2} \end{aligned}$$

Continuing in this manner, we will eventually arrive at a permutation of the vertex v^{--}, which then connects to v^{--} using Result 4.2.2.

If $\mathsf{ht}(x) < \mathsf{ht}(v)$, then we recall that x must have at least two duplicates, say at x_i and x_j. Using Result 4.2.2, rewrite x with x_i, x_j at the front, i.e. we now consider the vertex

$$x^{--} = x_i x_j x_1 x_2 \dots x_{n-2}.$$

Then we have the undirected path:

$$\begin{aligned} x_i x_j x_1 x_2 \dots x_{n-2} &\rightarrow x_j x_1 x_2 \dots x_{n-2} x_{n-1} \\ &\rightarrow x_1 x_2 \dots x_{n-2} x_{n-1} x_n \\ &\leftarrow (x_j + 1) x_1 x_2 \dots x_{n-2} x_{n-1} \\ &\leftarrow (x_i - 1)(x_j + 1) x_1 x_2 \dots x_{n-2} \end{aligned}$$

We continue to decrease x_i and increase x_j until either $x_i = 0$ or $x_j = \mathsf{ht}(x)+1$. In either case, we have removed a duplicate. Continuing, we will eventually arrive at a vertex with either 0 or 1 duplicates, which as stated previously must be a permutation of the minimum vertex v^{--}.

Next, we note that it is not possible to have $\mathsf{ht}(x) > \mathsf{ht}(v)$, as otherwise x would have been chosen as the minimum vertex (v was chosen so as to have maximum height). Thus in all cases we have shown an undirected path from x^{--} to v^{--}, so the graph must be weakly connected.

Lastly, the graph must be balanced, since any outgoing edge from vertex $w_1 \dots w_{n-2}$ to $w_2 \dots w_{n-1}$ can be paired with the incoming edge $w_{n-1} w_1 \dots w_{n-3}$ to $w_1 \dots w_{n-2}$ and this pairing gives a unique incoming edge for each outgoing edge. □

The next theorem discusses fixed height weak orders. Note that fixed height weak orders can be thought of as surjective functions from $[n]$ to $\{0, 1, \dots, h\}$. While the following theorem appears in [4], we offer a shorter alternative proof to our Corollary 4.6.8.

Theorem 4.6.7. [4] *A ucycle of onto functions from* $[n]$ *to* $\{0, 1, \dots, h\}$ *exists if and only if* $n > h + 1$.

Corollary 4.6.8. *For all $n \in \mathbb{Z}^+$ and all $h \in \mathbb{N}$ with $0 \leq h < n-1$, there exists a universal cycle for $\mathcal{W}(n,h)$.*

Proof. We construct the standard transition graph $G(n,h)$ as follows. We define

$$V(G(n,h)) = \mathcal{W}^-(n,h) = \mathcal{W}^+(n,h)$$

and

$$\begin{aligned} E(G(n,h)) &= \{(v,w) \mid v \in \mathcal{W}^-(n,h), w \in \mathcal{W}^+(n,h), \\ &\qquad \text{and } v_i = w_{i+1} \text{ for } 1 \leq i < n-1\} \\ &= \mathcal{W}(n,h). \end{aligned}$$

Note that if $w_1 w_2 \dots w_n \in \mathcal{W}(n,h)$ then $w_n w_1 w_2 \dots w_{n-1} \in \mathcal{W}(n,h)$ and so $G(n,h)$ is balanced.

To finish the proof, we must show that the graph is connected. Define our minimum vertex in the graph to be $v^- = 0^{n-h-1}[1,h]$. Let $x^- = x_1 x_2 \dots x_{n-1}$ be an arbitrary vertex, and define x_n so that $x = x_1 x_2 \dots x_{n-1} x_n \in \mathcal{W}(n,h)$. We will show a path from x^- to the minimum vertex v^- by first illustrating that all permutations of the minimum vertex are connected, and then describing a path from x^- to some permutation of v^-.

Starting from the vertex $v^- = 0^{n-h-1}[1,h]$, we show that any adjacent transposition can be reached by a path. Since adjacent transpositions generate all permutations, this proves that all permutations of v^- are connected. Let $w_1 w_2 \dots w_{n-1}$ be a permutation of v^-. We will show that the letters w_i and w_{i+1} may be transposed. First, we note that $\{w_1, w_2, \dots, w_{n-1}\} = \{0, 1, 2, \dots, h\}$ by the definition of v^-. Then we find the path as shown in Figure 4.3. Note that along this path, every edge contains all letters from

$$
\begin{aligned}
w_1w_2\dots w_{n-1} &\to w_2w_3\dots w_{n-1}w_i \\
&\to w_3w_4\dots w_{n-1}w_iw_1 \\
&\vdots \quad \text{(rotations of the weak order } w_1w_2\dots w_{n-1}w_i\text{)} \\
&\to w_{i+1}w_{i+2}\dots w_{n-1}w_iw_1w_2\dots w_{i-1} \\
&\to w_{i+2}w_{i+3}\dots w_{n-1}w_iw_1w_2\dots w_{i-1}w_{i+1} \\
&\to w_{i+3}w_{i+4}\dots w_{n-1}w_iw_1w_2\dots w_{i-1}w_{i+1}w_i \\
&\vdots \quad \text{(rotations of the weak order} \\
&\qquad w_1w_2\dots w_{i-1}w_{i+1}w_iw_{i+2}w_{i+3}\dots w_{n-1}w_i\text{)} \\
&\to w_1w_2\dots w_{i-1}w_{i+1}w_iw_{i+2}w_{i+3}\dots w_{n-1}
\end{aligned}
$$

Figure 4.3: Path in $G(n,h)$ illustrating adjacent transpositions

the set $\{w_1, w_2, \dots, w_{n-1}\} = \{0, 1, 2, \dots, h\}$, and so is a valid weak order from $\mathcal{W}(n,h)$.

Now let $x^- = x_1x_2\dots x_{n-1} \in V(G(n,h))$ be arbitrary. We want to define a path from x^- to some permutation of the minimum vertex in $G(n,h)$ by repeatedly replacing any duplicates in x with 0. To create this path, we first define x_n so that $x = x_1x_2\dots x_n \in \mathcal{W}(n,h)$. Then if x has a duplicate at index i, we can replace it by following the path:

$$
\begin{aligned}
x_1x_2\dots x_{n-1} &\to x_2x_3\dots x_{n-1}x_n \\
&\vdots \quad \text{(rotations of } x\text{)} \\
&\to x_ix_{i+1}\dots x_nx_1x_2\dots x_{i-2} \\
&\to x_{i+1}x_{i+2}\dots x_nx_1x_2\dots x_{i-1} \\
&\to x_{i+2}x_{i+3}\dots x_nx_1x_2\dots x_{i-1}0
\end{aligned}
$$

Repeating this procedure, we will eventually arrive at a vertex that is the prefix of some weak order y that is a permutation of $0^{n-h}[1,h]$. Following

rotations, we can find a path to a vertex that is a permutation of v^-. Thus there exists a path from x^- to v^-.

By Theorem 2.1.2, since $G(n,h)$ is balanced and connected, it is eulerian. Therefore we can find a ucycle by following the Euler tour in $G(n,h)$. $\square$

Finally, we prove the following result on the subset of fixed weight, fixed height weak orders on $[n]$.

Result 4.6.9. *For every $n,k,h \in \mathbb{Z}^+$ with $k \leq \binom{n}{2}$, $n > 2$, and $0 \leq h < n$, there is a ucycle for $\mathcal{W}_k^-(n,h)$.*

Proof. We construct the transition graph $G_k(n,h)$ as usual, with

$$V(G_k(n,h)) = \mathcal{W}_k^{--}(n,h)$$

and

$$E(G_k(n,h)) = \mathcal{W}_k^-(n,h).$$

First, we note that the graph is even, since if $w_1 w_2 \dots w_{n-1} \in \mathcal{W}_k^-(n,h)$, then $w_2 \dots w_{n-1} w_1 \in \mathcal{W}_k^-(n,h)$.

Next we must show that the graph is connected. We define the minimum vertex $\mathbf{v} = v_1 v_2 \dots v_{n-2}$ by constructing a specific weak order $\mathbf{w}$ in $\mathcal{W}_k(n,h)$, and then removing the last two elements. Any weak order in $\mathcal{W}_k(n,h)$ must contain the letters $0,1,2,\dots,h$, which have total weight $k' = \sum_{i=0}^{h} i$. Define $\mathbf{s} = s_1 s_2 \dots s_{n-(h+1)}$ to be the lexicographically minimum element of the set $\mathcal{B}_{k-k'}^{h+1}(n-(h+1))$, the set of words of length $n-(h+1)$ and weight $k-k'$ using the alphabet $\{0,1,2,\dots,h\}$. Then extend s to be a weak order by defining $\mathbf{w} = s_1 s_2 \dots s_{n-(h+1)} 01 \dots h$. At this point we have $\mathbf{w} \in \mathcal{W}_k(n,h)$, but we reorder the letters of $\mathbf{w}$ so that

$$\mathbf{w} = [0,h] w_{h+2} w_{h+3} \dots w_n, \text{ where } w_{h+2} \leq w_{h+3} \leq \dots \leq w_n,$$

$$\text{or } \mathbf{w} = [0,h]s_1 s_2 \dots s_{n-(h+1)}.$$

Then define the minimum vertex to be $\mathbf{v} = w_1 w_2 \dots w_{n-2}$.

Now we consider an arbitrary vertex $\mathbf{x} = x_1 x_2 \dots x_{n-2}$, and we will show that it is connected to the minimum vertex. Since $\mathbf{x} \in \mathcal{W}_k^{--}(n,h)$, we define x_{n-1} and x_n so that $x_1 x_2 \dots x_n \in \mathcal{W}_k(n,h)$. Then, by Lemma 4.2.2, we know that $\mathbf{x}$ must be connected to some vertex

$$\mathbf{y} = [0,h]y_{h+2}y_{h+3}\dots y_{n-2}, \text{ where } y_{h+2} \le y_{h+3} \le \dots \le y_{n-2}.$$

If $\mathbf{y} = \mathbf{v}$, then we are done. Otherwise, there exists $i, j \in \{h+2, h+3, \dots, n-2\}$ so that $y_i > v_i$ and $y_j < v_j$. Using rotations and Lemma 4.2.2, there exists a path in $G_k(n,h)$ from $\mathbf{y}$ to the vertex

$$y_i y_j y_1 y_2 \dots y_{i-1} y_{i+1} \dots y_{j-1} y_{j+1} \dots y_{n-2}.$$

Define y_{n-1} and y_n so that $y_1 y_2 \dots y_n \in \mathcal{W}_k(n,h)$. Then we construct the following path in $G_k(n,h)$:

$$\begin{aligned}
& y_i y_j y_1 y_2 \dots y_{i-1} y_{i+1} \dots y_{j-1} y_{j+1} \dots y_{n-2} \\
\to\ & y_j y_1 y_2 \dots y_{i-1} y_{i+1} \dots y_{j-1} y_{j+1} \dots y_{n-2} y_{n-1} \\
\to\ & y_1 y_2 \dots y_{i-1} y_{i+1} \dots y_{j-1} y_{j+1} \dots y_{n-1} y_n \\
\to\ & y_2 \dots y_{i-1} y_{i+1} \dots y_{j-1} y_{j+1} \dots y_n v_i \\
\to\ & y_3 \dots y_{i-1} y_{i+1} \dots y_{j-1} y_{j+1} \dots y_n v_i v_j
\end{aligned}$$

Reordering (by Lemma 4.2.2) and shuffling/replacing elements, we can find a path to the vertex

$$y_1 y_2 \dots y_{i-1} v_i y_{i+1} \dots y_{j-1} v_j y_{j+1} \dots y_{n-2}.$$

Note that by requiring that both $\mathbf{v}$ and $\mathbf{y}$ start with $[0, h]$, we are ensured that these intermediary vertices always contain the set $\{0, 1, 2, \ldots, h\}$, and hence all edges represent valid weak orders in $\mathcal{W}_k^-(n, h)$. Continuing this process, eventually we will arrive at $\mathbf{v}$. Since the graph is even and connected, it must be eulerian by Theorem 2.1.2. □

4.7 Partitions

4.7.1 Partitions of an Integer

Finding universal cycles for the set of partitions of an integer appears to be an area not yet considered in the literature. We pose several open problems to consider.

Open Problem 4.7.1. *For what values of n is there a universal cycle for:*

1. *the set of ordered partitions of n?*
2. *the set of unordered partitions of n?*
3. *the set of k-partitions of n?*

We may also consider partitions in which the actual parts are bounded, such as Fibonacci sequences (see Section 3.7.1 for definitions).

Open Problem 4.7.2. *For what values of n is there a universal cycle for the set of Fibonacci sequences?*

This last question may be easier to answer using an alphabet with more than two symbols, so we are also interested in the following.

Open Problem 4.7.3. *What is the smallest number of symbols needed to find a universal cycle for the Fibonacci sequences of n?*

4.7.2 Ordered Partitions of a Set

As was explored in Section 3.7.2, there is a direct correspondence between the set of ordered partitions of an n-set and the set of weak orders of $[n]$. Thus we have the following results, obtained as corollaries to results in Section 4.6.

Result 4.7.4. *For all $n \in \mathbb{Z}^+$, there is a universal cycle for the set of all ordered partitions of an n-set in which an ordered partition is represented by its corresponding weak order.*

Proof. This is direct from Result 4.6.2.

□

Result 4.7.5. *For all $n \in \mathbb{Z}^+$, there is a universal cycle for the set of all ordered partitions of an n-set with fixed part sizes in which an ordered partition is represented using the prefix representation of the corresponding weak order.*

Proof. The set of ordered partitions with fixed part sizes corresponds directly to the weak orders with fixed multiset. Thus, a universal cycle exists as a direct corollary to Result 4.2.2. □

Result 4.7.6. *For all $n \in \mathbb{Z}^+$ and all $h \in \mathbb{Z}^+ \cup \{0\}$ with $0 \leq h < n$, there exists a universal cycle for the set of all ordered partitions of an n-set into exactly h parts.*

Proof. This is direct from Result 4.6.8. □

Since we have been able to apply several of the results from Section 4.6, it may be interesting to consider the following question.

Open Problem 4.7.7. *Under the correspondence from Theorem 3.7.4, what does the set $\mathcal{W}_k(n)$ correspond to in ordered partitions of an n-set?*

4.7.3 Unordered Partitions of a Set

Universal cycles for unordered partitions of a set were first introduced in [7]. Chung, et al., chose a representation of the partition as a word in which the ith and jth symbols of the word are the same if and only if i and j are in the same part of the partition. Using this representation, they create the standard transition graph. Then the graph is modified to group together elements depending on the partition structure of the first $n-1$ letters of each word. From this, they are able to find the following result.

Theorem 4.7.8. [7] *Universal cycles of unordered partitions of $[n]$ using only n symbols always exist for $n \geq 4$.*

Using the same representation, Hurlbert proved the following theorem, in which part sizes are restricted.

Theorem 4.7.9. [25] *Universal cycles of unordered partitions of* $[n]$ *with part sizes at most* k *exist for all* $4 \leq k \leq n$.

Another consideration is to restrict the number of parts.

Theorem 4.7.10. [25] *Universal cycles for* $(n-1)$*-partitions of* $[n]$ *exist for all odd* $n \geq 3$.

This theorem uses a slightly different representation. In this case, each $(n-1)$-partition is represented by its unique part of size two. Thus these words can be viewed as all possible 2-subsets of $[n]$.

4.8 Designs

To discuss universal cycles for BIBDs, we need one more definition. Let $(V, \mathcal{B})$ be a (v, k, λ)-BIBD and let $\kappa \in \mathbb{Z}^+$ with $\kappa \leq k$. Let $b = |\mathcal{B}|$ and let each $B \in \mathcal{B}$ be represented by a string $x_0 x_1 \ldots x_{\kappa-1}$, where $x_i \in B$ for $0 \leq i < \kappa$. The string $a_0 a_1 \ldots a_{b-1}$ is a **universal cycle of rank** κ for $(V, \mathcal{B})$ if for each block $B \in \mathcal{B}$ there exists some i such that $a_i a_{i+1} \ldots a_{i+\kappa-1} = x_0 x_1 \ldots x_{\kappa-1}$, where $x_0 x_1 \ldots x_{\kappa-1}$ is the string representative for B. This is essentially the same definition for universal cycle that we have been considering throughout this chapter, with the modification that each block may be represented (in a possibly non-unique and non-distinct manner) by a string of length $\kappa \leq k$.

Using this notion of rank κ ucycles, Dewar finds the following results. The first is quite similar to Theorem 4.5.1 (a necessary condition for ucycles for k-subsets of $[n]$).

Theorem 4.8.1. [13] *Let $S = (V, \mathcal{B})$ be a (v, k, λ)-BIBD. If there exists a rank k ucycle for S, then k divides r, where $r = \lambda(v-1)/(k-1)$.*

Theorem 4.8.2. [13] *For $v \equiv 1, 4, 7 \pmod{12}$, there exists a TTS($v$) that admits a 2-intersecting Gray cycle for its blocks, which can be written as a ucycle of rank three.*

Theorem 4.8.3. [13] *Every cyclic symmetric (v, k, λ)-BIBD, $k \geq 3$, admits a ucycle of rank two.*

Dewar also states the following conjectures.

Conjecture 4.8.4. [13] *For each $v \equiv 10 \pmod{12}$, $v \geq 22$, there exists a TTS(v) that admits a rank three ucycle.*

Conjecture 4.8.5. [13] *For each $v \equiv 1 \pmod{12}$, with $v \equiv 0 \pmod{5}$ there exists a TTS(v) that admits a ucycle of rank three.*

Conjecture 4.8.6. [13] *For each $v \equiv 4 \pmod{12}$, with $v \equiv 0 \pmod{5}$, there exists a TTS(v) that admits a ucycle of rank three.*

Chapter 5

OVERLAP CYCLES

5.1 Introduction and Definitions

Definition 5.1.1. *Let $\mathcal{S}$ be a set of objects represented as words from an alphabet. An s-**overlap cycle**, or s-**ocycle**, $O(\mathcal{S}, s)$ for $s \in \mathbb{N}$ is an ordered listing of the elements of $\mathcal{S}$ so that the last s letters of a word are the first s letters of its successor in the listing.*

An example is shown below in Figure 5.1 to illustrate an overlap cycle for 4-subsets of [6] with $s = 2$. We may also write the overlap cycle as a string, as we do for universal cycles. This example corresponds to the word

$$12341526134612451635243625 4635.$$

When writing out overlap cycles, we can list out the sequence fully, or we can choose to omit letters that do not appear as an overlap (**hidden** letters).

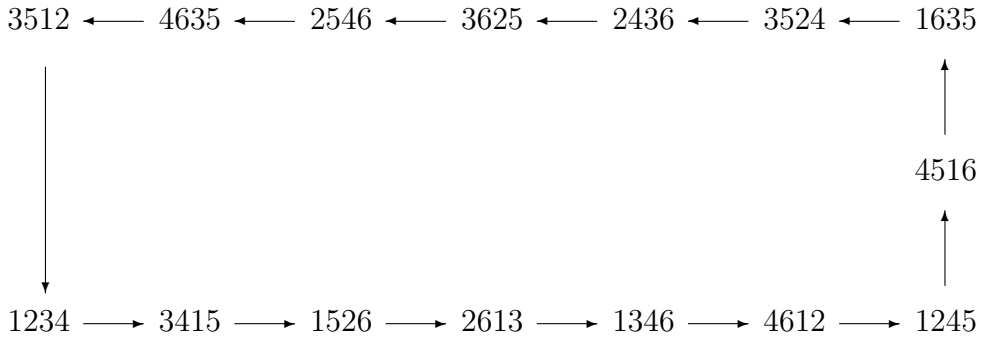

Figure 5.1: 2-Overlap cycle for 4-subsets of [6]

When we omit hidden letters, the cycle is written in **compressed form**. For example, a 1-ocycle for an STS(7) with overlap points underlined is

$$\underline{2}, 1, \underline{0}, 3, \underline{4}, 2, \underline{5}, 0, \underline{6}, 4, \underline{1}, 5, \underline{3}, 6, \underline{2},$$

and in compressed form we use only the underlined points and write

$$20456132.$$

Note that by this definition, a universal cycle of words of length n is an overlap cycle with $s = n-1$. We can view overlap cycles as a "near" result for universal cycles. In this case, we are hoping to maximize the overlap as much as possible with the hope of eventually achieving an overlap of $n-1$. In the case of an overlap of size s, the total length of the compressed word is $n-s$ times the length of a universal cycle on the set of objects, should one exist.

Definition 5.1.2. *Let $\mathcal{S}(n,m)$ be the set of all n-letter words from an alphabet of size m. For $s \in \mathbb{N}$ with $n > s$, the **alphabet overlap graph** $G(\mathcal{S}(n,m), s)$ is the graph whose vertex set is $\mathcal{S}(n,m)$ and an edge from v to w if and only if the last s letters of v are the first s letters of w.*

This definition defines a transition graph, which is very useful in conjunction with the following theorem.

Theorem 5.1.3. [19] *The alphabet overlap graph is hamiltonian for all non-trivial values of the parameters.*

Aside from this important and useful result, the area of overlap cycles seems largely undeveloped.

5.2 Binary Words of Length n

The set of binary words of length n, $\mathcal{B}(n)$, corresponds to the set of all n-letter words from an alphabet of size two. Thus we have the following direct corollary from Theorem 5.1.3.

Corollary 5.2.1. *For all $n, s \in \mathbb{N}$ with $n > s$, there exists an overlap cycle $O(\mathcal{B}(n), s)$.*

Proof. By Theorem 5.1.3, the corresponding alphabet overlap graph $G(\mathcal{S}(n, 2), s)$ is hamiltonian. From a Hamilton cycle in the graph we can obtain the desired overlap cycle. □

Note that since Theorem 5.1.3 refers only to the complete set of binary strings of length n, we must consider any restricted subset separately.

Open Problem 5.2.2. *For what restricted subsets of $\mathcal{B}(n)$ do there exist alphabet overlap cycles?*

However, we do have a corresponding lemma to Lemma 4.2.2.

Lemma 5.2.3. *Let $n, s \in \mathbb{Z}^+$ with $\frac{n}{2} \leq s \leq n-2$, and let M be some fixed multiset of size n. Define the set A to be the set of all permutations of M. If $gcd(s, n) = 1$, then there is an s-ocycle for A.*

Proof. We first construct the transition graph with vertices as s-prefixes and suffixes of words in A, and edges representing the words themselves. Since any s-prefix of a string in A is also an s-suffix, the graph must be balanced.

In order to prove that the transition graph is eulerian, and thus complete the proof, all that remains is to show that the graph must be connected, which

will be done by showing that from an arbitrary vertex we may transpose any two adjacent letters. Consider $w = w_1 w_2 \dots w_n \in A$ with s-prefix $w^{s-} = w_1 w_2 \dots w_s$, and let $g = \gcd(s, n)$. When we examine all rotations of the vertex w, beginning with w^{s-}, we see that through rotations the only vertices reached are ones that begin with w_{cg} for $1 \leq c \leq \frac{n}{g}$. If $g = 1$, then our rotations will take us through every possible consecutive s-substring of w. Thus we need only show *one* adjacent transposition in w, and all others may be reached through rotations. This is done through the following path, in which $u \rightsquigarrow v$ denotes that v is reached by following rotations of u. We can transpose $w_{s-1} \leftrightarrow w_s$ by following the undirected path given below.

$$\begin{aligned} w_1 w_2 \dots w_{s-2} w_{s-1} w_s &\rightsquigarrow w_n w_1 \dots w_{s-2} w_{s-1} \\ &\rightsquigarrow w_{n-1} w_n w_1 \dots w_{s-2} \\ &\rightsquigarrow w_n w_1 \dots w_{s-2} w_s \\ &\rightsquigarrow w_1 \dots w_{s-2} w_s w_{s-1} \end{aligned}$$

Thus we can transpose a pair of adjacent letters in w^{s-}, and so we are done. □

In fact, when $s < \frac{n}{2}$ we can always find an s-ocycle.

Lemma 5.2.4. *Let $n, s \in \mathbb{Z}^+$ with $1 \leq s < \frac{n}{2}$. Let M be a multiset of size n. Define the set A to be the set of all permutations of M. Then there is an s-ocycle for A.*

Proof. Construct the transition graph with vertices as s-prefixes and s-suffixes of words in A, and edges representing the words themselves. Fix an arbitrary vertex $v^{s-} = v_1 v_2 \dots v_s$ as the minimum vertex. To prove the existence of an Euler tour, and thus prove the existence of an s-ocycle, we will show that from any vertex $w^{s-} = w_1 w_2 \dots w_s$, we can find a path to the minimum vertex.

Compare v^{s-} and w^{s-}, and consider the first position in which they differ, say index i. In other words, $v_i \neq w_i$, and for all $1 \leq j < i$ we have $v_j = w_j$. We will set $w = w_1 w_2 \dots w_n \in A$ so that w has s-prefix w^{s-}. We have two cases.

1. First, if $v_i \in \{w_{i+1}, w_{i+2}, \dots w_s\}$, then we use the following undirected path, in which we merely transpose letters w_i and v_i in w^{s-}.

$$\begin{aligned} w_1 w_2 \dots w_{i-1} w_i w_{i+1} \dots v_i \dots w_s &\rightarrow w_{n-s+1} w_{n-s+2} \dots w_n \\ &\leftarrow w_1 w_2 \dots w_{i-1} v_i w_{i+1} \dots w_i \dots w_s \end{aligned}$$

2. Second, if $v_i \in \{w_{s+1}, w_{s+2}, \dots w_n\}$, then v_i does not appear in vertex w^{s-}. Note that vertex $w^{s-} = w_1 w_2 \dots w_s$ is connected to any s-suffix, which consists of all s-permutations of the set

$$B = M \setminus \{w_1, w_2, \dots w_s\}.$$

   Thus $v_i \in B$, so we may choose an edge leaving w^{s-} that leads to an s-suffix not containing v_i, i.e. v_i does not appear in either vertex. In this case, we may simply replace w_i with v_i as shown in the following path.

$$\begin{aligned} w_1 w_2 \dots w_{i-1} w_i w_{i+1} \dots w_s &\rightarrow w_{n-s+1} w_{n-s+2} \dots w_n \\ &\leftarrow w_1 w_2 \dots w_{i-1} v_i w_{i+1} \dots w_s \end{aligned}$$

At this point, we are one step closer to the minimum vertex, as now the two vertices agree in the first i positions. Repeating, we will eventually find a path to the minimum vertex. Thus, the graph is connected. Finally, since clearly any s-suffix of a string in A is also an s-prefix, the graph is balanced and hence eulerian. □

5.3 m-ary Words of Length n

As in Section 5.2, we have a direct corollary from Theorem 5.1.3.

Corollary 5.3.1. *For all $m, n, s \in \mathbb{N}$ with $n > s$, there exists an overlap cycle $O(\mathcal{B}^m(n), s)$, where $\mathcal{B}^m(n)$ is the set of all m-ary words of length n.*

Proof. By Theorem 5.1.3, the corresponding alphabet overlap graph $G(\mathcal{S}(n, m), s)$ is hamiltonian. From a Hamilton cycle in the graph we can obtain the desired overlap cycle. □

To consider any subsets of $\mathcal{B}^m(n)$, we must consider some alternative approach.

Open Problem 5.3.2. *For what restricted subsets of $\mathcal{B}^m(n)$ do there exist alphabet overlap cycles?*

5.4 Permutations

When searching for overlap cycles of permutations, we can no longer apply Theorem 5.1.3, since a set of permutations is not a set of *all* words of a given length over some fixed alphabet. For example, a set of permutations does not contain any words with repeated letters, while these are clearly to be included in the set referred to in Theorem 5.1.3.

In the following question, define the **permutation overlap graph** $P(k, n, s)$ to be the graph whose vertex set is the set of all k-permutations of $[n]$ and an

edge from v to w if and only if the last s letters of v are the first s letters of w.

Question 5.4.1. ([29], Problem 481.) *Is the permutation overlap graph $P(k, n, s)$ Hamiltonian whenever $k < n$? In other words, is there always an s-overlap cycle for k-permutations of $[n]$ when $k < n$?*

To answer this question, we will use several lemmas. We begin by extending Lemma 5.2.3 from Section 5.2.

Result 5.4.2. *Let $n, s \in \mathbb{Z}^+$ with $n \geq 2$. If either (1) $1 \leq s < \frac{n}{2}$, or (2) $gcd(s, n) = 1$ with $\frac{n}{2} \leq s < n - 1$, then there exists an s-ocycle on the set of permutations of $[n]$.*

Proof. This is merely a concise restatement of Lemmas 5.2.3 and 5.2.4 with $M = \{1, 2, \ldots, n\}$. □

Note that this result also shows that under the given conditions, all permutations of an $[n]$-set are connected within a larger transition graph. We can extend this to k-permutations of $[n]$ to begin to answer the Question 5.4.1 from [29].

Result 5.4.3. *Let $n, s, k \in \mathbb{Z}^+$ with $1 \leq k < n$. If either (1) $1 \leq s < \frac{k}{2}$, or (2) $gcd(s, k) = 1$ with $\frac{k}{2} \leq s < k - 1$, then there exists an s-ocycle on the set of k-permutations of $[n]$.*

Proof. First, construct the transition graph with vertices of length s (s-prefixes of k-permutations) and edges representing k-permutations. We allow an edge from vertex u to vertex v if and only if u is an s-prefix and v is an s-suffix

for some k-permutation. We will show that this graph is balanced and weakly connected, and thus is eulerian. From an Euler tour, we can find an s-ocycle.

Define the minimum k-permutation $v = 123\ldots k$, and the minimum vertex, v^{s-}, in the transition graph to be the s-prefix of v. Let $w^{s-} = w_1 w_2 \ldots w_s$ be the prefix of an arbitrary k-permutation $w_1 w_2 \ldots w_k$. By Result 5.4.2, all permutations of a k-set are connected under our hypotheses, so we may assume that w is ordered as $w_1 < w_2 < \cdots < w_k$. We will show that there is a path in the graph from w^{s-} to v^{s-}. Define

$$D = \{w_1, w_2, \ldots, w_k\} \setminus \{1, 2, 3, \ldots, k\}$$

and

$$\overline{D} = \{1, 2, 3, \ldots, k\} \setminus \{w_1, w_2, \ldots, w_k\}.$$

Note that if $D = \emptyset$, then w is a permutation of v and so by the comments following Result 5.4.2, we know that there exists a path. For $D \neq \emptyset$, we choose d letters $a_1, a_2, \ldots, a_d \in \overline{D}$, where $d = \min\{k - s, |\overline{D}|\}$. If $d < k - s$, then we may also select letters $a_{d+1}, a_{d+2}, \ldots, a_{k-s} \in \{w_{s+1}, w_{s+2}, \ldots, w_k\}$. Now in our graph we follow the edge corresponding to the k-permutation $w_1 w_2 \ldots w_s a_1 a_2 \ldots a_{k-s}$. By following this edge, we have found a k-permutation with more letters in common with v than w did. Since Result 5.4.2 implies that all permutations of a k-set are connected in the transition graph, we may arrange this string in increasing order, and by repeating this procedure we will eventually find a k-permutation that is simply a permutation of v, at which point we are done.

Since we have shown that the graph is connected, we need only show that the graph is balanced in order to prove that an Euler tour exists. However it

is clear that the graph is balanced, as any prefix of a k-permutation is also a suffix of a k-permutation. $\square$

Finally, we may extend this result to consider s-ocycles for k-permutations of $[n]$, without the conditions needed in Result 5.4.3.

Result 5.4.4. *For all $n, s, k \in \mathbb{Z}^+$ with $1 \leq s < k < n$, there is an s-ocycle for k-permutations of $[n]$.*

Proof. We construct the standard transition graph G where vertices of length s correspond to s-prefixes of k-permutations of $[n]$, and edges correspond to k-permutations of $[n]$. If we can show that this graph is eulerian, then we have shown that there exists an s-ocycle for k-permutations of $[n]$. First, we note that since any s-prefix of a k-permutation is also an s-suffix of a k-permutation, the graph is balanced. All that remains is to show that the graph is connected.

Define the minimum vertex $v^{s-} = 12 \dots s$, and let $w^{s-} = w_1 w_2 \dots w_s$ be an arbitrary vertex in the graph, which we assume to be an s-prefix of the k-permutation $w = w_1 w_2 \dots w_k$. We will frequently refer to **rotations** of a vertex. This is defined as following the edges of the cycle corresponding to rotations of a k-permutation that the vertex is an s-prefix for.

We next compare w^{s-} with v^{s-}. Let i be the first index in which $w_i \neq v_i$ and let $g = \gcd(s, k)$. If $g = 1$, we are done by the previous result. Otherwise, we note that rotations of w partition the string into blocks of length g. All addition in indices will be modulo k. We have two cases.

Case 1: If $i \notin \{w_{i+1}, w_{i+2}, \dots, w_k\}$:

Rotate w so that w_i is in the first block. This means that we are con-

sidering some vertex

$$w_{i-t}w_{i-t+1}\dots w_{i-t+s-1},$$

with $t \leq g-1$, or $i \in [i-t, i-t+g-1]$. Follow the edge out of this vertex that corresponds to a rotation of w. This takes us to the vertex

$$w_{i-t+k-s}w_{i-t+k-s+1}\dots w_{i-t+k-1}.$$

Next we follow the backwards edge corresponding to the k-permutation

$$w_{i-t}w_{i-t+1}\dots w_{i-1}(i)w_{i+1}\dots w_{i-t+k-1}.$$

Now we are at the vertex

$$w_{i-t}w_{i-t+1}\dots w_{i-1}(i)w_{i+1}\dots w_{i-t+s-1}.$$

Finally we follow rotations of this vertex to end at the vertex

$$12\dots(i)w_{i+1}w_{i+2}\dots s.$$

This vertex is closer to the minimum vertex since the first i letters agree. Repeating this procedure, we will eventually arrive at the minimum vertex.

Case 2: If $i \in \{w_{i+1}, w_{i+2}, \dots, w_k\}$:

Rotate w so that i is in the first block. We are now considering some vertex

$$a_1a_2\dots a_s$$

with $i \in \{a_1, a_2, \dots, a_g\}$, so assume that $i = a_j$. Note that

$$|[n] \setminus \{w_1, w_2, \dots, w_k\}| > 1,$$

so choose some x in this set. We follow the edge from $a_1a_2\ldots a_s$ corresponding to a rotation of $a_1a_2\ldots a_{j-1}(i)a_{j+1}\ldots a_k$, which we recall is a rotation of w. This takes us to the vertex

$$a_{k-s+1}a_{k-s+2}\ldots a_k,$$

which does not contain the letter i. From this vertex we follow the backward edge

$$a_1a_2\ldots a_{j-1}(x)a_{j+1}\ldots a_k.$$

Note that i does not appear in this edge, so we can go to Case (1).

When we have finished, we will have arrived at the minimum vertex. Thus the graph is weakly connected, and so is eulerian. □

We finish the section by answering Question 5.4.1.

Result 5.4.5. *The permutation overlap graph $P(k,d,s)$ is hamiltonian whenever $k < d$.*

Proof. The permutation overlap graph $P(k,d,s)$ is hamiltonian if and only if there exists an s-ocycle for k-permutations of $[d]$. Thus by Result 5.4.4, we are done. □

5.5 Subsets

The example in Section 5.1, Figure 5.1, shows a 2-overlap cycle for 4-subsets of [6]. Define $\binom{[n]}{k}$ to be the set of all k-subsets of $[n]$.

Open Problem 5.5.1. *For what values of n, k, s does there exist an overlap cycle $O\left(\binom{[n]}{k}, s\right)$?*

In this case, we are not searching for a Hamilton cycle or an Euler tour in the transition graph, since we must have one vertex or edge to represent each possible ordering of a subset. Recall from Theorem 4.5.1 that it is necessary that $k \mid \binom{n-1}{k-1}$ for a universal cycle for k-subsets of $[n]$ to exists. Expanding on this, we suggest the following question.

Open Problem 5.5.2. *Is there a necessary condition for overlap cycles for k-subsets of $[n]$, as Theorem 4.5.1 is for ucycles?*

5.6 Weak Orders

We begin with a general result.

Result 5.6.1. *For all $n, s \in \mathbb{Z}^+$ with $1 \leq s \leq n-1$, there is an s-ocycle for $\mathcal{W}(n)$.*

To prove this, we have two cases: $1 \leq s \leq \frac{n}{2}$ (vertices do not need to overlap to make a weak order), and $\frac{n}{2} < s \leq n-1$ (vertices must overlap to make a weak order). First, we define the s-prefix, w^{s-}, and s-suffix, w^{s+}, of a word as:

$$w^{s-} = w_1 w_2 \dots w_s$$

and

$$w^{s+} = w_{n-s+1} w_{n-s+2} \dots w_n.$$

Fact 5.6.2. *If $w = w_1 w_2 \dots w_n \in \mathcal{W}(n)$, then any permutation of the letters of w is also in $\mathcal{W}(n)$.*

In light of this fact, we note that $\{w^{s-} \mid w \in \mathcal{W}(n)\} = \{w^{s+} \mid w \in \mathcal{W}(n)\}$. In fact, these two sets are also equal to the set

$$\mathcal{W}^s(n) = \{w = w_1 w_2 \dots w_s \mid w \text{ is a subword of some } w' \in \mathcal{W}(n)\}.$$

Proof of Result 5.6.1. We define a transition graph $G^s(n)$ as follows. Let $V(G^s(n))$ be equal to the set of all possible overlaps, i.e. $V(G^s(n)) = \mathcal{W}^s(n)$. Define $E(G^s(n))$ to contain one edge for each weak order by creating a directed edge (u, v) for each weak order w that begins with u and ends with v. We will show that this graph contains an Euler tour, which will give us an s-ocycle for $\mathcal{W}(n)$.

First, $G^s(n)$ must have $d^+(u) = d^-(u)$ for all $u \in V(G^s(n))$, since any permutation of a weak order is again a weak order. That is, if $u = u_1 u_2 \dots u_s$ is a prefix of some weak order $u_1 u_2 \dots u_n$, then we have the incoming edge:

$$u_{s+1} u_{s+2} \dots u_n u_1 u_2 \dots u_s$$

and the outgoing edge

$$u_1 u_2 \dots u_s u_{s+1} \dots u_n.$$

Thus we can pair together each incoming edge with an outgoing edge, so we must have $d^+(u) = d^-(u)$.

Lastly we must show that $G^s(n)$ is connected. We will show that any vertex v is connected to the vertex 0^s. Let $v = v_1 v_2 \dots v_s \in V(G^s(n))$ with $h = \mathsf{ht}(v)$. If $s \leq \frac{n}{2}$, then we can always find an edge (v, u) to some vertex u with $\mathsf{ht}(u) < h$. If $s > \frac{n}{2}$, the weak order $v_1 v_2 \dots v_s v_{s+1} \dots v_n$ is represented by

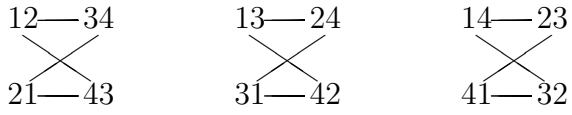

Figure 5.2: Transition Graph: 2-ocycles for permutations of $\{1, 2, 3, 4\}$

the edge $(v, v_{n-s}v_{n-s+1}\dots v_n)$. Note that $\mathsf{ht}(v_{n-s}v_{n-s+1}\dots v_n) \leq h$, and that any letter v_i of maximum height in $v_{n-s}v_{n-s+1}\dots v_n$ must have $n-s \leq i \leq s$. Thus by repeating this procedure (at most n times if $s = n-1$ and v_{n-1} has maximum height in v), we reach some vertex $u \in V(G^s(n))$ with $\mathsf{ht}(u) < h$.

In either case, we have moved to a vertex with smaller height. By repeating this procedure, we will eventually reach a vertex with height 0. The only vertex with height 0 is the vertex 0^s, and so we have arrived at our destination.

Since our graph $G^s(n)$ is connected and $d^+(u) = d^-(u)$ for all $u \in V(G^s(n))$, we must have an Euler tour in the graph. This Euler tour will translate to an s-ocycle on $\mathcal{W}(n)$. □

When we consider s-ocycles for fixed weight weak orders on $[n]$, we notice that the following theorems follow immediately from their corresponding results about ucycles with very small adjustments. Note, however, that we must consider whether or not s and n are relatively prime. For example, if we consider all permutations of the set $\{1, 2, 3, 4\}$, the transition graph for 2-ocycles is disconnected, as shown in Figure 5.2.

Result 5.6.3. *Let $n, s, k \in \mathbb{Z}^+$ with $1 \leq s \leq n-2$ and $k \leq \binom{n}{2}$. If we have $gcd(s, n) = 1$, then there is an s-ocycle for $\mathcal{W}_k(n)$.*

Proof. Small adjustments to the proof of Result 4.6.4 (see Result 5.2.3). □

Result 5.6.4. *Let $n, s, h, k \in \mathbb{Z}^+$ with $1 \leq s \leq n-2$, $k \leq \binom{n}{2}$, and $0 \leq h < n$. If $gcd(s, n) = 1$, then there is an s-ocycle for $\mathcal{W}_k(n, h)$.*

Proof. Small adjustments to the proof of Theorem 4.6.9. □

To produce an ocycle equivalent of Theorem 4.6.8, we can simplify the proof since we are now dealing with overlaps of at most $n-2$, and again we require that $\gcd(s, n) = 1$.

Result 5.6.5. *For all $n, s, h \in \mathbb{Z}^+$ with $1 \leq s \leq n-2$, $gcd(s, n) = 1$, and also with $0 \leq h \leq n-1$, there is an s-ocycle for $\mathcal{W}(n, h)$.*

Proof. We define our transition graph as usual, with

$$V(G^s(n, h)) = \mathcal{W}^s(n, h) \text{ and } E(G^s(n, h)) = \mathcal{W}(n, h).$$

Clearly the graph is balanced, so we need only show that it is connected. Define the minimum vertex v^s to be the first s letters of the weak order $v = [0, h]0^{n-h-1}$. Let $x^s = x_1 x_2 \dots x_s$ be an arbitrary vertex in the graph. We assume that x^s is an s-prefix of some $x = x_1 x_2 \dots x_n \in \mathcal{W}(n, h)$. Applying Lemma 5.2.3, we may assume that x is ordered in any manner we would like, and so we can assume that $x = [0, h]x_{h+2}x_{h+3} \dots x_n$. If $s \leq h+1$, then the first s letters of x equals v^s and we are done. Otherwise, $s \geq h+2$, and we follow the edge that corresponds to the weak order

$$[0, h]x_{h+2}x_{h+3} \dots x_s 0 \cdots 0.$$

Applying Lemma 5.2.3 again, we can reorder this weak order to find a path to the vertex that consists of the first s letters of $[0, h]0 \cdots 0x_{h+2}x_{h+3} \dots x_s$, and we are one step closer to the minimum vertex. Repeating, we will eventually

arrive at the minimum vertex v^s. Thus the graph is connected, and so contains an Euler tour, and hence an s-ocycle. □

5.7 Partitions

As seen in Sections 5.4 and 5.5, we cannot apply Theorem 5.1.3 to any types of partitions.

5.7.1 Partitions of an Integer

Let $n \in \mathbb{Z}^+$, and let $P(n)$ be the set of partitions of n.

Open Problem 5.7.1. *For what values of n and s does there exist an overlap cycle $O(P(n), s)$?*

We may also ask the same question for restricted subsets of $P(n)$, such as the Fibonacci sequences.

Open Problem 5.7.2. *For what restricted subsets of $P(n)$ can we find an overlap cycle?*

5.7.2 Ordered Partitions of a Set

Under the correspondence discussed in Theorem 3.7.4, we obtain the following results as corollaries to similar results in Section 5.6.

Corollary 5.7.3. *For all $n, s \in \mathbb{Z}^+$ with $1 \leq s \leq n-1$, there is an s-ocycle for the set of all ordered partitions of an n-set.*

Proof. This is immediate from Result 5.6.1. □

Corollary 5.7.4. *For all n, s, h with $1 \leq s \leq n-2$, $gcd(s, n) = 1$, and $h \leq n-1$, there exists an s-ocycle for the set of all ordered partitions of an n-set into exactly h parts.*

Proof. This is immediate from Result 5.6.5. □

Corollary 5.7.5. *Let $n, s \in \mathbb{Z}^+$ with $1 \leq s \leq n-2$. If $gcd(n, s) = 1$, then there is an s-ocycle for the set of all ordered partitions of an n-set with fixed part sizes.*

Proof. This is immediate from Lemma 5.2.3. □

5.7.3 Unordered Partitions of a Set

Open Problem 5.7.6. *For what types of sets can we find overlap cycles for the set of unordered partitions of the set?*

5.8 Designs

In this section, we will often use the notation $x \oplus B$ where B is a block in the block set of a design. If $B = \{a, b, c\}$, we define $x \oplus B = \{xa, xb, xc\}$, where xa is short-hand for the ordered pair (x, a). This notation will also be

used with ucycles and overlap cycles, i.e.,

$$x \oplus \mathcal{A} = xa_0, xa_1, \ldots, xa_n, xa_0$$

if $\mathcal{A} = a_0, a_1, \ldots, a_n, a_0$ is a ucycle or ocycle.

Open Problem 5.8.1. *For what types of designs do there exist overlap cycles?*

We can answer this question easily for transversal designs and 1-overlap cycles. We know from Result 3.8.3 that for all $n \geq 2$, $k \geq 3$, and any $TD(k, n)$, we have a basis representation for the blocks of the design in which each block is represented by a string of length two, and the set of these strings corresponds to the set of all n-ary words of length two. Thus we have the following result.

Result 5.8.2. *For all $n \geq 2$, $k \geq 3$, and any $TD(k, n)$, there is a universal cycle on the blocks of the design using a basis representation in which each block is represented by a string of length two, which corresponds to its points from groups 1 and 2.*

Proof. By Result 3.8.3, we know that there is a direct correspondence between this basis representation and the set of all n-ary words of length two. Then by Theorem 4.3.1, there is a universal cycle for this basis set. □

As stated previously, this representation discards many points in each block, so the representation of two different transversal designs could be identical. Depending on the application, it might be wiser to consider the overlap cycle listing structure to preserve this data.

Result 5.8.3. *For all $n \geq 2$ and $k \geq 3$ and any $TD(k, n)$, there exists a 1-overlap cycle, $O(TD(k, n), 1)$.*

Proof. Using Result 5.8.2, we know that a universal cycle exists for any $TD(k, n)$ using a basis representation in which each block is represented by a string of length two. Consider one block in particular, say $\{b_1, b_2, \ldots, b_k\}$ represented by the string b_1b_2. Somewhere in the universal cycle appears the pair b_1b_2 consecutively. We can replace b_1b_2 by $b_1b_3b_4 \ldots b_kb_2$. Doing this for each block, we have transformed the universal cycle into a 1-overlap cycle. □

Recall the definition of κ-intersecting Gray code for a (v, k, λ)-BIBD. This Gray code requires that successive blocks intersect in exactly κ points. Unfortunately, the location of these κ points is not specified. If we can restrict the positions of these κ points, we may be able to create an overlap cycle for the designs that admit κ-intersecting Gray codes.

While the κ-intersecting Gray code may be a step in the right direction, Dewar's discussion of universal cycles over designs ([13], p. 87) may be of more use to us. Dewar utilizes the basis representation for designs, in particular showing an example that uses two elements to represent each block from the Fano plane (see Figure 2.2). This representation allows the ucycle 5134620. However, note that for each block $B = \{a, b, c\}$ we may increase the representation from two elements to the full block representation by "stuffing" the third block element in between, as in the proof of Result 5.8.3. This converts our ucycle to the 1-overlap cycle 52103541632406. For a more general example, we have the following corollary to Dewar's Theorem 4.8.3 in Section 4.8.

Corollary 5.8.4. *Every cyclic symmetric $(v, k, 1)$-BIBD with $k \geq 3$ admits a 1-overlap cycle.*

Proof. Let $S = (V, \mathcal{B})$ be a cyclic symmetric $(v, k, 1)$-BIBD with $k \geq 3$. Since

$\lambda = 1$, any pair of elements from V appears in only one block from $\mathcal{B}$. Then any block $B \in \mathcal{B}$ can be represented by any two elements from B. Thus a ucycle of rank two can be converted to a 1-overlap cycle by stuffing the additional $k-2$ elements in each block between its representatives. By Dewar's result 4.8.3 in Section 4.8, we know that such a ucycle must exist. □

Design theory has many constructions for BIBDs that are recursive. We are interested in these recursive constructions, i.e. given base designs that admit s-overlap cycles, does the design constructed also admit an s-overlap cycle? We now present several different constructions paired with corresponding results on overlap cycles.

Theorem 5.8.5. (Sum Construction, [46].) *Suppose there exists a (v, k, λ_1)-BIBD and a (v, k, λ_2)-BIBD. Then there exists a $(v, k, \lambda_1 + \lambda_2)$-BIBD.*

Construction. Let $(X, \mathcal{A})$ be a (v, k, λ_1)-BIBD and $(X, \mathcal{B})$ be a (v, k, λ_2)-BIBD. Then the pair $(X, \mathcal{A} \cup \mathcal{B})$ is a $(v, k, \lambda_1 + \lambda_2)$-BIBD. □

Result 5.8.6. *Let $(X, \mathcal{A})$ and $(X, \mathcal{B})$ be $(v, 3, 1)$-BIBDs. Suppose that both $(X, \mathcal{A})$ and $(X, \mathcal{B})$ admit 1-overlap cycles called $O(\mathcal{A})$ and $O(\mathcal{B})$, respectively. Then the $(v, 3, 2)$-BIBD $(X, \mathcal{A} \cup \mathcal{B})$ using the sum construction also admits a 1-overlap cycle.*

Proof. Define a graph G on X with edge (v, w) if vxw appears (in order) as a block in the ucycle $O(\mathcal{A})$ or $O(\mathcal{B})$ for some $x \in X$. Note that the graph is balanced, since the edges of $O(\mathcal{A})$ and $O(\mathcal{B})$ form two edge-disjoint circuits on X. If we can show that the graph is connected, then G must be eulerian, and an Euler tour will correspond to a 1-overlap cycle on $(X, \mathcal{A} \cup \mathcal{B})$.

Note that if the graph is not connected, then the two circuits $C_{\mathcal{A}}$ and $C_{\mathcal{B}}$ that correspond to $O(\mathcal{A})$ and $O(\mathcal{B})$ must be not only edge-disjoint, but also vertex-disjoint. If we can show that both $C_{\mathcal{A}}$ and $C_{\mathcal{B}}$ contain at least $|X|/2$ vertices, then the two cycles cannot be vertex-disjoint, and hence the graph must be connected.

Consider $C_{\mathcal{A}}$. If fewer than $|X|/2$ vertices appear as overlap points in $C_{\mathcal{A}}$, then the maximum number of blocks that can appear in $C_{\mathcal{A}}$ is:

$$\binom{v/2}{2} = \frac{(v^2 - 2v)}{8}.$$

We know that the number of blocks in $\mathcal{A}$ is equal to:

$$\frac{(v^2 - v)}{k^2 - k} = \frac{(v^2 - v)}{6}.$$

In order for $C_{\mathcal{A}}$ to cover the entire design $(X, \mathcal{A})$, we must have that:

$$\frac{(v^2 - 2v)}{8} \geq \frac{(v^2 - v)}{6}. \tag{5.1}$$

We can rewrite (5.1) as:

$$\begin{aligned} \frac{(v^2 - 2v)}{8} &\geq \frac{(v^2 - v)}{6} \\ \frac{v - 2}{4} &\geq \frac{v - 1}{3} \\ -2 &\geq v \end{aligned}$$

Thus we have a contradiction, and so $C_{\mathcal{A}}$ must contain more than $|X|/2$ vertices. The same argument follows to show that $C_{\mathcal{B}}$ must contain more than $|X|/2$ vertices. Therefore the two cycles cannot be vertex-disjoint, and hence the graph is connected. □

Open Problem 5.8.7. *Can we generalize Result 5.8.6 using the Sum Construction with $\lambda_1, \lambda_2 \in \mathbb{Z}^+$?*

For example, we know that there exists a 1-overlap cycle for the unique $(7, 3, 1)$-BIBD. It is:

$$\overline{752}\underline{13}\overline{74}\underline{26}\overline{3}\underline{54}\overline{167}$$

when our design has $X = \{1, 2, 3, 4, 5, 6, 7\}$ and

$$\mathcal{B} = \{1, 2, 3\}, \{1, 4, 5\}, \{1, 6, 7\}, \{2, 4, 6\}, \{2, 5, 7\}, \{3, 4, 7\} \text{ and } \{3, 5, 6\}.$$

From this, we can apply Result 5.8.6 to find a $(7, 3, 2)$-BIBD that admits a 1-overlap cycle. Alternatively, we could also have used the following construction theorem and corresponding ocycle result.

Theorem 5.8.8. [46] *Suppose there exists a (v, k, λ)-BIBD. Then there exists a $(v, k, s\lambda)$-BIBD for all integers $s \geq 1$.*

Construction. If $(X, \mathcal{A})$ is a (v, k, λ)-BIBD, construct a $(v, k, s\lambda)$-BIBD by repeating each block from $\mathcal{A}$ s times. □

Result 5.8.9. *If $(X, \mathcal{A})$ is a (v, k, λ)-BIBD that admits a t-overlap cycle, then the $(v, k, s\lambda)$-BIBD constructed from Theorem 5.8.8 also admits a t-overlap cycle.*

Proof. Let $O(\mathcal{A})$ be the t-overlap cycle for $(X, \mathcal{A})$. Then concatenating $O(\mathcal{A})$ with itself s times gives a t-overlap cycle for $(X, s \cdot \mathcal{A})$, the $(v, k, s\lambda)$-BIBD constructed from Theorem 5.8.8. □

Note that Result 5.8.6 could potentially find a **simple** design (a design without repeated blocks) with a 1-overlap cycle, while Result 5.8.9 will never find a simple design.

Theorem 5.8.10. (Block Complementation, [46].) *Suppose that for $k \leq v-2$ there exists a (v, b, r, k, λ)-BIBD. Then there also exists a $(v, b, b-r, v-k, b-2r+\lambda)$-BIBD.*

Construction. If $(X, \mathcal{A})$ is a (v, b, r, k, λ)-BIBD with $k \leq v-2$, then $(X, \{X \backslash A \mid A \in \mathcal{A}\})$ is a $(v, b, b-r, v-k, b-2r+\lambda)$-BIBD. □

Result 5.8.11. *Given $(X, \mathcal{A}) = (v, b, r, k, \lambda)$-BIBD, we can construct the block complementation of it to create a new design. This design is $(X, \overline{\mathcal{A}})$, which is a $(v, b, b-r, v-k, b-2r+\lambda)$-BIBD. If $v-2k \geq 2$ and $(X, \mathcal{A})$ has a 1-overlap cycle, then $(X, \overline{\mathcal{A}})$ also has a 1-overlap cycle.*

Proof. Let the blocks in $\mathcal{A}$ appear in the following order in the sequence: $A_1, A_2, \ldots, A_b$. We claim that one can find a 1-overlap cycle for $\overline{\mathcal{A}}$ in which the blocks are ordered as $\overline{A_1}, \overline{A_2}, \ldots, \overline{A_b}$.

Note that we have:

$$\begin{aligned} |\overline{A_i} \cap \overline{A_j}| &= |X| - |A_i \cup A_j| \\ &= |X| - (|A_i| + |A_j| - |A_i \cap A_j|) \\ &= |X| - |A_i| - |A_j| + |A_i \cap A_j| \\ &\geq v - 2k \\ &\geq 2. \end{aligned}$$

We order the points in the blocks as follows. Let $x \in \overline{A_1} \cap \overline{A_b}$. We know that such an x must exist, since we just showed that any two blocks intersect in at least two points. Next, we order the points of block $\overline{A_i}$, assuming that $\overline{A_1}, \ldots, \overline{A_{i-1}}$ have already been ordered in a manner that ensures that the last point y of $\overline{A_{i-1}}$ is a point in $\overline{A_i}$. We need to order the points of $\overline{A_i}$ so that y is

first and so that the last point z in $\overline{A_i}$ appears in $\overline{A_{i+1}}$. Since $|\overline{A_i} \cap \overline{A_{i+1}}| \geq 2$, we know that we have two choices for z. Thus, even if $y \in \overline{A_i} \cap \overline{A_{i+1}}$, there is still another point in the intersection to choose for z. Putting z last in $\overline{A_i}$, we may order the remaining points however we choose. This will give us a 1-overlap cycle for $(X, \mathcal{A})$, but the sequence might not be cyclic.

To show that we must be able to make a cyclic overlap cycle, note that the existence of a 1-overlap cycle for $(X, \mathcal{A})$ implies that $|A_i \cap A_{i+1}| \geq 1$ for each $i \in [b]$. Thus we have

$$|\overline{A_i} \cap \overline{A_{i+1}}| \geq v - 2k \geq 2 + 1 = 3.$$

So when we are choosing an element from $\overline{A_b}$ to place last, we want to ensure that the first element in A_1 is the same. The only fixed elements are the first element from $\overline{A_b}$ and the last element from $\overline{A_i}$, so we must have a third element in $\overline{A_b} \cap \overline{A_1}$ that can be placed last in $\overline{A_b}$ and first in $\overline{A_1}$. □

5.8.1 Steiner Triple Systems

Because any two blocks in a Steiner triple system (STS) can share at most one point in common, finding a ucycle over the blocks of an STS is clearly impossible - they would need to overlap in two points. To remedy this problem, Dewar introduces a modified ucycle structure in [13]. A **rank two universal cycle** is a ucycle on a block design in which each block is represented by just two of its elements. Since any pair of points appears in exactly one block of an STS, they completely identify a unique block in the triple system. Dewar constructs rank two ucycles for the special class of cyclic Steiner triple

systems. A **cyclic** design has automorphism group containing the cyclic group of order v, isomorphic to $\mathbb{Z}_v$, as a subgroup. Since this subgroup contains the automorphism $\pi : i \mapsto i + 1 \pmod{v}$, this implies that we can partition the blocks into classes so that within one class each block can be obtained from any other by repeated applications of π on the block elements.

Theorem 5.8.12. ([13], p. 200) *Every cyclic STS(v) with $v \neq 3$ admits a ucycle of rank two.*

One weakness in the rank k ucycle structure is that much information from the design is discarded. While the result allows us to write the list of blocks as a modified ucycle, it is not easy to recover the design from a given ucycle. If we are considering a different type of object that is easily computed from the basis representation, rank k ucycles are a suitable listing structure. However, given just two points of a block, the only way to recover the missing point is to have a lookup table at hand, which (depending on applications) may defeat the purpose of creating a compact listing in the first place. In some instances a lookup table is practical, for example if the block representatives obey the same cyclic symmetry. For general designs, however, this does not apply, and so we consider overlap cycles as a way to preserve the necessary data.

When writing out ocycles, we can list the sequence fully or we can choose to omit points that do not appear as an overlap (**hidden** points). When we omit the hidden points, we say that the cycle is written in **compressed form**. Using this concept, we can view Dewar's rank two ucycles as compressed 1-ocycles and easily obtain the following corollary to Theorem 5.8.12.

Corollary 5.8.13. *Every cyclic STS(v) with $v \neq 3$ admits a 1-ocycle.*

It is widely known that there exists a cyclic STS(v) for all $v \equiv 1, 3 \pmod 6$ except $v = 9$ (See [10], Theorem 7.3). We will consider a different class of Steiner triple systems, namely **automorphism free** (AF) Steiner triple systems, and prove the following result using recursive constructions.

Result 5.8.14. *For every $n \equiv 1, 3 \pmod 6$ with $n \geq 15$, there exists an AF STS(n) with a 1-ocycle.*

We also include two other direct constructions of 1-ocycles for a Steiner triple system of each order (Results 5.8.23 and 5.8.25), as well as another recursive construction (Result 5.8.21). While Dewar constructs rank two ucycles for all cyclic designs of each order, these direct constructions may be a simpler method of finding an ocycle when any STS(v) will do.

Our method of finding ocycles will be the same for each construction. We will first construct one long cycle out of many smaller cycles of various sizes by connecting them together. The connection of two cycles is performed as follows. When two cycles have an overlap point in common, we can connect the two cycles at that point. See Figure 5.3 for an example of connecting Cycle 1 and Cycle 2 at point x. Following Cycle 1 clockwise to the point x, we then continue on Cycle 2 clockwise until we return to x, at which point we return to Cycle 1 and continue clockwise on to our starting point.

5.8.1.1 Recursive Constructions of AF Steiner Triple Systems

It is interesting to note that the following constructions are still valid when we remove the automorphism-free requirement. However, the newly constructed STS is no longer guaranteed to be automorphism-free.

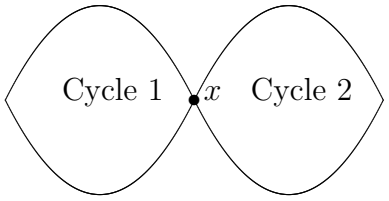

Figure 5.3: Connecting two cycles at overlap point x

Theorem 5.8.15. [38]. *Let $v \in \mathbb{Z}^+$ with $v \geq 15$. If there exists an AF STS(v), then there exists an AF STS($2v+1$).*

Construction. Let $(X, \mathcal{A})$ be some STS(v) with $v \geq 15$, and identify X with $\mathbb{Z}_v$. Then we construct a new design $(Y, \mathcal{B})$ with points:

$$Y = (\mathbb{Z}_2 \times \mathbb{Z}_v) \cup \{\infty\}$$

and blocks:

1. $\{(1,a),(1,b),(1,c)\}$ with $\{a,b,c\} \in \mathcal{A}$,
2. $\left\{(0,x),(0,y),\left(1,\frac{x+y}{2}\right)\right\}$ with $\{x,y\} \subset X$, and
3. $\{(0,x),(1,x),\infty\}$ with $x \in X$.

□

Theorem 5.8.16. [38]. *Let $v \in \mathbb{Z}^+$ with $v \geq 15$. If there exists an AF STS(v), then there exists an AF STS($2v+7$).*

Construction. Suppose that $(X, \mathcal{A})$ is an STS(v) with $v \geq 15$, and identify X with $\mathbb{Z}_v$. Then we construct a new design $(Y, \mathcal{B})$ with points:

$$Y = (\mathbb{Z}_2 \times \mathbb{Z}_v) \cup \{\infty_i \mid |i| \leq 3\}.$$

Fix $(Z, \mathcal{C})$ as some STS(7). The blocks in our new design are as follows:

1. $\{(1,i),(1,j),(1,k)\}$ with $\{i,j,k\} \in \mathcal{A}$,
2. $\{\infty_i, \infty_j, \infty_k\}$ with $\{i,j,k\} \in \mathcal{C}$,
3. $\{(0,x),(0,x+2),(0,x+6)\}$ with $x \in \mathbb{Z}_v$,
4. $\{(0,x),(1,x+y),(0,x+2y)\}$ with $\{x,y\} \subset \mathbb{Z}_v$ and $|y| > 3$, and
5. $\{\infty_i,(1,j),(0,i+j)\}$ with $|i| \leq 3$ and $j \in \mathbb{Z}_v$.

□

5.8.1.2 Base Cases

The recursive constructions given in the previous subsection require six base cases in order to construct recursively an AF STS(v) for every $v \equiv 1,3 \pmod 6$ with $v \geq 15$. These base cases are AF Steiner triple systems of order $v = 15, 19, 21, 25, 27, 33$. We also provide 1-ocycles for a non-cyclic STS(v) for $v = 9, 13$. We include a 1-ocycle for the cyclic STS(7) as it is used in the second recursive construction.

$\mathbf{v = 7}$: We use the cyclic $(7,3,1)$-design and produce the 1-ocycle:

$$\underline{2}, 1, \underline{0}, 3, \underline{4}, 2, \underline{5}, 0, \underline{6}, 4, \underline{1}, 5, \underline{3}, 6, \underline{2}$$

or, since each pair appears in exactly one block, we may omit the non-overlap points to write it in compressed form as:

$$(2, 0, 4, 5, 6, 1, 3, 2).$$

$\mathbf{v = 9}$: We use the non-cyclic design from [40] and produce the 1-ocycle:

$$\underline{0}, 1, \underline{2}, 8, \underline{5}, 3, \underline{4}, 1, \underline{7}, 8, \underline{6}, 3, \underline{0}, 4, \underline{8}, 1, \underline{3}, 7, \underline{2}, 4, \underline{6}, 1, \underline{5}, 7, \underline{0}$$

or in compressed form:

$$(0, 2, 5, 4, 7, 6, 4, 8, 3, 2, 6, 5).$$

$\mathbf{v = 13}$: We use the non-cyclic design from [38] and produce the 1-ocycle:

$$\underline{1}, 2, \underline{0}, 9, \underline{10}, 12, \underline{1}, 3, \underline{5}, 7, \underline{11}, 9, \underline{6}, 7, \underline{12}, 8, \underline{4}, 9, \underline{5}, 10, \underline{8}, 6, \underline{3}, 11, \underline{10}, 7, \underline{4},$$

$$\underline{4}, 0, \underline{3}, 7, \underline{2}, 5, \underline{12}, 3, \underline{9}, 8, \underline{2}, 10, \underline{6}, 5, \underline{0}, 12, \underline{11}, 2, \underline{4}, 6, \underline{1}, 9, \underline{7}, 0, \underline{8}, 11, \underline{1}.$$

$\mathbf{v = 15}$: We use the AF design from [40] and produce the 1-ocycle:

210	807	5b7	da4	b94
0a9	73c	742	4e5	48c
971	ce1	2dc	52a	c95
153	1db	cb0	a7e	58d
304	b82	0de	eb6	d76
461	236	e83	6ca	689
1a8	605	39d	a3b	9e2

$\mathbf{v = 19}$: We use the AF design from [34] and produce the 1-ocycle:

$1, 2, 3$	$6, 15, 8$	$15, 17, 10$	$16, 11, 14$	$19, 1, 18$	$17, 2, 19$
$3, 5, 6$	$8, 1, 9$	$10, 5, 14$	$14, 17, 7$	$18, 2, 16$	$19, 14, 8$
$6, 2, 4$	$9, 2, 11$	$14, 6, 9$	$7, 15, 9$	$16, 3, 19$	$8, 16, 7$
$4, 10, 13$	$11, 5, 13$	$9, 19, 13$	$9, 12, 18$	$19, 15, 4$	$7, 4, 3$
$13, 1, 12$	$13, 8, 17$	$13, 18, 7$	$18, 15, 11$	$4, 8, 12$	$3, 17, 18$
$12, 2, 14$	$17, 9, 5$	$7, 12, 11$	$11, 1, 10$	$12, 15, 3$	$18, 8, 5$
$14, 1, 15$	$5, 12, 19$	$11, 4, 17$	$10, 2, 8$	$3, 13, 14$	$5, 4, 1$
$15, 13, 2$	$19, 11, 6$	$17, 12, 6$	$8, 11, 3$	$14, 18, 4$	
$2, 5, 7$	$6, 13, 16$	$6, 18, 10$	$3, 9, 10$	$4, 9, 16$	
$7, 1, 6$	$16, 5, 15$	$10, 12, 16$	$10, 7, 19$	$16, 1, 17$	

$\mathbf{v = 21}$: We use the AF design from [38] to produce the 1-ocycle:

$00, \infty_1, 11$	$05, \infty_1, 16$	$12, 14, 08$	$16, 18, 03$	$01, 10, 15$	$05, 10, 14$	$05, 03, 04$
$11, \infty_0, 01$	$16, \infty_0, 06$	$08, \infty_2, 11$	$03, \infty_2, 15$	$15, 12, 18$	$14, 13, 06$	$04, 01, 07$
$01, \infty_1, 12$	$06, \infty_1, 17$	$11, 13, 17$	$15, 17, 02$	$18, 10, 02$	$06, 11, 15$	$07, 08, 06$
$12, \infty_0, 02$	$17, \infty_0, 07$	$07, \infty_2, 10$	$02, \infty_2, 14$	$02, 11, 16$	$15, 14, 07$	$06, 03, 00$
$02, \infty_1, 13$	$07, \infty_1, 18$	$10, 12, 06$	$14, 16, 01$	$16, 13, 10$	$07, 12, 16$	$00, 04, 08$
$13, \infty_0, 03$	$18, \infty_0, 08$	$06, \infty_2, 18$	$01, \infty_2, 13$	$10, 11, 03$	$16, 15, 08$	$08, 01, 03$
$03, \infty_1, 14$	$08, \infty_1, 10$	$18, 11, 05$	$13, 15, 00$	$03, 17, 12$	$08, 13, 17$	$03, 07, 02$
$14, \infty_0, 04$	$10, 00, \infty_0$	$05, \infty_2, 17$	$00, 18, 14$	$12, 11, 04$	$17, 16, 00$	$02, 04, 06$
$04, \infty_1, 15$	$\infty_0, \infty_1, \infty_2$	$17, 10, 04$	$14, 11, 17$	$04, 18, 13$	$00, 01, 02$	$06, 01, 05$
$15, \infty_0, 05$	$\infty_2, 00, 12$	$04, \infty_2, 16$	$17, 18, 01$	$13, 12, 05$	$02, 08, 05$	$05, 07, 00$

$\mathbf{v = 25}$: We use the AF design from [38] to produce the 1-ocycle:

$\infty_1, 00, 11$	$18, \infty_0, 08$	$06, \infty_3, 10$	$04, \infty_5, 11$	$07, 08, 06$	$13, 16, 14$	$14, \infty_6, 06$
$11, \infty_0, 01$	$08, \infty_1, 10$	$10, \infty_2, 07$	$11, \infty_4, 05$	$06, 03, 00$	$14, 17, 15$	$06, 01, 15$
$01, \infty_1, 12$	$10, \infty_0, 00$	$07, \infty_3, 11$	$05, \infty_5, 12$	$00, 04, 08$	$15, 18, 16$	$15, \infty_6, 07$
$12, \infty_0, 02$	$00, \infty_3, 13$	$11, \infty_2, 08$	$12, \infty_4, 06$	$08, 01, 03$	$16, 10, 17$	$07, 02, 16$
$02, \infty_1, 13$	$13, \infty_2, 01$	$08, \infty_3, 12$	$06, \infty_5, 13$	$03, 07, 02$	$17, 11, 18$	$16, \infty_6, 08$
$13, \infty_0, 03$	$01, \infty_3, 14$	$12, \infty_2, 00$	$13, \infty_4, 07$	$02, 04, 06$	$18, 12, 10$	$08, 03, 17$
$03, \infty_1, 14$	$14, \infty_2, 02$	$00, \infty_5, 16$	$07, \infty_5, 14$	$06, 01, 05$	$10, \infty_6, 02$	$17, 00, \infty_6$
$14, \infty_0, 04$	$02, \infty_3, 15$	$16, \infty_4, 01$	$14, \infty_4, 08$	$05, 07, 00$	$02, 06, 11$	$\infty_6, \infty_2, \infty_0$
$04, \infty_1, 15$	$15, \infty_2, 03$	$01, \infty_5, 17$	$08, \infty_5, 15$	$00, 04, 18$	$11, \infty_6, 03$	$\infty_0, \infty_3, \infty_1$
$15, \infty_0, 05$	$03, \infty_3, 16$	$17, \infty_4, 02$	$15, \infty_4, 00$	$18, \infty_6, 01$	$03, 07, 12$	$\infty_1, \infty_4, \infty_2$
$05, \infty_1, 16$	$16, \infty_2, 04$	$02, \infty_5, 18$	$00, 01, 02$	$01, 05, 10$	$12, \infty_6, 04$	$\infty_2, \infty_5, \infty_3$
$16, \infty_0, 06$	$04, \infty_3, 17$	$18, \infty_4, 03$	$02, 08, 05$	$10, 13, 11$	$04, 08, 13$	$\infty_3, \infty_6, \infty_4$
$06, \infty_1, 17$	$17, \infty_2, 05$	$03, \infty_5, 10$	$05, 03, 04$	$11, 14, 12$	$13, \infty_6, 05$	$\infty_4, \infty_0, \infty_5$
$17, \infty_0, 07$	$05, \infty_3, 18$	$10, \infty_4, 04$	$04, 01, 07$	$12, 15, 13$	$05, 00, 14$	$\infty_5, \infty_6, \infty_1$
$07, \infty_1, 18$	$18, \infty_2, 06$					

$\mathbf{v = 27}$: We use the AF design from [38] to produce the 1-ocycle:

$00, \infty, 10$	$19, 18, 12$	$07, \infty, 17$	$04, 00, 12$	$01, 06, 1(10)$	$0(11), 03, 17$
$10, 0(12), 01$	$12, 1(10), 16$	$17, 06, 08$	$12, 0(12), 05$	$1(10), 05, 02$	$17, 02, 0(12)$
$01, \infty, 11$	$16, 15, 10$	$08, \infty, 18$	$05, 01, 13$	$02, 07, 1(11)$	$0(12), 04, 18$
$11, 12, 10$	$10, 1(12), 1(11)$	$18, 07, 09$	$13, 00, 06$	$1(11), 06, 03$	$18, 03, 00$
$10, 19, 1(10)$	$1(11), 12, 14$	$09, \infty, 19$	$06, 02, 14$	$03, 08, 1(12)$	$00, 07, 01$
$1(10), 1(12), 11$	$14, 16, 11$	$19, 08, 0(10)$	$14, 01, 07$	$1(12), 07, 04$	$01, 08, 02$
$11, 13, 15$	$11, 19, 17$	$0(10), \infty, 1(10)$	$07, 03, 15$	$04, 09, 10$	$02, 09, 03$
$15, 17, 1(11)$	$17, 10, 18$	$1(10), 09, 0(11)$	$15, 02, 08$	$10, 08, 05$	$03, 0(10), 04$
$1(11), 19, 16$	$18, 1(11), 11$	$0(11), \infty, 1(11)$	$08, 04, 16$	$05, 0(10), 11$	$04, 0(11), 05$
$16, 17, 1(12)$	$11, 00, 02$	$1(11), 0(10), 0(12)$	$16, 03, 09$	$11, 09, 06$	$05, 0(12), 06$
$1(12), 18, 14$	$02, \infty, 12$	$0(12), \infty, 1(12)$	$09, 05, 17$	$06, 0(11), 12$	$06, 00, 07$
$14, 19, 15$	$12, 01, 03$	$1(12), 0(11), 00$	$17, 04, 0(10)$	$12, 0(10), 07$	$07, 01, 08$
$15, 1(10), 18$	$03, \infty, 13$	$00, 09, 1(11)$	$0(10), 06, 18$	$07, 0(12), 13$	$08, 02, 09$
$18, 16, 13$	$13, 02, 04$	$1(11), 08, 01$	$18, 05, 0(11)$	$13, 0(11), 08$	$09, 03, 0(10)$
$13, 1(11), 1(10)$	$04, \infty, 14$	$01, 0(10), 1(12)$	$0(11), 07, 19$	$08, 00, 14$	$0(10), 04, 0(11)$
$1(10), 17, 14$	$14, 03, 05$	$1(12), 09, 02$	$19, 06, 0(12)$	$14, 0(12), 09$	$0(11), 05, 0(12)$
$14, 10, 13$	$05, \infty, 15$	$02, 0(11), 10$	$0(12), 08, 1(10)$	$09, 01, 15$	$0(12), 06, 00$
$13, 17, 12$	$15, 04, 06$	$10, 0(10), 03$	$1(10), 07, 00$	$15, 00, 0(10)$	
$12, 15, 1(12)$	$06, \infty, 16$	$03, 0(12), 11$	$00, 05, 19$	$0(10), 02, 16$	
$1(12), 13, 19$	$16, 05, 07$	$11, 0(11), 04$	$19, 04, 01$	$16, 01, 0(11)$	

$\mathbf{v = 33}$: We use the AF design from [38] to produce the 1-ocycle:

$13, \infty_1, 02$	$\infty_2, 00, 13$	$12, 15, 0(12)$	$08, 15, 1(14)$	$01, 0(10), 08$	$02, 1(11), 10$
$02, \infty_2, 15$	$13, 01, 15$	$0(12), 19, 13$	$1(14), 12, 09$	$08, 00, 07$	$10, 0(13), 12$
$15, \infty_1, 04$	$15, 03, 17$	$13, 16, 0(13)$	$09, 16, 10$	$07, 03, 0(12)$	$12, 00, 14$
$04, \infty_2, 17$	$17, 05, 19$	$0(13), 1(10), 14$	$10, 0(10), 13$	$0(12), 0(14), 01$	$14, 02, 16$
$17, \infty_1, 06$	$19, 07, 1(11)$	$14, 17, 0(14)$	$13, 1(10), 0(11)$	$01, 0(13), 0(11)$	$16, 04, 18$
$06, \infty_2, 19$	$1(11), 09, 1(13)$	$0(14), 1(11), 15$	$0(11), 15, 19$	$0(11), 08, 02$	$18, 06, 1(10)$
$19, \infty_1, 08$	$1(13), 0(11), 10$	$15, 18, 00$	$19, 12, 0(10)$	$02, 03, 06$	$1(10), 08, 1(12)$
$08, \infty_2, 1(11)$	$10, 19, 01$	$00, 1(12), 16$	$0(10), 14, 18$	$06, 00, 05$	$1(12), 1(14), 0(10)$
$1(11), \infty_1, 0(10)$	$01, 1(10), 1(14)$	$16, 19, 01$	$18, 11, 09$	$05, 0(11), 07$	$0(10), 15, 16$
$0(10), \infty_2, 1(13)$	$1(14), 17, 00$	$01, 1(13), 17$	$09, 13, 17$	$07, 04, 02$	$16, \infty_0, 06$
$1(13), \infty_1, 0(12)$	$00, \infty_0, 10$	$17, 1(10), 02$	$17, 10, 08$	$02, 0(13), 0(12)$	$06, 11, 12$
$0(12), \infty_2, 10$	$10, 15, 1(10)$	$02, 1(14), 18$	$08, 12, 16$	$0(12), 0(11), 00$	$12, \infty_0, 02$
$10, \infty_1, 0(14)$	$1(10), 1(11), 00$	$18, 1(11), 03$	$16, 1(14), 07$	$00, 0(14), 0(13)$	$02, 1(12), 1(13)$
$0(14), \infty_2, 12$	$00, 19, 1(13)$	$03, \infty_0, 13$	$07, 11, 15$	$0(13), 08, 03$	$1(13), \infty_0, 0(13)$
$12, \infty_1, 01$	$1(13), 16, 0(14)$	$13, 18, 1(13)$	$15, 1(13), 06$	$03, 09, 0(13)$	$0(13), 18, 19$
$01, \infty_2, 14$	$0(14), \infty_0, 1(14)$	$1(13), 1(14), 03$	$06, 10, 14$	$0(13), 0(10), 04$	$19, \infty_0, 09$
$14, \infty_1, 03$	$1(14), 14, 19$	$03, 10, 19$	$14, 1(12), 05$	$04, 0(14), 05$	$09, 14, 15$
$03, \infty_2, 16$	$19, 1(10), 0(14)$	$19, 1(12), 04$	$05, 1(14), 13$	$05, 02, 0(10)$	$15, \infty_0, 05$
$16, \infty_1, 05$	$0(14), 18, 1(12)$	$04, 11, 1(10)$	$13, 1(11), 04$	$0(10), 07, 0(14)$	$05, 10, 11$
$05, \infty_2, 18$	$1(12), 15, 0(13)$	$1(10), 1(13), 05$	$04, 1(13), 12$	$0(14), 0(11), 06$	$11, \infty_0, 01$
$18, \infty_1, 07$	$0(13), 17, 1(11)$	$05, 12, 1(11)$	$12, 1(10), 03$	$06, 0(12), 0(10)$	$01, 1(11), 1(12)$
$07, \infty_2, 1(10)$	$1(11), 14, 0(12)$	$1(11), 1(14), 06$	$03, 1(12), 11$	$0(10), 03, 0(11)$	$1(12), \infty_0, 0(12)$
$1(10), \infty_1, 09$	$0(12), 16, 1(10)$	$06, 13, 1(12)$	$11, 19, 02$	$0(11), 09, 04$	$0(12), 17, 18$
$09, \infty_2, 1(12)$	$1(10), \infty_0, 0(10)$	$1(12), 10, 07$	$02, 01, 00$	$04, 08, 0(12)$	$18, \infty_0, 08$
$1(12), \infty_1, 0(11)$	$0(10), 17, 11$	$07, \infty_0, 17$	$00, 0(10), 09$	$0(12), 09, 05$	$08, 13, 14$
$0(11), \infty_2, 1(14)$	$11, 14, 0(11)$	$17, 1(12), 12$	$09, 07, 01$	$05, 08, 0(13)$	$14, \infty_0, 04$
$1(14), \infty_1, 0(13)$	$0(11), \infty_0, 1(11)$	$12, 13, 07$	$01, 05, 03$	$0(13), 07, 06$	$04, 10, 1(14)$
$0(13), \infty_2, 11$	$1(11), 11, 16$	$07, 14, 1(13)$	$03, 00, 04$	$06, 08, 09$	$1(14), 0(12), 11$
$11, 00, \infty_1$	$16, 17, 0(11)$	$1(13), 11, 08$	$04, 06, 01$	$09, 0(14), 02$	$11, 0(14), 13$
$\infty_1, \infty_0, \infty_2$	$0(11), 18, 12$				

5.8.1.3 Recursive Constructions of 1-Overlap Cycles

Result 5.8.17. *If there exists an AF STS(v) with a 1-ocycle, then there exists an AF STS($2v+1$) with a 1-ocycle when $v \geq 15$.*

Proof. Using Construction 5.8.15, we construct an overlap cycle for $(Y, \mathcal{B})$ as follows. We will construct a 1-overlap cycle for triples of type (1), and then for the triples of types (2) and (3), and finally show that these cycles can be joined.

Step 1: Triples of type (1). Let O be a 1-ocycle on $\mathcal{A}$. Then $\{1\} \oplus \mathcal{A}$ (placing a 1 as a first coordinate before each point in every block in $\mathcal{A}$) also has a 1-ocycle, given by $\{1\} \oplus O$.

Step 2: Triples of type (2). We first define the **difference** of a triple to be the smaller of $x - y$ and $y - x$ (modulo v). Then we partition the set of triples of type (2) depending on their difference d. This creates an equivalence relation on the set of triples of type (2). We will construct 1-ocycles for each equivalence class separately.

$\mathbf{d = 1}$: We have the overlap cycle (in compressed form, with hidden elements removed):

$$(0,0),(0,1),(0,2),\ldots,(0,v-1),(0,0).$$

$\mathbf{d = 2}$: Similar to $d = 1$, we have the compressed ocycle:

$$(0,0),(0,2),(0,4),\ldots,(0,v-1),(0,1),(0,3),\ldots,(0,v-2),(0,0).$$

$\mathbf{d \geq 3}$: We follow the same procedure as for $d = 1, 2$, except if $d \mid v$ the procedure will not produce a cycle that covers all triples. However, when

this happens we can repeat the process beginning with the first triple that remains unused. In this manner, we will obtain several disjoint ocycles, and every triple of type (2) with the given difference d will be covered by one of these cycles.

Note that for difference $d = 1$, every point of type $(0, x)$ for $x \in X$ appears as an overlap point. Thus for all overlap cycles associated with $d \geq 3$, we can join them to the cycle for $d = 1$. We reserve the ocycle corresponding to $d = 2$ to include with the triples of type (3).

Step 3: Triples of type (3). We construct an ocycle to include triples of type (2) with $d = 2$ and triples of type (3) as follows. First, connect pairs of triples of the form:

$$(1, x+1) \quad (0, x) \quad (0, x+2)$$

and

$$(0, x+2) \quad \infty \quad (1, x+2).$$

Then we may use all of these pairs to form the ocycle (displayed as a circle):

$$\begin{array}{cccccc} \underline{(1,1)}, & (0,0), & \underline{(0,2)}, & \infty, & \underline{(1,2)}, & (0,1), \\ \infty, & & & & & \underline{(0,3)}, \\ \underline{(0,1)}, & (0,v-1), & \underline{(1,0)}, & \ldots, & \underline{(1,3)}, & \infty, \end{array}$$

This cycle accounts for v of these pairs of triples, and since no triple is covered twice it must cover all triples of type (2) with $d = 2$ and all triples of type (3).

To connect all of the constructed cycles, we note that the cycle created in

Step 3 contains the points $(0, x)$ and $(1, x)$ for every $x \in X$ as an overlap. Thus we can connect the cycle created in Step 1 to this cycle, as well as the cycle created in Step 2. This produces one long 1-ocycle that covers all triples. □

Result 5.8.18. *If there exists an AF STS(v) with a 1-ocycle, then there exists an AF STS($2v + 7$) with a 1-ocycle, when $v \geq 15$.*

Proof. Using Construction 5.8.16, we will find ocycles for subsets of blocks, and show that they can be combined to form one long ocycle for the entire design.

Step 1: Triples of type (1). Let O be a 1-ocycle on $\mathcal{A}$. Then $\{1\} \oplus \mathcal{A}$ also has a 1-ocycle, given by $\{1\} \oplus O$.

Step 2: Triples of type (3). We construct one long cycle (displayed as a circle):

$$\begin{array}{cccccc} \underline{(0,0)}, & (0,6), & \underline{(0,2)}, & (0,8), & \underline{(0,4)}, \\ (0,4), & & & & \ldots, \\ \underline{(0,v-2)}, & \ldots, & \underline{(0,1)}, & (0,5), & \underline{(0,v-1)}, \end{array}$$

Note that since v must always be odd, we see the point $(0, x)$ for every $x \in \mathbb{Z}_v$ as an overlap point in this cycle.

Step 3: Triples of type (4) with $|y| > 4$. We start by creating the ocycle(s) (in compressed form, with hidden elements removed):

$$(0,0), (0,2y), (0,4y), (0,6y), \ldots, (0,0).$$

Note that if $\gcd(y, v) = 1$ this creates one cycle, but otherwise will create several disjoint cycles. These cycles contain all triples of type (4) associated with a particular y. Note also that in each cycle, all of the points $(0, x)$ for every $x \in \mathbb{Z}_v$ appear as overlaps. Thus we can connect all of these cycles to the cycle from Step 2.

Step 4: Triples of type (4) with $|y| = 4$ and type (5) with $i = -3$. We begin by pairing up blocks as follows so as to partition $\mathcal{B}$:

$$\{(0, x), \quad (0, x + 8), \quad (1, x + 4)\}$$

and

$$\{(1, x + 4), \quad \infty_{-3}, \quad (0, x + 1)\}.$$

We can connect up these pairs in order, starting with the pair that begins $(0, 0)$ and then moving to the pair that begins $(0, 1)$, and so on. We will eventually end with the pair starting $(0, v - 1)$ and ending $(0, 0)$. Thus we have an overlap cycle. Note that in this cycle the points $(0, x)$ and $(1, x)$ appear as overlap points for every $x \in \mathbb{Z}_v$.

Step 5: Triples of type (2). The triples of type (2) correspond to an STS(7). We have shown in Section 5.8.1.2 that a 1-ocycle exists for the unique STS(7). We will use the cycle from Step 4 to join the triples of type (2). If we break the cycle from Step 4 between the blocks

$$\{(0, v - 8), \quad (0, 0), \quad (1, v - 4)\} \quad \text{and}$$
$$\{(1, v - 4), \quad \infty_{-3}, \quad (0, v - 7)\},$$

and also between the blocks

$$\{(1,3), \quad \infty_{-3}, \quad (0,0)\} \quad \text{and}$$
$$\{(0,0), \quad (0,8), \quad (1,4)\},$$

then we now have two 1-overlap paths:

$$\underline{(0,0)}, (0,8), \underline{(1,4)}, \ldots, \underline{(0,v-8)}, (0,0), \underline{(1,v-4)}$$

and

$$\underline{(1,v-4)}, \infty_{-3}, \underline{(0,v-7)}, \ldots, \underline{(1,3)}, \infty_{-3}, \underline{(0,v-7)}.$$

We can swap the order of the last two elements in the first path, and swap the order of the first two and the order of the last two elements in the second path to obtain the following two 1-ocycles:

$$\underline{(0,0)}, (0,8), \underline{(1,4)}, \ldots, \underline{(0,v-8)}, (1,v-4), \underline{(0,0)}$$

and

$$\underline{\infty_{-3}}, (1,v-4), \underline{(0,v-7)}, \ldots, \underline{(1,3)}, (0,v-7), \underline{\infty_{-3}}.$$

Now we have ∞_{-3} as an overlap point in the second cycle and so we can join this ocycle to the STS(7) ocycle (which contains every point ∞_i as an overlap point).

Step 6: Triples of type (5) with $i \neq -3$. We construct three separate ocycles as follows. For $k \in \{-2, 0, 2\}$, construct the cycle:

$$(0,0), (1,k), (0,1), (1,k+1), \ldots$$

When $k = -2$, this covers all triples of type

$$\{(0,x), (1,x-2), \infty_2\} \text{ and } \{(0,x), (1,x-3), \infty_3\}.$$

When $k = 0$, this covers all triples of type

$$\{(0, x), (1, x), \infty_0\} \text{ and } \{(0, x), (1, x - 1), \infty_1\}.$$

When $k = 2$, this covers all triples of type

$$\{(0, x), (1, x + 2), \infty_{-2}\} \text{ and } \{(0, x), (1, x + 1), \infty_{-1}\}.$$

These three cycles cover all triples of type (5) with $i \neq -3$.

The cycles from Steps 2, 3, 5, and 6 all contain the point $(0, x)$ for every $x \in \mathbb{Z}_v$, and so can be connected. The cycles from Steps 1, 5, and 6 all contain the point $(1, x)$ for every $x \in \mathbb{Z}_v \setminus \{v - 4\}$, and so can be connected. Since the cycle from Step 5 appears in both cases, these two long cycles can also be connected. Thus, all triples are contained in one of the connected cycles, and so we have a 1-ocycle that covers all blocks. □

We are now ready to prove Result 5.8.14.

Proof of Result 5.8.14. We proceed by induction on n. For $n = 15, 19, 21, 25, 27,$ and 33, we have shown ocycles in Section 5.8.1.2.

For $n \geq 37$ and $n \equiv 1 \pmod{12}$, there exists $v \equiv 3 \pmod 6$ so that $n = 2v + 7$. Note that $n \geq 37$ implies that $v \geq 15$. Thus we use Result 5.8.18 to find the STS(n).

For $n \geq 39$ and $n \equiv 3 \pmod{12}$, there exists $v \equiv 1 \pmod 6$ so that $n = 2v + 1$. Note that $n \geq 39$ implies that $v \geq 19$. Thus we use Result 5.8.17 to find the STS(n).

For $n \geq 31$ and $n \equiv 7 \pmod{12}$, there exists $v \equiv 3 \pmod 6$ so that $n = 2v + 1$. Note that $n \geq 31$ implies that $v \geq 15$, so we use Result 5.8.17 to find the STS(n).

For $n \geq 45$ and $n \equiv 9 \pmod{12}$, there exists $v \equiv 1 \pmod{6}$ so that $n = 2v + 7$. Note that $n \geq 45$ implies that $v \geq 19$, so we use Result 5.8.18 to find the STS(n). □

Corollary 5.8.19. *For every $n \geq 15$ with $n \equiv 1, 3 \pmod{6}$, there exists an AF STS(n) with a rank two ucycle.*

Proof. Using Result 5.8.14, we construct an AF STS(n) with a 1-ocycle. The 1-ocycle in compressed form is a rank two ucycle. □

5.8.1.4 Other STS Constructions with Overlap Cycles

In this section, we look at several other known constructions for Steiner triple systems, and show their corresponding 1-ocycle constructions.

Recall Construction 3.8.9, the direct product of two Steiner triple systems. An interesting consequence of the direct product is the following theorem.

Theorem 5.8.20. ([10], Lemma 7.12) *The automorphism group of the direct product of two triple systems is the direct product of their automorphism groups.*

This theorem implies another method for constructing AF Steiner triple systems that admit ocycles. Beginning with two AF Steiner triple systems with corresponding 1-ocycles, we can use the following result to construct a 1-ocycle on their AF direct product.

Result 5.8.21. *If there exists an STS(u) with a 1-overlap cycle and an STS(v) with a 1-overlap cycle, then there exists an STS(uv) with a 1-ocycle.*

Proof. Let $(X, \mathcal{A})$ be an STS(u) and $(Y, \mathcal{B})$ be an STS(v) that admit the 1-ocycles $O(\mathcal{A})$ and $O(\mathcal{B})$, respectively. We construct an STS(uv) using Con-

struction 3.8.9 (the direct product). For each $i \in \mathbb{Z}_u$, we have a 1-ocycle covering the triples of type (1), namely $i \oplus O(\mathcal{B})$. Similarly, for each $a \in \mathbb{Z}_v$, we have a 1-ocycle covering the triples of type (2): $O(\mathcal{A}) \oplus a$. Lastly, for each $A = \{i, j, k\} \in \mathcal{A}$ and each $B = \{a, b, c\} \in \mathcal{B}$, we can construct the following 1-ocycle:

$$\{(i,a), \quad (j,b), \quad (k,c)\}$$
$$\{(k,c), \quad (j,a), \quad (i,b)\}$$
$$\{(i,b), \quad (j,c), \quad (k,a)\}$$
$$\{(k,a), \quad (j,b), \quad (i,c)\}$$
$$\{(i,c), \quad (j,a), \quad (k,b)\}$$
$$\{(k,b), \quad (j,c), \quad (i,a)\}$$

To connect cycles covering triples of types (1) and (2), we connect wherever possible. Starting with $0 \oplus O(\mathcal{B})$, we connect all cycles over triples of type (2). Then, starting with an arbitrary, already connected, cycle over triples of type (2), we repeat the process by adding cycles over triples of type (1) wherever possible. We continue this process of extending our cycle until we no longer are able to add any more cycles.

We will always be able to continue to connect cycles, except when all cycles are connected, or:

1. there exists $i \in \mathbb{Z}_u$ that never appears as an overlap point in $O(\mathcal{A})$, and/or,

2. there exists $a \in \mathbb{Z}_v$ that never appears as an overlap point in $O(\mathcal{B})$.

If we have both cases, then we choose a block $A \in \mathcal{A}$ containing i and a block $B \in \mathcal{B}$ containing a. Then we arrange the cycle covering the triples from $A \times B$ to begin with (i, a). Since two points of A must appear as overlap points in

$O(\mathcal{A})$ and i is not one of them, we must have that j and k are overlap points in $O(\mathcal{A})$. Similarly, b and c must be overlap points in $O(\mathcal{B})$. Thus we can connect the cycle for $A \times B$ to the cycles $k \oplus O(\mathcal{B})$ and $O(\mathcal{A}) \oplus c$. Note that since each block can only contain one hidden element, this process will never use a block from $\mathcal{A}$ or $\mathcal{B}$ more than once. If only case (1) or case (2) holds (but not both), this process is repeated with an arbitrary choice of block from $\mathcal{B}$. □

We now consider the direct constructions of Steiner triple systems.

Theorem 5.8.22. ([6], Prop. 8.2.1, p. 110) *If $n \equiv 3 \pmod 6$, there exists an STS(n).*

Construction. Suppose that $n \equiv 3 \pmod 6$; then $n = 3m$ for some m odd. The point set is made up of three copies of the integers modulo m. Formally:

$$X = \mathbb{Z}_3 \times \mathbb{Z}_m.$$

Blocks are of two types:

1. $\{(a, i), (a, j), (a + 1, k)\}$ with $i + j = 2k$ for each $a \in \mathbb{Z}_3$
2. $\{(0, i), (1, i), (2, i)\}$ for each $i \in \mathbb{Z}_m$

□

Result 5.8.23. *For $n \equiv 3 \pmod 6$ with $n > 3$, there exists an STS(n) that admits a 1-ocycle.*

Proof. We will use Construction 5.8.22 to create an STS(n), then construct 1-ocycles to cover each type of triples, and finally show how to connect them into one large cycle. First, note that we have $m \geq 3$ since $n > 3$, and so there

exists at least three triples of each kind.

Step 1: Triples of type (1) with $a = 1$. Define the **distance** for the triple $\{(1,i),(1,j),(2,k)\}$ to be the value $\min\{i-j, j-i\}$, where subtraction is done in the group $\mathbb{Z}_m$. Partition the blocks of type (1) into classes so that the blocks $\{(1,i),(1,j),(2,k)\}$ and $\{(1,r),(1,s),(2,t)\}$ are in the same class if and only if they have the same distance. This defines an equivalence relation on the set of blocks of type (1) with $\frac{m-1}{2}$ different equivalence classes. Create a cycle using the set of blocks having the form $\{(1,i),(1,i+1),(2,i+\frac{m-1}{2}+1)\}$ as shown below in compressed form:

$$(1,0)(1,1)(1,2)\cdots(1,m-1)(1,0).$$

Create similar (possibly shorter) cycles using the blocks within each other equivalence class. This creates at least one cycle, if not several disjoint cycles, for each equivalence class. Since the first cycle created (using blocks with distance 1) has every point $(1,i)$ as an overlap point, we can combine all of these cycles to make one long cycle.

Step 2: Triples of type (1) with $a = 2$. Repeat as in Step 1. We pay careful attention to attach the cycle for distance 2 blocks at the point $(2,0)$. Note that this is possible since distance 2 also creates one long cycle covering the entire equivalence class, as m must be odd. Now we may be assured that the cycle corresponding to distance 2 does not have any cycles attached at the overlap point $(2,1)$ between the blocks $\{(2,m-1),(0,0),(2,1)\}$ and $\{(2,1),(0,2),(2,3)\}$. Then, when we have combined all blocks of type (1) with $a = 2$ to make a cycle, we convert the cycle to a string by cutting it

between these two blocks and then reversing the order of the last two points. In other words, we now have a string that begins with $(2,1)(0,2)(2,3)$ and ends with $(2,m-1)(2,1)(0,0)$.

Step 3: Triples of type (2) and (1) with $a = 0$. Repeat as in Step 1 excluding the equivalence class with distance 2. For these excluded blocks, we partition the set of blocks of type (1) and (2) into sets of size two by grouping together:

$$\{(0,i),(2,i),(1,i)\} \text{ and } \{(1,i),(0,i-1),(0,i+1)\}.$$

Clearly the blocks in each set of size two can form a 1-overlap string, and then we can combine each of these strings to obtain a 1-ocycle of the form (displayed in a circle):

$$\begin{array}{ccccccccc} \underline{(0,1)} & & (2,1) & \underline{(1,1)} & (0,0) & \underline{(0,2)} & \cdots & & \underline{(0,i)} \\ (0,m-1) & & & & & & & & (2,i) \\ \underline{(1,0)} & & (2,0) & \underline{(0,0)} & \cdots & \underline{(0,i+1)} & (0,i-1) & & \underline{(1,i)} \end{array}$$

or in compressed form

$$(0,1)(1,1)(0,2)(1,2)\cdots(0,i)(1,i)(0,i+1)(1,i+1)\cdots(0,0)(1,0)(0,1).$$

Step 4: Combining the triples from Step 1 and Step 3. Since the cycles created in Step 1 and Step 3 both contain every point $(1,i)$ as an overlap point, we can combine these two cycles. More importantly, we have a choice of where to combine the cycles, since we have at least two choices for an overlap point $(1,i)$. We choose to combine the two cycles at an overlap point other

than $(1,1)$. Then, we can create a string from this cycle by cutting it between the blocks $\{(0,1),(2,1),(1,1)\}$ and $\{(1,1),(0,0),(0,2)\}$ (from cycle from Step 3), and reversing the order of the first two and the last two elements. In other words, we now have a string that begins with $\{(2,1),(0,1),(1,1)\}$ and ends with $\{(1,1),(0,2),(0,0)\}$.

To create our final 1-ocycle, we recall that our string from Step 2 also begins with the point $(2,1)$ and ends with the point $(0,0)$, and so we can combine these two strings into one large cycle by reversing the order of the string from Step 4. □

Theorem 5.8.24. ([46], p. 128) *If $n \equiv 1 \pmod 6$, then there is an STS(n).*

Construction. If $n \equiv 1 \pmod 6$, then $n = 6t+1$ for some $t \in \mathbb{Z}$. We define the point set as

$$Y = (\mathbb{Z}_{2t} \times \mathbb{Z}_3) \cup \{\infty\}.$$

Then we define three types of blocks:

1. $A_x = \{(x,0),(x,1),(x,2)\}$ for $0 \leq x \leq t-1$.

2. $B_{x,y,i} = \{(x,i),(y,i),(x \circ y, i+1)\}$ for each $x, y \in \mathbb{Z}_{2t}$ with $x < y$ and each $i \in \mathbb{Z}_3$, with the operation $x \circ y = \pi(x+y \pmod{2t})$ where

$$\pi(z) = \begin{cases} z/2, & \text{if } z \text{ is even,} \\ (z+2t-1)/2, & \text{if } z \text{ is odd.} \end{cases}$$

3. $C_{x,i} = \{\infty,(x+t,i),(x,i+1)\}$ for each $0 \leq x \leq t-1$ and $i \in \mathbb{Z}_3$.

□

Result 5.8.25. *For $n \equiv 1 \pmod 6$ with $n > 1$, there exists an STS(n) that admits a 1-overlap cycle.*

Proof. We will use Construction 5.8.24 to construct an STS(n). Then we show how to construct disjoint cycles for most triples of type (2), as well as disjoint cycles for triples of types (1) and (3), and finally we show how to combine them to make one large 1-ocycle containing all triples.

Step 1: Triples of type (2). The triples of type (2) can be partitioned based on the pair $\{(x, i), (y, i)\}$. Similar to Result 5.8.23, we define the **distance** of the triple to be the smaller of $x - y$ and $y - x$ (modulo $2t$). Then, we can partition the set of triples of type (2) into classes that share the same distance for each difference $k < t$. Following the method from Result 5.8.23 (Step 1), we can create disjoint cycles that contain all of these triples. Note that the triples corresponding to distance 1 make one long cycle for each second coordinate. This cycle is (in compressed form):

$$(0, i)(1, i)(2, i) \dots (2t - 1, i) \text{ for } i \in \mathbb{Z}_3.$$

These cycles contain every point from $\mathbb{Z}_{2t} \times \{i\}$ as an overlap point, and so we can hook up all of them to make three long cycles - one for each $i \in \mathbb{Z}_3$. These cycles cover all triples of type (2) except those with distance t.

Step 2: Triples of types (1), (2), (3). We begin by partitioning the triples of type (3) into classes that contain the following blocks:

$$\{\infty, (x + t, 0), (x, 1)\}, \{\infty, (x + t, 1), (x, 2)\}, \{\infty, (x + t, 2), (x, 0)\}.$$

Note that no other triples of type (3) contain any points with x or $x+t$ as a first coordinate. This set of blocks has a corresponding triple of type (1):

$$\{(x,0),(x,1),(x,2)\}.$$

It also has a corresponding triple of type (2) with distance t:

$$\{(x,i),(x+t,i),(x\circ(x+t),i+1)\} \text{ for } i\in\mathbb{Z}_3.$$

Using these blocks and defining $y=x+t$, we create the following cycle (displayed as a circle):

$$\begin{array}{cccccc} \underline{x2} & (x\circ y)0 & \underline{y2} & x0 & \underline{\infty} & x1 \\ \infty & & & & & \underline{y0} \\ \underline{y1} & (x\circ y)2 & \underline{x1} & x2 & \underline{x0} & (x\circ y)1 \end{array}$$

or in compressed form

$$x2 \quad y2 \quad \infty \quad y0 \quad x0 \quad x1 \quad y1 \quad x2\ .$$

This creates a set of disjoint cycles that cover all of the remaining triples.

To combine all of our cycles and create our final 1-ocycle, we note that the cycles from Step 2 each have at least one overlap point of the form $(x,1)$ with $x\in\mathbb{Z}_t$, and so we can hook these cycles all up to the cycle from Step 1 that corresponds to $i=1$. Also, each of the cycles from Step 2 has overlap points (x,i) corresponding to each $i=1,2$ and each $x\in\mathbb{Z}_t$, so we can connect the remaining two cycles from Step 1. □

We may use these direct constructions to provide an alternative proof that there exists a 1-ocycle for an STS of each possible order.

Theorem 5.8.26. *For every $n \equiv 1, 3 \pmod 6$, there exists an STS(n) that admits a 1-ocycle.*

Proof. For $n \equiv 3 \pmod 6$ with $n \geq 7$, we apply Result 5.8.23 to obtain the desired system. For $n \equiv 1 \pmod 6$ with $n \geq 7$, we apply Result 5.8.25 to obtain the desired system. □

Corollary 5.8.27. *For every $n \geq 7$ with $n \equiv 1, 3 \pmod 6$, there exists an STS(n) with a rank two ucycle.*

Proof. Using Theorem 5.8.26, we construct an STS(n) with a 1-ocycle. The 1-ocycle in compressed form is a rank two ucycle. □

5.8.2 Steiner Quadruple Systems

A 3-$(v, 4, 1)$-design is known as a *Steiner quadruple system of order v*, or SQS(v). The existence of Steiner quadruple systems was completely determined by H. Hanani in 1960.

Theorem 5.8.28. [23] *An SQS(v) exists if and only if $v \equiv 2, 4 \pmod 6$.*

To prove this result, Hanani used six different recursive constructions and various base cases. Using these constructions, we create 1-overlap ucycles for each SQS(v).

5.8.2.1 Hanani's Constructions

To begin, we make a note that Hanani defines two systems of unordered pairs in [23]. These systems, referred to as $P_\alpha(m)$ and $\overline{P}_\xi(m)$, are necessary for

his constructions of Steiner quadruple systems. However, our methods of constructing ocycles do not depend on the precise definitions of these sets, so we refer the reader to [23] for a complete definition, which will be omitted here. Also, within each construction, once a condition is imposed on a variable the same conditions are to be upheld for the remainder of the construction.

The first construction produces an SQS($2n$) from an SQS(n).

Construction 5.8.29. *Let $(X, \mathcal{B})$ be an SQS(n). Let $X = \{0, 1, \ldots, n-1\}$, and define a new point set $\mathcal{Y} = \{0, 1\} \times X$. The blocks on $\mathcal{Y}$ that form an SQS($2n$) are as follows.*

1. *$\{a_1x, a_2y, a_3z, a_4t\}$ where $\{x, y, z, t\} \in \mathcal{B}$ and $a_1 + a_2 + a_3 + a_4 \equiv 0 \pmod 2$.*
2. *$\{0j, 0j', 1j, 1j'\}$ for $j \neq j'$.*

The second construction produces an SQS($3n-2$) from an SQS(n).

Construction 5.8.30. *Let $(X, \mathcal{B})$ be an SQS(n). Define*

$$X = \{0, 1, 2, \ldots, n-2\} \cup \{A\}.$$

Let $\mathcal{B} = \mathcal{B}_A \cup \overline{\mathcal{B}_A}$, where $\mathcal{B}_A$ denotes all blocks containing A, and $\overline{\mathcal{B}_A}$ denotes all blocks not containing the point A. We construct new blocks on the set

$$\mathcal{Y} = (\{0, 1, 2\} \times \{0, 1, 2, \ldots, n-2\}) \cup \{A\}.$$

Note that $\mathcal{Y}$ has cardinality $1 + 3(n-1) = 3n-2$. The new blocks are as follows.

1. *$\{a_1x, a_2y, a_3z, a_4t\}$ for $\{x, y, z, t\} \in \overline{\mathcal{B}_A}$ and $a_1+a_2+a_3+a_4 \equiv 0 \pmod 3$.*

2. $\{A, b_1u, b_2v, b_3w\}$ *for* $\{A, u, v, w\} \in \mathcal{B}_A$ *and* $b_1 + b_2 + b_3 \equiv 0 \pmod 3$.

3. $\{iu, iv, (i+1)w, (i+2)w\}$ *for* $i \in \{0, 1, 2\}$ *and* $\{A, u, v, w\} \in \mathcal{B}_A$.

4. $\{ij, ij', (i+1)j, (i+1)j'\}$ *for* $i \in \{0, 1, 2\}$, $j, j' \in \{0, 1, 2, \ldots, n-2\}$ *and* $j \neq j'$.

5. $\{A, 1j, 2j, 3j\}$ *for* $j \in \{0, 1, 2, \ldots n-2\}$.

The third construction creates an SQS($3n-8$) from an SQS(n) if $n \equiv 2 \pmod{12}$.

Construction 5.8.31. *Let* $(X, \mathcal{B})$ *be an SQS*(n) *with* $n \equiv 2 \pmod{12}$. *Let*

$$X = \{0, 1, 2, \ldots, n-5\} \cup \{Ah : h \in \{0, 1, 2, 3\}\}.$$

We will make the assumption that $\{A0, A1, A2, A3\}$ *is a block in* $\mathcal{B}$. *Define*

$$\mathcal{Y} = (\{0, 1, 2\} \times \{0, 1, 2, \ldots, n-5\}) \cup \{Ah : h \in \{0, 1, 2, 3\}\}.$$

Note that $\mathcal{Y}$ *has cardinality* $3(n-4) + 4 = 3n - 8$. *We construct blocks on* $\mathcal{Y}$ *as follows.*

1. $\{A0, A1, A2, A3\}$

2. $\{ix, iy, iz, it\}$ *where* $\{x, y, z, t\} \in \mathcal{B} \backslash \{A0, A1, A2, A3\}$. *(If one of* x, y, z, t *is* Ah, *omit the* i.*) We denote this operation by* $i \oplus (\mathcal{B} \backslash \{A0, A1, A2, A3\})$.

3. $\{Aa_1, 0a_2, 1a_3, 2a_4\}$ *where* $a_1 + a_2 + a_3 + a_4 \equiv 0 \pmod{n-4}$.

4. $\{(i+2)b_3, i(b_1 + 2k + 1 + i(4k+2) - d), i(b_1 + 2k + 2 + i(4k+2) + d), (i+1)b_2\}$ *where* $n - 4 = 12k + 10$, $b_1 + b_2 + b_3 \equiv 0 \pmod{n-4}$, *and* $d \in \{0, 1, \ldots, 2k\}$.

5. $\{ir_\alpha, is_\alpha, (i+1)r'_\alpha, (i+1)s'_\alpha\}$ *where* $[r_\alpha, s_\alpha], [r'_\alpha, s'_\alpha] \in P_\alpha(6k+5)$ *(possibly the same) where* $\alpha = 4k+2, 4k+3, \ldots, 12k+8$.

The fourth construction makes an SQS($3n-4$) from an SQS(n) whenever $n \equiv 10 \pmod{12}$.

Construction 5.8.32. *Let* $(X, \mathcal{B})$ *be an SQS*(n) *with* $n \equiv 10 \pmod{12}$. *Let*

$$X = \{0, 1, 2, \ldots, n-3\} \cup \{A0, A1\}.$$

Define

$$\mathcal{Y} = (\{0,1,2\} \times \{0, 1, \ldots, n-3\}) \cup \{A0, A1\}.$$

Note that $\mathcal{Y}$ *has cardinality* $3(n-2)+2 = 3n-4$. *We construct blocks on* $\mathcal{Y}$ *as follows.*

1. $\{ix, iy, iz, it\}$ *where* $\{x, y, z, t\} \in \mathcal{B}$, *or* $i \oplus \mathcal{B}$. *(If one of* x, y, z, t *is* Ah, *omit the* i.*)*

2. $\{Aa_1, 0a_2, 1a_3, 2a_4\}$ *where* $a_1 + a_2 + a_3 + a_4 \equiv 0 \pmod{n-2}$ *and also* $a_1 \in \{0,1\}$ *and* $a_2, a_3, a_4 \in \{0, 1, 2, \ldots, n-2\}$.

3. $\{(i+2)b_3, i(b_1+2k+1+i(4k+2)-d), i(b_1+2k+2+i(4k+2)+d), (i+1)b_2\}$ *where* $n = 12k+10$, $b_1 + b_2 + b_3 \equiv 0 \pmod{n-2}$, *and* $d = 0, 1, \ldots, 2k$.

4. $\{ir_\alpha, is_\alpha, (i+1)r'_\alpha, (i+1)s'_\alpha\}$ *where* $[r_\alpha, s_\alpha], [r'_\alpha, s'_\alpha] \in P_\alpha(6k+4)$ *(possibly the same) where* $\alpha = 4k+2, 4k+3, \ldots, 12k+6$.

The fifth construction produces an SQS($4n-6$) from an SQS(n).

Construction 5.8.33. *Let* $(X, \mathcal{B})$ *be an SQS*(n) *with*

$$X = \{0, 1, \ldots, n-2\} \cup \{A0, A1\}.$$

Define

$$\mathcal{Y} = (\{0,1\} \times \{0,1\} \times \{0,1,\ldots,n-3\}) \cup \{A0, A1\}.$$

Note that $\mathcal{Y}$ *has cardinality* $(2)(2)(n-2)+2 = 4n-6$. *We construct blocks on* $\mathcal{Y}$ *as follows:*

1. $h \oplus i \oplus \mathcal{B}$ *where* $h \in \{0,1\}$ *and* $i \in \{0,1\}$, *and we ignore the prefix* hi *from the points* $A0, A1 \in X$.

2. $\{A\ell, 00(2c_1), 01(2c_2-\epsilon), 1\epsilon(2c_3+\ell)\}$ *where* $\ell, \epsilon \in \{0,1\}$ *and, defining* $n = 2k$, *we also have* $c_1 + c_2 + c_3 \equiv 0 \pmod{k}$.

3. $\{A\ell, 00(2c_1+1), 01(2c_2-1-\epsilon), 1\epsilon(2c_3+1-\ell)\}$.

4. $\{A\ell, 10(2c_1), 11(2c_2-\epsilon), 0\epsilon(2c_3+1-\ell)\}$.

5. $\{A\ell, 10(2c_1+1), 11(2c_2-1-\epsilon), 0\epsilon(2c_3+\ell)\}$.

6. $\{h0(2c_1+\epsilon), h1(2c_2-\epsilon), (h+1)0\overline{r}_{c_3}, (h+1)0\overline{s}_{c_3}\}$ *where* $[\overline{r}_{c_3}, \overline{s}_{c_3}] \in \overline{P}_{c_3}(k)$ *and* $c_3 \in \{0,1,\ldots,k-1\}$.

7. $\{h0(2c_1-1+\epsilon), h1(2c_2-\epsilon), (h+1)1\overline{r}_{c_3}, (h+1)1\overline{s}_{c_3}\}$.

8. $\{h0(2c_1+\epsilon), h1(2c_2-\epsilon), (h+1)1\overline{r}_{k+c_3}, (h+1)1\overline{s}_{k+c_3}\}$.

9. $\{h0(2c_1-1+\epsilon), h1(2c_2-\epsilon), (h+1)0\overline{r}_{k+c_3}, (h+1)0\overline{s}_{k+c_3}\}$.

10. $\{h0r_\alpha, h0s_\alpha, h1r'_\alpha, h1s'_\alpha\}$ *with* $[r_\alpha, s_\alpha], [r'_\alpha, s'_\alpha] \in P_\alpha(k)$ *where we have* α *from the set* $\{0,1,\ldots,n-4\}$.

The sixth and final construction produces an SQS($12n-10$) from an SQS(n), and begins with the constructions of an SQS(14) and an SQS(38).

SQS(14) (listed as a 1-ocycle):

2043	$0BC2$	$C841$	$38B5$	$0BD6$	$B9CA$	$259C$
3162	$28B1$	$19B4$	$5AC3$	$6BC1$	$A05B$	$CA68$
$25D3$	1250	4095	$36AD$	$18D6$	$B79D$	$80B9$
$37A2$	$03B1$	5134	$DBC3$	$629D$	$DA7C$	9158
$28C3$	1460	4265	$36C4$	$DB51$	$CD40$	8249
3560	0781	$57A4$	$48B6$	$17C5$	$08A4$	9368
$07C3$	$19D0$	$48D5$	$60A7$	$51A6$	$4B07$	$87A9$
$38D0$	$0AC1$	$57D0$	7196	$69B5$	$749C$	$9CD8$
$09A3$	$19A2$	$08C5$	$62C7$	$5C6D$	$CDA4$	$87CB$
$39B2$	$2CD1$	$5BC4$	$73B6$	$D59A$	$4B2D$	$B17A$
2680	$17D3$	4783	$64D7$	$A26B$	$D872$	$A964$
0792	$38A1$	$3D49$	7586	$B34A$	$27B5$	$4AD1$
$2AD0$	$139C$	9573	$69C0$	$A8DB$	$58A2$	1472

SQS(38):

We identify the points from the SQS(14) with the set

$$X = \{0, 1, 2, \ldots, 11\} \cup \{A0, A1\},$$

and let $\mathcal{B}$ be the set of blocks form the SQS(14) on X. Then we define our new point set $\mathcal{Y}$ as

$$\mathcal{Y} = (\{0, 1, 2\} \times \{0, 1, 2, \ldots, 11\}) \cup \{A0, A1\}.$$

Note that $\mathcal{Y}$ has cardinality $(3)(12) + 2 = 38$. The blocks on $\mathcal{Y}$ are as follows:

1. $i \oplus \mathcal{B}$, where we omit the prefix i for $A0$ or $A1$.

2. $\{Ah, 0b_1, 1b_2, 2(b_3+3h)\}$, where $b_1+b_2+b_3 \equiv 0 \pmod{12}$ and $h \in \{0, 1\}$.

3. $\{i(b_1+4+i), i(b_1+7+i), (i+1)b_2, (i+2)b_3\}$.

4. $\{ij, (i+1)(j+6\epsilon), (i+2)(6\epsilon-2j+1), (i+2)(6\epsilon-2j-1)\}$ where $\epsilon \in \{0,1\}$.

5. $\{ij, (i+1)(j+6\epsilon), (i+2)(6\epsilon-2j+2), (i+2)(6\epsilon-2j-2)\}$.

6. $\{ij, (i+1)(j+6\epsilon-3), (i+2)(6\epsilon-2j+1), (i+2)(6\epsilon-2j+2)\}$.

7. $\{ij, (i+1)(j+6\epsilon+3), (i+2)(6\epsilon-2j-1), (i+2)(6\epsilon-2j-2)\}$.

8. $\{ij, i(j+6), (i+1)(j+3\epsilon), (i+1)(j+6+3\epsilon)\}$.

9. $\{i(2g+3\epsilon), i(2g+6+3\epsilon), i'(2g+1), i'(2g+5)\}$ for $i' \neq i$ and g from the set $\{0,1,2,3,4,5\}$.

10. $\{i(2g+3\epsilon), i(2g+6+3\epsilon), i'(2g+2), i'(2g+4)\}$.

11. $\{ij, i(j+1), (i+1)(j+3e), (i+1)(j+3e+1)\}$ for $e = 0,1,2,3$.

12. $\{ij, i(j+2), (i+1)(j+3e), (i+1)(j+3e+2)\}$.

13. $\{ij, i(j+4), (i+1)(j+3e), (i+1)(j+3e+4)\}$.

14. $\{ir_\alpha, is_\alpha, i'r'_\alpha, i's'_\alpha\}$ where $[r_\alpha, s_\alpha], [r'_\alpha, s'_\alpha] \in P_\alpha(6)$ for $\alpha = 4,5$.

Construction 5.8.34. *Let $(X, \mathcal{B})$ be an SQS(n). Let*

$$X = \{B\} \cup \{0,1,2,\ldots,n-2\}.$$

Let $\mathcal{B}_B$ be the subset of $\mathcal{B}$ with blocks containing B, and $\overline{\mathcal{B}_B}$ to be the complement. Define

$$\mathcal{Y} = (\{0,1,2,\ldots,n-2\} \times \{0,1,2,\ldots,11\}) \cup \{A0, A1\}.$$

Note that $\mathcal{Y}$ has cardinality $(n-1)(12)+2 = 12n-10$. We construct blocks on $\mathcal{Y}$ as follows:

1. $i \oplus \mathcal{B}(14)$ *for* $i \in \{0, 1, \ldots, n-2\}$ *and where* i *is omitted if the point is of the form* Ah.

2. *a)* $\{Ah, ub_1, vb_2, w(b_3+3h)\}$ *where* $\{u, v, w, B\} \in \mathcal{B}_B$ *and with* b_1, b_2, b_3 *satisfying* $b_1 + b_2 + b_3 \equiv 0 \pmod{12}$.

   b) $\{u\alpha_1, v\alpha_2, w\alpha_3, w\alpha_4\}$ *where* $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ *are the second indices (in order) of blocks of types (3) - (7) in the SQS(38).*

   c) $\{i\beta_1, i\beta_2, i'\beta_3, i'\beta_4\}$ *where* $\{i, i', B\}$ *defines a unique block in* $\mathcal{B}_B$ *and* $\beta_1, \beta_2, \beta_3, \beta_4$ *are the second indices (in order) of blocks of type (8) - (14) in the SQS(38).*

3. $\{xa_1, ya_2, za_3, ta_4\}$, *where* $\{x, y, z, t\} \in \overline{\mathcal{B}_B}$ *and* $\{a_1, a_2, a_3, a_4\}$ *satisfy* $a_1 + a_2 + a_3 + a_4 \equiv 0 \pmod{12}$.

Theorem 5.8.35. *For* $n \equiv 2, 4 \pmod{6}$ *with* $n > 4$, *there exists an SQS*(n).

Proof. Proceed by induction on n. Since it is not possible to create a 1-ocycle for an SQS(2) or SQS(4), we begin our base cases with $n = 8, 10$. If $n = 8$, we have the following SQS, arranged in 1-ocycle form:

$$2148, 8523, 3684, 4578, 8156, 6287, 7813, 3576, 6471, 1572, 2163, 3274, 4135, 5462.$$

When $n = 10$, we have the following SQS(10), arranged in 1-ocycle form:

2145	6907	1372	7268	2694	7491
5263	7108	2483	8379	4681	1693
3674	8192	3594	9480	1583	3608
4785	2903	4506	0591	3705	8520
5896	3401	6157	1602	5297	0472

Before Construction 5.8.34, we illustrated an SQS(14) and an SQS(38).

Let $n \geq 16$ with $n \equiv 4, 8 \pmod{12}$. Then for some $v \equiv 2, 4 \pmod{6}$, we have $n = 2v$. Since $n \geq 16$ implies $v \geq 8$, we use Construction 5.8.29.

Let $n \geq 22$ with $n \equiv 4, 10 \pmod{18}$. Then for some $v \equiv 2, 4 \pmod{6}$, we have $n = 3v - 2$. Since $n \geq 22$ implies $v \geq 8$, we use Construction 5.8.30.

Let $n \geq 26$ with $n \equiv 2, 10 \pmod{24}$. Then for some $v \equiv 2, 4 \pmod{6}$, we have$n = 4v - 6$. Since $n \geq 26$ implies $v \geq 8$, we use Construction 5.8.33.

Let $n \geq 26$ with $n \equiv 26 \pmod{36}$. Then for some $v \equiv 10 \pmod{12}$, we have $n = 3v - 4$. Since $n \geq 26$ implies $v \geq 10$, we use Construction 5.8.32.

Let $n \geq 34$ with $n \equiv 34 \pmod{36}$. Then for some $v \equiv 2 \pmod{12}$, we have $n = 3v - 8$. Since $n \geq 34$ implies $v \geq 14$, we use Construction 5.8.31.

Let $n \geq 86$ with $n \equiv 14, 38 \pmod{72}$. Then for some $v \equiv 2, 4 \pmod{6}$, we have $n = 12v - 10$. Since $n \geq 86$ implies $v \geq 8$, we use Construction 5.8.34.

□

5.8.2.2 Ocycles and Hanani's Constructions

Every result in this section will be proven in the same manner. First, each type of block will generate individual ocycles. Then we will illustrate how to connect all cycles together to form one 1-ocycle that covers all blocks.

Result 5.8.36. *Let $(X, \mathcal{B})$ be an SQS(n). Then there exists an SQS($2n$) that admits a 1-ocycle.*

Proof. We use Construction 5.8.29. To create cycles on the quadruples we do the following.

1. Fix $a_1, a_2 \in \{0, 1\}$ and $\{x, y, z, t\} \in \mathcal{B}$. We have two choices for a_3, and this choice completely determines a_4. Thus by fixing a_1, a_2 and

$\{x, y, z, t\}$, we have identified two blocks of type (1). We can construct the short 2-block 1-ocycle as shown below:

$$a_1 x \ldots a_2 y \ldots a_1 x.$$

2. For each $d \in \{1, 2, \ldots, \lfloor \frac{n}{2} \rfloor\}$, define the following cycle:

$$\begin{array}{cccc} 00 & 10 & 1d & 0d \\ 0d & 1d & 1(2d) & 0(2d) \\ 0(2d) & 1(2d) & 1(3d) & 0(3d) \\ \vdots & & & \end{array}$$

We continue this cycle until we arrive back at 00. At this point, if $\gcd(d, n) = 1$, we will have covered all blocks of difference d. However, if $\gcd(d, n) > 1$, then we may need multiple cycles to cover all blocks. In this case we just start anew with the first block missed.

To connect these cycles, we look to the blocks of type (2). Note that when $d = 1$ we will obtain one long cycle, call it $\mathcal{C}$, that has all points $0j$ with $j \in \{0, 1, 2, \ldots n-1\}$ as overlap points. Then we can join all cycles from (2) to $\mathcal{C}$. To attach the cycles of type (1), we utilize cycles from (1) with $a_1 = 0$. We can connect these all to $\mathcal{C}$ using the overlap points $a_1 x = 0x$. For the remaining cycles with $a_1 = 1$, it is again possible to connect to $\mathcal{C}$ as follows. If $a_2 = 0$ we can attach at the points $a_2 y = 0y$ on $\mathcal{C}$, and if $a_2 = 1$ we can attach at the points $a_2 y = 1y$ that exist on the cycles from (1) with $a_1 = 0, a_2 = 1$. □

Result 5.8.37. *Let $(X, \mathcal{B})$ be an SQS(n). Then there exists an SQS($3n-2$) that admits a 1-ocycle.*

Proof. We use Construction 5.8.30. To create cycles on the quadruples we do the following.

1. Fix $a_1, a_2 \in \{0,1,2\}$ and a block $\{x,y,z,t\} \in \overline{\mathcal{B}_A}$. Note that these choices determine a set of three blocks in the SQS($3n-2$) since we may choose any $a_3 \in \{0,1,2\}$, but then our choices have completely determined a_4. For each $a_1 \in \{0,1,2\}$ and $\{x,y,z,t\} \in \overline{\mathcal{B}_A}$, we create the 1-ocycle

$$\begin{array}{cccc|c} 0z & a_1x & a_4t & 0y & \\ 0y & 1z & a_4t & a_1x & a_2 = 0 \\ a_1x & 0y & a_4t & 2z & \\ \hline 2z & 1y & a_4t & a_1x & \\ a_1x & a_4t & 1z & 1y & a_2 = 1 \\ 1y & a_4t & 0z & a_1x & \\ \hline a_1x & 2z & a_4t & 2y & \\ 2y & 1z & a_4t & a_1x & a_2 = 2 \\ a_1x & a_4t & 2y & 0z & \end{array}$$

2. Fix $\{A,u,v,w\} \in \mathcal{B}_A$ and choose $b_1 \in \{0,1,2\}$. This identifies three blocks:

$$\{A, 0v, b_3^0 w, b_1 u\}, \ \{A, 1v, b_3^1 w, b_1 u\}, \text{ and } \{A, 2v, b_3^2 w, b_1 u\},$$

where $\{b_3^0, b_3^1, b_3^2\} = \{0,1,2\}$. Note that the order of $\{b_3^0, b_3^1, b_3^2\}$ depends entirely on our choice of b_1. We can string together the groups of three

blocks for each choice of b_1 to create the following 1-ocycle:

$$\begin{array}{cccc|c} 0v & A & 0w & 0u & \\ 0u & 1v & 2w & A & b_1 = 0 \\ A & 0u & 1w & 2v & \\ \hline 2v & 1u & 0w & A & \\ A & 1v & 1w & 1u & b_1 = 1 \\ 1u & 0v & 2w & A & \\ \hline A & 2v & 2w & 2u & \\ 2u & 1v & 0w & A & b_1 = 2 \\ A & 2u & 1w & 0v & \end{array}$$

3. Fix $\{A, u, v, w\} \in \mathcal{B}_A$ and $i \in \{0, 1, 2\}$. The three blocks identified create a 1-ocycle:

$$\begin{array}{cccc} iu & (i+1)w & (i+2)w & iv \\ iv & (i+1)u & (i+2)u & iw \\ iw & (i+1)v & (i+2)v & iu \end{array}$$

4. For each choice of j and j', we create the short cycle:

$$\begin{array}{cccc} 0j & 0j' & 1j' & 1j \\ 1j & 1j' & 2j' & 2j \\ 2j & 2j' & 0j' & 0j \end{array}$$

5. We can create the short strings:

$$\begin{array}{cccc} 0j & 1j & 2j & A \\ A & 2(j+1) & 1(j+1) & 0(j+1) \end{array}$$

To create an ocycle, we will utilize a few of the cycles from (4). For each even $j \in \{0, 1, 2, \ldots, n-1\}$, set $j' = j + 1$. Then we modify the

corresponding cycle from (4) to include one of the short strings as shown:

$$\begin{array}{cccc} & & & \\ 0j & 0j' & 1j' & 1j \\ 1j & 1j' & 2j' & 2j \\ 2j & 2j' & 0j' & 0j \\ & & & \end{array} \rightarrow \begin{array}{cccc} 0j & 0j' & 1j' & 1j \\ 1j & 1j' & 2j' & 2j \\ 2j & 2j' & 0j & 0j' \\ \hline 0j' & 1j' & 2j' & A \\ A & 2j & 1j & 0j \end{array}$$

Note that this also ensures that A appears as an overlap point, so, together with $\mathcal{C}$ from (4), we are ensured that *every* point appears as an overlap point.

To connect these cycles and make one ocycle, we consider the blocks of type (3). Fix $i \in \{0,1,2\}$ and $u \in \{0,1,2,\ldots,n-2\}$ and let v vary through $\{0,1,2,\ldots,n-2\}\backslash\{u\}$. Then each of these cycles from (3) containing $\{A,u,v\}$ has the point iu as an overlap point, so we can connect all of the cycles to make a long cycle, call it $\mathcal{C}_{i,u}$. To connect $\mathcal{C}_{0,u}, \mathcal{C}_{1,u}$, and $\mathcal{C}_{2,u}$ we use a cycle from (4) with $j = u$. Now we have created a cycle containing every point $ij \in \{0,1,2\} \times \{0,1,2,\ldots,n-2\}$ as an overlap point, so every other cycle can connect to this one. □

Result 5.8.38. *Let $(X,\mathcal{B})$ be an SQS(n) with $n \equiv 2 \pmod{12}$ that admits a 1-ocycle. Then there exists an SQS($3n-8$) that admits a 1-ocycle.*

Proof. We use Construction 5.8.31. We create cycles as follows on each set of blocks.

1. We will add this block to a cycle over blocks of type (2).

2. If we have an overlap cycle on $(X, \mathcal{B})$, call it $\mathcal{C}$, then for each $i \in \{0, 1, 2\}$ we get a cycle $\mathcal{C}_i$ by preceding each letter $x \in \{0, 1, 2, \ldots, n-5\}$ by i and leaving all terms Ah with $h \in \{0, 1, 2, 3\}$ unchanged. However, since each cycle uses the block $\{A0, A1, A2, A3\}$ so we can only use one of these cycles, say $\mathcal{C}_0$. For $\mathcal{C}_1$ and $\mathcal{C}_2$, we get two strings by removing the block $\{A0, A1, A2, A3\}$. Note that $\mathcal{C}_1$ and $\mathcal{C}_2$ can then be joined together at the endpoints. For example, if the block appears as $A0, A1, A2, A3$ (in order) in $\mathcal{C}$, then we create the cycle

$$A3 \quad \mathcal{C}_1 \quad A0 \quad \overline{\mathcal{C}_2} \quad A3$$

where $\overline{\mathcal{C}_2}$ is the string $\mathcal{C}_2$ listed in reverse order.

3. If we fix a_1, a_2, then we have restricted our attention to $n-4$ distinct blocks. Since $n-4$ is even, so we make the following cycles:

$$\begin{array}{ccc} Aa_1 & \cdots & 0a_2 \\ 0a_2 & \cdots & Aa_1 \\ Aa_1 & \cdots & 0a_2 \\ \vdots & & \\ 0a_2 & \cdots & Aa_1 \end{array}$$

4. We will arrange each block so that the overlap points are $(i+2)b_3$ and $(i+1)b_2$. Any choice of $b_2, b_3 \in \{0, 1, 2, \ldots n-5\}$ completely determines b_1, and by choosing b_2, b_3, i, d, we have identified a unique block. Fix $b_2 \neq b_3$ and d, and then we will create one cycle as follows. The cycle

when $b_2 \neq b_3$ is:

$$\begin{array}{ccc} 0b_3 & \cdots & 1b_2 \\ 1b_2 & \cdots & 2b_3 \\ 2b_3 & \cdots & 0b_2 \\ 0b_2 & \cdots & 1b_3 \\ 1b_3 & \cdots & 2b_2 \\ 2b_2 & \cdots & 0b_3 \end{array}$$

When $b_2 = b_3$, we have the shorter cycle:

$$\begin{array}{ccc} 0b_2 & \cdots & 1b_2 \\ 1b_2 & \cdots & 2b_2 \\ 2b_2 & \cdots & 0b_2 \end{array}$$

5. Fix α, and arbitrarily order the set

$$P_\alpha(6k+5) = \{[r(1), s(1)], [r(2), s(2)], \ldots, [r(t), s(t)]\}.$$

For each $d = 0, 1, 2, \ldots, t-1$, we create a cycle C_d as follows:

$$\begin{array}{cccc} 0r(1) & 0s(1) & 1s(1+d) & 1r(1+d) \\ 1r(1+d) & 1s(1+d) & 2s(1+2d) & 2r(1+2d) \\ 2r(1+2d) & 2s(1+2d) & 0s(1+3d) & 0r(1+3d) \\ \vdots & & & \\ 2r(1-d) & 2s(1-d) & 0s(1) & 0r(1) \end{array}$$

This cycle will always eventually reach the final block. However, it may or may not also cover (among others) the intermediary blocks

$\{1r(1-d), 1s(1-d), 2s(1), 2r(1)\}$ and $\{0r(1-d), 0s(1-d), 1s(1), 1r(1)\}$.

If C_d misses these intermediary blocks, then we create a new cycle similar to C_d, but we replace $0, 1, 2$ with either $1, 2, 0$ or $2, 0, 1$, respectively.

To connect all of these cycles, we first focus on the cycles from (4). We create one long cycle by fixing $b_2 = 0$ and creating a cycle C_{b_3} for each b_3 from the set $\{0, 1, 2, \ldots, n-5\}$. We connect all of these cycles at their points 00 to create one long cycle C that has every element from the set

$$\{0, 1, 2\} \times \{0, 1, 2, \ldots, n-5\}$$

as an overlap point Using this cycle C, we can connect every other cycle to C to make one long cycle containing all quadruples. □

Result 5.8.39. *Let $(X, \mathcal{B})$ be an SQS(n) with $n \equiv 10 \pmod{12}$ that admits a 1-ocycle. Then there exists an SQS($3n-4$) that admits a 1-ocycle.*

Proof. We use Construction 5.8.32. To form cycles, we do the following for each type.

1. If $\mathcal{C}$ is a cycle on $(X, \mathcal{B})$, then $i \oplus \mathcal{C}$ is a cycle on quadruples of type (1) for each $i \in \{0, 1, 2\}$.

2. If we fix a_1, a_2, then we have restricted our attention to $n-2$ distinct blocks. Note that $n-2$ is even, and so we make the following cycles:

$$\begin{array}{ccc} Aa_1 & \cdots & 0a_2 \\ 0a_2 & \cdots & Aa_1 \\ Aa_1 & \cdots & 0a_2 \\ \vdots & & \\ 0a_2 & \cdots & Aa_1 \end{array}$$

3. We will arrange each block so that $(i+2)b_3$ and $(i+1)b_2$ are the overlap points. Any choice of $b_2, b_3 \in \{0, 1, 2, \ldots n-3\}$ completely determines b_1, and by choosing b_2, b_3, i, d we have identified a unique block. Fix $b_2 \neq b_3$ and d, and we can create one cycle as follows. The cycle when $b_2 \neq b_3$ is:

$$\begin{aligned} 0b_3 &\cdots 1b_2 \\ 1b_2 &\cdots 2b_3 \\ 2b_3 &\cdots 0b_2 \\ 0b_2 &\cdots 1b_3 \\ 1b_3 &\cdots 2b_2 \\ 2b_2 &\cdots 0b_3 \end{aligned}$$

When $b_2 = b_3$, we have the shorter cycle:

$$\begin{aligned} 0b_2 &\cdots 1b_2 \\ 1b_2 &\cdots 2b_2 \\ 2b_2 &\cdots 0b_2 \end{aligned}$$

4. Fix α. Arbitrarily order the set

$$P_\alpha(6k+4) = \{[r(1), s(1)], [r(2), s(2)], \ldots, [r(t), s(t)]\}.$$

For each $d = 0, 1, 2, \ldots, t-1$, we create a cycle C_d as follows:

$$\begin{array}{llll} 0r(1) & 0s(1) & 1s(1+d) & 1r(1+d) \\ 1r(1+d) & 1s(1+d) & 2s(1+2d) & 2r(1+2d) \\ 2r(1+2d) & 2s(1+2d) & 0s(1+3d) & 0r(1+3d) \\ \vdots & & & \\ 2r(1-d) & 2s(1-d) & 0s(1) & 0r(1) \end{array}$$

This cycle will always eventually reach the final block. However, it may or may not also cover (among others) the intermediary blocks

$$\{1r(1-d), 1s(1-d), 2s(1), 2r(1)\} \text{ and } \{0r(1-d), 0s(1-d), 1s(1), 1r(1)\}.$$

If C_d misses these intermediary blocks, then we create a new cycle similar to C_d, but we replace $0, 1, 2$ with either $1, 2, 0$ or $2, 0, 1$, respectively.

To connect all of these cycles, we first focus on (3). We create one long cycle by fixing $b_2 = 0$ and creating a cycle C_{b_3} for each $b_3 \in \{0, 1, 2, \ldots, n-3\}$. We connect all of these cycles at their points 00 to create one long cycle C that has every overlap from the set

$$\{0, 1, 2\} \times \{0, 1, 2, \ldots, n-3\}.$$

Using this cycle C, we can connect every other cycle to C to make one longer cycle containing all quadruples. □

Result 5.8.40. *Let $(X, \mathcal{B})$ be an SQS(n) that admits a 1-ocycle. Then there exists an SQS($4n-6$) that admits a 1-ocycle.*

Proof. We use Construction 5.8.33. To create cycles on these blocks, we do the following.

1. If $\mathcal{C}$ is a 1-overlap cycle on $\mathcal{B}$, then for each choice of $h, i \in \{0, 1\}$ we create the cycle $h \oplus i \oplus \mathcal{C}$.

2. We will combine these with the triples of type (3). Note in particular that each block is completely determined by our choice of $2c_2^{(2)} - \epsilon^{(2)}$ and $2c_3^{(2)} + \ell^{(2)}$ from $\{0, 1, 2, \ldots, n-3\}$, where superscript (2) denotes a

variable corresponding to a block of type (2), and similarly a superscript (3) for blocks of type (3).

3. Note that each block is completely determined by choosing $2c_2^{(3)} - 1 - \epsilon^{(3)}$ and $2c_3^{(3)} + 1 - \ell^{(3)}$ from $\{0, 1, 2, \ldots, n-3\}$. Fix $x \in \{0, 1, 2, \ldots, n-3\}$, and define the cycles as follows, alternating between blocks of type (2) (where $x = 2c_3^{(2)} + \ell^{(2)}$) and (3) (where $x = 2c_3^{(3)} + 1 - \ell^{(3)}$).

   We will connect pairs of blocks (one of type (2) and one of type (3)) with $2c_2^{(2)} - \epsilon^{(2)} = 2c_2^{(3)} - 1 - \epsilon^{(3)}$. Note that this implies that $\epsilon^{(2)} \neq \epsilon^{(3)}$. These blocks are connected to make short strings as shown by matching a block of type (2) and (3) in which $2c_2^{(2)} - \epsilon^{(2)} = 2c_2^{(3)} - 1 - \epsilon^{(3)}$:

$$\begin{array}{llll} 1\epsilon^{(2)}x, & A\ell^{(2)}, & 00(2c_1^{(2)}), & 01(2c_2^{(2)} - \epsilon^{(2)}) \\ 01(2c_2^{(3)} - 1 - \epsilon^{(3)}), & 00(2c_1^{(3)} + 1), & A\ell^{(3)}, & 1\epsilon^{(3)}x \end{array}$$

   To connect these two-block strings, we define

$$y = 2c_2^{(2)} - \epsilon^{(2)} = 2c_2^{(3)} - 1 - \epsilon^{(3)},$$

   and we let y range from 0 up to $n-3$ to create a cycle covering all of these strings.

$$
\begin{array}{lll}
10x & \cdots & 010 \\
010 & \cdots & 11x \\
11x & \cdots & 011 \\
011 & \cdots & 10x \\
10x & \cdots & 012 \\
012 & \cdots & 11x \\
\vdots & & \\
01(n-4) & \cdots & 11x \\
11x & \cdots & 01(n-3) \\
01(n-3) & \cdots & 10x
\end{array}
$$

For each choice of x we will have one cycle, and each cycle has length $2(n-2)$, covering a total of $2(n-2)^2$ blocks.

4. We will combine these with the triples of type (5). Note in particular that each block is completely determined by our choice of $2c_2 - \epsilon$ and $2c_3 + 1 - \ell$ from $\{0, 1, 2, \ldots, n-3\}$.

5. We proceed in a manner similar to (3). Note that each block is completely determined by choosing $2c_2 - 1 - \epsilon$ and $2c_3 + \ell$ from the set $\{0, 1, 2, \ldots, n-3\}$. Fix $x \in \{0, 1, 2, \ldots, n-3\}$, and define a cycle, which alternates between blocks of type (4) (where $x = 2c_3 + 1 - \ell$) and blocks

of type (5) (where $x = 2c_3 + \ell$), as shown below.

$$
\begin{array}{lcl}
00x & \cdots & 110 \\
110 & \cdots & 01x \\
01x & \cdots & 111 \\
111 & \cdots & 00x \\
00x & \cdots & 112 \\
112 & \cdots & 01x \\
\vdots & & \\
11(n-4) & \cdots & 01x \\
01x & \cdots & 11(n-3) \\
11(n-3) & \cdots & 00x
\end{array}
$$

6. We will combine these with the triples of type (8).

7. We will combine these with the triples of type (9).

8. Note that the first two terms in both (6) and (8) are the same. For each choice of $2c_1+\epsilon, 2c_2-\epsilon \in \{0,1,2,\ldots,n-3\}$, we create the short 2-block cycle:

$$h0(2c_1+\epsilon)\cdots h1(2c_2-\epsilon)$$

$$h1(2c_2-\epsilon)\cdots h0(2c_1+\epsilon)$$

where the first block is of type (6) and the second block is of type (8).

9. Note that the first two terms in both (7) and (9) are the same. For each choice of $2c_1 - 1 + \epsilon, 2c_2 - \epsilon \in \{0, 1, 2, \ldots, n-3\}$ we create the short 2-block cycle:

$$h0(2c_1 - 1 + \epsilon) \cdots h1(2c_2 - \epsilon)$$

$$h1(2c_2 - \epsilon) \cdots h0(2c_1 - 1 + \epsilon)$$

where the first block is of type (7) and the second block is of type (9).

10. We use a similar method as in previous constructions. Fix α and write $P_\alpha(k) = \{[r(1), s(1)], [r(2), s(2)], \ldots, [r(t), s(t)]\}$. For each difference d from 2 to $\lfloor \frac{t}{2} \rfloor$ we define several cycles. For each difference, the pairs will be partitioned into sets of a certain size, say s_d. The number of cycles defined will depend on whether s_d is even or odd.

    For example, for difference 2, if s_1 is even then we have the four cycles (two for each choice of $h \in \{0, 1\}$) for each partition class:

$$\begin{array}{lcl} h0r(1) & \cdots & h1r(3) \\ h1r(3) & \cdots & h0r(5) \\ h0r(5) & \cdots & h1r(7) \\ \vdots & & \\ h1r(t-1) & \cdots & h0r(1) \end{array}$$

    and

$$\begin{array}{lcl} h1r(1) & \cdots & h0r(3) \\ h0r(3) & \cdots & h1r(5) \\ h1r(5) & \cdots & h0r(7) \\ \vdots & & \\ h0r(t-1) & \cdots & h1r(1) \end{array}$$

$h0r(1)$	$h0s(1)$	$h1s(2)$	$h1r(2)$
$h1r(2)$	$h1s(2)$	$h0r(2)$	$h0s(2)$
$h0s(2)$	$h1s(1)$	$h1r(1)$	$h0r(2)$
$h0r(2)$	$h0s(2)$	$h1s(3)$	$h1r(3)$
$h1r(3)$	$h1s(3)$	$h0r(3)$	$h0s(3)$
$h0s(3)$	$h1s(2)$	$h1r(2)$	$h0r(3)$
$\vdots$			
$h0r(t)$	$h0s(t)$	$h1s(1)$	$h1r(1)$
$h1r(1)$	$h1s(1)$	$h0r(1)$	$h0s(1)$
$h0s(1)$	$h1s(t)$	$h1r(t)$	$h0r(1)$

Figure 5.4: Cycle for the proof of Result 5.8.39

If s_1 is odd we have the following two cycles (one for each choice of h):

$$
\begin{array}{lcl}
h0r(1) & \cdots & h1r(3) \\
h1r(3) & \cdots & h0r(5) \\
h0r(5) & \cdots & h1r(7) \\
\vdots & & \\
h0r(t-1) & \cdots & h1r(1) \\
h1r(1) & \cdots & h0r(3) \\
h0r(3) & \cdots & h1r(5) \\
h1r(5) & \cdots & h0r(7) \\
\vdots & & \\
h0r(t-1) & \cdots & h1r(1)
\end{array}
$$

We construct cycles in a similar manner for each partition set of each difference.

All that remains are the blocks of differences 0 and 1. We will construct two cycles, one for each choice of h as shown in Figure 5.4:

Now we must connect all of these cycles. Note that the cycles described in (3) and (5) contain as overlap points all points of the form $11y$ for every

$y \in \{0, 1, 2, \ldots, n-3\}$, regardless of our choice of x. Thus we can connect all of these cycles together to make one cycle. This one long cycle contains as overlap points all points of the form hiy for $h \in \{0, 1\}$, $i \in \{0, 1\}$, and $y \in \{0, 1, 2, \ldots, n-3\}$. Thus we can connect everything to this cycle, since no points $A\ell$ are used as overlap points. □

Result 5.8.41. *Let $(X, \mathcal{B})$ be an SQS(n) that admits a 1-ocycle. Then there exists an SQS($12n - 10$) that admits a 1-ocycle.*

Proof. We begin by proving that the construction for an SQS(38) admits a 1-ocycle. To construct a 1-overlap cycle, we look at each type of block separately.

1. Since we have a 1-overlap cycle for the SQS(14), clearly we can construct one cycle for each i on these types of blocks.

2. Fix b_1 and b_2 and construct a short 2-block cycle by changing h from 0 to 1.

$$\begin{array}{cccc} 0b_1 & A0 & 2b_3 & 1b_2 \\ 1b_2 & 2(b_3+3) & A1 & 0b_1 \end{array}$$

3. Fix $\{b_2, b_3\} = \{x, y\}$ with $x \neq y$ and create the cycle:

$$\begin{array}{ccc} 0x & \cdots & 1y \\ 1y & \cdots & 2x \\ 2x & \cdots & 0y \\ 0y & \cdots & 1x \\ 1x & \cdots & 2y \\ 2y & \cdots & 0x \end{array}$$

When we have $b_2 = b_3 = x$ we have the shorter cycle:

$$\begin{aligned} 0x &\cdots 1x \\ 1x &\cdots 2x \\ 2x &\cdots 0x \end{aligned}$$

4. We will combine blocks of type (4) with blocks of type (6).

5. We will combine blocks of type (5) with blocks of type (7).

6. Note that the first term and the third term in the blocks of type (4) are the same as those for the blocks of type (6). Using this, we create short two-block cycles by fixing ϵ and connecting the blocks as shown:

$$ij \ldots (i+2)(6\epsilon - 2j + 1) \ldots ij.$$

7. Note that the first term and the last term in the blocks of type (5) are the same as those for the blocks of type (7). Using this, we create short two-block cycles by fixing ϵ and connecting the blocks as shown:

$$ij \ldots (i+2)(6\epsilon - 2j - 2) \ldots ij.$$

8. For each choice of $\epsilon \in \{0,1\}$, we have two blocks of type (8) containing both ij and $i(j+6)$. We connect these as shown to create short two-block cycles:

$$ij \ldots i(j+6) \ldots ij.$$

9. We will combine these with the blocks of type (10).

10. Fix $g \in \{0,1,2,3,4,5\}$, $\epsilon \in \{0,1\}$ and $i \in \{0,1,2\}$. Then we have two choices for $i' \in \{0,1,2\} \setminus \{i\}$, so we have narrowed our consideration to 2 blocks of type (9) and 2 blocks of type (10). In any order, we list them in a 1-overlap cycle as:

$$i(2g+3\epsilon) \ldots i(2g+6+3\epsilon) \ldots i(2g+3\epsilon) \ldots i(2g+6+3\epsilon) \ldots i(2g+3\epsilon)$$

11. Fix i, j. This restricts us to four blocks - one for each $e \in \{0,1,2,3\}$. Form a 1-overlap cycle as follows:

$$ij \ldots i(j+1) \ldots ij \ldots i(j+1) \ldots ij.$$

12. Same as (11).

13. Same as (11).

14. Fix $i, i' \in \{0, 1, 2\}$ (distinct) and $\alpha \in \{4, 5\}$. Order the pairs in $P_\alpha(6)$ arbitrarily as $\{[r(1), s(1)], [r(2), s(2)], \ldots, [r(t), s(t)]\}$. For each difference $d = 1, 2, \ldots, \lfloor t/2 \rfloor$, we construct the string S_1 as follows:

$$\begin{array}{llll} ir(1) & is(1) & i's(1+d) & i'r(1+d) \\ i'r(1+d) & i's(1+d) & is(1+2d) & ir(1+2d) \\ ir(1+2d) & is(1+2d) & i's(1+3d) & i'r(1+3d) \\ \vdots & & & \end{array}$$

Continue this string until we arrive at a block that ends with either $ir(1)$ or $i'r(1)$. We construct S_2 from S_1 by swapping i and i' everywhere. If S_1 ends in $ir(1)$, then S_2 ends in $i'r(1)$, and both are cycles. If S_1 ends in $i'r(1)$, then S_2 ends in $ir(1)$ and we can connect S_1 and S_2 to make one cycle. If d divides t, then we repeat the procedure, replacing $[r(1), s(1)]$ with a pair from $P_\alpha(6)$ that was not used in creating S_1 and S_2.

To connect all of these cycles, we look to the blocks of type (2). Fix b_1, and connect all of the short cycles corresponding to this b_1 together at the point $0b_1$. Note that this cycle has as overlap points $0b_1$ and every point $1j$ for $j \in \{0, 1, 2, \ldots, 11\}$. Do this for each choice of b_1, and then each of these cycles contains the point 10 as an overlap, so they can all be connected. From this, we have constructed one cycle that contains every point ij with $i, j \in \{0, 1, 2, \ldots, 11\}$ as an overlap point. All other cycles can be connected to this.

The general SQS(n) $\rightarrow$ SQS(12n $-$ 10) construction: To construct an overlap cycle, we look at each type of block separately.

1. Since we can construct a 1-ocycle on $\mathcal{B}(14)$, we can construct a 1-ocycle on $i \oplus \mathcal{B}(14)$ for each choice of i.

2. a) These blocks are completely determined by our choice of any $b_1, b_2 \in \{0, 1, \ldots, 11\}$ and $h \in \{0, 1\}$. If we fix b_1, b_2, then we can connect the two blocks identified as follows:

$$ub_1 \cdots vb_2 \cdots ub_1.$$

   We do this for each choice of b_1 and b_2 to make many short 2-block cycles.

   b) These blocks have structure similar to their corresponding blocks from the SQS(38). Therefore, we make cycles in exactly the same way for blocks of type (4) - (7) in the SQS(38). For blocks of type (3) in the SQS(38), we can use the same structure, since only the last 2 points are used as overlaps. In this SQS($12n - 10$), this corresponds to the points $w\alpha_3, w\alpha_4$, and so we can create cycles in the same way.

   c) We make cycles in the same way as in the SQS(38) for blocks of type (8) - (14).

3. Fix $\{x, y, z, t\} \in \overline{\mathcal{B}_B}$. Our new block is completely determined by choosing $a_1, a_2, a_3 \in \{0, 1, \ldots, 11\}$. If we fix a_1, a_2 and let a_3 run through

the set $\{0, 1, \ldots, 11\}$, then we have identified 12 blocks, all of which contain the points xa_1 and ya_2. Using these as our overlap points, we have a 1-ocycle covering these twelve blocks, which in compressed form is $xa_1, ya_2, xa_1, ya_2, \ldots, xa_1, ya_2$.

To connect all of these cycles, we look to the blocks of type (2). Fix u, b_1, and let everything else vary. These many short cycles can all be connected at the point ub_1, and the cycle created contains every point as an overlap point, except $A0, A1$. We can connect all other cycles to this one. □

All of these constructions together with base cases proves the following result.

Result 5.8.42. *For all $n \equiv 2, 4 \pmod 6$ with $n \geq 8$, there exists an SQS(n) that admits a 1-ocycle.*

Proof. These constructions, together with Theorem 5.8.35 and the corresponding base cases, produce a 1-ocycle for each order. □

We end with the following open problem.

Open Problem 5.8.43. *For which of Hanani's six SQS constructions can we find 2-overlap cycles?*

Chapter 6

RESULTS

6.1 Gray Codes

Of the three list types discussed (Gray codes, universal cycles, and overlap cycles), Gray codes are the oldest, and hence most developed. One of our main contributions to this area is a simplified Gray code for fixed weight m-ary words (Theorem 3.3.2).

3.2.9: For all $n \geq 0$ and $k \geq 1$, there exists a Gray code listing for the set

$$\hat{\mathcal{B}}_k(n) = \{b \in \mathcal{B}(n) \mid \nexists i \in [n-1] \text{ with } b_i = b_{i+1} = \cdots = b_{i+k-1} = 0\}$$

in which successive elements differ in exactly one position.

3.3.2: There exists a Gray code listing for $\mathcal{B}_k^m(n)$ in which successive words differ in at most two positions.

3.4.10: There exists a Gray code for permutations using the disjoint cycle representation in which consecutive elements differ by a transposition of adjacent elements.

3.6.3: In a Gray code for weak orders using the operations $w_i \leftrightarrow w_{i+1}$ and $w_i \leftarrow w_{i+1}$, all words from $\mathcal{W}(n,k)$ must appear before all words from $\mathcal{W}(n,k-1)$, where $0 \leq k \leq n-1$ and $1 \leq i \leq n-1$.

3.6.6: There is a Gray code for $\mathcal{B}^*(n)$ using the operations $w_i \leftrightarrow w_{i+1}$, $w_i \leftrightarrow w_{i+2}$, and $w_i \leftarrow w_{i+1}$.

3.7.3: There is a Gray code listing for the set of Fibonacci sequences in which a partition differs from its predecessor in two parts: one part increased by one and one part decreased by one.

3.8.4: For every $n \in \mathbb{Z}^+$, there exists a Gray code listing for the blocks of any TD(k, n) using the basis representation so that each word differs from its successor in exactly one index position.

6.2 Universal Cycles

There has been much interest in the applications of universal cycles and de Bruijn sequences in the past few years. Our major achievement in this area was to determine the existence of ucycles for weak orders (Theorem 4.6.2) and several restricted subsets (Theorems 4.6.4, 4.6.8, and 4.6.9). This is an area that has not yet been considered but works very nicely within the ucycle structure.

4.2.2: Let $n \in \mathbb{Z}^+$, and let M be some fixed multiset of size n. Define the set A to be the set of all permutations of M. Then there exists a ucycle for A using the prefix representation.

4.2.3: There is a universal cycle on $\mathcal{B}_k(n)$ using the prefix representation.

4.2.4: The sets $\mathcal{B}_o(n)$ and $\mathcal{B}_e(n)$ admit universal cycles using the prefix representation.

4.4.4: Universal cycles exist for the set of permutations of $[n]$ using prefix representations. (Alternative proof)

4.6.2: For all $n \in \mathbb{Z}^+$, there exists a ucycle on $\mathcal{W}(n)$.

4.6.4: For all $n, k \in \mathbb{Z}^+$ with $n \geq 3$ and $k \leq \binom{n}{2}$, there exists a ucycle for $\mathcal{W}_k^-(n)$.

4.6.8: For all $n \in \mathbb{Z}^+$ and all $h \in \mathbb{N}$ with $0 \leq h < n-1$, there exists a universal cycle for $\mathcal{W}(n,h)$. (Alternative Proof)

4.6.9: For every $n, k, h \in \mathbb{Z}^+$ with $k \leq \binom{n}{2}$ and $0 \leq h < n$, there is a ucycle for $\mathcal{W}_k^-(n,h)$.

4.7.4: There is a universal cycle for the set of all ordered partitions of an n-set for all $n \in \mathbb{Z}^+$ in which an ordered partition is represented by its corresponding weak order.

4.7.5: For all $n \in \mathbb{Z}^+$, there is a universal cycle for the set of all ordered partitions of an n-set with fixed part sizes in which an ordered partition is represented using the prefix representation of the corresponding weak order.

4.7.6: For all $n \in \mathbb{Z}^+$ and all $h \in \mathbb{Z}^+ \cup \{0\}$ with $0 \leq h < n$, there exists a universal cycle for the set of all ordered partitions of an n-set into exactly h parts.

5.8.2: For all $n \geq 2$, $k \geq 3$, and any $TD(k,n)$, there is a universal cycle on the blocks of the design using a basis representation in which each block is represented by a string of length two.

6.3 Overlap Cycles

Overlap cycles are the newest listing structure discussed. Very little research has been done in this area, and we proved many interesting results. One in particular that has been posed as a research problem at conferences and in papers is the question of the existence of s-ocycles for k-permutations of $[n]$. We were able to positively answer this question (Theorem 5.4.4), as well as consider 1-ocycles over other interesting objects, such as Steiner triple and quadruple systems (Theorems 5.8.14 and 5.8.42, respectively).

5.2.3: Let $n, s \in \mathbb{Z}^+$ with $\frac{n}{2} \leq s \leq n-2$, and let M be some fixed multiset of size n. Define the set A to be the set of all permutations of M. If $\gcd(s, n) = 1$, then there is an s-ocycle for A.

5.2.4: Let $n, s \in \mathbb{Z}^+$ with $1 \leq s < \frac{n}{2}$. Let M be a multiset of size n. Define the set A to be the set of all permutations of M. Then there is an s-ocycle for A.

5.4.2: Let $n, s \in \mathbb{Z}^+$ with $n \geq 2$. If either (1) $1 \leq s < \frac{n}{2}$, or (2) $\gcd(s, n) = 1$ with $\frac{n}{2} \leq s < n-1$, then there exists an s-ocycle on the set of permutations of $[n]$.

5.4.3: Let $n, s, k \in \mathbb{Z}^+$ with $n \geq 2$ and $k < n$. If either (1) $1 \leq s < \frac{k}{2}$, or (2) $\gcd(s, k) = 1$ with $\frac{k}{2} \leq s < k-1$, then there exists an s-ocycle on the set of k-permutations of $[n]$.

5.4.4: For all $n, s, k \in \mathbb{Z}^+$ with $1 \leq s < k < n$, there is an s-ocycle for k-permutations of $[n]$.

5.6.1: For all $n \in \mathbb{Z}^+$ and for all $s \in \mathbb{Z}^+$ with $1 \leq s \leq n-1$, there is an s-ocycle for $\mathcal{W}(n)$.

5.6.3: Let $n, s, k \in \mathbb{Z}^+$ with $1 \leq s \leq n-2$ and $k \leq \binom{n}{2}$. If $\gcd(s, n) = 1$, then there is an s-ocycle for $\mathcal{W}_k(n)$.

5.6.4: Let $n, s, h, k \in \mathbb{Z}^+$ with $1 \leq s \leq n-2$, $k \leq \binom{n}{2}$, and $0 \leq h < n$. If $\gcd(s, n) = 1$, then there is an s-ocycle for $\mathcal{W}_k(n, h)$.

5.6.5: For all $n, s, h \in \mathbb{Z}^+$ with $1 \leq s \leq n-2$, $\gcd(s, n) = 1$, and $0 \leq h \leq n-1$, there is an s-ocycle for $\mathcal{W}(n, h)$.

5.7.3: For all $n, s \in \mathbb{Z}^+$ with $1 \leq s \leq n-1$, there is an s-ocycle for the set of all ordered partitions of an n-set.

5.7.4: For all n, s, h with $1 \leq s \leq n-2$, $\gcd(s, n) = 1$, and $h \leq n-1$, there exists an s-ocycle for the set of all ordered partitions of an n-set into exactly h parts.

5.7.5: Let $n, s \in \mathbb{Z}^+$ with $1 \leq s \leq n-2$. If $\gcd(n, s) = 1$, then there is an s-ocycle for the set of all ordered partitions of an n-set with fixed part sizes.

5.8.3: For all $n \geq 2$ and $k \geq 3$ and any $TD(k, n)$, there exists a 1-overlap cycle, $O(TD(k, n), 1)$.

5.8.6: Let $(X, \mathcal{A})$ and $(X, \mathcal{B})$ be $(v, 3, 1)$-BIBDs. Suppose that both $(X, \mathcal{A})$ and $(X, \mathcal{B})$ admit 1-overlap cycles called $O(\mathcal{A})$ and $O(\mathcal{B})$, respectively. Then the $(v, 3, 2)$-BIBD $(X, \mathcal{A} \cup \mathcal{B})$ using the sum construction also admits a 1-overlap cycle.

5.8.9: If $(X, \mathcal{A})$ is a (v, k, λ)-BIBD that admits a t-overlap cycle, then the $(v, k, s\lambda)$-BIBD constructed from Theorem 5.8.8 also admits a t-overlap cycle.

5.8.11: Given $(X, \mathcal{A}) = (v, b, r, k, \lambda)$-BIBD, we can construct the block complementation of it to create a new design. This design is $(X, \overline{\mathcal{A}}) = (v, b, b - r, v - k, b - 2r + \lambda)$-BIBD. If $v - 2k \geq 2$ and $(X, \mathcal{A})$ has a 1-overlap cycle, then $(X, \overline{\mathcal{A}})$ also has a 1-overlap cycle.

5.8.14: For every $v \equiv 1, 3 \pmod 6$ with $v \geq 15$, there exists an AF STS(v) with a 1-ocycle.

5.8.17: If there exists an AF STS(v) with a 1-ocycle, then there exists an AF STS($2v + 1$) with a 1-ocycle when $v \geq 15$.

5.8.18: If there exists an AF STS(v) with a 1-ocycle, then there exists an AF STS($2v + 7$) with a 1-ocycle, when $v \geq 15$.

5.8.21: If there exists an STS(u) with a 1-overlap cycle and an STS(v) with a 1-overlap cycle, then there exists an STS(uv) with a 1-ocycle.

5.8.23: For $n \equiv 3 \pmod 6$ with $n > 3$, there exists an STS(n) that admits a 1-ocycle.

5.8.25: For $n \equiv 1 \pmod 6$ with $n > 1$, there exists an STS(n) that admits a 1-overlap cycle.

5.8.36: Let $(X, \mathcal{B})$ be an SQS of order n. Then there exists an SQS of order $2n$ that admits a 1-ocycle.

5.8.37: Let $(X, \mathcal{B})$ be an SQS of order n. Then there exists an SQS of order $3n - 2$ that admits a 1-ocycle.

5.8.38: Let $(X, \mathcal{B})$ be an SQS(n) with $n \equiv 2 \pmod{12}$ that admits a 1-ocycle. Then there exists an SQS($3n - 8$) that admits a 1-ocycle.

5.8.39: Let $(X, \mathcal{B})$ be an SQS(n) with $n \equiv 10 \pmod{12}$ that admits a 1-ocycle. Then there exists an SQS($3n - 4$) that admits a 1-ocycle.

5.8.40: Let $(X, \mathcal{B})$ be an SQS(n) that admits a 1-ocycle. Then there exists an SQS($4n - 6$) that admits a 1-ocycle.

5.8.41: Let $(X, \mathcal{B})$ be an SQS(n) that admits a 1-ocycle. Then there exists an SQS($12n - 10$) that admits a 1-ocycle.

5.8.42: For every $v \equiv 2, 4 \pmod{6}$ with $v > 4$, there exists an SQS(v) that admits a 1-ocycle.

Chapter 7

OPEN PROBLEMS

7.1 Gray Codes

3.2.5: Is there a Gray code for the set $\mathcal{B}_o(n)$, the set of all odd-weight binary strings of length n? What about for $\mathcal{B}_e(n)$, the set of all even-weight binary strings of length n?

3.2.8: Let $r(n)$ be the maximum value r such that a Gray code for $\mathcal{B}(n)$ exists with run length at least r. What are the values of $r(n)$?

3.4.6: What is a good minimal change property for the set of permutations with fixed rank?

3.4.7: Does there exist a Gray code listing for the set of permutations with fixed rank?

3.4.8: Does there exist a Gray code listing for the set of permutations with fixed rank k when these permutations are written as a product of k generators?

3.4.9: If we choose to represent permutations in cycle form, is there an interesting Gray code listing?

3.6.2: For which positive integers n does there exist a Gray code for the weak orders on $[n]$ allowing only elementary operations of the form $w_i \leftrightarrow w_{i+1}$ and $w_i \leftarrow w_{i+1}$?

3.6.4: For which positive integers n and h with $0 \leq h < n$ does there exist a Gray code for $\mathcal{W}(n,h)$ using the operations $w_i \leftrightarrow w_{i+1}$ and $w_i \leftarrow w_{i+1}$?

3.6.5: Can we find a Gray code for the set $\mathcal{B}^*(n)$ using the operations $w_i \leftrightarrow w_{i+1}$ and $w_i \leftarrow w_{i+1}$?

3.6.9: For which positive integers n does there exist a Gray code for the weak orders on $[n]$ where two consecutive objects always differ in exactly one position?

3.7.7: Is there a Gray code listing for the set of all partitions of an n-set with part size at most k?

3.8.6: Is there a Gray code for projective planes or affine planes?

3.8.8: Does there exist a TTS(v) with $v \equiv 0, 6, 9 \pmod{12}$ that admits a 2-intersecting Gray cycle?

3.8.10: Suppose that the design $(X, \mathcal{B})$ is the direct product of two Steiner triple systems that admit Gray codes. Does $(X, \mathcal{B})$ also admit a Gray code under the same (or a similar) minimal change property?

7.2 Universal Cycles

4.2.5: Is there a universal cycle on the set of words of length $2n$ of balanced parentheses using some representation?

4.3.2: Is there a universal cycle on the set of fixed weight words?

4.4.6: What sets of restricted permutations of $[n]$ allow universal cycles?

4.5.2: Given $k \in \mathbb{Z}^+$, there is some $n_0(k) \in \mathbb{Z}^+$ so that for all $n \geq n_0(k)$, a universal cycle for k-subsets of $[n]$ exists, provided k divides $\binom{n-1}{k-1}$.

4.7.1: For what values of n is there a universal cycle for:

1. the set of ordered partitions of n?
2. the set of unordered partitions of n?
3. the set of k-partitions of n?

4.7.2: For what values of n is there a universal cycle for the set of Fibonacci sequences?

4.7.3: What is the smallest number of symbols needed to find a universal cycle for the Fibonacci sequences of n?

4.7.7: Under the correspondence from Result 3.7.4, what does the set $\mathcal{W}_k(n)$ correspond to in ordered partitions of an n-set?

4.8.4: For each $v \equiv 10 \pmod{12}$, $v \geq 22$, there exists a TTS(v) that admits a rank three ucycle.

4.8.5: For each $v \equiv 1 \pmod{12}$, with $v \equiv 0 \pmod{5}$ there exists a TTS(v) that admits a ucycle of rank three.

4.8.6: For each $v \equiv 4 \pmod{12}$, with $v \equiv 0 \pmod{5}$, there exists a TTS(v) that admits a ucycle of rank three.

7.3 Overlap Cycles

5.2.2: For what restricted subsets of $\mathcal{B}(n)$ do there exist alphabet overlap cycles?

5.3.2: For what restricted subsets of $\mathcal{B}^m(n)$ do there exist alphabet overlap cycles?

5.5.1: For what values of n, k, s does there exist an overlap cycle $O\left(\binom{[n]}{k}, s\right)$?

5.5.2: Is there a necessary condition for overlap cycles for k-subsets of $[n]$, as Theorem 4.5.1 is for ucycles?

5.7.1: For what values of n and s does there exist an overlap cycle $O(P(n), s)$?

5.7.2: For what restricted subsets of $P(n)$ can we find an overlap cycle?

5.7.6: For what types of sets can we find overlap cycles for the set of unordered partitions of the set?

5.8.1: For what types of designs do there exist overlap cycles?

5.8.7: Can we generalize Result 5.8.6 using the Sum Construction with $\lambda_1, \lambda_2 \in \mathbb{Z}^+$?

5.8.43: For which of Hanani's six SQS constructions can we find 2-overlap cycles?

BIBLIOGRAPHY

[1] B. Alspach, K. Heinrich, and B. Mohar, *A note on hamilton cycles in block-intersection graphs*, Finite Geometries and Combinatorial Designs - Contemporary Mathematics **111** (1990), 1–4.

[2] J.-L. Baril, *Gray code for permutations with a fixed number of cycles*, Discrete Mathematics **30** (2007), 1559–1571.

[3] J.-L. Baril and V. Vajnovszki, *Gray code for derangements*, Discrete Applied Mathematics **140** (2004), 207–221.

[4] A. Bechel, B. LaBounty-Lay, and A. Godbole, *Universal cycles of discrete functions*, Congr. Numer. **189** (2008), 121–128.

[5] A. Blanca and A.P. Godbole, *On universal cycles for new classes of combinatorial structures*, SIAM Journal on Discrete Mathematics **25** (2011), 1832–1842.

[6] P. Cameron, *Combinatorics: Topics, techniques, algorithms*, Cambridge University Press, 1995.

[7] F. Chung, P. Diaconis, and R. Graham, *Universal cycles for combinatorial structures*, Discrete Mathematics **110** (1992), 43–59.

[8] ———, *Universal cycles for combinatorial structures*, Discrete Mathematics **110** (1993), 43–59.

[9] M. B. Cohen and C. J. Colbourn, *Optimal and pessimal orderings of Steiner triple systems in disk arrays*, Theoretical Computer Science **297** (2003), 103–117.

[10] C. J. Colbourn and A. Rosa, *Triple systems*, Oxford Mathematical Monographs, 1999.

[11] D. Curtis, T. Hines, G. Hurlbert, and T. Moyer, *Near-universal cycles for subsets exist*, SIAM Journal of Discrete Mathematics **23** (2009), no. 3, 1441–1449.

[12] N. G. de Bruijn, *A combinatorial problem*, Nederl. Akad. Wetensch **49** (1948), 758–764.

[13] M. Dewar, *Gray codes and universal cycles for designs*, Ph.D. thesis, Carleton University, 2007.

[14] R. Diestel, *Graph theory*, electronic ed., Springer-Verlag Heidelberg, New York, 2005.

[15] W.M.B. Dukes, M.F. Flanaga, T. Mansour, and V. Vajnovszki, *Combinatorial Gray codes for classes of pattern avoiding permutations*, Theoretical Computer Science **396** (2008), 35–49.

[16] P. Eades and B. McKay, *An algorithm for generating subsets of fixed size with a strong minimal change property*, Information Processing Letters **19** (1984), 131–133.

[17] M.C. Er, *On generating the n-ary reflected Gray codes*, IEEE Transactions on Computers **C-33** (1984), no. 8, 739–741.

[18] H. Fredricksen and J. Maiorana, *Necklaces of beads in k colors and k-ary de Bruijn sequences*, Discrete Mathematics **23** (1978), 207–210.

[19] A. Godbole, D. Knisley, and R. Norwood, *On α-overlap graphs*, Proc. of the Forty-First Southeastern International Conference on Combinatorics, Graph Theory, and Computing, Congr. Numer. **204**, 2010, pp. 161–171.

[20] I. J. Good, *Normally recurring decimals*, J. London Math. Soc. **21** (1946), 167–179.

[21] F. Gray, *Pulse code communication*, U.S. Patent 2632058, Filed November 13, 1947, Issued March 1953.

[22] H. Gupta, *On permutation-generating strings and rosaries*, Lecture Notes in Mathematics: Combinatorics and Graph Theory (1981), 272–275.

[23] M. Hanani, *On quadruple systems*, Canadian Journal of Math. **12** (1960), 145–157.

[24] D. R. Hare, *Cycles in the block-intersection graph of pairwise balanced designs*, Discrete Mathematics **137** (1995), 211–221.

[25] G. Hurlbert, *Universal cycles: on beyond de Bruijn*, Ph.D. thesis, Rutgers University, 1990.

[26] ———, *On universal cycles for k-subsets of an n-set*, SIAM Journal of Discrete Mathematics **7** (1994), no. 4, 598–604.

[27] ———, *Multicover ucycles*, Discrete Mathematics **137** (1995), 241–249.

[28] G. Hurlbert and G. Isaak, *Equivalence class universal cycles for permutations*, Discrete Mathematics **149** (1996), 123–129.

[29] B. Jackson, B. Stevens, and G. Hurlbert, *Research problems on Gray codes and universal cycles*, Discrete Mathematics **309** (2009), 5341–5348.

[30] B. W. Jackson, *Universal cycles for k-subsets and k-permutations*, Discrete Mathematics **117** (1993), 141–150.

[31] J. R. Johnson, *Universal cycles for permutations*, Discrete Mathematics **309** (2009), 5264–5270.

[32] S. M. Johnson, *Generation of permutations by adjacent transposition*, Mathematics of Computation **17** (1963), no. 83, 282–285.

[33] J. T. Joichi and D. E. White, *Gray codes in graphs of subsets*, Discrete Mathematics **31** (1980), no. 1, 29–41.

[34] P. Kaski, P.R.J. Östergård, O. Pottonen, and L. Kiviluoto, *A catalogue of the Steiner triple systems of order 19*, Bull. Inst. Combin. Appl **57** (2009), 35–41.

[35] R. Kaye, *A Gray code for set partitions*, Information Processing Letters **5** (1976), no. 6, 171–173.

[36] D. E. Knuth, *The art of computer programming*, vol. 4, Addison-Wesley Professional, 2005.

[37] P.J. Koutas and T.C. Hu, *Shortest string containing all permutations*, Discrete Mathematics **11** (1975), 125–132.

[38] C. C. Lindner and A. Rosa, *On the existence of automorphism free Steiner triple systems*, Journal of Algebra **34** (1975), 430–443.

[39] C. Flye-Sainte Marie, *Solution to problem number 58*, l'Intermediaire des Mathematiciens **1** (1894), 107–110.

[40] R. Mathon and A. Rosa, $2-(v,k,\lambda)$ *designs of small order*, The CRC Handbook of Combinatorial Designs (C.J. Colbourn and J.H.Dinitz, eds.), CRC Press, Boca Raton, FL, 2007, pp. 25–58.

[41] A. Nijenhuis and H. S. Wilf, *Combinatorial algorithms: For computers and calculators*, second ed., Academic Press, Inc., 1978.

[42] D. Pike, *Hamilton decompositions of block-intersection graphs of steiner triple systems*, Ars Combinatorica **51** (1999), 143–148.

[43] C. D. Savage, *Gray code sequences of partitions*, Journal of Algorithms **10** (1989), 577–595.

[44] R. P. Stanley, *Enumerative combinatorics: Volume i*, Cambridge University Press, 1997.

[45] B. Stevens, P. Buskell, P. Ecimovic, C. Ivanescu, A. Muslim Malik, A. Savu, T. S. Vassilev, H. Verrall, B. Yang, and Z. Zhao, *Solution of an outstanding conjecture: the non-existence of universal cycles with* $k = n - 2$, Discrete Mathematics **258** (2002), 193–204.

[46] D. R. Stinson, *Combinatorial designs: Constructions and analysis*, Springer-Verlag New York, Inc., 2004.

[47] T. Ueda, *Gray codes for necklaces*, Discrete Mathematics **219** (2000), 235–248.

[48] T. R. Walsh, *Loop-free sequencing of bounded integer compositions*, J. Combin. Math. Combin. Comput. **33** (2000), 323–345.

[49] D. B. West, *Introduction to graph theory*, second ed., Prentice Hall, 2001.

[50] H. S. Wilf and A. Nijenhuis, *Combinatorial algorithms: An update*, Society for Industrial Mathematics, January 1, 1987.

[51] Aaron Williams, *Solving Wilf's sigma-tau*, SIAM Conference on Discrete Mathematics (2012).