Coding Theory of Permutations/Multipermutations and in the LEAN Theorem Prover

May 2018

Chiba University
Graduate School of Science

Division of

Department of Mathematics and Informatics

Justin Kong

for Lukela

I. Abstract

This thesis investigates problems in Coding theory, a fundamental field of study for the reliable transmission or storage of data. Contributions are in two general areas, permutation/multipermutation codes and coding theory formalization. Linear Codes are well studied, but less is known about nonlinear codes. Permutation and multipermutation codes are types of nonlinear codes. Formalization involves the precise statement of mathematical theorems in a specified language in such a way that the veracity of these theorems may be verified algorithmically. Although coding theory is a field of study well-suited for formalization, work in coding theory formalization has been relatively limited.

In this thesis we advance the study of permutation and multipermutation codes in three ways. First, we extend an LP (linear programming) decoding method originally designed for permutation codes in the Euclidean metric to permutation codes in the Kendall tau metric. Second, we provide new methods for calculating Ulam permutation sphere sizes. The results are used to present new bounds on the maximum possible size of permutation codes in the Ulam metric, including the fundamental theorem that nontrivial perfect Ulam permutation codes do not exist. Third, new methods for calculating Ulam multipermutation sphere sizes for certain parameters are provided as well as resulting bounds on the maximum possible code size.

In this thesis we advance the study of formalization in coding theory in two ways. Contributions are made using the Lean theorem proof assistant, a software designed specifically for formalization. The first contribution is a set of files that formalize relevant basic mathematical definitions and theorems as well as the famous Repetition codes and Hamming (7,4) code. The second contribution is a set of files that formalize Levenshtein codes and related definitions and theorems. Levenshtein codes are a type of nonlinear deletion and/or insertion correcting code.

II. Acknowledgements

Firstly, I would like to thank my Lord and Savior Jesus Christ, who has lead me throughout my journey of education and without whom I cannot so much as breathe.

I am grateful for the love and support of my family, especially my wife Keiko who has been a constant encouragement.

I am indebted greatly to my advisor, Dr. Manabu Hagiwara, who has patiently guided me step by step throughout my post-graduate studies. I am constantly amazed by his mathematical prowess and insight, and simultaneous wisdom and kindness.

Contents

III. List of Figures

IV. LIST OF TABLES

1. Introduction

In the information age that we live in, data is communicated and stored through a plethora of digital devices. Without the advent of coding theory, reliable communication and data storage would not be possible. Error-correcting codes, explained in chapter 2, are utilized constantly to combat the inherent noise of communication channels. The study of these codes is called coding theory.

Codes have also been improving since their advent in the late 1940's and early 1950's. With their improvement communication becomes faster and data storage denser. In fact, current codes can approach known limits to what is possible for certain communication schemes. However, the quest to improve codes and to expand their underlying theory is not over by any means.

A. Contributions

The main contributions of this thesis can be divided into two large categories: 1) contributions to coding theory in the area of permutation/multipermutation codes and 2) contributions to coding theory in the area of formalization. We begin by providing some context for each contribution before stating the actual results of each category.

The first category of contribution is to the field of permutation and multipermuation codes. Permutation and multipermutation codes are types of nonlinear codes. Linear codes are well studied and largely well-understood because of their close relation to linear algebra. On the other hand, there remains much mystery in the field of nonlinear codes. Depending on the physical scenario, nonlinear codes may demonstrate superior performance over linear codes. Hence it is worthwhile to devote some energy to research in this area.

One category of nonlinear codes are permutation and multipermutation codes. Certain properties, discussed in later chapters, make permutation and multipermutation codes attractive candidates for use in applications such as flash memory devices. There are several questions that must be answered before this is possible, and it is also necessary to generally expand the theory of permutation and multipermutation codes.

Our main contributions to permutation/multipermutation codes are threefold. We state them here, although some explanation of the terminology of these statements is found in subsequent chapters. First, we extend a decoding method to a new class of permutation codes, calling them Kendall tau LP-decodable permutation codes. Second, we provide new ways of calculating Ulam permutation sphere sizes, and use this to prove the fundamental bound on maximal code size that perfect Ulam permutation codes do not exist. Third, we provide new ways of calculating Ulam multipermutation sphere sizes, and use this to prove new bounds on the maximal code size of Ulam multipermutation codes. These results were partially published in [37, 51, 52].

The second category of contribution is to the field of formalization in coding theory. Mathematical formalization involves the precise statement of mathematical theorems in a specified language in such a way that the veracity of these theorems may be verified algorithmically. When an error-correcting code fails to perform its purported function, errors in communication may result. Moreover, the study of coding theory is often complex and proofs of the properties of a particular error-correcting code may be difficult to understand or verify for non-specialists in coding theory. Even among coding theorists, the vast quantity of coding schemes can sometimes lead to miscommunication.

In the future, formalization may become a valuable tool for verifying properties of codes. It may also be a unifying force, as formalization removes the ambiguity of language. Current work in the area of the formalization of coding theory is limited. Libraries of coding theory are necessary to aid future work, since any definitions or lemmas that are formalized in a proof-assistant can be subsequently applied in future formalization.

As our main contribution to formalization in coding theory, we provide, to the best of our knowledge, the first coding theory library for the Lean theorem proof assistant. The library is called “Cotoleta” (COding Theory Over the LEan Theorem proof Assistant). The Lean theorem proof assistant is a free software released by Microsoft Research in cooperation with Carnegie Mellon University. Our library includes a couple of key components. First, it includes structures that serve as templates for formalizing error correcting systems, and provides examples of these with two well-known codes, repetition codes and the Hamming (7,4) code. Second, it includes the formalization of definitions and lemmas related to Levenshtein codes, a type of deletion-correcting code. These results were partially published in [38].

B. Organization of Thesis

The thesis is organized as follows. Chapter 2 introduces some pertinent basic ideas and terminology of coding theory. It also provides a literature review of permutation/multipermutation codes and formalization in coding theory. The next three chapters contain the main contributions to the field of coding theory of permutations and multipermutations. In Chapter 3, Kendall tau LP-decodable permutation codes are introduced, and some of their properties are examined. The chapter includes a necessary and sufficient condition for extending LP (linear programming) decoding methods as well as explicit examples of codes satisfying this condition. Chapter 4 discusses Ulam permutation codes and proves the nonexistence of nontrivial perfect Ulam permutation codes. Chapter 5 considers Ulam multipermutation codes, and provides new bounds on maximal code size.

Chapter 6 contains the main contributions to the field of coding theory formalization. It introduces Cotoleta, the library for coding theory over the Lean theorem proof assistant. Some of the important definitions, theorems, and examples in this library are presented. Finally, Chapter 7 concludes the thesis with some remarks on the overall research and possible future directions.

2. Preliminaries and Literature Overview

A. Introduction

Coding theory is both a narrow and a broad field of study. It is narrow in the sense that it revolves around a single central topic: the properties and construction of error-correcting codes. It is broad in the sense that the variety of codes is limitless, and their properties and the approaches to their study is equally multifaceted.

In Sections 2-B and 2-C of this chapter, we introduce some of the basic ideas of coding theory pertinent to the current work. Other ideas are introduced in their respective chapters as necessary. The remaining sections provide an overview of some of the relevant coding theory literature, particularly in the fields of permutation/multipermutation codes and formalization.

B. The Idea of Coding

Coding theory, or the study of error-correcting codes, is an answer to the problem of sending or storing data reliably. What is a code, and what is its purpose? We provide here a rudimentary look at the basic idea behind error-correcting codes. This is only meant to provide some context for subsequent chapters, with the goal of keeping the current work as self-contained as possible. This section is purposely broad and more precise definitions are introduced for specific codes in later sections. For more detailed information on Coding theory, the reader is referred to texts such as [12, 63, 66].

Firstly, let us describe one situation in which a code might arise. This is not the only situation, but it illuminates the major concepts of coding theory. For this purpose, we recall the famous engineering problem of Alice sending a message to Bob. Alice wishes to send a message m of some set M of possible messages to Bob through a noisy channel. By noisy channel, we mean that the message Alice intends to send to Bob may be corrupted (altered) as it passes through the channel. The type of corruption that can occur will depend upon the channel in question. For example, perhaps Alice wishes to send one of two possible messages, 0 or 1, corresponding to no and yes. She may send a 0, but because of noise it is perceived by Bob as a 1. Or perhaps she sends a 0 but because of noise no message is perceived at all. Of course the type of possible noise is not limited to these examples.

Rather than simply sending the original message across the channel, Alice can add some sort of redundancy to her message (and possibly change the format altogether) to increase the probability of it successfully reaching Bob. The process of adding redundancy is called encoding, and the set C (typically a subset of a larger set X) of encoded messages is called a code. Members $c \in C$ are called codewords. Alice transmits a codeword through the channel instead of the original message, and as a result of noise, Bob receives a potentially altered word $\hat{c}$ at the other end of the channel. This received word belongs to some superset $Y \supseteq C$ of possible receivable words. The received word $\hat{c}$ is then decoded to obtain an estimation $\hat{m} \in M$ of the original message sent by Alice. If the decoding is successful, then $m = \hat{m}$.

The most well-known and well-studied class of codes are linear codes. A linear code may be defined as follows. Let F_q be a finite field of cardinality q, and F_q^n the n-dimensional vector space over F_q. A linear code C is any k-dimensional linear subspace of F_q^n, where n and k are positive integers with $k \leq n$. The term linear comes from the fact that any linear combination of codewords is again another codeword (a characteristic of being a linear subspace). Referring to the previous explanation, the message set M is typically taken to be the set F_q^k of k-length vectors over F_q, of which there are q^k. This means that the encoding process takes a k-length vector and converts it to an n-length vector. In this context the parameter k is known as the number of information bits and the parameter n is known as the length of the code.

The rate of a linear code is defined as the ratio between the number k of information bits and the length n of the code. In other words, the rate R of a linear code is defined as k/n. Notice that $k/n = (\ln(q^k))/(\ln(q^n)) = (\ln \#F_q^k)/(\ln \#F_q^n) = (\ln \#C)/(\ln \#F_q^n)$, where $\#A$ is notation for the cardinality of the set A. Hence the rate may also be defined as the ratio between the natural log of the size $\#C$ of the code, and the natural log of the size of F_q^n, of which C is a linear subspace. Later we will define rate of permutation codes analogously. A large rate is desirable, as it corresponds to faster communication through a channel (or to denser storage in a similar context). At the same time, having no redundancy means a lower probability of successful transmission or storage of a message.

One of the triumphs of Claude Shannon's famous 1948 paper, "A mathematical theory of communication" ([68]), was to show that it is possible to successfully transmit a message through a noisy channel with arbitrarily high probability for any rate below what he defined as the capacity of the channel. However, no practical codes were provided at that time. A big question of coding theory is how to choose codes C with high probability of successful communication, i.e. arbitrarily small probability of error, while maintaining high rates.

As the name suggests, an error-correcting code is simply a code where errors that occur as a result of a noisy channel can be corrected. Correcting errors introduced by the noisy channel enables the original intended message to be recovered. One key concept in error-correcting codes relevant to the current work is the idea of minimum distance. This is discussed in the next section.

C. Minimum Distance in Coding Theory

We begin the discussion of minimum distance by considering the most commonly studied distance in coding theory, the Hamming distance. Given vectors $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ in F_q^n, the Hamming distance $\mathrm{d}_H(\mathbf{x}, \mathbf{y})$ between $\mathbf{x}$ and $\mathbf{y}$ is defined as $\mathrm{d}_H(\mathbf{x}, \mathbf{y}) := \#\{i \in [n] \ : \ x_i \neq y_i\}$. In the definition, $[n]$ is notation for the set of integers $\{1, 2, \ldots, n\}$. In words, the Hamming distance is simply the number of pairwise disagreements between two vectors of equal length. The minimum Hamming distance d of a linear code C is defined as $d := \min\{d_H(c_1, c_2) \in \mathbb{Z}_{\geq 0} \ : \ c_1, c_2 \in C, c_1 \neq c_2\}$.

In terms of the communication model, suppose that a codeword $\mathbf{c} = (c_1, c_2, \ldots c_n)$ is sent through the channel one element at a time, beginning with c_1 and ending with c_n. The Hamming distance corresponds to any errors introduced by noise in the channel that causes anything other than the transmitted c_i's to be received. For example, suppose $(0, 0, 0, 0, 0)$ is sent through the channel but $(0, 1, 0, 0, 3)$ is received. In such a situation we would say that 2 disagreement errors occurred, and accordingly the Hamming distance between $(0, 0, 0, 0, 0)$ and $(0, 1, 0, 0, 3)$ is 2. Given $t \in \mathbb{Z}_{\geq 0}$, whenever the minimum Hamming distance of a linear code C is $2t+1$ or greater, then t disagreement errors of the nature described above are correctable.

To see why t disagreement errors are correctable given a minimum Hamming distance of at least $2t+1$, suppose that any codeword c is sent through a channel. If t or fewer errors occur, then the resulting received word $\hat{c}$ will be within Hamming distance t of the transmitted codeword c. Meanwhile, since all other codewords are at least distance $2t+1$ from c, it follows that $\hat{c}$ is at least distance $t+1$ away from all other codewords that are not c. Hence the transmitted codeword c is uniquely determined by simply choosing the codeword that has the smallest Hamming distance from $\hat{c}$.

The Hamming distance is appropriate for certain types of errors and codes, but not for others. For instance, the Hamming distance can account for errors known as bit flips, where a 0 or 1 is sent through a channel but the opposite is received (1 instead of 0, or 0 instead of 1). In this case the length of a transmitted vector is preserved so that the Hamming distance makes sense and it also corresponds exactly to the number of errors that occur. However, it is not suited for a situation, for example, when an element within a transmitted codeword is deleted, causing the received word to be of a shorter length than the original sent codeword.

Moreover, until now we have focused on the linear code case, but there are other classes of codes. For instance, permutation codes, defined in the next section, are nonlinear codes. A plethora of distance metrics other than the Hamming distance may be applied in coding theory. Different metrics are more appropriate for different types of codes or errors. The current work considers several metrics: the Hamming metric, the Kendall tau metric, the Euclidean metric, the Ulam metric, and the Levenshtein metric. While the metrics differ, the general idea is preserved: the larger the minimum distance of a code, the more errors are correctable.

D. Permutation/Multipermutation Codes Literature Review

In this subsection we briefly review some of the publications related to permutation and multipermutation codes. In 1965, D. Slepian published a paper entitled “Permutation Modulation” [69], where he constructed a code by beginning with an initial sequence and then taking all distinctive sequences formed by permuting the order of the numbers in this sequence. The class of codes he constructed was suitable for the transmission of digital information in the presence of white Gaussian noise, a commonly used noise model. The codes introduced by Slepian may be considered to be the first instance of permutation codes.

Efficient encoding is one aspect necessary for practical implementation of error-correcting codes. Berger *et. al.* considered the problem of encoding for permutation codes introduced previously by Slepian [11]. They demonstrated the relative simplicity of encoding these permutation codes.

As mentioned previously, another question in coding theory concerns the limits of maximal code size for a given minimum distance. One technique to determine bounds on permutation code size involves calculating spheres, or the number of permutations within a given distance (radius). Deza and Frankl used calculations of permutation sphere sizes under the Hamming distance to determine upper and lower bounds on permutation code size utilizing the Hamming distance [25]. Other aspects of permutation codes in the Hamming metric were further expounded upon by Blake, Cohen, and Deza, including questions of decoding algorithms [14, 15, 23].

Permutation codes have been suggested for a number of practical applications. One such application of note was the use of permutation codes in powerline communications, proposed by Vinck in 2000 [72, 73]. The proposal incorporated the Hamming distance for error-correction. Chu *et. al.* later considered constructions of permutation codes for use in powerline communications [21]. A decoding method for permutation codes for powerline communications applying distance-preserving mapping algorithms was discussed in [71].

Another recent application of permutation codes is to DNA storage. Church *et. al.* ([22]) and subsequently others ([16, 34]) demonstrated a remarkable proof of concept of the use of artificial DNA for the storage of digital data. This has profound implications on potential data storage density. Kiah *et. al.* showed in [49] that it is possible to integrate rank modulation codes (permutation codes) into DNA coding schemes, with an advantage of high error tolerance. In 2017 Raviv *et.al.* provided an upper bound on the number of feasible permutations under the aforementioned scheme.

One more application of permutation codes, which renewed interest in the subject since Jiang *et. al.* first proposed it in 2009, is the application of permutation codes to flash memory [44].

The scheme described would improve the stability and efficiency in programming flash memory cells. Their proposal utilized the Kendall tau metric, which measures the minimum number of adjacent transpositions to transform one permutation into another.

Barg *et. al.* and others further investigated permutation codes in the Kendall tau metric. They provided upper and lower bounds on the maximum size of codes (including upper bounds based on sphere-size calculations) and presented code constructions [10, 19, 28, 57, 61]. Snake-in-the-box-codes (also called Gray codes) for rank modulation were also studied in the Kendall tau metric in recent years [41, 76]–[78]. These codes are a type of permutation code that can detect single errors and have the quality that successive codewords are obtainable by a single "push-to-the-top" operation, an operation that played a pivotal role in Jiang *et. al.*'s original scheme.

In 2013 and 2014, Farnoud *et. al.* proposed the use of the Ulam metric in permutation and multipermutation codes, again with applications to flash memory [29, 30]. While the Kendall tau metric is appropriate to model certain errors in flash memory, the Ulam metric would better model errors in certain situations resulting from faulty cells in flash memory devices. They provided some loose bounds on the maximal possible code size of both permutation and multipermutation code size in the Ulam metric. Gologlu *et. al.* gave new bounds for the maximal possible code size of permutation codes in the Ulam metric [35]. They employed integer programming techniques for these new bounds.

E. Formalization in Coding Theory Literature Review

Interest in formalization has been increasing in recent years. Numerous papers have been published on the subject, including formalization papers related to information theory and coding theory [1]–[3, 5, 8, 38, 62]. A journal dedicated to "automated or semi-automated formalization efforts in any area,"[1] called the "Journal of Formalized Reasoning" was also established in 2009. More recently, Microsoft Research and Carnegie Mellon University released a new open source theorem prover called Lean in 2015 (our research in formalization focuses on formalization utilizing Lean).

[1]from "Journal of Formalized Reasoning" website, https://jfr.unibo.it

Information theory is closely related to coding theory. Among other things, it establishes theoretical bounds on what is possible for codes. One of the prominent libraries for the formalization of coding theory is the "*infotheo*" library for Coq, developed by Affeldt, Garrigue, and others. As the name suggests, this library also formalizes fundamental definitions and theorems of information theory, including perhaps most notably formalizations of Shannon's famous theorems as detailed in [5]. This library also includes the formalization of several famous codes, including Hamming codes, Reed-Solomon codes, and LDPC codes [2, 4].

Another well-known formalization software is the HOL theorem prover, which was also released in the 1980's. The lines of code referenced in [58] and [60] for the HOL theorem prover contain information theory-related formalization. Both this work as well as the *infotheo* library formalize the AEP (Asymptotic Equipartition Property) as well as concepts of entropy. A considerable amount formalization for HOL (as well as another well-known theorem prover, Isabelle), was also completed in in [40], including notions of entropy, mutual information, and Markov chains. These are fundamental topics of information theory, and underlying concepts of coding theory.

The information theory formalization work in the HOL theorem prover also includes work to formalize general mathematics useful for future formalization such as Lebesgue integration and probability theories over the extended reals ([59]) used in the formalization in [58]. Similarly, the *infotheo* library in Coq makes use of the *mathcomp* and *ssreflect* extensions that contain general mathematical definitions and theorems that can be utilized for formalization in other areas like coding theory.

F. Conclusion

We provided in this chapter some of the basic notions and terms of coding theory. We also outlined briefly some of the literature pertinent to the current study, especially in the areas of permutation/multipermutation codes and coding theory formalization. Subsequent chapters explain the actual contributions of the current research.

3. KENDALL TAU LP-DECODABLE PERMUTATION CODES

A. *Introduction*

One way of defining a permutation code C is as an orbit $\{X\vec{\mu}\ :\ X \in G\}$, where G is a set of permutation matrices[2] and $\vec{\mu}$ is a Euclidean vector. If we consider the Euclidean distance to model the noise of a communication channel, similarly to the way we saw the Hamming distance model noise in previous examples, then one of the possible main goals of permutation codes is: "for a given Euclidean vector $\vec{\lambda}$, find an $X\vec{\mu}$ which minimizes the distance $||X\vec{\mu}-\vec{\lambda}||$ over $X \in G$ by an efficient algorithm." This would correspond to the decoding process of communication, where a received Euclidean vector is decoded to the codeword closest to it in terms of the Euclidean distance. If the cardinality of G is large, then minimizing the aforementioned distance may be computationally difficult.

In [75], Wadayama and Hagiwara introduced a novel solution to the above problem using linear programming methods. They considered the following problem[3]:

$$\text{maximize } \vec{\lambda}^T X\vec{\mu}, \text{ for fixed Euclidean vectors } \vec{\mu}, \vec{\lambda}$$

where X is taken over the Birkhoff polytope, which consists of doubly stochastic matrices, or a subset of that polytope (see the appendices for the definitions of doubly stochastic matrices and the Birkhoff polytope). They proved the fundamental theorem that a doubly stochastic matrix X_0 maximizes the problem above if and only if the matrix X_0 minimizes the linear programing problem below:

$$\text{minimize } ||X\vec{\mu} - \vec{\lambda}||, \text{ for fixed Euclidean vectors } \vec{\mu}, \vec{\lambda}$$

where X is taken over the Birkhoff polytope and the distance $||\cdot||$ is the Euclidean distance. The set of vertices of the Birkhoff polytope is exactly the set of permutation matrices. In other words, Wadayama-Hagiwara's problem is equivalent to the aforementioned permutation code problem, in the form of a linear programming (LP) problem.

[2]In some references, G is chosen as a generalized permutation group, e.g., a signed permutation group.

[3]The original problem is to maximize $\text{Trace}(\vec{\mu}\vec{\lambda}^T X)$. It is directly obtained that $\text{Trace}(\vec{\mu}\vec{\lambda}^T X) = \text{Trace}(\vec{\lambda}^T X\vec{\mu}) = \vec{\lambda}^T X\vec{\mu}$

This implies that LP techniques may be applied for decoding if G is the set of all permutation matrices for a given dimension. The reader may have the question: “Is it possible to apply this approach to a subset of permutation matrices?” The answer is yes. In [75], new classes of permutation codes were proposed by considering sub-polytopes of the Birkhoff polytope. However, these new classes potentially contain fractional vertices in which case its sub-polytope contains vertices which are not permutation matrices. A novel method of constructing compact constraints called consolidation was introduced in the first half of [37]. Some of the important definitions and the major theorem that establishes the viability of consolidation as a method of constructing compact constraints are stated in Appendix A.

B. Extending LP decoding Methods

The LP decoding methods expressed in the introduction of this chapter were originally intended for permutation codes where the metric in use is the Euclidean metric. With the proper choice of polytope, decoding to the nearest (in the Euclidean metric) geometric vertex will yield the same result as when decoding to the nearest permutation in terms of the Kendall tau distance, which we define subsequently. When the linear constraints are compact, LP decoding will naturally yield a permutation matrix. We provide in Definition 3-B and subsequent lemmas a necessary and sufficient condition for the LP decoding methods proposed in [75] to be utilized to detect/correct Kendall tau errors.

Extending LP-decoding in this manner is significant, since the Kendall tau distance has been of recent interest in the context of permutation codes particularly in coding for flash memory. Rank modulation was suggested by Jiang. et al. for improving the efficiency and stability of flash devices [44, 45]. In this scheme the Kendall tau distance was the primary metric and was used to account for errors of the relative order or rank in permutation entries. Moreover, it was proven by Buchheim *et. al.* that finding an element of a permutation group with minimal Kendall tau distance from a given permutation of S_n is an NP-complete problem [18]. However, linear programming problems are guaranteed to be solvable in polynomial time [48], meaning that we have an efficient decoding algorithm to correct errors in the Kendall tau distance whenever the LP-decoding extension condition is satisfied.

We begin our discussion by introducing notation and definitions necessary to read this chapter. The permutation group S_n on $\{0, 1, \ldots, n-1\}$ may be embedded into the set $M_n(\mathbb{R})$ of matrices by associating each permutation σ of S_n with an n-by-n matrix $X^\sigma := \delta_{j=\sigma(i)}$, where δ is the Kronecker's delta. For the ease of discussion of metrics, we prefer the following notation for subsequent sections. For any permutation σ in S_n, we use the notation $\sigma = [\sigma_0, \sigma_1, \ldots, \sigma_{n-1}]$, where σ_i is in the set $\{0, 1, \ldots, n-1\}$ and whenever $i \neq j$, then $\sigma_i \neq \sigma_j$. We may define an action of S_n on $\mathbb{R}^n$ by allowing $\sigma = [\sigma_0, \sigma_1, \ldots, \sigma_{n-1}]$ to be the permutation sending the ith position of a vector $\vec{\mu} \in \mathbb{R}^n$ upon which σ acts to the σ_ith position. We denote this action by $\sigma \circ \vec{\mu}$. Note that if X^σ is the permutation matrix associated with the permutation σ and $\vec{\mu} \in \mathbb{R}^n$, then the action $\sigma \circ \vec{\mu}$ is equivalent to the natural action $X^\sigma \vec{\mu}$.

Permutations are multiplied in the typical manner, from right to left. The identity permutation is denoted by $e := [0, 1, \ldots, n-1] \in S_n$, while the inverse of a permutation $\sigma \in S_n$ is denoted by σ^{-1}. We embed S_n into $\mathbb{R}^n$ in the following manner. For any permutation $\sigma = [\sigma_0, \sigma_1, \ldots, \sigma_{n-1}]$ in S_n, we associate the vector $\vec{\sigma} := ((\sigma^{-1})_0, (\sigma^{-1})_1, \ldots, (\sigma^{-1})_{n-1})$ in $\mathbb{R}^n$. It is worth noting that $\sigma \circ \vec{\mu} = \overrightarrow{\sigma\mu}$ and in particular $\sigma \circ \vec{e} = \vec{\sigma}$.

It is known that the Kendall tau distance between two permutations σ and τ is equivalent to the minimum number of pairwise adjacent transpositions necessary to transform σ into τ, or τ into σ [26]. We denote this distance by $\mathrm{d}_K(\sigma, \tau)$. The precise definition is as follows.

Definition (Kendall tau Distance)**.** Given $\sigma, \tau \in S_n$, the Kendall tau distance $\mathrm{d}_K(\sigma, \tau)$ between σ and τ is defined as

$$\mathrm{d}_K(\sigma, \tau) := \#\{(i, j) \ : \ 0 \leq i < j \leq n-1, (\sigma^{-1}\tau)_i > (\sigma^{-1}\tau)_j\},$$

where recall that $\#A$ denotes the cardinality of the set A.

It is also known that the Kendall tau distance is left-invariant, *i.e.,* given any $\sigma, \tau, \lambda \in S_n$, then $\mathrm{d}_K(\sigma, \tau) = \mathrm{d}_K(\lambda\sigma, \lambda\tau)$. The standard Euclidean distance between two vectors $\vec{\sigma}$ and $\vec{\tau}$ is denoted by $\mathrm{d}_E(\vec{\sigma}, \vec{\tau})$ in this paper. The following table compares the Kendall tau and Euclidean distances between some permutations and their associated vectors respectively.

TABLE I
KENDALL TAU, EUCLIDEAN DISTANCES

σ	τ	$\mathrm{d}_K(\sigma,\tau)$	$\mathrm{d}_E(\vec{\sigma},\vec{\tau})$
$[1,0,3,2]$	$[1,0,3,2]$	0	0
$[1,0,3,2]$	$[1,0,2,3]$	1	$\sqrt{2}\approx 1.1421$
$[0,1,2,3]$	$[1,0,3,2]$	2	2
$[0,1,2,3]$	$[0,3,1,2]$	2	$\sqrt{6}\approx 2.44949$
$[0,1,2,3]$	$[1,3,2,0]$	4	$\sqrt{14}\approx 3.74166$
$[0,1,2,3]$	$[2,3,0,1]$	4	4
$[1,0,3,2]$	$[2,3,0,1]$	6	$2\sqrt{5}\approx 4.4721$

We are now equipped to discuss what it means for a code to be Kendall tau LP-Decodable. That is, we are able to explain the conditions under which LP-decoding methods may be used to correct Kendall tau metric errors. Let us first consider a small subgroup example where LP-decoding can be used to correct Kendall tau errors.

As discussed previously, there exists a compact constraint set $\mathcal{L}$ such that $\mathrm{Ver}(D[\mathcal{L}]) = C_n \subset S_n$, where C_n is the cyclic group of order n. This makes C_n a natural candidate for extending LP-decoding. Moreover, a construction for C_n using only the "push-to-the-top" operation was provided in [44]. The "push-to-the-top" operation was proposed as a promising improvement over programming operations currently employed in flash memory technology.

As a small example, consider the permutation code $(C_4, \vec{e})$ consisting of four cyclic rotations of the identity vector. Explicitly, the codewords are: $(0,1,2,3)$, $(1,2,3,0)$, $(2,3,0,1)$, and $(3,0,1,2)$, with minimum Euclidean distance of about 3.4641. The associated permutations for these codewords are $[0,1,2,3]$, $[1,2,3,0]$, $[2,3,0,1]$, and $[3,0,1,2]$ respectively, with minimum Kendall tau distance of 3 between permutations.

Suppose now that the codeword $(0,1,2,3)$ is transmitted, but the corrupted message $(1,0,2,3)$ is received. Notice that in this instance sufficient corruption occurred so that the relative ranks of vector components were altered. The following table shows a comparison between the Euclidean distances between the vector $(1,0,2,3)$ and codewords, and the Kendall tau distances between the permutation [1,0,2,3] and the permutations associated with each codeword.

From Table II we can see that the received vector $(1,0,2,3)$ would be decoded to the codeword

TABLE II
EUCLIDEAN, KENDALL TAU DISTANCES FROM $(1,0,2,3)$ AND $[1,0,2,3]$

Codeword	d_E from $(1,0,2,3)$	Permutation	d_K from $[1,0,2,3]$
$(0,1,2,3)$	1.4142	$[0,1,2,3]$	1
$(1,2,3,0)$	3.7417	$[1,2,3,0]$	4
$(2,3,0,1)$	4.2426	$[2,3,0,1]$	5
$(3,0,1,2)$	2.4495	$[3,0,1,2]$	2

$(0,1,2,3)$ based on Euclidean distance, and likewise the permutation $[1,0,2,3]$ would be decoded to $[0,1,2,3]$ based on Kendall tau distance. In fact, it is easily verified that for any element $\sigma \in S_4$, the closest codeword to $\vec{\sigma} \in \mathbb{R}^4$ in terms of d_E has an associated permutation that is also closest to $\sigma \in S_n$ in terms of d_K. We call such a code Kendall tau LP-Decodable.

Definition. [Kendall tau LP-Decodable] Let $\lambda, \mu \in S_n$, and G be a subgroup of S_n. Let $g_0 \in G$. We say $G\vec{\mu}$ is **Kendall tau LP-decodable** if the permutation code $(G, \vec{\mu})$ is first LP-decodable *i.e.*, $G_{\mathcal{L}} := \mathrm{Ver}(D_n[\mathcal{L}]) \cap S_n$ for some doubly stochastic constraint $\mathcal{L}$, and the following statement called the **LP-decoding extension condition** is satisfied.

$$\begin{aligned} &\mathrm{d}_E(\vec{\lambda}, \overrightarrow{g_0\mu}) \leq \mathrm{d}_E(\vec{\lambda}, \overrightarrow{g\mu}) \quad \text{for all } g \in G \\ \implies &\mathrm{d}_K(\lambda, g_0\mu) \leq \mathrm{d}_K(\lambda, g\mu) \quad \text{for all } g \in G. \end{aligned} \tag{1}$$

In this scheme, suppose a potentially corrupted transmitted vector $\vec{\lambda}$ is received. The decoder will attempt to find the closest codeword $g_0 \circ \vec{\mu} = \overrightarrow{g_0\mu}$ from $\vec{\lambda}$ in terms of d_E via linear programming methods [75]. If the LP-decoding extension condition holds, then the permutation, $g_0\mu$ associated with $\overrightarrow{g_0\mu}$ will also be the closest permutation to $\lambda \in S_n$ in terms of d_K. This essentially means that we have a polynomial time algorithm for finding/correcting Kendall tau metric errors for the right choice of subgroup of S_n. Actually, we are not limited to subgroups, but this paper focuses its analysis on subgroups because of their mathematical structure. Further research may be conducted to consider non-group subsets of S_n.

In order to simplify the comparison of Euclidean and Kendall tau distance, we make use of the left invariance of both metrics. Since Kendall tau distance is left invariant, there always exists a permutation such that the distance between two permutations can be rewritten in terms of the distance between the identity permutation and some other permutation. More precisely, $\mathrm{d}_K(\sigma,\tau) = \mathrm{d}_K(e,\sigma^{-1}\tau)$. The distance $\mathrm{d}_K(e,\sigma)$ between the identity permutation e and $\sigma \in S_n$ is a previously studied value called the length of σ [43]. The Kendall tau distance between the identity e and an element has also been referred to by Barg et al. as the weight of that element. We will denote the Kendall tau weight of a permutation σ by $\mathrm{wt}_K(\sigma)$. [10].

Similarly to Kendall tau distance, Euclidean distance is also left invariant, *i.e.* given $\lambda \in S_n$ and $\vec{\sigma}, \vec{\tau}$ in $\mathbb{R}^n$, we have $\mathrm{d}_E(\vec{\sigma},\vec{\tau}) = \mathrm{d}_E(\lambda\vec{\sigma},\lambda\vec{\tau})$. Hence since $\mathrm{d}_E(\lambda\vec{\sigma},\lambda\vec{\tau}) = \mathrm{d}_E(\overrightarrow{\lambda\sigma},\overrightarrow{\lambda\tau})$, then $\mathrm{d}_E(\vec{\sigma},\vec{\tau}) = \mathrm{d}_E(\vec{e},\overrightarrow{\sigma^{-1}\tau})$. Thus, it is natural to consider the distance between an element of $\mathbb{R}^n$ and the vector $\vec{e}$ associated with the identity permutation. In this paper we refer to one half of the square of Euclidean distance between an a vector $\vec{\sigma}$ and $\vec{e}$ as the Euclidean weight of $\vec{\sigma}$ and denote this weight by $\mathrm{wt}_E(\vec{\sigma})$.

Definition (Euclidean Weight)**.** Given $\sigma \in S_n$, the Euclidean weight $\mathrm{wt}_E(\vec{\sigma})$ of $\vec{\sigma}$ is defined as

$$\mathrm{wt}_E(\vec{\sigma}) := \frac{1}{2}(\mathrm{d}_E(\vec{e},\vec{\sigma}))^2.$$

We also include the factor of $\frac{1}{2}$ since all squared distances between $\vec{e}$ and another permutation vector $\vec{\sigma}$ are even. Indeed, given $\vec{\sigma} \in \mathbb{R}^n$, we have $(\mathrm{d}_E(\vec{e},\vec{\sigma}))^2 = 2\langle\vec{e},\vec{e}\rangle - 2\langle\vec{e},\vec{\sigma}\rangle$, where $\langle x,y\rangle$ denotes the standard dot product between x and y so that $\langle\vec{e},\vec{e}\rangle$ and $\langle\vec{e},\vec{\sigma}\rangle$ are integers. The following table compares Kendall tau and Euclidean weights for some permutations and their associated vectors.

TABLE III
KENDALL TAU, EUCLIDEAN WEIGHTS

σ	$\mathrm{wt}_K(\sigma)$	$\mathrm{wt}_E(\vec{\sigma})$
$[0,1,2,3,5,4]$	1	1
$[1,0,2,3,5,4]$	2	2
$[0,2,3,5,4,2]$	4	7
$[1,2,3,4,5,0]$	5	15
$[2,1,0,5,4,3]$	6	8

Because of the left invariance of both metrics, the LP-decoding extension condition can be simplified as follows.

Lemma 3.1. *Let $\lambda, \mu \in S_n$, and G be a subgroup of S_n. Let $g_0 \in G$.*

$$\begin{aligned} &\mathrm{d}_E(\vec{\lambda}, \vec{\mu}) \leq \mathrm{d}_E(\vec{\lambda}, \overrightarrow{g\mu}) \textit{ for all } g \in G \\ \implies &\mathrm{d}_K(\lambda, \mu) \leq \mathrm{d}_K(\lambda, g\mu) \textit{ for all } g \in G \end{aligned} \tag{2}$$

if and only if

$$\begin{aligned} &\mathrm{d}_E(\vec{\lambda}, \overrightarrow{g_0\mu}) \leq \mathrm{d}_E(\vec{\lambda}, \overrightarrow{g\mu}) \textit{ for all } g \in G \\ \implies &\mathrm{d}_K(\lambda, g_0\mu) \leq \mathrm{d}_K(\lambda, g\mu) \textit{ for all } g \in G. \end{aligned}$$

Proof. To justify this lemma, notice that $\mathrm{d}_E(\vec{\lambda}, \overrightarrow{g_0\mu}) = \mathrm{d}_E(\overrightarrow{g_0^{-1}\lambda}, \vec{\mu})$ and likewise $\mathrm{d}_K(\lambda, g_0\mu) = \mathrm{d}_K(g_0^{-1}\lambda, \mu)$. Since λ is taken over all of S_n, the desired result follows. □

For the case when $\mu = e$, we may simplify the LP-decoding extension condition even further using the weights discussed earlier.

Lemma 3.2. *Let $\lambda, \in S_n$, $\mu = e$, and G be a subgroup of S_n. Let $g_0 \in G$.*

$$\begin{aligned} &\mathrm{wt}_E(\overrightarrow{\lambda^{-1}}) \leq \mathrm{wt}_E(\overrightarrow{\lambda^{-1}g}) \;\textit{ for all } g \in G \\ \implies &\mathrm{wt}_K(\lambda^{-1}) \leq \mathrm{wt}_K(\lambda^{-1}g) \;\textit{ for all } g \in G \end{aligned} \tag{3}$$

if and only if

$$\begin{aligned} &\mathrm{d}_E(\vec{\lambda}, \overrightarrow{g_0\mu}) \leq \mathrm{d}_E(\vec{\lambda}, \overrightarrow{g\mu}) \textit{ for all } g \in G \\ \implies &\mathrm{d}_K(\lambda, g_0\mu) \leq \mathrm{d}_K(\lambda, g\mu) \textit{ for all } g \in G. \end{aligned}$$

Proof. The lemma follows immediately from the definitions of $\mathrm{d}_K, \mathrm{wt}_K, \mathrm{d}_E$, and wt_E. □

The above lemma essentially reduces the LP-decoding extension condition to a comparison of weights whenever $\mu = e$. It states that if a received vector $\overrightarrow{\lambda^{-1}}$ having minimal Euclidean weight among vectors of the form $\overrightarrow{\lambda^{-1}}g$ implies that the Kendall tau weight of λ^{-1} is also minimal in the coset $\lambda^{-1}G$, then the LP-decoding extension condition is satisfied.

Through computer programming methods, the authors have confirmed that C_n is Kendall tau LP-decodable for all $1 \leq n \leq 10$. However, it remains an open question to prove that C_n is

Kendall tau LP-decodable for all $n \in \mathbb{N}$. The following is an example of a subgroup that does not satisfy the LP-decoding extension condition.

Example 3.3 (D_{12}). *The dihedral group D_{12} does not satisfy statement (3). Consider $\lambda^{-1} := [1,0,4,3,2,5]$. $\lambda^{-1}D_{12} = \{\lambda^{-1}g \,:\, g \in D_{12}\}$. It is easily verified that $\min_{\lambda^{-1}g \,\in\, \lambda^{-1}D_{12}} \mathrm{wt}_E(\overrightarrow{\lambda^{-1}g}) = 5 = \mathrm{wt}_E(\overrightarrow{\lambda^{-1}})$. However, $g_0 := [1,0,5,4,3,2] \in D_{12}$ which implies that $\lambda^{-1}g_0 = [0,1,5,2,3,4] \in \lambda^{-1}D_{12}$ and $\mathrm{wt}_K(\lambda^{-1}g_0) = 3 < 4 = \mathrm{wt}_K(\lambda^{-1})$. Therefore $\mathrm{wt}_E(\overrightarrow{\lambda^{-1}}) \leq \mathrm{wt}_E(\overrightarrow{\lambda^{-1}g})$ for all $g \in G$, but there exists $g \in G$ such that $\mathrm{wt}_K(\lambda^{-1}) > \mathrm{wt}_K(\lambda^{-1}g)$.*

C. Minimum Weight

Determining minimal weights is often an important question in coding theory, since this provides insight into the minimum distance between codewords. A non-identity element is said to have minimum Kendall tau (Euclidean) weight if there exist no elements with a smaller Kendall tau (Euclidean) weight value (other than the identity element.) In the cases of the Kendall tau and Euclidean distances, the minimum distance is equivalent to the respective minimum weight since in both instances we can rewrite the distance between any two elements as the weight of some element.

We begin this section with a discussion on $C_n \subset S_n$, the cyclic subgroup of order n generated by $\sigma^{[n-1]} := [1,2,3\ldots,n-1,0]$. Before determining the elements of minimal Kendall tau and Euclidean weight in C_n and explicitly calculating their values, we first prove a linear relationship between the two weights.

Proposition 3.4. *Let $\sigma^{[i]} := (\sigma^{[n-1]})^{n-i}$, the unique permutation of C_n with 0 in the (i)th position. Then $\mathrm{wt}_K(\sigma^{[i]}) = (i)(n-i)$.*

Proof. For any $\sigma^{[i]} \in C_n$, we have

$$\sigma^{[i]} = [\underbrace{n-i, n-i+1, \ldots, n-1}_{i}, \underbrace{0, 1, \ldots, n-i-1}_{n-i}].$$

Note that $\sigma^{[i]}$ splits into two sub-sequences of length i and $n-i$ respectively. For any element j of the left sub-seqence, $j > k$ for all k in the right sub-sequence. Thus the Kendall tau weight of $\sigma^{[i]}$

is determined by the product of the length of each sub-sequence, *i.e.*, $\mathrm{wt}_K(\sigma^{[i]}) = (i)(n-i)$. $\square$

There is a linear relationship between $\mathrm{wt}_K(\sigma)$ and $\mathrm{wt}_E(\vec{\sigma})$ for all $\sigma \in C_n$. The following proposition is directly obtained by routine calculation.

Proposition 3.5. *Let* $\sigma^{[i]} := (\sigma^{[n-1]})^{n-i}$, *the unique permutation of* C_n *with* 0 *in the* i*th position. Then* $\mathrm{wt}_E(\overrightarrow{\sigma^{[i]}}) = \frac{n}{2}(i)(n-i)$ *i.e.,* $\mathrm{wt}_E(\overrightarrow{\sigma^{[i]}}) = \frac{n}{2}\mathrm{wt}_K(\sigma^{[i]})$.

The subsequent table, showing the elements of C_5 and their respective Kendall tau and Euclidean weights, depicts a concrete example of the linear relationship between the two weights for C_n.

TABLE IV
KENDALL TAU, EUCLIDEAN WEIGHTS OF C_5

$\sigma \in C_5$	$\mathrm{wt}_K(\sigma)$	$\mathrm{wt}_E(\vec{\sigma})$
$\sigma^{[0]} = [0,1,2,3,4]$	0	0
$\sigma^{[1]} = [4,0,1,2,3]$	4	10
$\sigma^{[2]} = [3,4,0,1,2]$	6	15
$\sigma^{[3]} = [2,3,4,0,1]$	6	15
$\sigma^{[4]} = [1,2,3,4,0]$	4	10

Remark 3.6. Based on proposition 3.4, it is clear that the the elements of minimal Kendall tau weight in C_n are $\sigma^{[1]}$ and $\sigma^{[n-1]}$, with $\mathrm{wt}_K(\sigma^{[1]}) = \mathrm{wt}_K(\sigma^{[n-1]}) = n-1$.

Since there is a linear relationship between the Kendall tau weight and the Euclidean weight for elements of C_n and their associated vectors, it is a simple matter to determine the elements of minimal Euclidean weight and their corresponding values in the C_n case.

Remark 3.7. The elements of minimal Euclidean weight for all associated vectors of permutations of C_n are $\overrightarrow{\sigma^{[1]}}$ and $\overrightarrow{\sigma^{[n-1]}}$, with $\mathrm{wt}_E(\overrightarrow{\sigma^{[1]}}) = \mathrm{wt}_E(\overrightarrow{\sigma^{[n-1]}}) = \frac{n}{2}(n-1)$.

In the case of S_n, calculating minimal weight elements is a trivial matter, since for any natural number n, there exists a permutation of S_n with a Kendall tau or Euclidean weight of 1. Specifically, any adjacent transposition $(i, i+1)$ in S_n will have both Kendall tau and Euclidean weight of 1.

D. Weight Enumerator Polynomials

Continuing our exposition on the relationship between Kendall tau and Euclidean weights, we consider now their respective weight enumerator polynomials [39], also known as the generating functions [53]. Weight enumerator polynomials are useful for analyzing the performance of a code such as the probability of error detection.

The weight enumerator polynomial is defined as follows.

Definition (Weight Enumerator Polynomials)**.** Let $G \subset S_n$. The weight enumerator polynomial W_K (resp. W_E) of G for wt_K (resp. wt_E) is:

$$W_K(G;t) := \sum_{\sigma \in G} t^{\mathrm{wt}_K(\sigma)} \text{ (resp. } W_E(G;t) := \sum_{\sigma \in G} t^{\mathrm{wt}_E(\vec{\sigma})} \text{).}$$

We shall begin our discussion of weight enumerator polynomials with the cyclic subgroup case.

Example 3.8. *The Kendall tau weight enumerator polynomials of C_n for $n = 1, \ldots, 7$ are as follows.*

$$\begin{aligned}
W_K(C_1;t) &= 1. \\
W_K(C_2;t) &= 1 \quad +t. \\
W_K(C_3;t) &= 1 \quad +2t^2. \\
W_K(C_4;t) &= 1 \quad +2t^3 \quad +t^4. \\
W_K(C_5;t) &= 1 \quad +2t^4 \quad +2t^6. \\
W_K(C_6;t) &= 1 \quad +2t^5 \quad +2t^8 \quad +t^9. \\
W_K(C_7;t) &= 1 \quad +2t^6 \quad +2t^{10} \quad +2t^{12}.
\end{aligned}$$

Proposition 3.9. $W_K(C_n;t) = \sum_{i=1}^{n} t^{(i-1)(n-i+1)}$.

Proof. The desired result follows immediately from Proposition 3.4. $\square$

Example 3.10. *The Euclidean weight enumerator polynomials of C_n for $n = 1, \ldots, 7$ are as follows.*

$$\begin{aligned}
W_E(C_1;t) &= 1. \\
W_E(C_2;t) &= 1 \quad +t. \\
W_E(C_3;t) &= 1 \quad +2t^3. \\
W_E(C_4;t) &= 1 \quad +2t^6 \quad +t^8. \\
W_E(C_5;t) &= 1 \quad +2t^{10} \quad +2t^{15}. \\
W_E(C_6;t) &= 1 \quad +2t^{15} \quad +2t^{24} \quad +t^{27}. \\
W_E(C_7;t) &= 1 \quad +2t^{21} \quad +2t^{35} \quad +2t^{42}.
\end{aligned}$$

Proposition 3.11. *The Euclidean weight enumerator polynomials for C_n is characterized by:* $W_E(C_n;t) = \sum_{i=1}^{n} t^{\frac{n}{2}(i-1)(n-i+1)}$.

Proof. The desired result follows immediately from Proposition 3.5. □

Proposition 3.12. $W_E(C_n;t) = W_K(C_n;t^{\frac{n}{2}})$.

Proof. The desired result follows from the previous two propositions. □

Example 3.13. *The Kendall tau weight enumerator polynomials of S_n for $n = 1, \ldots, 7$ are as follows.*

$$
\begin{array}{llllllll}
W_K(S_1;t) & =1 \\
\\
W_K(S_2;t) & =1 & +t \\
\\
W_K(S_3;t) & =1 & +2t & +2t^2 & +t^3 \\
\\
W_K(S_4;t) & =1 & +3t & +5t^2 & +6t^3 & +5t^4 & +3t^5 & +t^6 \\
\\
W_K(S_5;t) & =1 & +4t & +9t^2 & +15t^3 & +20t^4 & +22t^5 & +20t^6 & +15t^7 \\
 & & +9t^8 & +4t^9 & +t^{10} \\
\\
W_K(S_6;t) & =1 & +5t & +14t^2 & +29t^3 & +49t^4 & +71t^5 & +90t^6 & +101t^7 \\
 & & +101t^8 & +90t^9 & +71t^{10} & +49t^{11} & +29t^{12} & +14t^{13} & +5t^{14} \\
 & & +t^{15} \\
\\
W_K(S_7;t) & =1 & +6t & +20t^2 & +49t^3 & +98t^4 & +169t^5 & +259t^6 & +359t^7 \\
 & & +455t^8 & +531t^9 & +573t^{10} & +573t^{11} & +531q^{12} & +455t^{13} & +359t^{14} \\
 & & +259t^{15} & +169t^{16} & +98t^{17} & +49t^{18} & +20t^{19} & +6t^{20} & +t^{21}
\end{array}
$$

The following general formula of $W_K(S_n;t)$ for $n \geq 2$ is well-known [6].

$$W_K(S_n;t) = (1+t)(1+t+t^2)\cdots(1+t+\cdots+t^{n-1}).$$

This formula is a special case of Weyl's character formula for Lie theory [32]. It is a relatively simple matter to see why it is true. From the above example the formula is easily verifiable for $n = 2$. Notice that any permutation of S_3 can be obtained from a permutation of S_2 by simply inserting the number 2 into some position. For example, the permutation $[0, 1, 2]$ is simply the permutation $[0, 1]$, with 2 inserted into the third position. Similarly $[0, 2, 1]$ is simply the permutation $[0, 1]$ with 2 inserted in the second position and $[2, 0, 1]$ is $[0, 1]$ with 2 inserted into the first position. In general, inserting $n-1$ into the $(n-i)$th position for $i = 0, \ldots, n-1$ in each of the permutation σ of S_{n-1} corresponds to a new permutation of S_n with weight $\mathrm{wt}_K(\sigma) + i$. Thus an inductive argument yields the desired formula.

It is clear that the powers of $W_K(S_n; t)$ are consecutive, running from 1 through $\frac{1}{2}(n^2 - n)$. That is, for all values k between 0 and $\frac{1}{2}(n^2 - n)$, there exists a $\sigma \in S_n$ such that $\mathrm{wt}_K(\sigma) = k$. It remains an open question to find a concise formula for $W_E(S_n; t)$ similar to the formula above.

The following example shows the Euclidean weight enumerator polynomials corresponding to the example above.

Example 3.14. *The Euclidean weight enumerator polynomials of S_n for $n = 1, \ldots, 7$ are as follows.*

$$W_E(S_1;t) = 1$$

$$W_E(S_2;t) = 1 \quad +t$$

$$W_E(S_3;t) = 1 \quad +2t \quad +2t^3 \quad +t^4$$

$$\begin{aligned} W_E(S_4;t) = 1 \quad & +3t \quad +t^2 \quad +4t^3 \quad +2t^4 \quad +2t^5 \quad +2t^6 \quad +4t^7 \\ & +t^8 \quad +3t^9 \quad +t^{10} \end{aligned}$$

$$\begin{aligned} W_E(S_5;t) = 1 \quad & +4t \quad +3t^2 \quad +6t^3 \quad +7t^4 \quad +6t^5 \quad +4t^6 \quad +10t^7 \\ & +6t^8 \quad +10t^9 \quad +6t^{10} \quad +10t^{11} \quad +6t^{12} \quad +10t^{14} \quad +4t^{14} \\ & +6t^{15} \quad +7t^{16} \quad +6t^{17} \quad +3t^{18} \quad +4t^{19} \quad +t^{20} \end{aligned}$$

$$\begin{aligned} W_E(S_6;t) = 1 \quad & +5t \quad +6t^2 \quad +9t^3 \quad +16t^4 \quad +12t^5 \quad +14t^6 \quad +24t^7 \\ & +20t^8 \quad +21t^9 \quad +23t^{10} \quad +28t^{11} \quad +24t^{12} \quad +34t^{13} \quad +20t^{14} \\ & +32t^{15} \quad +42t^{16} \quad +29t^{17} \quad +29t^{18} \quad +42t^{19} \quad +32t^{20} \quad +20t^{21} \\ & +34t^{22} \quad +24t^{23} \quad +28t^{24} \quad +23t^{25} \quad +21t^{26} \quad +20t^{27} \quad +24t^{28} \\ & +14t^{29} \quad +12t^{30} \quad +16t^{31} \quad +9t^{32} \quad +6t^{33} \quad +5t^{34} \quad +t^{35} \end{aligned}$$

$$\begin{aligned} W_E(S_7;t) = 1 \quad & +6t \quad +10t^2 \quad +14t^3 \quad +29t^4 \quad +26t^5 \quad +35t^6 \quad +46t^7 \\ & +55t^8 \quad +54t^9 \quad +74t^10 \quad +70t^{11} \quad +84t^{12} \quad +90t^{13} \quad +78t^{14} \\ & +90t^{15} \quad +129t^{16} \quad +106t^{17} \quad +123t^{18} \quad +134t^{19} \quad +147t^{20} \quad +98t^{21} \\ & +168t^{22} \quad +130t^{23} \quad +175t^{24} \quad +144t^{25} \quad +168t^{26} \quad +144t^{27} \quad +184t^{28} \\ & +144t^{29} \quad +168t^{30} \quad +144t^{31} \quad +175t^{32} \quad +130t^{33} \quad +168t^{34} \quad +98t^{35} \\ & +147t^{36} \quad +134t^{37} \quad +123t^{38} \quad +106t^{39} \quad +129t^{40} \quad +90t^{41} \quad +78t^{42} \\ & +90t^{43} \quad +84t^{44} \quad +70t^{45} \quad +74t^{46} \quad +54t^{47} \quad +55t^{48} \quad +46t^{49} \\ & +35t^{50} \quad +26t^{51} \quad +29t^{52} \quad +14t^{53} \quad +10t^{54} \quad +6t^{55} \quad +t^{56} \end{aligned}$$

From Example 3.14 it can be observed that all the Euclidean weight values appear to be consecutive for $n \neq 3$. In other words, for all values k between 0 and $\frac{1}{6}(n^3 - n)$, where $\mathrm{wt}_K(e) = 0$ and $\mathrm{wt}_K(\omega_0) = \frac{1}{2}(n^2 - n)$, there exists a $\sigma \in S_n$ such that $\mathrm{wt}_E(\vec{\sigma}) = k$. To prove that this observation is true, we first introduce a small theorem about the maximal Kendall tau weight element, which we will show corresponds exactly with the maximal Euclidean weight element.

It is well-known that $\omega_0 := [n-1, n-2, \ldots, 0]$ is the unique element of S_n having maximal Kendall tau weight, $\mathrm{wt}_K(\omega_0) = \frac{n(n-1)}{2}$; it is called the longest element. It is also a known property that for all σ in S_n, $\mathrm{wt}_K(\omega_0\sigma) = \mathrm{wt}_K(\omega_0) - \mathrm{wt}_K(\sigma)$ [43]. We proceed to show that a similar property also holds for the associated vector of ω_0 in terms of the wt_E, making $\vec{\omega_0}$ the element of maximal Euclidean weight.

Theorem 3.1. *Let $\omega_0 \in S_n$ be the longest element and let $\sigma \in S_n$. Then $\mathrm{wt}_E(\overrightarrow{\omega_0\sigma}) = \mathrm{wt}_E(\vec{\omega_0}) - \mathrm{wt}_E(\vec{\sigma})$.*

Proof. Note first that $2\mathrm{wt}_E(\vec{\sigma}) = \langle \vec{e}, \vec{e} \rangle - 2\langle \vec{e}, \vec{\sigma} \rangle + \langle \vec{\sigma}, \vec{\sigma} \rangle$. Furthermore, $\langle \vec{e}, \vec{e} \rangle = \langle \vec{\sigma}, \vec{\sigma} \rangle$. Thus to prove that $\mathrm{wt}_E(\overrightarrow{\omega_0\sigma}) = \mathrm{wt}_E(\vec{\omega_0}) - \mathrm{wt}_E(\vec{\sigma})$, it suffices to show that $\langle \vec{e}, \vec{e} \rangle + \langle \vec{e}, \vec{\omega_0} \rangle = \langle \vec{e}, \vec{\sigma} \rangle + \langle \vec{e}, \overrightarrow{\omega_0\sigma} \rangle$. On the left hand side we have $\langle \vec{e}, \vec{e} \rangle + \langle \vec{e}, \vec{\omega_0} \rangle = \langle \vec{e}, (n-1, n-1, \ldots, n-1) \rangle$. On the right hand side we have $\langle \vec{\sigma}, \vec{e} \rangle + \langle \vec{\sigma}, \vec{\omega_0} \rangle = \langle \vec{\sigma}, (n-1, n-1, \ldots, n-1) \rangle$. Equality holds since $\vec{\sigma}$ is simply a permutation of $\vec{e}$. $\square$

Corollary 3.15. *Let $\omega_0 \in S_n$ be the longest element. Then ω_0 is the unique element such that $\mathrm{wt}_E(\vec{\omega_0}) > \mathrm{wt}_E(\vec{\sigma})$ for all $\omega_0 \neq \sigma \in S_n$.*

Proof. By Theorem 3.1, for any $\sigma \in S_n$, we have $\mathrm{wt}_E(\vec{\sigma}) = \mathrm{wt}_E(\vec{\omega_0}) - \mathrm{wt}_E(\overrightarrow{\omega_0\sigma}) \leq \mathrm{wt}_E(\vec{\omega_0})$, since $\mathrm{wt}_E(\overrightarrow{\omega_0\sigma}) \geq 0$. Equality holds only if $\omega_0\sigma = e$, which is true only when $\sigma = \omega_0$. Thus the statement holds. $\square$

It is a simple matter to calculate the Euclidean weight of $\vec{\omega_0}$, which will be useful when we prove that the Euclidean Weight Enumerator Polynomial of S_n is consecutive in powers.

Proposition 3.16. $\mathrm{wt}_E(\vec{\omega_0}) = \frac{1}{6}(n^3 - n) = \frac{n+1}{3}\mathrm{wt}_K(\omega_0)$.

Proof. For any even positive integer n,

$$\mathrm{wt}_E(\vec{\omega_0}) \;=\; \sum_{i=0}^{\frac{n}{2}-1}(1+2i)^2 = \frac{1}{6}(n^3 - n).$$

For any odd positive integer n,

$$\mathrm{wt}_E(\vec{\omega_0}) \;=\; \sum_{i=0}^{\frac{n-1}{2}}(2i)^2 = \frac{1}{6}(n^3 - n).$$

$\square$

Proposition 3.17. *For $n \geq 1$ and $n \neq 3$, the powers of t in $W_E(S_n;t)$ are consecutive.*

Proof. Note that t in $W_E(S_n;t)$ are consecutive for $n = 1$ and $n = 2$. We proceed by induction on n. For the base case of $n = 4$, note that $W_E(S_4;t) = 1 + 3t + t^2 + 4t^3 + 2t^4 + 2t^5 + 2t^6 + 4t^7 + t^8 + 3t^9 + t^{10}$, where we can easily observe that the powers of t are consecutive. Suppose now that the powers of t in $W_E(S_n;t)$ are consecutive. We will show that this implies that the powers of t are consecutive for $W_E(S_{n+1};t)$.

Let us begin by showing that the first $\frac{1}{6}(n^3 - n) + 1$ powers of t are consecutive. Since $S_n \subset S_{n+1}$, we conclude that each of the powers 0 through $\max\{\mathrm{wt}_E(\vec{\sigma}) \,:\, \sigma \in S_n\} = \frac{1}{6}(n^3 - n)$ are contained in $W_E(S_{n+1};t)$. Hence the first $\frac{1}{6}(n^3 - n) + 1$ powers are consecutive. Next, let us show that the last $\frac{1}{6}(n^3 - n) + 1$ powers of t are consecutive.

Let $j = 0, \ldots, \frac{1}{6}(n^3 - n)$ Then for each j there exists a $\sigma^{(j)} \in S_{n+1}$ such that $\mathrm{wt}_E(\overrightarrow{\sigma^{(j)}}) = j$. By Theorem 3.1, for each $\sigma^{(j)} \in S_{n+1}$ there exists a $\sigma^{(j^*)} \in S_{n+1}$ such that $\mathrm{wt}_E(\overrightarrow{\sigma^{(j^*)}}) = \mathrm{wt}_E(\vec{\omega_0}) - \mathrm{wt}_E(\vec{\sigma}) = \frac{1}{6}((n+1)^3 - (n+1)) - j$. Since $j = 0, \ldots, \frac{1}{6}(n^3 - n)$, the last $\frac{1}{6}(n^3 - n) + 1$ powers of t are in fact consecutive. Thus to show that the powers of t in $W_E(S_{n+1};t)$ are consecutive, it suffices to show that $2 \cdot \frac{1}{6}(n^3 - n) \geq \frac{1}{6}((n+1)^3 - (n+1))$.

$$2 \cdot \frac{1}{6}(n^3 - n) \geq \frac{1}{6}((n+1)^3 - (n+1))$$

$$\Longleftrightarrow\; n^3 - 3n^2 - 4n \geq 0 \;\Longleftrightarrow\; n(n-4)(n+1) \geq 0$$

The above inequality is satisfied for all $n \geq 4$. Hence by induction, the powers of $W_E(S_n; t)$ are consecutive for $n \geq 4$. $\square$

E. Parabolic Subgroups and Reflection Subgroups

In this section we consider Parabolic subgroups and more generally, Reflection subgroups. These subgroups can have a cardinality that is significantly larger than that of C_n discussed previously, and the minimum Kendall tau and Euclidean distances can be arbitrarily large (at the expense of code size). As the final main contribution of this chapter we will prove that these subgroups are Kendall tau LP-Decodable and realizable as the vertex set of a doubly stochastic polytope for the appropriate choice of consolidated constraint. We also provide some examples of such subgroups and calculate their size and minimum distances.

It is known that S_n is a reflection group whose fundamental set of generators is $\{s_0, \ldots, s_{n-2}\}$, where s_i denotes the adjacent transposition $(i, i+1)$. A parabolic subgroup P of S_n is simply any subgroup generated by a subset of the fundamental set of generators $\{s_0, \ldots, s_{n-2}\}$ [47]. We would like to show that such a parabolic subgroup is Kendall tau LP-decodable. We first show that P is LP-decodable. Notice that for any σ in P, the associated permutation matrix $X_{i,j}^{\sigma}$ will have blocks of permutation matrices along the main diagonal. For example, consider the parabolic subgroup $P \subset S_6$ such that P is generated by $s_0, s_3,$ and s_4. Then any permutation of P will have an associated matrix of the following form:

$$\begin{pmatrix} X_{0,0} & X_{0,1} & 0 & 0 & 0 & 0 \\ X_{1,0} & X_{1,1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & X_{3,3} & X_{3,4} & X_{3,5} \\ 0 & 0 & 0 & X_{4,3} & X_{4,4} & X_{4,5} \\ 0 & 0 & 0 & X_{5,3} & X_{5,4} & X_{5,5} \end{pmatrix}.$$

Here the matrices $\begin{pmatrix} X_{0,0} & X_{0,1} \\ X_{1,0} & X_{1,1} \end{pmatrix}$ and $\begin{pmatrix} X_{3,3} & X_{3,4} & X_{3,5} \\ X_{4,3} & X_{4,4} & X_{4,5} \\ X_{5,3} & X_{5,4} & X_{5,5} \end{pmatrix}$

have the form of permutation matrices for S_2 and S_3 respectively. In other words, for any parabolic subgroup $P \subset S_n$, the set of all permutations in P is simply the set of all permutations of S_n with the added constraint that $X_{i,j} = X_{j,i} = 0$ for all (i,j) in a subset of $\{0,\ldots,n-1\} \times \{0,\ldots,n-1\}$. We can explicitly construct this subset by first characterizing the entries of matrices in P that potentially have non-zero entries.

Definition (Block Index Set)**.** Given a parabolic subgroup P of S_n, the **block index set** $C_P := \{(i,\sigma_i) \ : \ 0 \leq i \leq n-1, \sigma \in P\}$ is the set of all ordered pairs (i,j) such that there exists a permutation matrix X of P with $X_{i,j} \neq 0$.

As an example, let us calculate C_P corresponding to the parabolic subgroup P of S_6 generated by s_0, s_3, and s_4. For any permutation $\sigma \in P$, either $\sigma_0 = 0$ or $\sigma_0 = 1$ and similarly $\sigma_1 = 0$ or $\sigma_1 = 1$. Hence the ordered pairs $(0,0), (0,1), (1,0)$, and $(1,1)$ are in C_P, but no other ordered pairs beginning with 0 or 1 are in C_P. For all $\sigma \in P$, we have $\sigma_2 = 2$, so that $(2,2)$ is the only ordered pair C_P beginning with 2. Continuing in this manner, the only other entries that are included in C_P are $(3,3), (3,4), (3,5), (4,3), (4,4), (4,5), (5,3), (5,4)$, and $(5,5)$. For any $(i,j) \notin C_P$, we have $X_{i,j} = 0$.

Theorem 3.2. *Let P be a parabolic subgroup of S_n generated by a subset $\{s_{k_1},\ldots,s_{k_m}\}$ of $\{s_0,\ldots,s_{n-2}\}$. Then P is LP-decodable, i.e., there exists a doubly stochastic constraint $\mathcal{L}$ such that $P = \mathrm{Ver}(D[\mathcal{L}]) \cap S_n$, where $\mathcal{L}$ consists of the following linear constraints:*

(1) For $j = 0,\ldots,n-1$, $\sum_{i=0}^{n-1} X_{i,j} = 1$.
(2) For $i = 0,\ldots,n-1$, $\sum_{j=0}^{n-1} X_{i,j} = 1$.
(3) For all $0 \leq i,j \leq n-1$, $X_{i,j} \geq 0$.

(4) For all $(i,j) \in \{0,\ldots,n-1\} \times \{0,\ldots,n-1\}$ *such that* $(i,j) \notin C_P,\ \ X_{i,j} = 0.$

Proof. Notice that the first three constraints of $\mathcal{L}$ are exactly those in the definition of a doubly stochastic matrix. By the Birkhoff von Neumann theorem, we know that the Birkhoff polytope B_n consisting of stochastic matrices is the convex polytope satisfying the first three constraints, and $\mathrm{Ver}(B_n) = S_n$. Since constraint (4) consists strictly of linear equations, including constraint (4) will remove permutations matrices such that $X_{i,j} = 1$ or $X_{j,i} = 1$ from the vertex set $\mathrm{Ver}(D(\mathcal{L}))$ but will have no effect on other vertices. Thus $\mathrm{Ver}(D[\mathcal{L}]) \cap S_n$ consists of all permutation matrices such that $X_{i,j} \neq 1$ and $X_{j,i} \neq 1$, for all $(i,j) \in \{0,\ldots,n-1\} \times \{0,\ldots,n-1\}$ but $(i,j) \notin C_P$. From the previous discussion, the linear constraint (4) is exactly the constraint that retains permutation matrices of P while excluding permutation matrices outside of P. This implies that $\mathrm{Ver}(D(\mathcal{L})) \cap S_n = P$. □

A stronger result holds for parabolic subgroups P of S_n, namely that there exists a doubly stochastic constraint $\mathcal{L}$ such that $P = \mathrm{Ver}(D[\mathcal{L}])$, rather than $P = \mathrm{Ver}(D[\mathcal{L}]) \cap S_n$ This result is a consequence of Theorem A.2. This is significant, since in the decoding algorithm described above, if the vertex set of the linear constraint $\mathcal{L}$ is a subset of S_n, then the solution to the linear programming problem will be a permutation.

Theorem 3.3. *Let* P *be a parabolic subgroup of* S_n *generated by a subset* $\{s_{k_1}, \ldots s_{k_m}\}$ *of* $\{s_0, \ldots, s_{n-2}\}$. *Then* $P = \mathrm{Ver}(D[\mathcal{M} \boxplus \mathcal{H}])$ *where* $\mathcal{H}$ *consists of the following linear constraints:*

(1) For $j = 0, \ldots, n-1,\ \sum_{i=0}^{n-1} X_{i,j} = 1,$

(2) For $i = 0, \ldots, n-1,\ \sum_{j=0}^{n-1} X_{i,j} = 1,$

(3) For all $0 \leq i,j \leq n-1,\ X_{i,j} \geq 0,$

and $\mathcal{M}$ *consists of the following linear constraints:*

(4) For all $(i,j) \in \{0,\ldots,n-1\} \times \{0,\ldots,n-1\}$ *such that* $(i,j) \notin C_P, \quad X_{i,j} = 0.$

Proof. Note first that for all $0 \leq r_0, r_1 \leq R-1$, then $\mathcal{M}^{[r_0,r_1]}$ is a quasi-homogeneous constraint since each $M^{[r_0,r_1]}$ is a linear constraint with a constant term of 0. By definition, $\mathcal{H}$ is a doubly stochastic matrix for an $n \times n$ matrix. Each $\mathcal{M}^{[r_0,r_1]}$ is also trivially a compact constraint as it is a constraint on a 1×1 matrix. By the Birkhoff-von Neumann theorem, $\mathcal{H}$ is also compact. Therefore by Theorem A.2, $\mathrm{Ver}(D[\mathcal{M} \boxplus \mathcal{H}]) = P$. $\square$

Thus far we have only considered parabolic subgroups generated by a subset of the fundamental set of reflections: $\{s_0, \ldots, s_{n-2}\}$. It is known that any reflection subgroup W of S_n can be generated by a set of transpositions of the form (i, j), and furthermore that any such subgroup is a conjugate of some parabolic subgroup $P \subset S_n$. That is, any reflection subgroup of S_n is of the form $\phi P \phi^{-1}$, where ϕ is an element of S_n. Furthermore, since conjugation is an automorphism, there is a one-to-one correspondence between the block index set C_P of P and the block index set $C_{\phi P \phi^{-1}}$ of $\phi P \phi^{-1}$.

As an example, consider the parabolic subgroup $P \subset S_6$ generated by s_0, s_3, and s_4, which was previously discussed. Let $\phi := [4, 3, 2, 5, 0, 1]$. Then $\phi^{-1} = [4, 5, 2, 1, 0, 3]$. Any element $\sigma \in P$ has an associated permutation matrix of the form

$$X^\sigma = \begin{pmatrix} X_1 & X_2 & 0 & 0 & 0 & 0 \\ X_3 & X_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & X_5 & X_6 & X_7 \\ 0 & 0 & 0 & X_8 & X_9 & X_{10} \\ 0 & 0 & 0 & X_{11} & X_{12} & X_{13} \end{pmatrix}$$

where $X_i \in \{0, 1\}$ for all i.

We also have $X^{\phi} = \begin{pmatrix} 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \\ 0&0&1&0&0&0 \\ 0&1&0&0&0&0 \\ 1&0&0&0&0&0 \\ 0&0&0&1&0&0 \end{pmatrix}$ and $X^{\phi^{-1}} = \begin{pmatrix} 0&0&0&0&1&0 \\ 0&0&0&1&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&0&0&1 \\ 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \end{pmatrix}$

so that $X^{\phi}X^{\sigma}X^{\phi^{-1}} = \begin{pmatrix} X_9 & X_{10} & 0 & 0 & 0 & X_8 \\ X_{12} & X_{13} & 0 & 0 & 0 & X_{11} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & X_4 & X_3 & 0 \\ 0 & 0 & 0 & X_2 & X_1 & 0 \\ X_6 & X_7 & 0 & 0 & 0 & X_5 \end{pmatrix}$.

It follows from the previous theorem that for any reflection subgroup W of S_n, there exists a doubly stochastic constraint having W as a vertex set.

Corollary 3.18. *Let $\phi \in S_n$, and $P \subset S_n$ a parabolic subgroup generated by a subset of $\{s_0, \ldots, s_{n-2}\}$. Let $\mathcal{H}$ consist of the following linear constraints:*

(1) For $j = 0, \ldots, n-1$, $\sum_{i=0}^{n-1} X_{i,j} = 1$,

(2) For $i = 0, \ldots, n-1$, $\sum_{j=0}^{n-1} X_{i,j} = 1$,

(3) For all $0 \leq i, j \leq n-1$, $X_{i,j} \geq 0$,

and let $\mathcal{M}'$ consist of the following linear constraints:

(4) For all $(i,j) \in \{0, \ldots, n-1\} \times \{0, \ldots, n-1\}$ such that $(i,j) \notin C_{\phi P \phi^{-1}}$, $\quad X_{i,j} = 0$.

Then $\phi P \phi^{-1} = \mathrm{Ver}(D[\mathcal{M}' \boxplus \mathcal{H}])$.

Proof. Since $P \cong \phi P \phi^{-1}$, we have that each ordered pair (i,j) in the block index set C_P

corresponds exactly to an ordered pair (i', j') in the block index set $C_{\phi P \phi^{-1}}$. Moreover, $\phi P \phi^{-1}$ consists of all the permutation matrices satisfying the constraints of $\mathcal{M}'$. The remainder of the proof is the same as that of Theorem 3.3. $\square$

We have proven that any reflection subgroup W of S_n is LP-decodable. In fact, we have seen that there exists a doubly stochastic constraint $\mathcal{L}$ such that $\mathrm{Ver}(D[\mathcal{L}]) = W$. Thus to show that W is Kendall tau LP-decodable, it remains only to show that W satisfies the LP-decoding extension condition. With this goal in mind, we next prove a relation between $\mathrm{wt}_K(\sigma)$ and $\mathrm{wt}_E(\vec{\sigma})$ that holds for all $\sigma \in S_n$. To prove this relation we recall a partial ordering known as the weak Bruhat ordering.

Definition (Weak (right) Bruhat Ordering)**.** Let $\sigma, \tau \in S_n$. Define $\sigma^{(0)} := \sigma$ and $\sigma^{(\mathrm{wt}_K(\tau) - \mathrm{wt}_K(\sigma))} := \tau$. The weak (right) Bruhat ordering on S_n is a partial ordering $\leq$ where $\sigma < \tau$ if and only if $\mathrm{wt}_K(\sigma) < \mathrm{wt}_K(\tau)$, and for all $1 \leq r \leq \mathrm{wt}_K(\tau) - \mathrm{wt}_K(\sigma)$, there exists $\sigma^{(r)} \in S_n$, and $1 \leq i_r < n$ such that $(\sigma^{(r-1)})^{-1}(\sigma^{(r)}) = (i_r, i_r + 1)$ and $\mathrm{wt}_K(\sigma^{(r)}) = \mathrm{wt}_K(\sigma) + r$. Here $(i_r, i_r + 1)$ denotes the adjacent transposition switching the (i_r)th and $(i_r + 1)$th position. We say $\sigma \leq \tau$ if either $\sigma < \tau$ or $\sigma = \tau$.

Intuitively, the above definition states that a permutation σ is strictly less than a permutation τ if σ has a smaller Kendall tau weight and τ can be obtained by applying a series of adjacent transpositions to σ with the Kendall tau weight increasing by 1 with each adjacent transposition. The following figure illustrates the weak (right) Bruhat ordering for S_4. In the diagram, two permutations are comparable under the weak Bruhat ordering if there is a strictly ascending or strictly descending connected path between the two permutations. For example, $[0,1,2,3] < [1,0,2,3] < [1,2,0,3] < [1,2,3,0] < [2,1,3,0] < [2,3,1,0] < [3,2,1,0]$ forms an ascending chain under the weak Bruhat ordering. However, as another example, neither of the following statements is true: $[1,0,2,3] \leq [0,3,1,2]$ or $[0,3,1,2] \leq [1,0,2,3]$. Notice that both $\omega_0 = [3,2,1,0]$ and $e = [0,1,2,3]$ are comparable to all other permutations of S_n.

Lemma 3.19. *If* $\sigma < \tau$ *in the weak Bruhat ordering, then* $\mathrm{wt}_E(\vec{\sigma}) < \mathrm{wt}_E(\vec{\tau})$.

Proof. We proceed by induction. Suppose $\sigma < \tau$. To prove the base case let $\sigma^{-1}\tau = (i, i+1)$

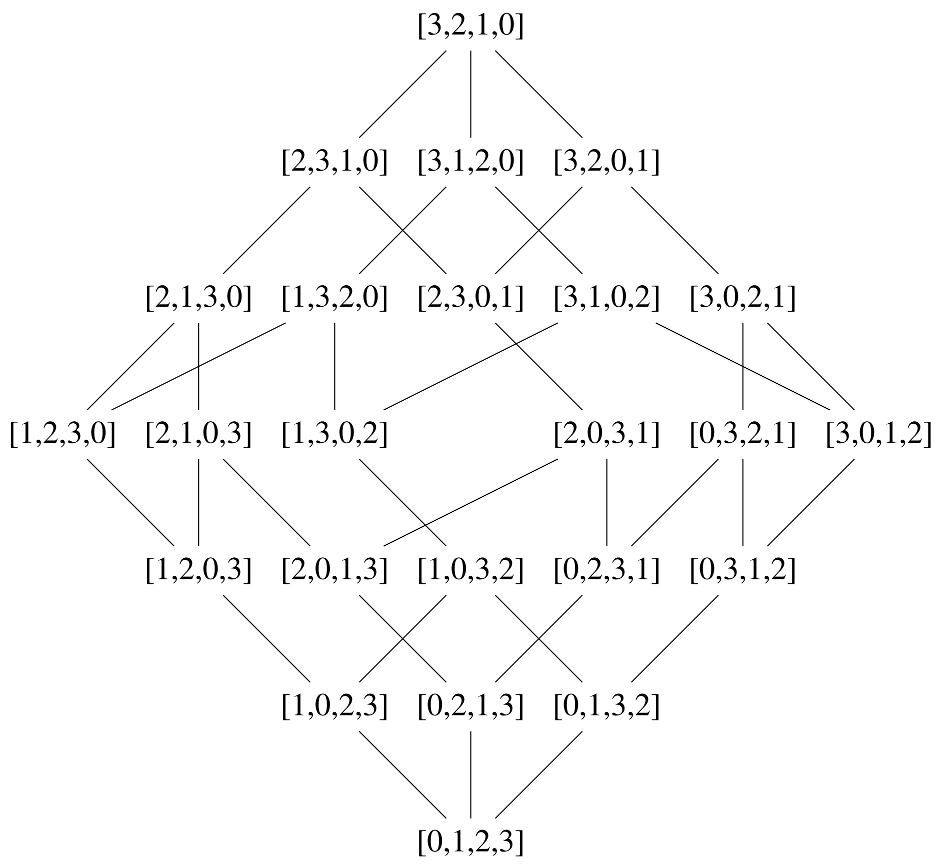


Fig. 1. Weak Bruhat ordering for S_4

and $\mathrm{wt}_K(\tau) = \mathrm{wt}_K(\sigma) + 1$. Then $\tau = \sigma s_i$ where $s_i \in S_n$ is the transposition $(i, i+1)$. Hence $\mathrm{wt}_K(\sigma s_i) = \mathrm{wt}_K(\sigma) + 1$. It follows that $\sigma_{i+1} > \sigma_i$. Therefore $((i+1) - \sigma_i)^2 + (i - \sigma_{i+1})^2 > (i - \sigma_i)^2 + ((i+1) - \sigma_{i+1})^2$. Notice that $\vec{\sigma}$ and $\vec{\tau}$ differ only in the σ_i and σ_{i+1}th position, with $\vec{\sigma}_{\sigma_i} = i, \vec{\sigma}_{\sigma_{i+1}} = i+1, \vec{\tau}_{\sigma_i} = i+1$, and $\vec{\tau}_{\sigma_{i+1}} = i$. Hence $\mathrm{wt}_E(\vec{\sigma}) < \mathrm{wt}_E(\vec{\tau})$.

For our induction hypothesis, we shall suppose that $\tau = \sigma s_{i_1} \cdots s_{i_m}$ where
$\mathrm{wt}_K(\sigma s_{i_1} \cdots s_{i_r}) = \mathrm{wt}_K(\sigma s_{i_1} \cdots s_{i_{r-1}}) + 1$ for all $1 \le r \le m$, implies that $\mathrm{wt}_E(\vec{\sigma}) < \mathrm{wt}_E(\vec{\tau})$. Consider now $\tau = \sigma s_{i_1} \cdots s_{i_{m+1}}$ where $\mathrm{wt}_K(\sigma s_{i_1} \cdots s_{i_r}) = \mathrm{wt}_K(\sigma s_{i_1} \cdots s_{i_{r-1}}) + 1$ for all $1 \le r \le m+1$. Then there exists $\sigma^{(m)}$ such that $\sigma^{(m)} = \sigma s_{i_1} \cdots s_{i_m}$ with $\mathrm{wt}_K(\sigma s_{i_1} \cdots s_{i_r}) = \mathrm{wt}_K(\sigma s_{i_1} \cdots s_{i_{r-1}}) + 1$ for all $1 \le r \le m$. We also have $\tau = \sigma^{(m)} s_{i_{m+1}}$ and $\mathrm{wt}_K(\tau) = \mathrm{wt}_K(\sigma^{(m)}) + 1$. Thus by the base case and the induction hypothesis, $\mathrm{wt}_E(\vec{\sigma}) < \mathrm{wt}_E(\overrightarrow{\sigma^{(m)}}) < \mathrm{wt}_E(\vec{\tau})$. □

Corollary 3.20. $\mathrm{wt}_K(\sigma) \le \mathrm{wt}_E(\vec{\sigma})$ *for all* $\sigma \in S_n$.

Theorem 3.4. *If W is a reflection subgroup of S_n and $\mu = e$, then W satisfies the LP-decoding extension condition.*

Proof. By Theorem 12 of [64], for any reflection subgroup G of S_n, there exists a unique element g_0 of minimal Kendall tau weight in the coset g_0G. Therefore, since W is a reflection subgroup of S_n, for each coset of W in S_n there exists a unique element $\lambda^{-1} \in S_n$ such that $\mathrm{wt}_K(\lambda^{-1}) < \mathrm{wt}_K(\lambda^{-1}\sigma)$ for any $\sigma \in W$. We saw previously that for any element $e \neq \sigma \in S_n$, $e < \sigma$ in the weak (right) Bruhat ordering. Thus for all $\sigma \in W$, it follows that $\lambda^{-1} < \lambda^{-1}\sigma$. By Lemma 3.19, $\overrightarrow{\lambda^{-1}}$ is the unique element of minimal Euclidean weight among all associated vectors for permutations in the coset $\lambda^{-1}W$, which implies that the LP-decoding extension condition is satisfied. $\square$

Now that we have shown that reflection subgroups satisfy the LP-decoding extension condition, we will consider some of the minimum distance and code size properties. Because of the varied nature of reflection subgroups, we will focus on a brief example. This example is chosen in part because of the ease with which it can be examined.

Example 3.21. *Consider the reflection subgroup $W \subset S_8$ generated by the fundamental set of reflections:* $\{(0,2),(1,3),(2,4),(3,5),(4,6),(5,7)\}$. *Any permutation $\sigma \in W$ will be of the following form:* $[\boldsymbol{\sigma_0}, \sigma_1, \boldsymbol{\sigma_2}, \sigma_3, \boldsymbol{\sigma_4}, \sigma_5, \boldsymbol{\sigma_6}, \sigma_7]$, *where the bold-faced entries in the permutation can be any permutation of* $\{0,2,4,6\}$ *while the remaining entries can be any permutation of* $\{1,3,5,7\}$. *Thus* $\#W = (4!)^2 = 576$. *It is clear that in this situation a minimum weight permutation is obtained by applying to the identity permutation any single fundamental reflection, such as* $(0,2)$, *resulting in* $[2,1,0,3,4,5,6,7]$. *Therefore the minimum Kendall tau and Euclidean weights are* $\mathrm{wt}_K([2,1,0,3,4,5,6,7]) = 3$ *and* $\mathrm{wt}_E(2,1,0,3,4,5,6,7) = 4$.

The above example is easily generalized to a reflection subgroup W of S_n generated by $\{(k,k+2) \ : \ 0 \leq k \leq n-3\}$ where n is an even number. The size of W would be $\left(\frac{n}{2}!\right)^2$ and the minimum Kendall tau and Euclidean weights would remain 3 and 4 respectively. We may also further generalize the above example to increase the minimum Kendall tau and Euclidean weights by choosing fundamental reflections with larger gaps. For example, we might take the generating set to be of the form $\{(k, k+3)\ :\ 0 \leq k \leq n-4\}$, where n is a multiple of 3. In this case the minimum Kendall tau and Euclidean weights would be 5 and 9 respectively, and $\#W = \left(\frac{n}{3}!\right)^3$.

F. Conclusion

In this chapter we provided a necessary and sufficient condition for extending the LP decoding of [75]. We gave some sample subgroups satisfying the condition and began to analyze related weight distribution. We also proved that reflection subgroups satisfy the LP decoding extenstion condition.

4. Ulam Permutation Codes

A. Introduction

The history of permutation codes dates as far back as the 1960's and 70's, with Slepian, Berger, Blake, and others [11, 15, 69]. However, the application of permutation codes and multipermutation codes for use in non-volatile memory storage systems such as flash memory has received attention in the coding theory literature in recent years [44, 45, 57, 75]. One of the main distance metrics in the literature has been the Kendall tau metric, which is suitable for correction of the type of error expected to occur in flash memory devices. Errors occur in these devices when the electric cell charges storing information leak over time or there is an overshoot of charge level in the rewriting process. For relatively small leak or overshoot errors the Kendtall-τ metric is appropriate. However, it may not be well-suited for large errors within a single cell, which could result when a cell is faulty or damaged.

In 2013, Farnoud et al. proposed permutation codes using the Ulam metric [29]. They showed that the use of the Ulam metric would allow a large leakage or overshoot error within a single cell to be viewed as a single error. This means the Ulam metric may be better suited to combat errors resulting from faulty or damaged cells. Subsequent papers expounded on the use of Ulam metric in multipermutation codes and bounds on the size of permutation codes in the Ulam metric [30, 35]. Meanwhile, Buzaglo et al. discovered the existence of a perfect permutation code under the cyclic Kendall tau metric, and proved the non-existence of perfect permutation codes under the Kendall tau metric for certain parameters [20]. However, the possibility of perfect permutation codes in the Ulam metric had not previously been considered. Exploring this possibility requires first understanding the sizes of Ulam permutation spheres, of which only limited research exists. Even less is known about the size of multipermutation Ulam spheres, which we consider in the following chapter.

In the current chapter we consider two main questions. The first question is: How can permutation Ulam sphere sizes be calculated? One answer to this question is to use Young Tableaux and the RSK-Correspondence (Theorem 4.1). The second question is: Do perfect Ulam permutation codes exists? The answer to this question is that nontrivial perfect Ulam permutation codes do not exist (Theorem 4.2). These two questions are closely related to each other since perfect Ulam permutation code sizes are characterized by Ulam sphere sizes. These main results are summarized in Tables V and VI. Notation appearing on the tables is defined subsequently.

TABLE V
PERMUTATION ULAM SPHERE SIZES

Permutation Ulam Sphere Size Formulas	**Reference**
$\#S(\sigma,t) \;=\; \#S(e,t) \;=\; \sum_{\lambda\in\Lambda}(f^{\lambda})^2$	Theorem 4.1
$\#S(\sigma,1) \;=\; 1+(n-1)^2$	Proposition 4.4

TABLE VI
THEORETICAL LIMIT ON MAXIMUM ULAM PERMUTATION CODE SIZE

Theorem on perfect Ulam permutation codes	**Reference**
Nontrivial perfect t-error correcting permutation codes do not exist	Theorem 4.2

B. Preliminaries and Notation

In this chapter and the next, we utilize the following notation and definitions, generally following conventions established in [29] and [30]. Throughout the next two chapters we will assume that n and r are positive integers, with r dividing n. The symbol $[n]$ denotes the set of integers $\{1, 2, \ldots, n\}$. The symbol $\mathbb{S}_n$ stands for the set of permutations (automorphisms) on $[n]$, i.e., the symmetric group of order $n!$. Note that in this chapter and the next we begin indexes from 1 instead of 0.

For a permutation $\sigma \in \mathbb{S}_n$, we use the notation $\sigma = [\sigma(1), \sigma(2), \ldots, \sigma(n)]$, where for all $i \in [n]$, $\sigma(i)$ is the image of i under σ. With some abuse of notation, we may also use σ to refer to the sequence $(\sigma(1), \sigma(2), \ldots, \sigma(n)) \in [n]^n$. Given two permutations $\sigma, \pi \in \mathbb{S}_n$, the product $\sigma\pi$ is defined in this chapter and the next by $(\sigma\pi)(i) = \sigma(\pi(i))$. In other words, we define multiplication of permutations by composition, e.g., $[2, 1, 5, 4, 3][5, 1, 4, 2, 3] = [3, 2, 4, 1, 5]$. The reader should be cautioned that this differs from multiplication of the previous chapter, keeping rather with the conventions of [29]. The identity permutation $[1, 2, \ldots, n] \in \mathbb{S}_n$ is denoted by e.

An r-regular multiset is a multiset such that each of its element appears exactly r times (i.e., each element is repeated r times). For example, $\{1, 1, 2, 2, 3, 3\}$ is a 2-regular multiset. A **multipermutation** is an ordered tuple whose entries exactly correspond to the elements of a multiset, and in the instance of an r-regular multiset, we call the multipermutation an **r-regular multipermutation**. For example, $(3, 2, 2, 1, 3, 1) \in [3]^6$ is a 2-regular multipermutation of $\{1, 1, 2, 2, 3, 3\}$. Following the work of [30], and because r-regular multipermutations result in the largest potential code space [28], in this chapter we only consider r-regular multipermutations. Hence for the remainder of this chapter "multipermutation" will always refer to an r-regular multipermutation.

Definition ($\mathbf{m}^r_\sigma$). Given $\sigma \in \mathbb{S}_n$ we define a corresponding r-regular multipermutation $\mathbf{m}^r_\sigma$ as follows: for all $i \in [n]$ and $j \in [n/r]$,

$$\mathbf{m}^r_\sigma(i) := j \text{ if and only if } (j-1)r + 1 \leq \sigma(i) \leq jr,$$

and $\mathbf{m}^r_\sigma := (\mathbf{m}^r_\sigma(1), \mathbf{m}^r_\sigma(2), \ldots, \mathbf{m}^r_\sigma(n)) \in [n/r]^n$.

As an example of $\mathbf{m}^r_\sigma$, let $n = 6$, $r = 2$, and $\sigma = [1, 5, 2, 4, 3, 6]$. Then $\mathbf{m}^r_\sigma = (1, 3, 1, 2, 2, 3)$. Note that this definition differs slightly from the correspondence defined in [30], which was defined in terms of the inverse permutation. This is so that certain properties (Remarks 4.1 and 4.2) of the Ulam metric for permutations (the case when $r = 1$) will also hold for general multipermutations. Notice that $\mathbf{m}^1_\sigma = (\sigma(1), \ldots, \sigma(n)) \in [n]^n$, so based on our abuse of notation described in the first paragraph of this section, we may denote $\mathbf{m}^1_\sigma$ simply by σ. In other words, whenever $r = 1$, r-regular multipermutations reduce to permutations, or more accurately their associated sequences.

With the correspondence above, we may define an equivalence relation between elements of $\mathbb{S}_n$. For permutations $\sigma, \pi \in \mathbb{S}_n$, we say that $\sigma \equiv_r \pi$ if and only if $\mathbf{m}^r_\sigma = \mathbf{m}^r_\pi$. The equivalence class $R_r(\sigma)$ of $\sigma \in \mathbb{S}_n$ is defined by $R_r(\sigma) := \{\pi \in \mathbb{S}_n \ : \ \pi \equiv_r \sigma\}$. Note that if $r = 1$, then

$R_r(\sigma)$ is simply the singleton $\{\sigma\}$. For a subset $S \subseteq \mathbb{S}_n$, define $\mathcal{M}_r(S) := \{\mathbf{m}_\sigma^r \,:\, \sigma \in S\}$, i.e. the set of r-regular multipermutations corresponding to elements of S. When $r = 1$, we may identify $\mathcal{M}_r(S)$ simply by S.

We next define the r-regular Ulam distance. For the definition, it is first necessary to define $\ell(\mathbf{x}, \mathbf{y})$. Given sequences $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$, then $\ell(\mathbf{x}, \mathbf{y})$ denotes the length of the longest common subsequence of $\mathbf{x}$ and $\mathbf{y}$ (not to be confused with the longest common substring). More precisely, $\ell(\mathbf{x}, \mathbf{y})$ is the largest integer $k \in \mathbb{Z}_{>0}$ such that there exists a sequence $(a_1, a_2, \ldots, a_k)$ where for all $l \in [k]$, we have $a_l = \mathbf{x}(i_l) = \mathbf{y}(j_l)$ with $1 \leq i_1 < i_2 < \cdots < i_k \leq n$ and $1 \leq j_1 < j_2 < \cdots < j_k \leq n$. For example, $\ell((3,1,2,1,2,3),(1,1,2,2,3,3)) = 4$, since $(1,1,2,3)$ is a common subsequence of both $(3,1,2,1,2,3)$ and $(1,1,2,2,3,3)$ and its length is 4. It does not matter that other equally long common subsequences exist (e.g. $(1,2,2,3)$), as long as there do not exist any longer common subsequences. If $\sigma \in \mathbb{S}_n$, then $\ell(\sigma, e)$ is the length of the longest increasing subsequence of σ, which we denote simply by $\ell(\sigma)$. Similarly, for an r-regular multipermutation $\mathbf{m}_\sigma^r$, we denote the length of the longest non-decreasing subsequence $\ell(\mathbf{m}_\sigma^r, \mathbf{m}_e^r)$ of $\mathbf{m}_\sigma^r$ simply by $\ell(\mathbf{m}_\sigma^r)$.

Definition ($\mathrm{d}_\circ(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r)$, r-regular Ulam distance)**.** Let $\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r \in \mathcal{M}_r(\mathbb{S}_n)$. Define

$$\mathrm{d}_\circ(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r) := \min_{\sigma' \in R_r(\sigma), \pi' \in R_r(\pi)} \mathrm{d}_\circ(\sigma', \pi'),$$

where $\mathrm{d}_\circ(\sigma, \pi) := n - \ell(\sigma, \pi)$. We call $\mathrm{d}_\circ(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r)$ the **r-regular Ulam distance** between $\mathbf{m}_\sigma^r$ and $\mathbf{m}_\pi^r$. In the case when $r = 1$, we may simply say the **Ulam distance** between σ and π and use the notation $\mathrm{d}_\circ(\sigma, \pi)$.

The definition of r-regular Ulam distance above follows the convention of [30], defining the distance in terms of equivalence classes comprised of permutations, although our notation differs. However, it is convenient to think of the distance instead in terms of the multipermutations themselves. A simple argument shows that the r-regular Ulam distance between multipermutations $\mathbf{m}_\sigma^r$ and $\mathbf{m}_\pi^r$ is equal to n minus the length of their longest common subsequence. The details of the argument can be found in the appendices.

Remark 4.1. Let $\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r \in \mathcal{M}_r(\mathbb{S}_n)$. Then

$$\mathrm{d}_\circ(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r) = n - \ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r).$$

Viewed this way, it is easily verified that the r-regular Ulam distance $\mathrm{d}_\circ(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r)$ is a proper metric between the multipermutations $\mathbf{m}_\sigma^r$ and $\mathbf{m}_\pi^r$. Additionally, it is known that in the permutation case, the case when $r = 1$, that the Ulam distance can be characterized in terms of a specific type of permutation known as translocations [29, 35]. We can show a similar relationship for multipermutations. We define translocations below and then give the relationship between the Ulam distance and translocations.

Definition ($\phi(i,j)$, translocation)**.** Given distinct $i, j \in [n]$, define $\phi(i,j) \in \mathbb{S}_n$ as follows:

$$\phi(i,j) := \begin{cases} [1, 2, \ldots, i-1, i+1, i+2, \ldots, j, i, j+1, \ldots, n] & \text{if } i < j \\ \\ [1, 2, \ldots, j-1, i, j, j+1, \ldots, i-1, i+1, \ldots, n] & \text{if } i > j \end{cases}$$

If $i = j$, then define $\phi(i,j) := e$. We refer to $\phi(i,j)$ as a **translocation**, and if we do not specify the indexes i and j we may denote a translocation simply by ϕ.

Intuitively, a translocation is the permutation that results in a delete/insertion operation. More specifically, given $\sigma \in \mathbb{S}_n$ and the translocation $\phi(i,j) \in \mathbb{S}_n$, the product $\sigma\phi(i,j)$ is the result of deleting $\sigma(i)$ from the ith position of σ, then shifting all positions between the ith and jth position by one (left if $i < j$ and right if $i > j$), and finally reinserting $\sigma(i)$ into the new jth position. The top half of Figure 2 illustrates the permutation $\sigma = [6, 2, 8, 5, 4, 1, 3, 9, 7]$ (or its related 3-regular multipermutation $\mathbf{m}_\sigma^3 = (2, 1, 3, 2, 2, 1, 1, 3, 3)$) represented physically by relative cell charge levels and the effect of multiplying σ (or $\mathbf{m}_\sigma^3$) on the right by the translocation $\phi(1, 9)$. The bottom half of Figure 2 illustrates the same σ (or $\mathbf{m}_\sigma^3$) and the effect of $\phi(7, 4)$. Notice that multiplying by $\phi(1, 9)$ corresponds to the error that occurs when the highest (1st) ranked cell suffers charge leakage that results in it being the lowest (9th) ranked cell. Multiplying by $\phi(7, 4)$ corresponds to the error that occurs when the 7th highest cell is overfilled so that it is the 4th highest cell.

It is well-known that $\mathrm{d}_\circ(\sigma, \pi)$ equals the minimum number of translocations needed to transform σ into π [29, 35]. That is, $\mathrm{d}_\circ(\sigma, \pi) = \min\{k \in \mathbb{Z}_{\geq 0} \ : \ \text{there exist } \phi_1, \phi_2, \ldots, \phi_k \text{ such that } \sigma\phi_1\phi_2 \cdots \phi_k = \pi\}$. By applying Remark 4.1, it is also a simple matter to prove that an analogous relationship holds for the r-regular Ulam distance. First, it is necessary to define multiplication between multipermutations and permutations.

The following definition is our own. We define the product $\mathbf{m}^r_\sigma \cdot \pi$ as $\mathbf{m}^r_\sigma \cdot \pi := \mathbf{m}^r_{\sigma\pi}$. Technically speaking, this can be seen as a right group action of the set $\mathbb{S}_n$ of permutations on the set $\mathcal{M}_r(\mathbb{S}_n)$. Since it is possible for different permutations to correspond to the same multipermutation, we should clarify that $\mathbf{m}^r_\sigma = \mathbf{m}^r_\tau$ implies $\mathbf{m}^r_{\sigma\pi} = \mathbf{m}^r_{\tau\pi}$. Indeed this is true because if $\mathbf{m}^r_\sigma = \mathbf{m}^r_\tau$ then for all $i \in [n]$ we have $\mathbf{m}^r_\sigma(i) = \mathbf{m}^r_\tau(i)$, which implies for $j := \mathbf{m}^r_\sigma(i)$ that $(j-1)r+1 \leq \sigma(i) \leq jr$ and $(j-1)r + 1 \leq \tau(i) \leq jr$. This in turn implies that $(j-1)r + 1 \leq \sigma\pi(\pi(i)) \leq jr$ and $(j-1)r + 1 \leq \tau\pi(\pi(i)) \leq jr$, which means $\mathbf{m}_{\sigma\pi}(\pi(i)) = \mathbf{m}_{\tau\pi}(\pi(i))$. Intuitively speaking, the same corresponding elements of the sequences σ and τ still correspond (with a different index) after being multiplied on the right by π. Hence $\mathbf{m}^r_{\sigma\pi} = \mathbf{m}^r_{\tau\pi}$, or by our notation $\mathbf{m}^r_\sigma \cdot \pi = \mathbf{m}^r_\tau \cdot \pi$.

If two multipermutations $\mathbf{m}^r_\sigma$ and $\mathbf{m}^r_\pi$ have a common subsequence of length k, then $\mathbf{m}^r_\sigma$ can be transformed into $\mathbf{m}^r_\pi$ with $n-k$ (but no fewer) delete/insert operations. As with permutations, delete/insert operations correspond to applying (multiplying on the right) a translocation. Hence by Remark 4.1 we can state the following remark about the r-regular Ulam distance. The details of the proof can be found in the appendices.

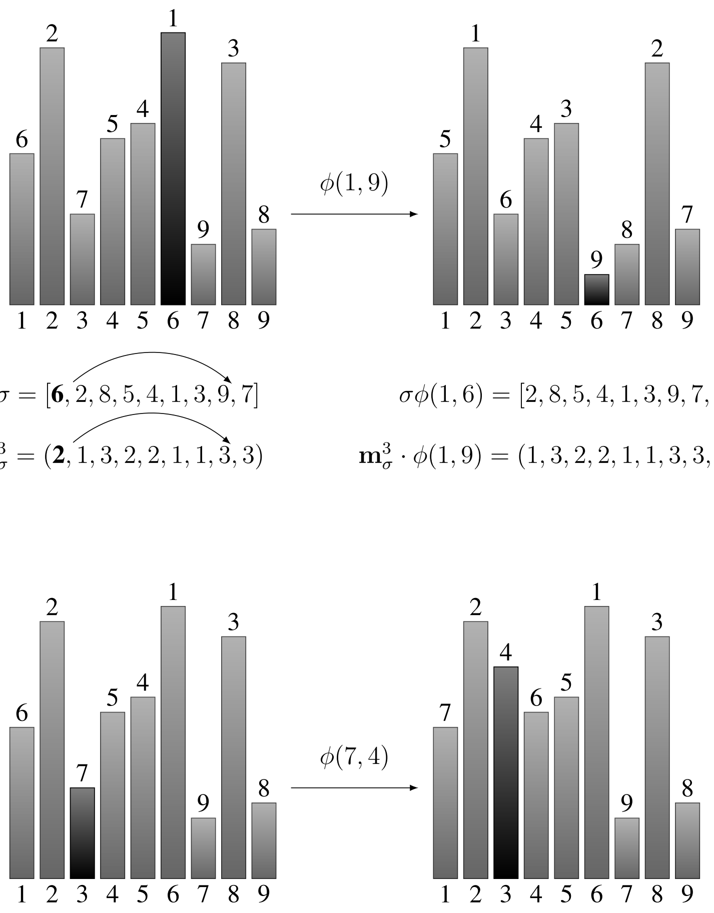
Remark 4.2. Let $\mathbf{m}^r_\sigma, \mathbf{m}^r_\pi \in \mathcal{M}_r(\mathbb{S}_n)$. Then

$$\mathrm{d}_\circ(\mathbf{m}^r_\sigma, \mathbf{m}^r_\pi) = \min\{k \in \mathbb{Z}_{\geq 0} \ : \ \text{there exist } \ \phi_1, \phi_2, \ldots, \phi_k \ \text{ such that } \ \mathbf{m}^r_\sigma \cdot \phi_1\phi_2 \cdots \phi_k = \mathbf{m}^r_\pi\}.$$

We now define the notions of a multipermutation code and an r-regular Ulam sphere.

Definition (r-regular multipermutation code, MPC(n, r), MPC$_\circ(n, r, d)$)**.** Recall that $n, r \in \mathbb{Z}_{>0}$ with $r|n$. An r**-regular multipermutation code** (or simply a **multipermutation code**) is a subset $C \subseteq \mathcal{M}_r(\mathbb{S}_n)$. Such a code is denoted by MPC(n, r), and we say that C is an MPC(n, r). If C is an MPC(n, r) such that $\min\limits_{\mathbf{m}^r_\sigma, \mathbf{m}^r_\pi \in C, \mathbf{m}^r_\sigma \neq \mathbf{m}^r_\pi} \mathrm{d}_\circ(\mathbf{m}^r_\sigma, \mathbf{m}^r_\pi) = d$, then we call C an MPC$_\circ(n, r, d)$.

Fig. 2. Translocation illustration



$$\sigma = [\mathbf{6}, 2, 8, 5, 4, 1, 3, 9, 7] \qquad \sigma\phi(1,6) = [2, 8, 5, 4, 1, 3, 9, 7, \mathbf{6}]$$

$$\mathbf{m}_\sigma^3 = (\mathbf{2}, 1, 3, 2, 2, 1, 1, 3, 3) \qquad \mathbf{m}_\sigma^3 \cdot \phi(1,9) = (1, 3, 2, 2, 1, 1, 3, 3, \mathbf{2})$$

$$\sigma = [6, 2, 8, 5, 4, 1, \mathbf{3}, 9, 7] \qquad \sigma\phi(7,4) = [6, 2, 8, \mathbf{3}, 5, 4, 1, 9, 7]$$

$$\mathbf{m}_\sigma^3 = (2, 1, 3, 2, 2, 1, \mathbf{1}, 3, 3) \qquad \mathbf{m}_\sigma^3 \cdot \phi(7,4) = (2, 1, 3, \mathbf{1}, 2, 2, 1, 3, 3)$$

We refer to any 1-regular multipermutation code simply as a **permutation code**.

Our definition of multipermutation codes is in terms of multipermutations, i.e. ordered tuples, rather than in terms of permutations, i.e. automorphisms. This differs slightly from [30], where multipermutation codes were defined as subsets of $\mathbb{S}_n$ with the requirement that the entire equivalence class of each element in a code was a subset of the code. Next, we define r-regular Ulam spheres.

Definition ($S(\mathbf{m}_\sigma^r, t)$, r-regular multipermutation Ulam sphere)**.** Let $t \in \mathbb{Z}_{\geq 0}$, and $\mathbf{m}_\sigma^r \in \mathcal{M}_r(\mathbb{S}_n)$.

Define

$$S(\mathbf{m}^r_\sigma, t) := \{\mathbf{m}^r_\pi \in \mathcal{M}_r(\mathbb{S}_n) \ : \ \mathrm{d}_\circ(\mathbf{m}^r_\sigma, \mathbf{m}^r_\pi) \leq t\}$$

We call $S(\mathbf{m}^r_\sigma, t)$ the r**-regular multipermutation Ulam sphere**, (or simply the **multipermutation Ulam sphere**) centered at $\mathbf{m}^r_\sigma$ of radius t. We refer to any 1-regular multipermutation Ulam sphere as a **permutation Ulam sphere** and use the simplified notation $S(\sigma, t)$ instead of $S(\mathbf{m}^r_\sigma, t)$.

By Remark 4.1, $S(\mathbf{m}^r_\sigma, t) = \{\mathbf{m}^r_\pi \in \mathcal{M}_r(\mathbb{S}_n) \ : \ n - \ell(\mathbf{m}^r_\sigma, \mathbf{m}^r_\pi) \leq t\}$. The r-regular Ulam sphere definition can also be viewed in terms of translocations. Remark 4.2 implies that $S(\mathbf{m}^r_\sigma, t)$ is equivalent to $\{\mathbf{m}^r_\pi \in \mathcal{M}_r(\mathbb{S}_n) \ : \ \text{there exist } k \in \{0, 1, \ldots, t\} \text{ and } \phi_1, \ldots, \phi_k \text{ such that } \mathbf{m}^r_\sigma \cdot \phi_1 \cdots \phi_k = \mathbf{m}^r_\pi\}$. This is the set of all multipermutations reachable by applying t translocations to the center multipermutation $\mathbf{m}^r_\sigma$.

It is well-known that an MPC$_\circ(n, r, d)$ code is t-error correcting if and only if $d \geq 2t+1$ [30]. This is because if the distance between two codewords is greater or equal to $2t + 1$, then after t or fewer errors (multiplication by t or fewer translocations), the resulting multipermutation remains closer to the original multipermutation than any other multipermutation. We finish this section by defining perfect t error-correcting codes.

Definition (perfect code)**.** Let $C \subseteq \mathcal{M}_r(\mathbb{S}_n)$ be an MPC(n, r). Then C is a perfect t-error correcting code if and only if for all $\mathbf{m}^r_\sigma \in \mathcal{M}_r(\mathbb{S}_n)$, there exists a unique $\mathbf{m}^r_c \in \mathcal{M}_r(C)$ such that $\mathbf{m}^r_\sigma \in S(\mathbf{m}^r_c, t)$. We call such C a **perfect t-error correcting** MPC(n, r), or simply a **perfect code** if the context is clear. A permutation code that is perfect is called a **perfect permutation code**.

A perfect MPC(n, r) partitions $\mathcal{M}_r(\mathbb{S}_n)$. This means the spheres centered at codewords fill the space without overlapping. A perfect code $C \subseteq \mathcal{M}_r(\mathbb{S}_n)$ is said to be **trivial** if either (1) $C = \mathcal{M}_r(\mathbb{S}_n)$ (occurring when $t = 0$); or (2) $\#C = 1$ (occurring when $t = n - r$).

C. Permutation Ulam Sphere Size

This section focuses on the first of two main questions in this chapter: how can we calculate the sizes of permutation Ulam spheres? The answer to this question, in the form of Theorem 4.1, is the first main result of this chapter. The theorem is actually stated in terms of multipermutations, making it also a partial answer to a main question of the following chapter concerning how to calculate multipermutation Ulam sphere sizes. However, unlike permutations, in the case of multipermutations sphere sizes may depend upon the choice of center, limiting the applicability of the theorem for multipermutation Ulam spheres. The proof of the theorem is provided after a necessary lemma is recalled and notation used in the theorem is clarified.

Theorem 4.1. *Let* $t \in \{0, 1, 2 \dots, n-r\}$*, and* $\Lambda := \{\lambda \vdash n \ : \ \lambda_1 \geq n-t\}$*. Then*

$$\#S(\mathbf{m}_e^r, t) \ = \ \sum_{\lambda \in \Lambda} (f^\lambda)(K_r^\lambda). \tag{4}$$

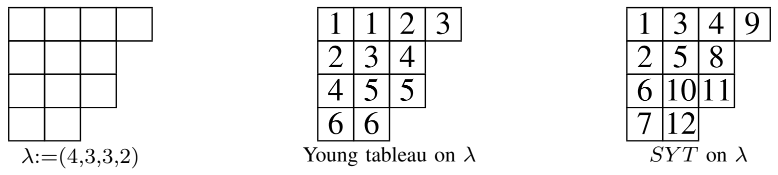
Although this section is primarily concerned with permutation Ulam sphere sizes, many of the results hold for multipermutation Ulam spheres as well, and lemmas and propositions in this section are stated with as much generality as possible. In the case of permutation codes, perfect codes and sphere sizes are related as follows: a perfect t-error correcting permutation code $C \subseteq \mathbb{S}_n$, if it exists, will have cardinality $\#C = n!/\#S(c,t)$, where $c \in C$. Hence one of the first questions that may be considered in exploring the possibility of a perfect code (the second main question of the chapter) is the feasibility of a code of such size. As noted in [29], for any $\sigma \in \mathbb{S}_n$, we have $\#S(\sigma, t) = \#S(e, t)$. Hence calculation of permutation Ulam sphere sizes can be reduced to the case when the identity is the center.

One way to calculate permutation Ulam sphere sizes centered at e is to use Young tableaux and the RSK-Correspondence. It is first necessary to introduce some basic notation and definitions regarding Young diagrams and Young tableaux. Additional information on the subject can be found in resources such as [32, 67, 70].

A **Young diagram** is a left-justified collection of cells with a (weakly) decreasing number of cells in each row below. Listing the number of cells in each row gives a partition $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_k)$ of n, where n is the total number of cells in the Young diagram. The notation

$\lambda \vdash n$ is used to mean λ is a partition of n. Because the partition $\lambda \vdash n$ defines a unique Young diagram and vice versa, a Young diagram may be referred to by its associated partition $\lambda \vdash n$. For example, the partition $\lambda := (4, 3, 3, 2) \vdash 12$ has the corresponding Young diagram pictured on the left side of Figure 3.

Fig. 3. Young diagram and SYT



A **Young tableau** is a filling of a Young diagram $\lambda \vdash n$ with the following two qualities: (1) cell values are weakly increasing across each row; and (2) cell values are strictly increasing down each column. One possible Young tableau is pictured in the center of Figure 3. A **standard Young tableau**, abbreviated by SYT, is a filling of a Young diagram $\lambda \vdash n$ with the following three qualities: (1) cell values are strictly increasing across each row; (2) cell values are strictly increasing down each column; and (3) each of the integers 1 through n appears exactly once. One possible SYT on $\lambda := (4, 3, 3, 2)$ is pictured in the right side of Figure 3.

Among other things, the famous RSK-correspondence ([32, 70]) provides a bijection between r-regular multipermutations $\mathbf{m}_\sigma^r$ and ordered pairs (P, Q) on the same Young diagram $\lambda \vdash n$, where P is a Young tableau whose members come from $\mathbf{m}_\sigma^r$ and Q is a SYT. The next lemma, a stronger form of which appears in [32], is an application of the RSK-correspondence.

Lemma 4.3. *Let $\mathbf{m}_\sigma^r \in \mathcal{M}_r(\mathbb{S}_n)$ and let P and Q, both on $\lambda \vdash n$, be the pair of Young tableaux associated with $\mathbf{m}_\sigma^r$ by the RSK-correspondence. Then*

$$\lambda_1 = \ell(\mathbf{m}_\sigma^r).$$

In words, the above lemma says that λ_1, the number of columns in the P (or equivalently Q) associated with $\mathbf{m}_\sigma^r$ by the RSK-correspondence, is equal to $\ell(\mathbf{m}_\sigma^r)$, the length of the longest non-decreasing subsequence of $\mathbf{m}_\sigma^r$. The lemma implies that for all $k \in [n]$, the size of the set $\{\mathbf{m}_\sigma^r \in \mathcal{M}_r(\mathbb{S}_n) \ : \ \ell(\mathbf{m}_\sigma^r) = k\}$ is equal to the sum of the number of ordered pairs (P, Q) on each Young diagram $\lambda \vdash n$. Following conventional notation ([32, 70]), f^λ denotes the number of SYT on $\lambda \vdash n$. We denote by K_r^λ (our own notation) the number of Young tableaux on $\lambda \vdash n$ such that each $i \in [n/r]$ appears exactly r times. We are now able to prove Theorem 4.1, which states the relationship between $\#S(\mathbf{m}_e^r, t)$, f^λ, and K_r^λ.

Proof of Theorem 4.1:
Assume $t \in \{0, 1, \ldots, n-1\}$, and let $\Lambda := \{\lambda \vdash n \ : \ \lambda_1 \geq n - t\}$. Furthermore, let $\Lambda^{(l)} := \{\lambda \vdash n \ : \ \lambda_1 = l\}$, the set of all partitions of n having exactly l columns. By the RSK-Correspondence and Lemma 4.3, there is a bijection between the set $\{\mathbf{m}_\sigma^r \ : \ \ell(\mathbf{m}_\sigma^r) = l\}$ and the set of ordered pairs (P, Q) where both P and Q have exactly l columns. This implies that $\#\{\mathbf{m}_\sigma^r \ : \ \ell(\mathbf{m}_\sigma^r) = l\} = \sum\limits_{\lambda \in \Lambda^{(l)}} (f^\lambda)(K_r^\lambda)$ (here $\#A$ is an alternate notation for the cardinality of a set that we prefer for conditionally defined sets). By Remark 4.1, $\#S(\mathbf{m}_e^r, t) = \#\{\mathbf{m}_\sigma \ : \ \mathrm{d}_\circ(\mathbf{m}_e^r, \mathbf{m}_\sigma^r) \leq t\} = \#\{\mathbf{m}_\sigma \ : \ \ell(\mathbf{m}_\sigma^r) \geq n - t\}$. Hence it follows that $\#S(\mathbf{m}_e^r, t) = \sum\limits_{\lambda \in \Lambda} (f^\lambda)(K_r^\lambda)$. □

Because K_1^λ is equivalent to f^λ by definition, in the case of permutation Ulam spheres, equation (4) simplifies to

$$\#S(e, t) = \sum_{\lambda \in \Lambda} (f^\lambda)^2. \tag{5}$$

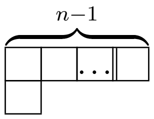
In both equation (4) and (5), the famous hook length formula, due to Frame, Robinson, and Thrall [31, 32], provides a way to calculate f^λ. Within the hook length formula, the notation $(i, j) \in \lambda$ is used to refer to the cell in the ith row and jth column of a Young diagram $\lambda \vdash n$. The notation $h(i, j)$ denotes the **hook length** of $(i, j) \in \lambda$, i.e., the number of boxes below or to the right of (i, j), including the box (i, j) itself. More formally, $h(i, j) := \#(\{(i, j^*) \in \lambda \ : \ j^* \geq j\} \cup \{(i^*, j) \in \lambda \ : \ i^* \geq i\})$. The **hook-length formula** is as follows:

$$f^\lambda = \frac{n!}{\prod\limits_{(i,j) \in \lambda} h(i, j)}.$$

Applying the hook length formula to Theorem 4.1, we may explicitly calculate Ulam permutation sphere sizes, as demonstrated in the following propositions. These propositions will be useful later to show the nonexistence of nontrivial t-error correcting perfect permutation codes for $t \in \{1, 2, 3\}$. Proposition 4.4 is stated in terms of general multipermutation Ulam spheres.

Proposition 4.4. $\#S(\mathbf{m}_e^r, 1) = 1 + (n-1)(n/r - 1)$.

Proof. First note that $\#S(\mathbf{m}_e^r, 0) = \#\{\mathbf{m}_e^r\} = 1$. There is only one possible partition $\lambda \vdash n$ such that $\lambda_1 = n-1$, namely $\lambda' := (n-1, 1)$, with its Young diagram pictured below.



Therefore by Theorem 4.1, $\#S(\mathbf{m}_e^r, 1) = 1 + (f^{\lambda'})(K_r^{\lambda'})$. Applying the hook length formula, we obtain $f^{\lambda'} = n - 1$. The value $K_r^{\lambda'}$ is characterized by possible fillings of row 2 with the stipulation that each $i \in [n/r]$ must appear exactly r times in the diagram. In this case, since there is only a single box in row 2, the possible fillings are $i \in [n/r - 1]$, each of which yields a unique Young tableau of the desired type. Hence $K_r^{\lambda'} = n/r - 1$, which implies that $\#S(\mathbf{m}_e^r, 1) = 1 + (n-1)(n/r - 1)$. $\square$

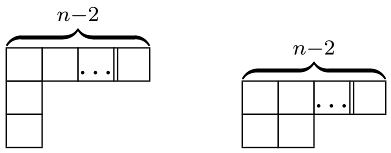
Setting $r = 1$, Proposition 4.4 implies that $\#S(e, 1) = 1 + (n-1)^2$. The next two propositions continue the same vein of reasoning, but focus on permutation Ulam spheres. Such individual cases could be considered indefinitely. In fact, a recurrence equation providing an alternative method of calculating permutation Ulam sphere sizes for reasonably small radii is also known [50]. However, the following two propositions are the last instances of significance in this chapter as their results will be necessary to prove the second main result of the chapter.

Proposition 4.5. *Let $n > 3$ and $\sigma \in \mathbb{S}_n$. Then*

$$\#S(\sigma, 2) = 1 + (n-1)^2 + \left(\frac{(n)(n-3)}{2}\right)^2 + \left(\frac{(n-1)(n-2)}{2}\right)^2.$$

Proof. Assume $n > 3$ and $\sigma \in \mathbb{S}_n$. Note first that $\#S(\sigma, 2) = \#S(\sigma, 1) + \#\{\pi \in \mathbb{S}_n : \ell(\pi) = n - 2\}$. The only partitions $\lambda \vdash n$ such that $\lambda_1 = n - 2$ are $\lambda^{(1)} := (n-2, 1, 1)$ and

$\lambda^{(2)} := (n-2, 2)$, with their respective Young diagrams pictured below.

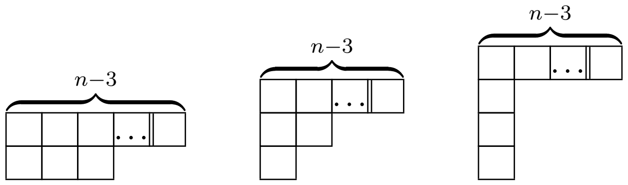


Using the hook length formula, $f^{\lambda^{(1)}}$ and $f^{\lambda^{(2)}}$ may be calculated to yield: $f^{\lambda^{(1)}} = ((n)(n-3))/2$ and $f^{\lambda^{(2)}} = ((n-1)(n-2))/2$. Following the same reasoning as in Proposition 4.4 yields the desired result. $\square$

Lemma 4.6. *Let $n > 5$ and $\sigma \in \mathbb{S}_n$ Then*

$$\begin{aligned} \#S(\sigma, 3) &= 1 + (n-1)^2 + \left(\frac{(n)(n-3)}{2}\right)^2 + \left(\frac{(n-1)(n-2)}{2}\right)^2 \\ &+ \left(\frac{(n)(n-1)(n-5)}{6}\right)^2 + \left(\frac{(n)(n-2)(n-4)}{3}\right)^2 + \left(\frac{(n-1)(n-2)(n-3)}{6}\right)^2. \end{aligned}$$

Proof. The proof is essentially the same as the proof for Proposition 4.5. In this case $\#\{\pi \in \mathbb{S}_n : \ell(\pi) = n-3\}$ can be calculated by considering the partitions $\lambda^{(1)} := (n-3, 3)$, $\lambda^{(2)} := (n-3, 2, 1)$, and $\lambda^{(3)} := (n-3, 1, 1, 1)$, the only Young diagrams having $n-3$ columns. These Young diagrams are pictured below.



Applying the hook length formula to $\lambda^{(1)}$, $\lambda^{(2)}$, and $\lambda^{(3)}$ and adding the value from Proposition 4.5 yields the result.

$\square$

D. Nonexistence of Nontrivial Perfect Ulam Permutation Codes

The previous section demonstrated how to calculate permutation Ulam sphere sizes. In this section, again focusing on permutation codes, we utilize sphere size calculations to prove the following theorem, establishing a theoretical limit on the maximum size of Ulam permutation codes. This is the second main contribution of this chapter. The proof of the theorem can be found at the end of the current section.

Theorem 4.2. *There do not exist any nontrivial perfect permutation codes in the Ulam metric.*

In 2013, Farnoud et. al ([29]) proved the following upper bound on the size of an Ulam permutation code $C \subseteq \mathbb{S}_n$ with minimum Ulam distance d (i.e. C is an MPC$(n, 1, d)$).

$$\#C \leq (n - d + 1)! \tag{6}$$

Hence one strategy to prove the non-existence of perfect permutation codes is to show that the size of a perfect code must necessarily be larger than the upper-bound given above. Note that for equation (6) to make sense, d must be less than or equal to $n - 1$. This is always true since the maximum Ulam distance between any two permutations in $\mathbb{S}_n$ is $n-1$, achieved when permutations are in reverse order of each other (e.g., $\mathrm{d}_\circ(e, [n, n-1, ..., 1]) = n - 1$).

Lemma 4.7. *There do not exist any (nontrivial) single-error correcting perfect permutation codes.*

Proof. Assume that $C \subseteq \mathbb{S}_n$ is a perfect single-error correcting permutation code. Recall that C is trivial code if either $C = \mathbb{S}_n$ or if $\#C = 1$. If $n \leq 2$, then for all $\sigma, \pi \in \mathbb{S}_n$, we have $\pi \in S(\sigma, 1)$, which implies that C is a trivial code. Thus we may assume that $n > 2$.

We proceed by contradiction. Since C is a perfect single-error correcting permutation code, C is an MPC$_\circ(n, 1, d)$ with $3 \leq d \leq n - 1$ and $\#C = n!/\#S(\sigma, 1) = n!/(1 + (n-1)^2)$ by Proposition 4.4. However, inequality (6) implies that the code size $\#C \leq (n-2)!$. Hence, it suffices to show that $\#C = n!/(1 + (n-1)^2) > (n-2)!$, which is true if and only if $n > 2$. $\square$

Similar arguments may also be applied to show that no nontrivial perfect t-error correcting codes exist for $t \in \{2, 3\}$. This is the subject of the next two lemmas. The remaining cases, when $t > 3$, are treated toward the end of this section.

Lemma 4.8. *There do not exist any (nontrivial) perfect 2-error correcting permutation codes.*

Proof. Assume that C is a perfect 2-error correcting permutation code. Similarly to the proof of Lemma 4.7, if $n \leq 3$, then C is a trivial code consisting of a single element, so we may assume $n > 3$. Again we proceed by contradiction.

Since $C \subseteq \mathbb{S}_n$ is a perfect 2-error correcting code, then C is an $\mathrm{MPC}_\circ(n, 1, d)$ code with $5 \leq d \leq n-1$ and Proposition 4.5 implies

$$\#C \;=\; \frac{n!}{\#S(\sigma, 2)} \;=\; \frac{n!}{1 + (n-1)^2 + \left(\frac{(n)(n-3)}{2}\right)^2 + \left(\frac{(n-1)(n-2)}{2}\right)^2}.$$

By Inequality (6), $\#C \leq (n-4)!$, so it suffices to prove that

$$\frac{n!}{1 + (n-1)^2 + \left(\frac{(n)(n-3)}{2}\right)^2 + \left(\frac{(n-1)(n-2)}{2}\right)^2} - (n-4)! \;>\; 0,$$

which is easily shown by elementary methods to be true for $n > 3$. □

Lemma 4.9. *There do not exist any (nontrivial) perfect 3-error correcting codes.*

Proof outline. Assume that $C \subseteq \mathbb{S}_n$ is a perfect 3-error correcting code. Similarly to the proof of Lemmas 4.7 and 4.8, if $n \leq 7$, then C is a trivial code, so we may assume that $n > 7$. The remainder of the proof follows the same reasoning as the proof for Lemma 4.8, utilizing the sphere size calculated in Proposition 4.6. □

For small values of t, explicit sphere calculations work well for showing the non-existence of nontrivial perfect t-error correcting codes. However, for each radius t, the size of the sphere $S(e, t)$ is equal to $\#S(e, t-1) + \#\{\pi \in \mathbb{S}_n \;:\; \ell(\pi) = n - t\}$. This means each sphere size calculation of radius t requires calculation of sphere sizes for radii from 0 through $t-1$. Hence such explicit calculations are impractical for large values of t. For values of $t > 3$, another method can be used to show that nontrivial perfect codes do not exist. The next lemma provides a sufficient condition to conclude that perfect codes do not exist. In the proof of the lemma, the notation $\binom{n}{t}$ denotes the usual combinatorial choice function.

Lemma 4.10. *Let t be a nonnegative integer such that $n \geq 2t$. If the following inequality holds,* *then no nontrivial perfect t-error correcting permutation codes exist in $\mathbb{S}_n$:*

$$F(n,t) := \frac{((n-t)!)^2\, t!}{n!(n-2t)!} > 1. \tag{7}$$

*We call the above inequality the **overlapping condition**.*

Proof. Assume that t is a nonnegative integer such that $n \geq 2t$. We proceed by contrapositive. Suppose $C \subset \mathbb{S}_n$ is a nontrivial perfect t-error correcting permutation code. We want to show that $F(n,t) \leq 1$. Since C is a perfect code, we know it is also an MPC$_\circ(n,1,d)$ code with $2t+1 \leq d$ and by inequality (6), $\#C \leq (n-2t)!$. At the same time, for any $\sigma \in \mathbb{S}_n$, we have $\#S(\sigma,t) = \#S(e,t)$, which is less than or equal to $\binom{n}{n-t}(n!)/(n-t)!$, since any permutation $\pi \in S(e,t)$ can be obtained by first choosing $n-r$ elements of e to be in increasing order, and then arranging the remaining t elements into π. Of course this method will generally result in double counting some permutations in $S(e,t)$, hence the inequality. Now

$$\#S(\sigma,t) \;\leq\; \binom{n}{n-t}\frac{n!}{(n-t)!} \text{ implies that } \frac{(n-t)!}{\binom{n}{t}} \;\leq\; \frac{n!}{\#S(\sigma,t)} \;=\; \#C \;\leq\; (n-2t)!.$$

Moreover, $(n-t)!/\binom{n}{t} \leq (n-2t)!$ if and only if $F(n,t) \leq 1$. □

Notice that the overlapping condition is never satisfied for $t = 1$. However, the following proposition will imply that as long as $t > 1$, then the overlapping condition may be satisfied for sufficiently large n.

Proposition 4.11. *Let t be a nonnegative integer such that $n \geq 2t$. Then $\lim_{n\to\infty} F(n,t) = t!$.*

Proof. Assume t is a nonnegative integer such that $n \geq 2t$. Then

$$\begin{aligned}
\lim_{n\to\infty} F(n,t) &= \lim_{n\to\infty} \frac{(n-t)(n-t-1)\cdots(n-2t+1)(n-2t)!(n-t)!t!}{(n)(n-1)\cdots(n-t+1)(n-t)!(n-2t)!} \\
&= \lim_{n\to\infty} \frac{(n-t)(n-t-1)\cdots(n-2t+1)t!}{(n)(n-1)\cdots(n-t+1)} \\
&= \lim_{n\to\infty} \frac{(n^{t-1})t!}{n^{t-1}} = t!
\end{aligned}$$

□

The proposition above means that for any nonnegative integer t less than or equal to $n/2$, there is some value k such that for all values of n larger than k, there does not exist a perfect t-error correcting code. The question remains of how large the value of k must be before it is guaranteed that perfect t-error correcting codes do not exist.

TABLE VII
NON-FEASIBILITY OF PERFECT t-ERROR CORRECTING CODES

t	$\min n$ satisfying (7)	t	$\min n$ satisfying (7)
1	N/A	6	13
2	8	7	14
3	8	8	16
4	10	9	18
5	11	10	20

Table VII compares positive integer values t versus $\min\{n \in \mathbb{Z}_{>0} : F(n,t) > 1\}$. Values were determined via numerical computer calculation. The table suggests that for $t > 6$, the minimum value of n satisfying the overlapping condition is $n = 2t$. If what the table appears to suggest is true, then in view of Proposition 4.11, we may rule out perfect t-correcting codes for any $t > 6$. The next lemma formalizes what is implied in the table by providing parameters for which the overlapping condition is always satisfied. In combination with Lemma 4.10, the implication is that nontrivial perfect permutation codes do not exist for these parameters. The remaining cases are also easily dealt with.

Lemma 4.12. *Let t be an integer greater than 6. Then $n \geq 2t$ implies that the overlapping condition is satisfied.*

Proof. Assume t is an integer greater than 6. We begin the proof of the lemma by showing that if $n = 2t$, then the desired inequality holds. We assume that $n = 2t$ and proceed by induction on t.

For the base case, let $t = 7$. Then $n = 14$, and $F(n,t) = ((7!)^3)/(14!) \approx 1.46 > 1$. As the induction hypothesis, suppose it is true that $F(2t,t) = ((t!)^3)/((tk)!) > 1$. We wish to show that the following inequality holds:

$$F(2(t+1), t+1) \quad = \quad \frac{((t+1)!)^3}{(2(t+1))!} \quad > \quad 1.$$

Here

$$\frac{((t+1)!)^3}{(2(t+1))!} \quad = \quad \left(\frac{(t!)^3}{(2t)!}\right)\left(\frac{(t+1)^3}{(2t+1)(2t+2)}\right).$$

By our induction hypothesis, the first term of the right hand side, $(t!)^3/(2t)!$, is greater than 1, so it suffices to show that $(t+1)^3/(2t+1)(2t+2)) \;\geq\; 1$. Note here that

$$\frac{(t+1)^3}{(2t+1)(2t+2)} \quad > \quad \frac{(t+1)^3}{(2t+2)(2t+2)} \quad = \quad \frac{(t+1)^3}{4(t+1)^2} \quad = \quad \frac{t+1}{4},$$

which is greater than 1 whenever $t > 3$. Of course $t > 6$ by assumption, so the desired conclusion follows.

Thus far we have technically only proven that $F(n,t) > 1$ whenever $n = 2t$. However, it is a simple matter to show that the same is true whenever $n > 2t$ as well. We begin by supposing that $F(n,t) > 1$. Then

$$F(n+1,t) \quad = \quad \frac{((n+1-t)!)^2 t!}{(n+1)!(n+1-2t)!} \quad = \quad F(n,t) \cdot \frac{(n+1-t)^2}{(n+1)(n+1-2t)}$$

is necessarily greater than 1 whenever $((n+1-t)^2)/((n+1)(n+1-2t)) \;\geq\; 1$, which is true for all values of n and t. $\square$

Lemma 4.12 required that $n \geq 2t$. However, if $n < 2t$, then it is impossible for a nontrivial perfect t-error correcting permutation code to exist. In fact, we may say something even stronger.

Remark 4.13. If $t \in \mathbb{Z}_{>0}$ such that $n \leq 2t+1$, then it is impossible for a nontrivial perfect t-error correcting permutation code to exist.

To understand why Remark 4.13 is true, consider two permutations within $\mathbb{S}_n$ of maximal Ulam distance apart. The most obvious example of which would be the identity element e and the only-decreasing permutation $\omega^* := [n, n-1, ..., 1]$. Notice that $S(e,t) = \{\pi \in \mathbb{S}_n \;:\; \ell(\pi) \geq n-t\}$, which means that every permutation whose longest increasing subsequence is at least $n-t$ is in the sphere centered at e. Meanwhile, there is at least one permutation $\sigma \in \mathbb{S}_n$ such that $\ell(\sigma) = 1+t$ and $\sigma \in S(\omega^*, t)$, since we may apply successive translocations to ω^* in such a way that the longest increasing subsequence is increased with each translocation. As long as $n \leq 2t+1$, then $n-t \leq t+1 = 1+t$, implying that $\ell(\sigma) = 1+t \geq n-t$, which implies that

$\sigma \in S(e,t) \cap S(\omega^*,t)$. Therefore the only perfect code possible when $n \leq 2t+1$ is a single element code, i.e. a trivial code. Consolidating all previous results, we are now able to prove Theorem 4.2.

Proof of Theorem 4.2: First, by Lemmas 4.7, 4.8, and 4.9, there do not exist any nontrivial perfect t-error correcting permutation codes for $t \in \{1,2,3\}$. Next note that $F(n,r)$ increases as n increases, and thus by numerical results (see Table VII), for all $t \in \{4,5,6\}$ the overlapping condition is satisfied whenever $n \geq 2t+2$. Therefore by Lemma 4.10, and Remark 4.13, there are no nontrivial perfect t-error correcting permutation codes for $t \in \{4,5,6\}$. Finally, by Lemmas 4.10, 4.12, and Remark 4.13, there are no nontrivial perfect r-error correcting permutation codes for $t > 6$. □

E. Conclusion

This chapter first considered and answered two questions. The first question concerned Ulam sphere sizes and the second concerned the possibility of perfect codes. It was shown that Ulam sphere sizes can be calculated explicitly for reasonably small radii using an application of the RSK-correspondence. It was then shown, partially using the aforementioned sphere-calculation method, that nontrivial perfect Ulam permutation codes do not exist. These new results are summarized in Tables V and VI, found in the introduction.

5. Ulam Multipermutation Codes

A. Introduction

The previous chapter focused on permutation codes. This chapter focuses on multipermutation codes, We consider two questions. First: how can multipermutation Ulam sphere sizes be calculated? Theorem 4.1 and Theorem 5.1 show how to calculate sphere sizes for certain parameters. Second: What is the maximum possible Ulam multipermutation code size? Lemmas 5.11, 5.13, and 5.14 (as well as Lemmas 5.26 and 5.27 for the special binary case) provide new upper and lower bounds on the maximal code size. These main results are summarized in Tables VIII and IX. Notation appearing on the tables is defined in subsequent sections.

TABLE VIII
Multipermutation Ulam Sphere Sizes

Multipermutation Ulam Sphere Size Formulas and Bounds	**Reference**
$\#S(\mathbf{m}_e^r, t) \;=\; \sum\limits_{\lambda \in \Lambda} (f^\lambda)(K_r^\lambda)$	Theorem 4.1
$\#S(\mathbf{m}_\sigma^r, 1) \;=\; 1 + (n-1)^2 - \#SD(\mathbf{m}_\sigma^r) - \#AD(\mathbf{m}_\sigma^r)$	Theorem 5.1
$1 + (n-1)(n/r - 1) \;=\; \#S(\mathbf{m}_e^r, 1) \;\leq\; \#S(\mathbf{m}_\sigma^r, 1)$	Lemma 5.10 and Theorem 4.1
Non-Binary Case: $\#S(\mathbf{m}_\sigma^r, 1) \;\leq\; \#S(\mathbf{m}_\omega^r, 1) \;=\; 1 + (n-1)^2 - (r-1)n$	Lemma 5.12
Binary Case: $\#S(\mathbf{m}_\sigma^r, 1) \;<\; U(r)$	Corollary 5.25

B. Multipermutation Ulam Sphere Size and Duplication Sets

Thus far we have focused primarily on permutations, but we wish to extend the discussion to multipermutations. With both permutations and multipermutations, the number of possible messages is limited by the number of distinguishable relative rankings in the physical scheme. However, multipermutations may significantly increase the total possible messages compared to ordinary permutations, as observed in [30]. For example, if only k different charge levels are utilized at a given time, then permutations of length k can be stored. Hence, in r blocks of length k, one may store $(k!)^r$ potential messages. On the other hand, if one uses r-regular multipermutations in the same set of blocks, then $(kr)!/(r!)^k$ potential messages are possible.

TABLE IX
THEORETICAL LIMITS ON MAXIMUM ULAM MULTIPERMUTATION CODE SIZE

Ulam Multipermutation Code Max Size Bounds	**Value**	**Reference**
1-error correcting code upper bound	$\#C \leq \frac{n!}{(r!)^{n/r}(1+(n-1)(n/r-1))}$	Lemma 5.11
Non-Binary Case: Perfect 1-error correcting code lower bound	$\frac{n!}{(r!)^{n/r}((1+(n-1)^2)-(r-1)n)} \leq \#C$	Lemma 5.13
Binary Case: Perfect 1-error correcting code lower bound	$\frac{n!}{(r!)^2(U(r))} \leq \#C$	Lemma 5.26
Non-Binary Case: MPC$_\circ(n,r,d)$ lower bound	$\frac{n!}{(r!)^{n/r}(1+(n-1)^2-(r-1)n)^{d-1}} \leq \#C$	Lemma 5.14
Binary Case: MPC$_\circ(n,r,d)$ lower bound	$\frac{n!}{(r!)^2(U(r))^{d-1}} \leq \#C$	Lemma 5.27

The r-regular multipermutation Ulam sphere sizes play an important role in understanding the potential code size for MPC$_\circ(n,r,d)$'s. For example, the well-known sphere-packing bounds and Gilbert-Varshamov type bounds rely on calculating, or at least bounding sphere sizes. In this section we analyze how to calculate r-regular multipermutation Ulam sphere sizes, providing an answer to the first of the two main questions addressed in this chapter. Recall that a partial answer to this question was given in Theorem 4.1, but the theorem was applicable to the special case when $\mathbf{m}_e^r$ was chosen as the center. The next theorem provides a way to calculate radius 1 spheres for any center using the concept of duplication sets. Notation used in the theorem is defined subsequently and the proof is given toward the end of the section.

Theorem 5.1. *Recall that $n, r \in \mathbb{Z}_{>0}$ and $r|n$. Let $\mathbf{m}_\sigma^r \in \mathcal{M}_r(\mathbb{S}_n)$. Then*

$$\#S(\mathbf{m}_\sigma^r, 1) = 1 + (n-1)^2 - \#SD(\mathbf{m}_\sigma^r) - \#AD(\mathbf{m}_\sigma^r).$$

In the permutation case, the Ulam metric is known to be left-invariant, i.e. given $\sigma, \pi, \tau \in \mathbb{S}_n$, we have $\mathrm{d}_\circ(\sigma, \pi) = \mathrm{d}_\circ(\tau\sigma, \tau\pi)$ [29]. Left-invariance implies that permutation sphere sizes do not depend on the choice of center. Unfortunately, it is easily confirmed by counterexample that left invariance does not generally hold for the r-regular Ulam metric. Moreover, it is also easily confirmed that in the multipermutation Ulam sphere case, the choice of center has an impact on the size of the sphere, even when the radius remains unchanged (e.g. compare Proposition 4.4

to Proposition 5.12 in the next section). Hence we wish to consider spheres with various center multipermutations.

To aid with calculating such sphere sizes, we first find it convenient to introduce (as our own definition) the following subset of the set of translocations.

Definition (T_n, unique set of translocations)**.** Define $T_n := \{\phi(i,j) \in \mathbb{S}_n \,:\, i-j \neq 1\}$.

We call T_n the **unique set of translocations**.

In words, T_n is the set of all translocations, except translocations of the form $\phi(i, i-1)$. We exclude translocations of this form because they can be modeled by translocations of the form $\phi(i-1, i)$, and are therefore redundant. We claim that the set T_n is precisely the set of translocations needed to obtain all unique permutations within the Ulam sphere of radius 1 via multiplication (right action). Moreover, there is no redundancy in the set, meaning no smaller set of translocations yields the entire Ulam sphere of radius 1 when multiplied with a given center permutation. These facts are stated in the next lemma.

Lemma 5.1. *Let $\sigma \in \mathbb{S}_n$. Then $S(\sigma,1) = \{\sigma\phi \in \mathbb{S}_n \,:\, \phi \in T_n\}$, and $\#T_n = \#S(\sigma,1)$.*

Proof. Let $\sigma \in \mathbb{S}_n$. We will first show that $S(\sigma,1) = \{\sigma\phi \in \mathbb{S}_n \,:\, \phi \in T_n\}$. Note that

$$S(\sigma,1) \quad = \quad \{\pi \in \mathbb{S}_n \,:\, \mathrm{d}_\circ(\sigma,\pi) \leq 1\} \quad = \quad \{\sigma\phi(i,j) \in \mathbb{S}_n \,:\, i,j \in [n]\}.$$

It is trivial that

$$T_n \quad = \quad \{\phi(i,j) \in \mathbb{S}_n \,:\, i-j \neq 1\} \quad \subseteq \quad \{\phi(i,j) \in \mathbb{S}_n \,:\, i,j \in [n]\}.$$

Therefore $\{\sigma\phi \in \mathbb{S}_n \,:\, \phi \in T_n\} \subseteq S(\sigma,1)$.

To see why $S(\sigma,1) \subseteq \{\sigma\phi \in \mathbb{S}_n \,:\, \phi \in T_n\}$, consider any $\sigma\phi(i,j) \in \{\sigma\phi(i,j) \in \mathbb{S}_n \,:\, i,j \in [n]\} = S(\sigma,1)$. If $i-j \neq 1$, then $\phi(i,j) \in T_n$, and thus $\sigma\phi(i,j) \in \{\sigma\phi \in \mathbb{S}_n \,:\, \phi \in T_n\}$. Otherwise, if $i-j = 1$, then $\sigma\phi(i,j) = \sigma\phi(j,i)$, and $i-j=1$ implies $j-i = -1 \neq 1$, so $\phi(j,i) \in T_n$. Hence $\sigma\phi(i,j) = \sigma\phi(j,i) \in \{\sigma\phi \in \mathbb{S}_n \,:\, \phi \in T_n\}$.

Next we show that $\#T_n = \#S(\sigma,1)$. By Proposition 4.4, $\#S(\sigma,1) = 1 + (n-1)^2$. On the other hand, $\#T_n = \#\{\phi(i,j) \in \mathbb{S}_n \,:\, i-j \neq 1\}$. If $i = 1$, then there are n values $j \in [n]$ such that $i-j \neq 1$. Otherwise, if $i \in [n]$ but $i \neq 1$, then there are $n-1$ values $j \in [n]$ such that $i-j \neq 1$. However, for all $i,j \in [n]$, $\phi(i,i) = \phi(j,j) = e$ so that there are $n-1$ redundancies. Therefore $\#T_n = n + (n-1)(n-1) - (n-1) = 1 + (n-1)^2$. $\square$

Although the Ulam sphere centered at $\sigma \in \mathbb{S}_n$ of radius 1 can be characterized by all permutations obtainable by applying (multiplying on the right) a translocation to σ, the previous lemma shows that some translocations are redundant. That is, there are translocations $\phi_1 \neq \phi_2$ such that $\sigma\phi_1 = \sigma\phi_2$. In the case of permutations, the set T_n has no such redundancies. If $\phi_1, \phi_2 \in T_n$, then $\sigma\phi_1 = \sigma\phi_2$ implies $\phi_1 = \phi_2$. However, in the case of multipermutations, the set T_n can generally be shrunken further to exclude redundancies.

Given $\mathbf{m}_\sigma^r \in \mathcal{M}_r(\mathbb{S}_n)$, the sphere $S(\mathbf{m}_\sigma^r, 1) = \{\mathbf{m}_\pi^r \in \mathcal{M}_r(\mathbb{S}_n) \ : \ \text{there exist } \phi \text{ such that } \mathbf{m}_\sigma^r \cdot \phi = \mathbf{m}_\pi\} = \{\mathbf{m}_\sigma^r \cdot \phi \in \mathcal{M}_r(\mathbb{S}_n) \ : \ \phi \in T_n\}$. However, it is possible that there exist $\phi_1, \phi_2 \in T_n$ such that $\phi_1 \neq \phi_2$, but $\mathbf{m}_\sigma^r \cdot \phi_1 = \mathbf{m}_\sigma^r \cdot \phi_2$. In such an instance we may refer to either ϕ_1 or ϕ_2 as a **duplicate translocation** for $\mathbf{m}_\sigma^r$. If we remove all duplicate translocations for $\mathbf{m}_\sigma^r$ from T_n, then the resulting set will have the same cardinality as the r-regular Ulam sphere of radius 1 centered at $\mathbf{m}_\sigma^r$. The next definition (our own) is a standard set of duplicate translocations. It is called standard because as long as $r \neq 1$ it always exists and is of predictable size.

Definition ($SD(\mathbf{m})$, standard duplication set)**.** Given a tuple $\mathbf{m} \in \mathbb{Z}^n$, define

$$SD(\mathbf{m}) := \{\phi(i,j) \in T_n \backslash \{e\} \ : \ \mathbf{m}(i) = \mathbf{m}(j) \text{ or } \mathbf{m}(i) = \mathbf{m}(i-1)\}$$

We call $SD(\mathbf{m})$ the **standard duplication set** for $\mathbf{m}$.

If we take an r-regular multipermutation $\mathbf{m}_\sigma^r$, then removing the general set of duplications from T_n equates to removing a set of duplicate translocations. These duplications come in two varieties. The first variety corresponds to the first condition of the $SD(\mathbf{m})$ definition, when $\mathbf{m}(i) = \mathbf{m}(j)$. For example, if $\mathbf{m}_\sigma^2 = (1,3,2,2,3,1)$, then we have $\mathbf{m}_\sigma^2 \cdot \phi(1,5) = (3,2,2,3,1,1) = \mathbf{m}_\sigma^2 \cdot \phi(1,6)$, since $\mathbf{m}_\sigma^2(1) = 1 = \mathbf{m}_\sigma^2(6)$. This is because moving the first 1 to the left or to the right of the last 1 results in the same tuple. The second variety corresponds to the second condition of the of $SD(\mathbf{m})$ definition above, when $\mathbf{m}(i) = \mathbf{m}(i-1)$. For example, if $\mathbf{m}_\sigma^2 = (1,3,2,2,3,1)$ as before, then for all $j \in [6]$, we have $\mathbf{m}_\sigma^2 \cdot \phi(3,j) = \mathbf{m}_\sigma^2 \cdot \phi(4,j)$. This is because any translocation that deletes and inserts the second of the two adjacent 2's does not result in a different tuple when compared to deleting and inserting the first of the two adjacent 2's.

Lemma 5.2. *Let* $\mathbf{m}_\sigma^r \in \mathcal{M}_r(\mathbb{S}_n)$. *Then* $S(\mathbf{m}_\sigma^r, 1) = \{\mathbf{m}_\sigma^r \cdot \phi \in \mathcal{M}_r(\mathbb{S}_n) \ : \ \phi \in T_n \backslash SD(\mathbf{m}_\sigma^r)\}$.

Proof. Assume $\mathbf{m}_\sigma^r \in \mathcal{M}_r(\mathbb{S}_n)$. First note that $S(\mathbf{m}_\sigma^r, 1) = \{\mathbf{m}_\sigma^r \cdot \phi \in \mathcal{M}_r(\mathbb{S}_n) \ : \ \phi \in T_n\}$. Hence it suffices to show that for all $\phi(i,j) \in SD(\mathbf{m}_\sigma^r)$, there exist some $i', j' \in [n]$ such that $\phi(i',j') \in T_n \backslash SD(\mathbf{m}_\sigma^r)$ and $\mathbf{m}_\sigma^r \cdot \phi(i,j) = \mathbf{m}_\sigma^r \cdot \phi(i',j')$. We proceed by dividing the proof into two main cases. Case I is when $\mathbf{m}_\sigma^r(i) \neq \mathbf{m}_\sigma^r(i-1)$ or $i = 1$. Case II is when $\mathbf{m}_\sigma^r(i) = \mathbf{m}_\sigma^r(i-1)$.

Case I (when $(\mathbf{m}_\sigma^r(i) \neq \mathbf{m}_\sigma^r(i-1)$ or $i = 1)$ can be split into two subcases:

Case IA: $i < j$

Case IB: $i > j$.

We can ignore the instance when $i = j$, since $\phi(i,j) \in SD(\mathbf{m}_\sigma^r)$ implies $i \neq j$. For case IA, if for all $p \in [i,j]$ (for $a, b \in \mathbb{Z}$ with $a < b$, the notation $[a,b] := \{a, a+1, \ldots, b\}$) we have $\mathbf{m}_\sigma^r(i) = \mathbf{m}_\sigma^r(p)$, then $\mathbf{m}_\sigma^r \cdot \phi(i,j) = \mathbf{m}_\sigma^r \cdot e$. Thus setting $i' = j' = 1$ yields the desired result. Otherwise, if there exists $p \in [i,j]$ such that $\mathbf{m}_\sigma^r(i) \neq \mathbf{m}_\sigma^r(p)$, then let $j^* := j - \min\{k \in \mathbb{Z}_{>0} \ : \ \mathbf{m}_\sigma^r(i) \neq \mathbf{m}_\sigma^r(j-k)\}$. Then $\phi(i,j^*) \in T_n \backslash SD(\mathbf{m}_\sigma^r)$ and $\mathbf{m}_\sigma^r \cdot \phi(i,j) = \mathbf{m}_\sigma^r \cdot \phi(i,j^*)$. Thus setting $i' = i$ and $j' = j^*$ yields the desired result. Case IB is similar to Case IA.

Case II (when $\mathbf{m}_\sigma^r(i) = \mathbf{m}_\sigma^r(i-1)$), can also be divided into two subcases.

Case IIA: $i < j$

Case IIB: $i > j$.

As in Case I, we can ignore the instance when $i = j$. For Case IIA, if for all $p \in [i,j]$ we have $\mathbf{m}_\sigma^r(i) = \mathbf{m}_\sigma^r(p)$, then $\mathbf{m}_\sigma^r \cdot \phi(i,j) = \mathbf{m}_\sigma^r \cdot e$, so setting $i = j = 1$ achieves the desired result. Otherwise, if there exists $p \in [i,j]$ such that $\mathbf{m}_\sigma^r(i) \neq \mathbf{m}_\sigma^r(p)$, then let $i^* := i - \min\{k \in \mathbb{Z}_{>0} \ : \ (\mathbf{m}_\sigma^r(i) \neq \mathbf{m}_\sigma^r(i-k-1))$ or $(i - k = 1)\}$. Then $\mathbf{m}_\sigma^r \cdot \phi(i,j) = \mathbf{m}_\sigma^r \cdot \phi(i^*,j)$ and either one of the following is true: (1) $\phi(i^*, j) \notin D_{i^*}(\mathbf{m}^r_\sigma)$ implies $\phi(i^*, j) \notin SD(\mathbf{m}^r_\sigma)$, so set $i' = i^*$ and $j' = j$; or (2) by Case IA there exist $i', j' \in [n]$ such that $\phi(i', j') \in T_n \backslash SD(\mathbf{m}^r_\sigma)$ and $\mathbf{m}^r_\sigma \cdot \phi(i', j') = \mathbf{m}^r_\sigma \cdot \phi(i^*, j) = \mathbf{m}^r_\sigma \cdot \phi(i, j)$. Case IIB is similar to Case IIA. $\square$

While Lemma 5.2 shows that $SD(\mathbf{m}^r_\sigma)$ is a set of duplicate translocations for $\mathbf{m}^r_\sigma$, we have not shown that $T_n \backslash SD(\mathbf{m}^r_\sigma)$ is the set of minimal size having the quality that $S(\mathbf{m}^r_\sigma, 1) = \{\mathbf{m}^r_\sigma \cdot \phi \in \mathcal{M}_r(\mathbb{S}_n) \ : \ \phi \in T_n \backslash SD(\mathbf{m}^r_\sigma)\}$. In fact it is not minimal. In some instances it is possible to remove further duplicate translocations to reduce the set size. We will define another set of duplicate translocations, but a few preliminary definitions are first necessary.

We say that $\mathbf{m} \in \mathbb{Z}^n$ is **alternating** if for all odd integers $1 \le i \le n$, $\mathbf{m}(i) = \mathbf{m}(1)$ and for all even integers $2 \le i' \le n$, $\mathbf{m}(i') = \mathbf{m}(2)$ but $\mathbf{m}(1) \neq \mathbf{m}(2)$. In other words, any alternating tuple is of the form $(a, b, a, b, \ldots, a, b)$ or $(a, b, a, b, \ldots, a)$ where $a, b \in \mathbb{Z}$ and $a \neq b$. Any singleton is also said to be alternating. Now for integers $1 \le i \le n$ and $0 \le k \le n - i$, the **substring** $\mathbf{m}[i, i+k]$ of $\mathbf{m}$ is defined as $\mathbf{m}[i, i+k] := (\mathbf{m}(i), \mathbf{m}(i+1), \ldots \mathbf{m}(i+k))$. Given a substring $\mathbf{m}[i, j]$ of $\mathbf{m}$, the **length** of $\mathbf{m}[i, j]$, denoted by $|\mathbf{m}[i, j]|$, is defined as $|\mathbf{m}[i, j]| := j - i + 1$. As an example, if $\mathbf{m}' := (1, 2, 2, 4, 2, 4, 3, 1, 3)$, then $\mathbf{m}'[3, 6] = (2, 4, 2, 4)$ is an alternating substring of $\mathbf{m}'$ of length 4.

Definition ($AD(\mathbf{m})$, alternating duplication set)**.** Given $\mathbf{m} \in \mathbb{Z}^n$, define

$$AD(\mathbf{m}) := \{ \ \phi(i, j) \in T_n \backslash SD(\mathbf{m}) \ : \ i < j \text{ and there exists } \ k \in [i, j-2] \text{ such that}$$
$$(\phi(j, k) \in T_n \backslash SD(\mathbf{m})) \text{ and } (\mathbf{m} \cdot \phi(i, j) = \mathbf{m} \cdot \phi(j, k)) \ \}.$$

We call $AD(\mathbf{m})$ the **alternating duplication set** for $\mathbf{m}$ because it is only nonempty when $\mathbf{m}$ contains an alternating substring of length at least 4. For each $i \in [n]$, also define $AD_i(\mathbf{m}) := \{\phi(i, j) \in AD(\mathbf{m}) \ : \ j \in [n]\}$. Notice that $AD(\mathbf{m}) = \bigcup_{i=1}^{n} AD_i(\mathbf{m})$.

In the example of $\mathbf{m}' := (1, 2, 2, 4, 2, 4, 3, 1, 3)$ above, $\mathbf{m}' \cdot \phi(2, 6) = \mathbf{m}' \cdot \phi(6, 3)$ and $\phi(2, 6), \phi(6, 3) \in T_9 \backslash SD(\mathbf{m}')$, implying that $\phi(2, 6) \in AD(\mathbf{m}')$. In fact, it can easily be shown that $AD(\mathbf{m}') = \{\phi(2, 6)\}$. In order to simplify the discussion of the alternating duplication set, we find the following lemma useful.

Lemma 5.3. *Let $\mathbf{m} \in \mathbb{Z}^n$ and $i \in [n]$. Then $AD_i(\mathbf{m}) \neq \varnothing$ if and only if*

1) $\mathbf{m}(i) \neq \mathbf{m}(i-1)$

2) There exists $j \in [i+1, n]$ and $k \in [i, j-2]$ such that

i) For all $p \in [i, k-1]$, $\mathbf{m}(p) = \mathbf{m}(p+1)$

ii) $\mathbf{m}[k, j]$ is alternating

iii) $|\mathbf{m}[k, j]| \geq 4$.

Proof. Let $\mathbf{m} \in \mathbb{Z}^n$ and $i \in [n]$. We will first assume 1) and 2) in the lemma statement and show that $AD_i(\mathbf{m})$ is not empty. Suppose $\mathbf{m}(i) \neq \mathbf{m}(i-1)$, and that there exists $j \in [i+1, n]$ and $k \in [i, j-2]$ such that for all $p \in [i, k-1]$, we have $\mathbf{m}(p) = \mathbf{m}(p+1)$. Suppose also that $\mathbf{m}[k, j]$ is alternating with $|\mathbf{m}[k, j]| \geq 4$.

For ease of notation, let $a := \mathbf{m}(k) = \mathbf{m}(k+2)$ and $b := \mathbf{m}(k+1) = \mathbf{m}(k+3)$ so that $\mathbf{m}[k, k+3] = (a, b, a, b) \in \mathbb{Z}^4$. Then

$$\begin{aligned}(\mathbf{m} \cdot \phi(i, k+3))[k, k+3] &= (\mathbf{m} \cdot \phi(k, k+3))[k, k+3] \\ &= (b, a, b, a) \\ &= (\mathbf{m} \cdot \phi(k+3, k))[k, k+3].\end{aligned}$$

Moreover, for all $p \notin [k, k+3]$, we have $(\mathbf{m} \cdot \phi(i, k+3))(p) = \mathbf{m}(p) = (\mathbf{m} \cdot \phi(k+3, k))(p)$. Therefore $\mathbf{m} \cdot \phi(i, k+3) = \mathbf{m} \cdot \phi(k+3, k)$. Also notice that $\mathbf{m}(i) \neq \mathbf{m}(i-1)$ implies that $\mathbf{m} \cdot \phi(i, k+3) \notin SD(\mathbf{m})$. Hence $\phi(i, k+3) \in AD_i(\mathbf{m})$.

We now prove the second half of the lemma. That is, we assume that $AD_i(\mathbf{m}) \neq \varnothing$ and then show that 1) and 2) necessarily hold. Suppose that $AD_i(\mathbf{m})$ is nonempty. Then $\mathbf{m}(i) \neq \mathbf{m}(i-1)$, since otherwise there would not exist any $\phi(i, j) \in T_n \backslash SD(\mathbf{m})$.

Let $j \in [i+1, n]$ and $k \in [i, j-2]$ such that $\phi(j, k) \in T_n \backslash SD(\mathbf{m})$ and $\mathbf{m} \cdot \phi(i, j) = \mathbf{m}(j, k)$. Existence of such j, k, and $\phi(j, k)$ is guaranteed by definition of $AD_i(\mathbf{m})$ and the fact that $AD_i(\mathbf{m})$ was assumed to be nonempty. Then for all $p \in [i, k-1]$, we have $\mathbf{m}(p) = \mathbf{m}(p+1)$ and for all $p \in [k, j-2]$, we have $\mathbf{m}(p) = \mathbf{m}(p+2)$. Hence either $\mathbf{m}[k, j]$ is alternating, or else for all $p, q \in [k, j]$, we have $\mathbf{m}(p) = \mathbf{m}(q)$. However, the latter case is impossible, since it would imply that for all $p, q \in [i, j]$ that $\mathbf{m}(p) = \mathbf{m}(q)$, which would mean $\phi(j, k) \notin T_n \backslash SD(\mathbf{m})$, a contradiction. Therefore $\mathbf{m}[k,j]$ is alternating.

It remains only to show that $|\mathbf{m}[k,j]| \geq 4$. Since $k \in [i, j-2]$, it must be the case that $|\mathbf{m}[k,j]| \geq 3$. However, if $|\mathbf{m}[k,j]| = 3$ (which occurs when $k = j-2$), then $(\mathbf{m} \cdot \phi(i,j))(j) = \mathbf{m}(i) = \mathbf{m}(k) \neq \mathbf{m}(k+1) = (\mathbf{m} \cdot \phi(j,k)(j)$, which implies that $\mathbf{m} \cdot \phi(i,j) \neq \mathbf{m} \cdot \phi(j,k)$, a contradiction. Hence $|\mathbf{m}[k,j]| \geq 4$. $\square$

One implication of Lemma 5.3 is that there are only two possible forms for $\mathbf{m}[i,j]$ where $\phi(i,j) \in AD_i(\mathbf{m})$. The first possibility is that $\mathbf{m}[i,j]$ is an alternating substring of the form $(a,b,a,b,\ldots,a,b)$ (here $a,b \in \mathbb{Z}$), so that $\mathbf{m}[i,j] \cdot \phi(i,j)$ is of the form $(b,a,b,a\ldots,b,a)$. In this case, as long as $|\mathbf{m}[i,j]| \geq 4$, then setting $k = i$ implies that $k \in [i, j-2]$, that $\phi(j,k) \in T_n \backslash SD(\mathbf{m})$, and that $\mathbf{m}[i,j] \cdot \phi(i,j) = \mathbf{m}[i,j] \cdot \phi(j,k)$.

The other possibility is that $\mathbf{m}[i,j]$ is of the form $(\underbrace{a,a,a,\ldots,a}_{k},\underbrace{b,a,b,\ldots,a,b}_{n-k})$ (again $a,b \in \mathbb{Z}$), so that $\mathbf{m}[i,j] \cdot \phi(i,j)$ is of the form $(\underbrace{a,\ldots,a}_{k-1},\underbrace{b,a,b,\ldots,b,a}_{n-k+1})$. Again in this case, as long as $|\mathbf{m}[i,j]| \geq 4$, then $k \in [i, j-2]$ with $\phi(j,k) \in T_n \backslash SD(\mathbf{m})$ and $\mathbf{m}[i,j] \cdot \phi(i,j) = \mathbf{m}[i,j] \cdot \phi(j,k)$. To simplify the calculation of $\#AD(\mathbf{m}_\sigma^r)$, we wish to define a set of equal size that is easier to count. The two remarks that follow the definition are obvious, but are helpful in proving that the size of the new set is equal to the size of $AD(\mathbf{m})$.

Definition $(AD^*(\mathbf{m}))$**.** Given $\mathbf{m} \in \mathbb{Z}^n$, define

$$AD^*(\mathbf{m}) := \{\ (i,j) \in [n] \times [n] \ \ : \ \ (\mathbf{m}[i,j] \text{ is alternating}),\ (|\mathbf{m}[i,j]| \geq 4),\ \text{ and } (|\mathbf{m}[i,j]| \text{ is even})\ \}.$$

For each $i \in [n]$, also define $AD_i^*(\mathbf{m}) := \{(i,j) \in AD^*(\mathbf{m}) \ : \ j \in [n]\}$. Notice that $AD^*(\mathbf{m}) = \bigcup\limits_{i=1}^{n} AD_i^*(\mathbf{m})$.

Remark 5.4. If $\mathbf{m} \in \mathbb{Z}^n$ is alternating and n is even, then $\mathbf{m} \cdot \phi(1,n) = \mathbf{m} \cdot \phi(n,1)$.

Remark 5.5. If $\mathbf{m} \in \mathbb{Z}^n$ is alternating, $n \geq 3$, and n is odd, then $\mathbf{m} \cdot \phi(1,n) \neq \mathbf{m} \cdot \phi(n,1)$.

Lemma 5.6. *Let* $\mathbf{m} \in \mathbb{Z}^n$*. Then* $\#AD(\mathbf{m}) = \#AD^*(\mathbf{m})$

Proof. Let $\mathbf{m} \in \mathbb{Z}^n$. The idea of the proof is simple. Each element $\phi(i,j) \in AD(\mathbf{m})$ involves exactly one alternating sequence of length greater or equal to 4, so the set sizes must be equal.

We formalize the argument by showing that $\#AD(\mathbf{m}) \leq \#AD^*(\mathbf{m})$ and then that $\#AD^*(\mathbf{m}) \leq \#AD(\mathbf{m})$.

To see why $\#AD(\mathbf{m}) \leq \#AD^*(\mathbf{m})$, we define a mapping $map : [n] \to [n]$, which maps index values either to the beginning of the nearest alternating subsequence to the right, or else to n. For all $i \in [n]$, let

$$map(i) := \begin{cases} i + \min\{p \in \mathbb{Z}_{\geq 0} \; : \; (\mathbf{m}(i) \neq \mathbf{m}(i+p+1)) \text{ or } (i+p=n)\} & (\text{if } \mathbf{m}(i) \neq \mathbf{m}(i-1) \\ & \text{or } i=1) \\ n & (\text{otherwise}) \end{cases}$$

Notice by definition of map, if $i, i' \in [n]$ such that $i \neq i'$, and if $\mathbf{m}(i) \neq \mathbf{m}(i-1)$ or $i = 1$ and at the same time $\mathbf{m}(i') \neq \mathbf{m}(i'-1)$ or $i' = 1$, then $map(i) \neq map(i')$.

Now for each $i \in [n]$, if $\mathbf{m}(i) \neq \mathbf{m}(i-1)$ or $i = 1$, then $\#AD_i(\mathbf{m}) = \#AD^*_{map(i)}(\mathbf{m})$ by Lemma 5.3 and the two previous remarks. Otherwise, if $\mathbf{m}(i) = \mathbf{m}(i-1)$, then $\#AD_i(\mathbf{m}) = \#AD^*_{map(i)}(\mathbf{m}) = 0$. Therefore $\#AD_i(\mathbf{m}) \leq \#AD^*_i(\mathbf{m})$. This is true for all $i \in [n]$, so $\#AD(\mathbf{m}) \leq \#AD^*(\mathbf{m})$.

The argument to show that $\#AD^*(\mathbf{m}) \leq \#AD(\mathbf{m})$ is similar, except it uses the following function $map^* : [n] \to [n]$ instead of map. For all $i \in [n]$, let

$$map^*(i) := \begin{cases} i - \min\{p \in \mathbb{Z}_{\geq 0} \; : \; (\mathbf{m}(i) \neq \mathbf{m}(i-p-1)) \text{ or } (i-p=1)\} & (\text{if } \mathbf{m}(i) \neq \mathbf{m}(i-1) \\ & \text{or } i=n) \\ n & (\text{otherwise}) \end{cases}$$

□

By definition, calculating $\#AD^*(\mathbf{m})$ equates to calculating the number of alternating substrings $\mathbf{m}[i,j]$ of $\mathbf{m}$ such that the length of the substring is both even and longer than 4. We can simplify the calculation of $AD(\mathbf{m})$ further by establishing a relation to the following quantity.

Definition ($\psi(n)$, $\psi(\mathbf{x})$)**.** Define

$$\psi(n) := \left\lfloor \frac{(n-2)^2}{4} \right\rfloor, \qquad \text{and for } \mathbf{x} \in \mathbb{Z}^*, \text{ define} \qquad \psi(\mathbf{x}) := \sum_{i=1}^{|\mathbf{x}|} \psi(\mathbf{x}(i)),$$

where $|\mathbf{x}|$ denotes the length of the tuple $\mathbf{x}$.

While we define both $\psi(n)$ and $\psi(\mathbf{x})$ here, we will not make use of $\psi(\mathbf{x})$ until the following section. The next lemma relates $\psi(n)$ to the calculation of $AD(\mathbf{m})$.

Lemma 5.7. *Let* $\mathbf{m}$ *be an alternating string. Then*

$$\#AD(\mathbf{m}) = \psi(|\mathbf{m}|)$$

Proof. Assume $\mathbf{m}$ is an alternating string and let $|\mathbf{m}| = n$. By Lemma 5.6, $\#AD(\mathbf{m}) = \#AD^*(\mathbf{m}) = \#\left(\bigcup_{i=1}^{n} AD_i^*(\mathbf{m})\right)$. Since $\mathbf{m}$ was assumed to be alternating,

$$\begin{aligned} \#\left(\bigcup_{i=1}^{n} AD_i^*(\mathbf{m})\right) &= \#\{(i,j) \in [n] \times [n] \ : \ |\mathbf{m}[i,j]| \geq 4 \text{ and } |\mathbf{m}[i,j]| \text{ is even}\} \\ &= \#\{(i,j) \in [n] \times [n] \ : \ j - i + 1 \in A\}, \end{aligned}$$

where A is the set of even integers between 4 and n, i.e. $A := \{a \in [4,n] \ : \ a \text{ is even}\}$. For each $a \in A$, we have

$$\begin{aligned} \#\{(i,j) \in [n] \times [n] \ : \ j - i + 1 = a\} &= \#\{i \in [n] \ : \ i \in [1, n-a+1]\} \\ &= n - a + 1. \end{aligned}$$

Therefore $\#AD(\mathbf{m}) = \sum_{a \in A} (n - a + 1)$. In the case that n is even, then

$$\sum_{a \in A} (n - a + 1) \quad = \quad \sum_{i=2}^{n/2} (n - 2i + 1) \quad = \quad \left(\frac{n-2}{2}\right)^2 \quad = \quad \psi(n).$$

In the case that n is odd, then

$$\sum_{a \in A} (n - a + 1) \quad = \quad \sum_{i=2}^{(n-1)/2} (n - 2i + 1) \quad = \quad \left(\frac{n-3}{2}\right)\left(\frac{n-1}{2}\right) \quad = \quad \psi(n).$$

□

Notice that by Lemma 5.7, it suffices to calculate $\#AD(\mathbf{m})$ for locally maximal length alternating substrings of $\mathbf{m}$. An alternating substring $\mathbf{m}[i, j]$ is of **locally maximal length** if and only if 1) $\mathbf{m}[i-1]$ is not alternating or $i = 1$; and 2) $\mathbf{m}[i, j+1]$ is not alternating or $j = n$.

Finally, we define the general set of duplications, $D(\mathbf{m})$. The lemma that follows the definition also shows that removing the set $D(\mathbf{m}_\sigma^r)$ from T_n removes all duplicate translocations associated with $\mathbf{m}_\sigma^r$.

Definition ($D(\mathbf{m})$, duplication set)**.** Given $n \in \mathbb{Z}_{>0}$ and $\mathbf{m} \in \mathbb{Z}^n$, define

$$D(\mathbf{m}) := SD(\mathbf{m}) \cup AD(\mathbf{m}).$$

We call $D(\mathbf{m})$ the **duplication set** for $\mathbf{m}$. For each $i \in [n]$, we also define $D_i(\mathbf{m}) := \{\phi(i, j) \in D(\mathbf{m}) \ : \ j \in [n]\}$.

Lemma 5.8. *Let* $\mathbf{m}_\sigma \in \mathcal{M}_r(\mathbb{S}_n)$ *and* $\phi_1, \phi_2 \in T_n \backslash D(\mathbf{m}_\sigma^r)$*. Then* $\phi_1 = \phi_2$ *if and only if* $\mathbf{m}_\sigma^r \cdot \phi_1 = \mathbf{m}_\sigma^r \cdot \phi_2$.

Proof. Assume $\mathbf{m}_\sigma \in \mathcal{M}_r(\mathbb{S}_n)$ and $\phi_1, \phi_2 \in T_n \backslash D(\mathbf{m}_\sigma^r)$. If $\phi_1 = \phi_2$ then $\mathbf{m}_\sigma^r \cdot \phi_1 = \mathbf{m}_\sigma^r \cdot \phi_2$ trivially. It remains to prove that $\mathbf{m}_\sigma^r \cdot \phi_1 = \mathbf{m}_\sigma^r \cdot \phi_2$ implies $\phi_1 = \phi_2$. We proceed by contrapositive. Suppose that $\phi_1 \neq \phi_2$. We want to show that $\mathbf{m}_\sigma^r \cdot \phi_1 \neq \mathbf{m}_\sigma^r \cdot \phi_2$. Let $\phi_1 := \phi(i_1, j_1)$ and $\phi_2 := \phi(i_2, j_2)$. The remainder of the proof can be split into two main cases: Case I is if $i_1 = i_2$ and Case II is if $i_1 \neq i_2$.

Case I (when $i_1 = i_2$), can be further divided into two subcases:

$$\text{Case IA: } \mathbf{m}_\sigma^r(i_1) = \mathbf{m}_\sigma^r(i_1 - 1)$$

$$\text{Case IB: } \mathbf{m}_\sigma^r(i_1) \neq \mathbf{m}_\sigma^r(i_1 - 1).$$

Case IA is easy to prove. We have $D_{i_1}(\mathbf{m}_\sigma^r) = D_{i_2}(\mathbf{m}_\sigma^r) = \{\phi(i_1, j) \in T_n \backslash \{e\} \ : \ j \in [n]\}$, so $\phi_1 = e = \phi_2$, a contradiction. For Case IB, we can first assume without loss of generality that

$j_1 < j_2$ and then split into the following smaller subcases:

i) $(j_1 < i_1)$ and $(j_2 > i_1)$

ii) $(j_1 < i_1)$ and $(j_2 \leq i_1)$

iii) $(j_1 > i_1)$ and $(j_2 > i_1)$

iv) $(j_1 > i_1)$ and $(j_2 \leq i_1)$.

However, subcase iv) is unnecessary since it was assumed that $j_1 < j_2$, so $j_1 > i_1$ implies $j_2 > j_1 > i_1$. Subcase ii) can also be reduced to $(j_1 < i_1)$ and $(j_2 < i_1)$ since $j_2 \neq i_2 = i_1$. Each of the remaining subcases is proven by noting that there is some element in the multipermutation $\mathbf{m}_\sigma^r \cdot \phi_1$ that is necessarily different from $\mathbf{m}_\sigma^r \cdot \phi_2$. For example, in subcase i), we have $\mathbf{m}_\sigma^r \cdot \phi_1(j_1) = \mathbf{m}_\sigma^r(i_1) \neq \mathbf{m}_\sigma^r(j_1) = \mathbf{m}_\sigma^r \cdot \phi_2(j_1)$. Subcases ii) and iii) are solved similarly.

Case II (when $i_1 \neq i_2$) can be divided into three subcases:

Case IIA: $(\mathbf{m}_\sigma^r(i_1) = \mathbf{m}_\sigma^r(i_1 - 1)$ and $\mathbf{m}_\sigma^r(i_2) = \mathbf{m}_\sigma^r(i_2 - 1))$,

Case IIB: either

$(\mathbf{m}_\sigma^r(i_1) = \mathbf{m}_\sigma^r(i_1 - 1)$ and $\mathbf{m}_\sigma^r(i_2) \neq \mathbf{m}_\sigma^r(i_2 - 1))$

or $(\mathbf{m}_\sigma^r(i_1) \neq \mathbf{m}_\sigma^r(i_1 - 1)$ and $\mathbf{m}_\sigma^r(i_2) = \mathbf{m}_\sigma^r(i_2 - 1))$,

Case IIC: $(\mathbf{m}_\sigma^r(i_1) \neq \mathbf{m}_\sigma^r(i_1 - 1)$ and $\mathbf{m}_\sigma^r(i_2) \neq \mathbf{m}_\sigma^r(i_2 - 1))$.

Case IIA is easily solved by mimicking the proof of Case IA. Case IIB is also easily solved as follows. First, without loss of generality, we assume that $\mathbf{m}_\sigma^r(i_1) = \mathbf{m}_\sigma^r(i_1 - 1)$ and $\mathbf{m}_\sigma^r(i_2) \neq \mathbf{m}_\sigma^r(i_2 - 1)$. Then $D_{i_1}(\mathbf{m}_\sigma^r) = \{\phi(i_1, j) \in T_n \backslash \{e\} \ : \ j \in [n]\}$, so $\phi_1 = e$. Therefore we have $\mathbf{m}_\sigma^r \cdot \phi_1(j_2) = \mathbf{m}_\sigma^r(j_2) \neq \mathbf{m}_\sigma^r(i_2) = \mathbf{m}_\sigma^r \cdot \phi_2(i_2 - 1)$.

Finally, for Case IIC, without loss of generality we may assume that $i_1 < i_2$ and then split into the following four subcases:

$$\text{i) } (j_1 < i_2) \text{ and } (j_2 \geq i_2)$$

$$\text{ii) } (j_1 < i_2) \text{ and } (j_2 < i_2)$$

$$\text{iii) } (j_1 \geq i_2) \text{ and } (j_2 \geq i_2)$$

$$\text{iv) } (j_1 \geq i_2) \text{ and } (j_2 < i_2).$$

However, since $\phi(i_2, j_2) \in T_n \backslash D(\mathbf{m}_\sigma^r)$ implies $i_2 \neq j_2$, subcases i) and iii) can be reduced to $(j_1 < i_2)$ and $(j_2 > i_2)$ and $(j_1 \geq i_2)$ and $(j_2 > i_2)$ respectively. For subcase i), we have $\mathbf{m}_\sigma^r \cdot \phi_1(j_1) = \mathbf{m}_\sigma^r(i_1) \neq \mathbf{m}_\sigma^r(j_1) = \mathbf{m}_\sigma^r \cdot \phi_2(j_1)$. Subcases ii) and iii) are solved in a similar manner. For subcase iv), if $j_1 > i_2$, then $\mathbf{m}_\sigma^r \cdot \phi_1(j_1) = \mathbf{m}_\sigma^r(i_1) \neq \mathbf{m}_\sigma^r(j_1) = \mathbf{m}_\sigma^r \cdot \phi_2(j_1)$. Otherwise, if $j_1 = i_2$, then $\phi_1 = \phi(i_1, i_2)$ and $\phi_1 = \phi(i_2, j_2)$. Thus if $\mathbf{m}_\sigma^r \cdot \phi_1 = \mathbf{m}_\sigma^r \cdot \phi_2$ then $\phi_1 \in D_{i_1}(\mathbf{m}_\sigma^r)$, which implies that $\phi_1 \notin T_n \backslash D(\mathbf{m}_\sigma^r)$, a contradiction. □

Lemma 5.8 implies that we can calculate r-regular Ulam sphere sizes of radius 1 whenever we can calculate the appropriate duplication set. This calculation can be simplified by noting that for a sequence $\mathbf{m} \in \mathbb{Z}^n$ that $SD(\mathbf{m}) \cap AD(\mathbf{m}) = \varnothing$ (by the definition of $AD(\mathbf{m})$) and then decomposing the duplication set into these components. This idea is stated in Theorem 5.1 at the beginning of this section, which like Theorem 4.1, is a partial answer the the first main question of this chapter. We now have the machinery to prove Theorem 5.1.

proof of Theorem 5.1

Let $\mathbf{m}_\sigma^r \in \mathcal{M}_r(\mathbb{S}_n)$. By the definition of $D(\mathbf{m}_\sigma^r)$ and lemma 5.2,

$$\begin{aligned} & \{\mathbf{m}_\sigma^r \cdot \phi \in \mathcal{M}_r(\mathbb{S}_n) : \phi \in T_n \backslash D(\mathbf{m}_\sigma^r)\} \\ = \; & \{\mathbf{m}_\sigma^r \cdot \phi \in \mathcal{M}_r(\mathbb{S}_n) \; : \; \phi \in T_n \backslash SD(\mathbf{m}_\sigma^r)\} \\ = \; & S(\mathbf{m}_\sigma^r, 1). \end{aligned}$$

This implies $\# T_n \backslash D(\mathbf{m}_\sigma^r) \;\geq\; \# S(\mathbf{m}_\sigma^r, 1)$. By lemma 5.8, for $\phi_1, \phi_2 \in T_n \backslash D(\mathbf{m}_\sigma^r)$, if $\phi_1 \neq \phi_2$, then $\mathbf{m}_\sigma^r \cdot \phi_1 \neq \mathbf{m}_\sigma^r \cdot \phi_2$. Hence we have $\# T_n \backslash D(\mathbf{m}_\sigma^r) \;\leq\; \# S(\mathbf{m}_\sigma^r, 1)$, which implies that

$\#T_n \backslash D(\mathbf{m}^r_\sigma) = \#S(\mathbf{m}^r_\sigma, 1)$. It remains to show that $\#T_n \backslash D(\mathbf{m}^r_\sigma) = 1+(n-1)^2 - \#SD(\mathbf{m}^r_\sigma) - \#AD(\mathbf{m}^r_\sigma)$. This is an immediate consequence of the fact that $\#T_n = 1 + (n-1)^2$ and $SD(\mathbf{m}^r_\sigma) \cap AD(\mathbf{m}^r_\sigma) = \varnothing$. $\square$

Theorem 5.1 reduces the calculation of $\#S(\mathbf{m}^r_\sigma, 1)$ to calculating $\#SD(\mathbf{m}^r_\sigma)$ and $\#AD(\mathbf{m}^r_\sigma)$. It is an easy matter to calculate $\#SD(\mathbf{m}^r_\sigma)$, since it is exactly equal to $(n-2)$ times the number of $i \in [n]$ such that $\mathbf{m}^r_\sigma(i) = \mathbf{m}^r_\sigma(i-1)$ plus $(r-1)$ times the number of $i \in [n]$ such that $\mathbf{m}^r_\sigma(i) \neq \mathbf{m}^r_\sigma(i-1)$ or $i = 1$. We also showed how to calculate $\#AD(\mathbf{m})$ earlier. The next example is an application of Theorem 5.1

Example 5.9. *Suppose* $\mathbf{m}^3_\sigma = (1,1,1,2,3,2,3,2,4,4,3,4)$. *There are 3 values of* $i \in [12]$ *such that* $\mathbf{m}^3_\sigma(i) = \mathbf{m}^3_\sigma(i-1)$, *which implies that* $\#SD(\mathbf{m}^3_\sigma = (3)(12-2) + (12-3)(3-1) = 48$. *Meanwhile, by Lemmas 5.6 and 5.7,* $\#AD(\mathbf{m}^3_\sigma) = ((5-3)/2)((5-1)/2)) = 2$. *By Theorem 5.1,* $\#S(\mathbf{m}^3_\sigma), 1 = (12-1)^2 - 48 - 2 = 71$.

C. Min/Max Spheres and Code Size Bounds

In this section we show choices of center achieving minimum and maximum r-regular Ulam sphere sizes for the radius $t = 1$ case. As an application, we also state new upper and lower bounds on maximal code size in Lemmas 5.11, 5.13, and 5.14 (Lemmas 5.26 and 5.27 may also be included in this list, which are bounds in the special case when $n/r = 2$). These bounds represent the final main contribution of this chapter, answering the second main question.

The binary case, when $n/r = 2$, presents unique challenges because of the nature of its alternating duplication sets. In particular, the choice of center multipermutation yielding the maximal sphere size in the non-binary cases does not yield the maximal size in the binary case. Thus we divide this section into parts – the first subsection treating the non-binary case, and the remaining two subsections treating the binary case.

1) Non-Binary Case:

We begin by discussing the non-binary case in this subsection. The non-binary case is the general case where $n/r \neq 2$. Tight minimum and maximum values of sphere sizes are explicitly given.

We then discuss resulting bounds on code size. First let us consider the r-regular Ulam sphere of minimal size. The first two lemmas presented in this section apply to all cases, both non-binary and binary, while the remaining results only apply when $n/r \neq 2$.

Lemma 5.10. *Recall that* $n, r \in \mathbb{Z}_{>0}$ *and* $r|n$. *Let* $\mathbf{m}_\sigma^r \in \mathcal{M}_r(\mathbb{S}_n)$. *Then*

$$\#S(\mathbf{m}_e^r, 1) \quad \leq \quad \#S(\mathbf{m}_\sigma^r, 1).$$

Proof. Assume $\mathbf{m}_\sigma^r \in \mathcal{M}_r(\mathbb{S}_n)$. In the case that $n/r = 1$, then $\mathbf{m}_e^r = e$ and $\mathbf{m}_\sigma^r = \sigma$, so that $\#S(\mathbf{m}_e^r, 1) = \#S(\mathbf{m}_\sigma^r, 1)$. Therefore we may assume that $n/r \geq 2$. By Theorem 5.1, $\min\limits_{\sigma \in \mathbb{S}_n}(\#S(\mathbf{m}_\sigma^r, 1)) = 1 + (n-1)^2 - \max\limits_{\sigma \in \mathbb{S}_n}(\#SD(\mathbf{m}_\sigma^r) + \#AD(\mathbf{m}_\sigma^r))$. Since $n/r \geq 2$, we know that $n - 2 > r - 1$, which implies that for all $\sigma \in \mathbb{S}_n$, that $\#SD(\mathbf{m}_\sigma^r)$ is maximized by maximizing the number of integers $i \in [n]$ such that $\mathbf{m}_\sigma^r(i) = \mathbf{m}_\sigma^r(i-1)$. This is accomplished by choosing $\sigma = e$, and hence for all $\sigma \in \mathbb{S}_n$, we have $\#SD(\mathbf{m}_e^r) \geq \#SD(\mathbf{m}_\sigma^r)$.

We next will show that for any increase in the size of $\#AD(\mathbf{m}_\sigma^r)$ compared to $\#AD(\mathbf{m}_e^r)$, that $\#SD(\mathbf{m}_\sigma^r)$ is decreased by a larger value compared to $\#SD(\mathbf{m}_e^r)$, so that $(\#SD(\mathbf{m}_\sigma^r) + \#AD(\mathbf{m}_\sigma^r))$ is maximized when $\sigma = e$. By Lemmas 5.6 and 5.7, $\#AD(\mathbf{m}_\sigma^r)$ is characterized by the lengths of its locally maximal alternating substrings. For every locally maximal alternating substring $\mathbf{m}_\sigma^r[a, a+k-1]$ (here $a, k \in \mathbb{Z}_{>0}$) of $\mathbf{m}_\sigma^r$ of length k, there are at least $k-2$ fewer instances where $\mathbf{m}_\sigma^r = \mathbf{m}_\sigma^r(i-1)$ or $i = 1$ when compared to instances where $\mathbf{m}_e^r(i) = \mathbf{m}_e^r(i-1)$. This is because for all $i \in [a+1, a+k-1]$, $\mathbf{m}_\sigma^r(i) \neq \mathbf{m}_\sigma^r(i-1)$ and $i+1 \neq 1$. Hence for each locally maximal alternating substring $\mathbf{m}_\sigma^r[a, a+k-1]$, then $\#SD(\mathbf{m}_\sigma^r)$ is decreased by at least $(k-2)(n-2-(r-1)) \geq (k-2)(r-1)$ when compared to $\#SD(\mathbf{m}_e^r)$. Meanwhile, $\#AD(\mathbf{m}_\sigma^r)$ is increased by the same locally maximal alternating substring by at most $(k-2)((k-2)/4)$ by Lemma 5.7. However, since $k \leq 2r$, we have $(k-2)((k-2)/4) \leq (k-2)(r-1)/2$, which is of course less than $(k-2)(r-1)$. $\square$

Lemma 5.10, along with Proposition 4.4 implies that the r-regular Ulam sphere size of radius $t = 1$ is bounded (tightly) below by $(1 + (n-1)(n/r - 1))$. This in turn implies the following sphere-packing type upper bound on any single-error correcting code.

Lemma 5.11. *If C is a single-error correcting* $\mathsf{MPC}_\circ(n,r)$ *code, then*

$$\#C \ \le \ \frac{n!}{(r!)^{n/r}\left(1+(n-1)(n/r-1)\right)}.$$

Proof. Let C be a single-error correcting $\mathsf{MPC}_\circ(n,r)$ code. A standard sphere-packing bound argument implies that $\#C \le (n!)/((r!)^{n/r}(\min_{\sigma\in\mathbb{S}_n}\#S(\mathbf{m}^r_\sigma,1))$. The remainder of the proof follows from Proposition 4.4 and Lemma 5.10. □

We have seen that $\#S(\mathbf{m}^r_\sigma)$ is minimized when $\sigma = e$. We now discuss the choice of center yielding the maximal sphere size. Let $\omega \in \mathbb{S}_n$ be defined as follows: $\omega(i) := ((i-1) \mod (n/r))r + \lceil ir/n \rceil$ and $\omega := [\omega(1), \omega(2), \dots, \omega(n)]$. With this definition, for all $i \in [n]$, we have $\mathbf{m}^r_\omega(i) = i \mod (n/r)$ For example, if $r=3$ and $n=12$, then $\omega = [1,4,7,10,2,5,8,11,3,6,9,12]$ and $\mathbf{m}^r_\omega = (1,2,3,4,1,2,3,4,1,2,3,4)$. We can use Theorem 5.1 to calculate $\#S(\mathbf{m}^r_\omega,1)$, and then show that this is the maximal r-regular Ulam sphere size (except for the case when $n/r=2$).

Lemma 5.12. *Suppose* $n/r \ne 2$. *Then*

$$\#S(\mathbf{m}^r_\sigma,1) \ \le \ \#S(\mathbf{m}^r_\omega,1) \ = \ 1+(n-1)^2-(r-1)n.$$

Proof. Assume $n/r \ne 2$. First notice that if $n/r = 1$ then for any $\sigma \in \mathbb{S}_n$ (including $\sigma = \omega$), the sphere $S(\mathbf{m}^r_\sigma,1)$ contains exactly one element (the tuple of the form $(1,1,\dots,1)$). Hence the lemma holds trivially in this instance. Next, assume that $n/r > 2$. We will first prove that $\#S(\mathbf{m}^r_\omega,1) = 1+(n-1)^2-(r-1)n$.

Since $n/r > 2$, it is clear that $\mathbf{m}^r_\omega$ contains no alternating subsequences of length greater than 2. Thus by Lemma 5.3, $AD(\mathbf{m}^r_\omega) = \varnothing$ and therefore by Theorem 5.1, $\#S(\mathbf{m}^r_\omega,1) = 1+(n-1)^2-\#SD(\mathbf{m}^r_\omega)$. Since there does not exist $i\in[n]$ such that $\mathbf{m}^r_\omega(i) = \mathbf{m}^r_\omega(i-1)$, we have $\#SD(\mathbf{m}^r_\omega) = (r-1)n$, completing the proof of the first statement in the lemma.

We now prove that $\#S(\mathbf{m}^r_\sigma,1) \le \#S(\mathbf{m}^r_\omega,1)$. Recall that $\#SD(\mathbf{m}^r_\sigma)$ is equal to $(n-2)$ times the number of $i\in[n]$ such that $\mathbf{m}^r_\sigma(i) = \mathbf{m}^r_\sigma(i-1)$ plus $(r-1)$ times the number of $i\in[n]$ such that $\mathbf{m}^r_\sigma(i) \ne \mathbf{m}^r_\sigma(i-1)$. But $n/r>2$ implies that $r-1<n-2$, which implies $\min_{\mathbf{m}^r_\pi\in\mathcal{M}_r(\mathbb{S}_n)} \#SD(\mathbf{m}^r_\pi,1) = (r-1)n$. Therefore

$$\begin{aligned}
\#S(\mathbf{m}^r_\sigma, 1) &\leq 1 + (n-1)^2 - \min_{\mathbf{m}^r_\pi \in \mathcal{M}_r(\mathbb{S}_n)} \#SD(\mathbf{m}^r_\pi, 1) - \min_{\mathbf{m}^r_\pi \in \mathcal{M}_r(\mathbb{S}_n)} \#AD(\mathbf{m}^r_\pi, 1) \\
&\leq 1 + (n-1)^2 - \min_{\mathbf{m}^r_\pi \in \mathcal{M}_r(\mathbb{S}_n)} \#SD(\mathbf{m}^r_\pi, 1) \\
&= 1 + (n-1)^2 - (r-1)n \\
&= \#S(\mathbf{m}^r_\omega, 1).
\end{aligned}$$

□

The upper bound of lemma 5.12 implies a lower bound on a perfect single-error correcting MPC(n, r).

Lemma 5.13. *Suppose $n/r \neq 2$. If C is a perfect single-error correcting MPC(n, r), then*

$$\frac{n!}{(r!)^{n/r}((1 + (n-1)^2) - (r-1)n)} \quad \leq \quad \#C.$$

Proof. Assume $n/r \neq 2$, and that C is a perfect single-error correcting MPC(n, r). Then $\sum_{\mathbf{m}^r_c \in \mathcal{M}_r(C)} \#S(\mathbf{m}^r_c, 1) = (n!)/((r!)^{n/r})$. This means

$$\frac{n!}{(r!)^{n/r}} \quad \leq \quad (\#C) \cdot \left(\max_{\mathbf{m}^r_c \in \mathcal{M}_r(C)} (\#S(\mathbf{m}^r_c, 1)) \right),$$

which by Lemma 5.12 implies the desired result. □

A more general lower bound is easily obtained by applying Lemma 5.12 with a standard Gilbert-Varshamov bound argument. While the lower bound of Lemma 5.13 applies only to perfect codes that are MPC$_\circ(n, r, d)$ with $d \geq 3$, the next lemma applies to any MPC$_\circ(n, r, d)$, which may or may not be perfect.

Lemma 5.14. *Suppose $n/r \neq 2$, and let $C \subseteq \mathcal{M}_r(\mathbb{S}_n)$ be an MPC$_\circ(n, r, d)$ code of maximal cardinality. Then*

$$\frac{n!}{(r!)^{n/r}(1 + (n-1)^2 - (r-1)n)^{d-1}} \quad \leq \quad \#C$$

Proof. Assume that $n/r \neq 2$, and that C is an MPC$_\circ(n, r, d)$ code of maximal cardinality. For all $\mathbf{m}_\sigma^r \in \mathcal{M}_r(\mathbb{S}_n)$, there exists $c \in C$ such that $\mathrm{d}_\circ(\mathbf{m}_\sigma^r, c) \leq d-1$. Otherwise, we could add $\mathbf{m}_\sigma^r \notin C$ to C while maintaining a minimum distance of d, contradicting the assumption that $\#C$ is maximal.

Therefore $\bigcup_{\mathbf{m}_c^r \in \mathcal{M}_r(C)} S(\mathbf{m}_c^r, d-1) = \mathcal{M}_r(\mathbb{S}_n)$. This in turn implies that

$$\frac{n!}{(r!)^{n/r}} \quad \leq \quad \sum_{\mathbf{m}_c^r \in \mathcal{M}_r(C)} \#S(\mathbf{m}_c^r, d-1).$$

The right hand side of the above inequality is less than or equal to $(\#C) \cdot \left(\max_{\mathbf{m}_c^r \in \mathcal{M}_r(C)} \#S(\mathbf{m}_c^r, d-1) \right)$. Finally Lemma 5.12 implies that

$$\max_{\mathbf{m}_c^r \in \mathcal{M}_r(C)} (\#S(\mathbf{m}_c^r, d-1)) \quad \leq \quad (1 + (n-1)^2 - (r-1)n)^{d-1}$$

so the conclusion holds. □

2) Binary Case – Cut Location Maximizing Sphere Size:

In the previous subsection we were able to find center multipermutations whose sphere sizes were both minimal (Lemma 5.10) and maximal (Lemma 5.12). These were used to provide bounds on the maximum code size (Lemmas 5.11, 5.13, 5.14). However, a complication arises that prevents Lemma 5.12 from applying to the binary case, the case when $n/r = 2$. We say that $\mathbf{m}_\sigma^r \in \mathcal{M}_r(\mathbb{S}_n)$ is a **binary multipermutation** if and only if $n/r = 2$. The next two subsections focus on determining the maximum sphere size for binary multipermutations. The current subsection addresses the question of cut location. The notion of cuts is defined in the following paragraphs. For the remainder of the chapter we assume that n is an even integer and that $n/r = 2$ (equivalently $r = n/2$).

Since we are assuming that $n/r = 2$, by definition $\mathbf{m}_\omega^r$ is an n-length alternating string, which results in the size of the alternating duplication set $AD(\mathbf{m}_\omega^r)$ increasing rapidly as n increases. This in turn results in $\#S(\mathbf{m}_\omega^r, 1)$ no longer being maximal (in the sense of Lemma 5.12). For example, if $n = 12$, then we have $\mathbf{m}_\omega^r = (1,2,1,2,1,2,1,2,1,2,1,2,1,2)$, which would imply that $\#AD(\mathbf{m}_\omega^r) = \psi(12) = 25$.

To compensate for this problem, it is best to "cut" the original $\mathbf{m}_\omega^r$ into some number c of locally maximal alternating substrings. Whenever $\mathbf{m}$ is a tuple in two symbols, for example when $\mathbf{m} \in \{1,2\}^n$, we use the term **cut** to refer to any locally maximal alternating substring of $\mathbf{m}$. This language applies to binary multipermutations. Considering the example above when $n = 12$, we could instead take the binary multipermutation $(1,2,1,2,1,2,2,1,2,1,2,1)$, which has two cuts of length 6, namely $(1,2,1,2,1,2)$ and $(2,1,2,1,2,1)$ as opposed to a single length 12 cut in the original $\mathbf{m}^r_\omega$. Notice here that the standard duplication set increases by 5 but the new alternating duplication set size is now $\psi(6) + \psi(6) = 8$, a decrease of 17.

Intuitively, these cuts should be chosen so that each is as similar in length as possible in order to minimize the total size of the alternating duplication set. For example, $(\underbrace{1,2,1,2,1,2}_{\text{1st cut}},\underbrace{2,1,2,1,2,1}_{\text{2nd cut}})$, which has an alternating duplication set of size 8 is preferable to $(\underbrace{1,2}_{\text{1st cut}},\underbrace{2,1,2,1,2,1,2,1,2,1}_{\text{2nd cut}})$, which has an alternating duplication set of size 16. This idea is proven subsequently. Another question concerns the optimal number of such cuts, since each time a cut is introduced the standard duplication set size necessarily increases. This question is addressed in the next subsection, and it turns out that having approximately $\sqrt{r}$ cuts minimizes total duplications and thus results in the maximum sphere size.

To start this subsection, we will show that given a multipermutation with a fixed number c of cuts, the alternating duplication set is minimized when these cut lengths are as similar in length as possible. In order to simplify the argument, the following two lemmas reduce the discussion to the lengths of these alternating substrings.

Lemma 5.15. *Let $\mathbf{m} \in \{1,2\}^n$. Then there exists a binary multipermutation $\mathbf{m}^r_\sigma \in \mathcal{M}_r(\mathbb{S}_n)$ such that $\mathbf{m}^r_\sigma = \mathbf{m}$ if and only if $\#\{i \in [n] : \mathbf{m}(i) = 1\} = \#\{i \in [n] : \mathbf{m}(i) = 2\}$, i.e. the number of 1's and 2's of $\mathbf{m}$ are equal.*

Proof. Assume $\mathbf{m} \in \{1,2\}^n$. First suppose that there exists a binary multipermutation $\mathbf{m}^r_\sigma \in \mathbb{S}_n$ such that $\mathbf{m}^r_\sigma = \mathbf{m}$. Then by the definition of binary multipermutations, $\#\{i \in [n] : \mathbf{m}(i) = 1\} = r = \#\{i \in [n] : \mathbf{m}(i) = 2\}$, completing the first direction.

For the second direction of the proof, suppose that $\#\{i \in [n] : \mathbf{m}(i) = 1\} = \#\{i \in [n] : \mathbf{m}(i) = 2\} = n/2 = r$. Then we can construct a binary multipermutation $\mathbf{m}^r_\sigma$ with the property that $\mathbf{m}^r_\sigma = \mathbf{m}$ as follows: Define $\{i_1, i_2, \ldots, i_r\} := \{i \in [n] \ : \ \mathbf{m}(i) = 1\}$ and $\{i_{r+1}, i_{r+2}, \ldots, i_n\} := \{i \in [n] \ : \ \mathbf{m}(i) = 2\}$. For all $j \in [n]$, set $\sigma(i_j) := j$ and define $\sigma := (\sigma(1), \sigma(2), \ldots \sigma(n))$. Then $\mathbf{m}^r_\sigma = \mathbf{m}$. $\square$

Lemma 5.16. *Let $c \in [n-1]$, and $(q(1), q(2), \ldots, q(c)) \in \mathbb{Z}^c_{>0}$ such that $\sum_{i=1}^c q(i) = n$. Then there exists $i \in [c]$ such that $q(i)$ is even if and only if there exists a binary multipermutation $\mathbf{m}^r_\sigma \in \mathcal{M}_r(\mathbb{S}_n)$ such that*

$$\mathbf{m}^r_\sigma = (\underbrace{\mathbf{m}^r_\sigma[a_1, b_1]}_{q(i_1)}, \underbrace{\mathbf{m}^r_\sigma[a_2, b_2]}_{q(i_2)}, \ldots, \underbrace{\mathbf{m}^r_\sigma[a_c, b_c]}_{q(i_c)}),$$

where for all $i \in [c]$, $a_i, b_i \in [n]$, and $a_i \leq b_i$ such that $\mathbf{m}^r_\sigma[a_i, b_i]$ is a cut (locally maximal alternating substring).

The proof of Lemma 5.16 can be found in the appendices. In words, the lemma states that given any tuple of positive integers $(q(1), q(2), \ldots q(c))$ whose entries sum to n, as long as there is at least one even integer in the tuple, then the entries can be made to correspond to the lengths of the cuts of some binary mutlipermutation $\mathbf{m}^r_\sigma$. Notice that in the formulation resulting from Lemma 5.16, the number of cuts c in a binary multipermutation $\mathbf{m}^r_\sigma$ is one more than the number of repeated adjacent digits. In other words, $c = \#\{i \in [2, n] \ : \ \mathbf{m}^r_\sigma(i) = \mathbf{m}^r_\sigma(i-1)\} + 1$. Hence for a fixed number of cuts c, the standard duplication set size $\#SD(\mathbf{m}^r_\sigma)$ does not depend on the lengths of individual cuts.

On the other hand, the size of the alternating duplication set $\#AD(\mathbf{m}^r_\sigma)$ does depend on the lengths of the cuts. This means that if the number of cuts is fixed at c then by Lemma 5.7 and Lemma 5.16, finding the maximum sphere size equates to minimizing $\psi((q(1), q(2), \ldots, q(c))$, where $(q(1), q(2), \ldots q(c)) \in \mathbb{Z}^c_{>0}$ has at least one even entry and whose entries sum to n. We claim that the tuple defined next minimizes the sum in question.

Definition (q_c, rem_c, $\mathbf{q}_c$)**.** Let $c \in [n-1]$. Denote by $q_c \in \mathbb{Z}_{>0}$ and $rem_c \in \mathbb{Z}_{\geq 0}$ the unique quotient and remainder when n is divided by c, i.e. $c * q_c + rem_c = n$ where $rem_c < c$. Define

$$\mathbf{q}_c := \begin{cases} (q_c+1, \underbrace{q_c, \dots q_c}_{c-2}, q_c - 1) & \text{if } q_c \text{ is odd and } rem_c = 0 \\ (\underbrace{q_c+1, \dots q_c+1}_{rem_c}, \underbrace{q_c, \dots q_c}_{c-rem_c}) & \text{otherwise} \end{cases}$$

We also use the notation $\mathbf{q}_c = (\mathbf{q}_c(1), \dots \mathbf{q}_c(c)) \in \mathbb{Z}^c_{>0}$.

The above definition guarantees that two important conditions are satisfied: (1) the entries of $\mathbf{q}_c$ sum to n; and (2) there exists some $i \in [c]$ such that $\mathbf{q}_c(i)$ is even. These two conditions correspond with the conditions and statement of Lemma 5.16. Additionally, by definition, $\mathbf{q}_c$ is a weakly decreasing sequence with all entries being positive integers, and thus it is a partition of n.

Standard calculation (see Remark E.3 in Appendix D) indicates that if two cuts of a binary multipermutation differ by 2 or more, then the size of the alternating duplication set associated with that multipermutation can be reduced by bringing the length of those two cuts closer together. Generalizing over all the cuts in the multipermutation, we may minimize the alternating duplication set and hence maximize sphere size by choosing all cuts to be as similar in length as possible. Another way of saying that the cut sizes are as similar in length as possible is to say that the cut sizes are precisely the values of $\mathbf{q}_c$. The fact that cut sizes equaling the values of $\mathbf{q}_c$ minimizes the associated alternating duplication set size is stated in the next theorem.

Theorem 5.2. *Let $c \in [n-1]$. Then*

$$\min_{\mathbf{m}^r_\sigma \in \mathcal{M}^c_r(\mathbb{S}_n)} \#AD(\mathbf{m}^r_\sigma) \;=\; \psi(\mathbf{q}_c),$$

where $\mathcal{M}^c_r(\mathbb{S}_n) := \{\mathbf{m}^r_\pi \in \mathcal{M}_r(\mathbb{S}_n) \;:\; \#\{\mathbf{m}^r_\pi(i) = \mathbf{m}^r_\pi(i-1)\} + 1 = c\}$, i.e. $\mathcal{M}^c_r(\mathbb{S}_n)$ is the set of binary multipermutations with exactly c cuts.

Proof. Assume $c \in [n-1]$. Note first that by Lemma 5.16, there exists a binary multipermutation with exactly c cuts, whose cut lengths correspond to $\mathbf{q_c}$. Now let $(a(1), a(2), \dots, a(c)) \in \mathbb{Z}^c_{>0}$ such that $\sum_{i=1}^c a(i) = n$ and there exists $i \in [c]$ such that $a(i)$ is even. Again by Lemma 5.16, $(a(1), a(2), \dots a(c)$ corresponds to the cut lengths of an arbitrary binary multipermutation with exactly c cuts. Hence by Lemma 5.7 it suffices to show that $\psi(\mathbf{q}_c) \leq \psi((a(1), a(2), \ldots, a(c))$. We divide the remainder of the proof into two halves corresponding to the the split definition of $\mathbf{q}_c$.

First, suppose q_c is odd and $rem_c = 0$ so that $\mathbf{q}_c = (q_c + 1, q_c, \ldots, q_c, q_c - 1)$. Then since there exists $i \in [c]$ such that $a(i)$ is even, there must be distinct i' and j' in $[c]$ such that $a_{i'} = q_c + h_{i'}$ and $a_{j'} = q_c - h_{j'}$ where $h_{i'}, h_{j'} \in \mathbb{Z}_{>0}$. Hence, by Remark E.3 (see Appendix E),

$$\psi((\underbrace{q_c, q_c, \ldots, q_c}_{c})) + 1 \quad \leq \quad \psi((\underbrace{a(1), a(2), \ldots, a(c)}_{c})),$$

but also by Remark E.3 (applied to the first and last entry of $\mathbf{q}_c$),

$$\psi(\mathbf{q}_c) \quad = \quad \psi(q_c + 1) + \psi((\underbrace{q_c, q_c, \ldots, q_c}_{c-2})) + \psi(q_c - 1) \quad = \quad \psi((\underbrace{q_c, q_c, \ldots, q_c}_{c})) + 1.$$

For the second half, suppose that q_c is even or that $rem_c \neq 0$. Then $\mathbf{q}_c = (\underbrace{q_c + 1, \ldots, q_c + 1}_{rem_c}, \underbrace{q_c, \ldots, q_c}_{c - rem_c})$. This means that for all $i, j \in [c]$, that $|\mathbf{q}_c(i) - \mathbf{q}_c(j)| \leq 1$. Hence, by Remark E.3,

$$\psi(\mathbf{q}_c) \quad \leq \quad \psi(\underbrace{a(1), a(2), \ldots, a(c)}_{c})).$$

□

We have shown that choosing cuts to be as evenly distributed as possible results in minimizing the alternating duplication set. However, as mentioned before, while increasing cuts generally decreases the size of the alternating duplication set, it also increases the size of the standard duplication set. The question of the optimal number of cuts in a multipermutation $\mathbf{m}_\sigma^r$ minimizing $\#SD(\mathbf{m}_\sigma^r) + \#AD(\mathbf{m}_\sigma^r)$ remains.

D. binary case – number of cuts maximizing sphere size

The previous subsection demonstrated the nature of cuts maximizing sphere size in the binary case once the number of cuts c is fixed. This subsection focuses on determining the number of cuts maximizing binary multipermutation sphere size. Computer analysis for values of r up to $10,000$ suggests that $c \approx \sqrt{r}$ cuts minimizes the sum of $\#SD(\mathbf{m}_\sigma^r)$ and $\#AD(\mathbf{m}_\sigma^r)$ (and therefore maximizes the sphere size). The next remark and subsequent lemmas prove that this is indeed the case. We therefore call $\sqrt{r}$ the **ideal cut value** and use the notation $\hat{c} := \sqrt{r}$. In practice the actual optimal number of cuts is only approximately equal to $\hat{c}$ since $\hat{c}$ is not generally an integer. As in the previous subsection, recall that we assume n is a positve even integer and that $n/r = 2$ for the remainder of this chapter.

Remark 5.17. Let $\mathbf{m}^r_\sigma \in \mathcal{M}_r(\mathbb{S}_n)$ be a binary multipermutation. Then

$$\#SD(\mathbf{m}^r_\sigma) \;=\; (c-1)(n-2) + (n-(c-1))(r-1) \;=\; c(r-1) + (n-1)(r-1),$$

where $c := \#\{i \in [2,n] \;:\; \mathbf{m}^r_\sigma(i) \;=\; \mathbf{m}^r_\sigma(i-1)\} + 1$.

Note that the remark could technically be simplified by rewriting n as $2r$, but here and elsewhere n is kept in favor of $2r$ to retain intuition behind the meaning and for ease of comparison with previous results in the non-binary case. Although the remark is obvious, its significance is that the only component that depends upon c is $c(r-1)$. This means that each time the number of cuts is increased by 1, the size of the standard duplication set is increased by $r-1$.

Therefore to show that $\hat{c}$ cuts minimizes duplications, it is enough to show the following two facts: (1) if the number of cuts is greater than or equal to $\hat{c}$, then increasing the number of cuts by one causes a decrease in the alternating duplication set by at most $r-1$; and (2) if the number of cuts is less than or equal to $\hat{c}$, then a further decrease in cuts by one will enlarge the alternating duplication set by at least $r-1$. These two facts are expressed in the next two lemmas.

Lemma 5.18. *Let $c \in [n-2]$ and $\hat{c} \leq c$. Then*

$$\psi(\mathbf{q}_c) - \psi(\mathbf{q}_{c+1}) \quad \leq \quad r-1. \tag{8}$$

The proof for Lemma 5.18 is in the appendices. The next example demonstrates how to construct $\mathbf{q}_{c+1}$ from $\mathbf{q}_c$ when $\hat{c} \leq c < n-1$. This corresponds to increasing the number of cuts from c to $c+1$. Notice that $q_{c+1} > c$ and that each cut is decreased by at most 2, with some cuts decreased by only 1. This corresponds to the second case in the proof of Lemma 5.18.

Example 5.19. *Let $n = 30$ and $c = 4$. Notice that $\hat{c} = \sqrt{15} \approx 3.873$ so that $\hat{c} < c < n - 1$. We also have $q_4 = 7$ and $rem_4 = 2$ while $q_5 = 6$ and $rem_5 = 0$. Therefore $\mathbf{q}_4 = (8, 8, 7, 7)$ and $\mathbf{q}_5 = (6, 6, 6, 6, 6)$.*

We may visualize $\mathbf{q}_4$ and $\mathbf{q}_5$ respectively as the left and right diagrams in Figure 4, with the ith row of the diagram corresponding to the ith cut, $\mathbf{q}_4(i)$ or $\mathbf{q}_5(i)$. The numbers in the blocks in the left diagram of Figure 4 represent the order in which each row would be shortened to construct the last cut of $\mathbf{q}_4$.

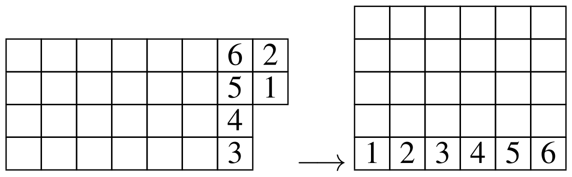


Fig. 4. Constructing $\mathbf{q}_5$ from $\mathbf{q}_4$ (when $n = 30$)

If $\mathbf{m}^r_\sigma$ is a multipermutation with four cuts whose lengths correspond to $\mathbf{q}_4$, then applying Remark 5.17 and Lemma 5.7, $\#SD(\mathbf{m}_\sigma) + \#AD(\mathbf{m}_\sigma) = 492$. By Theorem 5.1, this means $\#S(\mathbf{m}_\sigma, 1) = 238$. On the other hand, if $\mathbf{m}_\pi$ is a multipermutation with five cuts whose lengths correspond to $\mathbf{q}_5$, then similar methods show $\#SD(\mathbf{m}_\pi) + \#AD(\mathbf{m}_\pi) = 496$, which implies $\#S(\mathbf{m}_\pi, 1) = 234$, a smaller value.

Lemma 5.18 implied that if the number of cuts is greater or equal to $\hat{c}$, then increasing cuts shrinks the overall possible sphere size. The next lemma is analogous. It says that if the number of cuts is less than or equal to $\hat{c}$, then reducing the number of cuts shrinks the overall possible sphere size.

Lemma 5.20. *Let $c \in [n-1]$ and $c \leq \hat{c}$. Then*

$$\psi(\mathbf{q}_{c-1}) - \psi(\mathbf{q}_c) \quad > \quad r - 1. \tag{9}$$

The proof for Lemma 5.20 is in the appendices. The next example demonstrates how to construct $\mathbf{q}_c$ from $\mathbf{q}_{c-1}$ when $c \leq \hat{c}$. Notice that each cut length is decreased by at least 2.

Example 5.21. *Let $n = 34$ and $c = 4$. Notice that $\hat{c} = \sqrt{17} \approx 4.123$ so that $c < \hat{c}$. We also have $q_3 = 11$ and $rem_3 = 1$ while $q_4 = 8$ and $rem_4 = 2$. Therefore $\mathbf{q}_3 = (12, 11, 11)$ and $\mathbf{q}_4 = (9, 9, 8, 8)$. We can visualize $\mathbf{q}_3$ and $\mathbf{q}_4$ respectively as the left and right diagrams in Figure 5, with the ith row of each diagram corresponding to the ith cut, $\mathbf{q}_3(i)$ or $\mathbf{q}_4(i)$. The numbers in the blocks in the left diagram of Figure 5 represent the order in which each row would be shortened to construct the last cut of $\mathbf{q}_4$.*

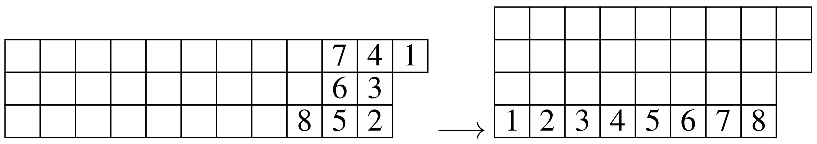


Fig. 5. Constructing $\mathbf{q}_4$ from $\mathbf{q}_3$ (when $n = 34$)

If $\mathbf{m}_\sigma$ is a multipermutation with three cuts whose lengths correspond to $\mathbf{q}_3$ above, then applying Remark 5.17 and Lemma 5.7, $\#SD(\mathbf{m}_\sigma) + \#AD(\mathbf{m}_\sigma) = 641$. By Theorem 5.1, this means $\#S(\mathbf{m}_\sigma, 1) = 449$. On the other hand, if $\mathbf{m}_\pi$ is a multipermutation with four cuts whose lengths correspond to $\mathbf{q}_4$ above, then similar methods show $\#SD(\mathbf{m}_\pi) + \#AD(\mathbf{m}_\pi) = 634$, which implies that $\#S(\mathbf{m}_\pi, 1) = 456$, a larger value.

Lemmas 5.18 and 5.20 imply that the number of cuts c minimizing the sum $\#SD(\mathbf{m}_\sigma^r) + \#AD(\mathbf{m}_\sigma^r)$ (and thus maximizing sphere size) observes the inequalities $\hat{c} - 1 < c < \hat{c} + 1$. This answers the question of the optimal number of cuts. For a particular value r, it is a relatively simple matter to calculate the exact size of the maximal Ulam multipermutation sphere. One simply has to determine whether $c = \lfloor \hat{c} \rfloor$ or $c = \lceil \hat{c} \rceil$ yields a smaller $\#SD(\mathbf{m}_\sigma^r) + \#AD(\mathbf{m}_\sigma^r)$ (here $\lceil x \rceil$ denotes the ceiling function on $x \in \mathbb{R}$, i.e. the least integer greater than or equal to x). Once the best choice for c is ascertained, an application of Theorem 5.1 will yield the maximum size for that particular r. The above statements are summarized in the next theorem.

Theorem 5.3.

$$\max_{\mathbf{m}_\sigma^r \in \mathcal{M}_r(\mathbb{S}_n)} \#S(\mathbf{m}_\sigma^r, 1) \ = \ 1 + (n-1)^2 - \min_{c \in \{\lfloor \hat{c} \rfloor, \lceil \hat{c} \rceil\}} \left(c(r-1) + (n-1)(r-1) + \psi(\mathbf{q}_c) \right). \tag{10}$$

Proof. The proof is an immediate consequence of Theorem 5.1, Theorem 5.2, Remark 5.17, and Lemmas 5.18 and 5.20, □

Once again we retain n instead of $2r$ in the theorem statement for intuition purposes. Theorem 5.3 indicates that the number of cuts c maximizing sphere size should be either $\lfloor \hat{c} \rfloor$ or $\lceil \hat{c} \rceil$, but it does not state when $\lfloor \hat{c} \rfloor$ or $\lceil \hat{c} \rceil$ is optimal. It turns out that whichever is closer to the true value of $\hat{c}$ will yield the maximal sphere size. That is, if $\hat{c} - \lfloor \hat{c} \rfloor \ \leq \ \lceil \hat{c} \rceil - \hat{c}$, then $\lfloor \hat{c} \rfloor$ (appropriately chosen) cuts will yield the maximal sphere size and visa versa. Stated another way, if $\hat{c} \ \leq \ \lfloor \hat{c} \rfloor + 0.5$, then $\lfloor \hat{c} \rfloor$ cuts provides the largest possible sphere size, but if $\hat{c} \ > \ \lfloor \hat{c} \rfloor + 0.5$, then $\lceil \hat{c} \rceil$ cuts provides the largest possible sphere size. To prove these facts, the next lemma is helpful.

Lemma 5.22. *Recall $r \in \mathbb{Z}_{>0}$ and $\hat{c} = \sqrt{r}$. We have $r \ \leq \ \lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor$ if and only if $\hat{c} \ \leq \ \lfloor \hat{c} \rfloor + 0.5$.*

Proof. We begin by showing that if $\hat{c} \ \leq \ \lfloor \hat{c} \rfloor + 0.5$, then $r \ \leq \ \lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor$. Assume that $\hat{c} \ \leq \ \lfloor \hat{c} \rfloor + 0.5$. Squaring both sides, we have $r \ \leq \ \lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor + 0.25$, which implies that $r \ \leq \ \lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor$.

Next we will show that if $r \ \leq \ \lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor$, then $\hat{c} \ \leq \ \lfloor \hat{c} \rfloor + 0.5$. We proceed by contrapositive. Suppose that $\hat{c} \ > \ \lfloor \hat{c} \rfloor + 0.5$. Squaring both sides, we have $r \ > \ \lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor + 0.25$, which implies that $r \ > \ \lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor$. □

Lemma 5.22 means that if $\hat{c}$ is closer to $\lfloor \hat{c} \rfloor$ than it is to $\lceil \hat{c} \rceil$, then the inequality $r \leq \lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor$ is satisfied. Otherwise, if $\hat{c}$ is closer to $\lceil \hat{c} \rceil$, then the opposite inequality, $r > \lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor$, is satisfied. Besides being useful to prove the following two lemmas, Lemma 5.22 in conjunction with the next two lemmas allows us to easily determine the number of cuts that will yield the maximal sphere size for each r. This is explained after the statement of the next lemma.

Lemma 5.23. $\hat{c} \leq \lfloor \hat{c} \rfloor + 0.5$ *if and only if*

$$\psi(\mathbf{q}_{\lfloor \hat{c} \rfloor}) - \psi(\mathbf{q}_{\lceil \hat{c} \rceil}) \quad \leq \quad r - 1 \tag{11}$$

The full proof can be found in the appendices. The crux of the argument is the same as that of Lemmas 5.18 and 5.20.

Lemma 5.23 characterizes precisely when $\lfloor \hat{c} \rfloor$ or $\lceil \hat{c} \rceil$ cuts is optimal for maximizing sphere size. The lemmas imply, as mentioned previously, that whichever of $\lfloor \hat{c} \rfloor$ and $\lceil \hat{c} \rceil$ is closer to $\hat{c}$ is the optimal cut value. However, also as mentioned previously, Lemma 5.22 allows us to easily determine which is optimal by simply looking at r. Notice that $\lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor$ is exactly half way between $\lfloor \hat{c} \rfloor^2$ and $\lfloor \hat{c} \rfloor^2 + 2\lfloor \hat{c} \rfloor = \lceil \hat{c} \rceil - 1$. Hence, Lemmas 5.22 and 5.23 imply that for r closer to $\lfloor \hat{c} \rfloor^2$, that $\lfloor \hat{c} \rfloor$ cuts are better, but for r closer to $\lceil \hat{c} \rceil^2$, that $\lceil \hat{c} \rceil$ cuts are better. For example, if $r = 11$, then $\lfloor \hat{c} \rfloor^2 = 9$ and $\lceil \hat{c} \rceil^2 = 16$. Since 11 is closer to 9, we know that $\lfloor \hat{c} \rfloor = 3$ cuts is optimal. Moreover, if $r \in \{9, 10, 11, 12\}$, then $\lfloor \hat{c} \rfloor = 3$ cuts is optimal, but if $r \in \{13, 14, 15, 16\}$, then $\lceil \hat{c} \rceil = 4$ cuts is optimal.

Returning to Theorem 5.3, we can also easily obtain an upper bound on maximum sphere size. This is shown in the next lemma and corollary.

Lemma 5.24.
Let $c \in [n-1]$. *Then*

$$\psi(\mathbf{q}_c) \quad \geq \quad c\left(\left(\frac{r}{c} - 1\right)^2 - \frac{1}{4}\right).$$

Proof. Suppose $c \in [n-1]$ and let $\mathbf{a} := (a(1), a(2), \ldots, a(c)) \in \mathbb{R}^c_{>0}$ such that $\sum_{i=1}^{c} a(i) = n$. Note first that

$$\sum_{i=1}^{c} \left(\frac{a(i) - 2}{2}\right)^2 \quad = \quad \sum_{i=1}^{c} \left(\frac{a(i)^2}{4} - a(i) + 1\right) \quad = \quad \frac{1}{4}\sum_{i=1}^{c} a(i)^2 - n + c. \tag{12}$$

Note also that by applying the Cauchy-Schwarz inequality to $\mathbf{a}$ and $\mathbf{1} \in \mathbb{R}^c$, the all-1 vector of length c, we obtain

$$\sum_{i=1}^{c} a(i)^2 \quad \geq \quad \frac{\sum_{i=1}^{c} a(i)^2}{c} \quad = \quad \frac{n^2}{c} \quad = \quad \sum_{i=1}^{c} \left(\frac{n}{c}\right)^2 \tag{13}$$

Equation (12) and inequality (13) imply that choosing $\mathbf{a} = (n/c, n/c, \ldots, n/c)$ minimizes the sum on the far left of Equation (12). The minimum of the left side of Equation (12) is less than or equal to $\sum_{i=1}^{c}((\mathbf{q}_c(i)-2)/2)^2$, with equality only holding when $n/c \in \mathbb{Z}$. Thus an application of Remark E.1 completes the proof. □

Corollary 5.25. *Let $\mathbf{m}_\sigma^r$ be a binary multipermutation. Also define*

$$U(r) \quad := \quad 1 + (n-1)^2 - \left((\hat{c}-1)(r-1) \; + \; (n-1)(r-1) + \Big(\hat{c}-1\Big) \left(\left(\frac{r}{\hat{c}+1} - 1 \right)^2 - \frac{1}{4} \right) \right).$$

Then

$$\#S(\mathbf{m}_\sigma^r, 1) \quad < \quad U(r).$$

Proof. The proof follows from Theorem 5.3, Lemma 5.24, and the fact that $\hat{c}-1 < \lfloor \hat{c} \rfloor \leq \lceil \hat{c} \rceil < \hat{c}+1$. □

The following table compares values from Corollary 5.25 versus the size of the actual largest multipermutation sphere for given values of r. The actual values of largest sphere sizes were calculated using Theorem 5.3.

As the table suggests, the estimated value of the maximum sphere size obtained by applying Corollary 5.25 is asymptotically good. By asymptotically good we mean that the ratio between the true maximum sphere size, $\max\limits_{\mathbf{m}_\sigma^r \in \mathcal{M}_r(\mathbb{S}_n)} \#S(\mathbf{m}_\sigma^r, 1)$, and the upper bound value, $U(r)$ from Corollary 5.25, approaches 1 as r approaches infinity. This can be confirmed by observing that

TABLE X
MAXIMUM SPHERE SIZE VERSES BOUNDED VALUE

r	Max sphere size (10)	Inequality (12)	ratio
10	148	~ 168	$\sim .8819$
100	$18,101$	$\sim 18,423$	$\sim .9825$
1000	$1,937,753$	$\sim 1,941,489$	$\sim .9981$

if

$$L(r) \ := \ 1+(n-1)^2 - \left((\hat{c}+1)(r-1) \ + \ (n-1)(r-1) + \Big(\hat{c}+1\Big)\left(\frac{r}{\hat{c}-1} - \frac{1}{2}\right)^2 \right),$$

then $L(r) \ < \max_{\mathbf{m}^r_\sigma \in \mathcal{M}_r(\mathbb{S}_n)} \#S(\mathbf{m}^r_\sigma, 1)$. After making this observation, the Squeeze Theorem can then be applied with $(\max_{\mathbf{m}^r_\sigma \in \mathcal{M}_r(\mathbb{S}_n)} \#S(\mathbf{m}^r_\sigma, 1))/U(r)$ being squeezed between $L(r)/U(r)$ and $U(r)/L(r)$. As before, in the definitions of both $U(r)$ and $L(r)$, we keep n in favor of $2r$.

Finally, Corollary 5.25 can be applied to establish a new lower bound on perfect single-error correcting MPC(n,r)'s in the binary case. It can also be applied to establish a new Gilbert Varshamov type lower bound. These two bounds are stated as the last two lemmas.

Lemma 5.26. *Let C be a perfect single-error correcting* MPC(n,r). *Also let $U(r)$ be defined as in Corollary 5.25. Then*

$$\frac{n!}{(r!)^2(U(r))} \ \leq \ \#C.$$

Proof outline. The proof follows from Corollary 5.25. □

Lemma 5.27. *Let C be an* MPC$_\circ(n,r,d)$. *Also let $U(r)$ be defined as in Corollary 5.25. Then*

$$\frac{n!}{(r!)^2(U(r))^{d-1}} \ \leq \ \#C$$

Proof outline. The proof follows from Corollary 5.25 and a standard Gilbert-Varshamov argument (see the proof of Lemma 5.14 for such an argument). □

E. Conclusion

This chapter considered multipermutation Ulam codes. Two questions were addressed. The first question of calculating r-regular Ulam sphere sizes was addressed for the cases when the center is $\mathbf{m}_e^r$ or when the radius $t = 1$. This lead to new upper and lower bounds on maximal code size, providing an answer to the second question. These new results are summarized in the Tables VIII and IX, found in the introduction.

6. Formalization of Coding Theory in Lean

A. Introduction

In 1994, the infamous Pentium FDIV computer bug was discovered in Intel Pentium processors, eventually costing the company hundreds of millions of dollars. More recently, in May 2015, a bug was fixed in a prominent software package used for functional magnetic resonance imaging (fMRI). The existence of the bug, which was previously unnoticed, potentially invalidates 15 years of brain research. Underlying this type of problem is a need for mathematical verification. This is particularly true in the field of coding theory, which plays a major role in any form of digital communication. When a code that is utilized fails to perform its purported function to the degree of accuracy expected, errors in communication can occur. These errors can be especially serious if sensitive/vital information is sent or if errors occur within heavily relied-upon computer systems.

At the same time, advancements of communication technology often call for improved error-correcting code schemes. The complete testing of these codes can be costly and time-consuming. Hence competition between companies can lead to the utilization of codes before they are fully reliable. Moreover, the study of coding theory is often complex and proofs of the properties of a particular error-correcting code may be difficult to understand or verify for non-specialists in coding theory. Even among coding theorists, the vast quantity of coding schemes can sometimes lead to miscommunication.

Although in practice it may not always be necessary for every aspect of a particular error-correcting code to be rigorously proven in order for it to be implemented, formally proving correctness of codes confirms the theoretical foundations of the code and bolsters reliability. This is especially important for newer codes. In the case of well-established codes with a long history, their mathematical foundations have largely been established through numerous publications and scrutiny in the mathematical community. On the other hand, coding theory is an active area of research with new schemes being proposed every year. These proposals will not have had the same benefit of prolonged scrutiny. It is also possible that private companies use codes whose details are not publicly available, but whose verification is no less important. Mathematical formalization may help to solve these issues.

Mathematical formalization, or formal verification, is the precise statement of mathematical theorems in a specified language in such a way that the veracity of these theorems may be verified algorithmically. This verification may be carried out by computerized proof-assistants such as the well-known HOL, Coq or Isabelle. These (and other) proof-assistants allow for interactive theorem proving where a human interacts with a computer toward formalization with the computer confirming the logical soundness of each input. When theorems of coding theory are formalized, the mathematical soundness is simultaneously established. In fact, the formalization of coding theory theorems achieves the most rigorous of mathematical standards, beyond that of normal paper-and-pencil proofs.

This chapter focuses on work done toward formalizing theorems of coding theory in the Lean Theorem Prover [7, 55]. One of our main contributions is the introduction of "error correcting system" structures, which provide templates and systematic methods for formally defining an error-correcting code and certain basic properties of the code in Lean. To the best of the authors' knowledge, there has not been such an attempt to provide a systematic structure for formalizing a code. Such structures may also be augmented to state and verify properties about future error-correcting codes. Our current results include early examples of using these structures to formalize repetition codes and the Hamming (7,4) code.

Another contribution of this chapter is the formalization of definitions and lemmas concerning the Levenshtein distance and insertion/deletion correcting spheres and codes. In particular, we define inductive versions of deletion, insertion, and edit spheres. We then formalize a lemma relating edit spheres and the Levenshtein distance. The definitions and lemmas will aid formalization in Lean moving forward.

The organization of this chapter is as follows. In Section 6-B, we introduce Lean as a theorem prover and explain reasons for choosing Lean. In Section 6-C, we explain how to access our library and give an overview of its contents. Section 6-D introduces our formalization of error correcting systems and provides examples of how to formalize a code. Sections 6-E and 6-F explain our formalization of definitions and lemmas concerning the Levenshtein distance and insertion/deletion spheres, topics in insertion/deletion correcting codes.

B. About Lean

In this section we briefly discuss the Lean Theorem Prover and reasons for using Lean to formalize coding theory. As previously mentioned, Lean was released in 2015 and is still in the process of further development by Microsoft Research and Carnegie Mellon University. According to developers, Lean "aims to bridge the gap between interactive and automated theorem proving, by situating automated tools and methods in a framework that supports user interaction and the construction of fully specified axiomatic proofs" [55]. In other words, Lean combines strengths of both interactive and automated theorem proving.

In interactive theorem proving, the focus is on verifying proofs. The strength of interactive theorem proving is the soundness of proofs, as each step is justified by appealing to axioms or rules. The weakness is that this can sometimes require large amounts of input and interaction from the user. In automated theorem proving, the focus is on a program's ability to find proofs. The strength here is power and efficiency. Unfortunately, this is sometimes at the cost of absolute reliability. Lean combines both reliability and efficiency. Lean maintains reliability because of a relatively small and trusted kernel of axioms and inference rules based on dependent type theory [55]. At the same time, Lean utilizes a powerful elaborator that is able to infer much information such as type classes, coercions, overloading, etc.

Besides power and reliability, the flexibility of Lean makes it attractive for use in formalizing coding theory. Lean supports a variant of the Calculus of Inductive Constructions (CIC), itself a variant of the Calculus of Constructions (CoC), the strongest of the lambda calculi. This allows for flexible formalization of various mathematical notions.

Lean is also attractive for formalizing coding theory because of the ease of access and use. An Emacs version Lean is available free online. For newcomers to formalization, an in-depth tutorial is also available [7], allowing readers to immediately begin formalization in coding theory.

C. File access and overview

Our library, *Cotoleta*, can be downloaded from [36]. There are two .zip files, one for Lean 2 and one for Lean 3.3. The first file, *Cotoleta*, contains the formalization of Repetition codes and Hamming codes, as well as error correcting systems and other helpful mathematical lemmas. The second file, *CotoletaInsDel* contains formalization for insertion/deletion correcting codes and related definitions and lemmas. Once the .zip files are downloaded and files are decompressed, the .lean files should be placed in the Lean emacs search path. In the emacs version, this is accomplished, for example, by placing files in the folder emacs/Lean/library that is included with the Lean emacs installation. Files from *Cotoleta* should be compiled in the order of their dependencies, beginning with *binaryField.lean* and *ECCsystems.lean*. The file dependencies are depicted in Figure 1 below. Each file is a ".lean" file, but the suffix is omitted in the figure for brevity.

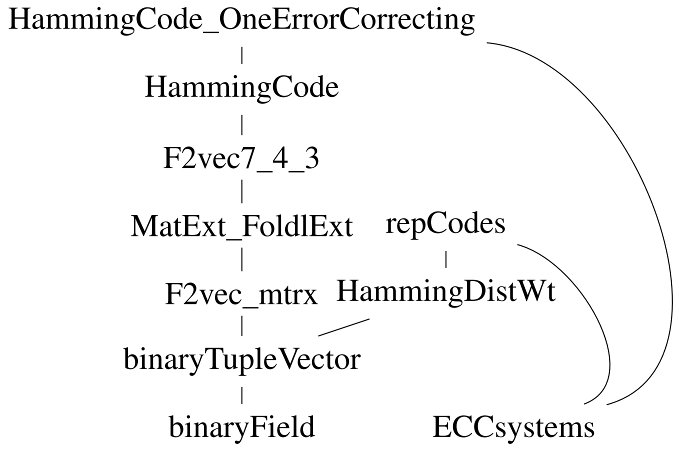


Fig. 6. File dependencies

The first file, *binaryField.lean*, defines the binary field $\mathbb{F}_2$ as a type named `F2` and provides other related definitions and theorems. In addition, the file contains definitions and lemmas for proving that a binary set endowed with the proper addition and multiplication operations is isomorphic to $\mathbb{F}_2$. As an example, the file contains a proof that the boolean values "`tt`" (true) and "`ff`" (false) with the "`band`" (and) and "`bxor`" (excluded or) operations is isomorphic to `F2`. The file *binaryTupleVector.lean*, provides definitions and theorems concerning binary vectors $v \in \mathbb{F}_2^n$ (where $n \in \mathbb{N}$), ubiquitously used in coding theory. The file defines the type of such vectors as follows:

```
definition F2TVec (n : ℕ) := tuple F2 n
```

This definition takes advantage of definitions and lemmas for the type "`tuple`" in Lean's library. A member of `(F2TVec n)` is comprised of two components: an ordered list of values from `F2` and a proof that the length of this list is equal to the natural number `n`. Members of `(F2TVec n)` can be written and displayed explicitly and intuitively. This type is especially convenient for examples or for use in conjunction with Lean's "`eval`" command, which evaluates and simplifies expressions.

However, a shortcoming of the type `(F2TVec n)` is that its members cannot immediately be used with Lean's standard definition of matrices and their associated operations and theorems. The file *F2vec_mtrx.lean* provides the following alternate definition for binary vectors as column matrices:

```
definition F2MVec (n : ℕ) := matrix F2 1 n
```

Members of `(F2MVec n)` may be multiplied with parity check matrices (a type of matrix commonly used in coding theory) by using the standard matrix operations provided in Lean's library. The file *F2vec_mtrx.lean* also provides functions to transform members of `(F2TVec n)` into members of `(F2MVec n)` and vice versa. These functions are proven to be inverses of each other. A function is also provided so that a matrix may be written explicitly as a tuple of members of `(F2TVec n)` and then converted to a matrix according to Lean's standard definition. This allows for explicitly writing a parity check or generation matrix.

One property of matrices necessary for our formalization that was not found in the original Lean matrix library is proven in the file *MatExt_FoldlExt.lean*. The property is the multiplicative transpose property of matrices, that given matrices A and B, we have $(AB)^T = B^T A^T$. In addition, the file contains lemmas concerning the underlying function, `foldl`, used in Lean's definition of matrix multiplication. These lemmas should aid future proofs involving this function.

The file *HammingDistWt.lean* defines the Hamming weight, `wtH v` of a member `(v : F2TVec n)`. The file also defines the Hamming distance, `dH(x, y)` between members `(x y : F2TVec n)`. Various lemmas and theorems are also proven, including the distance axioms establishing `dH` as a metric. These definitions and lemmas are utilized in the formalization of repetition codes and the Hamming (7,4) code.

The remaining files are *ECCsystems.lean*, *repCodes.lean*, *F2vec7_4_3.lean*, *HammingCode.lean*, and *HammingCode_OneErrorCorrecting.lean*. First, *ECCsystems.lean* provides frameworks for formalizing general error correcting codes. Second, the file *repCodes.lean* provides the formalization of repetition codes. Finally, the remaining files, *F2vec7_4_3.lean*, *HammingCode.lean*, and *HammingCode_OneErrorCorrecting.lean*, formalize the Hamming (7,4) code and some of its properties. These files are discussed in greater detail in section Section 6-D.

One advantage of our current formalization is the use of Lean's `calc` environment in writing proofs. The calc environment allows for a chain of equalities to be written in a natural and intuitive manner. For example, the following theorem and proof from the *HammingDistWt.lean* file proves the symmetric property of the Hamming distance:

```
theorem hammingDist_symm {n : ℕ}
(x y : F2TVec n) : dH(x, y) = dH(y, x) :=
calc dH(x,y) = wtH(x + y) : hammingDistWtTest
         ... = wtH(y + x) : addF2TVec_com
         ... = dH(y, x) : hammingDistWtTest
```

Note how the `calc` environment enables us to write versions of proof that mimic their paper and pencil counterparts. Another unique strength of our current library is the possibility for defining error-correcting codes as a `structure`. The `structure` command is unique to Lean. It enables users to define a structure containing many fields, as well as a built-in mechanism for obtaining these individual fields from any member of the defined structure. These fields can be thought of as projections for a structure member. For example, in the aforementioned *binaryField.lean* file, we define the structure F2_ISO as follows:

```
structure F2_ISO [class] (carrier : Type) : Type := mk ::
 (mapF2_C : F2 → carrier)
 (mapC_F2 : carrier → F2)
 (add : carrier → carrier → carrier)
 (mul : carrier → carrier → carrier)
 (cancel_F2_C_F2 : ∀ (x : F2),
  mapC_F2 (mapF2_C x) = x)
 (cancel_C_F2_C : ∀ (x : carrier),
  mapF2_C (mapC_F2 x) = x)
 (Hom_add : ∀ (x y : F2),
  add (mapF2_C x)(mapF2_C y) = mapF2_C (x + y))
 (Hom_mul : ∀ (x y : F2),
  mul (mapF2_C x)(mapF2_C y) = mapF2_C (x * y))
```

In other words, a member of F2_ISO carrier contains eight different fields. Providing an instance of F2_ISO carrier for a particular carrier involves instantiating each of these fields. Providing such an instance equates to proving that carrier is isomorphic to F2. Furthermore, given a member (mem : F2_ISO carrier), any of the individual fields can be extracted using the syntax "F2_ISO.⟨field⟩." For instance, F2_ISO.mapF2_C carrier is the mapping from F2 → carrier provided during the instantiation of mem. Much more detailed information on the structure command can be found in chapter 10 of the Lean tutorial [7]. We will discuss the use of the structure command to systematically formalize error-correcting codes along with the repetition code and Hamming (7,4) code examples in the second half of the following section.

Similarly to lean files in the first zip file, *Cotoleta*, files in the second zip file, *CotoletaInsDel*, should be placed in Leans search path and then compiled in the order of their dependencies. The order is as follows: 1) *Subseqnece.lean*, 2) *LongestCommon.lean* 3) *LevenshteinDist.lean*, 4) *Supersequence.lean*, 5) *InsDel.lean* 6) *InsDelSpheres.lean*, and 7) *InsDelCodes.lean*. These files are compatible with the current version of Lean, Lean 3.3.0, and bugs may occur if attempted to run with older or newer versions.

In the first file, *Subsequence.lean*, the notions of a subsequence, a common subsequence, and a longest common subsequence are defined, along with related lemmas. In standard mathematical language terms, we may define a subsequence as follows. Let Σ be an arbitrary set (alphabet), and $X = (x_1, x_2, \ldots, x_m)$ and $Y = (y_1, y_2, \ldots y_n)$ tuples in Σ^m and Σ^n respectively (here m and n are positive integers). We say that X is a subsequence of Y if and only if $X = (y_{i_1}, y_{i_2}, \ldots y_{i_m})$, where $1 \leq i_1 < i_2 < \cdots \leq n$. In the language of Lean, a subsequence is defined in *Subsequence.lean* as follows:

```
def subseq: list α → list α → Prop
| [] _ := true
| (x :: X) [] := false
| (x :: X) (y :: Y) := ((x=y) ∧ (subseq X Y))
                        ∨ (subseq (x :: X) Y)
```

The definition can be thought of as a function. The first line gives it a name “subseq” and states that the defined function takes two lists (essentially tuples, although the length is not indicated) over some arbitrary alphabet (technically a “Type”) α, and returns a proposition. The next two lines go on to say that an empty list is always a subset of any other list, and that any non-empty list is never a subset of the empty list. The final line, which is split into two in this paper because of space constraints, states that whenever new elements `x` and `y` are added to the head of their respective lists `X` and `Y`, then the new list `(x :: X)` (this is notation for the list obtained when `x` is inserted at the head of list `X`) is a subsequence of `(y :: Y)` whenever `x = y` and `X` is a subsequence of `Y`, or else `x :: X` is a subsequence of `Y`. In this way, a subsequence is inductively defined within Lean. Since completing our formalization, it has come to our attention that a similar definition, `sublist`, exists within the current Lean library. An interesting subquestion would be to formalize an equivalence between the definitions.

The remainder of the file *Subsequence.lean* states three more important definitions, `com_subseq`, `longest_common`, and `lcs_naive`. The first, `com_subseq`, is the definition of a common subsequence, and simply states that a list `V` over α is a common subsequence of two other lists `X` and `Y` whenever it is a subsequence of both `X` and `Y` individually. The next definition, `longest_common` defines a longest common subsequence, and states that a list `V` is a longest common subsequence of `X` and `Y` if it is first a common subsequence of `X` and `Y`, and then if for any other common subsequence `W` of `X` and `Y`, the length of `V` is greater or equal to the length of `W`. These definitions take as inputs respectively two or three lists and return a proposition that is either true or false.

The last important definition, `lcs_naive`, is another inductive definition that algorithmically defines Included in the file is the definition of `lcs_naive`, which is a function that takes two tuples (technically elements of type "list" in Lean) over an arbitrary alphabet (say ℕ), and returns a particular longest common subsequence of the two input tuples (lists). In the definition name, "lcs" is short for longest common subsequence, and "naive" is added because the definition follows a naive algorithm for finding a longest common subsequence, rather than known dynamic programming algorithms that boast superior computational cost. In our formalization the naive approach is preferred because of the relative ease with which other proofs can be carried out with this definition, and because for the purpose of formalization we are not concerned with extremely long lists where computation cost is a significant issue. However, an interesting side question would be to define a dynamic programming based longest common subsequence within Lean and prove that the length is equal to that of `lcs_naive`.

The next file, *LongestCommon.lean* primarily proves the theorem that `lcs_naive`, defined in the first file, is a longest common subsequence. In the notation of Lean, this theorem is stated as follows:

```
theorem lcs_naive_longest_com (X Y : list ℕ) :
  longest_common (lcs_naive X Y) X Y
```

In words, the theorem states that given two arbitrary lists `X` and `Y` over $\mathbb{N}$, that the list `(lcs_naive X Y)` satisfies the definition of being a longest common subsequence of `X` and `Y`. Note that here and in future definitions and lemmas, the α of the `subseq` definition in *Subsequence.lean* is often replaced by $\mathbb{N}$, the Type of natural numbers. This is because $\mathbb{N}$ provides some additional structure within Lean that makes proving theorems easier, while still maintaining enough arbitrariness for the purposes of insertion/deletion codes. It should also be noted that we consider 0 to belong to $\mathbb{N}$. *LongestCommon.lean* also contains other lemmas related to longest common subsequences.

After this, the file *LevenshteinDist.lean* defines the Levenshtein distance. This file is explained in greater detail in Section 6-E. The file *Supersequence.lean* defines supersequences in a manner similar to subsequences in the first file. It also includes lemmas necessary for proving an equivalence between different types of error-correcting codes. *InsDel.lean*, and *InsDelSpheres.lean*, provide definitions and lemmas about insertions and deletions, and about insertion, deletion, and edit spheres. These are expounded upon in Section 6-F. The last file, *InsDelCodes* defines insertion, deletion, and insertion/deletion (edit) correcting codes. It includes formal proofs of the fact that all three of these are equivalent given a fixed codeword length.

D. Error correcting systems definition and examples

A t-error correcting code may be formalized by defining a few specific components and related theorems. In particular, it is possible to formalize a t-error correcting code by first deciding upon a code length $n \in \mathbb{N}$, a message set M, a code alphabet Σ (for example binary or ternary digits), and a receivable alphabet $\Sigma^* \supseteq \Sigma$ (appropriate for the channel). Next, an encoder function $\mathrm{enc} : M \to \Sigma^n$ and a decoder function $\mathrm{dec} : (\Sigma^*)^n \to M$ are defined. Finally an error-evaluator function $\mathrm{d} : \Sigma^n \times (\Sigma^*)^n :\to \mathbb{N}$ is defined and the t-error correcting property, that for any message $m \in M$ and received word $y \in (\Sigma^*)^n$, if $\mathrm{d}(\mathrm{enc}(m), y) \leq t$, then $\mathrm{dec}(y) = m$ should be proven.

This schematic is reflected in the `structure ErrorCorrectingSystemOfType1` definition below, which may be found in the *ECCsystems.lean* file. We later define another similar structure in the same file that separates the decoding process into two steps, which may be beneficial for certain decoding algorithms.

```
structure ErrorCorrectingSystemOfType1 [class]
mk :: (t : ℕ) (leng : ℕ)
 (message_alphabet: Type)
 (code_alphabet: Type)
 (receivable_alphabet: Type)
 (Encoder: message_alphabet →
   tuple (code_alphabet) leng)
 (Decoder: tuple (receivable_alphabet) leng →
   message_alphabet)
 (ErrorEvaluator: tuple (code_alphabet) leng →
   tuple (receivable_alphabet) leng → ℕ)
 (tErrorCorrectingCode: ∀ (m: message_alphabet)
   (y: tuple (receivable_alphabet) leng),
   ErrorEvaluator (Encoder m) y ≤ t
   → m = Decoder y)
```

In the above structure definition, there are nine fields: `t`, `leng`, `message_alphabet`, `code_alphabet`, `receivable_alphabet`, `Encoder`, `Decoder`, `ErrorEvaluator`, and `tErrorCorrectingCode`. Each field corresponds to a component in the schematic described earlier. The file *repCodes.lean* contains the formalization of the well-known repetition codes. The following scripts provides a simple example that uses our structure definition to establish that `repCode` is a $\lfloor$`(len-1)/2`$\rfloor$ error-correcting code of length `len` with the message alphabet, code alphabet, and receivable alphabet all being `F2`.

```
variable len : ℕ
variable len_gt_0 : len > 0
```

```
definition repCode [trans_instance] :=
{|ErrorCorrectingSystemOfType1
 t := ((len-1)/2),
 leng := len,
 message_alphabet := F2,
 code_alphabet := F2,
 receivable_alphabet := F2,
 Encoder := repEnc len,
 Decoder := repDec,
 ErrorEvaluator := hammingDistance,
 tErrorCorrectingCode := reptErrCorr len_gt_0 |}
```

Before the repCode definition, the `variable` command is used to declare an arbitrary natural number `len` and a proof `len_gt_0` that `len` is greater than 0. These act as assumptions in the definition that follows. In the definition, each instantiation for the last four fields is given by another definition or theorem in the same file. For example, `repEnc` is defined as follows:

```
definition repEnc (n : ℕ) (m : F2) :   F2TVec n := [[ m ]]^n
```

In other words, `repEnc` is a function that takes a natural number `n` (in the case of the `repCode` definition that would be `len`) and a member `m` of `F2`, and then returns a member of `F2TVec n` by essentially taking `n` repetitions of `m`.

The Hamming (7,4) code is similarly formalized using our structure definition. First, the file *F2vec7_4_3.lean* provides some basic definitions and lemmas for the special cases of vectors of length $7, 4$, and 3. It also sets up some simple notions for use in Hamming codes, such as a mapping from `F2TVec 3` to ℕ. Next, the file *HammingCode.lean* defines the generator matrix `G` and parity-check matrix `H` for the Hamming (7,4) code, as well as encoder and decoder functions `Enc` and `Dec`. Some other properties are also proved, such as the fact that `ImEnc`, the image of `G`, is equivalent to `CodeSpace`, defined as the kernel of H. Finally, the file *HammingCode_OneErrorCorrecting.lean* establishes the single-error correcting capability of the Hamming code. These three files culminate in the following formalization at the end of *HammingCode_OneErrorCorrecting.lean* establishing the Hamming (7,4) code as a single-error correcting code:

```
definition HammingCodeOfType1 [trans_instance] :=
{| ErrorCorrectingSystemOfType1
 t := 1,
 leng := 7,
 message_alphabet := F2TVec 4,
 code_alphabet := F2,
 receivable_alphabet := F2,
 Encoder := Enc,
 Decoder := DecEC,
 ErrorEvaluator := hammingDistance,
 tErrorCorrectingCode :=  HammingCode_OneErrorCorrectableOfType1 |}
```

It is worth mentioning a little more detail about the above formalization. For instance, the encoder function `Enc` is defined as multiplication of members of `F2TVec 4` with the appropriate generator matrix. To write the generator matrix we make use of the function `Tvecs_to_mtrx` from the file *F2vec_mtrx.lean*, which allows us to explicitly write sufficiently small matrices. In *HammingCode.lean*, we define the generator matrix `G` and encoder `Enc` as follows:

```
definition G : matrix F2 4 7 :=
Tvecs_to_mtrx (of_list ([
(tag (-[I, I, I, O, O, O, O]) rfl),
(tag (-[I, O, O, I, I, O, O]) rfl),
(tag (-[O, I, O, I, O, I, O]) rfl),
(tag (-[I, I, O, I, O, O, I]) rfl)]))

definition Enc (v : F2TVec 4) : F2TVec 7 :=
F2MVec_to_F2TVec (v × G)
```

In the `G` definition, we use our own notation "`-[ ]`", which allows us to write an n-length list in the natural order with the ith element in the ith position (from the left), where $i \in \{0, 1, \ldots, n-1\}$. In the `Enc` definition above, the coercion of `v : F2TVec 4` to a member of `F2MVec 4` before being multiplied with `G` is hidden in the notation "$\times$". The other definitions and theorems used to instantiate each field of `HammingCodeOfType1` are found in the files *HammingCode.lean* and *HammingCode_OneErrorCorrecting.lean* as well.

The use of structures as templates for formalizing codes has another advantage. Lean provides a convenient mechanism that will make extending structure definitions a simple matter in the future. A structure definition can be extended to include all of the fields of the structure it extends from, with additional fields being declared (see chapter 10 of the Lean tutorial). In this way, something like a template for linear codes can be easily created by extending `structure ErrorCorrectingSystemOfType1` and adding the additional stipulation of closure under linear combinations. Here a member of a linear code defined in this manner will also automatically be coercible into a member of the parent structure.

E. Formalization of Levenshtein distance

In the file *LevenshteinDist.lean*, we define a distance function `LDist` between lists and prove that the defined function satisfies three of the four metric axioms. The final axiom is proven in *insDelSpheres.lean*. LDist stands for Levenshtein distance, and is defined in Lean as follows:

```
def LDist (X Y : list ℕ) := (length X) + (length Y) - (length_lcs_naive X Y)
                                                    - (length_lcs_naive X Y)
```

This definition is based on one of the standard ways that the Levenshtein distance $\ell_D(X,Y)$ between tuples X and Y may be defined, as the sum of the lengths of X and Y subtracted by two times the length of a longest common subsequence of X and Y ([56]). Note that in the Lean implementation, rather than subtract twice the length of a longest common subsequence, we instead subtract the length of a longest subsequence two times. This small adjustment helps to make formalization of lemmas or theorems involving `LDist` easier.

In Lean's notation, `LDist X Y` denotes the Levenshtein distance between `X` and `Y`. The first three metric axioms for `LDist`, non-negativity, identity of indiscernibles, and symmetry are stated and proved in *LevenshteinDist.lean*. The statements are as follows:

```
theorem LDist_non_neg (X Y : list ℕ) :
  LDist X Y ≥ 0

theorem LDist_zero_iff_eq (X Y : list ℕ) :
  LDist X Y = 0 ↔ X = Y

theorem LDist_symm (X Y : list ℕ) :
  LDist X Y = LDist Y X
```

In the statement of `LDist_zero_iff_eq`, the symbol ↔ is Lean's notation for "if and only if." Several other useful lemmas are also proven in *LevenshteinDist.lean*, such as the fact that if `x :: X` is a subsequence of `y :: Y`, then `X` is also a subsequence of `Y`. Recall that `x :: X` is notation for the list obtained when `x` is inserted at the head of list `X`. The first three metric axioms can be shown directly, but it is not obvious how to prove directly from the definition of `LDist` that the triangle inequality holds. Instead, we take advantage of an alternative view of the Levenshtein distance in order to prove the triangle inequality.

It is known that the Levenshtein distance $\ell_D(X, Y)$ between two tuples X and Y is equivalent to the least number of insertions or deletions (edits) necessary to transform X into Y. From this definition, it is obvious that the triangle inequality holds. To see why this is the case, suppose that X, Y, and Z are tuples such that the Levenshtein distance between X and Y and between Y and Z are $k_1 \in \mathbb{N}$ and $k_2 \in \mathbb{N}$ respectively. Then X can be transformed into Y with k_1 edits and Y can be transformed into Z with k_2 edits. Thus clearly X can also be transformed into Z in at most $k_1 + k_2$ edits, since X can first be transformed into Y and then into Z. This implies that the Levenshtein distance between X and Z is at most $k_1 + k_2$.

Our actual formal proof of the triangle inequality in Lean follows a similar stream of thought, but uses the idea of edit spheres instead. The next section explains the formal definitions of edit spheres, and their relationship to the triangle inequality for `LDist`, but we state the final metric axiom as it is stated and proven in *InsDelSpheres.lean* below:

```
theorem LDist_tri_ineq (X Y Z : list ℕ) :   LDist X Z ≤ LDist X Y + LDist Y Z
```

F. Formalization of Insertion/Deletion spheres

In this section we introduce the formal definitions of insertion spheres, deletion spheres, and edit spheres. This section contains two contributions: 1) inductive versions of the definitions of deletion, insertion, and edit spheres; and 2) a formal lemma proving a relationship between edit spheres and the Levenshtein distance that significantly simplifies proofs involving edit spheres.

Before introducing any of the spheres, we must first mention the related notions of deletions and insertions as defined in *InsDel.lean*. The deletion definition is as follows:

```
def del_nth: list α → ℕ → list α
| [] _ := []
| [x] _ := []
| (x :: X) 0 := X
| (x :: X) (i + 1) := x :: del_nth X i
```

In the definition, the first line names the deletion function `del_nth` and states that it takes as input values: 1) a list over an arbitrary alphabet (technically a Type) α, and 2) a natural number (the location of the deletion), and then outputs a list over α. The second and third lines state that given the empty list or any singleton list, and any natural number as inputs, the resulting output is the empty list. The fourth line states that given any non-empty list `(x :: X)` with `x` at the head and the natural number `0` as inputs, the resulting output is the tail of the list, i.e. the list with the head deleted. Finally, the last line gives a recursion for any non-empty list `(x :: X)` and natural number `(i + 1)` by keeping the original head `x` and executing the deletion function `del_nth X i`.

In this way the deletion function is inductively defined. The intuition behind the name of the function is that given a list `X`, and natural number `n`, then `del_nth X n` is the resulting list after the `n`th element of list `X` is deleted. Here, the index of elements in list `X` begins at `0`. As a quick example, note that `del_nth [1,2,3] 0 = [2,3]` and `del_nth [1,2,3] 1 = [1,3]`.

Insertions are also defined in *InsDel.lean*. The definition is as follows:

```
def ins_nth: list α → ℕ → α → list α
| [] _ a := [a]
| (x :: X) 0 a := a :: x :: X
| (x :: X) (n+1) a := x :: ins_nth X n a
```

The format is similar to that of the `del_nth` definition, but in this instance the `ins_nth` function takes three input values: 1) the original list over α, 2) a natural number (the location of the insertion), and 3) an element of α (to be inserted), and outputs a new list over α. There is also one less line needed in the definition compared to the definition of `del_nth` since the last line, where the recursive portion of the definition is given, accounts for the case of a singleton list input and non-zero index. As a quick example, note that `ins_nth [1,2,3] 0 4 = [4,1,2,3]` and `ins_nth [1,2,3] 3 4 = [1,2,3,4]`.

Besides the above definitions, definitions of a list of deletions or insertions are also provided in *InsDel.lean*, as well as several lemmas related to the defined deletions and insertions. The next file, *InsDelSpheres.lean* uses these definitions to define three types of spheres: deletion spheres, insertion spheres, and edit spheres. In typical mathematical language, we may define a deletion sphere $dS_t(Y)$ centered at a tuple $Y \in \Sigma^*$ of radius $t \in \mathbb{N}$ as follows: $dS_t(Y) := \{X \in \Sigma^*$ such that X is obtainable from Y by t or fewer deletions$\}$. In terms of Lean, this would be difficult to define directly, but we define the equivalent sphere inductively below. This new way of defining deletion spheres (and insertion spheres and edit spheres defined subsequently), is the first of two contributions in this section.

```
def del_sphere: ℕ → list α → list α → Prop
| 0 X Y := X = Y
| (t + 1) X Y := del_sphere t X Y
                  ∨ ∃ (n : ℕ), del_sphere t X (del_nth Y n)
```

Similarly to previous definitions, the first line names the deletion sphere function `del_sphere` and states that the defined function takes as inputs a natural number (the radius) and two lists, then returns a proposition that is true or false. The second line states that in order for `X` to be within a sphere of radius `0` centered at `Y`, then `X` must be equal to `Y`. The third line says that `Y` is in the deletion sphere of radius $t + 1$ centered at `Y` if either of the following is true: 1) `X` is in the deletion sphere of radius `t` centered at `Y`, or 2) there exists a natural number `n` such that `X` is in the deletion sphere of radius `t` centered at `del_nth Y n`.

The definition of an insertion sphere $iS_t(Y)$ of radius t centered at $Y \in \Sigma^*$ is defined analogously to the deletion sphere: $iS_t(Y) := \{X \in \Sigma^*$ such that X is obtainable from Y by t or fewer insertions$\}$. The Lean version, ins_sphere, is analogous to the definition of del_sphere, except using ins_nth rather than del_nth:

```
def ins_sphere: ℕ → list α → list α → Prop
| 0 X Y := X = Y
| (t+1) X Y := ins_sphere t X Y
                ∨ ∃ (n : ℕ)(a : α), ins_sphere t X (ins_nth Y n a)
```

Finally, an edit sphere $eS_t(Y)$ centered at Y may be defined as: $eS_t(Y) := \{X \in \Sigma^*$ such that X is obtainable from Y by t or fewer edits (deletions or insertions)$\}$. The Lean version, then, is as follows:

```
def edit_sphere: ℕ → list α → list α → Prop
| 0 X Y := X = Y
| (t + 1) X Y := edit_sphere t X Y
                  ∨ ∃ (n : ℕ), edit_sphere t X (del_nth Y n)
                  ∨ ∃ (n : ℕ)(a : α), edit_sphere t X (ins_nth Y n a)
```

Based on the above definition, the notation edit_sphere t X Y means that X is in the edit sphere of radius t centered at Y. In order to formally prove that a list X is in the t edit sphere of Y requires showing that one of the above statements in the definition is true. In the case that X is equal to Y, this is a relatively trivial matter. In fact, in *InsDelSpheres.lean* we prove the lemma that for for any natural number t and list X, we have ins_sphere t X X. Moreover, the proof is only two lines and uses only the definition of ins_sphere and built-in tactics within the standard Lean library. However, it is not so trivial to prove that a list X is within a t radius edit sphere of Y when X is not equal to Y. For instance, the following example shows what a typical proof may look like that [1,3,4] is in the edit sphere of radius 2 centered at [1,2,3].

```
example : edit_sphere 2 [1,3,4][1,2,3] :=
begin
unfold edit_sphere, apply or.inr, apply or.inr,
apply exists.intro 3, apply exists.intro 4,
apply or.inr, apply or.inl,
apply exists.intro 1, refl
end
```

In the above example, the first line states an unnamed theorem, that [1,3,4] is in the radius 2 edit sphere of [1,2,3]. The "begin" and "end" below the statement indicate the start and finish of a proof of that statement in Lean's tactic mode (see [7] for more information on Lean's tactic mode). The remaining lines instruct Lean explicitly on which of the three "or" statements in the definition of edit_sphere will be shown to complete the proof and then explicitly providing the position to be deleted or the position and value to be inserted in order to transform [1,2,3] into [1,3,4]. This process is carried out one edit at a time, so that the first two lines, beginning with "unfold," amount to a proof that inserting a 4 at position 3 brings the resulting list ([1,2,3,4]) within the radius 1 sphere of [1,3,4]. The following two lines amount to a proof that subsequently deleting position 1 brings the result within a radius 0 sphere of [1,3,4]. It is only at this point that Lean can automatically infer the validity of the proof using its built-in simplifier.

The example above was only for an edit sphere radius of 2, but illustrates how cumbersome proofs can become. In fact, each time the radius is increased by 1, the current style of proof, directly from the definition, will require two new lines. This is because the definition is inductive, and the proof requires reducing to the base case. However, by relating edit_sphere and LDist, we can significantly reduce the proof complexity. The following lemma allows us to treat a logical proposition as a calculable function, and is the second contribution of this section.

```
lemma edit_sphere_iff_LDist_le (t : ℕ)(X Y : list ℕ) :
 edit_sphere t X Y ↔ LDist X Y ≤ t
```

The first line states the name of the lemma and indicates that an arbitrary natural number `t` (the radius) and two arbitrary lists `X` and `Y` over ℕ are assumed. The second line, which is the actual statement of the lemma, states that `X` is in the `t` edit sphere of `Y` if and only if the Levenshtein distance `LDist X Y` between `X` and `Y` is less than or equal to `t`. An application of this lemma permits a significant simplification of proofs involving `edit_sphere`, because LEAN can automatically and trivially determine the validity of statements involving `LDist`. After applying `edit_sphere_iff_LDist_le` to define the decidability of `edit_sphere`, Lean can similarly check the validity of statements involving `edit_sphere`. This allows us to rewrite the previous example with a single line proof as follows:

```
example: edit_sphere 2 [1,3,4] [1,2,3] := dec_trivial
```

Moreover, the same proof will work for arbitrarily large radii and arbitrarily long lists. For instance, the following is also a valid statement and proof:

```
example : edit_sphere 10 [1,2,3,4,5][6,7,8,9,0] := dec_trivial
```

In addition, the formalized relationship between `LDist` and `edit_sphere` enables us to prove the triangle inequality mentioned at the end of section ref. To prove the triangle inequality, we first formalize the fact that for lists `X`, `Y`, and `Z` and for natural numbers `a` and `b`, that `edit_sphere a X Y` and `edit_sphere b Y Z` implies `edit_sphere (a+b) X Z`. This can be thought of as a pseudo triangle inequality for edit spheres, although of course it is not a true triangle inequality since `edit_sphere (a+b) X Y` is a proposition, not a number. This fact is in turn used to complete the proof of the triangle inequality for `LDist`.

G. Conclusion

In this chapter we reported on the creation of Cotoleta, the first coding theory library for the Lean theorem-proof assistant. In particular, we defined structures that serve as templates for formalizing error correcting systems and provided examples of code formalization using repetition codes and the Hamming (7,4) code. Further research plans include building more structures to serve as alternative templates for formalizing codes, defining other codes, and proving properties about them.

We also formalized several fundamental topics for insertion/deletion codes in the Lean theorem prover. This includes concepts of deletions, insertions, and edit spheres using inductive definitions. We formalized the Levenshtein distance (also proving it satisfies the metric axioms) in terms of longest common subsequences and proved a relationship between this distance and edit spheres. The relationship simplifies proofs related to edit spheres.

7. Conclusion

Contributions were divided into two general categories. The first was contributions to permutation and multipermutation codes, classes of nonlinear codes. For permutation codes in the Kendall tau metric, we provided definitions and theorems that help to extend a linear programming decoding technique that was originally invented for use in permutation codes in the Euclidean metric. Efficient decoding is a necessity for codes to be practically implemented, so this is a helpful result.

We also proved limits on the maximum possible code size of Ulam permutation codes by first giving new methods of calculating Ulam sphere sizes. In particular, we proved the theorem that perfect error correcting Ulam permutation codes do not exist. In the case of Ulam multipermutation codes, we also showed how to calculate Ulam multipermutation spheres for certain parameters, and used these results to provide new bounds on the maximal code sizes. It is helpful to know what is the maximum possible code size when attempting to construct codes because this gives a frame of reference for how good the rate of the code, and hence to a degree how good the code can be.

The second general category of contribution was to formalization in coding theory. We provided a new library for coding theory in the Lean theorem proof assistant, named Cotoleta. Two groups of files are contained in the library, the first group containing several underlying mathematical definitions and lemmas as well as the formalization of Repetition codes and the Hamming (7,4) code. The first group also contained structures for error correcting systems.

The second group contained definitions, lemmas, and theorems related to Levenshtein codes. These are nonlinear codes capable of correcting deletions or insertions. The Levenshtein metric and deletion, insertion, and edit spheres were all defined, as well as deletion, insertion, and edit correcting codes.

APPENDIX A

CONSOLIDATION FOR COMPACT CONSTRAINTS

Definition (Linear Constraints, Satisfy, $\models$)**.** A **linear constraint** $l(X)$ for an n-by-n matrix is defined as either a linear equation or linear inequality on entries of a matrix.

Formally speaking, by regarding an entry $X_{i,j}$ as a variable ($0 \leq i, j < n$), we state either

$$l(X) : \sum_{0 \leq i,j < n} c_{i,j} X_{i,j} = c_0,$$

or

$$l(X) : \sum_{0 \leq i,j < n} c_{i,j} X_{i,j} \geq c_0,$$

for some $c_0, c_{i,j} \in \mathbb{R}$. The relation $=$ or $\geq$ is uniquely determined by $l(X)$. Instead of the symbols $=$ and $\geq$, we may use $\trianglerighteq_l$ (or simply $\trianglerighteq$), e.g.,

$$l(X) : \sum_{0 \leq i,j < n} c_{i,j} X_{i,j} \trianglerighteq_l c_0.$$

If we do not need to clarify the variable X of a linear constant $l(X)$, we denote it simply by l.

For a linear constraint $l \in \mathcal{L}$ and a matrix $X \in \mathrm{M}_n(\mathbb{R})$, if X satisfies l, we write $X \models l$. If $X \models l$ for every $l \in \mathcal{L}$, we write $X \models \mathcal{L}$.

Definition (Doubly Stochastic Constraint)**.** A **doubly stochastic constraint** $\mathcal{L}$ for an n-by-n matrix is a set of linear constraints such that, for any matrix X for which $X \models \mathcal{L}$, X satisfies the following three types of constraints:

1) (row-sum constraints) $\sum_{0 \leq j < n} X_{i_0,j} = 1$, for any $0 \leq i_0 < n$,

2) (column-sum constraints) $\sum_{0 \leq i < n} X_{i,j_0} = 1$, for any $0 \leq j_0 < n$,

3) (positivity) $X_{i_0,j_0} \geq 0$, for any $0 \leq i_0, j_0 < n$.

Note A.1. A trivial example of a doubly stochastic constraint $\mathcal{L}_D$ is defined as

$$\begin{aligned}\mathcal{L}_D \quad := \quad & \{\text{row-sum constraints}\}\\ & \cup\{\text{column-sum constraints}\}\\ & \cup\{\text{positivity}\}.\end{aligned}$$

In other words, $\mathcal{L}_D$ consists of only of the linear constraints that are in the definition of a doubly stochastic constraint, with no additional constraints. For clarifying the size n of a matrix, we may denote $\mathcal{L}_D$ by $\mathcal{L}_{D^{(n)}}$.

For any doubly stochastic constraint $\mathcal{L} \supset \mathcal{L}_{D^{(n)}}$, the set of $n \times n$ matrices satisfying $\mathcal{L}$ is called a **doubly stochastic polytope** of $\mathcal{L}$ and is denoted by $\mathcal{D}_n[\mathcal{L}]$.
Let us define the set DSM_n as

$$\mathrm{DSM}_n := \{X \in \mathrm{M}_n(\mathbb{R}) \ : \ X \models \mathcal{L}_{D^{(n)}}\}.$$

An element of DSM_n is said to be a **doubly stochastic matrix**. Notice that the set DSM_n comprises precisely the set of matrices satisfying $\mathcal{L}_{D^{(n)}}$, but no other linear constraints.

It can be shown that the symmetric group S_n of order $n!$ is a subset of DSM_n. Indeed, we may embed the permutation group S_n on $\{0, 1, \ldots, n-1\}$ into the set $\mathrm{M}_n(\mathbb{R})$ of n by n matrices over the real numbers in the following manner: for a permutation σ in S_n, we define an n-by-n matrix X^σ by

$$X^\sigma_{i,j} := \delta_{j=\sigma(i)}, \tag{14}$$

where δ is the Kronecker's delta.

Since $X^\sigma \models \mathcal{L}_D$ in Note A.1 for any permutation $\sigma \in S_n$, we have

$$S_n \subset \mathrm{DSM}_n.$$

Geometrically speaking, DSM_n is a convex polytope and is known as the **Birkhoff Polytope**. More generally, for any doubly stochastic constraint $\mathcal{L} \supset \mathcal{L}_{D^{(n)}}$, the set of $n \times n$ matrices satisfying $\mathcal{L}$ is called a **doubly stochastic polytope** of $\mathcal{L}$ and is denoted by $\mathcal{D}_n[\mathcal{L}]$. We may also begin with a set of matrices rather than with a doubly stochastic constraint and utilize the following variation in notation. A subset $\mathcal{D} \subset \mathrm{M}_n(\mathbb{R})$ is a doubly stochastic polytope if there exists a doubly stochastic constraint $\mathcal{L}$ such that $\mathcal{D} = \mathcal{D}_n[\mathcal{L}]$.

The Birkhoff Polytope, DSM_n, is a specific example of a doubly stochastic polytope. Moreover, any doubly stochastic polytope is a subset of DSM_n. This is because adding any nontrivial linear constraint to the set $\mathcal{L}_{D^{(n)}}$ yields a doubly stochastic constraint that excludes some element of DSM_n.

Definition (Vertex)**.** Let $\mathcal{D}$ be a doubly stochastic polytope.

An element $X \in \mathcal{D}$ is said to be a **vertex** if there are neither elements $X_0, X_1 \in \mathcal{D}$ with $X_0 \neq X_1$ nor positive numbers $c_0, c_1 \in \mathbb{R}$, $c_0 + c_1 = 1$, such that

$$X = c_0 X_0 + c_1 X_1.$$

We denote the set of vertices for $\mathcal{D}$ by $\mathrm{Ver}(\mathcal{D})$.

Definition (LP-decodable permutation code)**.** We call a permutation code $(G, \vec{\mu})$ an **LP (Linear Programing)-decodable permutation code** if there exists a doubly stochastic constraint $\mathcal{L}$ such that $G = G_{\mathcal{L}}$, where $G_{\mathcal{L}} := \mathrm{Ver}(\mathcal{D}_n[\mathcal{L}]) \cap S_n$.

Definition (Compact Constraint)**.** Let $\mathcal{L}$ be a doubly stochastic constraint for an n-by-n matrix.

We call $\mathcal{L}$ a **compact constraint** if

- $\mathcal{L}$ consists of a finite number of linear constraints,
- the doubly stochastic polytope $\mathcal{D}_n[\mathcal{L}]$ is a bounded set,
- the vertex set satisfies $\mathrm{Ver}(\mathcal{D}_n[\mathcal{L}]) \subset S_n$.

Theorem A.1 (Birkhoff von-Neuman Theorem [13, 74])**.**

$$\mathrm{Ver}(\mathrm{DSM}_n) = S_n.$$

Merged Constraints

Merged constraints provide a novel technique to construct compact constraints. They were introduced in [37] First, constraints that are obtained by relaxing doubly stochastic constraints are defined. We begin with a relaxed version of the "positivity" (all matrix entries ≥ 0) component of a doubly stochastic constraint, calling this relaxed version a homogeneous constraint.

Definition (Homogeneous Constraints)**.** Let l be a linear constraint for an n-by-n matrix. We call l **homogeneous** if the constant term of l is 0.

Formally speaking,

$$l(X) : \sum_{0 \leq i,j < n} c_{i,j} X_{i,j} \trianglerighteq 0,$$

for some $c_{i,j} \in \mathbb{R}$. The symbol "$\trianglerighteq$" is used to signify that the linear constraint could be an equation ($=$) or an inequality ($\geq$).

Example A.2. *The "positivity" of a doubly stochastic constraint is homogeneous but the "row-sum" constraint (all entries of a matrix row add to* 1*) is not. Linear constraints* $XA^{\Gamma} = A^{\Gamma}X$ *obtained from a graph* Γ *are homogeneous.*

The following constraints are relaxed versions of the "row-sum" (and "column-sum") constraints of a doubly stochastic constraint. Recall that the original "row-sum" (or "column-sum") constraints required all entries of a particular row (or column) to sum to 1. The relaxed counterparts maintain the requirement that all rows (columns) are equal, but waive the requirement of summing to 1. Although the original row and column sum constraints were not homogeneous, these relaxed constraints, which we call weak row-sum (weak column-sum) constraints are homogeneous constraints.

Definition (Weak Row-sum (Column-sum) Constraint)**.** We call the following n-linear constraints **weak row-sum constraints**:

$$\sum_{0 \leq j < n} X_{i_0,j} = \sum_{0 \leq j < n} X_{0,j}, \text{ for any } 0 \leq i_0 < n.$$

Similarly, we call the following n-linear constraints **weak column-sum constraints**:

$$\sum_{0 \leq i < n} X_{i,j_0} = \sum_{0 \leq i < n} X_{i,0}, \text{ for any } 0 \leq j_0 < n.$$

The following constraint, termed a quasi-homogeneous constraint, is a relaxed version of a doubly stochastic constraint. It is obtained by replacing the original “positivity” constraints with homogeneous constraints.

Definition (Quasi-homogeneous Constraint)**.** A linear constraint $\mathcal{L}$ is said to be a **quasi-homogeneous constraint** if $\mathcal{L}$ consists of homogeneous constraints, “row-sum” constraints, and “column-sum” constraints.

Next, a merged constraint is a relaxed version of a quasi-homogenous constraint. The intuition behind the term “merged” is borrowed from the economic term of a merged company, where two companies combine under a new set of rules. In context, a merged constraint is the result of combining the row-sum (column-sum) constraints and homogeneous constraints of a quasi-homogeneous constraint, but in the process changing the row-sum (column-sum) constraints to weak row-sum (column-sum) constraint.

Definition (Merged Constraint)**.** Let $\mathcal{L}$ be a quasi-homogeneous constraint. For $\mathcal{L}$, we define another set $\mathcal{L}^{\square}$ of linear constraints by replacing row-sum (column-sum) constraints in $\mathcal{L}$ with weak row-sum (column-sum) constraints, while keeping the original homogeneous constraints of $\mathcal{L}$. We call $\mathcal{L}^{\square}$ a **merged constraint** for $\mathcal{L}$.

Remark A.3. Merged constraints are homogeneous.

Example A.4. *A merged constraint $\mathcal{L}_D^{\square}$ for the constraints $\mathcal{L}_D$ in Note A.1 consists of three kinds of linear constraints:*

- *weak row-sum constraints,*
- *weak column-sum constraints,*
- *positivity (positivity remains unchanged as it is homogeneous).*

Several constraints were introduced. To avoid confusion and aid comprehension, the following tables summarize the relationship between categories of constraints and their relaxed counterparts that have been discussed thus far. Table 2 compares positivity with its relaxed version of a homogeneous constraint. Similarly, Table 3 compares Row and Column-sum constraints with their relaxed versions. Finally, Table 4 shows the components that make up a doubly stochastic, quasi-homogeneous, and merged constraint respectively.

TABLE XI
POSITIVITY, HOMOGENEOUS CONSTRAINTS

Positivity	**Homogeneous Constraints**
$X_{i,j} \geq 0$ for any $0 \leq i, j < n$	$\sum_{0 \leq i,j < n} c_{i,j} X_{i,j} \trianglerighteq 0$

TABLE XII
ROW-SUM (COLUMN-SUM) AND WEAK ROW-SUM (COLUMN-SUM) CONSTRAINTS

Row-sum Constraints	**Weak Row-sum Constraints**
$\sum_{0 \leq j < n} X_{i,j} = \sum_{0 \leq j < n} X_{0,j} = 1$, for any $0 \leq i < n$	$\sum_{0 \leq j < n} X_{i,j} = \sum_{0 \leq j < n} X_{0,j}$, for any $0 \leq i < n$

Column-sum Constraints	**Weak Column-sum Constraints**
$\sum_{0 \leq i < n} X_{i,j} = \sum_{0 \leq i < n} X_{i,0} = 1$, for any $0 \leq j < n$	$\sum_{0 \leq i < n} X_{i,j} = \sum_{0 \leq i < n} X_{i,0}$, for any $0 \leq j < n$

TABLE XIII
DOUBLY STOCHASTIC CONSTRAINT, QUASI-HOMOGENEOUS CONSTRAINT, MERGED CONSTRAINT

Doubly Stochastic Constraint	**Quasi-Homogeneous Constraint**	**Merged Constraint**
Row-sum Constraints	Row-sum Constraints	Weak Row-sum Constraints
Column-sum Constraints	Column-sum Constraints	Weak Column-sum Constraints
Positivity	Homogeneous Constraints	Homogeneous Constraints

Holding Constraints

Next we propose an extension method for a given linear constraint of a small matrix to another constraint of a larger matrix. Again, borrowing economic terminology, we call such constraints "Holding Constraints," as in a holding company that holds another company's stock but does not produce it's own goods. In context, holding constraints essentially consist of constraints for matrices that originate from other constraints for smaller matrices.

Let ν and R be positive integers. For a νR-by-νR matrix X, we may divide X into R-by-R block matrices $X^{[r_0,r_1]}$ of size ν-by-ν via the following relation:

$$X^{[r_0,r_1]}_{i,j} = X_{r_0\nu+i, r_1\nu+j},$$

where $0 \leq i, j < \nu$ and $0 \leq r_0, r_1 < R$.

For example, if $\nu = 3$ and $R = 2$, we have

$$\begin{aligned}
&\begin{pmatrix}
X_{00} & X_{01} & X_{02} & X_{03} & X_{04} & X_{05} \\
X_{10} & X_{11} & X_{12} & X_{13} & X_{14} & X_{15} \\
X_{20} & X_{21} & X_{22} & X_{23} & X_{24} & X_{25} \\
X_{30} & X_{31} & X_{32} & X_{33} & X_{34} & X_{35} \\
X_{40} & X_{41} & X_{42} & X_{43} & X_{44} & X_{45} \\
X_{50} & X_{51} & X_{52} & X_{53} & X_{54} & X_{55}
\end{pmatrix} \\
= &\begin{pmatrix}
X^{[00]} & X^{[01]} \\
X^{[10]} & X^{[11]}
\end{pmatrix}
\end{aligned}$$

and

$$X^{[01]} = \begin{pmatrix}
X_{03} & X_{04} & X_{05} \\
X_{13} & X_{14} & X_{15} \\
X_{23} & X_{24} & X_{25}
\end{pmatrix}.$$

We call $X^{[r_0, r_1]}$ the $(r_0; r_1)$**-th block** of X.

Definition (Holding Constraints)**.** Let $\mathcal{H}$ be a set of linear constraints for an R-by-R matrix H. We associate with $\mathcal{H}$ another set of linear constraints $\mathcal{H}^{\#}$ of degree ν for a νR-by-νR matrix X.

For $h(H) \in \mathcal{H}$, we define the linear constraint $h^{\#}(X)$ for a νR-by-νR matrix X by replacing each component H_{r_0, r_1} with a linear sum $\sum_{0 \leq j < \nu} X_{0,j}^{[r_0, r_1]}$. For $\mathcal{H}$, we define a set $\mathcal{H}^{\#}$ of linear constraints for νR-by-νR matrix as

$$\mathcal{H}^{\#} := \{h^{\#} \ : \ h \in \mathcal{H}\}.$$

We call $\mathcal{H}^{\#}$ a **holding constraint** associated with $\mathcal{H}$ of degree ν.

Example A.5. *Let us define a linear constraint set $\mathcal{H}$ for a 2-by-2 matrix H (that is $R = 2$)*

by $\mathcal{H} := \{h_1(H) : H_{00} + H_{01} = 1, h_2(H) : H_{00} + H_{10} = 1, h_3(H) : H_{11} \geq 0\}$. The holding constraint $\mathcal{H}^{\#}$ of degree 3 (that is $\nu = 3$) is

$$\begin{aligned}\mathcal{H}^{\#} = \{ & \\ & h_1^{\#}(X) : \quad (X_{00}^{[00]} + X_{01}^{[00]} + X_{02}^{[00]}) \\ & \qquad\qquad + (X_{00}^{[01]} + X_{01}^{[01]} + X_{02}^{[01]}) = 1, \\ & h_2^{\#}(X) : \quad (X_{00}^{[00]} + X_{01}^{[00]} + X_{02}^{[00]}) \\ & \qquad\qquad + (X_{00}^{[10]} + X_{01}^{[10]} + X_{02}^{[10]}) = 1, \\ & h_3^{\#}(X) : \quad (X_{00}^{[11]} + X_{01}^{[11]} + X_{02}^{[11]}) \geq 0 \\ \}. & \end{aligned}$$

To better understand the above example, the reader is referred to the prior discussion on the "$(r_0; r_1)$-th block, $X^{[r_0,r_1]}$ of a matrix X. For instance, $h_1^{\#}(X)$ in example 9 would be equivalent to the following constraint: $(X_{00} + X_{01} + X_{02}) + (X_{03} + X_{04} + X_{05}) = 1$.

Consolidation

The method of "consolidation" was introduced in [37] to construct compact linear constraints from two given types of constraints. Once again the intuition behind the name comes from economics, where a consolidation is the gathering of smaller companies under a single head. In the context of constraints, a consolidation is the result of taking the union of Holding constraints and Merged constraints. We recommend reading the following definition in conjunction with Figure 7.

Definition (Consolidation)**.** Let $\mathcal{M}^{[r_0,r_1]}$ be a quasi-homogeneous constraint for a ν-by-ν matrix for $0 \leq r_0, r_1 < R$. Let $\mathcal{H}$ be a set of linear constraints for an R-by-R matrix.

For $\{\mathcal{M}^{[r_0,r_1]}\}$ and $\mathcal{H}$, we define another linear constraint $\mathcal{M} \boxplus \mathcal{H}$ for a νR-by-νR matrix as follows:

$$\begin{aligned}\mathcal{M} \boxplus \mathcal{H} \;&:=\; \{m^{[r_0,r_1]\square}(X^{[r_0,r_1]}) \;:\; m^{[r_0,r_1]} \in \mathcal{M}^{[r_0,r_1]}, 0 \le r_0, r_1 < R\} \\ &\cup\{h^{\#} \;:\; h \in \mathcal{H}\},\end{aligned}$$

where X is a νR-by-νR matrix and $X^{[r_0,r_1]}$ is the $(r_0; r_1)$-th block of X of size ν-by-ν. In plain language, $\mathcal{M} \boxplus \mathcal{H}$ consists of the union of all the merged constraint components $m^{[r_0,r_1]\square}$ originating from quasi-homogeneous constraint components $m^{[r_0,r_1]}$ of $\mathcal{M}^{[r_0,r_1]}$ as well as all holding constraints $h^{\#}$ originating from constraints h of $\mathcal{H}$. This is depicted in Figure 6.

We call $\mathcal{M} \boxplus \mathcal{H}$ the **consolidation** of $\{\mathcal{M}^{[r_0,r_1]}\}$ and $\mathcal{H}$.

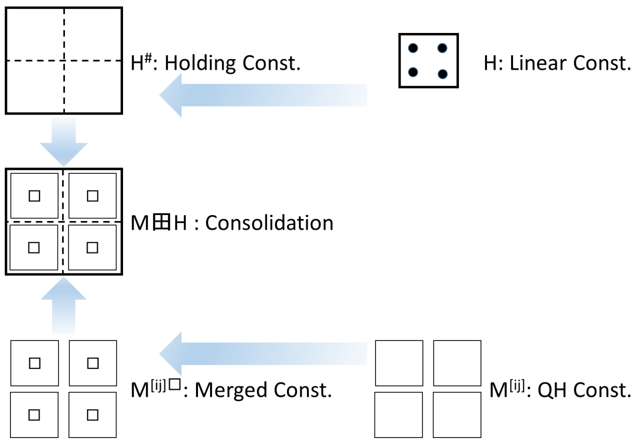


Fig. 7. Consolidation

Example A.6. *Let $\mathcal{L}_{D^{(2)}}$ be the doubly stochastic constraint in Note A.1 (all rows and columns add to 1 and all matrix entries are ≥ 0.) for a 2-by-2 matrix. Let $\mathcal{M}^{[0,0]} := \mathcal{L}_{D^{(2)}}$, $\mathcal{M}^{[0,1]} := \mathcal{L}_{D^{(2)}}$, and $\mathcal{M}^{[1,0]} := \mathcal{L}_{D^{(2)}}$. Let $\mathcal{M}^{[1,1]} := \mathcal{L}_{D^{(2)}} \cup \{X_{0,0} + X_{1,1} = 0\}$. Let $\mathcal{H}$ be the doubly stochastic constraint in Note A.1 for a 3-by-3 matrix, i.e. $\mathcal{H} = \mathcal{L}_{D^{(3)}}$.*

Then the consolidation $\mathcal{M}\boxplus\mathcal{H}$ is a doubly stochastic constraint for a 6-by-6 matrix X. $\mathcal{M}\boxplus\mathcal{H}$

consists of the following constraints, numbered 1 - 6:

$$1) \sum_{0 \le i < 6} X_{i,j_0} \;=\; 1, \textit{ for } 0 \le j_0 < 6,$$

$$2) \sum_{0 \le j < 6} X_{i_0,j} \;=\; 1, \textit{ for } 0 \le i_0 < 6,$$

These first two constraints (arguably constraint 3 as well) originate from the Holding constraints formed from $\mathcal{H}$ and basically correspond to $\mathcal{H}^{\#}$ in Figure 6. However, because of the interplay with the merged constraints that are added in the consolidation process, the stated constraints are stronger. For example, by the original $\mathcal{H}^{\#}$, the top row of X must sum to 1, but nothing is said of the second row. Here the weak-row constraints of $\mathcal{M}^{[0,0]\square}$ and $\mathcal{M}^{[0,1]\square}$, cause the second row to also sum to 1.

$$3) X_{i_0,j_0} \;\ge\; 0, \textit{ for } 0 \le i_0, j_0 < 6,$$

$$\begin{aligned} 4) \sum_{0 \le i < 3} X_{2r_0+i,2r_1+j_0} \;&=\; \sum_{0 \le j < 3} X_{2r_0,2r_1+j}, \textit{ for} \\ & \quad 0 \le j_0 < 3, 0 \le r_0, r_1 < 2, \end{aligned}$$

$$\begin{aligned} 5) \sum_{0 \le j < 3} X_{2r_0+i_0,2r_1+j} \;&=\; \sum_{0 \le i < 3} X_{2r_0+i,2r_1}, \textit{ for} \\ & \quad 0 \le i_0 < 3, 0 \le r_0, r_1 < 2, \end{aligned}$$

$$6) X_{3,3} + X_{4,4} + X_{5,5} \;=\; 0.$$

The bottom four constraints listed above are essentially the merged constraints from $\mathcal{M}$ (although there is some overlap with the holding constraints) and correspond to $\mathcal{M}^{[i,j]\square}$ in Figure

6. Lemma 1 is used implicitly so that the stated constraints 4) and 5) are slightly stronger.

The main contribution of this section is the following:

Theorem A.2. *[Theorem 2 of [37]] Let $\mathcal{M}^{[r_0,r_1]}$ be a quasi-homogeneous constraint for a ν-by-ν matrix for $0 \leq r_0, r_1 < R$. Let $\mathcal{H}$ be a doubly stochastic constraint for an R-by-R matrix.*

If $\mathcal{M}^{[r_0,r_1]}$ is compact for all r_0, r_1 and $\mathcal{H}$ is also compact, we have the following:

1) $\mathrm{Ver}(\mathcal{D}_{\nu R}[\mathcal{M} \boxplus \mathcal{H}]) = \{(H_{r_0,r_1} X^{[r_0,r_1]}) \ : \ H \in \mathrm{Ver}(\mathcal{D}_R[\mathcal{H}]), X^{[r_0,r_1]} \in \mathrm{Ver}(\mathcal{D}_\nu[\mathcal{M}^{[r_0,r_1]}]), 0 \leq r_0, r_1 < R\}$.
2) *the consolidation $\mathcal{M} \boxplus \mathcal{H}$ is compact.*
3) *the cardinality of* $\mathrm{Ver}(\mathcal{D}_{\nu R}[\mathcal{M} \boxplus \mathcal{H}])$ *is*

$$\sum_{\sigma \in \mathrm{Ver}(\mathcal{D}_R[\mathcal{H}])} v^{[0,\sigma(0)]} v^{[1,\sigma(1)]} \cdots v^{[R-1,\sigma(R-1)]},$$

where $v^{[r,\sigma(r)]}$ denotes the cardinality of $\mathrm{Ver}(\mathcal{M}^{[r,\sigma(r)]})$.

APPENDIX B

Proof of Remark 4.1:

Let $\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r \in \mathcal{M}_r(\mathbb{S}_n)$. We will first show that $\mathrm{d}_\circ(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r) \geq n - \ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r)$. By definition of $\mathrm{d}_\circ(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r)$, there exist $\sigma' \in R_r(\sigma)$ and $\pi' \in R_r(\pi)$ such that $\mathrm{d}_\circ(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r) = \mathrm{d}_\circ(\sigma', \pi') = n - \ell(\sigma', \pi')$. Hence if for all $\sigma' \in R_r(\sigma)$ and $\pi' \in R_r(\pi)$ we have $\ell(\sigma', \pi') \leq \ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r)$, then $\mathrm{d}_\circ(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r) \geq n - \ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r)$ (subtracting a larger value from n results in a smaller overall value). Therefore it suffices to show that that for all $\sigma' \in R_r(\sigma)$ and $\pi' \in R_r(\pi)$, that $\ell(\sigma', \pi') \leq \ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r)$. This is simple to prove because if two permutations have a common subsequence, then their corresponding r-regular multipermutations will have a related common subsequence. Let $\sigma' \in R_r(\sigma)$, $\pi' \in R_r(\pi)$, and $\ell(\sigma', \pi') = k$. Then there exist indexes $1 \leq i_1 < i_2 < \cdots < i_k \leq n$ and $1 \leq j_1 < j_2 < \cdots < j_k \leq n$ such that for all $p \in [k]$, $\sigma'(i_p) = \pi'(j_p)$. Of course, whenever $\sigma'(i) = \pi'(j)$, then $\mathbf{m}_{\sigma'}^r(i) = \mathbf{m}_{\pi'}^r(j)$. Therefore $\ell(\sigma', \pi') = k \leq \ell(\mathbf{m}_{\sigma'}^r, \mathbf{m}_{\pi'}^r) = \ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r)$.

Next, we will show that $\mathrm{d}_\circ(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r) \leq n - \ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r)$. Note that

$$\begin{aligned} \mathrm{d}_\circ(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r) &= \min_{\sigma' \in R_r(\sigma), \pi' \in R_r(\pi)} \mathrm{d}_\circ(\sigma', \pi') \\ &= \min_{\sigma' \in R_r(\sigma), \pi' \in R_r(\pi)} (n - \ell(\sigma', \pi')) \\ &= n - \max_{\sigma' \in R_r(\sigma), \pi' \in R_r(\pi)} \ell(\sigma', \pi'). \end{aligned}$$

Here if $\max\limits_{\sigma' \in R_r(\sigma), \pi' \in R_r(\pi)} \ell(\sigma', \pi') \geq \ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r)$, then $\mathrm{d}_\circ(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r) \leq n - \ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r)$ (subtracting a smaller value from n results in a larger overall value). It is enough to show that there exist $\sigma' \in R_r(\sigma)$ and $\pi' \in R_r(\pi)$ such that $\ell(\sigma', \pi') \geq \ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r)$. To prove this fact, we take a longest common subsequence of $\mathbf{m}_\sigma^r$ and $\mathbf{m}_\pi^r$ and then carefully choose $\sigma' \in R_r(\sigma)$ and $\pi' \in R_r(\pi)$ to have an equally long common subsequence. The next paragraph describes how this can be done.

Let $\ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r) = k$ and let $(1 \leq i_1 < i_2 < \cdots < i_k \leq n)$ and $(1 \leq j_1 < j_2 < \cdots < j_k \leq n)$ be integer sequences such that for all $p \in [k]$, $\mathbf{m}_\sigma^r(i_p) = \mathbf{m}_\pi^r(j_p)$. The existence of such sequences is guaranteed by the definition of $\ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r)$. Now for all $p \in [k]$, define $\sigma'(i_p)$ to be the smallest integer $l \in [n]$ such that $\mathbf{m}_\sigma(l) = \mathbf{m}_\sigma(i_p)$ and if $q \in [k]$ with $q < p$, then $\mathbf{m}_\sigma^r(i_q) = \mathbf{m}_\pi^r(i_p)$ implies $\sigma'(i_q) < \sigma'(i_p) = l$. For all $p \in [k]$, define $\pi(j_p)$ similarly. Then for all $p \in [k]$,

$\sigma'(i_p) = \pi'(j_p)$. The remaining terms of σ' and π' may easily be chosen in such a manner that $\sigma' \in R_r(\sigma)$ and $\pi' \in R_r(\pi)$. Thus there exist $\sigma' \in R_r(\sigma)$ and $\pi' \in R_r(\pi)$ such that $\ell(\sigma', \pi') \geq \ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r)$. $\square$

APPENDIX C

Proof of Remark 4.2:

Suppose $\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r \in \mathcal{M}_r(\mathbb{S}_n)$. There exists a translocation $\phi \in \mathbb{S}_n$ such that $\ell(\mathbf{m}_\sigma^r \cdot \phi, \mathbf{m}_\pi^r) = \ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r) + 1$, since it is always possible to arrange one element with a single translocation. This then implies that $\min\{k \in \mathbb{Z} \ : \ \text{there exists } (\phi_1, \ldots, \phi_k) \text{ such that } \mathbf{m}_\sigma^r \cdot \phi_1 \cdots \phi_k = \mathbf{m}_\pi^r\} \leq n - \ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r) = \mathrm{d}_\circ(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r)$. At the same time, given $\ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r) \leq n$, then for all translocations $\phi \in \mathbb{S}_n$, we have that $\ell(\mathbf{m}_\sigma^r \cdot \phi, \mathbf{m}_\pi^r) \leq \ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r) + 1$, since a single translocation can only arrange one element at a time. Therefore by Remark 4.1, $\min\{k \in \mathbb{Z} \ : \ \text{there exists } (\phi_1, \ldots, \phi_k) \text{ s.t } \mathbf{m}_\sigma^r \cdot \phi_1 \cdots \phi_k = \mathbf{m}_\pi^r\} \geq n - \ell(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r) = \mathrm{d}_\circ(\mathbf{m}_\sigma^r, \mathbf{m}_\pi^r)$. $\square$

APPENDIX D

Proof of Lemma 5.16:

Recall that n is an even integer. Assume $c \in [n-1]$ and $\mathbf{q} := (q(1), q(2), \ldots, q(c)) \in \mathbb{Z}^c_{>0}$ such that $\sum_{i=1}^{c} q(i) = n$. For the first direction, suppose there exists some $i \in [c]$ such that $q(i)$ is even. Since n is even, the number of odd values in $\mathbf{q}$ is even, i.e. $\#\{i \in [c] \ : \ q(i) \text{ is odd}\} = 2k$ for some nonnegative integer k. We will now construct an $\mathbf{m} \in \{1,2\}^n$ with an equal number of 1's and 2's, whose cuts correspond to $\mathbf{q}$. We begin by defining two sets: first $\{q^*(1), q^*(2), \ldots, q^*(2k)\} := \{q(i) \ : \ q(i) \text{ is odd}\}$ and then $\{q^*(2k+1), q^*(2k+2), \ldots q^*(c)\} := \{q(i) \ : \ q(i) \text{ is even}\}$. Then define $\mathbf{m}$ as follows:

$$\begin{aligned}\mathbf{m} \ := \ & (\underbrace{\mathbf{m}[a_1, b_1]}_{g^*(1)}, \underbrace{\mathbf{m}[a_2, b_2]}_{g^*(2)}, \ldots, \underbrace{\mathbf{m}[a_k, b_k]}_{g^*(k)}, \ \underbrace{\mathbf{m}[a_{k+1}, b_{k+1}]}_{g^*(2k+1)}, \\ & \underbrace{\mathbf{m}[a_{k+2}, b_{k+2}]}_{g^*(k+1)}, \underbrace{\mathbf{m}[a_{k+3}, b_{k+3}]}_{g^*(k+2)}, \ldots, \underbrace{\mathbf{m}[a_{2k+1}, b_{2k+1}]}_{g^*(2k)}, \ \underbrace{\mathbf{m}[a_{2k+2}, b_{2k+2}]}_{g^*(2k+2)} \ldots, \underbrace{\mathbf{m}[a_c, b_c]}_{g^*(c)})\end{aligned}$$

where $\mathbf{m}(1) = 1$ and for all $j \in [c]$, $\mathbf{m}[a_j, b_j]$ is a cut.

The idea here is simple. By the definition of $\mathbf{m}$, each of the first k cuts begin and end with 1 since they are all odd length cuts, and thus each will has one more 1 than 2. The $(k+1)$th cut, $\mathbf{m}[a_{k+1}, b_{k+1}]$, is taken to be of even length, which reverses the order of the subsequent k cuts. Hence the k cuts from $\mathbf{m}[a_{k+2}, b_{k+2}]$ through $\mathbf{m}[a_{2k+1}, b_{2k+1}]$ each begin and end with 2, so each will have one more 2 than 1. The remaining cuts from $\mathbf{m}[a_{k+2}, b_{k+2}]$ through $\mathbf{m}[a_c, b_c]$ are even, which implies that the number of 1's and 2's in each of these cuts is equal. Hence, we may write the following:

$$\begin{aligned}
\#\{i \in [n] \ : \ \mathbf{m}(i) = 1\} \quad = \quad & \#\{i \in [a_1, b_k] \ : \ \mathbf{m}(i) = 1\} \\
& + \#\{i \in [a_{k+2}, b_{2k+1}] \ : \ \mathbf{m}(i) = 1\} \\
& + \#\{i \in [a_{k+1}, b_{k+1}] \cup [a_{2k+2}, b_c] \ : \ \mathbf{m}(i) = 1\} \\
\\
= \quad & \#\{i \in [a_1, b_k] \ : \ \mathbf{m}(i) = 2\} + k \\
& + \#\{i \in [a_{k+2}, b_{2k+1}] \ : \ \mathbf{m}(i) = 2\} - k \\
& + \#\{i \in [a_{k+1}, b_{k+1}] \cup [a_{2k+2}, b_c] \ : \ \mathbf{m}(i) = 2\} \\
\\
= \quad & \#\{i \in [n] \ : \ \mathbf{m}(i) = 2\}.
\end{aligned}$$

Applying Lemma 5.15 completes the first direction.

For the second direction, let $\mathbf{m}_\sigma \in \mathcal{M}_r(\mathbb{S}_n)$ be a binary multipermutation such that

$$\mathbf{m}_\sigma^r = (\mathbf{m}_\sigma^r[a_1, b_1], \mathbf{m}_\sigma^r[a_2, b_2], \ldots \mathbf{m}_\sigma^r[a_c, b_c]),$$

with each $\mathbf{m}_\sigma^r[a_j, b_j]$ a cut whose length corresponds to an element of $\mathbf{q}$. We want to show then that there exists some $i \in [c]$ such that $\mathbf{q}(i)$ is even. We proceed by contradiction.

Suppose that there does not exist an $i \in [c]$ such that $\mathbf{q}(i)$ is even. Then for all $j \in [c]$, $\mathbf{m}_\sigma^r[a_j, b_j]$ is odd. This then implies that for each cut $\mathbf{m}_\sigma^r[a_j, b_j]$, that $\mathbf{m}_\sigma^r(a_j) = \mathbf{m}_\sigma^r(b_j)$. In other words, an odd-length cut necessarily begins and ends with the same element. However, to end one cut and begin another, it is necessary to repeat a digit. This implies that for each $j \in [c]$, and cut $\mathbf{m}_\sigma^r[a_j, b_j]$,

$$\#\{i \in [a_j, b_j] \ : \ \mathbf{m}_\sigma^r(i) = \mathbf{m}_\sigma^r(1)\} = \#\{i \in [a_j, b_j] \ : \ \mathbf{m}_\sigma^r(i) \neq \mathbf{m}_\sigma^r(1)\} + 1,$$

which in turn implies that the total number of elements of $\mathbf{m}_\sigma^r$ that equal to $\mathbf{m}_\sigma^r(1)$ is exactly c more than the number of elements not equal to $\mathbf{m}_\sigma^r(1)$, contradicting Lemma 5.15. $\square$

Appendix E

Remark E.1. For any integer $a \in \mathbb{Z}$, it is easily verified that

1) If a is even, then $\lfloor (a/2)^2 \rfloor = (a/2)^2$

2) If a is odd, then $\lfloor (a/2)^2 \rfloor = (a/2)^2 - 1/4$.

Remark E.2. Let $a \in \mathbb{Z}$. Then the following is a direct consequence of the previous remark.

1) If a is even, then $\lfloor (a/2)^2 \rfloor - \lfloor ((a-1)/2)^2 \rfloor = (a/2)$.

2) If a is odd, then $\lfloor (a/2)^2 \rfloor - \lfloor ((a-1)/2)^2 \rfloor = (a/2) - 1/2$.

Remark E.3. Let $a, b \in \mathbb{Z}_{\geq 1}$ such that $a - b \geq 2$. Then

$$\left\lfloor \left(\frac{a-2}{2}\right)^2 \right\rfloor + \left\lfloor \left(\frac{b-2}{2}\right)^2 \right\rfloor \;\geq\; \left\lfloor \left(\frac{(a-1)-2}{2}\right)^2 \right\rfloor + \left\lfloor \left(\frac{(b+1)-2}{2}\right)^2 \right\rfloor, \qquad (15)$$

with equality holding only if $a - b = 2$ and both a and b are odd.

Proof. Assume that $a, b \in \mathbb{Z}_{\geq 1}$ and $a - b \geq 2$. Then inequality (15) holds if and only if the following inequality also holds.

$$\left\lfloor \left(\frac{a-2}{2}\right)^2 \right\rfloor - \left\lfloor \left(\frac{(a-1)-2}{2}\right)^2 \right\rfloor - \left(\left\lfloor \left(\frac{(b+1)-2}{2}\right)^2 \right\rfloor - \left\lfloor \left(\frac{b-2}{2}\right)^2 \right\rfloor \right) \geq 0. \qquad (16)$$

Applying Remark E.2 to the four cases when a is either even or odd and b is either even or odd, a routine calculation shows the following: If both a and b are even, then the left side of inequality (16) equals $(a-b)/2 > 0$. If a is even and b is odd or if a is odd and b is even, then the left side of inequality (16) equals $(a-b-1)/2 > 0$. Finally, if both a and b are odd, then the left side of inequality (16) equals $(a-b-2)/2 \geq 0$. $\square$

APPENDIX F

Proof of Lemma 5.18:

Assume $c \in [n-2]$ and $\hat{c} \leq c$. We will split the proof into two cases, when $q_{c+1} \leq c$ and when $q_{c+1} > c$. Let us first suppose $q_{c+1} \leq c$. This corresponds roughly to the case where $\hat{c} \leq c \leq \sqrt{n}$. In this instance, for all $i \in [2, c]$, we have $\mathbf{q}_c(i) - \mathbf{q}_{c+1}(i) \leq 1$ and $\mathbf{q}_c(1) - \mathbf{q}_{c+1}(1) \leq 2$. This is because $\mathbf{q}_{c+1}$ can be constructed from $\mathbf{q}_c$ by shortening each cut $\mathbf{q}_c$ in order to create the $(c+1)$st cut. Since $q_{c+1} \leq c$, each cut will decrease by at most one, except for one exceptional case when q_c is an odd integer and $q_{c+1} = c$, in which case $\mathbf{q}_c(1) - \mathbf{q}_{c+1}(1) = 2$.

Since for all $i \in [2, c]$, $\mathbf{q}_c(i) - \mathbf{q}_{c+1}(i) \leq 1$ and $\mathbf{q}_c(1) - \mathbf{q}_{c+1}(1) \leq 2$, the left hand side of inequality (9) is less than or equal to

$$\psi(\mathbf{q}_c) \; - \; \psi(\mathbf{q}_c(1) - 2) \; - \; \psi(\mathbf{q}_c(2) - 1, \mathbf{q}_c(3) - 1, \ldots, \mathbf{q}_c(c) - 1) \; - \; \psi(\mathbf{q}_{c+1}(c+1)). \quad (17)$$

By adding and subtracting $\psi(\mathbf{q}_c(1) - 1)$ from expression (21) and also disregarding the last term, $-\psi(\mathbf{q}_{c+1}(c+1))$, after some rearrangement expression (21) is less than or equal to

$$\psi(\mathbf{q}_c) - \psi(\mathbf{q}_c - 1) + \psi(\mathbf{q}_c(1) - 1) - \psi(\mathbf{q}_c(1) - 2),$$

which by Remark E.2 is less than or equal to

$$\begin{aligned} \sum_{i=1}^{c} \left(\frac{\mathbf{q}_c(i)}{2} - 1 \right) + \left(\frac{\mathbf{q}_c(1) - 1}{2} - 1 \right) \quad & \leq \quad \left(\frac{n}{2} - c \right) + \left(\frac{(q_c + 1) - 1}{2} - 1 \right) \\ & = \quad r - c + \frac{q_c}{2} - 1, \end{aligned}$$

where this last expression is less than or equal to $r - 1$ since $q_{c+1} \leq c$ implies that $q_c \leq q_{c+1} + 1 \leq 2c$. This concludes the case where $q_{c+1} \leq c$.

Next assume that $q_{c+1} > c$. This corresponds roughly to the case where $\sqrt{n} < c < n - 1$. Note that

$$q_{c+1} = \frac{n - rem_{c+1}}{c+1} \; < \; \frac{n}{c} \; \leq \; \frac{2r}{\hat{c}} \; = \; 2\hat{c} \leq 2c.$$

Since q_{c+1} is strictly less than $2c$, for all $i \in [c]$ we have $\mathbf{q}_c(i) - \mathbf{q}_{c+1}(i) \leq 2$. Moreover, because $c < q_{c+1} < 2c$, the number of $i \in [c]$ such that $\mathbf{q}_c(i) - \mathbf{q}_{c+1}(i) = 2$ is $q_{c+1} - c$. This means then that the number of $i \in [c]$ such that $\mathbf{q}_c(i) - \mathbf{q}_{c+1}(i) = 1$ is equal to $c - (q_{c+1} - c)$. Similarly to before, the reasoning for these set sizes comes from constructing $\mathbf{q}_{c+1}$ from $\mathbf{q}_c$ and considering how much each cut length is decreased to construct the final cut, $\mathbf{q}_{c+1}(c+1)$. Example (5.19) helps here to aid comprehension.

It should be noted that there is one exceptional case, when q_{c+1} is an odd integer and $rem_{c+1} = 0$. In this instance, $\mathbf{q}_{c+1}(c+1) = q_{c+1} - 1$, which means the number of $i \in [c]$ such that $\mathbf{q}_c(i) - \mathbf{q}_{c+1}(i) = 2$ is decreased by one, while the number of $i \in [c]$ such that $\mathbf{q}_c(i) - \mathbf{q}_{c+1}(i) = 1$ is increased by one. It is easily shown that the final effect on the size of the left hand side of inequality (9) is a decrease, so it is enough to prove the inequality in the typical case, when $\#\{i \in [c] \ : \ \mathbf{q}_c(i) - \mathbf{q}_{c+1}(i) = 2\} = q_{c+1} - c$ and $\#\{i \in [c] \ : \ \mathbf{q}_c(i) - \mathbf{q}_{c+1}(i) = 1\} = c - (q_{c+1} - c)$. These set sizes imply that the left hand side of inequality (20) is less than or equal to

$$\begin{aligned} \psi(\mathbf{q}_c) \ &- \ \psi(\mathbf{q}_c(1) - 2, \mathbf{q}_c(2) - 2, \ldots, \mathbf{q}_c(q_{c+1} - c) - 2) \\ &- \ \psi(\mathbf{q}_c(q_{c+1} - c + 1) - 1, \mathbf{q}_c(q_{c+1} - c + 2) - 1, \ldots, \mathbf{q}_c(c) - 1)) \ - \ \psi(\mathbf{q}_{c+1}(c+1)). \end{aligned}$$

By adding and subtracting $\sum_{i=1}^{q_{c+1}-c} \lfloor ((\mathbf{q}_c(i) - 3)/2)^2 \rfloor$, after some rearrangement expression (22) can be rewritten as

$$\begin{aligned} \psi(\mathbf{q}_c) \ &+ \ \psi(\mathbf{q}_c(1) - 1, \mathbf{q}_c(2) - 1, \ldots, \mathbf{q}_c(q_{c+1} - c) - 1) \ - \ \psi(\mathbf{q}_{c+1}(c+1)) \\ - \ \psi(\mathbf{q}_c - 1) \ &- \ \psi(\mathbf{q}_c(1) - 2, \mathbf{q}_c(2) - 2, \ldots, \mathbf{q}_c(q_{c+1} - c) - 2) \end{aligned}$$

which by Remark E.2 is less than or equal to

$$\sum_{i=1}^{c}\left(\frac{\mathbf{q}_c(i)}{2}-1\right) \quad + \quad \sum_{i=1}^{q_{c+1}-c}\left(\frac{\mathbf{q}_c(i)-1}{2}-1\right) \quad - \quad \psi(\mathbf{q}_{c+1}(c+1))$$

$$\leq \quad \left(\frac{n}{2}-c\right) \quad + \quad (q_{c+1}-c)\left(\frac{(q_c+1)-1}{2}-1\right) \quad - \quad \left(\left(\frac{q_{c+1}-2}{2}\right)^2-\frac{1}{4}\right)$$

$$\leq \quad (r-c) \quad + \quad (q_c-c-1)\left(\frac{q_c}{2}-1\right) \quad - \quad \left(\left(\frac{q_c-3}{2}\right)^2-\frac{1}{4})\right),$$

which reduces to $r+q_c^2/4-(cq_c)/2-1$. Because of the fact that $q_{c+1} < 2c$ implies $q_c \leq q_{c+1}+1 \leq 2c$, this final expression is guaranteed to be less than or equal to $r-1$, completing the proof.

Appendix G

Proof of Lemma 5.20:

Assume $c \in [n-1]$ and $c \leq \hat{c}$. The left hand side of inequality (9) depends on the difference between $\mathbf{q}_{c-1}(i)$ and $\mathbf{q}_c(i)$ for each $i \in [c-1]$. We wish to show that these differences are sufficiently large to cause inequality (9) to be satisfied. Note that

$$q_c \;=\; \frac{n - rem_c}{c} \;>\; \frac{n-c}{c} \;=\; \frac{n}{c} - 1 \;>\; \frac{n}{c^2}(c-1).$$

Since $c \leq \hat{c}$, we have $n/c^2 = 2r/c^2 \geq 2$, which implies that $q_c > 2(c-1)$. Therefore, for all $i \in [c-1]$, we have $\mathbf{q}_{c-1}(i) - \mathbf{q}_c(i) \geq 2$. This is because $\mathbf{q}_c$ can be constructed from $\mathbf{q}_{c-1}$ by shortening each cut of $\mathbf{q}_{c-1}$ in order to create the cth cut of $\mathbf{q}_c$, whose length is at least $q_c - 1$. Example 5.21 helps comprehension here.

Next, let $k := \#\{i \in [c-1] \;:\; \mathbf{q}_{c-1}(i) - \mathbf{q}_c(i) > 2\}$. In other words, k is the number of cuts in $\mathbf{q}_{c-1}$ that are decreased by more than 2 in the construction of $\mathbf{q}_c$ from $\mathbf{q}_{c-1}$. Notice that if $\mathbf{q}_{c-1}(i) - \mathbf{q}_c(i) = 2$ then by Remark E.1,

$$\psi(\mathbf{q}_{c-1}) - \psi(\mathbf{q}_c) \;=\; \left(\frac{\mathbf{q}_{c-1}(i) - 2}{2}\right)^2 - \left(\frac{\mathbf{q}_c(i) - 2}{2}\right)^2.$$

Moreover, Remark E.1 also implies that in all other instances,

$$\psi(\mathbf{q}_{c-1} - \psi(\mathbf{q}_c) \;\geq\; \left(\frac{\mathbf{q}_{c-1}(i) - 2}{2}\right)^2 - \left(\frac{\mathbf{q}_c(i) - 2}{2}\right)^2 - \frac{1}{4}.$$

Hence the left hand side of inequality (9) is greater than or equal to

$$\sum_{i=1}^{c-1} \left(\frac{\mathbf{q}_{c-1}(i) - 2}{2}\right)^2 - \sum_{i=1}^{c-1} \left(\frac{(\mathbf{q}_c(i) - 2}{2}\right)^2 - \frac{k}{4} - \left(\frac{\mathbf{q}_c(c) - 2}{2}\right)^2. \tag{18}$$

At this point, we split the remainder of the proof into two possibilities, the general case where $\mathbf{q}_c(c) = q_c$ and the exceptional case where $\mathbf{q}_c(c) = q_c - 1$, which only occurs when n/c is an odd integer. We will treat the general case first and then end with some comments about the exceptional case. Since we are first assuming that $\mathbf{q}_c(c) = q_c$, expression (18) is equal to

$$\sum_{i=1}^{c-1}\left(\frac{\mathbf{q}_{c-1}(i)-2}{2}\right)^2 - \sum_{i=1}^{c-1}\left(\frac{(\mathbf{q}_c(i)-2}{2}\right)^2 - \frac{k}{4} - \left(\frac{q_c-2}{2}\right)^2$$

$$= \sum_{i=1}^{c-1}\left(\frac{\mathbf{q}^2_{c-1}(i)}{4} - \mathbf{q}_{c-1} + 1\right) - \sum_{i=1}^{c-1}\left(\frac{\mathbf{q}^2_c(i)}{4} - \mathbf{q}_c + 1\right) - \frac{k}{4} - \frac{g_c^2}{4} + q_c - 1$$

$$= \sum_{i=1}^{c-1}\left(\frac{\mathbf{q}^2_{c-1}(i)}{4}\right) - n + (c-1) - \left[\sum_{i=1}^{c-1}\left(\frac{\mathbf{q}^2_c(i)}{4}\right) - (n-q_c) + (c-1)\right] - \frac{k}{4} - \frac{q_c^2}{4} + q_c - 1$$

$$= \frac{1}{4}\sum_{i=1}^{c-1}\left(\mathbf{q}^2_{c-1}(i) - \mathbf{q}^2_c(i)\right) - \frac{k}{4} - \frac{q_c^2}{4} - 1. \tag{19}$$

From here, we focus on the summation $\sum_{i=1}^{c-1}(\mathbf{q}^2_{c-1}(i) - \mathbf{q}^2_c(i))$ to prove that the overall expression is sufficiently large. The summation can be viewed as the sum of all shaded areas in Figure 8. In the figure, squares of area $\mathbf{q}^2_{c-1}(i)$ (with $i \in [c-1]$), are placed along the diagonal of an n-by-n square. Within the bottom left corner of each of these squares is placed another square of area $\mathbf{q}^2_c(i)$ (again $i \in [c-1]$). By carefully examining the total area of all shaded regions in the figure, we can lower bound $\sum_{i=1}^{c-1}(\mathbf{q}^2_{c-1}(i) - \mathbf{q}^2_c(i))$ to satisfy the desired inequality.

Figure 9 depicts the difference $\mathbf{q}^2_{c-1}(i) - \mathbf{q}^2_c(i)$ for an individual $i \in [c-1]$. This is a closer view of one of the individual squares along the main diagonal in Figure 8. From Figure 9, we can observe that the value $(\mathbf{q}^2_{c-1}(i) - \mathbf{q}^2_c(i))$ can be visualized geometrically as the combined areas of two types of shapes - the rectangles shaded light gray and the square shaded dark gray. First, note that there are two identical rectangles (shaded light grey) whose dimensions are $\mathbf{q}_{c-1}(i) - \mathbf{q}_c(i)$ by $\mathbf{q_c}$. We know that for all $i \in [c-1]$, that $\mathbf{q}_c(i) \geq q_c$, and that $\sum_{i=1}^{c-1}(\mathbf{q}_{c-1}(i) - \mathbf{q}_c(i)) = \mathbf{q}_c(c) = q_c$. Hence, the sum of the area of all lightly shaded rectangles in Figure 8 is at least $2q_c^2$.

Next, we will focus on the square shaped region (shaded dark grey) in Figure 9. The dimensions

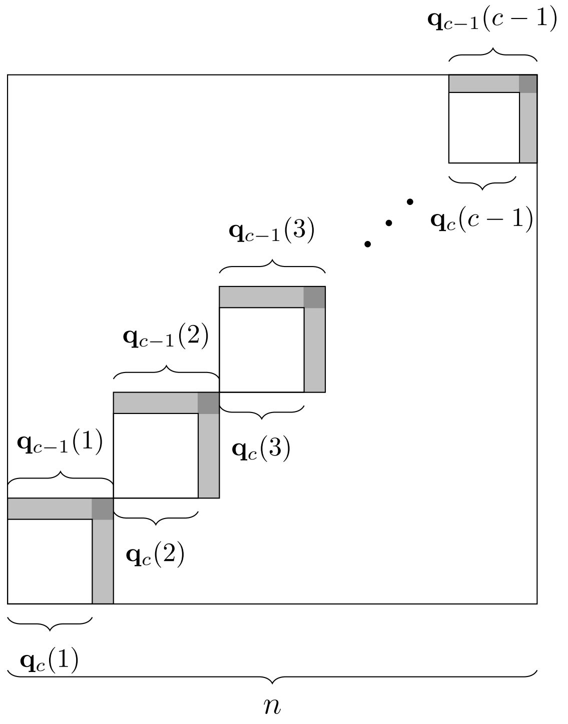


Fig. 8. Diagram of $\sum_{i=1}^{c-1}\left(\mathbf{q}_{c-1}^2(i) - \mathbf{q}_c^2(i)\right)$

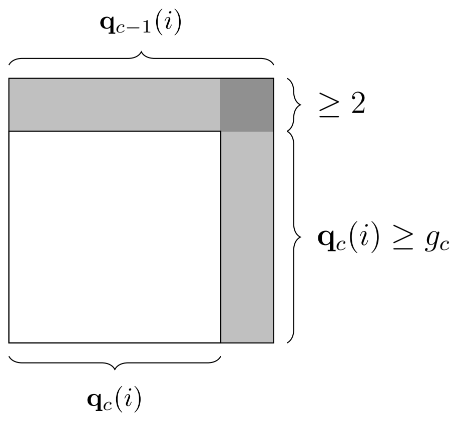


Fig. 9. Diagram of $\mathbf{q}_{c-1}^2(i) - \mathbf{q}_c^2(i)$

of this square are $(\mathbf{q}_{c-1}(i) - \mathbf{q}_c(i))$ by $(\mathbf{q}_{c-1}(i) - \mathbf{q}_c(i))$. Therefore

$$\sum_{i=1}^{c-1}\left(\mathbf{q}_{c-1}^2(i) - \mathbf{q}_c^2(i)\right) \quad \geq \quad 2q_c^2 + \sum_{i=1}^{c-1}(\mathbf{q}_{c-1}(i) - \mathbf{q}_c(i))^2.$$

We saw previously that $\mathbf{q}_{c-1}(i) - \mathbf{q}_c(i) \geq 2$, and again using the fact that $\sum_{i=1}^{c-1}(\mathbf{q}_{c-1}(i) - \mathbf{q}_c(i)) =$

$\mathbf{q}_c(c) = q_c$, the total area of all the sum of all dark grey shaded square regions in Figure 8 is at least $2q_c$. Moreover, each time the difference between $\mathbf{q}_{c-1}(i)$ and $\mathbf{q}_c(i)$ is greater than 2, this means an overall increase of $(\mathbf{q}_{c-1}(i) - \mathbf{q}_c(i))^2$ by at least 3. Hence $\sum_{i=1}^{c-1}(\mathbf{q}_{c-1}(i) - \mathbf{q}_c(i))^2 \geq 2q_c + 3k$, which implies that expression (19) is greater than or equal to

$$\frac{q_c^2}{2} + \frac{q_c}{2} + \frac{3k}{4} - \frac{q_c^2}{4} - \frac{k}{4} - 1 \quad = \quad \frac{q_c^2}{4} + \frac{q_c}{2} + \frac{k}{2} - 1. \tag{20}$$

To complete the proof in the general case, recall that $q_c > n/c - 1$. Thus, by replacing q_c with $n/c - 1$, we have that the right hand side of equation (20), afer some basic reduction, is greater than

$$\frac{r^2}{c^2} - \frac{1}{4} + \frac{k}{2} - 1. \tag{21}$$

In this final expression, since $c \leq \hat{c}$, we have $r^2/c^2 \geq r$. Also, since q_c was strictly greater than $2(c-1)$, we know that $k \geq 1$, completing the proof in the general case.

For the exceptional case, when $\mathbf{q}_c(c) = q_c - 1$, we can follow the same argument as in the general case with slight modification. In this instance the last term in expression (19) is reduced since $\mathbf{q}_c(c) = q_c - 1$ rather than q_c, resulting in a larger overall value. Using this fact, we can then show that whenever $\mathbf{q}_c(c) = q_c - 1$, expression (19) is greater or equal to

$$\frac{1}{4}\sum_{i=1}^{c-1}\left(\mathbf{q}_{c-1}^2(i) - \mathbf{q}_c^2(i)\right) - \frac{k}{4} - \frac{q_c^2}{4} + \frac{q_c}{2} - \frac{5}{4}. \tag{22}$$

By a similar argument to the general case, we can also show that if $\mathbf{q}_c(c) = q_c - 1$, then

$$\sum_{i=1}^{c-1}\left(\mathbf{q}_{c-1}^2(i) - \mathbf{q}_c^2(i)\right) \quad \geq \quad 2q_c^2 + 2.$$

This fact, along with the fact that $q_c > n/c - 1$, implies that expression (19), and therefore the left hand side of inequality (9), is greater than

$$\frac{r^2}{c^2} + \frac{k}{2} - 1 \quad \geq \quad r - 1.$$

Appendix H

Proof of Lemma 5.23:

We first show that $\hat{c} \leq \lfloor \hat{c} \rfloor + 0.5$ implies

$$\psi(\mathbf{q}_{\lfloor \hat{c} \rfloor}) - \psi(\mathbf{q}_{\lceil \hat{c} \rceil}) \quad \leq \quad r - 1.$$

Let $\hat{c} \leq \lfloor \hat{c} \rfloor + 0.5$. Then by Lemma 5.22, we know that $r \leq \lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor$. From here, we will split the proof into two possibilities: (1) where $r < \lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor$; and (2) where $r = \lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor$.

First, suppose that $r < \lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor$. In the event that $\hat{c} = \lfloor \hat{c} \rfloor$, then $\lfloor \hat{c} \rfloor = \lceil \hat{c} \rceil$, which implies that the left hand side of inequality (11) is equal to 0, so that the conclusion holds trivially. Thus we will assume that $\lfloor \hat{c} \rfloor < \hat{c}$, which implies that $\lceil \hat{c} \rceil = \lfloor \hat{c} \rfloor + 1$. From here, for ease of notation, and in order to see the connection to Lemma 5.18 more clearly, let $c := \lfloor \hat{c} \rfloor$ so that $c + 1 = \lceil \hat{c} \rceil$.

Next, note that

$$q_{c+1} \;=\; \frac{n - rem_{c+1}}{c+1} \;\leq\; \frac{n}{c+1} \;=\; \frac{2r}{c+1},$$

and that

$$\frac{2r}{c+1} \;<\; 2c \quad \text{if and only if} \quad r \;<\; c(c+1) \;=\; \lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor,$$

which is true by assumption. Therefore $q_{c+1} < 2c$. From this point, the proof for the case where $r < \lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor$ is the same as that of Lemma 5.18.

We now consider the second possibility. Assume that $r = \lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor$. We claim that this implies that the left hand side of inequality (11) is exactly equal to $r - 1$. This also has the implication that $\lfloor \hat{c} \rfloor$ and $\lceil \hat{c} \rceil$ both yield the same maximum sphere size. To see why the claim is true, note first that

$$r \;=\; \lfloor \hat{c} \rfloor^2 + \lfloor \hat{c} \rfloor \;=\; \lfloor \hat{c} \rfloor(\lfloor \hat{c} \rfloor + 1) \;=\; \lfloor \hat{c} \rfloor \cdot \lceil \hat{c} \rceil.$$

This implies that

$$q_{\lfloor \hat{c} \rfloor} \;=\; \frac{n}{\lfloor \hat{c} \rfloor} \;=\; 2\lceil \hat{c} \rceil \quad \text{and that} \quad q_{\lceil \hat{c} \rceil} \;=\; \frac{n}{\lceil \hat{c} \rceil} \;=\; 2\lfloor \hat{c} \rfloor.$$

Therefore we have

$$\psi(\mathbf{q}_{\lfloor\hat{c}\rfloor}) \;=\; \psi((\underbrace{2\lceil\hat{c}\rceil, 2\lceil\hat{c}\rceil, \ldots 2\lceil\hat{c}\rceil}_{\lfloor\hat{c}\rfloor})) \;=\; \lfloor\hat{c}\rfloor\,(\lfloor\hat{c}\rfloor)^2 \;=\; \lfloor\hat{c}\rfloor^3, \tag{23}$$

and similarly,

$$\psi(\mathbf{q}_{\lceil\hat{c}\rceil}) \;=\; \psi((\underbrace{2\lfloor\hat{c}\rfloor, 2\lfloor\hat{c}\rfloor, \ldots 2\lfloor\hat{c}\rfloor}_{\lceil\hat{c}\rceil})) \;=\; (\lfloor\hat{c}\rfloor + 1)\,(\lfloor\hat{c}\rfloor - 1)^2 \;=\; \lfloor\hat{c}\rfloor^3 - \lfloor\hat{c}\rfloor^2 - \lfloor\hat{c}\rfloor + 1. \tag{24}$$

Finally, subtracting (23) and (24), we have that the left hand side of inequality (11) is equal to $\lfloor\hat{c}\rfloor^2 + \lfloor\hat{c}\rfloor - 1$, which is equal to $r - 1$ by the assumption that $r = \lfloor\hat{c}\rfloor^2 + \lfloor\hat{c}\rfloor$. This completes the first half of the proof.

We next show that $\hat{c} \;>\; \lfloor\hat{c}\rfloor + 0.5$ implies

$$\psi(\mathbf{q}_{\lfloor\hat{c}\rfloor}) - \psi(\mathbf{q}_{\lceil\hat{c}\rceil}) \quad > \quad r - 1. \tag{25}$$

Let $\hat{c} \;>\; \lfloor\hat{c}\rfloor + 0.5$. For ease of notation and to see the connection to Lemma 5.20, let $c := \lceil\hat{c}\rceil$ so that $c - 1 = \lfloor\hat{c}\rfloor$. By Lemma 5.22, we have

$$r \;>\; (c-1)^2 + (c-1) \;=\; (c-1)(c) \quad \text{which implies that} \quad \frac{r}{c} \;>\; c - 1.$$

Note that

$$q_c \;=\; \frac{n - rem_c}{c} \;>\; \frac{n}{c} - 1 \;=\; \frac{2r}{c} - 1 \;>\; 2(c-1) - 1, \quad \text{which implies that} \;\; q_c \geq 2(c-1).$$

At the same time, note that

$$q_c \;=\; \frac{n - rem_c}{c} \;\leq\; \frac{n}{c} \;=\; \frac{2r}{c} \;<\; \frac{2r}{\hat{c} - 1},$$

which is easily shown to be strictly less than $3(\hat{c} - 1)$ as long as $r \geq 30$, and clearly $3(\hat{c} - 1)$ is strictly less than $3(c - 1)$. It is also easily verified numerically that inequality (25) is satisfied for $r < 30$ (assuming that $\hat{c} > \lfloor \hat{c} \rfloor + 0.5$). Hence, for the remainder of the proof, we shall assume that $r \geq 30$, which means that

$$2(c-1) \;\leq\; q_c \;<\; 3(c-1).$$

From here we split into two cases, (1) when $q_c = 2(c-1)$ exactly; and (2) when $q_c > 2(c-1)$.

First, assume that $q_c = 2(c-1)$. Then following similar logic to the proof of Lemma 5.20, we know that for all $i \in [c-1]$, that $\mathbf{q}_{c-1}(i) - \mathbf{q}_c(i) \;=\; 2$. Technically this is assuming that we are not in the special case when $n/(c-1)$ is an odd integer, in which case $\mathbf{q}_{c-1}(1) - \mathbf{q}_c(1) = 3$ and $\mathbf{q}_{c-1}(c-1) - \mathbf{q}_c(c-1) = 1$. However, this would result in an overall increase of the left hand side of inequality (25), so it is enough to consider the general case when $n/(c-1)$ is not an odd integer.

Since $\mathbf{q}_{c-1}(i) - \mathbf{q}_c(i) = 2$ for each $i \in [c-1]$, then the left hand side of inequality (25) is equal to

$$\psi(\mathbf{q}_{c-1}) - \psi(\mathbf{q}_{c-1} - 2) - \psi(\mathbf{q}_c(c))$$

By Remark E.2 and the fact that $\mathbf{q}_c(c) = q_c = 2(c-1)$, the above expression is equal to

$$\begin{aligned}
&\sum_{i=1}^{c-1}\left(\frac{\mathbf{q}_{c-1}(i)-2}{2}\right)^2 - \sum_{i=1}^{c-1}\left(\frac{(\mathbf{q}_{c-1}(i)-4}{2}\right)^2 - \left(\frac{(2(c-1)-2}{2}\right)^2 \\
=\;& \sum_{i=1}^{c-1}\left(\frac{\mathbf{q}_{c-1}^2(i)}{4} - \mathbf{q}_{c-1}(i) + 1\right) - \sum_{i=1}^{c-1}\left(\frac{\mathbf{q}_{c-1}^2(i)}{4} - 2\mathbf{q}_{c-1}(i) + 4\right) - (c-1)^2 \\
=\;& \sum_{i=1}^{c-1}(\mathbf{q}_{c-1}(i) - 3) - \left(c^2 - 4c + 4\right)^2 \\
=\;& n - 3(c-1) - c^2 + 4c - 4 \\
=\;& 2r + c - c^2 - 1 \\
=\;& r - 1 + r - (c-1)c. \qquad (26)
\end{aligned}$$

We saw earlier that $r > (c-1)c$, so expression 26 is greater than $r-1$. This completes the proof for the case when $q_c = 2(c-1)$.

Next suppose that $q_c > 2(c-1)$. Let $k := \#\{i \in [c-1] : \mathbf{q}_{c-1}(i) - \mathbf{q}_c(i) = 3\}$. In other words, k is the number of cuts in $\mathbf{q}_{c-1}$ that are decreased by 3 in the construction of $\mathbf{q}_c$ from $\mathbf{q}_{c-1}$. Recall that $q_c < 3(c-1)$, so for all $i \in [c-1]$, we have $\mathbf{q}_{c-1}(i) - \mathbf{q}_c(i) \leq 3$. Since we are also assuming that $q_c > 2(c-1)$, we know that $q_c = 2(c-1) + k$. This is because q_c is equal to 2 times the number of cuts in $\mathbf{q}_{c-1}$ decreased by 2, plus 3 times the number of cuts in $\mathbf{q}_{c-1}$ decreased by 3 in the construction of $\mathbf{q}_c$.

Following the same reasoning as in the proof of Lemma 5.20, we can then show that the left hand side of inequality 25 is greater or equal to

$$\frac{q_c^2}{2} + \frac{q_c}{2} + \frac{k}{2} - 1. \tag{27}$$

Expression 27 is the same expression obtained as the right hand side of equation 20 in the proof of Lemma 5.20. At this stage, however, we recall the fact that under the current assumptions, $q_c = 2(c-1) + k$. Substituting $2(c-1) + k$ for q_c and simplifying, after some rearranging we obtain that expression 27 is equal to

$$c^2 - 1 + c(k-1) + \frac{k^2}{4},$$

which is greater than $r - 1 + c(k-1) + k^2/4$ since by the assumption that $\hat{c} > \lfloor \hat{c} \rfloor + 0.5$, we know $c = \lceil \hat{c} \rceil > \hat{c}$. Finally, this last expression is greater than $r - 1$ as long as $k \geq 1$, which we know is true since $q_c > 2(c-1)$. □

References

[1] R. Affeldt, D. Nowak, K. Yamada, "Certifying Assembly with Formal Cryptographic Proofs: the case of BBS," *J. Science of Computer Programming*, Vol. 77, pp. 1058-1074, 2012.

[2] R. Affeldt, J. Garrigue, "Formalization of Error-Correcting Codes: from Hamming to Modern Coding Theory," *Interactive Theorem Proving*, pp. 17-33, 2015.

[3] R. Affeldt, J. Garrigue, "Formalization of Error-Correcting Codes using ssreflect," *MI lecture note series*,vol. 61, pp. 76-78, 2015.

[4] R. Affeldt, J. Garrigue, T. Saikawa, "Formalization of Reed-Solomon codes and progress report on formalization of LDPC codes," *Proc. ISITA 2016*, pp. 532-536, Oct. - Nov. 2016, CA, USA.

[5] R. Affeldt, M. Hagiwara, J. Sénizergues, "Formalization of Shannon's Theorems," *J. Autom. Reasoning*, vol. 53, pp. 63-103, 2014.

[6] G.E. Andrews, "The Theory of Partitions," *Addison-Wesley*, Reading, Massachusetts, 1976.

[7] J. Avigad, L. de Moura, S. Kong, "Theorem Proving in Lean." https://Leanprover.github.io/tutorial/tutorial.pdf, 2017.

[8] C. Ballarin, L. C. Paulson, "A Pragmatic Approach to Extending Provers by Computer Algebra – with Applications to Coding Theory," *Fundamenta Informaticae*, vol. 34, pp. 1-20, 1999.

[9] H. Barendregt, "Introduction to generalized type systems," *Journal of Functional Programming*, vol. 1, Issue 2, pp. 125-154, 1991.

[10] A. Barg and A. Mazumdar, "Codes in permutations and error correction for rank modulation," *IEEE Trans. Inf. Theory*, vol. 56, pp. 3158-3165, 2010.

[11] T. Berger, F. Jelinek, and J. Wolf, "Permutation codes for sources," *IEEE Trans. Inf. Theory*, vol. I8, pp. 160-169, 1972.

[12] J. Bierbrauer, *Introduction to Coding Theory, (2nd ed.)*, Taylor and Francis, 2016.

[13] Birkhoff, "Three observations on linear algebra," *Univ. Nac. Tacuman*, Rev. Ser. A 5, pp. 147-151, 1946.

[14] I. F. Blake, "Permutation codes for discrete channels," *IEEE Trans. Inf. Theory*, vol. 20, pp. 138-140, 1974.

[15] I.F. Blake, G. Cohen, and M. Deza, "Coding with permutations," *Inform. Control* vol. 43, pp. 1-19, Oct. 1979.

[16] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, K. Strauss, "A DNA-based archival storage system," *Proc. ASPLOS*, pp. 637-649, 2016.

[17] M. Braverman, R. Gelles, J. Mao, R. Ostrovsky, "Coding for interactive communication correcting insertions and deletions," *IEEE Trans. Inf Theory*, vol. 63, Issue 10, pp. 6256-6270, 2017.

[18] Buchheim, C., Cameron, P.J., Wu, T.: "On the subgroup distance problem," *Disc. Math. 309*, pp. 962-968, 2009.

[19] S. Buzaglo, T. Etzion, "Bounds on the size of permutation codes with the Kendall τ-metric," *IEEE Trans. Inf. Theory*, vol. 61, pp. 3241-3250, 2015.

[20] S. Buzaglo, T. Etzion, "Perfect permutation codes with the Kendall's τ-metric," *Proc .ISIT*, pp. 2391-2395, July 2014.

[21] W. Chu, C.J. Colbourn, P.J. Dukes, "Constructions for permutation codes in powerline communications," *Designs, Codes, and Cryptography*, vol. 32, pp. 51-64, 2004.

[22] G. M. Church, Y. Gao, S. Kosuri, "Next-generation digital information storage in DNA," *Science*, 337.6102, pp. 1628-1628, 2012.

[23] G. Cohen, M. Deza, "Decoding of permutation codes," *International CNRS Colloquium*, France, 1977.

[24] L. de Moura, S. Kong, J. Avigad, F. van Doorn, J. von Raumer, "The Lean Theorem Prover (System Description)," in *Lecture Notes in Computer Science* Vol. 9195, pp. 378-388, 2015.

[25] M. Deza, P. Frankl, "On maximal numbers of permutations with given maximal or minimal distance," *J. Combinatorial Theory A*, 22, 1977.

[26] Deza, M., Huang, T.: "Metrics on permutations, a survey," *JCISS*, 1998.

[27] W. Fultion, J. Harris, "Representation Theory: A First Course," Springer, New York 1991.

[28] E. En Gad, E. Yaakobi, A. Jiang, and J. Bruck, "Rank-modulation rewrite coding for flash memories," *IEEE Trans. Inf. Theory*, vol 61, pp.4209-4226, 2015.

[29] F. Farnoud, V. Skachek, and O. Milenkovic,"Error-correction in flash memories via codes in the Ulam metric," *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 3003-3020, 2013.

[30] F. Farnoud and O. Milenkovic, "Multipermutation Codes in the Ulam Metric," in *Proc. ISIT 2014*, pp. 2754-2758, 2014.

[31] J. Frame, G. Robinson, and R. Thrall, "The hook graphs of the symmetric group," *Canad. J. Math.*, vol. 6, pp. 316-324, 1954.

[32] W. Fulton, *Young Tableaux*, London Mathematical Society Student Texts, Cambridge University Press, 1996.

[33] Garey, M., Johnson, D.: (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, ISBN 978-0-7167-1045-5, OCLC 11745039, 1979.

[34] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 7435, pp. 77-80, 2013.

[35] F. Gologlu, J. Lember, A. Riet, V. Skachek, "New Bounds for Permutation Codes in Ulam Metric," in *Proc. ISIT 2015*, pp. 1726-1730, June 2015.

[36] M. Hagiwara, "Cotoleta -Library for Formalized Coding Theory-," http://manau.jp/lean/cotoleta, 2016.

[37] M. Hagiwara, J. Kong, "Consolidation for compact constraints and Kendall tau LP decodable permutation codes," *Designs, Codes and Cryptography*, vol. 85, no. 3, pp. 483-521, Dec. 2017.

[38] M. Hagiwara, K. Nakano, J. Kong, "Formalization of Coding Theory using Lean," *Proc. ISITA 2016*, pp. 522-526, 2016.

[39] Hill, R.: *A First Course in Coding Theory, Oxford Applied Mathematics and Computing Science Series*, Oxford University Press, 165-173, 1986.

[40] J. Holzl, "Construction and Stochastic Applications of Measure Spaces in Higher-Order Logic," PhD thesis, Technische Universitat Munchen, Institut fur Informatik, 2012.

[41] M. Horvitz, T. Etzion, "Constructions of snake-in-the-box codes for rank modulation," *IEEE Trans. Inf. Theory*, vol. 60, pp. 7016-7025, 2014.

[42] W. Howard, "The formulae-as-types notion of construction," *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, pp. 479-490, 1980.

[43] J. E. Humphreys, *Reflection Groups and Coxeter Groups*, Cambridge University Press, 1992.

[44] A. Jiang, R. Mateescu, M. Schwartz, and J. Bruck, "Rank Modulation for Flash Memories," *IEEE Trans. Inf. Theory*, vol. 55, pp. 2659-2673, 2009.

[45] A. Jiang, M. Schwartz, and J. Bruck, "Correcting Charge-Constrained Errors in the Rank-Modulation Scheme," *IEEE Trans. Inf. Theory*, vol. 56, pp. 2112-2120, 2010.

[46] A. Jiang, M. Schwartz, J. Bruck, "Error-correcting codes for rank modulation," *Proc. ISIT 2008*, pp. 1736-1740, 2008.

[47] R. Kane, *Reflection Groups and Invariant Theory*, Springer-Verlag New York, Inc., 2001.

[48] N. Karmarkar, "A new polynomial time algorithm for linear programming," *Combinatorica 4*, pp. 373-395, 1984.

[49] H. M. Kiah, J. G. Puleo, O. Milenkovic, "Codes for DNA sequence profiles," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3125-3145, 2016.

[50] K. Kobayashi, H. Morita, and M. Hoshi, "Enumeration of permutations classified by length of longest monotone subsequences," *Proc. ISIT 1994*, pp. 318, 1994.

[51] J. Kong, M. Hagiwara, “Nonexistence of Perfect Permutation Codes in the Ulam Metric,” Proc. of ISITA 2016, pp. 691-695, OCt.-Nov. 2016.

[52] J. Kong, M. Hagiwara, “Multipermutation Ulam sphere analysis toward characterizing maximal code size,” in *Proc. IEEE ISIT*, pp. 1628-1632, Aug. 2017.

[53] Knuth, D.: *The Art of Computer Programming Volume 3*, Addison-Wesley, 1998.

[54] D. Kracht, S. Schober, “Insertion and deletion correcting DNA barcodes based on watermarks,” *BMC Bioinformatics*, 2015.

[55] https://Leanprover.github.io.

[56] V. I. Levenshtein, “On perfect codes in deletion and insertion metric,” *Descrete Math. Appl.*, vol. 2, no. 3, pp. 241-258, 1992.

[57] F. Lim and M. Hagiwara, “Linear programming upper bounds on permutation code sizes from coherent configurations related to the Kendall tau distance metric,” *Proc. ISIT 2012*, pp. 2998-3002, 2012.

[58] T. Mhamdi, O. Hasan, S. Tahar, “Formalization of Entropy Measures in HOL,” *ITP, M.C.J.D. van Eekelen, H. Geuvers, J. Schmaltz, and F. Wiedijk, Eds. Lecture Notes in Computer Science Series*, vol. 6898, Springer, pp. 233-248, 2011.

[59] T. Mhamdi, O. Hasan, S. Tahar, “Formalization of Measure and Lebesgue Integration over Extended Reals in HOL,” Technical Repert, ECE Dept., Concordia University (Feb 2011), http://hvg.ece.concordia.ca/Publications/TECH_REP/MLX_TR11/

[60] T. Mhambdi, “Information-Theoretic Analysis using Theorem Proving,” PhD Thesis, Concordia University, Montreal, QC, Canada, 2012.

[61] Mazumdar, A., Barg, A., Zemor, G.: “Constructions of Rank Modulation Codes,” *IEEE T. Inf. Theory*, vol. 59, pp. 1018-1029, 2012.

[62] K. Nakano, M. Hagiwara, “Formalization of Binary Symmetric Erasure Channel Based on Infotheo,” *Proc. ISITA 2016*, pp. 512-516, 2016.

[63] W. Peterson, E. J. Weldon, *Error-Correcting Codes* M.I.T. Press, 1972.

[64] W.W. Peterson, J.B. Nation, M.P. Fossorier, “Reflection group codes and their decoding,” *IEEE Trans. Inf. Theory*, vol. 56, pp. 6273-6293, 2010.

[65] N. Raviv, M. Schwartz, E. Yaakobi, “Rank Modulation Codes for DNA Storage,” *Proc. ISIT 2017*, pp. 139-143, 2017.

[66] T. Richardson, R. Urbanke, *Modern Coding Theory*, Cambridge University Press, 2008.

[67] C. Schensted, “Longest increasing and decreasing subsequences,” *Canad. J. Math.*, vol. 13, pp. 179-191, 1961.

[68] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, vol. 27, pp. 379-423, pp. 623-656, 1948.

[69] D. Slepian, “Permutation modulation,” *Proc. IEEE*, vol. 53, pp. 228-236, 1965.

[70] R. P. Stanley, “Algebraic Combinatorics: Walks, Trees, Tableaux, and More,” *Springer Science+Business Media New York*, 2013.

[71] T. G. Swart, H. C. Ferreira, “Decoding distance-preserving permutation codes for power-line communications,” *IEEE AFRICON 2007*, pp. 1-7, 2007.

[72] A.J.H. Vinck, “Coded modulation for powerline communications,” *AEÜ Int. J. Electron. Commun.*, vol. 54, pp. 45-49, 2000.

[73] A. J. H. Vinck, J. Haring, T. Wadayama, “Coded M-FSK for powerline communications,” *Proc. ISIT 2000*, pp. 137, 2000.

[74] J. Von Neumann, “A certain zero-sum two-person game equivalent to an optimal assignment problem,” *Ann. Math. Studies 28*, pp. 5-12, 1953.

[75] T. Wadayama and M. Hagiwara, “LP decodable permutation codes based on linearly constrained permutation matrices,” *IEEE Trans. Inf. Theory*, vol. 58, pp. 5454-5470, 2012.

[76] X. Wang, F.-W Fu, , “On the snake-in-the-box codes for rank modulation under Kendall’s τ-metric,” *Designs, Codes and Cryptography*, vol. 83, no. 2, pp. 455-465, 2017

[77] Y. Yehezkeally, M. Schwartz, “Snake-in-the-box codes for rank modulation,” *IEEE Trans. Inf. Theory* vol. 58, pp. 5471-5483, 2012.

[78] Y.W. Zhang, G.N. Ge, “Snake-in-the-box codes for rank modulation under Kendall’s τ-metric in S_{2n+1},” *IEEE Trans. Inf. Theory* vol. 62, pp. 151-158, 2016.