LOCAL AND GLOBAL COMPUTATION ON ALGEBRAIC DATA

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Venkata Surya Srikanth Gandikota

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2017

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Elena Grigorescu, Chair

Department of Computer Science, Purdue University

Dr. Samuel S. J. Wagstaff

Department of Computer Science, Purdue University

Dr. Greg N. Frederickson

Department of Computer Science, Purdue University

Dr. Mikhail J. Atallah

Department of Computer Science, Purdue University

Dr. Karthekeyan Chandrasekaran

Department of Industrial and Enterprise Systems Engineering, UIUC

Approved by:

Dr. Voicu Popescu by Dr. William J. Gorman

Head of the School Graduate Program

ACKNOWLEDGMENTS

Firstly, I would like to thank Elena for guiding and supporting me throughout my graduate studies. She inspired me to aim higher and to keep pushing my boundaries. I really appreciate her openness and enthusiasm towards collaboration. I should also thank Elena's spam filter for catching my wrong proofs accurately and thereby saving us a lot of time.

I also thank my all my committee members - Prof. Greg Frederickson, Prof. Samuel Wagstaff, Prof. Mikhail Atallah and Prof. Karthik Chandrasekaran for reading this dissertation and providing helpful feedback. I also thank Prof. Frederickson, Prof. Wagstaff and Prof. Atallah for introducing me to advanced theory courses at Purdue. I also thank Karthik for being a supportive mentor and collaborator. Working with him taught me the systematic approach to problem solving.

I am grateful to have had the opportunity to collaborate with Daniel Dadush, Mahdi Cheraghchi, Sid Jaggi, Badih Ghazi, Samson Zhou, Minshen Zhu, and Clay Thomas. I would like to thank them for their invaluable contributions.

I must also acknowledge all the theory group members who have been instrumental to my success by introducing me to various theory topics through talks, projects and discussions. I would also like to thank Akash for being the oracle that I could query whenever stuck on a problem. Also, thanks to Abhiram, Samson and Abram for being my sounding boards.

I would also like to extend my gratitude to all the faculty, staff and students at Purdue. Especially, Prof. Ananth Grama and Prof. Hemanta Maji for their guidance and valuable words of wisdom which will go a long way. I also thank Dr. William Gorman, Sandra Freeman, Mallus Renate, Tammy Muthig, Jennifer Deno for helping me with administrative affairs. I am also thankful to all my fellow graduate students of the TA suite and the $\mathcal{S}^3$ lab for making my workplace fun and productive.

Finally, I would like to thank my parents and sister for their unconditional love, encouragement and endless patience. A special thanks to Priya for walking together through this journey. Thanks to all my friends for always being there for me. Each one contributed to my success in their own unique way. I would also like to thank Vienna coffee shop and all its staff for providing a comfortable working environment and enough caffeine to take me through my graduate life.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Gandikota, Venkata PhD, Purdue University, August 2017. Local and Global Computation on Algebraic Data. Major Professor: Elena Grigorescu.

Point lattices and *error-correcting codes* are algebraic structures with numerous applications in communication, storage, and cryptography. In this dissertation we study error-correcting codes and lattices in the following models: classical *global* models, in which the algorithm can access its entire input, and *local* models, in which the algorithm has partial access to its input. We design fast algorithms that can reliably recover data from adversarial noise corruption, and show fundamental limitations of computation in these models.

The challenging problems in coding theory revolve around the following nearest-neighbor search abstraction: given a collection of points with some special properties, and a target point, find a special point close to the target. In our case, the collection of special points form lattices or error-correcting codes, and the search problems refer to notions of decoding to a near lattice point, or a near codeword. Some well-known examples, which we also study here in local or global variants, include the Closest Vector Problem, and Bounded Distance Decoding problems.

In the global model, we propose efficient algorithms for solving the Closest Vector Problem in some special Construction-A lattices, a problem known to be hard in general. Further, we make progress on a long-standing open problem regarding decoding Reed-Solomon codes, by showing NP-hardness results for an asymptotic noise range.

Motivated by applications to linear programming and cryptography, we propose the notion of *local testing* for membership in point lattices, extending the classical notion of local testing for error-correcting codes to the Euclidean space. We design local algorithms for classical families of lattices and show several impossibility results.

1 INTRODUCTION

Communication over any channel demands two things in particular, reliability and security. Reliability is achieved by adding structured redundancy to the original message before transmission. Errors introduced by the noisy channel can then be detected and corrected using efficient algorithms. Cryptography, unlike coding theory, relies on the difficulty to retrieve an encrypted message without the knowledge of a secret key, thereby providing secure communication over a channel. Algebraic objects such as codes and lattices have been successfully used to achieve these opposing goals.

The fundamentals of coding theory to achieve reliable communication were laid by Shannon [1] and Hamming [2], in the early 50's. An *error-correcting code* is simply a collection of vectors in $\mathbb{F}^n$, where $\mathbb{F}$ is a finite field. Alternately, a code can also be viewed as a map from messages in $\mathbb{F}^k$ to *codewords* in $\mathbb{F}^n$. A *linear* error-correcting code is a code closed under addition over $\mathbb{F}$.

A *lattice* is an algebraic structure over $\mathbb{R}^n$ similar to a linear error-correcting code. It is a discrete additive subgroup of $\mathbb{R}^n$ that can be viewed as a regular arrangement of points in space. Lattices have been studied since the works of Gauss, Hermite and Minkowski due to their connection to the geometry of numbers [3]. They are used to design efficient algorithms for polynomial factorization over the rationals [4], for solving integer programming problems [5, 6], and in the cryptanalysis of various number theoretic ciphers [7–9]. The interest in lattices from a cryptographic perspective is initiated by Ajtai [10].

In this dissertation, we focus on several fundamental computational problems on lattices and linear error-correcting codes, mostly relevant to their role in communication channels, but also with potential applications to cryptography. In order to effectively communicate over a noisy channel, the sender maps a k-length message, $m \in \mathbb{F}^k$ (or $m \in \mathbb{Z}^k$) to a codeword $c \in \mathbb{F}^n$ (or a lattice point $v \in \mathbb{R}^n$). This process of

mapping a message to a codeword or a lattice point is called *encoding*. The codeword (or lattice point) is then transmitted across the noisy channel, which may introduce some additive noise $e \in \mathbb{F}^n$ (or $e \in \mathbb{R}^n$). On receiving this noise-corrupted message $y = c + e$, the receiver *decodes* it to find the original codeword (lattice point) m.

One of the biggest challenges in coding theory is to design explicit codes with *efficient* encoding and decoding algorithms. The encoding algorithm is usually efficient and depends on the description of the code. Indeed, for linear codes and lattices, this encoding process is captured by a matrix-vector multiplication over the underlying field. Decoding however, is usually more difficult. The complexity of decoding largely depends on the amount of error introduced by the noise in the channel, and on the structure of the underlying code or lattice.

Surprisingly, many classical problems in communication and cryptography are still widely open. In this work we focus on families of codes and lattices that possess rich algebraic structure, and we make progress on several open problems from the literature. We next introduce some formal definitions and discuss our results.

Closest Vector Problem and Nearest Codeword Problem: Given a lattice L, and an arbitrary vector $y \in \mathbb{R}^n$, the Closest Vector Problem (CVP) asks for the closest lattice vector to y. The analogous problem over error-correcting codes is called the Nearest Codeword Problem (NCP) problem. Both CVP and NCP are known to be NP-hard even to approximate up to sub-polynomial factors in general [11–14].

Substantial research has been devoted to finding non-trivial families of lattices and error-correcting codes for which CVP or NCP are easy. The simplest lattice for which CVP is easy is the integer lattice $\mathbb{Z}^n$: indeed, finding the closest lattice vector to a target $y \in \mathbb{R}^n$, amounts to rounding the entries of y to the nearest integer. Extending this observation, it follows that CVP is also easy for lattices which are *equivalent* to $\mathbb{Z}^n$ up to an orthogonal transformation. These family of lattices which are *isomorphic* to $\mathbb{Z}^n$, are known as *orthogonal lattices*. For any orthogonal lattice L, given the orthogonal transformation R which maps L to $\mathbb{Z}^n$, one can easily solve CVP

in L: first apply the orthogonal transformation to the target, solve CVP in $\mathbb{Z}^n$ and then apply the inverse transformation. Therefore, CVP in an orthogonal lattice is easy, *provided the orthogonal transformation is known.* However, given a lattice L, it is not known how to even efficiently verify if there exists an orthogonal transformation which maps L to $\mathbb{Z}^n$. This is the *orthogonality decision problem*:

Question 1.0.1 (Lattice Orthogonality) *Given a lattice $L \in \mathbb{R}^n$, decide if there exists an orthogonal transformation R, such that $R(L) = \mathbb{Z}^n$.*

Deciding if a lattice is equivalent to $\mathbb{Z}^n$, and deciding if a lattice has an orthogonal basis, are special cases of the more general Lattice Isomorphism Problem (LIP). In LIP, given lattices L_1 and L_2 presented by their bases, one is asked to decide if they are isomorphic, meaning if there exists an orthogonal transformation that takes one to the other. LIP has significant cryptographic applications [15] and is known to have a $n^{O(n)}$ algorithm [16]. Earlier, [17] suggested an algorithm for LIP that works well for certain low dimensional lattices. Recent results of [18, 19] show that in certain highly symmetric lattices, isomorphism to $\mathbb{Z}^n$ can be decided efficiently. The complexity of LIP is not well understood, and is part of the broader study of isomorphism between mathematical objects, of which Graph Isomorphism (GI) is a well-known elusive problem [20]. Interestingly, there is a polynomial time reduction from GI to LIP [21].

Our contribution: Given that we do not know how to decide *isomorphism* to $\mathbb{Z}^n$ in general lattices, it is natural to address families of lattices where this problem can be solved efficiently. We focus on the problem of deciding orthogonality for a particular family of lattices, commonly known as Construction-A lattices [22]. Construction-A lattices, L are obtained from a linear error-correcting code C over a finite field of q elements (denoted $\mathbb{F}_q$) as $L = C + q\mathbb{Z}^n$. We resolve the problem of deciding orthogonality in Construction-A lattices for $q = 2$ and $q = 3$ by showing an efficient algorithm. In addition, the algorithm outputs an orthogonal transformation if such a transformation exists. Extending these results to larger values of q appear to require a new set of techniques.

Bounded Distance Decoding: Since CVP/NCP are hard in general, the decoding task is usually relaxed to settings when the amount of error incurred during transmission is known in advance. This is the setting of the Bounded Distance Decoding (BDD) problem, in which the goal is to recover a message corrupted by a bounded amount of error.

Reed-Solomon (RS) codes are a family on polynomial codes for which the BDD problem has been well studied. A Reed-Solomon code of length N, dimension K, defined over a finite field $\mathbb{F}$, is the set of codewords corresponding to evaluations of low-degree univariate polynomials on a given set of evaluation points $\mathcal{D} = \{\alpha_1, \ldots, \alpha_N\} \subseteq \mathbb{F}$. Formally, $RS_{\mathcal{D},K} = \{(p(\alpha_1), \ldots, p(\alpha_N)) : p(x) \in \mathbb{F}[X] \text{ deg(p) } < K\}$. The Hamming distance between $x, y \in \mathbb{F}^N$ is defined as $d_H(x, y) := |\{i \in [N] : \ x_i \neq y_i\}|$. In the BDD, given a target vector $y \in \mathbb{F}^N$ and a distance parameter λ, the goal is to output $c \in \mathcal{C}$ such that $d_H(c, y) \leq \lambda$.

It is well-known that if the number of errors is $\lambda \leq (N - K)/2$, there is a unique codeword within distance λ from the message, which can be found efficiently [23, 24]. Further, Sudan [25] and Guruswami and Sudan [26] show efficient decoding up to $\lambda = N - \sqrt{NK}$ errors (the "Johnson radius"), a setting in which the algorithm may output a small list of possible candidate messages. At the other extreme, if the number of errors is at least $N - K$ (the covering radius), finding one close codeword becomes trivial, amounting to interpolating a degree $K - 1$ polynomial through $\leq K$ points. However, just below that radius, namely at $N - K - 1$ errors, the problem becomes NP-hard, a celebrated result of Guruswami and Vardy [27]. The proof approach of [27] is only applicable to the nearest codeword setting of $N - K - 1$ errors, prompting the fundamental problem of understanding the complexity of BDD in the wide remaining range between $N - \sqrt{NK}$ and $N - K - 1$.

Question 1.0.2 (BDD for Reed-Solomon codes) *Given $\mathcal{D} = \{\alpha_1, \alpha_2, \ldots, \alpha_N\} \subseteq \mathbb{F}$, where $\alpha_i \neq \alpha_j$ for all $i \neq j$, target $y \in \mathbb{F}^N$, a distance parameter λ, and integer $K < N$. Decide if there exists $p \in RS_{\mathcal{D},K}$ such that $d_H(y, p) \leq \lambda$.*

Our contribution: We show the first NP-hardness results for asymptotically smaller decoding radii than the nearest codeword radius of Guruswami and Vardy. Specifically, for Reed-Solomon codes of length N and dimension $K = O(N)$, we show that it is NP-hard to decode more than $\lambda = N - K - O\left(\frac{\log N}{\log \log N}\right)$ errors.

These results follow from the NP-hardness of a generalization of the classical Subset Sum problem to higher moments, called *Moments Subset Sum*, which has been a known open problem, and which may be of independent interest. We further reveal a strong connection with the well-studied Prouhet-Tarry-Escott problem in Number Theory, which turns out to capture a main barrier in extending our techniques.

Testing Membership in a Code or a Lattice: Algorithms for the BDD problem fail to return a codeword or lattice point if the corruption in the received vector is too large. Moreover, these decoding algorithms are computationally expensive. In order to avoid a decoding failure due to the large amount of error, the receiver may want to preprocess the input, by quickly *testing* the amount of corruption beforehand. In this way, if the amount of error is too large, the preprocessing step immediately eliminates the input, while otherwise it decides that there is a good chance of successful decoding and allows the costly decoding step to complete. The model of *Property Testing* formalizes the notion of local testing relevant to the preprocessing step in decoding problems described above. The main underlying goal in Property Testing is to distinguish objects that satisfy a given property from objects that are far from satisfying the property, using few "probes"(queries) into the input.

Linear codes have been well-studied in local models of testing membership. A *locally testable code* (LTC) is defined to be an error correcting code that admits a sublinear-time algorithm that can test for membership in the code, known as a *tester*. Given an oracle access to a target $t \in \mathbb{F}^n$, the tester must accept t if it is a codeword, and must reject with high probability if its distance from the code is large, while making only a small number of queries to the oracle. The number of oracle queries made by the tester is called the *query complexity*. LTCs have found applications in

the construction of Probabilistically Checkable Proofs [28,29] and have been a subject of important research in the last two decades (See [30] for a survey).

Recall that a linear code maps k-length strings to n-length codewords, such that any two codewords are a minimum distance δ apart (known as *distance* of the code). One direction in the study of LTCs tries to understand the tradeoffs between the length of the code (i.e. n), distance of the code (i.e. δ) and the query complexity required to test it (a.k.a. the *locality* of the test). Many well-known families of codes were investigated (see [30] for a survey) in order to understand these tradeoffs, however, it is still not known whether there exist error-correcting codes of small length ($n = O(k)$) and large distance ($\delta = O(n)$) which can be tested with constant number of queries to its coordinates. Since lattices are real analogues of linear codes, it is natural to ask if we can ask similar questions about lattices:

Question 1.0.3 (Locally Testable Lattices) *Can lattices be tested locally?*

Our contributions: We initiate a systematic study of *local testing* for membership in lattices, complementing and building upon the extensive body of work on locally testable codes. In particular, we define a tester for a lattice to be a randomized sublinear-time algorithm that accepts an input which is a lattice point and rejects all inputs which are far from the lattice. This notion of testing membership in lattices is similar to the one relevant to linear codes. The main distinguishing feature is the metric for defining distance. When testing membership in linear codes, distance is captured by Hamming distance, whereas for lattices, the more natural metric is Euclidean distance. Therefore, testing membership in lattices can also be viewed as a natural extension of the study of LTCs to Euclidean spaces.

Similar to LTCs, we study the local testability of lattices with an aim to understand the necessary and sufficient properties of lattices which make them testable with small number of queries. We also investigate some well known families of lattices to understand the tradeoffs between various parameters of the lattice and the query complexity of its tester. Our technical results include the following:

- We show that in order to achieve low query complexity, it is sufficient to design *one-sided non-adaptive* canonical tests. This result is akin to, and based on an analogous result for error-correcting codes due to Ben-Sasson *et al.* (SIAM J. Computing 35(1) pp1–21). We show that any tester for the lattice reduces to the canonical test with almost the same query complexity. The result implies that it is sufficient to focus on designing the canonical tests for proving both lower and upper bounds on the query complexity.

- We demonstrate upper and lower bounds on the query complexity of local testing for membership in *code formula* lattices. These are lattices obtained from a family of nested binary linear codes. We instantiate our results for the code formula lattices constructed from Reed-Muller codes to obtain nearly-matching upper and lower bounds on the query complexity of testing such lattices.

- We show that *knapsack lattices* are not testable in general, but on the other hand, knapsack lattices with bounded coefficients have low-query testers if the inputs are promised to lie in the span of the lattice.

Organization: In Chapter 2 we formally define the various problems on lattices and codes considered in this dissertation and also discuss some related work. We will also setup formal notation to set the stage for chapters that follow. Chapters 3 and 4 focus on problems on linear codes and lattices in the global model of computation. In Chapter 3 we present the result on NP-Hardness of the Bounded Distance Decoding problem in Reed-Solomon codes. This chapter is based on the joint works with Badih Ghazi and Elena Grigorescu [31,32]. Chapter 4 describes an efficient algorithm to decide whether a Construction-A lattice obtained from binary or ternary linear codes is orthogonal or not. The algorithm also gives the orthogonal basis if there exists one. The results in Chapter 4 are based on [33]. In Chapter 5, we move our attention to local models of computation in lattices. We introduce the notion of testing membership in lattices and lay the foundations for further study. The results in this chapter are based on [34].

2 PRELIMINARIES

In this chapter we setup notations and formally define point lattices and linear error-correcting codes. We also define a few common families of lattices and linear codes which are studied in more detail in later chapters.

We denote the set of positive integers up to n by $[n]$, the $n \times n$ identity matrix by I_n and its j^{th} row by e_j. For a vector $b \in \mathbb{R}^n$, let b_j denote its j^{th} coordinate, and $\|b\|$ be its ℓ_2 norm. Denote the inner product of any two vector $x, y \in \mathbb{R}^n$ by $\langle x, y \rangle$. For any matrix M, M^T denotes its transpose. Let $\mathbb{F}_q$ denote a finite field with q elements. We sometimes drop the subscript q from the notation when the field size is clear from the context.

2.1 Linear Error-Correcting Codes

Let q be a prime power. A linear error-correcting code C of length n, and dimension $k \leq n$ over a finite field $\mathbb{F}$ with q elements is a k-dimensional subspace of $\mathbb{F}^n$. Alternately, they can also be defined as the set of all $\mathbb{F}$- linear combinations of the k rows of a generator matrix $G \in \mathbb{F}^{k \times n}$,

$$C = C(G) = \left\{ xG \mid x \in \mathbb{F}^k \right\}.$$

The set of vectors in C are called the codewords. We use hamming distance as the distance measure over codes. The Hamming distance between $x, y \in \mathbb{F}^n$ is defined as $d_H(x, y) := |\{i \in [n] : \ x_i \neq y_i\}|$. The distance of any point $t \in \mathbb{F}^n$ from the code C is defined as the minimum distance of t from any codeword.

$$d_H(t, C) = \min_{c \in C} \{d_H(t, c)\}.$$

We use $\delta(C)$ to denote the minimum distance between any two codewords in C. Since C is a linear code, this is also the length of shortest non-zero codeword in C. We

denote the family of linear codes over of length n, dimension k and distance δ over a finite field of q elements as $[n, k, \delta]_q$ code.

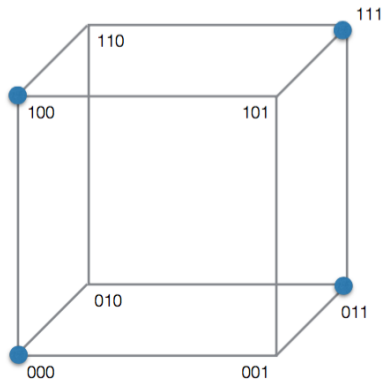

Figure 2.1. Example of a binary linear code $C = \{000, 100, 011, 111\}$

Another important quantity related to linear codes is the rate of the code. Rate of a linear code C measures of amount of redundancy in the code and is defined as

$$r := rate(C) = \frac{k}{n}.$$

Define the dual $C^{\perp}$ of the $[n, k, \delta]_q$ code C to be the set of all vectors in $\mathbb{F}^n$ which are orthogonal to C. Since C is a linear code, $C^{\perp}$ is also linear code over $\mathbb{F}^n$ of dimension $n - k$.

$$C^{\perp} = \{u \in \mathbb{F}^n \mid \langle c, u \rangle = 0 \ \forall \ c \in C\}$$

The generator matrix $H \in \mathbb{F}^{(n-k)\times n}$ for $C^{\perp}$ is called the parity-check matrix of the code C. This gives an alternate parity-check based definition of C as follows:

$$C = \{c \in \mathbb{F}^n \mid Hc^T = 0\}.$$

As mentioned earlier, the goal in coding theory to achieve reliable communication over a noisy channel is to construct codes with small rate and large distance equipped with efficient encoding and decoding algorithms. We note that for linear codes, the encoding map is efficiently computable. Given a $[n, k, \delta]_q$ code generated by G, a

message $m \in \mathbb{F}^k$ is mapped to a codeword $c = mG \in C \subseteq \mathbb{F}^n$. This encoding process takes $O(nk)$ operations and is therefore efficient. One can also efficiently detect an error in the received vector using the parity check matrix H of the underlying code. Given any arbitrary vector $y \in \mathbb{F}^n$, we can efficiently check if y is a codeword by verifying if $Hy^T = 0$ in $O(n(n-k))$ operations.

2.1.1 Some Families of Linear Codes

We now define a few important families of linear codes and prove some properties of these codes which will used in Chapters 3 and 5.

Reed-Solomon Codes: Reed-Solomon codes are an important family of univariate polynomial codes which were first introduced by Irving S. Reed and Gustave Solomon in 1990 [35]. The large distance of these codes, and efficient algorithms to decode in the small error regime made them a good choice for a variety of applications.

Given a set of evaluation points $\mathcal{D} = \{\alpha_1, \alpha_2, \ldots, \alpha_n\} \subseteq \mathbb{F}$, a Reed-Solomon (RS) code of length n, dimension k, is the set of vectors corresponding to evaluations of degree k univariate polynomials in $\mathbb{F}[X]$ over D. Formally,

$$RS_{\mathcal{D},k} = \{(p(\alpha_1), \ldots, p(\alpha_n)) : p \in \mathbb{F}[X],\ \deg(\mathrm{p}) < k\}.$$

The message $m = (m_1, \cdots, m_k) \in \mathbb{F}^k$ corresponds to the coefficients of a univariate polynomial in $\mathbb{F}[X]$ with degree at most k. The generator matrix for $RS_{\mathcal{D},k}$ is given by the following Vandermonde matrix.

$$G = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & & \cdots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \cdots & \alpha_n^{k-1} \end{bmatrix}$$

Since any univariate degree $k-1$ polynomial has at most $k-1$ roots, the weight of the shortest non-zero codeword, and hence the minimum distance $\delta(RS_{\mathcal{D},k}) =$

$n-(k-1)$. Therefore, Reed-Solomon codes are $[n, k, n-k+1]$ codes. The Singleton bound [36] states that any $[n, k, \delta]$ linear code satisfies $k \leq n - \delta + 1$. So, we know that $\delta = n - k + 1$ is the largest possible distance we can obtain for any linear codes of length n and dimension k. This property makes the Reed-Solomon codes amenable to many practical applications.

Reed-Muller Codes: Reed-Muller Codes are a generalization of Reed-Solomon codes to multivariate polynomials. Each codeword of a binary Reed-Muller code $RM(\ell, r) \subseteq \mathbb{F}_2^{2^r}$ corresponds to the evaluations of a degree ℓ, r-variate polynomial polynomial $p(x) \in \mathbb{F}_2[X_1, \cdots, X_r]$ over $\mathbb{F}_2^r$.

$$RM(\ell, r) = \{\langle p(x)\rangle_{x \in \mathbb{F}_2^r} \mid p(x) \in \mathbb{F}_2[X_1, \cdots, X_r],\ \deg(\mathrm{p}) \ < \ell\}.$$

Therefore, $RM(\ell, r)$ codewords are of length $n = 2^r$. The rank of $RM(\ell, r)$ is the number of r-variate polynomials of degree at most ℓ, which given by $k = \sum_{s=0}^{\ell} \binom{r}{s}$ and their minimum distance, $\delta(RS(\ell, r)) = 2^{r-\ell}$.

2.2 Lattices

A n-dimensional lattice is a discrete additive subgroup of $\mathbb{R}^n$. Given m linearly independent vectors $b_i \in \mathbb{R}^n$ as the row vectors of the matrix $B \in \mathbb{R}^{m \times n}$, the lattice $L \subseteq \mathbb{R}^n$ defined by the basis B is the set of all $\mathbb{Z}$-linear combinations of its rows.

$$L = L(B) = \{xB \mid x \in \mathbb{Z}^m\}.$$

If $m = n$, then the lattice is called a full rank lattice. Denote by $span(L)$, the real row-span of B. Note that when L is a full rank lattice, then $span(L) = \mathbb{R}^n$.

We will use ℓ_2-norm as the distance measure for problems defined over lattices, though other ℓ_p are also used for certain applications. The norm of the shortest non-

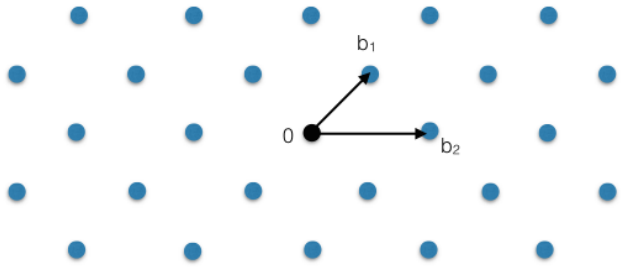

Figure 2.2. Example of a lattice $L \subseteq \mathbb{R}^2$ generated by $B = [b_1, b_2]$.

zero vector in a lattice L is denoted by $\lambda(L)$. For any two vectors, $x, y \in \mathbb{R}^n$, the distance between them is given by

$$d_p(x, y) = \|x - y\|_p. = \left(\sum_{i=1}^{n} (x_i - y_i)^p \right)^{\frac{1}{p}}.$$

The distance of any point t from the lattice L is defined as the minimum distance of t from any lattice vector

$$d_p(t, L) = \min_{v \in L} d_p(t, v).$$

A matrix U is *unimodular* if $U \in \mathbb{Z}^{n \times n}$ and $\det(U) \in \{\pm 1\}$. Any given lattice has multiple bases generating it. Two different bases B_1, B_2 generate the same lattice if and only if there exists a unimodular matrix U such that $B_1 = UB_2$.

Let $\mathcal{O}(n)$ denote the set of distance-preserving linear transformations of $\mathbb{R}^n$. Two n dimensional lattices L_1, L_2 are called *equivalent (or isomorphic)* if there exists an orthogonal transformation $R \in \mathcal{O}(n)$ which maps vectors in L_1 to L_2.

$$L_1 \cong L_2 \iff L_1 = R(L_2), R \in \mathcal{O}(n)$$

The *Hermite Normal Form (HNF) basis* for a full rank lattice $L \subseteq \mathbb{R}^n$ is a square, non-singular, upper triangular matrix $B \subseteq \mathbb{R}^{n \times n}$ such that off-diagonal elements satisfy : $0 \leq B_{i,j} < B_{j,j}$ for all $1 \leq i < j \leq n$.

Fact 2.2.1 *[37] There exists an efficient algorithm which on input a set of rational vectors B, computes a basis for the lattice generated by B: the algorithm simply computes the unique HNF basis of the lattice generated by B.*

The dual lattice $L^{\perp}$ of a lattice $L \in \mathbb{R}^n$ is defined as the set of vectors in $\mathbb{R}^n$ which have an integral inner product with all lattice points.

$$L^{\perp} = \{x \in span(L) \mid \langle x, v \rangle \in Z \ \forall \ v \in L\}.$$

For a lattice L with basis B, the dual lattice $L^{\perp}$ is generated by $B(B^T B)^{-1}$.

Similar to linear error-correcting codes, the encoding and error-detection algorithms for lattices are efficient. For any message $x \in \mathbb{Z}^m$, the encoding procedure for a lattice $L(B)$ generated by basis B is given by $v = xB$. Also a given input $y \in \mathbb{R}^n$ is a lattice point if and only if $Dy^T \in \mathbb{Z}^n$, where D is the generator of the dual lattice $L^{\perp}$. This gives an efficient algorithm to verify if a given input is a lattice point.

2.2.1 Some Families of Lattices

We now define a few important families of lattices and prove some properties which will set the stage for Chapters 4 and 5. For simplicity, in what follows we will slightly abuse notation and use binary code $C \subseteq \{0,1\}^n$ to denote both the code viewed over the field $\mathbb{F}_2 = \{0,1\}$ and the code embedded into $\mathbb{R}^n$ via the trivial embedding $0 \mapsto 0$ and $1 \mapsto 1$. For two sets A and B of vectors we define $A+B := \{a+b \mid a \in A, b \in B\}$.

Construction-A Lattices: The Construction-A of a lattice L_C from a linear error-correcting code $C \subseteq \mathbb{F}_q^n$, where q is a prime, is defined as $L_C := \{c + q \cdot z \mid c \in \phi(C), z \in \mathbb{Z}^n\}$, where ϕ is the (real embedding) mapping $i \in \mathbb{F}_q \mapsto i \in \mathbb{Z}$. Construction-A is often abbreviated as $L_C = C + q\mathbb{Z}^n$. The fact that L_C is a lattice follows from the linearity of C over $\mathbb{F}_q$, [22]. We note that $L_C = C + q\mathbb{Z}^n$ contains $q\mathbb{Z}^n$ as a sublattice and hence it is a full rank lattice.

For any vector $v = (v_1, \cdots, v_n) \in \mathbb{Z}^n$ define $v \bmod q := (v_1 \bmod q, \cdots, v_n \bmod q) \in \mathbb{F}_q^n$. We will now prove some simple properties of Construction-A lattices.

Claim 2.2.2 *Let q be a prime and L be an integral lattice. If $q\mathbb{Z}^n \subseteq L$ then $C = L \bmod q$ is a linear code over $\mathbb{F}_q$.*

Proof Let $v \in L$ and $v = (v \bmod q) + qz$ for some $z \in \mathbb{Z}^n$, where here we abuse notation and view $v \bmod q$ as embedded into the integers, instead of a vector in $\mathbb{F}_q^n$. Since $q\mathbb{Z}^n \subseteq L$, it follows that $v - qz = v \bmod q \in L$. To show that $C = L \bmod q$ is a linear code over $\mathbb{F}_q$, let $c_1, c_2 \in C$. Then $c_1 + c_2 \in L$ (where the addition is over $\mathbb{Z}$), and so $(c_1 + c_2) \bmod q \in C$. ■

Claim 2.2.3 *Let $L = C + q\mathbb{Z}^n$, for some q-ary linear code $C \subseteq \mathbb{F}_q^n$. If $L \cong L_1 \oplus L_2$, and $L_1 \subseteq \mathbb{Z}^k$, then $L_1 \cong C_1 + q\mathbb{Z}^k$ and $L_2 \cong C_2 + q\mathbb{Z}^{n-k}$, for q-ary linear codes C_1 and C_2 that are projections of C on the coordinates corresponding to L_1 and L_2 respectively.*

Fact 2.2.4 *A basis B for the lattice L_C specified by the generator matrix G for the code C can be computed efficiently by taking the HNF of the matrix $\begin{bmatrix} G \\ qI_n \end{bmatrix}$. Conversely, given a basis B of L_C, the generator matrix for C can be computed efficiently by finding a basis for $B \bmod q$ by row reduction over $\mathbb{F}_q$.*

Proof Let L_C be a lattice obtained by Construction-A from a q-ary linear code $C \subseteq \mathbb{F}_q^n$, $L_C = C + q\mathbb{Z}^n$. We first show that given a generator G for the linear code C, the $HNF(\begin{bmatrix} G \\ qI_n \end{bmatrix})$ gives a basis for the lattice L_C.

Let $B = HNF(\begin{bmatrix} G \\ qI_n \end{bmatrix})$. By definition of the HNF basis, B is a basis for the lattice which contains each generator vector $g \in G$ and each qe_j for all $j \in [n]$. We note that each vector $v \in L_C$ is a linear combination of the generators of C and qI_n which is exactly the lattice $L(B)$. Therefore, B is a basis for L_C.

Given a basis B for L_C, we now show that the set of linearly independent vectors in $\mathbb{F}_q^n$ obtained by embedding $B \bmod q$ into $\mathbb{F}_q$ gives a generator for the code C. L_C, contains $q\mathbb{Z}^n$ as a sublattice and form Claim 2.2.2, we can conclude that the code C

is the embedding of L_C mod q into $\mathbb{F}_q$. Since any lattice vector $v \in L_C$, is an integer linear combination of rows of B, all codewords in L_C mod q can be obtained as linear combinations of B mod q over $\mathbb{F}_q$. Therefore, the linearly independent set of vectors in B mod q form a generator for the code C. ■

Code formula lattices: Let $C_0 \subseteq C_1 \subseteq \cdots \subseteq C_{m-1} \subseteq C_m = \mathbb{F}_2^n$ be a family of nested binary linear codes. Then the code formula constructed from the family is defined as

$$C_0 + 2C_1 + \cdots + 2^{m-1}C_{m-1} + 2^m\mathbb{Z}^n.$$

The set of vectors in the Code-Formula are therefore given by

$$\{c_0 + 2c_1 + \cdots + 2^{m-1}c_{m-1} + 2^m z \mid c_i \in C_i, z \in \mathbb{Z}^n\}$$

Here, m is the *height* of the code-formula.

If the family satisfies the *Schur product condition*, namely, $c_1 * c_2 \in C_{i+1}$ for all codewords $c_1, c_2 \in C_i$, where the '*' operator is the coordinate-wise (Schur) product $c_1 * c_2 = (\ (c_1)_i \cdot (c_2)_i\ )_{i\in[n]}$, then the code-formula forms a *Construction-D lattice* (see [22, 38, 39]) and we denote it by $L(\langle C_i\rangle_{i=0}^{m-1})$. Construction-D is a well-known construction of asymptotically good families of lattices from binary linear codes and were primarily used in communication settings, e.g. see Forney [40].

Note that the Code formula lattices of height 1 are just Construction-A lattices. Construction-A lattices have constant rate if the constituent code C_0 has minimum Hamming distance $\Omega(n)$. Unfortunately, these lattices have tiny relative minimum distance (since $2\mathbb{Z}^n$ has constant length vectors). However, code formulas of larger height achieve much better relative distance. In particular, it is easy to see that code formula lattices of height $m \geq \log n$ in which each of the constituent codes C_i has minimum Hamming distance $\Omega(n)$ give asymptotically good families of lattices [22,41].

Knapsack Lattices: Knapsack lattices are a well-known family of lattices and have been investigated in the quest towards lattice-based cryptosystems [7, 42, 43].

We recall that a knapsack lattice is a rank $n-1$ lattice generated by a set of basis vectors $B = \{b_1, \ldots, b_{n-1}\}, b_i \in \mathbb{R}^n$ that are of the form

$$\begin{aligned} b_1 &= (1, 0, \ldots, 0, a_1) \\ b_2 &= (0, 1, \ldots, 0, a_2) \\ &\vdots \\ b_{n-1} &= (0, 0, \ldots, 1, a_{n-1}) \end{aligned}$$

where $a_1, \ldots, a_{n-1}$ are integers. We denote such a knapsack lattice by $L_{a_1,\ldots,a_{n-1}}$.

2.3 Computational Problems on Lattice and Codes

We categorize the various problems defined on lattices and linear codes based on the model of computation used. In the classical *global* model, the algorithm can access its entire input during computation. In the *local* models,the algorithm has partial access to its input, via queries to entries of the input.

2.3.1 Problems in Global Models of Computation:

The Closest Vector Problem (CVP) is one of the most well-studied decoding problems on lattices because of its numerous applications. The analogous problem on codes, the Nearest Codeword Problem (NCP) has also received a lot of attention. We now describe these decoding problems formally and also discuss the slightly relaxed problem of Bounded Distance Decoding (BDD) which has been well-studied in general and also for some particular families of codes and lattices.

Closest Vector Problem and Nearest Codeword Problem:
Given a basis $B \in \mathbb{R}^{m \times n}$ for a lattice and a target vector $t \in \mathbb{R}^n$ as input, the goal in the closest vector problem is to find the closest lattice vector to the target. Formally, we define this problem for any approximation factor $\gamma \geq 1$ as follows:

Definition 2.3.1 (CVP_γ) *Given a basis $B \in \mathbb{R}^{m\times n}$, an approximation factor $\gamma \geq 1$ and a target $t \in \mathbb{R}^n$, find a lattice vector $v \in L(B)$ such that*

$$d_p(t, v) \leq \gamma \cdot d_p(t, L(B)).$$

Analogously, the Nearest Codeword Problem (NCP) can be defined as:

Definition 2.3.2 (NCP_γ) *Given a generator $G \in \mathbb{F}^{k\times n}$ for a linear $[n, k, \delta]_q$ code C, an approximation factor $\gamma \geq 1$ and a target $t \in \mathbb{F}^n$, find a codeword $c \in C$ such that*

$$d_H(t, c) \leq \gamma \cdot d_H(t, C).$$

Both CVP and NCP are known to be NP-Hard upto certain subpolynomial factors [11–14]. For $\gamma > \sqrt{n}$, [44] show that CVP problem is in $NP \cap co - NP$ and therefore unlikely to be NP-Hard.

Bounded Distance Decoding (BDD):

Even though CVP and NCP are hard, most well-known families of lattices and codes possess efficient algorithm to find a lattice point (or codeword) provided the distance of the target to the lattice (or code) is small. This is known as the bounded distance decoding problem.

Definition 2.3.3 (BDD **in Codes**) *Given a target $t \in \mathbb{F}^n$ and a distance parameter r, such that $d_H(t, C) \leq r$, find a codeword $c \in C$ such that*

$$d_H(t, c) \leq r.$$

The problem on lattices is defined analogously with ℓ_p-norm being the distance measure. For general linear codes and lattices, the BDD problem does not have a polynomial time algorithm for distances larger than a constant factor of the minimum distance of the code (or lattice).

A variant of the BDD problem is the list decoding problem where the goal is to obtain a small list of codewords (or lattice vectors) at a distance of r from the target. The list decoding radius is defined as the maximum value r such that there are polynomial number of codewords (or lattice vectors) within that radius.

Isomorphism Problem:

Two lattices L_1, L_2 of same rank are called isomorphic or equivalent if there exists an orthogonal linear transformation R which maps L_1 to L_2. Analogously, two n dimensional linear codes C_1, C_2 defined over $\mathbb{F}$ are equivalent if there exists a permutation of the coordinates which maps C_1 to C_2.

In Lattice Isomorphism Problem (LIP), given lattices L_1 and L_2 presented by their bases, the goal is to decide if they are isomorphic, meaning if there exists an orthogonal transformation that takes one to the other.

Definition 2.3.4 (Lattice Isomorphism Problem (LIP)) *Given two lattices,* $L_1, L_2 \in \mathbb{R}^n$*, decide if* $L_1 \cong L_2$*.*

The analogous problem defined over linear codes is known as the Code Equivalence Problem (CEP).

2.3.2 Local Testing of Codes

The above discussed problems rely on the global properties of the underlying lattices and linear codes. Local models aim at predicting these global properties by sampling a small subset of the input. Note that this is a more restricted model where we are not allowed to look at the entire input therefore, the goal of problems defined in the local models is slightly more relaxed than decoding.

A tester for a linear code C is a randomized sublinear-time algorithm which makes few queries to the coordinates of the input $t \in \mathbb{F}^n$, and decides with high probability if the target is a codeword or far from C. Linear codes equipped with a tester are known as Locally Testable Codes (LTC). Formally, a tester for a linear code is defined as follows:

Definition 2.3.5 (Tester: $\mathbf{T}(q, \epsilon)$) *Given oracle access to $t \in \mathbb{F}^n$, $\epsilon > 0$, a tester T for the code C, makes at most q queries to t and*

- *(Completeness:) Accepts t probability at least $\frac{2}{3}$ if $t \in C$.*
- *(Soundness:) Rejects t with probability at least $\frac{2}{3}$ if $d_H(t, C) \geq \epsilon n$.*

The number of queries, q made by the tester T is called the query complexity of T. Testers which always accept a codeword are known as 1-sided testers. A tester is called adaptive if the $i + 1$ th query made by the tester depends on the answers of the previous i queries.

Slightly more relaxed variant of the above testing model, known as *tolerant testing* was studied by [45–47], in which the tester is allowed to accept inputs which are close to the code. More formally,

Definition 2.3.6 (Tolerant Tester: $\mathbf{T}(q, \epsilon_1, \epsilon_2)$) *Given oracle access to $t \in \mathbb{F}^n$, $\epsilon_2 \geq \epsilon_1 > 0$, a tolerant tester T for the code C, makes at most q queries to t and*

- *(Completeness:) Accepts t probability at least $\frac{2}{3}$ if $d_H(t, C) \leq \epsilon_1 n$.*
- *(Soundness:) Rejects t with probability at least $\frac{2}{3}$ if $d_H(t, C) \geq \epsilon_2 n$.*

The analogous notions of local testing for membership in lattices are introduced in Chapter 5.

3 HARDNESS OF BDD IN REED-SOLOMON CODES

In this chapter, we study the complexity of the decision version of the Bounded Distance Decoding problem for Reed-Solomon codes. In particular, we show that it remains NP-Hard to decode a family of Reed-Solomon codes even if the amount of noise is asymptotically smaller than the covering radius.

Recall from Chapter 2, in the *Bounded Distance Decoding (BDD) problem* for a code $\mathcal{C}$, given a target vector $y \in \mathbb{F}^N$ and a distance parameter r as inputs, the goal is to output $c \in \mathcal{C}$ such that $d_H(c, y) \leq r$. A Reed-Solomon (RS) code of length N, dimension K, defined over a finite field $\mathbb{F}$, is the set of vectors corresponding to evaluations of low-degree univariate polynomials on a given set of evaluation points $\mathcal{D} = \{\alpha_1, \ldots, \alpha_N\} \subseteq \mathbb{F}$, i.e, $RS_{\mathcal{D},K} = \{\langle p(\alpha)\rangle_{\alpha\in\mathcal{D}} : p(x) \in \mathbb{F}[X] \deg(\mathrm{p}) < K\}$. Formally, we define the BDD problem on RS codes with distance parameter d, RS-BDD(d) as follows:

> **Problem:** (RS-BDD(d)) *Bounded Distance Decoding of Reed-Solomon codes with distance parameter d*
>
> **Input:** $\mathcal{D} = \{\alpha_1, \alpha_2, \ldots, \alpha_N\} \subseteq \mathbb{F}$, where $\alpha_i \neq \alpha_j$ for all $i \neq j$, target $y = (y_1, y_2, \ldots, y_N)$, and integer $K < N$
>
> **Goal:** Decide if there exists $p \in RS_{\mathcal{D},K}$ such that $d_H(y, p) \leq (N - K) - d$

We emphasize that the BDD problem above is in fact the basic and natural Polynomial Reconstruction problem, where the input is a set of points $\mathcal{D} = \{(\alpha_i, y_i) \mid i \in [N]\} \subseteq \mathbb{F}^2$. and the goal is to decide if there exists a polynomial p of degree $< K$ that passes through at least $K + d$ points in $\mathcal{D}$.

We state our main result in both forms.

3.1 Results

Our main technical contribution is the first NP-hardness result for BDD of RS codes, for a number of errors that is asymptotically smaller than $N - K$, and its alternative view in terms of polynomial reconstruction.

Theorem 3.1.1 *There exists $c > 0$, such that for every $1 \leq d \leq c \cdot \frac{\log N}{\log \log N}$, the* RS-BDD($d$) *problem for Reed-Solomon codes of length N, dimension $K = N/2 - d + 1$ and field size $|\mathbb{F}| = 2^{\mathsf{poly}(N)}$ is NP-hard. Furthermore, there exists $c > 0$, such that for every $1 \leq d \leq c \cdot \log N$,* RS-BDD($d$) *over fields of size $|\mathbb{F}| = 2^{N^{O(\log N)}}$ does not have $N^{O(\log N)}$-time algorithms unless NP has quasi-polynomial time algorithms.*

Equivalently, there exists $c > 0$, such that for every $1 \leq d \leq c \cdot \frac{\log N}{\log \log N}$, it is NP-hard to decide whether there exists a polynomial of degree $< K = N/2 - d + 1$ passing through $K + d$ many points from a given set $\mathcal{D} = \{(\alpha_1, y_1), (\alpha_2, y_2), \ldots, (\alpha_N, y_N)\} \subseteq \mathbb{F} \times \mathbb{F}$, with $|\mathbb{F}| = 2^{\mathsf{poly}(N)}$. Furthermore, there exists $c > 0$, such that for every $1 \leq d \leq c \cdot \log N$, the same interpolation problem over fields of size $|\mathbb{F}| = 2^{N^{O(\log N)}}$ does not have $N^{O(\log N)}$-time algorithms unless NP has quasi-polynomial time algorithms.

Our results significantly extend [27, 31], which only show NP-hardness for $d \in \{1, 2, 3\}$. As in [27,31], we require the field size to be exponential in N.

The bulk of the proof of Theorem 3.1.1 is showing the NP-hardness of a natural generalization of the classic Subset Sum problem to higher moments, that may be of independent interest.

Problem: (MSS(d)) *Moments Subset Sum with parameter d, over a field $\mathbb{F}$*

Input: Set $A \subseteq \mathbb{F}$ of size $|A| = N$, integer k, elements $m_1, m_2, \ldots, m_d \in \mathbb{F}$

Goal: Decide if there exists $S \subseteq A$ such that $\sum_{s \in S} s^\ell = m_\ell$, for all $\ell \in [d]$, and $|S| = k$.

We note that the reduction from MSS(d) to RS-BDD(d) uses the equivalence between elementary symmetric polynomials and moments polynomials, when the field is of characteristic larger than $\Omega(d!)$(see Lemma 3.2.2 for a formal reduction.)

We point out that the Moments Subset Sum problem has natural analogs over continuous domains in the form of generalized moment problems and truncated moments problems, which arise frequently in economics, operations research, statistics and probability [48].

In this work, we also prove the NP-hardness of the Moments Subset Sum problem for large degrees.

Theorem 3.1.2 *There exists $c > 0$, such that for every $1 \leq d \leq c \cdot \frac{\log N}{\log \log N}$, the Moments Subset Sum problem* $\mathrm{MSS}(d)$ *over prime fields of size $|\mathbb{F}| = 2^{\mathsf{poly}(N)}$ is NP-hard. Furthermore, there exists $c > 0$, such that for every $1 \leq d \leq c \cdot \log N$, the Moments Subset Sum problem* $\mathrm{MSS}(d)$ *over fields of size $|\mathbb{F}| = 2^{N^{O(\log N)}}$ does not have $N^{O(\log N)}$-time algorithms unless NP has quasi-polynomial time algorithms.*

Furthermore, we reveal a connection with the famous Prouhet-Tarry-Escott (PTE) problem in Diophantine Analysis, which is the main barrier for extending Theorem 3.1.2 and Theorem 3.1.1 to $d = \omega(\log N)$, as explained shortly.

The PTE problem [49–51] first appeared in letters between Euler and Goldbach in 1750-1751, and it is an important topic of study in classical number theory (see, e.g., the textbooks of Hardy and Wright [52] and Hua [53]). It is also related to other classical problems in number theory, such as variants of the Waring problem and problems about minimizing the norm of cyclotomic polynomials, considered by Erdös and Szekeres [54, 55].

In the Prouhet-Tarry-Escott problem, given $k \geq 1$, the goal is to find disjoint sets of integers $\{x_1, x_2, \ldots, x_t\}$ and $\{y_1, y_2, \ldots, y_t\}$ satisfying the system:

$$\begin{aligned} x_1 + x_2 + \cdots + x_t &= y_1 + y_2 + \cdots + y_t \\ x_1^2 + x_2^2 + \cdots + x_t^2 &= y_1^2 + y_2^2 + \cdots + y_t^2 \\ &\cdots \\ x_1^k + x_2^k + \cdots + x_t^k &= y_1^k + y_2^k + \cdots + y_t^k. \end{aligned}$$

We call t the size of the PTE solution. It turns out that the completeness proof of our reduction in Theorem 3.1.2 relies on *explicit* solutions to this system for degree

$k = d$ and of size $t = 2^k$. As explained next, despite significant efforts that have been devoted to constructing PTE solutions during the last 100 years, no explicit solutions of size $t = o(2^k)$ are known. This constitutes the main barrier to extending our Theorem 3.1.2 and Theorem 3.1.1 to $d = \omega(\log N)$.

The main open problem that has been tackled in the PTE literature is constructing solutions of small size t compared to the degree k. It is relatively easy to show that $t \geq k+1$, and straightforward (yet non-constructive!) pigeon-hole counting arguments show the existence of solutions with $t = O(k^2)$. If we further impose the constraint that the system is not satisfied for degree $k + 1$ (which is a necessary constraint for our purposes), then solutions of size $t = O(k^2 \log k)$ are known to exist [53]. However, these results are non-constructive, and the only general explicit solutions have size $t = O(2^k)$ (e.g., [51, 55]). A special class of solutions studied in the literature is for $t = k + 1$ (of minimum possible size). Currently there are known explicit parametric constructions of infinitely many minimum-size solutions for $k \leq 12$ (e.g., [55,56]), and finding such solutions often involves numerical simulations and extensive computer-aided searches [56].

From a computational point of view, an important open problem is to understand whether PTE solutions of size $O(k^2)$ (which are known to exist) can be *efficiently constructed*, i.e., in time $\mathsf{poly}(k)$.

We identify the following generalization of the PTE problem as a current barrier to extending our results:

Problem 3.1.3 (Inhomogeneous PTE) *Given a field $\mathbb{F}$, integer d, and $a, b \in \mathbb{F}$, efficiently construct $x_1, \ldots, x_t$, $y_1, \ldots, y_t \in \mathbb{F}$, with $t = o(2^d)$, satisfying:*

$$x_1 + x_2 + \cdots + x_t = y_1 + y_2 + \cdots + y_t$$
$$a^i + \sum_{j=1}^{t} x_j^i = b^i + \sum_{j=1}^{t} y_j^i \quad \forall i \in \{2, \ldots, d\}$$

We believe that this question is worth further study in the theoretical computer science community. In this work, we prove the following theorem, which is at the core of the completeness of our reduction.

Theorem 3.1.4 *There is an explicit construction of solutions for Problem 3.1.3 with* $t = O(2^d)$*, and which can be computed in time* $\mathsf{poly}(t)$*.*

In the next section, we outline the proof of Theorem 3.1.2, and in the process, we explain how PTE solutions of degree d naturally arise when studying the computational complexity of MSS(d).

3.1.1 Proof Overview

To prove Theorem 3.1.2, we begin with the classical reduction from 1-in-3-SAT to Subset-Sum, in which one needs to construct a set of integers such that there is a subset whose sum equals a given target m_1, if and only if there is an assignment that satisfies exactly one literal of each clause of the 3-SAT formula (we refer the reader to Section 3.3 for more details about this standard reduction). Extending this reduction so that the 2nd moment also hits target m_2 raises immediate technical hurdles, since we have very little handle on the extra moment. In [31], the authors manage to handle a reduction for 2nd and 3rd moments via ad-hoc arguments and identities tailored to the degree-2 and degree-3 cases. The problem becomes much more complex as we need to ensure both completeness and soundness for a large number of moments. In this work, we achieve such a reduction where the completeness will rely on explicit solutions to "inhomogeneous PTE instances" and the soundness will rely on a delicate balancing of the magnitudes of these explicit solutions. We now describe the details of this reduction.

For each 1-in-3-SAT variable, we create a collection of *explicit* auxiliary numbers which "stabilize" the contribution of this variable to all i-th moment equations with $2 \leq i \leq d$, while having no net effect on the 1st moment equation. Concretely, if a

and b are the numbers corresponding to the two literals of the given variable, then we need to find numbers $x_1, \ldots, x_t, y_1, \ldots, y_t$ satisfying:

$$x_1 + x_2 + \cdots + x_t = y_1 + y_2 + \cdots + y_t$$
$$a^i + \sum_{j=1}^{t} x_j^i = b^i + \sum_{j=1}^{t} y_j^i \quad \forall i \in \{2, \ldots, d\} \tag{†}$$

Note that in order for the overall reduction to run in polynomial-time, the above auxiliary variables should be *efficiently constructible*. Moreover, we observe that (†) is an inhomogeneous PTE instance (Problem 3.1.3): for $a = b$, it reduces to a PTE instance of degree d. Of course, in our case a and b will not be equal, and (†) is a more general system (and is hence harder to solve) than PTE instances. Nevertheless, as we will see shortly, solving (†) can be essentially reduced to finding explicit PTE solutions of degrees $k \leq d$.

In addition, we need to ensure that the added auxiliary numbers satisfy some "bimodality" property regarding their magnitudes, which would allow the recovery of a satisfying 1-in-3-SAT assignment from any solution to the MSS(d) instance:

Property 3.1.5 (Bimodality (informal)) *Every subset S of the auxiliary variables is such that either $|\sum_{s \in S} s|$ is tiny, or $|\sum_{s \in S} s|$ is huge.*

We note that the existence of explicit and efficiently constructible solutions of small size $t = O(d)$ to system (†) (and hence to a PTE system too) would at least ensure the completeness of a reduction with $d = O(N)$. If soundness can also be ensured for such solutions, then our techniques would extend to radii closer to the Johnson Bound radius.

Overview of Procedure for Solving System (†) We build the variables x_i and y_i recursively, by reducing the construction for degree i to a solution to degree $i - 1$. Towards this goal, we design a sub-procedure, called AtomicSolver, that takes

as inputs an integer $i \in \{2, 3, \ldots, d\}$, and a number R_i, and outputs 2^i rational[1] numbers $\{x_{i,j}, y_{i,j}\}_{j \in [2^{i-1}]}$ that satisfy a PTE system of degree $i-1$, along with a non-homogeneous equation of degree i:

$$\sum_{\ell=1}^{2^{i-1}} (x_{i,\ell}^j - y_{i,\ell}^j) = 0 \quad \forall\ 2 \le j < i, \tag{3.2a}$$

$$\sum_{\ell=1}^{2^{i-1}} (x_{i,\ell}^i - y_{i,\ell}^i) = R_i. \tag{3.2b}$$

We can then run AtomicSolver sequentially on inputs $i \in \{2, \ldots, d\}$ with the R_i input corresponding to a "residual" term that accounts for the contributions to the degree-i equation of the outputs of AtomicSolver(j,R_j) for all $2 \le j < i$, namely,

$$R_i = b^i - a^i + \sum_{2 \le j < i} \sum_{\ell=1}^{2^{j-1}} (y_{j,\ell}^i - x_{j,\ell}^i). \tag{3.3}$$

Note that the aim of the AtomicSolver(i,R_i) procedure is to satisfy the degree-i equation (3.2b) without affecting the lower-degree equations (3.2a).

We then argue that the union $\cup_{2 \le i \le d} \{x_{i,j}, y_{i,j}\}_{j \in [2^{i-1}]}$ of all output variables satisfies the polynomial constraints in (†) with $t = \exp(d)$.

Specifics of the AtomicSolver We next illustrate the AtomicSolver procedure by describing its operation in the particular case where $i = d = 4$. In what follows, we drop "$i = 4$ subscripts" and denote $R = R_4$, $x_\ell = x_{4,\ell}$ and $y_\ell = y_{4,\ell}$ for all $1 \le \ell \le 8$. Then, Equation (3.2b) above that we need to satisfy becomes

$$\sum_{\ell=1}^{8} (x_\ell^4 - y_\ell^4) = R. \tag{3.4}$$

First, we let α be a constant parameter (to be specified later on) and we set

$$x_1 - y_1 = \alpha \tag{3.5a}$$

$$y_2 - x_2 = \alpha \tag{3.5b}$$

[1] In our case, we can afford having *rational* solutions to Equations (3.2a) and (3.2b). Note that this system is still a generalization of the PTE problem since we can always scale the rational solutions by their least common denominator to get a PTE solution of degree $i-1$.

Namely, in Equations (3.5a) and (3.5b), we "*couple*" the ordered pairs (x_1, y_1) and (y_2, x_2) in the same way. Then, using Equations (3.5a) and (3.5b), we substitute $y_1 = x_1 - \alpha$ and $x_2 = y_2 - \alpha$, and the sum of the $\ell = 1$ and $\ell = 2$ terms in Equation (3.4) can be written as

$$(x_1^4 - y_1^4) - (y_2^4 - x_2^4) = p_\alpha(x_1) - p_\alpha(y_2) \tag{3.6}$$

where p_α is a *cubic* polynomial. If we set $x_1 - y_2 = \beta$, then (3.6) further simplifies to

$$p_\alpha(x_1) - p_\alpha(y_2) = q_{\alpha,\beta}(x_1) \tag{3.7}$$

where $q_{\alpha,\beta}$ is a *quadratic* polynomial[2].

In the next step, we couple the ordered tuple (y_3, x_3, y_4, x_4) in the same way that we have so far coupled the tuple (x_1, y_1, x_2, y_2). The sum of the first four terms in the LHS of (3.4) then becomes

$$\begin{aligned} \sum_{\ell=1}^{4}(x_\ell^4 - y_\ell^4) &= (x_1^4 - y_1^4 + x_2^4 - y_2^4) - (y_3^4 - x_3^4 + y_4^4 - x_4^4) \\ &= q_{\alpha,\beta}(x_1) - q_{\alpha,\beta}(y_3). \end{aligned} \tag{3.8}$$

As before, we set $x_1 - y_3 = \gamma$ and (3.8) further simplifies to

$$q_{\alpha,\beta}(x_1) - q_{\alpha,\beta}(y_3) = w_{\alpha,\beta,\gamma}(x_1) \tag{3.9}$$

where $w_{\alpha,\beta,\gamma}(x_1)$ is a *linear* polynomial in x_1. Finally, we couple the ordered tuple $(y_5,\ x_5,\ y_6,\ x_6,\ y_7, x_7, y_8, x_8)$ in the same way that we have so far coupled the tuple $(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$, and we obtain that the following equation is equivalent to Equation (3.4) above:

$$w_{\alpha,\beta,\gamma}(x_1) - w_{\alpha,\beta,\gamma}(y_5) = R. \tag{3.10}$$

Setting $x_1 - y_5 = \theta$, Equation (3.10) further simplifies to

$$\theta \cdot h_{\alpha,\beta,\gamma} = R, \tag{3.11}$$

[2]Intuitively, we can think the LHS of (3.7) (along with the setting $x_1 - y_2 = \beta$) as being a *"derivative operator"*. This explains the fact that we are starting from a cubic polynomial $p_\alpha(\cdot)$ and getting a quadratic polynomial $q_{\alpha,\beta}(\cdot)$. This intuition was also used (twice) in (3.6), and will be again used in (3.9) and (3.10) in order to reduce the degree further.

where $h_{\alpha,\beta,\gamma}$ is the coefficient of x_1 in the linear polynomial $w_{\alpha,\beta,\gamma}(x_1)$. We conclude that to satisfy (3.4), it suffices to choose α, β, γ such that $h_{\alpha,\beta,\gamma} \neq 0$, and to then set $\theta = R/h_{\alpha,\beta,\gamma}$.

It is easy to see that there exist α, β, γ such that $h_{\gamma,\beta,\alpha} \neq 0$, and that the above recursive coupling of the variables guarantees that (3.2a) is satisfied. The more difficult part will be to choose α, β, γ in a way that ensures the soundness of the reduction. This is briefly described next.

Bimodality of Solutions In the above description of the particular case where $i = d = 4$, it can be seen that the produced solutions are $\{0, \pm 1\}$-linear combinations of $\{\alpha, \beta, \gamma, \theta\}$, which are required to satisfy (3.11). It turns out that in this case $h_{\alpha,\beta,\gamma} = 24 \cdot \alpha \cdot \beta \cdot \gamma$, and so (3.11) becomes

$$\theta \cdot \alpha \cdot \beta \cdot \gamma = \frac{R}{24}. \tag{3.12}$$

So assuming we can upper bound $|R|$,[3] we would be able to set θ to a sufficiently large power of 10 while letting α, β and γ to have tiny absolute values and satisfy (3.12). Using the fact that the auxiliary x_i and y_i variables are set to $\{0, \pm 1\}$-linear combinations of $\{\alpha, \beta, \gamma, \theta\}$, this implies that the bimodality property is satisfied. In Section 3.3, we show that the bimodality property ensures that in any feasible solution to MSS(d), the auxiliary variables should have no net contribution to the degree-1 moment equation (Proposition 3.3.4), which then implies the soundness of the reduction.

General Finite Fields We remark that as described above, our solution works over the rational numbers, and, by scaling appropriately, over the integers. By taking the integer solution modulo a large prime p (i.e., $p = 2^{\mathsf{poly}(N)}$) the same arguments extend to $\mathbb{F}_p$. Moving to general finite fields $\mathbb{F} = \mathbb{F}_{p^\ell}$, we first observe that system (†) (and thus a PTE system too) has non-constructive solutions of size $O(d)$, which follows from the Weil bound (see Section 3.6). Our reduction in the proof of Theorem 3.1.2

[3] which we will do by inductively upper bounding $|R_i|$.

also extends to general fields $\mathbb{F} = \mathbb{F}_{p^\ell}$, where p is a prime $p = \Omega(d!)$, and $\ell = \mathsf{poly}(N, d!)$. The reduction now uses a representation of field elements in a polynomial basis $\{1, \gamma, \gamma^2, \ldots, \gamma^{\ell-1}\} \subseteq \mathbb{F}$, instead of decimal representations. See Section 3.7 for the changes that need to be made to the proof over the integers.

3.1.2 Related Work

A number of fundamental works address the polynomial reconstruction problem in various settings. In particular, Goldreich *et al.* [57] show that that the polynomial reconstruction problem is NP-complete for univariate polynomials p over large fields. Håstad's celebrated results [58] imply NP-hardness for linear multivariate polynomials over finite fields. Gopalan *et al.* [59] show NP-hardness for multivariate polynomials of larger degree, over the field $\mathbb{F}_2$.

We note that in general, the polynomial reconstruction problem does not require that the evaluation points are all distinct (i.e., $x_i \neq x_j$ whenever $i \neq j$). This distinction is crucial to the previous results on polynomial reconstruction (eg. [57,59]). It is this distinction that prevents those results from extending to the setting of Reed-Solomon codes, and to their multivariate generalization, Reed-Muller codes.

On the algorithmic side, efficient algorithms for decoding of Reed-Solomon codes and their variants are well-studied. As previously mentioned, [25, 26] gave the first efficient algorithms in the list-decoding regime. Parvaresh and Vardy [60] and Guruswami and Rudra [61] construct capacity achieving codes based on variants of RS codes. Koetter and Vardy [62] propose soft decision decoders for RS codes. More recently, Rudra and Wooters [63] prove polynomial list-bounds for random RS codes.

A related line of work is the study of BDD and of Maximum Likelihood Decoding in general codes, possibly under randomized reductions, and when an unlimited amount of preprocessing of the code is allowed. These problems have been extensively studied under diverse settings, e.g., [13, 27, 64–69].

3.2 Preliminaries

We start by recalling the formal definition of the MSS(d) problem.

Definition 3.2.1 (Moments Subset-Sum: MSS(d)) *Given a set of n elements, $A = \{a_1, \ldots, a_N\}$, $a_i \in \mathbb{F}$, integer k, and $m_1, \ldots, m_d \in \mathbb{F}$, decide if there exists a subset $S \subseteq A$ of size k, satisfying $\sum_{a \in S} a^i = m_i$ for all $i \in [d]$. We call k the size of the* MSS(d) *instance.*

We next recall the reduction from MSS(d) to RS-BDD(d).

Lemma 3.2.2 ([31]) MSS(d) *is polynomial-time reducible to RS-BDD(d). Moreover, the reduction maps instances of* MSS(d) *on N numbers and of size k to Reed-Solomon codes of block length $N+1$ and of dimension $k-d+1$. The reduction holds over finite fields $\mathbb{F}$ of large characteristic.*

The reduction proceeds via SSS(d), a problem which is equivalent to MSS(d) over large fields.

Definition 3.2.3 (Symmetric Subset-Sum (SSS(d))) *Given a set of N distinct elements of $\mathbb{F}$, $A = \{a_1, a_2, \ldots, a_N\}$, integer k, and d targets, $E_1, E_2, \ldots E_d \in \mathbb{F}$, decide if there exists a subset $S \subseteq A$ of size k, such that for every $i \in [d]$ the elementary symmetric sums of the elements of $S = \{s_1, \ldots, s_k\}$ satisfy $E_i(S) = \sum_{1 \leq j_1 < j_2 < \cdots < j_i \leq k} s_{j_1} \ldots s_{j_i} = E_i$.*

Given an instance $\langle A, k, E_1, \ldots, E_d \rangle$ of SSS(d), we construct an instance $\langle \mathcal{D}, y, K \rangle$ of RS-BDD(d) such that there exists a Reed-Solomon codeword $p \in RS_{\mathcal{D},K}$ with $d_H(y, p) \leq (N - K) - d$ if and only if there is a solution to the given instance of SSS(d). SSS(d) can be easily seen to be equivalent to MSS(d) over large prime finite fields $\mathbb{F}$ using Newton's identities [70], which will complete the proof of Lemma 3.2.2. We note that this connection has been previously made (e.g. [71]).

Lemma 3.2.4 SSS(d) *is polynomial-time reducible to* RS-BDD(d).

Proof Given an instance $\langle A, k, E_1, E_2, \ldots, E_d \rangle$ of SSS(d), we construct an instance $\langle \mathcal{D}, y, K \rangle$ of RS-BDD(d) such that there exists a Reed-Solomon codeword $p \in RS_{\mathcal{D},K}$ with $d_H(y, p) \leq (N - K) - d$ if and only if there is a solution to the given instance of SSS(d). Here, $A = \{a_1, a_2, \ldots, a_N\}$ is a set of distinct, non-zero elements of $\mathbb{F}$, $E_1, E_2, \ldots, E_d \in \mathbb{F}$ and $k \in \mathbb{Z}$.

Let $K = k - d + 1$. Define the degree d polynomial $f(x) := x^d - E_1 x^{d-1} + \cdots + (-1)^{d-1} E_{d-1} x$. For each a_i of A, define an element $y_i \in \mathbb{F}$ as $y_i = -f(a_i)$. Define the target vector $y = (y_1, \cdots, y_N, (-1)^d E_d)$. The set $\mathcal{D}$ is then given by $\mathcal{D} = \{a_1^{-1}, \cdots, a_N^{-1}, 0\}$ Note that $\langle \mathcal{D}, y, K \rangle$ is an instance of RS-BDD(d) which can be constructed in polynomial time given the instance $\langle A, k, E_1, \ldots, E_d \rangle$ of SSS(d). Let $D = \{(a_i^{-1}, y_i) \text{ for all } a_i \in A\} \cup \{(0, (-1)^d E_d)\}$. Note that a Reed-Solomon codeword $p \in RS_{\mathcal{D},K}$ at a distance $(N - K) - d$ from y corresponds to a univariate polynomial $p(x)$ of degree at most $K - 1$ which agrees with D in $K + d$ points.

Let S be the solution to SSS(d). We now show that there exists a polynomial of degree at most $k - d$ $(= K - 1)$ which agrees with D in at least $k + 1$ $(= K + d)$ points. Define the following degree k polynomial,

$$g(x) := \prod_{a_i \in S} (x - a_i) = c_0 + c_1 x + \cdots + c_{k-1} x^{k-1} + x^k$$

The coefficients of this polynomial are the symmetric sums of the roots of $g(x)$. Therefore, $c_{k-d} = (-1)^d E_d, \ldots, c_{k-2} = E_2$, and $c_{k-1} = -E_1$. Now define,

$$\begin{aligned} p(x) &= (x^k g(1/x) - x^d f(1/x))/x^d \\ &= c_0 x^{k-d} + c_1 x^{k-d-1} + \cdots + c_{k-d} \end{aligned}$$

and note that $p(x)$ has degree $k - d$. We point out that $g(1/x)$ refers to the rational function obtained by replacing x by $1/x$ in the polynomial $g(x)$. Also, the constant term of this polynomial is $c_{k-d} = (-1)^d E_d$. Hence, $p(0) = (-1)^d E_d$ and since $g(a_i) = 0$, for all $a_i \in S$, it follows that $p(a_i^{-1}) = -f(a_i) = y_i$ for all $a_i \in S$. Therefore, $p(x)$ agrees with $k + 1$ points in D.

Conversely, we now show that if there is a polynomial $p(x)$ of degree at most $K-1$ $(= k-d)$ which agrees with $K+d$ $(= k+1)$ points in D, then there is a solution to SSS(d). We first observe that if a degree $k-d$ polynomial passes through $k+1$ points of D, then it has to pass through $(0, (-1)^d E_d)$. To show this, assume $p(x)$ agrees with $k+1$ points of the form $(a_i^{-1}, y_i) \in D$. Let $g(x)$ be a degree k polynomial defined as,

$$g(x) = x^{k-d}(p(1/x) + f(x))$$

Therefore, if $p(x) = c_0 + c_1 x + \cdots + c_{k-d}x^{k-d}$, $g(x)$ can be written as

$$g(x) = \ x^k + E_1 x^{k-1} + \cdots + (-1)^{d-1} E_{d-1} x^{k-d+1} + c_0 x^{k-d} + c_1 x^{k-d-1} + \cdots + c_{k-d}$$

If $p(a_i^{-1}) = y_i = -f(a_i)$ for $k+1$ points, we have by definition that $g(a_i) = 0$ for these $k+1$ $a_i's$. This is a contradiction since $g(x)$ has degree at most k and it cannot have $k+1$ roots. Therefore, $p(0) = c_0 = (-1)^d E_d$. Also, $g(x)$ has k roots which have their first d symmetric sums equal to $E_1, E_2, \ldots, E_d$ respectively. Hence, there exists a solution to the given instance of SSS(d). ■

Given an instance $\langle A, k, B_1, \cdots, B_d \rangle$ of MSS(d), we can construct an instance $\langle A, k, E_1, \cdots, E_d \rangle$ of SSS(d) by setting

$$E_j = \frac{1}{j!} \begin{vmatrix} B_1 & 1 & 0 & \cdots & \\ B_2 & B_1 & 2 & 0 & \cdots \\ \vdots & & \ddots & \ddots & \\ B_{j-1} & B_{j-2} & \cdots & B_1 & j-1 \\ B_j & B_{j-1} & \cdots & B_2 & B_1 \end{vmatrix} \quad \text{for every } j \in [d].$$

The reduction from MSS(d) to SSS(d) then follows from Newton's identities. We note that this reduction from from MSS(d) to SSS(d) also extends to finite fields $\mathbb{F}$ if $(j!)^{-1} \in \mathbb{F}$.

We will use the 1-in-3-SAT problem in which we are given a 3-SAT formula ϕ on n variables and m clauses and are asked to determine if there exists an assignment $z \in \{0,1\}^n$ satisfying exactly one literal in each clause. It is known that this problem is NP-hard even for $m = O(n)$ [72].

3.3 Reduction from 1-in-3-SAT to MSS(d)

We start proving Theorem 3.1.2 by describing the reduction from from 1-in-3-SAT to MSS(d) and its properties. Henceforth, we denote by 1^ℓ the concatenation of ℓ ones, and we let $(1^\ell)_{10}$ denote the positive integer whose decimal representation is 1^ℓ.

Subset Sum Reduction We start by recalling the reduction from 1-in-3-SAT to Subset-Sum which will be used in our reduction to MSS(d). In that reduction, each variable $(z_t, \overline{z_t})$, $t \in [n]$ is mapped to 2 integers a'_t (corresponding to z_t) and b'_t (corresponding to $\overline{z_t}$). The integers a'_t and b'_t and the target B have the following decimal representation of length-$(n+m)$:

- The decimal representations of a'_t and b'_t consist of two parts: a variable region consisting of the leftmost n digits and a clause region consisting of the (remaining) rightmost m digits.
- In the variable region, a'_t and b'_t have a 1 at the t-th digit and 0's at the other digits. Denote that by $(a_t)'^v$.
- In the clause region, for every $j \in [m]$, a'_t (resp. b'_t) has a 1 at the jth location if z_t (resp. $\overline{z_t}$) appears in clause j, and a 0 otherwise. We denote the clause part of a'_t by $(a_t)'^c$.
- We define $a'_t = 10^m a_t'^v + a_t'^c$. We define b'_t similarly.
- The target B is set to the integer whose decimal representation is the all 1's, i.e., we set $B = 10^m (1^n)_{10} + (1^m)_{10}$.

See Figure 3.1 for an illustration of the decimal representations. This reduction to Subset-Sum is complete and sound. Indeed given a satisfying assignment to the 3-SAT formula $\phi(z)$, the subset $S = \{a'_t \mid t \in [n], z_t = 1\} \cup \{b'_t \mid t \in [n], z_t = 0\}$ is seen to satisfy that $\sum_{s \in S} s = \sum_{\substack{t \in [n] \\ z_t = 1}} a'_t + \sum_{\substack{t \in [n] \\ z_t = 0}} b'_t = B$. Conversely, given a subset $S \subseteq \{a'_t, b'_t \mid$

$t \in [n]\}$ such that $\sum_{s \in S} s = B$, a satisfying assignment to $\phi(z)$ is constructed from it by setting $z_i = 1$ if $a'_t \in S$ and 0 otherwise.

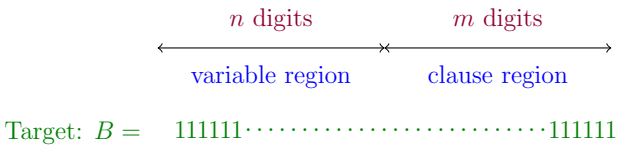

Figure 3.1. Decimal representations in the original reduction from 1-in-3-SAT to Subset-Sum.

Our Reduction from 1-in-3-SAT to MSS(d) An instance of MSS(d) consists of a tuple $\langle A, k, B_1, \ldots, B_d \rangle$. In this reduction, each variable $(z_t, \overline{z_t})$ is mapped to $2^{d+1} - 2$ distinct rationals: $\{a_t\} \cup \{x_{t,i} \mid i \in [2^d - 2]\}$ (corresponding to z_t) and $\{b_t\} \cup \{y_{t,i} \mid i \in [2^d - 2]\}$ (corresponding to $\overline{z}_t$). Let $\{a'_t, b'_t : t \in [n]\}$ be the integers produced by the above reduction to Subset-Sum. We denote by a'^v_t (resp. a'^c_t) the variable (resp. clause) region of a'_t. Let ν be a natural number to be specified later on. Define:

$$\begin{aligned} a_t &:= 10^\nu (10^m a'^v_t + a'^c_t) \quad \text{and,} \\ b_t &:= 10^\nu (10^m b'^v_t + b'^c_t). \end{aligned} \tag{3.13}$$

For each $t \in [n]$, we will explicitly construct two sets of $2^d - 2$ *auxiliary variables*, $X_t = \{x_{t,i} \mid i \in [2^d - 2]\}$ and $Y_t = \{y_{t,i} \mid i \in [2^d - 2]\}$ which satisfy the following properties:

(1) $\sum_{x \in X_t} x = \sum_{y \in Y_t} y = 0.$

(2) $\sum_{x \in X_t} x^k - \sum_{y \in Y_t} y^k = b_t^k - a_t^k$ for every $k \in \{2, \ldots, d\}$.

(3) For any subset $S \subseteq \bigcup_{t \in [n]} (X_t \cup Y_t)$, either $\left| \sum_{s \in S} s \right| > 10^{m+2n+\nu}$ or $\left| \sum_{s \in S} s \right| < 10^\nu$.

(4) Every rational number of $\bigcup_{t\in[n]} (X_t \cup Y_t)$ can be written as a fraction whose numerator and denominator are integers of magnitudes at most $10^{\mathsf{poly}(n,d!)}$. Moreover, $\left|\bigcup_{t\in[n]} (X_t \cup Y_t)\right| = n \cdot (2^{d+1} - 4)$.

Properties (1) and (2) will be used to ensure completeness, Property (3) will be used to ensure soundness, and Property (4) will guarantee the polynomial running-time. Constructing such auxiliary variables forms the crux of the reduction.

Define the set $A = \bigcup_{t\in[n]} (\{a_t\} \cup \{b_t\} \cup X_t \cup Y_t)$. We will observe that $|A| = n(2^{d+1}-2)$ by showing that all the variables $\{a_t\}, \{b_t\}$ and the auxiliary variables in X_t and Y_t for $t \in [n]$ are distinct.

Let $N = |A| = n(2^{d+1} - 2), k = \frac{N}{2}$. The targets $B_1, \ldots, B_d$ are defined as follows:

$$\begin{aligned} B_1 &:= 10^{\nu}(10^m(1^n)_{10} + (1^m)_{10}), \\ B_j &:= \sum_{t=1}^{n} a_t^j + \sum_{t=1}^{n} \sum_{x\in X_t} x^j \text{ for every } j \in \{2, \ldots, d\}. \end{aligned} \tag{3.14}$$

Note that a_t (and b_t and B_1, respectively) defined above are obtained by inserting ν zeros to the right of the decimal representation of a'_t (resp. b'_t and B). Therefore, $a_t = 10^{\nu} \cdot a'_t$. Similarly, $b_t = 10^{\nu} \cdot b'_t$ and $B_1 = 10^{\nu} \cdot B$ (see Figure 3.2 for a pictorial illustration). The following fact is immediate from the definitions,

Fact 3.3.1 *For any $x \in \{a_t, b_t \mid t \in [n]\} \cup B_1$, we have*

$$10^{\nu} < |x| < 10^{m+n+\nu+1}$$

The following lemma is proved using Property (4) (see Section 3.4 for its proof).

Lemma 3.3.2 *For any integer d, the total number of variables in the instance of* MSS(d) *is $N = n \cdot (2^{d+1} - 2)$ and every variable has a $\mathsf{poly}(n, d!)$ digit representation in base* 10.

In Section 3.3.1, we will show how to construct variables satisfying Properties (1), (2), (3) and (4). The proof of Theorem 3.1.2 will follow from the next lemma and Lemma 3.3.2. The proof of Theorem 3.1.1 will then follow from Theorem 3.1.2 and Lemma 3.2.2.

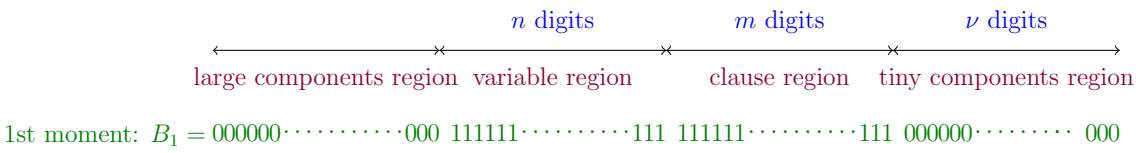

Figure 3.2. Decimal representations in the reduction from 1-in-3-SAT to MSS(d). The "large components region" only contains zeros in $\{a_t, b_t : t \in [n]\}$ but contains non-zeros in $\{|x_{t,i}|, |y_{t,i}| : t \in [n], i \in [2^d - 2]\}$.

Lemma 3.3.3 *(Main) There exists a satisfying assignment to a 3-SAT instance $\phi(z_1, \ldots, z_n)$ if and only if there exists a subset $S \subseteq A$ of size $|S| = n(2^d - 1)$ such that for every $k \in [d]$,*

$$\sum_{s \in S} s^k = B_k.$$

Proof of Theorem 3.1.2 Recall that $N = n(2^{d+1} - 2)$, and so $|S| = |A|/2 = N/2$. From Lemma 3.3.2 above, we know that every element constructed in the instance of MSS(d) has $\mathsf{poly}(n, d!)$ digit representation. Therefore, for $d = O(\frac{\log n}{\log \log n})$, the reduction runs in $\mathsf{poly}(n)$ time.

Let $c > 0$ be a sufficiently small absolute constant. The NP-hardness of MSS(d) for $d < c \log N / \log \log N$ (under polynomial-time reductions) and for $d < c \log N$ (under quasipolynomial time reductions) over the field of rationals then follows from Lemma 3.3.3.

By Lemma 3.3.2 above, we deduce the same hardness results for MSS(d) over prime fields of size $2^{\mathsf{poly}(N)}$. ■

We now prove Lemma 3.3.3.

Proof of Lemma 3.3.3 We start by proving the completeness of our reduction. We show that given a satisfying assignment z to the 3-SAT instance $\phi(z_1, \ldots, z_n)$, there exists a subset $S \subseteq A$ such that for every $k \in [d]$,

$$\sum_{s \in S} s^k = B_k.$$

Consider the following subset S of variables:

$$S \triangleq \bigcup_{t\in[n],z_t=1} \{a_t\} \bigcup_{t\in[n],z_t=1} X_t \bigcup_{t\in[n],z_t=0} \{b_t\} \bigcup_{t\in[n],z_t=0} Y_t.$$

Note that $|S| = n(2^d - 1) = \frac{N}{2}$ since the number of auxiliary variables included in S corresponding to each $t \in [n]$ is exactly $2^d - 2$.

For every $k \in [d]$, we have that

$$\sum_{s\in S} s^k = \sum_{\substack{t\in[n]\\ z_t=1}} \left(a_t^k + \sum_{x\in X_t} x^k \right) + \sum_{\substack{t\in[n]\\ z_t=0}} \left(b_t^k + \sum_{y\in Y_t} y^k \right) \tag{3.15}$$

By Property (2) of the auxiliary variables, we have that for any $t \in [n]$ and any $k \in \{2, 3, \ldots, d\}$,

$$\sum_{x\in X_t} x^k - \sum_{y\in Y_t} y^k = b_t^k - a_t^k.$$

Summing this equation over all $t \in [n]$, such that $z_t = 0$, we get

$$\sum_{\substack{t\in[n]\\ z_t=0}} \left(b_t^k + \sum_{y\in Y_t} y^k \right) = \sum_{\substack{t\in[n]\\ z_t=0}} \left(a_t^k + \sum_{x\in X_t} x^k \right) \tag{3.16}$$

From 3.15 and 3.16, we conclude that for every $k \in \{2, 3, \ldots, d\}$,

$$\sum_{s\in S} s^k = \sum_{t=1}^{n} \left(a_t^k + \sum_{x\in X_t} x^k \right) = B_k$$

For $k = 1$, Property (1) implies that for every $t \in [n]$, $\sum_{x\in X_t} x = 0$ and $\sum_{y\in Y_t} y = 0$. Therefore,

$$\sum_{s\in S} s = \sum_{\substack{t\in[n]\\ z_t=1}} a_t + \sum_{\substack{t\in[n]\\ z_t=0}} b_t \tag{3.17}$$

Recall the variables a'_t, b'_t and B from the Subset Sum reduction defined at the beginning of the proof. Note that $(\sum_{\substack{t\in[n]\\ z_t=1}} a'_t + \sum_{\substack{t\in[n]\\ z_t=0}} b'_t) = B$. Therefore, we can rewrite Equation (3.17) as:

$$\sum_{s\in S} s = 10^\nu \cdot \left(\sum_{\substack{t\in[n]\\ z_t=1}} a'_t + \sum_{\substack{t\in[n]\\ z_t=0}} b'_t \right) = 10^\nu \cdot B = B_1.$$

We now prove the soundness of our reduction. Let S be a solution to the MSS(d) instance. That is, $S \subseteq A$ is such that $\sum_{s \in S} s^k = B_k$ for every $k \in [d]$. Proposition 3.3.4 – which is stated below – shows that the auxiliary variables in S should sum to 0. Therefore, there exists a subset $S' \subseteq \{a_t, b_t \mid t \in [n]\}$ such that $\sum_{s \in S'} s = B_1$. By definition of a_t, b_t and B_1, it follows that there exists a subset of $\{a'_t, b'_t \mid t \in [n]\}$ which sums to B, and the soundness of our reduction then follows from the soundness of the Subset Sum reduction.

Proposition 3.3.4 *Let $S \subseteq A$ be such that $\sum_{s \in S} s = B_1$. Let $D = \bigcup_{t \in [n]} (X_t \cup Y_t)$ be the set of all the auxiliary variables. Then,*

$$\sum_{y \in S \cap D} y = 0.$$

Proof Since $\sum_{s \in S} s = B_1$, we have that

$$\sum_{y \in S \cap D} y + \sum_{s \in S \setminus D} s = B_1.$$

Note that $S \setminus D \subseteq \{a_t, b_t \mid t \in [n]\}$. Since the ν least significant digits of B_1 and those of each element of $S \setminus D$ are all equal to 0, either $\left| B_1 - \sum_{s \in S \setminus D} s \right| = 0$ or $\left| B_1 - \sum_{s \in S \setminus D} s \right| > 10^{\nu}$. If $\left| B_1 - \sum_{s \in S \setminus D} s \right| = 0$, then we are done. Henceforth, we assume that $\left| B_1 - \sum_{s \in S \setminus D} s \right| > 10^{\nu}$. By Fact 3.3.1, the elements of $S \setminus D$ as well as B_1 all have magnitudes at most $10^{m+n+\nu+1}$. Therefore, $\left| B_1 - \sum_{s \in S \setminus D} s \right| \leq (2n+1) \cdot 10^{m+n+\nu+1} < 10^{m+2n+\nu}$. On the other hand, by Property (3) of the auxiliary variables, we know that either $\left| \sum_{y \in S \cap D} y \right| > 10^{m+2n+\nu}$ or $\left| \sum_{y \in S \cap D} y \right| < 10^{\nu}$. Since $\left| \sum_{y \in S \cap D} y \right| = \left| B_1 - \sum_{s \in S \setminus D} s \right|$, we get a contradiction. Therefore, $\sum_{y \in S \cap D} y = 0$. ■

■

3.3.1 Constructing the Auxiliary Variables X_t, Y_t

We now show how to construct the auxiliary variables, starting from the a_t, b_t variables described before, for every $t \in [n]$. We do so in Algorithm 1, the AuxiliaryVariableGenerator. For every $t \in [n]$, we construct $2(2^d - 2)$ distinct auxiliary variables which satisfy the Properties (1), (2), (3) and (4) stated above. The AuxiliaryVariableGenerator outputs the union of the variables generated in Algorithm 2, the AtomicSolver, using the recursive coupling idea described in Section 3.1.1. We use $\mathbf{1}^\ell$ (and $\mathbf{0}^\ell$) to denote a column vector of ℓ 1's (0's) respectively.

Algorithm 1 AuxiliaryVariableGenerator:

Input: $\bigcup_{t\in[n]} \{a_t, b_t\}$

Output: Sets of auxiliary variables X_t, Y_t for every $t \in [n]$.

1: **for** $t \in [n]$ **do**
2: $X_t = \emptyset$
3: $Y_t = \emptyset$
4: **for** $i \in \{2, \ldots, d\}$ **do**
5: $R_{t,i} = (b_t^i - a_t^i) + \sum_{y\in Y_t} y^i - \sum_{x\in X_t} x^i$
6: Let $\{x_{t,i,j} \mid j \in [2^{i-1}]\} \bigcup \{y_{t,i,j} \mid j \in [2^{i-1}]\} =$ AtomicSolver$(t,i, R_{t,i})$
7: Let $X_t = X_t \bigcup \{x_{t,i,j} \mid j \in [2^{i-1}]\}$ and $Y_t = Y_t \bigcup \{y_{t,i,j} \mid j \in [2^{i-1}]\}$
8: **end for**
9: **end for**

We now give the details of AtomicSolver$(t,i, R_{t,i})$ for any $t \in [n]$ and $i \in \{2, 3, \ldots, d\}$. Let $\nu = n^2$, and $M = m+\nu+n+1$. For every $t \in [n], i \in \{2, 3, \ldots, d\}$ and $r \in [i]$, we define the functions $f(t, i) := (i-1)!\cdot\nu_t$ and $g(t, i, r) := (t-1)d^2+(i-1)i+r$, where ν_t is the t^{th} prime integer greater than n^4. Note that $M = O(n^3)$ and $10^M > B_1$, by Fact 3.3.1. We will use the fact that ν_t is much larger than M later. Using the Prime Number Theorem [73], it follows that the number of primes in the interval

$[n^4, n^5]$ is larger than n, and thus $\nu_n < n^5$. Moreover, these n primes can be found in deterministic polynomial time [74].

We will implement the recursive coupling idea of the AtomicSolver described in Section 3.1.1, in terms of matrix algebra. For example, recall that in the first step of the variable coupling, we set $x_1 - y_1 = \alpha$, $y_2 - x_2 = \alpha$ and $x_1 - y_2 = \beta$. We can then express x_1, x_2, y_1, y_2 as a linear combination of α, β, where we use the extra degree of freedom to choose $x_1 = -x_2$, as follows: $(x_1, x_2)^T = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \cdot (\alpha, \beta)^T$, and $(y_1, y_2)^T = \frac{1}{2}\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \cdot (\alpha, \beta)^T$. In general, the polynomial equations give rise to $2^i - 1$ linear constraints on 2^i unknowns $(x_1, \cdots, x_{2^{i-1}}, y_1, \cdots, y_{2^{i-1}})$. The extra degree of freedom allows us to preserve the symmetry of the solution, which enables us to describe the algorithm and its analysis in a clean form.

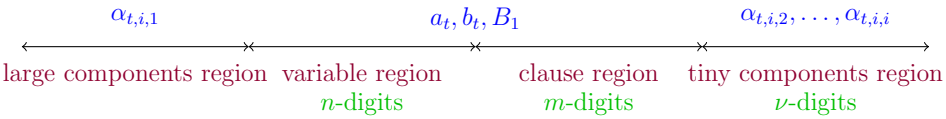

Figure 3.3. Relative magnitudes of $\alpha_{t,i,r}$ for any $i \in \{2, \cdots, d\}$ with respect to a_t, b_t and B_1.

3.4 Verifying Properties

In this section, we prove that the variables generated by the AuxiliaryVariableGenerator satisfy Properties (1), (2), (3), (4). This is done via the following lemmas.

Algorithm 2 AtomicSolver(t,i, $R_{t,i}$):

Input: $i, t, R_{t,i}$

Output: Set of auxiliary variables, $\{x_{t,i,j} \mid j \in [2^{i-1}]\} \bigcup \{y_{t,i,j} \mid j \in [2^{i-1}]\}$

Let ν_t be the t^{th} prime integer greater than n^4

Let $f(t,i) = (i-1)! \cdot \nu_t$

Let $g(t,i,r) = (t-1)d^2 + (i-1)i + r$ for all $1 < r < i$

$\alpha_{t,i,1} = 10^{f(t,i)}$

$\alpha_{t,i,r} = 10^{g(t,i,r)}$ for all $1 < r < i$

$\alpha_{t,i,i} = R_{t,i}/(i! \prod_{r \in [i-1]} \alpha_{t,i,r})$

$\alpha_{t,i} = [\alpha_{t,i,1}, \ldots, \alpha_{t,i,i}]^T$

if $i = 2$ **then**

$$A_2 = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \text{ and } B_2 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

else

$$A_i = \begin{bmatrix} A_{i-1} & \mathbf{1}^{2^{i-2}} \\ B_{i-1} & -\mathbf{1}^{2^{i-2}} \end{bmatrix} \text{ and } B_i = \begin{bmatrix} B_{i-1} & \mathbf{1}^{2^{i-2}} \\ A_{i-1} & -\mathbf{1}^{2^{i-2}} \end{bmatrix}$$

end if

$[x_{t,i,1}, \ldots, x_{t,i,2^{i-1}}]^T = \frac{1}{2} \cdot A_i \cdot \alpha_{t,i}$

$[y_{t,i,1}, \ldots, y_{t,i,2^{i-1}}]^T = \frac{1}{2} \cdot B_i \cdot \alpha_{t,i}$

Return $\{x_{t,i,j} \mid j \in [2^{i-1}]\} \bigcup \{y_{t,i,j} \mid j \in [2^{i-1}]\}$

Lemma 3.4.1 *For every $t \in [n]$, the auxiliary variables satisfy the following conditions*

$$\sum_{x \in X_t} x = \sum_{y \in Y_t} y = 0$$

$$\sum_{x \in X_t} x^k - \sum_{y \in Y_t} y^k = b_t^k - a_t^k \text{ for every } k \in \{2, \ldots, d\}.$$

Lemma 3.4.2 *For any subset* $S \subseteq \bigcup_{t\in[n]} X_t \cup Y_t$ *of the auxiliary variables, either*

$$|\sum_{y\in S} y| > 10^{m+2n+\nu} \quad \textit{or} \quad |\sum_{y\in S} y| < 10^{\nu}.$$

We restate Lemma 3.3.2 from Section 3.3.

Lemma 3.4.3 *For any integer* d*, the total number of variables in the instance of* MSS(d) *is* $N = n \cdot (2^{d+1} - 2)$ *and every variable has a* $\mathsf{poly}(n, d!)$ *digit representation in base* 10*.*

In order to prove Lemma 3.4.1, Lemma 3.4.2 and Lemma 3.3.2 we first state some properties of the auxiliary variables generated by the AtomicSolver(t,i, $R_{t,i}$) and prove them in Section 3.5.

Proposition 3.4.4 *For any* $(t, i) \in [n] \times \{2, \ldots, d\}$, AtomicSolver*($t$,$i$, $R_{t,i}$) on input a rational* $R_{t,i}$*, returns two sets of auxiliary variables* $\{x_{t,i,j} \mid j \in [2^{i-1}]\}$ *and* $\{y_{t,i,j} \mid j \in [2^{i-1}]\}$ *which satisfy:*

$$\sum_{j=1}^{2^{i-1}} (x^i_{t,i,j} - y^i_{t,i,j}) = R_{t,i},$$
$$\sum_{j=1}^{2^{i-1}} (x^k_{t,i,j} - y^k_{t,i,j}) = 0 \textit{ for every } k \in \{1, \ldots, i-1\}.$$

Proposition 3.4.5 *For any* $t \in [n]$*, and* $i \in \{2, 3, \ldots, d\}$*,*

$$\sum_{j=1}^{2^{i-1}} x_{t,i,j} = \sum_{j=1}^{2^{i-1}} y_{t,i,j} = 0.$$

Proposition 3.4.6 *For any $t \in [n], i \in \{2, \ldots, d\}$, we have*

(a) $i! \prod_{r=1}^{i} \alpha_{t,i,r} = R_{t,i}$

(b) $10^{n^4} < \alpha_{t,i,1} < 10^{d!n^5}$

(c) $\alpha_{t,i,r} < 10^{nd^2}$ *for* $1 < r < i-1$

(d) $|\alpha_{t,i,i}| < 2$

(e) $\sum_{r=2}^{i} |\alpha_{t,i,r}| < 10^{\nu - nd}$.

Proposition 3.4.7 *For any $t \in [n]$, $i \in \{2, \ldots, d\}$ and $j \in [2^{i-1}]$, we have that*

$$10^{(i-1)!\nu_t} - 10^{\nu - nd} \leq 2 \cdot |x_{t,i,j}| \leq 10^{(i-1)!\nu_t} + 10^{\nu - nd}.$$

The analogous statement also holds for $y_{t,i,j}$.

Proposition 3.4.8 *We have that:*

1. *For every $(t_1, i_1, j_1) \neq (t_2, i_2, j_2)$, we have that $x_{t_1,i_1,j_1} \neq x_{t_2,i_2,j_2}$.*
2. *For every $(t_1, i_1, j_1) \neq (t_2, i_2, j_2)$, we have that $y_{t_1,i_1,j_1} \neq y_{t_2,i_2,j_2}$.*
3. *For every $(t_1, i_1, j_1), (t_2, i_2, j_2)$, we have that $x_{t_1,i_1,j_1} \neq y_{t_2,i_2,j_2}$.*

3.4.1 Proof of Lemma 3.4.1

We now prove Lemma 3.4.1 which implies Properties (1) and (2) of the auxiliary variables.

Proof of Lemma 3.4.1 From Proposition 3.4.5, we have that for any $t \in [n]$, and $i \in \{2, 3, \ldots, d\}$, $\sum_{j=1}^{2^{i-1}} x_{t,i,j} = \sum_{j=1}^{2^{i-1}} y_{t,i,j} = 0$. Summing the variables over all $i \in \{2, 3, \ldots, d\}$, we get

$$\sum_{x \in X_t} x = \sum_{y \in Y_t} y = 0.$$

For the second part of the lemma, for any $k \in \{2, \ldots, d\}$

$$\sum_{x \in X_t} x^k - \sum_{y \in Y_t} y^k = \sum_{i=2}^{d} \sum_{j=1}^{2^{i-1}} (x^k_{t,i,j} - y^k_{t,i,j})$$
$$= \sum_{i=2}^{k-1} \sum_{j=1}^{2^{i-1}} (x^k_{t,i,j} - y^k_{t,i,j}) + \sum_{j=1}^{2^{k-1}} (x^k_{t,k,j} - y^k_{t,k,j}) + \sum_{i=k+1}^{d} \sum_{j=1}^{2^{i-1}} (x^k_{t,i,j} - y^k_{t,i,j})$$

From the definition of the residual, $R_{t,k}$, the first term, $\sum_{i=2}^{k-1} \sum_{j=1}^{2^{i-1}} (x^k_{t,i,j} - y^k_{t,i,j}) = b^k_t - a^k_t - R_{t,k}$. Also, from Proposition 3.4.4 it follows that $\sum_{j=1}^{2^{k-1}} (x^k_{t,k,j} - y^k_{t,k,j}) = R_{t,k}$ and $\sum_{i=k+1}^{d} \sum_{j=1}^{2^{i-1}} (x^k_{t,i,j} - y^k_{t,i,j}) = 0$. Substituting these values in the above equation, we get,

$$\sum_{x \in X_t} x^k - \sum_{y \in Y_t} y^k = b^k_t - a^k_t$$

■

3.4.2 Proof of Lemma 3.4.2

Before we prove Lemma 3.4.2, we note that each auxiliary variable, $x_{t,i,j}$ and $y_{t,i,j}$ is a $(\pm\frac{1}{2})$-linear combination of the $\alpha_{t,i,r}$ variables. From Proposition 3.4.6 (b), (c) we note that each variable $\alpha_{t,i,r}$ is either of small magnitude, i.e. $|\alpha_{t,i,r}| < 10^{nd^2}$ or of fairly large magnitude, i.e. $|\alpha_{t,i,r}| > 10^{n^4}$. Also, we note that there is only one large magnitude term, i.e., $\alpha_{t,i,1}$, for every pair $(t, i) \in [n] \times \{2, \cdots, d\}$.

Recall that D is the set of all the auxiliary variables

$$D = \{x_{t,i,j}, y_{t,i,j} \mid t \in [n], i \in \{2, 3, \ldots, d\}, j \in [2^{i-1}]\}.$$

For any auxiliary variable $z \in D$, we can split z into terms of the form $\pm\frac{1}{2}\alpha_{t,i,r}$ with large magnitudes and terms with small magnitudes.

$$z = z_U + z_L,$$

where z_U is the term with large magnitude and z_L is the linear combinations of terms with small magnitudes. We now state and prove two properties of the small magnitude sum and the large magnitude sum which will imply the proof of Lemma 3.4.2.

Claim 3.4.9 *For any subset $S \subseteq D$, $\sum_{z \in S} z_L < 10^{\nu}$.*

Proof For any subset $S \subseteq D$,

$$\sum_{z \in S} z_L \leq \frac{1}{2} \sum_{t=1}^{n} \sum_{i=2}^{d} \sum_{r=2}^{i} |\alpha_{t,i,r}|.$$

From Proposition 3.4.6 (e), we know that for any $(t, i) \in [n] \times \{2, \ldots, d\}$, $\sum_{r=2}^{i} |\alpha_{t,i,r}| \leq 10^{\nu - nd}$. Summing over all (t, i), we upper bound the sum of small magnitude terms as follows:

$$\begin{aligned} \sum_{z \in S} z_L &\leq \frac{1}{2} \sum_{t=1}^{n} \sum_{i=2}^{d} \sum_{r=2}^{i} |\alpha_{t,i,r}| \\ &\leq nd \cdot 10^{\nu - nd} \\ &< 10^{\nu} \end{aligned}$$

■

Claim 3.4.10 *Let $S \subseteq D$ such that $\sum_{z \in S} z_U \neq 0$, then $\left| \sum_{z \in S} z_U \right| \geq \frac{1}{2} \cdot 10^{n^4}$.*

Proof We show that for any subset of the auxiliary variables, the contribution of the large magnitudes is either 0, or larger than $\frac{1}{2} \cdot 10^{n^4}$. Note that all the large magnitude terms, i.e., $\alpha_{t,i,1}$ for any (t, i), are powers of 10 larger than n^4 and therefore, each z_U, being a $\pm\frac{1}{2}$ multiple of the large term, is divisible by $\frac{1}{2} \cdot 10^{n^4}$. Thus, the sum $\left| \sum_{z \in S} z_U \right|$ is divisible by $\frac{1}{2} \cdot 10^{n^4}$. If the sum is non-zero, then it is a non-zero multiple of $\frac{1}{2} \cdot 10^{n^4}$ and hence is larger than $\frac{1}{2} \cdot 10^{n^4}$. ■

The proof of Lemma 3.4.2 now follows by combining Claim 3.4.9 and Claim 3.4.10.

Proof of Lemma 3.4.2 For any subset $S \subseteq D$, we can split the sum of the variables as:

$$\sum_{z \in S} z = \sum_{z \in S} z_U + \sum_{z \in S} z_L.$$

If $\sum_{z \in S} z_U \neq 0$, then from Claim 3.4.9 and Claim 3.4.10 we have,

$$\begin{aligned}\left|\sum_{z \in S} z\right| &\geq \left|\sum_{z \in S} z_U\right| - \left|\sum_{z \in S} z_L\right| \\ &\geq \frac{1}{2} \cdot 10^{n^4} - 10^{\nu} = \Omega(10^{n^4}) > 10^{m+2n+\nu} \quad \text{[using the fact that } \nu = n^2\text{]}.\end{aligned}$$

On the other hand, if $\sum_{z \in S} z_U = 0$, then from Claim 3.4.9,

$$\left|\sum_{z \in S} z\right| = \left|\sum_{z \in S} z_L\right| \leq 10^{\nu}.$$

■

3.4.3 Proof of Lemma 3.3.2

Proof of Lemma 3.3.2 In the construction of the instance of MSS(d), we create 2 variables, i.e., a_t, b_t and $2^{d+1} - 4$ auxiliary variables $X_t \cup Y_t$ corresponding to each of the n literals in the 1-in-3 SAT instance. From Claim 3.4.14 below, we know that all variables in the set A are distinct. Therefore, the size of the set A in the instance of MSS(d), $N = n(2^{d+1} - 2)$. Now we show that every element constructed in the instance of MSS(d) has $\mathsf{poly}(n, d!)$ digit representation.

From Fact 3.3.1, Proposition 3.4.7 and Claim 3.4.11 below, we know that the magnitudes of all the numbers generated by the reduction are bounded by $10^{\mathsf{poly}(n,d!)}$. Therefore to complete the proof, it remains to show that the denominators of all the rational numbers in the instance of MSS(d) are also bounded by $10^{\mathsf{poly}(n,d!)}$

Observe from Definition 3.13 that a_t and b_t for every $t \in [n]$ are integers. Also, for any $t \in [n]$ and $i \in \{2, \cdots, d\}$ each $\alpha_{t,i,r}$ for $1 \leq r \leq i-1$ constructed by AtomicSolver(t,i, $R_{t,i}$) is a unique power of 10, and hence an integer, but $\alpha_{t,i,i}$ is

a rational number. Each auxiliary variable generated by AtomicSolver$(t,i,\ R_{t,i})$ is therefore a rational number due to the contribution from $\alpha_{t,i,i}$. From Claim 3.4.13 below, it follows that every rational number in the instance of MSS(d) has magnitude at most $10^{\mathsf{poly}(n,d!)}$ and therefore a $\mathsf{poly}(n,d!)$ digit representation. ■

The following claim bounds the magnitudes of the targets in the MSS(d) instance.

Claim 3.4.11 *For every $k \in \{2,\cdots,d\}$,*

$$|B_k| \leq 10^{k(d!)n^6}.$$

Proof Recall from Definition 3.14,

$$B_k = \sum_{t=1}^{n} a_t^k + \sum_{t=1}^{n}\sum_{x\in X_t} x^k \text{ for every } k \in \{2,\ldots,d\}.$$

Using bounds on the magnitudes of a_t and $x \in X_t$ from Fact 3.3.1 and Proposition 3.4.7 we get

$$\begin{aligned}|B_k| &\leq n(10^{k(m+n+\nu+1)}) + n2^d((10^{(d-1)!\nu_n} + 10^{\nu-nd})^k)\\ &\leq 10^{k(d!)n^6}.\end{aligned}$$

■

We now bound the magnitude of the denominators of $\alpha_{t,i,i}$ for every $(t,i) \in [n] \times \{2,\cdots,d\}$. This bound will be used in Claim 3.4.13 to bound the denominators of all the rational numbers in the instance of MSS(d). Let $D(x)$ denote the irreducible denominator of a rational number x.

Claim 3.4.12 *For any $(t,i) \in [n] \times \{2,\cdots,d\}$,*

$$D(\alpha_{t,i,i}) \leq 10^{(i!)^2\cdot n^6}.$$

Proof The proof proceeds by first obtaining a recursive expression for $D(\alpha_{t,i,i})$, and we then use induction on i to show the bound. Recall the definition of $\alpha_{t,i,i}$ from Algorithm 2,

$$\alpha_{t,i,i} = \frac{R_{t,i}}{i! \prod\limits_{r\in[i-1]} \alpha_{t,i,r}},$$

where $R_{t,i}$ is defined as

$$R_{t,i} = b_t^i - a_t^i + \sum_{u=2}^{i-1} \sum_{v=1}^{2^{u-1}} (y_{t,u,v}^i - x_{t,u,v}^i).$$

Therefore, it follows that the denominator of $\alpha_{t,i,i}$ is bounded by the product of the denominator of $R_{t,i}$ and $i! \cdot \prod_{r=1}^{i-1} \alpha_{t,i,r}$. i.e.,

$$\begin{aligned} D(\alpha_{t,i,i}) &\leq D(R_{t,i}) \cdot (i! \cdot \prod_{r=1}^{i-1} \alpha_{t,i,r}) \\ &= D(R_{t,i}) \cdot (i! \cdot 10^{(i-1)!\nu_t + \sum_{r=2}^{i-1} g(t,i,r)}) \\ &\leq D(R_{t,i}) \cdot (i! \cdot 10^{(i-1)!n^5 + nd^3}) \end{aligned}$$

The last inequality follows from the fact that $\sum_{r=2}^{i-1} g(t,i,r) = \sum_{r=2}^{i-1} (t-1)d^2 + (i-1)i + r \leq td^3$ for all $2 \leq i \leq d$ and $\nu_t < n^5$ for any $t \in [n]$. We now obtain an expression for $D(R_{t,i})$. Since b_t and a_t are both integers, note that $D(R_{t,i}) = D\left(\sum_{u=2}^{i-1} \sum_{v=1}^{2^{u-1}} (y_{t,u,v}^i - x_{t,u,v}^i)\right)$. Also, recall that all the auxiliary variables obtained from a given AtomicSolver$(t,u,\ R_{t,u})$, described in Algorithm 2, have the same denominator to which $D(\alpha_{t,u,u})$ contributes, i.e., $D(x_{t,u,v}) = D(y_{t,u,v}) = 2 \cdot D(\alpha_{t,u,u})$, for all $v \in [2^{u-1}]$. Therefore, $D(y_{t,u,v}^i - x_{t,u,v}^i) = 2^i \cdot D(\alpha_{t,u,u}^i)$. From this observation, it follows that $D(\sum_{v=1}^{2^{u-1}} y_{t,u,v}^i - x_{t,u,v}^i) = 2^i \cdot D(\alpha_{t,u,u}^i)$, and we get an expression for $D(R_{t,i})$ as follows:

$$D(R_{t,i}) = LCM(\{2^i \cdot D(\alpha_{t,u,u}^i) \mid u \in \{2, \cdots, i-1\}\}) \leq 2^{i^2} \cdot \prod_{u=2}^{i-1} D(\alpha_{t,u,u}^i).$$

Substituting the above expression for $D(R_{t,i})$ back in the expression obtained for $D(\alpha_{t,i,i})$, we get

$$D(\alpha_{t,i,i}) \leq \left(\prod_{u=2}^{i-1} D(\alpha_{t,u,u}^i)\right) \cdot (2^{i^2} \cdot i! \cdot 10^{(i-1)!n^5 + nd^3}) \tag{3.18}$$

We now use induction on i to show that that $D(\alpha_{t,i,i}) \leq 10^{(i!)^2 \cdot n^6}$ for every $i \in \{2, \cdots, d\}$. For the base case, $i = 2$, from definitions we know that

$$D(\alpha_{t,2,2}) = 2 \cdot 10^{\nu_t} < 10^{n^6}$$

Let us assume the induction hypothesis that for all $i < \ell \leq d$,

$$D(\alpha_{t,i,i}) \leq 10^{(i!)^2 \cdot n^6}.$$

From Equation 3.18, we know that

$$\begin{aligned}
D(\alpha_{t,\ell,\ell}) &\leq \left(\prod_{u=2}^{\ell-1} D(\alpha_{t,u,u}^{\ell}) \right) \cdot (2^{\ell^2} \cdot \ell! \cdot 10^{(\ell-1)! n^5 + nd^3}) \\
&\leq \left(\prod_{u=2}^{\ell-1} (10^{(u!)^2 \cdot n^6})^{\ell} \right) \cdot (10^{(\ell-1)! n^5 + nd^3 + 2\ell^2}) \\
&\leq 10^{\ell \sum_{u=2}^{\ell-1} ((u!)^2 \cdot n^6) + (\ell)! n^5 + nd^3 + 2\ell^2} \\
&\leq 10^{\ell \cdot (\ell-1) \cdot (\ell-1)!^2 \cdot n^6 + (\ell)! n^5 + nd^3 + 2\ell^2} \\
&\leq 10^{(\ell!)^2 \cdot n^6},
\end{aligned}$$

where the last inequality follows from the fact that $\ell(\ell-1)!^2 n^6 > (\ell)! n^5 + nd^3 + 2\ell^2$ for any $\ell \leq d$. ■

Claim 3.4.13 *For any $x \in A \bigcup \{B_1, \cdots, B_d\}$,*

$$D(x) < 10^{\mathsf{poly}(n, d!)}.$$

Proof We first observe that the elements constructed from the 3-SAT clauses and variables are all integers. So, $D(a_t) = D(b_t) = 1$ for all $t \in [n]$. Next, we argue about the denominators of the auxiliary variables and show that they are all bounded by $2 \cdot 10^{(d!)^2 \cdot n^6}$. Consider the set of auxiliary variables generated by AtomicSolver(t,i, $R_{t,i}$) for some $t \in [n]$ and $i \in \{2, 3, \cdots, d\}$. Each $x_{t,i,j}$ (or $y_{t,i,j}$) is a $\pm\frac{1}{2}$-linear combination of the $\{\alpha_{t,i,r} \mid r \in [i]\}$ variables. From the definitions in Algorithm 2, we note that all $\alpha_{t,i,r}$ variables constructed by the AtomicSolver are integers except for $\alpha_{t,i,i}$. Therefore, each $x_{t,i,j}$ and $y_{t,i,j}$ have the same denominator as $\alpha_{t,i,i}/2$. Using Claim 3.4.12, we get that for every (t,i), $D(\alpha_{t,i,i,}) \leq 10^{(i!)^2 \cdot n^6}$. Therefore, for any $j \in [2^{i-1}]$, $D(x_{t,i,j}) < 2 \cdot 10^{(i!)^2 \cdot n^6}$. A similar argument applies to $y_{t,i,j}$.

We now bound the magnitudes of the denominators of the target, $B_1, \cdots, B_d$ defined in the MSS(d) instance. Recall from Definition 3.14 that B_1 is an integer. Therefore, $D(B_1) = 1$. All other targets are rational numbers defined as

$$B_k = \sum_{t=1}^{n} a_t^k + \sum_{t=1}^{n} \sum_{x \in X_t} x^k \text{ for every } k \in \{2, \ldots, d\}.$$

The denominator of B_k is defined by the denominator of the sum, $\sum_{t=1}^{n} \sum_{x \in X_t} x^k$. This sum can be expanded as $\sum_{t=1}^{n} \sum_{i=2}^{d} \sum_{j=1}^{2^{i-1}} x^k$. From the fact that $D(\sum_{j=1}^{2^{i-1}} x^k) = D(\alpha_{t,i,i}^k)$ and Claim 3.4.12, we get,

$$\begin{aligned} D(B_k) &\leq \prod_{\substack{t \in [n] \\ i \in \{2\ldots,d\}}} D(\alpha_{t,i,i}^k) \\ &\leq \prod_{\substack{t \in [n] \\ i \in \{2\ldots,d\}}} 10^{k(i!)^2 \cdot n^6} \\ &\leq (10^{k(d!)^2 \cdot n^6})^{nd} = 10^{kd(d!)^2 \cdot n^7} \end{aligned}$$

Therefore, we conclude that every element of the instance of MSS(d) constructed by the reduction has a denominator of magnitude at most $10^{\mathsf{poly}(n,d!)}$. ■

Claim 3.4.14 *All variables in the set A are distinct.*

Proof From Proposition 3.4.8, we know that all auxiliary variables are distinct. Also, the distinctness of the variables $\{a_t, b_t \mid t \in [n]\}$ follows from the construction. The only thing that remains to show is that all the auxiliary variables are different from $\{a_t, b_t \mid t \in [n]\}$.

We show this fact by comparing the magnitudes of the two sets of variables. From Fact 3.3.1, we know that $|v| < 10^{m+n+\nu+1}$ for every $v \in \{a_t, b_t \mid t \in [n]\}$, and from Proposition 3.4.7, we know that all auxiliary variables are larger than $10^{\nu_1} - 10^{\nu - nd} > 10^{m+n+\nu+1}$. Therefore the two sets of variables are disjoint. ■

3.5 Proofs of the Helper Propositions

In this section, we prove the helper claims stated in the previous section.

Proof of Proposition 3.4.4 We first show a structural property of the auxiliary variables generated by any AtomicSolver. The proof of Proposition 3.4.4 follows from it.

Claim 3.5.1 *For any $i \in \{2, \cdots, d\}$, Let A_i, B_i are matrices defined in the* AtomicSolver *and let $\{\alpha_r \mid r \in [i]\}$ be some rational numbers. If*

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{2^{i-1}} \end{bmatrix} = \frac{1}{2} \cdot A_i \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_i \end{bmatrix}, \text{ and } \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{2^{i-1}} \end{bmatrix} = \frac{1}{2} \cdot B_i \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_i \end{bmatrix},$$

then $\{x_j \mid j \in [2^{i-1}]\}$ and $\{y_j \mid j \in [2^{i-1}]\}$ satisfy:

$$\sum_{j=1}^{2^{i-1}} (x_j^k - y_j^k) = 0 \text{ for every } k \in \{1, \ldots, i-1\}$$
$$\sum_{j=1}^{2^{i-1}} (x_j^i - y_j^i) = i! \prod_{r=1}^{i} \alpha_r.$$

Proof We use induction on i. For the base case, consider $i = 2$. From the definition of A_2 and B_2 we get,

$$x_1 = \frac{\alpha_1}{2} + \frac{\alpha_2}{2} \qquad x_2 = -\frac{\alpha_1}{2} - \frac{\alpha_2}{2}$$
$$y_1 = \frac{\alpha_1}{2} - \frac{\alpha_2}{2} \qquad y_2 = -\frac{\alpha_1}{2} + \frac{\alpha_2}{2}$$

Therefore,

$$x_1 + x_2 - y_1 - y_2 = 0$$
$$x_1^2 + x_2^2 - y_1^2 - y_2^2 = 2 \cdot \alpha_1 \cdot \alpha_2$$

and the claim holds for $i = 2$.

Let us assume the induction hypothesis for all $i < \ell \leq d$. For $i = \ell$, we have,

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{2^{\ell-1}} \end{bmatrix} = \frac{1}{2} \cdot A_\ell \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_\ell \end{bmatrix} \text{ and } \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{2^{\ell-1}} \end{bmatrix} = \frac{1}{2} \cdot B_\ell \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_\ell \end{bmatrix}$$

From the recursive definitions of the matrices A_ℓ, B_ℓ in Algorithm 2, we can split the above equations as

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{2^{\ell-2}} \end{bmatrix} = \frac{1}{2} \cdot A_{\ell-1} \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{\ell-1} \end{bmatrix} + \frac{1}{2} \cdot \begin{bmatrix} \alpha_\ell \\ \alpha_\ell \\ \vdots \\ \alpha_\ell \end{bmatrix}$$

$$\begin{bmatrix} x_{2^{\ell-2}+1} \\ x_{2^{\ell-2}+2} \\ \vdots \\ x_{2^{\ell-1}} \end{bmatrix} = \frac{1}{2} \cdot B_{\ell-1} \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{\ell-1} \end{bmatrix} - \frac{1}{2} \cdot \begin{bmatrix} \alpha_\ell \\ \alpha_\ell \\ \vdots \\ \alpha_\ell \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{2^{\ell-2}} \end{bmatrix} = \frac{1}{2} \cdot B_{\ell-1} \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{\ell-1} \end{bmatrix} + \frac{1}{2} \cdot \begin{bmatrix} \alpha_\ell \\ \alpha_\ell \\ \vdots \\ \alpha_\ell \end{bmatrix}$$

$$\begin{bmatrix} y_{2^{\ell-2}+1} \\ y_{2^{\ell-2}+2} \\ \vdots \\ y_{2^{\ell-1}} \end{bmatrix} = \frac{1}{2} \cdot A_{\ell-1} \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{\ell-1} \end{bmatrix} - \frac{1}{2} \cdot \begin{bmatrix} \alpha_\ell \\ \alpha_\ell \\ \vdots \\ \alpha_\ell \end{bmatrix}$$

Equivalently, they can be rewritten as

$$x_j = \begin{cases} x'_j + \frac{1}{2} \cdot \alpha_\ell \text{ if } j \leq 2^{\ell-2} \\ y'_{j-2^{\ell-2}} - \frac{1}{2} \cdot \alpha_\ell \text{ if } j > 2^{\ell-2} \end{cases}$$

Similarly,

$$y_j = \begin{cases} y'_j + \frac{1}{2} \cdot \alpha_\ell \text{ if } j \leq 2^{\ell-2} \\ x'_{j-2^{\ell-2}} - \frac{1}{2} \cdot \alpha_\ell \text{ if } j > 2^{\ell-2} \end{cases}$$

where, the $\{x'_j, y'_j \mid j \in [2^{\ell-2}]\}$ by induction hypothesis satisfy

$$\sum_{j=1}^{2^{\ell-2}} (x_j'^{\ell-1} - y_j'^{\ell-1}) = (\ell-1)! \prod_{r=1}^{\ell-1} \alpha_r$$

$$\sum_{j=1}^{2^{\ell-2}} (x_j'^{k} - y_j'^{k}) = 0 \text{ for every } k \in \{1, \ldots, \ell-2\}.$$

Therefore, for any $k \in \mathbb{N}$, we get that

$$\begin{aligned}
\sum_{j=1}^{2^{\ell-1}} (x_j^k - y_j^k) =& \sum_{j=1}^{2^{\ell-2}} (x'_j + \frac{1}{2} \cdot \alpha_\ell)^k - (y'_j + \frac{1}{2} \cdot \alpha_\ell)^k \\
&+ \sum_{j=2^{\ell-2}+1}^{2^{\ell-1}} (y'_{j-2^{\ell-2}} - \frac{1}{2} \cdot \alpha_\ell)^k - (x'_{j-2^{\ell-2}} - \frac{1}{2} \cdot \alpha_\ell)^k \\
=& \sum_{j=1}^{2^{\ell-2}} (x'_j + \frac{1}{2} \cdot \alpha_\ell)^k - (x'_j - \frac{1}{2} \cdot \alpha_\ell)^k \\
&- \sum_{j=1}^{2^{\ell-2}} (y'_j + \frac{1}{2} \cdot \alpha_\ell)^k - (y'_j - \frac{1}{2} \cdot \alpha_\ell)^k \\
=& \sum_{j=1}^{2^{\ell-2}} \left(2 \sum_{\substack{r=0 \\ r \equiv 1 \bmod 2}}^{k} \frac{1}{2^r} \cdot \binom{k}{r} x_j'^{k-r} \alpha_\ell^r \right) \\
&- \sum_{j=1}^{2^{\ell-2}} \left(2 \sum_{\substack{r=0 \\ r \equiv 1 \bmod 2}}^{k} \frac{1}{2^r} \cdot \binom{k}{r} y_j'^{k-r} \alpha_\ell^r \right) \\
=& \sum_{j=1}^{2^{\ell-2}} \left(2 \sum_{\substack{r=0 \\ r \equiv 1 \bmod 2}}^{k} \frac{1}{2^r} \cdot \binom{k}{r} (x_j'^{k-r} - y_j'^{k-r}) \alpha_\ell^r \right)
\end{aligned}$$

Observe that, for all $k \leq \ell - 1$, $k - r \leq \ell - 2$, since $r \equiv 1 \bmod 2$. Therefore, for all $k \leq \ell - 1$, from induction hypothesis, we have, $(x_j'^{k-r} - y_j'^{k-r}) = 0$. And,

$$\sum_{j=1}^{2^{\ell-1}} (x_j^k - y_j^k) = 0.$$

For $k = \ell$,

$$\begin{aligned}\sum_{j=1}^{2^{\ell-1}}(x_j^\ell - y_j^\ell) =& \sum_{j=1}^{2^{\ell-2}}\left(2\sum_{\substack{r=0\\ r\equiv 1 \bmod 2}}^{\ell}\frac{1}{2^r}\cdot\binom{\ell}{r}(x_j'^{\ell-r} - y_j'^{\ell-r})\alpha_\ell^r\right)\\ =& 2\cdot\frac{1}{2}\cdot\binom{\ell}{1}\cdot\alpha_\ell\sum_{j=1}^{2^{\ell-2}}(x_j'^{\ell-1} - y_j'^{\ell-1})\\ =& \ell\cdot(\ell-1)!\cdot\prod_{r=1}^{\ell-1}\alpha_r\cdot\alpha_\ell\\ =& \ell!\cdot\prod_{r=1}^{\ell}\alpha_r.\end{aligned}$$

■

Note that Claim 3.5.1 is independent of t and the choice of the α variables. Recall the construction of AtomicSolver$(t,i,\ R_{t,i})$ for any $(t,i) \in [n]\times\{2,\cdots,d\}$. It returns two sets of auxiliary variables $\{x_{t,i,j} \mid j \in [2^{i-1}]\}$ and $\{y_{t,i,j} \mid j \in [2^{i-1}]\}$ which are constructed using matrices A_i and B_i. From Claim 3.5.1, it then follows that these auxiliary variables satisfy:

$$\begin{aligned}&\sum_{j=1}^{2^{i-1}}(x_{tij}^i - y_{tij}^i) = i!\prod_{r=1}^{i}\alpha_{t,i,r}\\ &\sum_{j=1}^{2^{i-1}}(x_{tij}^k - y_{tij}^k) = 0 \text{ for every } k \in \{1,\ldots,i-1\}\end{aligned}$$

Using Proposition 3.4.6 ((a)), we get,

$$\sum_{j=1}^{2^{i-1}}(x_{tij}^i - y_{tij}^i) = b_t^i - a_t^i + R_{t,i}$$

■

Proof of Proposition 3.4.5 The proof uses the recursive structure of the matrices A_i and B_i. Recall that $\mathbf{1}^\ell$ denotes a vector of ℓ ones, and $\mathbf{0}^\ell$ denotes a vector of ℓ zeros. Note that for any $(t,i) \in [n]\times\{2,\cdots,d\}$,

$$\sum_{j=1}^{2^{i-1}} x_{t,i,j} = \frac{1}{2} \cdot (\mathbf{1}^{2^{i-1}})^T \cdot A_i \cdot \begin{bmatrix} \alpha_{t,i,1} & \cdots & \alpha_{t,i,i} \end{bmatrix}^T.$$

Similarly, the sum of all the $\{y_{t,i,j} \mid j \in [2^{i-1}]\}$ can be written as

$$\sum_{j=1}^{2^{i-1}} y_{t,i,j} = \frac{1}{2} \cdot (\mathbf{1}^{2^{i-1}})^T \cdot B_i \cdot \begin{bmatrix} \alpha_{t,i,1} & \cdots & \alpha_{t,i,i} \end{bmatrix}^T.$$

We show by induction on $i \geq 2$ that

$$(\mathbf{1}^{2^{i-1}})^T \cdot A_i = (\mathbf{0}^i)^T \text{ and } (\mathbf{1}^{2^{i-1}})^T \cdot B_i = (\mathbf{0}^i)^T.$$

For the base case, $i = 2$, it can be verified that

$$\begin{bmatrix} 1 & 1 \end{bmatrix} \cdot A_2 = \begin{bmatrix} 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix} \text{ and}$$

$$\begin{bmatrix} 1 & 1 \end{bmatrix} \cdot B_2 = \begin{bmatrix} 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

Let us assume the induction hypothesis for all $i < \ell \leq d$. For $i = \ell$, observe that

$$\begin{aligned} (\mathbf{1}^{2^{\ell-1}})^T \cdot A_i &= \begin{bmatrix} (\mathbf{1}^{2^{\ell-2}})^T & (\mathbf{1}^{2^{\ell-2}})^T \end{bmatrix} \cdot \begin{bmatrix} A_{\ell-1} & \mathbf{1}^{2^{\ell-2}} \\ B_{i-1} & -\mathbf{1}^{2^{\ell-2}} \end{bmatrix} \\ &= \begin{bmatrix} (\mathbf{1}^{2^{\ell-2}})^T \cdot A_{\ell-1} + (\mathbf{1}^{2^{\ell-2}})^T \cdot B_{\ell-1} & 0 \end{bmatrix} \end{aligned}$$

By the induction hypothesis, we know that $(\mathbf{1}^{2^{\ell-2}})^T \cdot A_{\ell-1} + (\mathbf{1}^{2^{\ell-2}})^T \cdot B_{\ell-1} = (\mathbf{0}^{\ell-1})^T$, Therefore,

$$(\mathbf{1}^{2^{\ell-1}})^T \cdot A_\ell = \begin{bmatrix} \mathbf{0}^{\ell-1} & 0 \end{bmatrix}$$

Similarly,

$$(\mathbf{1}^{2^{\ell-1}})^T \cdot B_\ell = \begin{bmatrix} (\mathbf{1}^{2^{\ell-2}})^T \cdot B_{\ell-1} + (\mathbf{1}^{2^{\ell-2}})^T \cdot A_{\ell-1} & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{0}^{\ell-1} & 0 \end{bmatrix}$$

■

We now show certain bounds on the magnitudes of $\alpha_{t,i,r}$ and hence on the auxiliary variables $x_{t,i,j}, y_{t,i,j}$. For any two tuples of same dimensions, we say that $(p_1, p_2, \cdots, p_d) > (q_1, q_2, \cdots, q_d)$ if there is an $i \in [d]$ such that $p_i > q_i$ and $p_j = q_j$ for all $j < i$. In order to prove 3.4.6, we will need the following claim.

Claim 3.5.2 *For any $t \in [n]$, $i \in \{2, \ldots, d\}$ and any $j \in [2^{i-1}]$,*

$$|x_{t,i,j} - y_{t,i,j}| = \alpha_{t,i,2}$$

Proof We use the recursive matrix definitions to show that for every $t, i, j \in [n] \times \{2, \cdots, d\} \times [2^{i-1}]$,

$$|x_{t,i,j} - y_{t,i,j}| = \alpha_{t,i,2}$$

We use induction on i.

For the base case, $i = 2$,

$$x_{t,2,1} - y_{t,2,1} = \frac{\alpha_{t,2,1}}{2} + \frac{\alpha_{t,2,2}}{2} - \frac{\alpha_{t,2,1}}{2} + \frac{\alpha_{t,2,2}}{2} = \alpha_{t,2,2}$$
$$x_{t,2,2} - y_{t,2,2} = -\frac{\alpha_{t,2,1}}{2} - \frac{\alpha_{t,2,2}}{2} + \frac{\alpha_{t,2,1}}{2} - \frac{\alpha_{t,2,2}}{2} = -\alpha_{t,2,2}$$

Let us assume the induction hypothesis for all $i < \ell \leq d$.

From the definition of the AtomicSolver we know that,

$$\begin{bmatrix} x_{t,\ell,1} \\ x_{t,\ell,2} \\ \vdots \\ x_{t,\ell,2^{\ell-1}} \end{bmatrix} = \frac{1}{2} \cdot A_\ell \cdot \begin{bmatrix} \alpha_{t,\ell,1} \\ \alpha_{t,\ell,2} \\ \vdots \\ \alpha_{t,\ell,\ell} \end{bmatrix}, \text{ and } \begin{bmatrix} y_{t,\ell,1} \\ y_{t,\ell,2} \\ \vdots \\ y_{t,\ell,2^{\ell-1}} \end{bmatrix} = \frac{1}{2} \cdot B_\ell \cdot \begin{bmatrix} \alpha_{t,\ell,1} \\ \alpha_{t,\ell,2} \\ \vdots \\ \alpha_{t,\ell,\ell} \end{bmatrix}.$$

Therefore,

$$\begin{bmatrix} x_{t,\ell,1} - y_{t,\ell,1} \\ x_{t,\ell,2} - y_{t,\ell,2} \\ \vdots \\ x_{t,\ell,2^{\ell-1}} - y_{t,\ell,2^{\ell-1}} \end{bmatrix} = \frac{1}{2} \cdot (A_\ell - B_\ell) \cdot \begin{bmatrix} \alpha_{t,\ell,1} \\ \alpha_{t,\ell,2} \\ \vdots \\ \alpha_{t,\ell,\ell} \end{bmatrix}.$$

From the recursive definition of the matrices A_ℓ and B_ℓ, we get that $A_\ell - B_\ell = \begin{bmatrix} A_{\ell-1} - B_{\ell-1} & \mathbf{0}^{2^{\ell-2}} \\ B_{\ell-1} - A_{\ell-1} & \mathbf{0}^{2^{\ell-2}} \end{bmatrix}$. From the induction hypothesis, we get,

$$A_\ell - B_\ell = \begin{bmatrix} A_2 - B_2 & 0 & \cdots & 0 \\ B_2 - A_2 & 0 & \cdots & 0 \\ & & \vdots & \\ A_2 - B_2 & 0 & \cdots & 0 \\ B_2 - A_2 & 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 0 & \cdots & 0 \\ 0 & -2 & 0 & \cdots & 0 \\ & & & \vdots & \\ 0 & 2 & 0 & \cdots & 0 \\ 0 & -2 & 0 & \cdots & 0 \end{bmatrix}$$

and therefore for every $j \in [2^{\ell-1}]$,

$$|x_{t,\ell,j} - y_{t,\ell,j}| = \alpha_{t,\ell,2}.$$

■

We are now ready to prove 3.4.6.

Proof of Proposition 3.4.6

(a) Follows from the definition of $\alpha_{t,i,i}$ in Algorithm 2.

(b) $\alpha_{t,i,1} = 10^{f(t,i)} = 10^{(i-1)!\nu_t}$, where ν_t is the t^{th} prime greater than n^4. Since ν_t is increasing in t, and for a fixed t, $\alpha_{t,i,1}$ is increasing in i, $\max\limits_{t,i}\{\alpha_{t,i,1}\} = \alpha_{n,d,1}$ and $\min\limits_{t,i}\{\alpha_{t,i,1}\} = \alpha_{1,2,1}$. We had noted earlier that from Prime Number Theorem, the n^{th} prime greater than n^4 is at most n^5. Therefore,

$$10^{n^4} < 10^{\nu_1} = \alpha_{1,2,1} \leq \alpha_{t,i,1} \leq \alpha_{n,d,1} = 10^{(d-1)!\nu_n} < 10^{d!n^5}.$$

(c) From the definitions in Algorithm 2, for every $1 < r < i-1$, $\alpha_{t,i,r} = 10^{g(t,i,r)}$. Note that $\max\limits_{t,i,r}\{g(t,i,r)\} = g(n,d,d-1) \leq nd^2$ and therefore,

$$\alpha_{t,i,r} \leq \alpha_{n,d,d-1} = 10^{g(n,d,d-1)} \leq 10^{nd^2}.$$

(d) and (e) Fix an arbitrary $t \in [n]$. We prove by induction on $i \in \{2, \cdots, d\}$ that,

$$|\alpha_{t,i,i}| < 2 \text{ and } \sum_{r=2}^{i} |\alpha_{t,i,r}| \leq 10^{\nu - nd}.$$

For the base case, $i = 2$, $\alpha_{t,2,2} = \frac{b_t^2 - a_t^2}{2\alpha_{t,2,1}}$. Recall from Definition 3.13 that the variable part of a_t and b_t is the same. Therefore $|b_t - a_t| \leq 10^{m+\nu}$. From Fact 3.3.1, we know that $|a_t|$ and $|b_t|$ are at most $10^{m+\nu+n+1} = 10^M$, so we get,

$$|b_t^2 - a_t^2| = |(b_t - a_t)(b_t + a_t)| \leq 10^{m+\nu} \cdot 2\max\{a_t, b_t\} < 10^{m+\nu} \cdot 2 \cdot 10^M.$$

Since $m+\nu < M$, $|\alpha_{t,2,2}| < \frac{10^{m+\nu}\cdot 2\cdot 10^M}{2\cdot 10^{f(t,2)}} < 10^{2M-f(t,2)}$. By definitions in Algorithm 2, $f(t,2) = \nu_t$ and ν_t is a prime larger than n^4. Also, $M = O(n^3)$ and $f(t,2) > 2M$ therefore, it follows that, $|\alpha_{t,2,2}| < 1 < 10^{\nu-nd}$ and the claim holds for $i = 2$.

Let us assume the induction hypothesis for all $i < \ell \leq d$, and we now prove the claim for $i = \ell$. We need to show that

$$|\alpha_{t,\ell,\ell}| < 2 \text{ and } \sum_{r=2}^{\ell} |\alpha_{t,\ell,r}| \leq 10^{\nu-nd}$$

We first bound the magnitude of $\alpha_{t,\ell,\ell}$ for any $t \in [n]$. Recall from the definitions in Algorithm 2,

$$|\alpha_{t,\ell,\ell}| = \frac{|R_{t,\ell}|}{i! \cdot \prod\limits_{r\in[\ell-1]} \alpha_{t,\ell,r}}, \text{ where, } R_{t,\ell} = b_t^\ell - a_t^\ell + \sum_{u=2}^{\ell-1}\sum_{v=1}^{2^{u-1}} y_{t,u,v}^\ell - x_{t,u,v}^\ell.$$

We will bound each individual term in the definition of $\alpha_{t,\ell,\ell}$ separately.

The term $|b_t^\ell - a_t^\ell|$ in $R_{t,\ell}$ can be factorized as $|b_t^\ell - a_t^\ell| = |(b_t - a_t)(\sum_{k=0}^{\ell-1} b_t^k a_t^{\ell-1-k})|$. We had seen earlier that $|(b_t - a_t)| < 10^{m+\nu} < 10^M$ and from Fact 3.3.1, $\max\{a_t, b_t\} < 10^M$. Using these observations, we get

$$\begin{aligned} |b_t^\ell - a_t^\ell| &= |(b_t - a_t)(\sum_{k=0}^{\ell-1} b_t^k a_t^{\ell-1-k})| \\ &< 10^M \cdot \ell \cdot \max\{a_t^{\ell-1}, b_t^{\ell-1}\} \\ &\leq 10^M \cdot \ell \cdot 10^{M(\ell-1)} = \ell \cdot 10^{M\ell}. \end{aligned} \tag{3.19}$$

Using the definitions of $\alpha_{t,\ell,r}$, the denominator in the expression for $\alpha_{t,\ell,\ell}$ can be written as

$$\begin{aligned} \ell! \prod_{r=1}^{\ell-1} \alpha_{t,\ell,r} &= \ell! \cdot 10^{f(t,\ell) + \sum_{r=2}^{\ell-1} g(t,\ell,r)} \\ &\geq \ell! \cdot 10^{f(t,\ell)+g(t,\ell,2)} \end{aligned} \tag{3.20}$$

Now to bound the magnitude of $\left|\sum_{u=2}^{\ell-1}\sum_{v=1}^{2^{u-1}} y^{\ell}_{t,u,v} - x^{\ell}_{t,u,v}\right|$, we have

$$\begin{aligned}\left|\sum_{u=2}^{\ell-1}\sum_{v=1}^{2^{u-1}} y^{\ell}_{t,u,v} - x^{\ell}_{t,u,v}\right| &\leq \sum_{u=2}^{\ell-1}\sum_{v=1}^{2^{u-1}} |y^{\ell}_{t,u,v} - x^{\ell}_{t,u,v}| \\ &= \sum_{u=2}^{\ell-1}\sum_{v=1}^{2^{u-1}} |(y_{t,u,v} - x_{t,u,v})(\sum_{k=0}^{\ell-1} y^{k}_{t,u,v} x^{\ell-1-k}_{t,u,v})| \\ &\leq \sum_{u=2}^{\ell-1}\sum_{v=1}^{2^{u-1}} |(y_{t,u,v} - x_{t,u,v})| \cdot \ell \cdot \max\{|x_{t,u,v}|^{\ell-1}, |y_{t,u,v}|^{\ell-1}\}\end{aligned}$$

Using Claim 3.5.2, we know that for any $(t, u, v) \in [n] \times \{2, \cdots, \ell-1\} \times [2^{u-1}]$,

$$|x_{t,u,v} - y_{t,u,v}| = \alpha_{t,u,2} = 10^{g(t,u,2)}.$$

Also, from definition of the auxiliary variables in Algorithm 2, each $x_{t,u,v}$ and $y_{t,u,v}$ for any $t \in [n]$, is a $(\pm\frac{1}{2})$-linear combinations of $\{\alpha_{t,u,r} \mid r \in [u]\}$. Therefore,

$$\max\{|x_{t,u,v}|, |y_{t,u,v}|\} \leq \frac{1}{2}\sum_{r=1}^{u} |\alpha_{t,u,r}|.$$

Since $u < \ell$, using the induction hypothesis, we know that $\sum_{r=2}^{u} |\alpha_{t,u,r}| < 10^{\nu - nd}$. So, we get

$$\max\{|x_{t,u,v}|, |y_{t,u,v}|\} \leq |\alpha_{t,u,1}| + \sum_{r=2}^{u} |\alpha_{t,u,r}| < \frac{1}{2}(10^{f(t,u)} + 10^{\nu - nd}) < 10^{f(t,u)}.$$

From these observations, we get

$$\left|\sum_{u=2}^{\ell-1}\sum_{v=1}^{2^{u-1}} y^{\ell}_{t,u,v} - x^{\ell}_{t,u,v}\right| \leq \sum_{u=2}^{\ell-1}\sum_{v=1}^{2^{u-1}} 10^{g(t,u,2)} \cdot \ell \cdot (10^{f(t,u)})^{\ell-1}$$

Note that $\max_{u}\{g(t, u, 2)\} = g(t, \ell-1, 2)$ and for a fixed t, $f(t, i)$ is increasing in i, therefore, $f(t, u) \leq f(t, \ell-1)$ for all $u \leq \ell - 1$. Therefore,

$$\left|\sum_{u=2}^{\ell-1}\sum_{v=1}^{2^{u-1}} y^{\ell}_{t,u,v} - x^{\ell}_{t,u,v}\right| \leq \ell \cdot 2^{\ell} \cdot 10^{g(t,\ell-1,2)} \cdot 10^{(\ell-1)f(t,\ell-1)} \tag{3.21}$$

Combining Equations 3.19, 3.20, 3.21, we get an upper bound on the magnitude $\alpha_{t,\ell,\ell}$ as

$$\begin{aligned}
|\alpha_{t,\ell,\ell}| &= \frac{|R_{t,\ell}|}{\ell! \prod_{r=1}^{\ell-1} \alpha_{t,\ell,r}} \\
&\leq \frac{|b_t^\ell - a_t^\ell|}{\ell! \prod_{r=1}^{\ell-1} \alpha_{t,\ell,r}} + \frac{\left|\sum_{u=2}^{\ell-1} \sum_{v=1}^{2^{u-1}} y_{t,u,v}^\ell - x_{t,u,v}^\ell\right|}{\ell! \prod_{r=1}^{\ell-1} \alpha_{t,\ell,r}} \\
&\leq \frac{\ell \cdot 10^{M\ell}}{\ell! \cdot 10^{f(t,\ell)+g(t,\ell,2)}} + \frac{\ell \cdot 2^\ell \cdot 10^{g(t,\ell-1,2)+(\ell-1)f(t,\ell-1)}}{\ell! \cdot 10^{f(t,\ell)+g(t,\ell,2)}}
\end{aligned}$$

We now show that each individual term is at most 1, and therefore, $|\alpha_{t,\ell,\ell}| < 2$.

The first term can be simplified by plugging in the definition of $f(t,\ell)$ and using the fact that $g(t,\ell,r) > 2$.

$$\frac{\ell \cdot 10^{M\ell}}{\ell! \cdot 10^{f(t,\ell)+g(t,\ell,2)}} < \frac{1}{(\ell-1)!} \cdot 10^{M \cdot \ell - (\ell-1)! \nu_t - 2}$$

Since $\ell \cdot M < (\ell-1)! \nu_t$, it follows that

$$\frac{\ell \cdot 10^{M\ell}}{\ell! \cdot 10^{f(t,\ell)+g(t,\ell,2)}} < 1.$$

For the second term, note that $f(t,\ell) = (\ell-1)!\nu_t = (\ell-1) \cdot (\ell-2)!\nu_t = (\ell-1) f(t,\ell-1)$ and $g(t,\ell,2) - g(t,\ell-1,2) = 2\ell - 2 \geq 2$ for $\ell \geq 2$. Also, for $\ell \geq 2$, we have $\frac{2^\ell}{(\ell-1)!} \leq 4$. Therefore,

$$\begin{aligned}
\frac{\ell \cdot 2^\ell \cdot 10^{g(t,\ell-1,2)+(\ell-1)f(t,\ell-1)}}{\ell! \cdot 10^{f(t,\ell)+g(t,\ell,2)}} &= \frac{2^\ell}{(\ell-1)!} \cdot 10^{g(t,\ell-1,2)-g(t,\ell,2)} \cdot 10^{(\ell-1)f(t,\ell-1)-f(t,\ell)} \\
&\leq 4 \cdot 10^{-1} < 1
\end{aligned}$$

Now that we have established $|\alpha_{t,\ell,\ell}| < 2$, we show that $\sum_{r=2}^{\ell} |\alpha_{t,\ell,r}| < 10^{\nu - nd}$. We split this summation into two terms as

$$\sum_{r=2}^{\ell} |\alpha_{t,\ell,r}| = \sum_{r=2}^{\ell-1} |\alpha_{t,\ell,r}| + |\alpha_{t,\ell,\ell}|.$$

From the definition of $\alpha_{t,\ell,r}$ for $1 < r < \ell$, we have

$$\sum_{r=2}^{\ell-1} |\alpha_{t,\ell,r}| = \sum_{r=2}^{\ell-1} 10^{g(t,\ell,r)} < 10^{g(t,\ell,\ell-1)+1}.$$

Since $g(t,i,r)$ is increasing in t,i,r, $g(t,\ell,\ell-1)+1 \leq g(n,d,d-1)+1 = nd^2$. Recall that $\nu = n^2$, and therefore, for any $d = o(\sqrt{n})$, we have,

$$10^{g(t,\ell,\ell-1)+1} \leq 10^{nd^2} \leq 10^{\nu-nd-1}.$$

Therefore, it follows that

$$\sum_{r=2}^{\ell} |\alpha_{t,\ell,r}| \leq \sum_{r=2}^{\ell-1} |\alpha_{t,\ell,r}| + |\alpha_{t,\ell,\ell}| < 10^{\nu-nd-1} + 2 < 10^{\nu-nd}.$$

■

Proof of Proposition 3.4.7 From the definition of $\alpha_{t,i,r}$ in Algorithm 2, we know that each auxiliary variable is a $(\pm\frac{1}{2})$-linear combination of $\{\alpha_{t,i,r} \mid r \in [i]\}$. i.e

$$x_{t,i,j} = \sum_{r=1}^{i} u_r \alpha_{t,i,r} \quad \text{for some } u_r \in \{\pm\frac{1}{2}\}.$$

Therefore,

$$\frac{1}{2} \cdot |\alpha_{t,i,1}| - \frac{1}{2} \cdot \sum_{r=2}^{i} |\alpha_{t,i,r}| \leq \left| \sum_{r=1}^{i} u_r \alpha_{t,i,r} \right| \leq \frac{1}{2} \cdot |\alpha_{t,i,1}| + \frac{1}{2} \cdot \sum_{r=2}^{i} |\alpha_{t,i,r}|$$

Using Proposition 3.4.6 (e), we know that $\sum_{r=2}^{i} |\alpha_{t,i,r}| \leq 10^{\nu-nd}$ and from definitions, $\alpha_{t,i,1} = 10^{(i-1)!\nu_t}$. Therefore,

$$\frac{1}{2} \cdot (10^{(i-1)!\nu_t} - 10^{\nu-nd}) \leq |x_{t,i,j}| \leq \frac{1}{2} \cdot (10^{(i-1)!\nu_t} + 10^{\nu-nd})$$

■

Proof of Proposition 3.4.8 Let $t_1, t_2 \in [n], i_1, i_2 \in \{2, \cdots, d\}, j_1 \in [2^{i_1} - 1]$ and $j_2 \in [2^{i_2} - 1]$. If $(t_1, i_1, j_1) = (t_2, i_2, j_2)$, then from Claim 3.5.2, we know $|x_{t_1,i_1,j_1} - y_{t_1,i_1,j_1}| = \alpha_{t,i,2} \neq 0$ and it follows that $x_{t_1,i_1,j_1} \neq y_{t_1,i_1,j_1}$. Now we show that if

$(t_1, i_1, j_1) \neq (t_2, i_2, j_2)$, then $x_{t_1,i_1,j_1} \neq x_{t_2,i_2,j_2}$. The proof holds if either or both the $x_{t,i,j}$'s replaced with $y_{t,i,j}$. Let,

$$x_{t_1,i_1,j_1} = u_1 \cdot 10^{(i_1-1)!\nu_{t_1}} + \sum_{r=2}^{i_1} u_r \cdot \alpha_{t_1,i_1,r} \text{ for some } u_r \in \{\pm\frac{1}{2}\}$$

and,

$$x_{t_2,i_2,j_2} = v_1 \cdot 10^{(i_2-1)!\nu_{t_2}} + \sum_{r=2}^{i_2} v_r \cdot \alpha_{t_2,i_2,r} \text{ for some } v_r \in \{\pm\frac{1}{2}\}.$$

If $x_{t_1,i_1,j_1} = x_{t_2,i_2,j_2}$, then on reordering the terms we get,

$$\left|u_1 \cdot 10^{(i_1-1)!\nu_{t_1}} - v_1 \cdot 10^{(i_2-1)!\nu_{t_2}}\right| = \left|\sum_{r=2}^{i_2} v_r \cdot \alpha_{t_2,i_2,r} - \sum_{r=2}^{i_1} u_r \cdot \alpha_{t_1,i_1,r}\right|$$

Note that if $|u_1 \cdot 10^{(i_1-1)!\nu_{t_1}} - v_1 \cdot 10^{(i_2-1)!\nu_{t_2}}|$ is non-zero, then using the fact that ν_{t_1} and ν_{t_2} are prime integers larger than n^4 we have,

$$\left|u_1 \cdot 10^{(i_1-1)!\nu_{t_1}} - v_1 \cdot 10^{(i_2-1)!\nu_{t_2}}\right| \geq 10^{n^4}$$

But from Proposition 3.4.6 (e),

$$\left|\sum_{r=2}^{i_2} v_r \cdot \alpha_{t_2,i_2,r} - \sum_{r=2}^{i_1} u_r \cdot \alpha_{t_1,i_1,r}\right| \leq \frac{1}{2}\sum_{r=2}^{i_2}|\alpha_{t_2,i_2,r}| + \frac{1}{2}\sum_{r=2}^{i_1}|\alpha_{t_1,i_1,r}| \leq 10^{\nu-nd}$$

which is a contradiction. Therefore, $t_1 = t_2, i_1 = i_2$ and $u_1 = v_1$.

Let us assume $t_1 = t_2 = t, i_1 = i_2 = i$ and $j_1 > j_2$. If $x_{t,i,j_1} = x_{t,i,j_2}$, then,

$$\sum_{r=2}^{i}(v_r - u_r) \cdot \alpha_{t,i,r} = 0$$

We know that $(v_r - u_r) \in \{0, \pm 1\}$, so there exists a $\{0, \pm 1\}$- linear combination of $\alpha_{t,i,r}$ equal to 0. If $u_r = v_r$ for every $r \in \{2, \cdots, i\}$, then $j_1 = j_2$ since each auxiliary variable is a distinct linear combination of the $\alpha_{t,i,r}$'s. So, there exists at least one $r \in \{2, \cdots, i\}$ such that $u_r \neq v_r$. Let r^* be the largest such r. We know that

$$0 = \left|\sum_{r=2}^{i}(v_r - u_r) \cdot \alpha_{t,i,r}\right| \geq \left||\alpha_{t,i,r^*}| - |\sum_{r=2}^{r^*-1}(v_r - u_r) \cdot \alpha_{t,i,r}|\right|$$

But each $\alpha_{t,i,r} = 10^{g(t,i,r)}$ for $r \in \{2, \cdots, i-1\}$ is a distinct power of 10 and $|\alpha_{t,i,i}| < 2$. So, $|\alpha_{t,i,r^*}| - \left|\sum_{r=2}^{r^*-1}(v_r - u_r) \cdot \alpha_{t,i,r}\right| \neq 0$, which is a contradiction. Therefore, $j_1 = j_2$.

■

3.6 Existence of (Inhomogeneous) PTE Solutions over $\mathbb{F}_p^\ell$

Recall that a solution to a PTE system of size s and degree d satisfies

$$\begin{aligned} x_1 + x_2 + \cdots + x_s &= y_1 + y_2 + \cdots + y_s \\ x_1^2 + x_2^2 + \cdots + x_s^2 &= y_1^2 + y_2^2 + \cdots + y_s^2 \\ &\cdots \\ x_1^d + x_2^d + \cdots + x_s^d &= y_1^d + y_2^d + \cdots + y_s^d. \end{aligned}$$

We will show that such a system always has a solution over a field $\mathbb{F} = \mathbb{F}_{p^\ell}$, for $d < |\mathbb{F}|^{1/2-\delta}$, for $\delta > 0$. In fact, the proof will also hold for inhomogeneous PTE systems such as (†).

Theorem 3.6.1 *Let $\mathbb{F}$ be a finite field, and let $r_1, r_2, \ldots, r_d \in \mathbb{F}$. Let d be a positive integer such that $d \leq |\mathbb{F}|^{1/2-\delta}$. Then, there exists a solution in $\mathbb{F}$ to the system $\sum_{i=1}^s x_i^j - \sum_{i=1}^s y_i^j = r_j$, for $j \in [d]$, with $s = 3d/\delta$.*

Moreover, if $|\mathbb{F}|$ is a sufficiently large function of δ, then we can ensure that the x_i's and the y_i's are all distinct.

Let G be a group. An additive character of G is a a function $\chi : G \to \mathbb{C}$ such that $\chi(x + y) = \chi(x)\chi(y)$ for all $x, y \in G$. We will now define characters over groups of the form $\mathbb{F}^n$, where $\mathbb{F} = \mathbb{F}_{p^\ell}$ and p is a prime.

Let $\omega = e^{2\pi i/p}$ be a primitive pth root of unity, and let $Tr : \mathbb{F}_{p^\ell} \to \mathbb{F}_p$ be the Trace operator $Tr(x) = \sum_{i=0}^{\ell-1} x^{p^i}$. Then, an additive character of $\mathbb{F}^n = (\mathbb{F}_{p^\ell})^n$ is $\chi_a(x) = \omega^{Tr(a \cdot x)}$, where $a, x \in \mathbb{F}^n$, and $a \cdot x$ denotes the inner product over $\mathbb{F}^n$.

We will use of some results of [75]. Let μ be a distribution over vectors in $\mathbb{F}^n$, and denote by $\mu^{(s)}$ the distribution of $x_1 + x_2 + \ldots + x_s$, where the x_i's are picked independently from μ.

Theorem 3.6.2 *([75], Appendix B) Suppose that for some β, any non-trivial character χ of $\mathbb{F}^s$ satisfies*

$$|\mathbf{E}_{x\sim\mu}\chi(x)| \leq \beta.$$

Then

$$\sum_{x\in\mathbb{F}^n}\left|\mu^{(s)}(x) - \frac{1}{|\mathbb{F}|^n}\right| \leq \beta^s|\mathbb{F}|^n,$$

and so $\mu^{(s)}$ is $\beta^s|\mathbb{F}|^n$- close to the uniform distribution over $\mathbb{F}^n$ in statistical distance.

Recall the Weil/Deligne bound.

Theorem 3.6.3 *(Weil [76], Deligne [77]) Let $f(x_1, x_2, \ldots, x_t)$ be a t-variate polynomial over $\mathbb{F}$ of degree at most $|\mathbb{F}|^{1/2-\delta}$, for some $\delta > 0$. Then, either $\chi(f(x))$ is constant for all $x \in \mathbb{F}$, or χ satisfies $|\mathbf{E}_{x\in\mathbb{F}}\ \chi(f(x))| \leq |\mathbb{F}|^{-\delta}$.*

Proof of Theorem 3.6.1 For $x, y \in \mathbb{F}$, let $v_{x,y} = (x-y, x^2-y^2, \ldots, x^d-y^d) \in \mathbb{F}^d$. Let μ be the distribution of $v_{x,y}$ when x, y are distributed independently and uniformly in $\mathbb{F}$. Note that for a nontrivial character χ_a with $a \in (\mathbb{F}^*)^d$, we have

$$\mathbf{E}_{v_{x,y}\sim\mu}\ [\chi_a(v_{x,y})] = \mathbf{E}[\omega^{a\cdot v_{x,y}}] = \mathbf{E}_{x,y}[\omega^{g(x,y)}] = \mathbf{E}_{x,y}[\chi_a(g(x,y))]$$

for the polynomial $g(x,y) = \sum_{i=1}^{d}\ a_i\ (x^i - y^i)$ of degree $d \leq |\mathbb{F}|^{1/2-\delta}$.

By Deligne's Theorem 3.6.3, we have that

$$|\mathbf{E}_{v_{x,y}\sim\mu}\ [\chi_a(v_{x,y})]| = |\mathbf{E}[\chi_a(g(x,y))]| \leq |\mathbb{F}|^{-\delta}. \tag{3.22}$$

Pick s vectors $v_{x_1,y_1}, v_{x_2,y_2}, \ldots, v_{x_s,y_s} \in \mathbb{F}^d$ independently, according to μ and let $\mu^{(s)}$ denote the distribution of $S = \sum_{i=1}^{s} v_{x_i,y_i}$. Note that $\mu^{(s)}$ is precisely the distribution of $(\sum x_i - \sum y_i, \sum x_i^2 - \sum y_i^2, \ldots, \sum x_i^d - \sum y_i^d)$, when we pick the x_i's and y_i's independently and uniformly in $\mathbb{F}$.

By Theorem 3.6.2 and Equation (3.22), it follows that

$$\sum_{v\in\mathbb{F}^d}\left|\mu^{(s)}(v) - \frac{1}{|\mathbb{F}|^d}\right| \leq (|\mathbb{F}|^{-\delta})^s|\mathbb{F}|^d = |\mathbb{F}|^{-\delta s+d}.$$

Picking $s = 3d/\delta$, we get that $\mu^{(s)}((r_1, r_2, \ldots, r_d)) \geq |\mathbb{F}|^{-d} - |\mathbb{F}|^{-2d} > 0$.

We can also ensure that all x_i's and y_i's are distinct, by noticing that the

$$\begin{aligned}\Pr[|\{x_1, x_2, \ldots, x_s, y_1, \ldots, y_s\}| = 2s] &= \prod_{i=0}^{2s-1} \frac{1}{|\mathbb{F}| - i} \\ &< (|\mathbb{F}| - 2s)^{-2s} < |\mathbb{F}|^{-2d} \\ &< |\mathbb{F}|^{-d} - |\mathbb{F}|^{-2d},\end{aligned}$$

for $|\mathbb{F}|$ being sufficiently large as a function of δ. ■

3.7 Hardness of MSS(d) over $\mathbb{F}_{p^\ell}$

We will choose prime $p = O(d!)$ and $\ell = \mathsf{poly}(n)$ for this reduction. To generate the field $\mathbb{F}_q = \mathbb{F}_{p^\ell}$, we consider an irreducible polynomial over $\mathbb{F}_p$ of degree ℓ. Let γ be a root of this polynomial in the algebraic closure of $\mathbb{F}_p$. Every element of $\mathbb{F}_q$ can then be generated as a linear combination of $1, \gamma, \cdots, \gamma^{\ell-2}, \gamma^{\ell-1}$ over $\mathbb{F}_p$ (We refer to [78] for a general treatment of finite fields.). Then, for $v = \sum v_i \gamma^i \in \mathbb{F}_q$, we will abuse notation and view v as the vector $(v_1, v_2, \ldots, v_{\ell-1})$. We define an analogue of the notion of "magnitude" used in the previous sections. For $v \in \mathbb{F}_q$, define $|v|$ to be the largest non-zero index $i \in [\ell]$ in the vector representation of v. Note that this definition of magnitude satisfies the property that $|u + v| \leq \max(|u|, |v|)$ for every $u, v \in \mathbb{F}_q$, and thus also satisfies that the triangle inequality.

We now sketch a proof of the reduction, which follows analogously to the proof over the rational field, with some small modifications, as described next.

An instance of MSS(d) consists of a tuple $\langle A, k, B_1, \ldots, B_d \rangle$. Similar to the rational field reduction, each variable $(z_t, \overline{z_t})$ is mapped to $2^{d+1} - 2$ distinct elements $\{a_t\} \cup \{x_{t,i} \mid i \in [2^d - 2]\}$ (corresponding to z_t) and $\{b_t\} \cup \{y_{t,i} \mid i \in [2^d - 2]\}$ (corresponding to $\overline{z}_t$). Let $\{a'_t, b'_t : t \in [n]\}$ be the elements of $\mathbb{F}_q$ produced by the reduction of 1-in-3 SAT to Subset-Sum defined as follows:

- The vector representations of a'_t and b'_t consist of two parts: a clause region consisting of the leftmost m coordinates and a variable region consisting of the next n indices.
- In the variable region, a'_t and b'_t have a 1 at the t-th index and 0's at the other indices. Denote that by $(a_t)'^v$.
- In the clause region, for every $j \in [m]$, a'_t (resp. b'_t) has a 1 at the jth location if z_t (resp. $\overline{z_t}$) appears in clause j, and a 0 otherwise. We denote the clause part of a'_t by $(a_t)'^c$.
- $a'_t = (a_t'^c, a_t'^v, 0^{\ell-m-n})$. Similarly for b'_t.
- The target B is set to the element whose field representation is the vector which takes 1's in the first $m+n$ indices and 0 everywhere else. i.e. $B = (1^m, 1^n, 0^{\ell-m-n})$.

Define,

$$a_t = (0^\nu, a_t'^c, a_t'^v, 0^{\ell-\nu-m-n}) \quad \text{and,} \quad b_t = (0^\nu, b_t'^c, b_t'^v, 0^{\ell-\nu-m-n}).$$

For each $t \in [n]$, we will explicitly construct two sets of $2^d - 2$ *auxiliary variables*, $X_t = \{x_{t,i} \mid i \in [2^d - 2]\}$ and $Y_t = \{y_{t,i} \mid i \in [2^d - 2]\}$ which satisfy the following properties:

(1) $\sum_{x \in X_t} x = \sum_{y \in Y_t} y = 0$

(2) $\sum_{x \in X_t} x^k - \sum_{y \in Y_t} y^k = b_t^k - a_t^k$ for every $k \in \{2, \ldots, d\}$.

(3) Additionally, an appropriately scaled set of auxiliary variables, can be shown to satisfy the bimodal property. Namely, for any subset $S \subseteq \bigcup_{t \in [n]} X_t \cup Y_t$, and a scaling factor $K = \gamma^h$, where $h = \mathsf{poly}(n, d!)$, either

$$\left| \sum_{s \in S} \gamma^h s \right| > h + n^4 \quad \text{or} \quad \left| \sum_{s \in S} \gamma^h s \right| < h + \nu.$$

Define the set $A = \bigcup_{t\in[n]} \{a_t\} \cup \{b_t\} \cup X_t \cup Y_t$. The targets $B_1, \ldots, B_d$ are defined as follows:

$$B_1 = (0^\nu, 1^m, 1^n, 0^{\ell-\nu-m-n}),$$
$$B_k = \sum_{t=1}^{n} a_t^k + \sum_{t=1}^{n} \sum_{x\in X_t} x^k \text{ for every } k \in \{2, \ldots, d\}$$

Note that $a_t = \gamma^\nu \cdot a'_t$. Similarly, $b_t = \gamma^\nu \cdot b'_t$ and $B_1 = \gamma^\nu \cdot B$.

We now define a scaled version of the subset sum instance over the finite fields. Let $h = \mathsf{poly}(n, d!)$ and let $K = \gamma^h$, be the scaling factor . Scaling of all the elements of the instance of MSS(d) is roughly equivalent to scaling the rational solutions by a large power of 10. The scaling of all the variables maintains Properties (1), and (2) of the auxiliary variables and also satisfies the solution to achieve Property (3).

Let $A_h = \{\gamma^h a \mid a \in A\}$. $B_{k,h} = \gamma^{kh} B_k$ for every $k \in \{2, \ldots, d\}$. The following lemma shows that the MSS(d) instance and its scaled version as defined above are equivalent.

Lemma 3.7.1 *Let $h = \mathsf{poly}(n, d!)$ and $K = \gamma^h$ be the scaling factor. There exists a subset $S \subseteq A$ such that for every $k \in [d]$*

$$\sum_{s\in S} s^k = B_k.$$

if and only if there exists a subset $S_h \subseteq A_h$ such that for every $k \in [d]$

$$\sum_{s\in S_h} s^k = B_{k,h}.$$

The proof of Lemmas 3.7.1 is straightforward and follows from the fact that the moment equations in MSS(d) are homogeneous and therefore scaling all the variables and the targets does not change the problem.

We can then state the analogous statement of Lemma 3.3.3, which implies the NP-hardness of MSS(d) over $\mathbb{F}_q$.

Lemma 3.7.2 *There exists a satisfying assignment to a 3-SAT instance $\phi(z_1, \ldots, z_n)$ if and only if there exists a subset $S \subseteq A_h$ such that for every $k \in [d]$,*

$$\sum_{s\in S} s^k = B_{k,h}.$$

The proof of Lemma 3.7.2 follows from the properties of the auxiliary variables stated above and all the steps of the proof over the rationals can be carried over here, because we chose to scale the instance by a large enough power of γ, and we chose p and ℓ large enough, in order to ensure that there is no wrapping around when we add terms with large magnitudes.

3.8 Discussion and Open Questions

The main open question that comes up from this work is to explicitly and efficiently construct degree-d PTE solutions of size subexponential in d (see Problem 3.1.3).

It would also be very interesting to prove analogous NP-hardness results for Bounded Distance Decoding of Reed-Solomon codes in the case where the evaluation set of the code is fixed. This is the case of BDD with preprocessing where the code is known beforehand and the input instance only consists of the target vector.

Finally, our NP-hardness results for Reed-Solomon codes apply to the case where the field size is exponential in the block length N; it would be very interesting to prove analogous NP-hardness results for smaller fields. In particular, for primitive Reed-Solomon codes, where the evaluation set is the entire field, the complexity of this problem remains an intriguing open question.

4 DECIDING ORTHOGONALITY IN CONSTRUCTION-A LATTICES

In this chapter, we study the orthogonality decision problem for the family of Construction -A lattices obtained from binary and ternary linear codes. A Construction-A lattices L is obtained from a linear error-correcting code C over a finite field of q elements (denoted $\mathbb{F}_q$) as $L = C + q\mathbb{Z}^n$. Our main technical contribution in this chapter, is a decomposition theorem for Construction-A lattices that admit an orthogonal basis. A natural way to obtain orthogonal lattices through Construction-A is by taking direct products of lower dimensional orthogonal lattices. We show that this is the only possible way.

4.1 Results

As mentioned above, we start by showing a structural decomposition of orthogonal lattices of the form $C + 2\mathbb{Z}^n$ and $C + 3\mathbb{Z}^n$ into constant-size orthogonal lattices. We remark that the decomposition holds up to permutations of the coordinates, and we use the notation $C_1 \cong C_2$ and $L_1 \cong L_2$ to denote the equivalence of codes and lattices under permutation of coordinates. We use the notation $L_1 \oplus L_2$ to denote the direct sum of two lattices.

Theorem 4.1.1 *Let $L_C = C + 2\mathbb{Z}^n$ be a lattice obtained from a binary linear code $C \subseteq \mathbb{F}_2^n$. Then the following statements are equivalent:*

1. *L_C is orthogonal.*
2. *$L_C \cong \oplus_i L_i$, where each L_i is either $\mathbb{Z}$, or $2\mathbb{Z}$, or the 2-dimensional lattice generated by the rows of the matrix $\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.*

3. *$C \cong \oplus_i C_i$, where each C_i is either a length-1 binary linear code $\subseteq \{0,1\}$, or the length-2 binary linear code $\{00, 11\}$.*

The decomposition characterization leads to an efficient algorithm to verify if a given lattice obtained from a binary linear code using Construction-A is orthogonal. For the purposes of this algorithmic problem, the input consists of a basis to the lattice. The algorithm finds the component codes given by the characterization thereby computing the orthogonal basis for such a lattice.

Theorem 4.1.2 *Given a basis for a lattice L obtained from a binary linear code using Construction-A, there exists an algorithm running in time $O(n^6)$ that verifies if L is orthogonal, and if so, outputs an orthogonal basis.*

We obtain a similar decomposition and algorithm for lattices obtained from ternary codes. For succinctness of presentation we define the following integer matrix:

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

Theorem 4.1.3 *Let $L_C = C + 3\mathbb{Z}^n$ be a lattice obtained from a ternary linear code $C \subseteq \mathbb{F}_3^n$. Then the following statements are equivalent:*

1. *L_C is orthogonal.*

2. *$L_C \cong \oplus_i L_i$, where each L_i is either $\mathbb{Z}$, or $3\mathbb{Z}$, or the 4-dimensional lattice generated by the rows of a matrix $\mathcal{T}(M)$ obtained from M by negating some subset of columns.*

3. *$C \cong \oplus_i C_i$, where each C_i is either a linear length-1 ternary code, or the linear length-4 ternary code generated by the rows of $(\mathcal{T}(M) \bmod 3) \in \mathbb{F}_3^{4\times 4}$, where $\mathcal{T}(M)$ is obtained from M by negating some subset of its columns.*

Theorem 4.1.4 *Given a basis for a lattice L obtained from a ternary linear code using Construction-A, there exists an algorithm running in time $O(n^8)$ that verifies if L is orthogonal, and if so, outputs an orthogonal basis.*

Theorems 4.1.1 and 4.1.2 are proved in Section 4.3. Theorems 4.1.3 and 4.1.4 are proved in Section 4.4.

4.2 Preliminaries

A *weighing matrix* of order n and weight k is a $n \times n$ matrix with entries in $\{0, 1, -1\}$ such that each row and column has exactly k non-zero entries and the row vectors are orthogonal to each other. By definition, a weighing matrix W satisfies $WW^T = kI_n$. For matrices $A \in \mathbb{R}^{n_1 \times n_1}$ and $B \in \mathbb{R}^{n_2 \times n_2}$, we denote the $(n_1 + n_2) \times (n_1 + n_2)$-dimensional block-diagonal matrix obtained using blocks A and B by $A \oplus B$. We will use the following characterization of weighing matrices of weights 2 and 3. For completeness we present a proof of Theorems 4.2.1 and 4.2.3 here.

Theorem 4.2.1 *[79] A matrix W is a weighing matrix of order n and weight 2 if and only if W can be obtained from*

$$\oplus_{i=1}^{n/2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

by negating some rows and columns and by interchanging some rows and columns.

Proof Let $W(n, 2)$ denote a weighing matrix of order n and weight 2. We prove this theorem by induction on the order n of $W(n, 2)$.

For $n = 2$, the matrix $\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ is the only possible 2×2 orthogonal matrix up to permutations of columns with entries in $\{1, -1\}$. Therefore, $W(2, 2) \cong \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.

Let us assume the induction hypothesis about all weighing matrices of order at most $n - 2$ and weight 2.

Let $W \in \{0,1,-1\}^{n\times n}$ be an orthogonal matrix such that each row of W has exactly two non-zero entires. Since we are characterizing W up to permutations of rows and columns, and negations of rows and columns, we can assume without loss of generality that the first row of W is,

$$w_1 = (1,1,0,\ldots,0).$$

Since W is orthogonal, the non-zero entries of every other row, w_i has even intersection with the non-zero entries of w_1, i.e.

$$|\text{Support}(w_i) \cap \text{Support}(w_1)| \in \{0,2\}.$$

Let us consider the case when $|\text{Support}(w_i) \cap \text{Support}(w_1)| = 0$ for all $i \in [n] \setminus \{1\}$. Note that the w_i's are mutually orthogonal and supported on $n-2$ coordinates. This would imply that W has at most $n-1$ rows in total which contradicts the fact that W is a $n \times n$ matrix. Therefore, there exists at least one row, say w_2 such that $|\text{Support}(w_2) \cap \text{Support}(w_1)| = 2$. Since w_2 is orthogonal to w_1 and it has exactly two non-zero entries, it is of the form

$$w_2 = (1,-1,0,\ldots,0).$$

We note that there cannot exist any other row w_3 of W, such that $|\text{Support}(w_3) \cap \text{Support}(w_1)| = 2$ since it is not possible for such a vector to be orthogonal to both w_1 and w_2. Therefore, for every other row $w_i, i \in \{3,\cdots,n\}$, we have $|\text{Support}(w_i) \cap \text{Support}(w_1)| = 0$. The weighing matrix is therefore of the form:

$$W \cong \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \oplus W'$$

where W' is a weighing matrices of order at $n-2$ and weight 2. The proof follows from the induction hypothesis. ■

The following lemma is needed for the proof of Theorem 4.2.3

Lemma 4.2.2 *A weighing matrix of order* 4 *and weight* 3 *is equivalent to* M *up to permutations of rows and columns, and negations of rows and columns.*

Proof Let W be a weighing matrix of order 4 and weight 3. Since each vector has weight at exactly 3, we can assume without loss of generality, $w_1 = (1, 1, 1, 0)$. All rows are mutually orthogonal, therefore, $|\text{Support}(w_i) \cap \text{Support}(w_j)| \in \{0, 2\}$ and $|\text{Support}(w_1) \cap \text{Support}(w_i)| \neq 0$ for all i.

Let us consider another row w_2 such that $|\text{Support}(w_1) \cap \text{Support}(w_2)| = 2$. So, $w_2 = (1, -1, 0, 1)$ up to permutations of coordinates. For any other row w_3, if $|\text{Support}(w_1) \cap \text{Support}(w_2) \cap \text{Support}(w_3)| = 2$ then the orthogonality condition with either w_1 or w_2 is violated. Therefore, w_3 is of the form $w_3 = (1, 0, -1, -1)$. This forces $w_4 = (0, 1, -1, 1)$ and, hence $W \equiv M$. ■

Theorem 4.2.3 *[79] A matrix W is a weighing matrix of order n and weight 3 if and only if W can be obtained from $\oplus_{i=1}^{n/4} M$ by negating some rows and columns and by interchanging some rows and columns.*

Proof Let $W(n, 3)$ denote a weighing matrix of order n and weight 3. We prove this theorem by induction on the order n of $W(n, 3)$.

For $n = 4$, from Lemma 4.2.2 we have $W(4, 3) \cong M$. Let us assume the induction hypothesis about all weighing matrices of order at most $n - 4$ and weight 3.

Let $W \in \{0, 1, -1\}^{n \times n}$ be an orthogonal matrix such that each row of W has exactly three non-zero entires. Since we are characterizing W up to permutations of rows and columns, and negations of rows and columns, we can assume without loss of generality that the first row of W is

$$w_1 = (1, 1, 1, 0, \ldots, 0).$$

Since W is orthogonal, the non-zero entries of every other row, w_i has even intersection with the non-zero entries of w_1, i.e.

$$|\text{Support}(w_i) \cap \text{Support}(w_1)| \in \{0, 2\} \quad \text{for all } i \in \{2, \ldots, n\}.$$

Let us consider the case when $|\text{Support}(w_i) \cap \text{Support}(w_1)| = 0$ for all $i \in [n] \setminus \{1\}$. Note that the w_i's are mutually orthogonal and supported on $n - 3$ coordinates. This

would imply that W has at most $n-2$ rows in total which contradicts the fact that W is a $n \times n$ matrix. Therefore, there exists at least two rows, say w_2, w_3 such that $|\text{Support}(w_1) \cap \text{Support}(w_2)| = 2$ and $|\text{Support}(w_1) \cap \text{Support}(w_3)| = 2$. Since these three vectors are mutually orthogonal and $\text{Support}(w_1) = 2$, it follows that $|\text{Support}(w_2) \cap \text{Support}(w_3)| > 0$. Without loss of generality, these three vectors are of the following form:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & -1 & 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & -1 & -1 & 0 & \cdots & 0 \end{bmatrix}$$

We observe that if $|\text{Support}(w_i) \cap \text{Support}(w_2)| = 0$ for all $i \in [n] \setminus \{1, 3\}$, then the number of vectors in W is at most $n-1$. Therefore, there exists at least one other row w_4 such that $|\text{Support}(w_4) \cap \text{Support}(w_2)| = 2$. Since w_4 is orthogonal to all w_1, w_2 and w_3, the unique candidate for w_4 is of the form $(0, 1, -1, 1, 0, \ldots, 0)$. We note that if there exists another row, w_5 such that $|\text{Support}(w_5) \cap \text{Support}(w_j)| > 0$ for any $j \in [4]$, then it cannot be orthogonal to all four vectors w_1, w_2, w_3 and w_4. So, $|\text{Support}(w_i) \cap \text{Support}(w_j)| = 0$ for any $j \in [4]$ and $i \geq 5$.

Therefore, $W(n, 3) \cong M \oplus W'$, where W' is a weighing matrix of order $n-4$ and weight 3. It then follows from the induction hypothesis that

$$W(n, 3) \cong \oplus_i M.$$

■

4.3 Orthogonal Lattices from Binary Codes

In this section we focus on Construction-A lattices obtained from binary linear codes. In Section 4.3.1, we show that any orthogonal lattice obtained from a binary linear code by Construction-A is equivalent to a product lattice whose components are one-dimensional or two-dimensional lattices. In Section 4.3.2, we show that given a lattice obtained from a binary linear code by Construction-A, there exists an efficient algorithm to verify if the lattice is orthogonal.

4.3.1 Decomposition Characterization

We prove Theorem 4.1.1 in this subsection.

Proof of Theorem 4.1.1 We show that (1) $\equiv$ (2) and (2) $\equiv$ (3) to complete the equivalence of the three statements.
(1) $\equiv$ (2): We show that $L_C = C + 2\mathbb{Z}^n$ is orthogonal if and only if it decomposes into a direct sum of lower dimensional orthogonal lattices, $L_C \cong \oplus_i L_i$.

If $L_C \cong \oplus_i L_i$ such that each L_i is orthogonal, then L_C is also orthogonal. This is because L_C would have a block diagonal orthogonal basis where each block is in itself orthogonal or a 1×1 matrix.

We prove the other direction of the equivalence by induction on the dimension, n, of the lattice L_C. For the base case consider $n = 1$. Since L is integral and is of the form $C + 2\mathbb{Z}$ for some binary linear code C, it follows that L has to be either $\mathbb{Z}$ or $2\mathbb{Z}$.

Let us assume the induction hypothesis for all $n-1$ or lower dimensional orthogonal lattices obtained from binary linear codes using Construction-A.

Let L_C be an n-dimensional orthogonal lattice and B be an orthogonal basis of L_C with the rows being basis vectors. Since L_C is an integral lattice, B has only integral entries. The next two claims summarize certain properties of the entries of the basis matrix B.

Claim 4.3.1 *For every row b of B and for every $j \in [n]$, we have that $2|b_j| \in \{0, \|b\|^2, 2\|b\|^2\}$.*

Proof Since B is an orthogonal basis, $BB^T = D$, where D is the diagonal matrix with entries $\|b^{(i)}\|^2$, where $b^{(i)}$ denotes the i^{th} basis vector.

$$D = \begin{bmatrix} \|b^{(1)}\|^2 & 0 & 0 & \cdots & 0 \\ 0 & \|b^{(2)}\|^2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & \|b^{(n)}\|^2 \end{bmatrix}$$

We know that $2\mathbb{Z}^n \subseteq L_C$ so, $2e_j \in L_C$ for every $j \in [n]$. Therefore, there is an integral matrix $X \in \mathbb{Z}^{n\times n}$ such that $XB = 2I_n$, i.e. $2B^{-1} \in \mathbb{Z}^{n\times n}$. Since B is an orthogonal basis,

$$B^{-1} = B^T D^{-1} \in \frac{1}{2}\mathbb{Z}^{n\times n}.$$

Each column of $B^T D^{-1}$ is given by $b/\|b\|^2$, where b is a basis vector. Therefore, for any $j \in [n]$, we have

$$2b_j \equiv 0 \bmod \|b\|^2 \text{ for all } j \in [n], \text{ and rows } b \text{ of } B.$$

Since b_j is integral and $|b_j| \leq \|b\|^2$ for every $j \in [n]$, it follows that $2|b_j| \in \{0, \|b\|^2, 2\|b\|^2\}$. ■

Claim 4.3.2 *Let b be a row of B.*

(1) If there exists $j \in [n]$ such that $2|b_j| = 2\|b\|^2$, then $b_j = \pm 1$ and $b_{j'} = 0$ for every $j' \in [n] \setminus \{j\}$.

(2) If there exists $j \in [n]$ such that $2|b_j| = \|b\|^2$ and $b_j = \pm 2$, then $b_{j'} = 0$ for every $j' \in [n] \setminus \{j\}$.

(3) If there exists $j \in [n]$ such that $2|b_j| = \|b\|^2$ and $b_j = \pm 1$, then there exist $j_1 \in [n] \setminus \{j\}$, such that $|b_{j_1}| = 1$ and $b_{j'} = 0$ for every $j' \in [n] \setminus \{j, j_1\}$.

Proof (1) Since, $\|b\|^2 = \sum_{i=1}^n b_i^2$, and each $b_i \in \mathbb{Z}$, we conclude that $|b_j| = 1$ and the remaining coordinates in b have to be 0, i.e $b_{j'} = 0$ for all $j' \in [n] \setminus \{j\}$.

(2) Follows from $2|b_j| = \|b\|^2$ and b being integral.

(3) We can re-write the condition $2|b_j| = \|b\|^2$ as $2|b_j| = \sum_{i=1}^n b_i^2$. Rearranging the terms, we have

$$|b_j|\ (2 - |b_j|) = \sum_{i\neq j} b_i^2.$$

If $b_j = \pm 1$, then $\sum_{i\neq j} b_i^2 = 1$. Further, b is integral. Hence, b has exactly 1 other non-zero coordinates b_{j_1}, $j \neq j_1$, such that $|b_{j_1}| = 1$.

■

Using the properties of the orthogonal basis B of L_C given in Claims 4.3.1 and 4.3.2, we show that B is equivalent (up to permutations of its columns) to a block diagonal matrix, i.e

$$B \cong \begin{bmatrix} B_1 & 0 & \cdots & 0 \\ 0 & B_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & B_k \end{bmatrix}$$

where each B_i is either the 1×1 matrix $\begin{bmatrix} 1 \end{bmatrix}$ or the 1×1 matrix $\begin{bmatrix} 2 \end{bmatrix}$ or the 2×2 matrix $\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. It follows that $L_C \cong \oplus_i L_i$ such that B_i is the basis for the lower dimensional lattice L_i.

Let us pick a row b of B with the smallest support. Fix an index $j \in [n]$ to be the index of a non-zero entry with minimum absolute value in b, i.e. $j := \arg\min_k\{|b_k|\}$. Since b is a row of a basis matrix, b cannot be the all-zeroes vector and therefore there exists a $j \in [n]$ such that $|b_j| > 0$. Since we are only interested in equivalence (that allows for permutation of coordinates), we may assume without loss of generality that $j = 1$ by permuting the coordinates. By Claim 4.3.1, we have that $2|b_1| \in \{\|b\|^2, 2\|b\|^2\}$. We consider each of these cases separately.

1. Suppose $2|b_1| = 2\|b\|^2$. By Claim 4.3.2(1), $b = (\pm 1, 0, \ldots, 0)$. Since B is an orthogonal basis, $\langle b, b' \rangle = 0 \Rightarrow b'_1 = 0$ for all $b' \neq b \in B$. The orthogonality of B therefore forces all other basis vectors to take a value of 0 in the 1'st coordinate. Thus B is of the form

$$B = \left(\begin{array}{c|c} \pm 1 & 0 \cdots 0 \\ \hline 0 & \\ \vdots & B' \\ 0 & \end{array}\right).$$

   Therefore, we obtain $L_C \cong \mathbb{Z} \oplus L'$, where L' is an orthogonal $(n-1)$-dimensional lattice generated by the basis matrix restricted to the coordinates other than 1,

say, B'. From Claim 2.2.3, it follows that $L' = C' + 2\mathbb{Z}^{n-1}$ for some binary linear code $C' \subseteq \mathbb{F}_2^{n-1}$. Thus L' satisfies the induction hypothesis and we have the desired decomposition.

2. Suppose $2|b_1| = \|b\|^2$. We can re-write this condition as $2|b_1| = \sum_{i=1}^{n} b_i^2$. Rearranging the terms, we have

$$|b_1|\ (2 - |b_1|) = \sum_{i \neq 1} b_i^2.$$

Since the RHS is a sum of squares, it should be non-negative.

(i) If RHS is 0, then $b_1 = \pm 2$ and therefore, it follows from Claim 4.3.2(2) that $b = (\pm 2, 0, \ldots, 0)$. The orthogonality of B forces all other basis vectors to take a value of 0 at the 1'st coordinate.

$$B = \left(\begin{array}{c|c} \pm 2 & 0 \cdots 0 \\ \hline \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} & B' \end{array}\right)$$

Therefore, we obtain $L_C \cong 2\mathbb{Z} \oplus L'$, where L' is an orthogonal $(n-1)$-dimensional lattice generated by the basis matrix restricted to the coordinates other than 1, say B'. From Claim 2.2.3, it follows that $L' = C' + 2\mathbb{Z}^{n-1}$ for some binary linear code $C' \subseteq \mathbb{F}_2^{n-1}$. Thus L' satisfies the induction hypothesis and we have the desired decomposition.

(ii) If RHS is strictly positive, then $|b_1| \in (0, 2) \cap \mathbb{Z} = \{1\}$. By Claim 4.3.2(3), we have that b has exactly two non-zero coordinates and they are ± 1. By permuting the coordinates of B, we may assume that $b \equiv (\pm 1, \pm 1, 0, \cdots, 0)$. Since we picked the row b to be the one with the smallest support, it follows that every row has at least 2 non-zero coordinates. By Claims 4.3.1 and 4.3.2, this is possible only if for every other row b', there exists $j' \in [n]$ such that $2|b'_{j'}| = \|b'\|^2$. By Claim 4.3.2(1) and (2), every other row b' has all its coordinates in $\{0, \pm 1, \pm 2\}$. By Claim 4.3.2(2), every other row b' has none of

its coordinates in $\{\pm 2\}$. Therefore, every other row b' has all its coordinates in $\{0, \pm 1\}$. By Claim 4.3.2(3), every row of the basis matrix has the same form as b: they have exactly two non-zero entries each of which is ± 1.

Since the rows of the basis matrix are orthogonal, it follows that the basis matrix B is a weighing matrix of order n with weight 2. By Theorem 4.2.1 the matrix B is obtained from $\oplus_{n/2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ by either negating some rows or columns and by interchanging rows or columns. We recall that interchanging or negating the rows of the basis matrix of a lattice preserves the basis property while interchanging columns is equivalent to permuting the coordinates. Hence $L_C = L(B) \cong \oplus_{i=1}^{n/2} L(\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix})$.

(2) $\equiv$ (3): We now show that L_C decomposes into a direct sum of lower dimensional lattices, $L_C \cong \oplus_i L_i$ if and only if the code C also decomposes, $C \cong \oplus_i C_i$.

Let $L_C \cong \oplus_i L_i$. Without loss of generality, we can consider $L_C = \oplus_i L_i$. We have $C = L_C \mod 2 = \oplus_i L_i \mod 2$. We observe that if L_i has dimension n_i, then $L_i \supseteq 2\mathbb{Z}^{n_i}$. Therefore, $C_i = L_i \mod 2$ is a binary code. Let $C_i := L_i \mod 2$ for every i. Then $C = \oplus_i C_i$. (If $c \in C$, then $c \in L$ and hence the projection of c to the subset of coordinates corresponding to L_i is in C_i. Let $c_i \in C_i$ for every i. The concatenated vector $\oplus_i c_i$ is in $\oplus_i L_i \mod 2$ and hence is in C).

To show the other direction of the equivalence, let $C \cong \oplus_i C_i$, where each $C_i \subseteq \mathbb{F}_2^{n_i}$ and $n = \sum_i n_i$. Therefore $L_C = C + 2\mathbb{Z}^n \cong \oplus_i C_i + 2\mathbb{Z}^n \cong \oplus_i (C_i + 2\mathbb{Z}^{n_i})$, since $\mathbb{Z}^n \cong \oplus_i \mathbb{Z}^{n_i}$. ■

4.3.2 Algorithm

Theorem 4.1.1 shows that a lattice of the form $C + 2\mathbb{Z}^n$ is orthogonal if and only if the underlying code decomposes into a direct sum of binary linear codes isomorphic to $\{0, 1\}$ or $\{0\}$ or the two dimensional code $\{00, 11\}$. We now give a polynomial time algorithm which finds the decomposition of the code C into the component codes, C_i,

if there exists one. Therefore, if the lattice L_C is orthogonal, the algorithm decides in polynomial time if it is orthogonal and also gives the orthogonal basis for the lattice.

The algorithm recursively attempts to find the component codes. If it is unable to decompose the code at any stage, then it declares that L_C is not orthogonal. At every step we check if $C \cong \{0,1\} \times C'$ or $\{0\} \times C'$ or $\{00,11\} \times C'$ and then recurse on C'.

Proof of Theorem 4.1.2 Given a basis for L_C as input, we first compute the generator for C. From Theorem 4.1.1, we know that if L_C is orthogonal, then $C \cong \oplus_i C_i$ where each C_i is either the length-1 code $\{0,1\}$ or the length-1 code $\{0\}$ or the 2-dimensional code $\{00, 11\}$.

The algorithm therefore in each step decides if $C \cong \{0,1\} \oplus C'$ or $C \cong \{0\} \oplus C'$ or $C \cong \{00,11\} \oplus C'$. Theorem 4.3.3 shows that using Algorithm 3 we can check in $O(n^4)$ time, if $C \cong \{0,1\} \oplus C'$. The same algorithm can be modified to check in $O(n^4)$ time, if $C \cong \{0\} \oplus C'$. Theorem 4.3.4 shows that Algorithm 4 can verify if $C \cong \{00,11\} \oplus C'$ in $O(n^5)$ time. If any one of the algorithms finds a decomposition, then we recurse on the lower dimensional code C' to find a further decomposition. We recurse at most n times. If all the algorithms fail to find a decomposition, then L_C is not orthogonal. Therefore, it takes $O(n^6)$ time to decide if L_C is orthogonal.

■

We now describe the individual algorithms to verify if $C \cong \{0,1\} \oplus C'$ or $C \cong \{0\} \oplus C'$ or $C \cong \{00,11\} \oplus C'$.

Theorem 4.3.3 *Let C be a binary linear code and $G = \{g_1, \ldots, g_n\} \in \mathbb{F}_2^{n \times n}$ be its generator. Then Algorithm 3 decides if $C \cong \{0,1\} \oplus C'$ for some linear code $C' \subseteq \mathbb{F}_2^{n-1}$ and if so outputs the coordinate corresponding to the direct sum decomposition. Moreover the algorithm runs in time $O(n^4)$.*

Proof For $j \in [n]$, let $C'_{\overline{j}} \subseteq \mathbb{F}_2^{n-1}$ be the projection of C on the indices $[n] \setminus \{j\}$ and for a vector $c \in C'_{\overline{j}}$, let $c^0, c^1 \in \mathbb{F}_2^n$ be extensions of c using $0,1$ respectively along the

Algorithm 3 : **decompose − length − 1(G)**:

Input: $G = \{g_1, \ldots, g_n\} \in \mathbb{F}_2^n$ (A generator for the code C)

1: **for** $j \in \{1, \cdots, n\}$ **do**
2: Let $G' \leftarrow$ projection of vectors in G on coordinates $[n] \setminus \{j\}$
3: For $g \in G'$, define $g^0, g^1 \in \mathbb{F}_2^n$ as the n-dimensional vectors obtained by extending g using 0 and 1 along the j'th coordinate respectively.
4: **if** $g^0,\ g^1 \in C$ for all $g \in G'$ **then**
5: **return** j
6: **end if**
7: **end for**
8: **return** FAIL

j'th coordinate. We note that $C \cong \{0,1\} \oplus C'$ for some binary linear code C' if and only if there exists an index $j \in [n]$, such that

$$C = \left\{ c^\ell \mid c \in C'_{\bar{j}}, \ell \in \{0,1\} \right\}.$$

From the definition of $C'_{\bar{j}}$, it follows that $C \subseteq \left\{ c^\ell \mid c \in C'_{\bar{j}}, \ell \in \{0,1\} \right\}$ up to a permutation of coordinates. So, the algorithm just needs to verify if the other side of the containment holds for some $j \in [n]$.

Let G' be the set of vectors of G projected on the coordinates $[n] \setminus \{j\}$. Algorithm 3 verifies if g^0 and g^1 are codewords in C, for every vector $g \in G'$. We now show that this is sufficient. Since C is a code, if $g^0,\ g^1 \in C$ for every $g \in G'$, then all linear combinations of these vectors are also in C. Therefore, $\left\{ c^\ell \mid c \in C'_{\bar{j}}, \ell \in \{0,1\} \right\} \subseteq C$.

It takes $O(n^2)$ time to compute a parity check matrix from the generator G and $O(n^2)$ time to verify if an input vector is a codeword using the parity check matrix. For every possible choice of the index j, Algorithm 3 checks if each of the $2n$ vectors of the form g^0, g^1 are in C. Therefore, Algorithm 3 takes $O(n^4)$ time to decide if $C \cong \{0,1\} \oplus C'$. ■

Algorithm 4 : **decompose − length − 2(G)**:

Input: $G = \{g_1, \ldots, g_n\} \in \mathbb{F}_2^n$ (A generator for the code C)

1: **for** $j_1, j_2 \in \{1, 2, \cdots, n\}$ **do**
2: Let $G' \leftarrow$ projection of vectors in G on coordinates $[n] \setminus \{j_1, j_2\}$
3: Let $G'' \leftarrow$ projection of vectors in G on coordinates $\{j_1, j_2\}$
4: **if** Code generated by $G'' \equiv \{00, 11\}$ **then**
5: For $g \in G'$ define $g^{00}, g^{11} \in \mathbb{F}_2^n$ be n-dimensional vectors obtained by extending g using 00 and 11 along the j_1, j_2 coordinates.
6: **if** $g^{00}, g^{11} \in C$ for all $g \in G'$ **then**
7: **return** j_1, j_2
8: **end if**
9: **end if**
10: **end for**
11: **return** FAIL

Theorem 4.3.4 *Let C be a binary linear code and $G = \{g_1, \ldots, g_n\} \in \mathbb{F}_2^{n \times n}$ be its generator. Then Algorithm 4 decides if $C \cong \{00, 11\} \oplus C'$ for some linear codes $C' \subseteq \mathbb{F}_2^{n-2}$ and if so outputs the coordinates corresponding to the direct sum decomposition. Moreover the algorithm runs in time $O(n^5)$.*

Proof For $j_1, j_2 \in [n]$, let C''_{j_1,j_2} be the projection of C on the indices $\{j_1, j_2\}$. We first verify if C''_{j_1,j_2} is the code $\{00, 11\}$. For this purpose, it is sufficient to check if C''_{j_1,j_2} is generated by $\{11\}$. Now, to see if $C \cong \{00, 11\} \oplus C'$ for some binary linear code $C' \subseteq \mathbb{F}_2^{n-2}$. Define $C'_{\bar{j_1},\bar{j_2}}$ to be the projection of C on the indices $[n] \setminus \{j_1, j_2\}$. For a vector $c \in C'_{\bar{j_1},\bar{j_2}}$, let $c^{00}, c^{11} \in \mathbb{F}_2^n$ be the extensions of c using $\{00, 11\}$ along the j_1, j_2 coordinates. We note that $C \cong \{00, 11\} \oplus C'$ for some binary linear code C' if and only if there exist indices $j_1, j_2 \in [n]$, such that

$$C = \left\{ c^{\ell} \mid c \in C'_{\bar{j_1},\bar{j_2}}, \ell \in \{00, 11\} \right\}. \tag{4.1}$$

From the definition of $C'_{\bar{j_1},\bar{j_2}}$ and $C''_{j_1,j_2} = \{00, 11\}$, it trivially follows that $C \subseteq \left\{c^\ell \mid c \in C'_{\bar{j_1},\bar{j_2}}, \ell \in \{00, 11\}\right\}$. So, the algorithm just needs to verify if the other side of the containment holds for some indices $j_1, j_2 \in [n]$.

Let G' be the set of vectors of G projected on the coordinates $[n] \setminus \{j_1, j_2\}$. Algorithm 4 verifies if g^{00} and g^{11} are codewords in C, for every vector $g \in G'$. We now show that this is sufficient. Since C is a code, if $g^{00}, g^{11} \in C$ for every $g \in G'$, then all linear combinations of these vectors are also in C. Therefore, $\left\{c^\ell \mid c \in C'_{\bar{j_1},\bar{j_2}}, \ell \in \{00, 11\}\right\} \subseteq C$.

For each choice of $\{j_1, j_2\}$, it takes $O(n)$ time to verify if $C''_{j_1,j_2} = \{00, 11\}$. Time to verify if an input vector is a codeword using the parity check matrix is $O(n^2)$. We perform this check for $2n$ vectors of the form $\{g^\ell \mid g \in G', \ell \in \{00, 11\}\}$.

It takes $O(n^3)$ time to verify if $C \cong \{00, 11\} \oplus C'_{\bar{j_1},\bar{j_2}}$ for every pair of indices $j_1, j_2 \in [n]$. There are at most $\binom{n}{2}$ possible choices of indices, j_1, j_2, therefore, it takes $O(n^5)$ time in total to decide if $C \cong \{00, 11\} \oplus C'$. ■

4.4 Orthogonal Lattices from Ternary Codes

In this section we focus on lattices obtained from ternary linear codes using Construction-A. In Section 4.4.1, we show that any orthogonal lattice obtained from a ternary linear code by Construction-A is equivalent to a product lattice whose components are one-dimensional or four-dimensional. In Section 4.4.2, we show that given a lattice obtained from a ternary linear code by Construction-A, there exists an efficient algorithm to verify if the lattice is orthogonal.

4.4.1 Decomposition Characterization

We prove Theorem 4.1.3 in this subsection.

Proof of Theorem 4.1.3 We show that (1) ≡ (2) and (2) ≡ (3) to complete the equivalence of the three statements.

(1) $\equiv$ (2): We show that $L_C = C + 3\mathbb{Z}^n$ is orthogonal if and only if it decomposes into a direct sum of lower dimensional orthogonal lattices, $L_C \cong \oplus_i L_i$.

If $L_C \cong \oplus_i L_i$ such that each L_i is orthogonal, then L_C is also orthogonal. This is because L_C has a block diagonal basis where each block is itself an orthogonal matrix (by definition, a 1×1-dimensional matrix is orthogonal).

We prove the other direction of the equivalence by induction on the dimension, n, of the lattice L_C. For the base case consider $n = 1$. Since L is integral and is of the form $C + 3\mathbb{Z}$ for some ternary code C, it follows that L has to be either $\mathbb{Z}$ or $3\mathbb{Z}$.

Let us assume the induction hypothesis for all $n-1$ or lower dimensional orthogonal lattices obtained from ternary linear codes using Construction-A.

Let L_C be an n-dimensional orthogonal lattice and B be an orthogonal basis of L_C with the rows being basis vectors. Since L_C is an integral lattice, B has only integral entries. The next two claims summarize certain properties of the entries of the basis matrix B.

Claim 4.4.1 *For every row b of B and for every $j \in [n]$, we have that $3|b_j| \in \{0, \|b\|^2, 3\|b\|^2\}$.*

Proof Since B is an orthogonal basis, $BB^T = D$, where D is the diagonal matrix with entries $\|b^{(i)}\|^2$, where $b^{(i)}$ denotes the i^{th} basis vector.

$$D = \begin{bmatrix} \|b^{(1)}\|^2 & 0 & 0 & \cdots & 0 \\ 0 & \|b^{(2)}\|^2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & \|b^{(n)}\|^2 \end{bmatrix}$$

We know that $3\mathbb{Z}^n \subseteq L_C$ so, $3e_j \in L_C$ for every $j \in [n]$. Therefore, there is an integral matrix $X \in \mathbb{Z}^{n \times n}$ such that $XB = 3I_n$, i.e. $3B^{-1} \in \mathbb{Z}^{n \times n}$. Since B is an orthogonal basis,

$$B^{-1} = B^T D^{-1} \in \frac{1}{3}\mathbb{Z}^{n \times n}.$$

Each column of $B^T D^{-1}$ is given by $b/\|b\|^2$, where b is a basis vector. Therefore, for any $j \in [n]$, we have

$$3b_j \equiv 0 \bmod \|b\|^2 \text{ for all } j \in [n], \text{ and rows } b \text{ of } B.$$

Since b_j is integral and $|b_j| \leq \|b\|^2$ for every $j \in [n]$, it follows that $3|b_j| \in \{0, \|b\|^2, 2\|b\|^2, 3\|b\|^2\}$. It now remains to exclude the case $3|b_j| = 2\|b\|^2$. Suppose there exists $j \in [n]$ such that $3|b_j| = 2\|b\|^2$. Since b is a basis vector, it follows that b is not all zeroes. Hence $b_j \neq 0$. We can re-write the condition $3|b_j| = 2\|b\|^2$ as $3|b_j| = 2\sum_{i=1}^{n} b_i^2$. Rearranging the terms, we have

$$|b_j|\ (3 - 2|b_j|) = 2\sum_{i \neq j} b_i^2.$$

Since the RHS is a sum of squares, it is always non-negative. The LHS is non-zero since $b_j \in \mathbb{Z}\backslash\{0\}$. So the LHS should be strictly positive. Therefore, $|b_j| \in (0, 3/2) \cap \mathbb{Z}$ and hence $|b_j| = 1$. However, this implies that $\sum_{i \neq j} b_i^2 = 1/2$, contradicting the fact that b is integral. Hence, $3||b_j|| = 2\|b\|^2$ is impossible. ■

Claim 4.4.2 *Let b be a row of B.*

(1) If there exists $j \in [n]$ such that $3|b_j| = 3\|b\|^2$, then $b_j = \pm 1$ and $b_{j'} = 0$ for every $j' \in [n] \setminus \{j\}$.

(2) If there exists $j \in [n]$ such that $3|b_j| = \|b\|^2$ and $b_j = \pm 3$, then $b_{j'} = 0$ for every $j' \in [n] \setminus \{j\}$.

(3) If there exists $j \in [n]$ such that $3|b_j| = \|b\|^2$ and $b_j = \pm 1$, then there exist $j_1, j_2 \in [n] \setminus \{j\}$, such that $|b_{j_1}| = |b_{j_2}| = 1$ and $b_{j'} = 0$ for every $j' \in [n] \setminus \{j, j_1, j_2\}$.

(4) If there exists $j \in [n]$ such that $3|b_j| = \|b\|^2$, then $b_{j'} \in \{0, \pm 1, \pm 3\}$ for every $j' \in [n]$.

Proof (1) Since, $\|b\|^2 = \sum_{i=1}^{n} b_i^2$, and each $b_i \in \mathbb{Z}$, we conclude that $|b_j| = 1$ and the remaining coordinates in b have to be 0, i.e $b_{j'} = 0$ for all $j' \in [n] \setminus \{j\}$.

(2) Follows from $3|b_j| = \|b\|^2$ and b being integral.

(3) We can re-write the condition $3|b_j| = \|b\|^2$ as $3|b_j| = \sum_{i=1}^{n} b_i^2$. Rearranging the terms, we have

$$|b_j| \ (3 - |b_j|) = \sum_{i \neq j} b_i^2. \tag{4.2}$$

If $b_j = \pm 1$, then $\sum_{i \neq j} b_i^2 = 2$. Further, b is integral. Hence, b has exactly 2 other non-zero coordinates b_{j_1}, b_{j_2}, $j \neq j_1, j_2$, such that $|b_{j_1}| = |b_{j_2}| = 1$.

(4) We have equation (4.2). The RHS is a sum of squares and hence the LHS is non-negative. Moreover, b is not all-zeroes vector implies that $b_j \neq 0$. Therefore, $|b_j| \in (0, 3] \cap \mathbb{Z}$. If $b_j = \pm 2$, then in order to satisfy $\sum_{i \neq j} b_i^2 = 2$ using integral b_i's, exactly two coordinates b_{j_1}, b_{j_2} should be ± 1, where $j \neq j_1, j_2$. However, in this case, $3|b_{j_1}| = 3|b_{j_2}| = 3 \notin \{0, \|b\|^2 = 6, 3\|b\|^2 = 18\}$, thus contradicting Claim 4.4.1. The conclusion follows from parts (2) and (3).

■

Using the properties of the orthogonal basis B of L_C given in Claims 4.4.1 and 4.4.2, we show that B is equivalent (up to permutations of its columns) to a block diagonal matrix, i.e

$$B \cong \begin{bmatrix} B_1 & 0 & \cdots & 0 \\ 0 & B_2 & \cdots & 0 \\ \vdots & & \ddots & 0 \\ 0 & 0 & \cdots & B_k \end{bmatrix}$$

where each B_i is either the 1×1 matrix $\begin{bmatrix} 1 \end{bmatrix}$ or the 1×1 matrix $\begin{bmatrix} 3 \end{bmatrix}$ or the 4×4 matrix obtained from M by negating a subset of its columns, $\mathcal{T}(M)$. It follows that $L_C \cong \oplus_i L_i$ such that B_i is the basis for the lower dimensional lattice L_i.

Let us pick a row b of B with the smallest support. Fix an index $j \in [n]$ to be the index of a non-zero entry with minimum absolute value in b, i.e. $j := \arg\min_k \{|b_k|\}$. Since b is a row of a basis matrix, b cannot be the all-zeroes vector and therefore there exists a $j \in [n]$ such that $|b_j| > 0$. Since we are only interested in equivalence (that

allows for permutation of coordinates), we may assume without loss of generality that $j = 1$ by permuting the coordinates. By Claim 4.4.1, we have that $3|b_1| \in \{\|b\|^2, 3\|b\|^2\}$. We consider each of these cases separately.

1. Suppose $3|b_1| = 3\|b\|^2$. By Claim 4.4.2(1), $b = (\pm 1, 0, \ldots, 0)$. Since B is an orthogonal basis, $\langle b, b' \rangle = 0 \Rightarrow b'_1 = 0$ for all $b' \neq b \in B$. The orthogonality of B therefore forces all other basis vectors to take a value of 0 in the 1^{st} coordinate. Thus B is of the form

$$B = \left(\begin{array}{c|c} \pm 1 & 0 \cdots 0 \\ \hline 0 & \\ \vdots & B' \\ 0 & \end{array}\right).$$

   Therefore, we obtain $L_C \cong \mathbb{Z} \oplus L'$, where L' is an orthogonal $(n-1)$-dimensional lattice generated by the basis matrix restricted to the coordinates other than 1, say, B'. From Claim 2.2.3, it follows that $L' = C' + 3\mathbb{Z}^{n-1}$ for some ternary linear code $C' \subseteq \mathbb{F}_3^{n-1}$. Thus L' satisfies the induction hypothesis and we have the desired decomposition.

2. Suppose $3|b_1| = \|b\|^2$. We can re-write this condition as $3|b_1| = \sum_{i=1}^{n} b_i^2$. Rearranging the terms, we have

$$|b_1| \ (3 - |b_1|) = \sum_{i \neq 1} b_i^2.$$

   Since the RHS is a sum of squares, it should be non-negative.

   (i) If RHS is 0, then $b_1 = \pm 3$ and therefore, it follows from Claim 4.4.2(2) that $b = (\pm 3, 0, \ldots, 0)$. The orthogonality of B forces all other basis vectors to take a value of 0 in the 1^{st} coordinate.

$$B = \left(\begin{array}{c|c} \pm 3 & 0 \cdots 0 \\ \hline 0 & \\ \vdots & B' \\ 0 & \end{array}\right)$$

Therefore, we obtain $L_C \cong 3\mathbb{Z} \oplus L'$, where L' is an orthogonal $(n-1)$-dimensional lattice generated by the basis matrix restricted to the coordinates other than 1, say B'. From Claim 2.2.3, it follows that $L' = C' + 3\mathbb{Z}^{n-1}$ for some ternary linear code $C' \subseteq \mathbb{F}_3^{n-1}$. Thus L' satisfies the induction hypothesis and we have the desired decomposition.

(ii) If RHS is strictly positive, then $|b_1| \in (0,3) \cap \mathbb{Z} = \{1,2\}$. By Claim 4.4.2(4), $b_1 \neq \pm 2$. Therefore, $b_1 = \pm 1$. By Claim 4.4.2(3), we have that b has exactly three non-zero coordinates and they are ± 1. By permuting the coordinates of B, we may assume that $b \equiv (\pm 1, \pm 1, \pm 1, 0, \cdots, 0)$.

Since we picked the row b to be the one with the smallest support, it follows that every row has at least 3 non-zero coordinates. By Claims 4.4.1 and 4.4.2, this is possible only if for every other row b', there exists $j' \in [n]$ such that $3|b'_{j'}| = \|b'\|^2$. By Claim 4.4.2(4), every other row b' has all its coordinates in $\{0, \pm 1, \pm 3\}$. By Claim 4.4.2(2), every other row b' has none of its coordinates in $\{\pm 3\}$. Therefore, every other row b' has all its coordinates in $\{0, \pm 1\}$. By Claim 4.4.2(3), every row of the basis matrix has the same form as b: they have exactly three non-zero entries each of which is ± 1.

Since the rows of the basis matrix are orthogonal, it follows that the basis matrix B is a weighing matrix of order n with weight 3. By Theorem 4.2.3, the matrix B is obtained from $\oplus_{n/4} M$ by either negating some rows or columns and by interchanging rows or columns. We recall that interchanging or negating the rows of the basis matrix of a lattice preserves the basis property while interchanging columns is equivalent to permuting the coordinates. Hence $L_C = L(B) \cong \oplus_{i=1}^{n/4} L(\mathcal{T}_i(M))$, where each $\mathcal{T}_i(M)$ is a 4×4 matrix obtained by negating a subset of columns of M.

(2) $\equiv$ (3): We now show that L_C decomposes into a direct sum of lower dimensional lattices, $L_C \cong \oplus_i L_i$ if and only if the code C also decomposes, $C \cong \oplus_i C_i$.

Let $L_C \cong \oplus_i L_i$. Without loss of generality, we can consider $L_C = \oplus_i L_i$. We have $C = L_C \mod 3 = \oplus_i L_i \mod 3$. We observe that if L_i has dimension n_i, then $L_i \supseteq 3\mathbb{Z}^{n_i}$. Therefore, $C_i = L_i \mod 3$ is a ternary code. Let $C_i := L_i \mod 3$ for every i. Then $C = \oplus_i C_i$. (If $c \in C$, then $c \in L$ and hence the projection of c to the subset of coordinates corresponding to L_i is in C_i. Let $c_i \in C_i$ for every i. The concatenated vector $\oplus_i c_i$ is in $\oplus_i L_i \mod 3$ and hence is in C.)

To show the other direction of the equivalence, let $C \cong \oplus_i C_i$, where each $C_i \subseteq \mathbb{F}_3^{n_i}$ and $n = \sum_i n_i$. Therefore $L_C = C + 3\mathbb{Z}^n \cong \oplus_i C_i + 3\mathbb{Z}^n \cong \oplus_i (C_i + 3\mathbb{Z}^{n_i})$, since $\mathbb{Z}^n \cong \oplus_i \mathbb{Z}^{n_i}$. ■

4.4.2 Algorithm

Theorem 4.1.3 shows that a lattice of the form $C + 3\mathbb{Z}^n$ is orthogonal if and only if the underlying code decomposes into a direct sum of ternary linear codes isomorphic to $\{0, 1, 2\}$ or $\{0\}$ or the four dimensional code generated by $\mathcal{T}(M) \mod 3$, where $\mathcal{T}(M)$ is obtained from matrix M by negating a subset of its columns. We now give a polynomial time algorithm which finds the decomposition of the code C into the component codes, C_i, if there exists one. Therefore, if the lattice L_C is orthogonal, the algorithm decides in polynomial time if it is orthogonal and also gives the orthogonal basis for the lattice.

The algorithm recursively attempts to find the component codes. If it is unable to decompose the code at any stage, then it declares that L_C is not orthogonal. At every step we check if $C \cong \{0, 1, 2\} \times C'$ or $\{0\} \times C'$ or $C_{\mathcal{T}(M)} \times C'$ where $C_{\mathcal{T}(M)}$ is the code generated by $\mathcal{T}(M) \mod 3$ and then recurse on C'.

Proof of Theorem 4.1.4 Given a basis for L_C as input, we first compute the generator for C. From Theorem 4.1.3, we know that if L_C is orthogonal, then $C \cong \oplus_i C_i$ where each C_i is either the length-1 code $\{0, 1, 2\}$ or the length-1 code $\{0\}$ or a 4-dimensional code generated by the rows of $\mathcal{T}(M) \mod 3$ where $\mathcal{T}(M)$ obtained from matrix M by negating a subset of its columns.

The algorithm therefore in each step decides if $C \cong \{0,1,2\} \oplus C'$ or $C \cong \{0\} \oplus C'$ or $C \cong C_{\mathcal{T}(M)} \oplus C'$, where $C_{\mathcal{T}(M)}$ denotes the code generated by $\mathcal{T}(M)$ mod 3. Theorem 4.4.3 shows that using Algorithm 5 we can check in $O(n^4)$ time, if $C \cong \{0,1,2\} \oplus C'$. The same algorithm can be modified to check in $O(n^4)$ time, if $C \cong \{0\} \oplus C'$. Theorem 4.4.4 shows that Algorithm 6 can verify if $C \cong C_{\mathcal{T}(M)} \oplus C'$ in $O(n^7)$ time. If any one of the algorithms finds a decomposition, then we recurse on the lower dimensional code C' to find a further decomposition. We recurse at most n times. If all the algorithms fail to find a decomposition, then L_C is not orthogonal. Therefore, it takes $O(n^8)$ time to decide if L_C is orthogonal. ■

We now describe the individual algorithms to verify if $C \cong \{0,1,2\} \oplus C'$ or $C \cong \{0\} \oplus C'$ or $C \cong C_{\mathcal{T}(M)} \oplus C'$.

Algorithm 5 : **decompose − length − 1(G)**:

Input: $G = \{g_1, \ldots, g_n\} \in \mathbb{F}_3^n$ (A generator for the code C)

1: **for** $j \in \{1, \cdots, n\}$ **do**
2: Let $G' \leftarrow$ projection of vectors in G on coordinates $[n] \setminus \{j\}$
3: For $g \in G'$, define $g^0, g^1, g^2 \in \mathbb{F}_3^n$ as the n-dimensional vectors obtained by extending g using 0, 1 and 2 along the j'th coordinate respectively.
4: **if** $g^0,\ g^1,\ g^2 \in C$ for all $g \in G'$ **then**
5: **return** j
6: **end if**
7: **end for**
8: **return** FAIL

Theorem 4.4.3 *Let C be a ternary linear code and $G = \{g_1, \ldots, g_n\} \in \mathbb{F}_3^{n \times n}$ be its generator. Then Algorithm 5 decides if $C \cong \{0,1,2\} \oplus C'$ for some linear code $C' \subseteq \mathbb{F}_3^{n-1}$ and if so outputs the coordinate corresponding to the direct sum decomposition. Moreover the algorithm runs in time $O(n^4)$.*

Proof For $j \in [n]$, let $C'_{\overline{j}} \subseteq \mathbb{F}_3^{n-1}$ be the projection of C on the indices $[n] \setminus \{j\}$ and for a vector $c \in C'_{\overline{j}}$, let $c^0, c^1, c^2 \in \mathbb{F}_3^n$ be extensions of c using $0, 1, 2$ respectively along the j'th coordinate. We note that $C \cong \{0,1,2\} \oplus C'$ for some ternary linear code C' if and only if there exists an index $j \in [n]$, such that

$$C = \left\{ c^\ell \mid c \in C'_{\overline{j}}, \ell \in \{0,1,2\} \right\}.$$

From the definition of $C'_{\overline{j}}$, it follows that $C \subseteq \left\{ c^\ell \mid c \in C'_{\overline{j}}, \ell \in \{0,1,2\} \right\}$ up to a permutation of coordinates. So, the algorithm just needs to verify if the other side of the containment holds for some $j \in [n]$.

Let G' be the set of vectors of G projected on the coordinates $[n] \setminus \{j\}$. Algorithm 5 verifies if g^0, g^1 and g^2 are codewords in C, for every vector $g \in G'$. We now show that this is sufficient. Since C is a code, if g^0, g^1, $g^2 \in C$ for every $g \in G'$, then all linear combinations of these vectors are also in C. Thus, $\left\{ c^\ell \mid c \in C'_{\overline{j}}, \ell \in \{0,1,2\} \right\} \subseteq C$.

It takes $O(n^2)$ time to compute a dual code basis from the generator G and $O(n^2)$ time to verify if an input vector is a codeword using the dual basis. For every possible choice of the index j, Algorithm 5 checks if each of the $3n$ vectors of the form g^0, g^1, g^2 are in C. Therefore, Algorithm 5 takes $O(n^4)$ time to decide if $C \cong \{0,1,2\} \oplus C'$. ■

Theorem 4.4.4 *Let C be a ternary linear code and $G = \{g_1, \ldots, g_n\} \in \mathbb{F}_3^{n \times n}$ be its generator. For a matrix $\mathcal{T}(M)$ obtained by negating a subset of columns of M, let $C_{\mathcal{T}(M)}$ be the length-4 code whose generators are the rows of $\mathcal{T}(M)$. Then Algorithm 6 decides if $C \cong C_{\mathcal{T}(M)} \oplus C'$ for some linear codes $C' \subseteq \mathbb{F}_3^{n-4}$ and $C_{\mathcal{T}(M)} \subseteq \mathbb{F}_3^4$ and if so outputs the coordinates corresponding to the direct sum decomposition as well as the matrix $\mathcal{T}(M)$. Moreover the algorithm runs in time $O(n^7)$.*

Proof For $1 \leq j_1 < j_2 < j_3 < j_4 \leq n$, let C''_{j_1,j_2,j_3,j_4} be the projection of C on the indices $\{j_1, j_2, j_3, j_4\}$. We first verify if C''_{j_1,j_2,j_3,j_4} is the code generated by the rows of $\mathcal{T}(M)$ (denoted as $C_{\mathcal{T}(M)}$) for some $\mathcal{T}(M)$ which is obtained by negating a subset of columns of M. We would like to check if every $c \in C''_{j_1,j_2,j_3,j_4}$ is in $C_{\mathcal{T}(M)}$ and vice versa. For this purpose, it is sufficient to check if the generator vectors of C''_{j_1,j_2,j_3,j_4}

Algorithm 6 : **decompose − length − 4(G)**:

Input: $G = \{g_1, \ldots, g_n\} \in \mathbb{F}_3^n$ (A generator for the code C)

1: **for** $j_1, j_2, j_3, j_4 \in \{1, 2, \cdots, n\}$ **do**
2: Let $G' \leftarrow$ projection of vectors in G on coordinates $[n] \setminus \{j_1, j_2, j_3, j_4\}$
3: Let $G'' \leftarrow$ projection of vectors in G on coordinates $\{j_1, j_2, j_3, j_4\}$
4: **for** $S \subseteq [4]$ **do**
5: Let $\mathcal{T}(M) \leftarrow M$ with columns in S negated
6: **if** $C_{\mathcal{T}(M)} \equiv$ Code generated by G'' **then**
7: For $g \in G'$ define $g^{p_1}, g^{p_2}, g^{p_3}, g^{p_4} \in \mathbb{F}_3^n$ be n-dimensional vectors obtained by extending g using the rows of $\mathcal{T}(M)$ along the j_1, j_2, j_3, j_4 coordinates.
8: **if** $g^{p_1}, g^{p_2}, g^{p_3}, g^{p_4} \in C$ for all $g \in G'$ **then**
9: **return** j_1, j_2, j_3, j_4 and $\mathcal{T}(M)$
10: **end if**
11: **end if**
12: **end for**
13: **end for**
14: **return** FAIL

are codewords in $C_{\mathcal{T}(M)}$ and each row of $\mathcal{T}(M)$ is a codeword in C''_{j_1,j_2,j_3,j_4}. We know that the generators of C''_{j_1,j_2,j_3,j_4} are contained in G'' where G'' is the set of vectors in G projected on the indices $\{j_1, j_2, j_3, j_4\}$.

Once we fix $\mathcal{T}(M)$ such that $C''_{j_1,j_2,j_3,j_4} = C_{\mathcal{T}(M)}$, it remains to verify if $C \cong C_{\mathcal{T}(M)} \oplus C'$ for some ternary linear code $C' \subseteq \mathbb{F}_3^{n-4}$. Define $C'_{\bar{j_1},\bar{j_2},\bar{j_3},\bar{j_4}}$ to be the projection of C on the indices $[n] \setminus \{j_1, j_2, j_3, j_4\}$. For a vector $c \in C'_{\bar{j_1},\bar{j_2},\bar{j_3},\bar{j_4}}$, let $c^p \in \mathbb{F}_3^n$ be the extensions of c using a codeword $p \in C_{\mathcal{T}(M)}$ along the j_1, j_2, j_3, j_4 coordinates. We note that $C \cong C_{\mathcal{T}(M)} \oplus C'$ for some ternary linear code C' if and only if there exist indices $j_1, j_2, j_3, j_4 \in [n]$, such that

$$C = \left\{ c^p \mid c \in C'_{\bar{j_1},\bar{j_2},\bar{j_3},\bar{j_4}}, p \in C_{\mathcal{T}(M)} \right\}. \tag{4.3}$$

From the definition of $C'_{\bar{j_1},\bar{j_2},\bar{j_3},\bar{j_4}}$ and C''_{j_1,j_2,j_3,j_4} $(= C_{\mathcal{T}(M)})$, it follows that $C \subseteq \{c^p \mid c \in C'_{\bar{j_1},\bar{j_2},\bar{j_3},\bar{j_4}}, p \in C_{\mathcal{T}(M)}\}$. So, the algorithm just needs to verify if the other side of the containment holds for some indices $j_1, j_2, j_3, j_4 \in [n]$.

Let G' be the set of vectors of G projected on the coordinates $[n] \setminus \{j_1, j_2, j_3, j_4\}$. Algorithm 6 verifies if g^{p_0}, g^{p_1}, g^{p_3} and g^{p_4} are codewords in C, for every vector $g \in G'$. We now show that this is sufficient. Since C is a code, if $g^{p_0}, g^{p_1}, g^{p_3}, g^{p_4} \in C$ for every $g \in G'$ and $p_i \in \mathcal{T}(M)$, then all linear combinations of these vectors are also in C. Therefore, $\{c^p \mid c \in C'_{\bar{j_1},\bar{j_2},\bar{j_3},\bar{j_4}}, p \in C_{\mathcal{T}(M)}\} \subseteq C$.

There are $2^4 4^4$ possible choices of $\mathcal{T}(M)$ including permutations. For each matrix $\mathcal{T}(M)$, it takes $O(n)$ time to verify if $C_{\mathcal{T}(M)} = C''_{j_1,j_2,j_3,j_4}$. Time to verify if an input vector is a codeword using the dual basis is $O(n^2)$. We perform this check for $4n$ vectors of the form $\{g^{p_0}, g^{p_1}, g^{p_3}, g^{p_4} \mid g \in G'\}$. So, for a given $\mathcal{T}(M)$ such that $C_{\mathcal{T}(M)} = C''_{j_1,j_2,j_3,j_4}$, It takes $O(n^3)$ time to verify $C \cong C_{\mathcal{T}(M)} \oplus C'$.

For every possible choice of indices, $\{j_1, j_2, j_3, j_4\}$, Algorithm 6 takes $O(n^3)$ time to verify if $C \cong C_{\mathcal{T}(M)} \oplus C'_{\bar{j_1},\bar{j_2},\bar{j_3},\bar{j_4}}$. Since there are at most $\binom{n}{4}$ possible choices of indices, it takes $O(n^7)$ time in total to decide if $C \cong C_{\mathcal{T}(M)} \oplus C'$. ■

4.5 Discussion and Open Questions

An immediate open question that arises from this work is to extend the result to general Construction-A lattices. Extending our results to values $q > 3$ might require new techniques. For larger q, a decomposition characterization seems to require a complete characterization of *weighing matrices* of weight q which is a known open problem. In particular, a direct product decomposition characterization of weighing matrices for the case of $q = 4$ is known [79]. However, even in this case the parts in the direct product decomposition may not be of constant dimension, so designing an efficient algorithm for the orthogonality decision problem through a direct product decomposition characterization appears to be non-trivial.

Recall that the main motivation to study this problem was to decide orthogonality (see Question 1.0.1) for general lattices which still seems elusive.

5 LOCAL TESTING OF LATTICES

In this chapter we initiate the study of local testability for membership in *point lattices*. Given a basis for a lattice L, we are interested in testing if a given input $t \in \mathbb{R}^n$ belongs to L, or is far from all points in L by querying a small number of coordinates of t. We emphasize that this setting does not limit the computational space or time in pre-processing the lattice as well as the queried coordinates. The main goal is to design a tester that queries only a small number of coordinates of the input.

Next, we motivate the model and present some potential applications of locally testable lattices.

Complexity theory. Lattices can be seen as coding theoretic objects naturally bringing features of error-correcting codes from the finite field domain to the real domain. As such, a study of local testing (and correction) procedures for lattices naturally extends the classical notions of Locally Testable Codes (LTCs) and Locally Decodable Codes (LDCs), which are in turn of significance to computational complexity theory (for example in constructing probabilistically checkable proofs and hardness amplification, among numerous other applications). Characterizing local testability, explicitly initiated by Kaufman and Sudan [80], has been an intensely investigated direction in the study of LTCs. We believe that an analogous investigation of lattices is likely to bring new insights and new connections in property testing.

Integer Programming. Lattices are the fundamental structures underlying integer programming problems. An integer programming problem (IP) is specified by a constraint matrix $A \in \mathbb{R}^{n \times m}$, a vector $b \in \mathbb{R}^n$. The goal is to verify if there exists an integer solution to the system $Ax = b, x \geq 0$. Although IP is NP-complete [81],

its instances are solved routinely in practice using cutting planes and branch-and-cut techniques [82]. The relaxed problem of verifying integer feasibility of the system $Ax = b$ is equivalent to verifying whether b lies in the lattice generated by the columns of A. Thus, the relaxation problem is the membership testing problem in a lattice. It is solvable efficiently and is a natural pre-processing step to solving IPs. Furthermore, if the number of constraints n in the problem is very large, then it would be helpful to run a tester that reads only a partial set of coordinates of the input b to verify if b could lie in the lattice generated by the columns of A or is far from it. If the test rejects, then this saves on the computational effort to search for a non-negative solution.

Cryptography. In cryptographic applications, it is imperative to understand which lattices are difficult to test in order to ensure security of lattice-based cryptosystems. In some cryptanalytic attacks on lattice-based cryptosystems, one needs to distinguish target vectors that are close to lattice vectors from those that are far from all lattice vectors, a problem commonly known as the gap version of the Closest Vector Problem (GapCVP). An approach to address GapCVP is to use expensive distance estimation algorithms inspired by Aharonov and Regev [44] and Liu *et al.* [83]. Local testing of lattices is closely related to both distance estimation [45] and GapCVP, and hence progress in the proposed testing model could lead to new insights in cryptanalytic attacks.

Lattice-based communication. Lattices are a major technical tool in communication systems as the analogue of error-correcting codes over reals, for applications such as wireless communication and transmission over analog lines. In lattice-coding, the message m is mapped to a point c in a chosen lattice L. The codeword c is transmitted over an analog channel. If the encoded message gets corrupted by the channel, then the channel output may not be a lattice point, thus enabling transmission error detection. In order to correct errors, computationally expensive decoding algorithms

are employed. Instead, the receiver may perform a local test for membership in the lattice beforehand, allowing the costly decoding computation to run only when there is a reasonably high chance of correct decoding.

We now give an informal description of our testing model motivated by its application in lattice-coding. The transmission of each coordinate of a lattice-codeword over the analog channel consumes power that is proportional to the square of the transmitted value. Thus the power consumption for transmitting the lattice-codeword $c \in L \subset \mathbb{R}^n$ is proportional to its squared ℓ_2 norm. The power consumption for transmitting a codeword over the channel is usually constrained by a power budget. The noise vector is also subject to a bound on its power. The power budget for transmission is typically formulated by considering the lattice-code $C(L)$ defined by the set of lattice points $c \in L$ that satisfy $\sum_{i=1}^{n} c_i^2 \leq \sigma n$ for some constant power budget $\sigma > 0$. In order to ensure that the receiver can tolerate adversarial noise budget δ per channel use, the shortest nonzero vector $v \in L$ should be such that $\sum_{i=1}^{n} v_i^2 \geq \delta n$. Thus, the *relative distance* of the lattice-code $C(L)$ is defined to be $\sum_{i=1}^{n} v_i^2/n$, where $v \in L$ is a shortest nonzero lattice vector. The *rate* of a lattice-code $C(L)$ is defined to be $(1/n) \log |C(L)|$ (note that this quantity could be larger than 1). An *asymptotically good family of lattices*, in this work, is one that achieves rate and relative distance that are both lower bounded by a positive constant. Such families are ideal for use in noisy communication channels.

We define a notion of a tester that will be useful as a pre-processor for decoding, and is similar to the established notion of code testing: An ℓ_2-tester of a lattice L for a given distance parameter $\epsilon > 0$ is a probabilistic procedure that given an input $t \in \mathbb{R}^n$, queries at most q coordinates of t, accepts with probability at least $2/3$ if $t \in L$, and rejects with probability at least $2/3$ if $\sum_{i=1}^{n} (t_i - w_i)^2 \geq \epsilon n$ for every $w \in L$.

In the next section we propose two main directions of research (Questions 5.1.2 and 5.1.3) analogous to similar directions that are fundamental to the study of locally testable codes.

5.1 Testing Model

In the above application, we focused on ℓ_2 distances. We now formalize the notion of testing lattices for ℓ_p distances. We consider ℓ_p distances since these are natural notions for real-valued inputs [84]. Denote the ℓ_p norm of the real vector 1^n by $\|1^n\|_p$. We focus on integral lattices, which are sub-lattices of $\mathbb{Z}^n$, as these are the most commonly encountered lattices in applications[1].

Definition 5.1.1 (Local test for lattices) *An ℓ_p-tester $T(\epsilon, c, s, q)$ for a lattice $L \subseteq \mathbb{Z}^n$ is a probabilistic algorithm that queries q coordinates of the input $t \in \mathbb{R}^n$, and*

- *(completeness)* accepts *with probability at least $1 - c$ if $t \in L$,*
- *(soundness)* rejects *with probability at least $1 - s$ if $d_p(t, L) \geq \epsilon \cdot \|1^n\|_p$ (we call such a vector t to be ϵ-far from L).*

If T always accepts inputs t that are in the lattice L then it is called 1-sided, otherwise it is 2-sided. If the queries performed by T depend on the answers to the previous queries, then T is called adaptive, otherwise it is called non-adaptive.

A test $T(\epsilon, 0, 0, q)$ is a test with perfect completeness and perfect soundness. 1-sided testers (i.e., testers with perfect completeness) are useful as a pre-processing step, as mentioned earlier. An asymptotically good family of lattices $L(n)$ for ℓ_p distances is one that has ℓ_p-relative distance lower bounded by a constant (i.e., $\min_{v \in L(n)} \|v\|_p^p / n = \Omega(1)$) and has $2^{\Omega(n)}$ lattice points in the origin-centered ℓ_p-ball of radius $n^{1/p}$.

Similar to the application in lattice-coding and locally testable codes, a main question in ℓ_p-testing of lattices is the following:

Question 5.1.2 *Is there an asymptotically good family of lattices that can be tested for membership with constant number of queries?*

[1] Arbitrary lattices can be approximated by rational lattices and rational lattices can be scaled to integral lattices.

Motivated by the applications in IP and cryptography, we identify another fundamental question in ℓ_p-testing of lattices:

Question 5.1.3 *What properties of a given lattice enable the design of ℓ_p-testers with constant query complexity?*

Tolerant Testing. Many applications can tolerate a small amount of noise in the input. Parnas *et al.* [45] introduced the notion of tolerant testing to account for a small amount of noise in the input. Tolerant testing has been studied in the context of codes (e.g. [46, 47]), and in the context of properties of real-valued data in the ℓ_p norm (e.g. [84]). We extend the tolerant testing model to lattices as follows.

Definition 5.1.4 (Tolerant test for lattices) *An ℓ_p-tolerant-tester $T(\epsilon_1, \epsilon_2, c, s, q)$ for a lattice $L \subseteq \mathbb{Z}^n$ is a probabilistic algorithm that queries q coordinates of the input $t \in \mathbb{R}^n$, and*

- *(completeness)* accepts *with probability at least $1 - c$ if $d_p(t, L) \leq \epsilon_1 \cdot \|1^n\|_p$,*
- *(soundness)* rejects *with probability at least $1 - s$ if $d_p(t, L) \geq \epsilon_2 \cdot \|1^n\|_p$.*

Tolerant testing with parameter $\epsilon_1 = 0$ corresponds to the notion of testing given in Definition 5.1.1. Tolerant testing and distance approximation are closely related notions. In fact, in the Hamming space, the ability to perform tolerant testing for *every* choice of $\epsilon_1 < \epsilon_2$ can be exploited to approximate distances [45].

5.2 Results

We initiate the study of membership testing in point lattices from the perspective of sublinear algorithms aiming to lay the ground work for further advances towards resolving Question 5.1.2 and Question 5.1.3. Our contributions draw on connections between lattices and codes, and on well-known techniques in property testing.

5.2.1 Upper and Lower Bounds for Testing Specific Lattice Families

Motivated by applications in lattice-based communication, we focus on an asymptotically good family of sets constructed from linear codes known as "code-formula" lattices (see Chapter 2, Section 2.2.1 for formal definition). We show upper and lower bounds on the query complexity of ℓ_1-testers for code formulas, as a function of the query complexity of the constituent code testers.

In this work we design a tester for code formula lattices using testers for the constituent codes.

Theorem 5.2.1 *Let $0 < \epsilon, s < 1$ and $C_0 \subseteq C_1 \subseteq \cdots \subseteq C_{m-1} \subseteq \{0,1\}^n$ be a family of binary linear codes satisfying the Schur product condition. Suppose every C_i has a 1-sided tester $T_i(\epsilon/m2^{i+1}, 0, s, q_i)$. Then, there exists an ℓ_1-tester $T(\epsilon, 0, s, q)$ for the lattice $L(\langle C_i \rangle_{i=0}^{m-1})$ with query complexity*

$$q = O\left(\frac{1}{\epsilon}\log\frac{1}{s}\right) + \sum_{i=1}^{m-1} q_i.$$

Next, we show a lower bound on the query complexity for testing code formula lattices, using lower bounds for testing the constituent codes.

Theorem 5.2.2 *Let $0 < \epsilon, c, s < 1$ and $C_0 \subseteq C_1 \subseteq \cdots \subseteq C_{m-1} \subseteq \{0,1\}^n$ be a family of binary linear codes satisfying the Schur product condition. Let $q_i = q_i(\epsilon, c, s)$ be such that any (possibly adaptive, 2-sided) ℓ_1-tester $T_i(\epsilon, c, s, q')$ for C_i satisfies $q' = \Omega(q_i)$, for every $i = 0, 1, \ldots, m-1$. Then every (possibly adaptive, 2-sided) ℓ_1-tester $T(\epsilon, c, s, q)$ for the lattice $L(\langle C_i \rangle_{i=0}^{m-1})$ has query complexity*

$$q = \Omega\left(\max\left\{\frac{1}{\epsilon}\log\frac{1}{s}, \max_{i=0,1,\ldots,m-1} q_i\right\}\right).$$

Code formula lattices from Reed-Muller codes. We instantiate the upper and lower bounds on the query complexity for a common family of code formula lattices constructed using Reed-Muller codes [40] to obtain nearly matching upper and lower bounds. We recall Reed-Muller codes below.

Definition 5.2.3 (Reed Muller Codes) *Each codeword of a binary Reed-Muller code* $RM(k,r) \subseteq \mathbb{F}_2^{2^r}$ *corresponds to a polynomial* $p(x) \in \mathbb{F}_2[x]$ *in* r *variables of degree at most* k *evaluated at all* 2^r *possible inputs* $x \in \mathbb{F}_2^r$.

For the family of Reed-Muller codes in $\mathbb{F}_2^{2^r}$, it is well-known that

$$RM(0,r) \subseteq RM(1,r) \subseteq RM(2,r) \subseteq \cdots \subseteq RM(r-1,r) \subseteq RM(r,r) = \mathbb{F}_2^{2^r}.$$

A particular family of RM codes that leads to code formula lattices is $\langle RM(k_i,r)\rangle_{i=0}^{\log r}$, with $k_i = 2^i$. Indeed, it can be easily verified that this family satisfies the Schur product condition since Reed-Muller codewords are evaluation tables of multivariate polynomials over the binary field and product of two degree k polynomials is a degree $2k$ polynomial. Hence for height $m \leq \log r$ the construction $\langle RM(2^i,r)\rangle_{i=0}^{m-1}$ gives rise to a lattice. We note these lattices have small relative minimum distance and are not asymptotically good families of lattices.

Corollary 5.2.4 *Let* $0 \leq k_0 < k_1 < \cdots < k_{m-1} < r$ *be integers such that the family of Reed-Muller codes* $RM(k_0,r) \subseteq RM(k_1,r) \subseteq \cdots \subseteq RM(k_{m-1},r)$ *satisfies the Schur product condition. Let* $0 < \epsilon, s < 1$ *and* L *be the lattice obtained from this family of codes using the code formula construction:*

$$L = RM(k_0,r) + 2RM(k_1,r) + \cdots + 2^{m-1}RM(k_{m-1},r) + 2^m\mathbb{Z}^{2^r}.$$

Then, there exists an ℓ_1*-tester* $T(\epsilon,0,s,q)$ *for* L *with query complexity*

$$q(\epsilon,s) = O\left(2^{k_{m-1}} \cdot \frac{1}{\epsilon}\log\frac{1}{s}\right).$$

In particular, when the height m and the degrees are constant, the query complexity of the tester is a constant.

For the lower bound, we obtain the following corollary using known lower bounds for testing Reed-Muller codes.

Corollary 5.2.5 *Let $0 \leq k_0 < k_1 < \cdots < k_{m-1} < r$ be integers such that the family of Reed-Muller codes $RM(k_0, r) \subseteq RM(k_1, r) \subseteq \cdots \subseteq RM(k_{m-1}, r)$ satisfies the Schur product condition. Let $0 < \epsilon, c, s < 1$ be constants and L be the lattice obtained from this family of codes using the code formula construction:*

$$L = RM(k_0, r) + 2RM(k_1, r) + \cdots + 2^{m-1}RM(k_{m-1}, r) + 2^m\mathbb{Z}^{2^r}.$$

Then, every (possibly 2-sided, adaptive) ℓ_1-tester $T(\epsilon, c, s, q)$ for L requires

$$q = \Omega(2^{k_{m-1}}).$$

We note that for code formula lattices of constant height obtained from Reed-Muller codes, Corollaries 5.2.4 and 5.2.5 show matching bounds (up to a constant factor depending on ϵ, s).

Random Lattices. There exists a distribution of random lattices which are impossible to test with small number of queries. This follows from Theorem 5.2.2 and considering random codes, which typically need at least a linear number of queries to test. We illustrate a concrete example by considering the following distribution of random lattices [85, 86]: For constants $b < a$, let $m = nb/a$ and let $H \in \mathbb{F}_2^{m \times n}$ be a random matrix such that each row and column has exactly a and b non-zeroes respectively. Consider the linear code $C_{a,b} := \{x \in \mathbb{F}_2^n : Hx = 0 (\text{mod } 2)\}$ and the code formula lattice $L(C_{a,b})$ associated with the linear code $C_{a,b}$.

Theorem 5.2.6 *There exist constants a, b, ϵ, c, s such that every (possibly 2-sided, adaptive) ℓ_1-tester $T(\epsilon, c, s, q)$ for $L(C_{a,b})$ has query complexity $q = \Omega(n)$.*

The above theorem follows as an immediate corollary of Theorem 5.2.2 and Theorem 3.7 of [86].

5.2.2 Tolerant Testing Code Formulas

We also obtain upper bounds for tolerantly testing code formula lattices.

Theorem 5.2.7 *Let $0 < \epsilon_1, \epsilon_2, c, s < 1$ and $C_0 \subseteq C_1 \subseteq \cdots \subseteq C_{m-1} \subseteq \{0,1\}^n$ be a family of binary linear codes satisfying the Schur product condition. Suppose every C_i has a tolerant tester $T_i(2\epsilon_1, \frac{\epsilon_2}{m2^{i+1}}, \frac{c}{m+1}, s, q_i)$. Let $\gamma = \min\{c/(m+1), s\}$, $\epsilon_2 > m2^{m+1}\epsilon_1$. Then there exists an ℓ_1-tolerant-tester $T(\epsilon_1, \epsilon_2, c, s, q)$ for the lattice $L(\langle C_i \rangle_{i=0}^{m-1})$ with query complexity*

$$q = O\left(\frac{1}{(\epsilon_2 - 2\epsilon_1)^2} \log\left(\frac{1}{\gamma}\right)\right) + \sum_{i=0}^{m-1} q_i.$$

Corollary 5.2.8 *Let $0 \leq k_0 < k_1 < \cdots < k_{m-1} < r$ be integers such that the family of Reed-Muller codes $RM(k_0, r) \subseteq RM(k_1, r) \subseteq \cdots \subseteq RM(k_{m-1}, r)$ satisfies the Schur product condition. Let L be the lattice obtained from this family of codes using the code formula construction:*

$$L = RM(k_0, r) + 2RM(k_1, r) + \cdots + 2^{m-1}RM(k_{m-1}, r) + 2^m\mathbb{Z}^{2^r}.$$

Then there exists a ℓ_1-tolerant-tester $T(\epsilon_1, \epsilon_2, 1/3, 1/3, q)$ for L for all $\epsilon_1 \leq \frac{c_1'}{2^{k_{m-1}}}$, $\epsilon_2 \geq \frac{c_2' m}{2^{k_0-1}}$ (for some constants c_1' and c_2') with query complexity $q = O(2^{k_{m-1}} \cdot \log m)$.

5.2.3 A Canonical/Linear Test for Lattices

We show a reduction from any given arbitrary test to a *canonical linear test*, thus suggesting that it is sufficient to design *canonical linear tests* for achieving low query complexity. In order to describe the intuition behind a canonical linear test, we first illustrate how to solve the membership testing problem when all coordinates of the input are known. For a given lattice L, its *dual lattice* is defined as

$$L^\perp := \{u \in span(L) \mid \langle u, v \rangle \in \mathbb{Z}, \text{ for all } v \in L\}.$$

It is easy to verify that $(L^\perp)^\perp = L$. Furthermore, a vector $v \in L$ if and only if for all $u \in L^\perp$, we have $\langle u, v \rangle \in \mathbb{Z}$. Thus, to test membership of t in L in the classical

decision sense, it is sufficient to verify whether t has integer inner products with a set of basis vectors of the dual lattice $L^\perp$. Inspired by this observation, we define a canonical *linear test* for lattices as follows. For a lattice $L \subseteq \mathbb{R}^n$ and $J \subseteq [n]$, let $L_J^\perp := \{x \in L^\perp \mid supp(x) \subseteq J\}$, where $supp(x)$ is the set of non-zero indices of the vector x.

Definition 5.2.9 (Linear Tester) *A* linear tester *for a lattice $L \subseteq \mathbb{Z}^n$ is a probabilistic algorithm which queries a subset $J = \{j_1, \ldots, j_q\} \subseteq [n]$ of coordinates of the input $t \in \mathbb{R}^n$ and accepts t if and only if $\langle t, x\rangle \in \mathbb{Z}$ for all $x \in L_J^\perp$.* [2]

Remark. By definition, the probabilistic choices of a linear tester are only over the set of coordinates to be queried: upon fixing the coordinate queries, the choice of the algorithm to accept or reject is fully determined. Furthermore, a linear tester is 1-sided since if the input t is a lattice vector, then for every dual vector $u \in L^\perp$, the inner product $\langle u, t\rangle$ is integral, and so it will be accepted with probability 1.

We show that non-adaptive linear tests are nearly as powerful as 2-sided adaptive tests for a full-rank lattice. We reduce any (possibly 2-sided, and adaptive) test for a full-rank lattice to a non-adaptive linear test for the same distance parameter ϵ, with a small increase in the query complexity and the soundness error.

Theorem 5.2.10 *Let $L \subseteq \mathbb{Z}^n$ be a lattice with $rank(L) = n$. If there exists an adaptive 2-sided ℓ_p-tester $T(\epsilon, c, s, q)$ with query complexity $q = q_T(\epsilon, c, s)$, then there exists a non-adaptive linear ℓ_p-tester $T'(\epsilon, 0, c + s, q')$ with query complexity $q' = q_T(\epsilon/2, c, s) + O((1/\epsilon^p)\log(1/s))$.*

Furthermore, if we are guaranteed that the inputs are in $\mathbb{Z}^n$, then the query complexity of the test T' above can be improved to be identical to that of T (up to a constant factor in the ϵ parameter). The increase in the query complexity comes from an extra step used to verify the integrality of the input.

[2] Verifying whether $\langle t, x\rangle \in \mathbb{Z}$ for all $x \in L_J^\perp$ can be performed efficiently by checking inner products with a set of basis vectors of the lattice $L_J^\perp$.

Theorem 5.2.10 suggests that, for the purposes of designing a tester with small query complexity, it is sufficient to design a non-adaptive linear tester, i.e., it suffices to only identify the probability distribution for the coordinates that are queried. Moreover, this theorem makes progress towards Question 5.1.3, since it shows that a lower bound on the query complexity of non-adaptive linear tests for a particular lattice implies a lower bound on the query complexity of all tests for that lattice. Thus in order to understand the existence of low query complexity tester for a particular lattice, it is sufficient to examine the existence of low query complexity *non-adaptive linear* tester for that lattice.

We note that Theorem 5.2.10 is the analogue of the result of [86] for linear error-correcting codes. In Section 5.3, we comment on the comparison between our proof and that in [86].

5.2.4 Testing Membership of Inputs Outside the Span of the Lattice

We also observe a stark difference between the membership testing problem for a linear code, and the membership testing problem for a lattice. In the membership testing problem for a linear code $C \subseteq \mathbb{F}^n$ defined over a finite field that is specified by a basis, the input is assumed to be a vector in $\mathbb{F}^n$ and the goal is to verify whether the input lies in the span of the basis. As opposed to codes, for a lattice $L \subseteq \mathbb{R}^n$, the input is an arbitrary real vector, and the goal is to verify whether the input is a member of L, and not to verify whether the input is a member of the span of the lattice. Thus, the inputs to the lattice membership testing problem could lie either in $span(L)$, or outside $span(L)$. Interestingly, for some lattices it is easy to show strong lower bounds on the query complexity if the inputs are allowed to lie outside $span(L)$, thus suggesting that such inputs are hard to test.

Theorem 5.2.11 *Let $L \subseteq \mathbb{Z}^n$ be a lattice of rank k. Let $P \subseteq [n]$ be the support of the vectors in $span(L)^\perp$. Let $0 < \epsilon, c, s < 1$. Every non-adaptive ℓ_p-tester $T(\epsilon, c, s, q)$ for L for inputs in $\mathbb{R}^n$ has query complexity*

$$q = \Omega(|P|).$$

On the other hand, testers for inputs in the $span(L)$ can be lifted to obtain testers for all inputs (including inputs that could possibly lie outside $span(L)$).

Theorem 5.2.12 *Let $L \subseteq \mathbb{Z}^n$ be a lattice of rank k. Let $P \subseteq [n]$ be the support of the vectors in $span(L)^\perp$. Let $0 < \epsilon, c, s < 1$, and suppose L has an ℓ_p-tester $T(\epsilon, c, s, q)$ for inputs $t \in span(L)$. Then L has a tester $T'(2\epsilon, c, s, q')$ for inputs in $\mathbb{R}^n$ with query complexity*

$$q' \leq q + |P|.$$

Theorem 5.2.12 implies that for lattices L of rank at most $n-1$, if the membership testing problem for inputs that lie in $span(L)$ is solvable using a small number of queries and if $span(L)^\perp$ is supported on few coordinates, then the membership testing problem for all inputs (including those that do not lie in $span(L)$) is solvable using a small number of queries.

Knapsack Lattices. Theorem 5.2.11 implies a linear lower bound for non-adaptively testing a well-known family of lattices, known as *knapsack lattices* defined in Section 2.2.1 of Chapter 2. We recall that a knapsack lattice is generated by a set of basis vectors $B = \{b_1, \ldots, b_{n-1}\}, b_i \in \mathbb{R}^n$ that are of the form

$$\begin{aligned} b_1 &= (1, 0, \ldots, 0, a_1) \\ b_2 &= (0, 1, \ldots, 0, a_2) \\ &\vdots \\ b_{n-1} &= (0, 0, \ldots, 1, a_{n-1}) \end{aligned}$$

where $a_1, \ldots, a_{n-1}$ are integers. We denote such a knapsack lattice by $L_{a_1,\ldots,a_{n-1}}$.

Corollary 5.2.13 *Let $a_1, \ldots, a_n$ be integers and $0 < \epsilon, c, s < 1$. Every non-adaptive ℓ_p-tester $T(\epsilon, c, s, q)$ for $L_{a_1,\ldots,a_{n-1}}$ has query complexity*

$$q = \Omega(n).$$

However, knapsack lattices with bounded coefficients are testable with a constant number of queries if the inputs are promised to lie in $span(L)$.

Theorem 5.2.14 *Let $a_1, \ldots, a_{n-1}$ be integers with $M = \max_{i\in[n]} |a_i|^p$ and $0 < \epsilon, s < 1$. There exists a non-adaptive ℓ_p-tester $T(\epsilon, 0, s, q)$ for $L_{a_1,\ldots,a_{n-1}}$ with query complexity $q = O\left(\frac{M}{\epsilon^p} \cdot \log \frac{1}{s}\right)$, if the inputs are guaranteed to lie in span(L).*

Theorem 5.2.14 indicates that the large lower bound suggested by Theorem 5.2.11 could be circumvented for certain lattices if we are promised that the inputs lie in $span(L)$. The assumption that the input lies in $span(L)$ is natural in decoding problems for lattices.

5.3 Proof Overview

5.3.1 Upper and Lower Bounds for Testing General Code Formula Lattices

The constructions of a tester for Theorem 5.2.1 and a tolerant tester for Theorem 5.2.7 follow the natural intuition that in order to test the lattice one can test the underlying codes individually. The proof relies on a triangle inequality that can be derived for such lattices. The application to code-formula lattices constructed from Reed-Muller codes follows from the tight analysis of Reed-Muller code testing from [87], which guarantees constant rejection probability of inputs that are at distance proportional to the minimum distance of the code. We note that the time complexity of the code-formula tester is given by the sum of the run-times of the component code testers. Since the component code testers can be assumed to be linear, and hence efficient, the code-formula lattice tester is also efficient.

While the tester that we construct from code testers for the purposes of proving Theorem 5.2.1 is an adaptive linear test, there is a simple variant that is a non-adaptive linear test with at least as good correctness and soundness. (see Remark 5.6.3 in Section 5.6).

The lower bound (Theorem 5.2.2) relies on the fact that if an input t is far from the code C_k in the code formula construction, then the vector $2^k t$ is far from the lattice . Moreover, if $t \in C_k$ then $2^k t$ belongs to the lattice. Therefore a test for the lattice can be turned into a test for the constituent codes.

5.3.2 From General Tests to Canonical Tests

We briefly outline our reduction for Theorem 5.2.10. Suppose $T(\epsilon, c, s, q)$ is a 2-sided, adaptive tester with query complexity $q = q_T(\epsilon, c, s)$ for a full rank integral lattice L. Such a tester handles all real-valued inputs. We first restrict T to a test that processes only integral inputs in the bounded set $\mathcal{Z}_d = \{0, 1, \ldots, d-1\}$ (for some carefully chosen d), and so the restricted test inherits all the parameters of T. We remark that $\mathcal{Z}_d \subset \mathbb{Z}$ is a subset of integers, and it should not be confused with $\mathbb{Z}_d$, the ring of integers modulo d.

A key ingredient in our reduction is choosing the appropriate value of d in order to enable the same guarantees as that of codes. We choose d such that $d\mathbb{Z}^n \subseteq L$. Such a d always exists [88]. This choice of d allows us to add any vector in $V = L \bmod d$ (embedded in $\mathbb{R}^n$) to any vector $x \in \mathbb{R}^n$ without changing the distance of x to L in any ℓ_p-norm (see Proposition 5.6.2).

Since our inputs are now integral and bounded, any adaptive test can be viewed as a distribution over deterministic tests, which themselves can be viewed as decision trees. This allows us to proceed along the same lines as in the reduction for codes over finite fields of [86].

We exploit the property that adding any vector in V to any vector $x \in \mathbb{R}^n$ does not change the distance to L. In the first step of our reduction we add a random vector

in V to the input and perform a probabilistic *linear* test. The idea is that one can relabel the decision tree of any test according to the decision tree of a linear test, such that the error shifts from the positive (yes) instances to the negative (no) instances (see Lemma 5.6.3). A simple property of lattices used in this reduction is that if the set of queries I and answers a_I do not have a local witness for non-membership in the lattice (in the form of a dual lattice vector v supported on I such that $\langle w_I, v_I \rangle \notin \mathbb{Z}$), then there exists $w \in L$ that extends a_I to the remaining set of coordinates (i.e., $a_I = w_I$).

In the next step we remove the adaptive aspect of the test to obtain a non-adaptive linear test for inputs in $\mathcal{Z}_d^n$ (see Lemma 5.6.4). We obtain this tester by performing the adaptive queries on a randomly chosen vector in V (and not on the input itself) and rejecting/accepting according to whether there exists a local witness for the non-membership of the input queried on the same coordinates.

We then lift this test to a non-adaptive linear test for inputs in $\mathbb{Z}^n$, by simulating the test over $\mathcal{Z}_d^n$ on the same queried coordinates but using the answers obtained after taking modulo d. Owing to the choice of d, this does not change the distance of the input to the lattice (see Lemma 5.6.5).

Finally, we extend this test to a non-adaptive linear test for inputs in $\mathbb{R}^n$ by performing some additional queries to rule out inputs that are not in $\mathbb{Z}^n$. For this, we design a tester for the integer lattice $\mathbb{Z}^n$ with query complexity $O((1/\epsilon^p)\log(1/s))$. This final step of testing integrality increases the overall query complexity to $q_T(\epsilon/2, c, s) + O((1/\epsilon^p)\log(1/s))$ (see Lemma 5.6.6).

5.4 Testing Code-Formula Lattices

5.4.1 Upper Bounds for Code-Formula Lattices

In this section we construct a tester for testing membership in lattices obtained from the code formula construction using a tester for the constituent codes.

Proof of Theorem 5.2.1 Let $L := L(\langle C_i \rangle_{i=0}^{m-1})$. First, we use Lemma 5.6.6 to reduce the task to testing integral inputs for distance parameter $\epsilon/2$. According to the lemma, it suffices to show that all such inputs can be tested with 1-sided error and using $\sum_{i=1}^{m-1} q_i$ queries.

Now, let $w \in \mathbb{Z}^n$ denote the input. We may assume that all coordinates of w are non-negative integers less than 2^m. Otherwise, we can shift each coordinate by an appropriate (possibly different) multiple of 2^m to make sure this condition holds. Observe that each such operation would correspond to shifting w by an integer multiple of the lattice vector $2^m e_i$, where e_i is the i^{th} basis vector, and that translating a vector by a lattice point does not affect its distance to the lattice. Moreover, observe that this transformation can be applied implicitly, locally, and efficiently by the testing algorithm as the queries are made.

Let $w_0, \ldots, w_{m-1} \in \{0,1\}^n$ where $w_i(j)$ is the $(i+1)^{th}$ least significant bit in the binary decomposition of the j^{th} coordinate of w. Thus we have $w = \sum_{i=0}^{m-1} 2^i w_i$. Once again, the coordinates of w_i can be computed implicitly, locally and efficiently by the algorithm as the queries are made.

The tester T would now proceed as follows: Run $T_i(\epsilon/(m2^{i+1}), 0, s, q_i)$ on w_i for every $i = 0, 1, \ldots, m-1$. Accept if and only if all tests accept.

The overall query complexity of this tester is $\sum_{i=1}^{m-1} q_i$.

The completeness of this tester is easy to deduce. Indeed, if $w \in L(\langle C_i \rangle_{i=0}^{m-1})$, then by definition of the code formula construction, there exist $\tilde{w}_i \in C_i$ for every $i = 0, 1, \ldots, m-1$ and an integer $\tilde{w}_m \in 2^m\mathbb{Z}^n$ such that $w = \sum_{i=0}^{m-1} 2^i \tilde{w}_i + \tilde{w}_m$. Since entries of w are non-negative integers less than 2^m, we must have $\tilde{w}_m = 0$. Moreover, since $\tilde{w}_i \in \{0,1\}^n$ and the binary representation is unique, it must be that $\tilde{w}_i \in C_i$ for $i = 0, \ldots, m-1$. That is, each of the w_i embedded in $\mathbb{F}_2^n$ are in C_i and therefore, each $T_i(\epsilon/(2^{i+1}m), 0, s, q_i)$ (and thus the overall tester) will accept w_i.

Before analyzing the soundness, we observe the following simple inequality.

Claim 5.4.1 $d_1(w, L) \leq d_1(w_0, C_0) + 2d_1(w_1, C_1) + \cdots + 2^{m-1} d_1(w_{m-1}, C_{m-1})$.

Proof Let $c_i \in \{0,1\}^n$ be the closest codeword to w_i in C_i for every $i = 0, 1, \ldots, m-1$. From the definition of the code formula construction, we know that the vector $v = c_0 + 2c_1 + \cdots + 2^{m-1}c_{m-1}$ is a lattice vector. Therefore,

$$\begin{aligned}\sum_{i=0}^{m-1} 2^i d_1(w_i, C_i) &= \sum_{i=0}^{m-1} 2^i \|w_i - c_i\|_1 \\ &\geq \left\| \sum_{i=0}^{m-1} 2^i (w_i - c_i) \right\|_1 \qquad \text{(by the triangle inequality)} \\ &= d_1(w, v) \geq d_1(w, L).\end{aligned}$$

■

Now, if $d_1(w, L) \geq \epsilon n/2$, then by Claim 5.4.1 we have

$$d_1(w_0, C_0) + 2d_1(w_1, C_1) + \cdots + 2^{m-1} d_1(w_{m-1}, C_{m-1}) \geq \epsilon n/2.$$

Therefore, by averaging, for some $i \in \{0, 1, \ldots, m-1\}$ we must have $d_1(w_i, C_i) \geq (\epsilon/(m2^{i+1}))n$. Thus the tester $T_i(\epsilon/(m2^{i+1}), 0, s, q_i)$ will reject with probability at least $1 - s$. Hence the soundness follows. ■

We now apply the result of Theorem 5.2.1 to the lattice obtained by applying code formula on a nested family of Reed-Muller codes. In order to do so, we use the following result.

Theorem 5.4.2 *[87] For any $0 \leq k \leq r$ and $0 < s < 1$, $RM(k, r)$ has a 1-sided tester $T(\epsilon, 0, s, q(\epsilon, s))$ with query complexity $q(\epsilon, s) = O((\log \frac{1}{s})(2^k + \frac{1}{\epsilon}))$ whose queries are each uniformly distributed.*

Using the above result in Theorem 5.2.1, we obtain Corollary 5.2.4 whose proof we present next.

Proof of Corollary 5.2.4 From Theorem 5.2.1, for any $\epsilon > 0$, there exists a $T(\epsilon, 0, s, q(\epsilon, s))$ tester for the code formula lattice $L(\langle C_i \rangle_{i=0}^{m-1})$ with query complexity $q(\epsilon, s) = O\left(\frac{1}{\epsilon} \log \frac{1}{s}\right) + \sum_{i=0}^{m-1} q_i\left(\frac{\epsilon}{m2^{i+1}}, s\right)$, where $q_i(\epsilon, s)$ is the query complexity of testing the code C_i (with distance parameter ϵ and soundness error s). By Theorem

5.4.2, each $RM(k_i, r)$ has a 1-sided tester $T_i(\epsilon_i, 0, s, q_i(\epsilon_i, s))$ with query complexity $q_i(\epsilon_i, s) = O(\log(1/s))(2^{k_i} + 1/\epsilon_i)$. Therefore, the number of queries made by a 1-sided tester $T(\epsilon, 0, s, q)$ for the code formula lattice $L(\langle RM(i, r)\rangle_{i=k_0}^{k_{m-1}})$ is given by

$$\begin{aligned} q(\epsilon, s) &= O\left(\frac{1}{\epsilon}\log\frac{1}{s}\right) + \sum_{i=0}^{m-1} q_i\left(\frac{\epsilon}{m2^{i+1}}, s\right) \\ &= O\left(\frac{1}{\epsilon}\log\frac{1}{s}\right)\sum_{i=0}^{m-1}\left(2^{k_i} + m2^{i+1}\right) \\ &= O\left(\frac{1}{\epsilon}\log\frac{1}{s}\right)\left(m2^m + \sum_{i=0}^{m-1} 2^{k_i}\right) \\ &= O\left(\frac{1}{\epsilon}\log\frac{1}{s}\right)\left(m2^m + 2^{k_{m-1}}\right) \\ &= O\left(\frac{1}{\epsilon}\log\frac{1}{s}\cdot 2^{k_{m-1}}\right). \end{aligned}$$

To see the last step of the above equation, recall that in order for $L(\langle RM(i, r)\rangle_{i=k_0}^{k_{m-1}})$ to be a lattice, we must have $k_i \geq 2^{i-1}$ for $i > 0$, and in particular $k_{m-1} \geq 2^{m-2}$. So $2^{k_{m-1}} \geq 2^{2^{m-2}} \geq m2^m$, for $m \geq 5$. ■

5.4.2 Lower Bounds for Code-Formula Lattices

In this section, we prove Theorem 5.2.2. We will use the following lemma.

Lemma 5.4.3 *Let $C_0, C_1, \ldots, C_{m-1}$ be a family of codes satisfying the Schur product condition and $L = L(\langle C_i\rangle_{i=0}^{m-1})$. Let $t \in \{0,1\}^n$ and $k \in \{0, 1, \ldots, m-1\}$. Then*

$$d_1(t, C_k) \leq d_1(2^k t, L) \leq 2^k d_1(t, C_k).$$

Proof Since $2^k C_k$ is contained in L, $d_1(2^k t, L) \leq d_1(2^k t, 2^k C_k) = 2^k d_1(t, C_k)$. So, the distance of $2^k t$ to the lattice is at most $2^k d_1(t, C_k)$. We now show the inequality

$$d_1(2^k t, L) \geq d_1(t, C_k).$$

Let $v = \sum_{j=0}^{m-1} 2^j c_j + 2^m z$ for some arbitrary $c_j \in C_j$ (for every $j \in \{0, 1, \ldots, m-1\}$) and some $z \in \mathbb{Z}^n$, be an arbitrary lattice vector. We will show that $d_1(2^k t, v) \geq d_1(t, C_k)$. Let $u = c_k - t$, and $S \subseteq [n]$ be the support of u, then $|S| \geq d_1(t, C_k)$.

By Claim 5.4.4, $d_1(2^k t, v) \geq \sum_{i \in S} |v(i) - 2^k t(i)| \geq |S|$ (where $v(i)$ and $t(i)$ represent the ith entry in the respective vectors), which completes the proof. ■

Claim 5.4.4 *For every $i \in S$, $|v(i) - 2^k t(i)| \geq 1$.*

Proof Let $i \in S$. Since $2^k u = 2^k c_k - 2^k t$, we have $2^k |u(i)| = 2^k$. We also have

$$|v(i) - 2^k t(i)| = \left| \sum_{j=0}^{k-1} 2^j c_j(i) + 2^k u(i) + \sum_{j=k+1}^{m-1} 2^j c_j(i) + 2^m z(i) \right|.$$

Since $c_j(i) \in \{0,1\}$ for every $j \in \{0,1,\ldots,k-1\}$, the first term in the above sum is at least zero and at most $2^k - 1$. The maximum is achieved when all $c_j(i) = 1$ for all $j \in [k-1]$, and $u(i) = 1$. Hence, $1 \leq \left| \sum_{j=0}^{k-1} 2^j c_j(i) + 2^k u(i) \right| \leq 2^{k+1} - 1$. Since $c_{k+1}(i) \in \{0,1\}$, we have

$$\begin{aligned} 1 \leq \left| \sum_{j=0}^{k-1} 2^j c_j(i) + 2^k u(i) + 2^{k+1} c_{k+1}(i) \right| &\leq \left| \sum_{j=0}^{k-1} 2^j c_j(i) + 2^k u(i) \right| + \left| 2^{k+1} c_{k+1}(i) \right| \\ &\leq 2^{k+2} - 1. \end{aligned}$$

Proceeding similarly, since $c_j(i) \in \{0,1\}$ for $j = k+2, \ldots, m-1$, we have

$$1 \leq \left| \sum_{j=0}^{k-1} 2^j c_j(i) + 2^k u(i) + \sum_{j=k+1}^{m-1} 2^j c_j(i) \right| \leq 2^m - 1.$$

Since $z_m \in \mathbb{Z}$, we get, $\left| \sum_{j=0}^{k-1} 2^j c_j(i) + 2^k u(i) + \sum_{j=k+1}^{m-1} 2^j c_j(i) + 2^m z(i) \right| \geq 1$. ■

Proof of Theorem 5.2.2 Let $T(\epsilon, c, s, q)$ be a test for the code lattice, and let $k \in \{0,1,\ldots,m-1\}$. We construct a tester $T_k(\epsilon, c, s, q)$ for C_k as follows: On input $w \in \{0,1\}^n$, run $T(\epsilon, c, s, q)$ on $2^k w$ and accept if and only if T accepts. The query complexity of T_k is the same as the query complexity of T. If the input w is a codeword in C_k, then by the definition of the lattice, $2^k w$ is a lattice vector, and T_k will accept w with probability at least $1 - c$. If $d_1(w, C_k) \geq \epsilon n$, then by Lemma 5.4.3, we have that $d_1(2^k w, L(\langle C_i \rangle)_{i=0}^{m-1}) \geq \epsilon n$. Therefore, T_k will reject w with probability at least $1 - s$. Finally, since $L \subseteq \mathbb{Z}^n$ we have that $d(w, \mathbb{Z}^n) \leq d(w, L)$ and so we could use T to test membership in $\mathbb{Z}^n$. By Claim 5.4.5 testing $\mathbb{Z}^n$ requires $q = \Omega(\frac{1}{\epsilon} \log(1/s))$ queries. ■

Claim 5.4.5 *Any test $T_k(\epsilon, c, s, q)$ for $\mathbb{Z}^n$ requires $q = \Omega(\frac{1}{\epsilon}\log(1/s))$ queries.*

Proof First, we use Yao's duality theorem [89] which is a standard tool in proving lower bounds and assume that the testing algorithm is, without loss of generality, deterministic (but possibly adaptive). We exhibit two distributions on the inputs which the algorithm is expected to distinguish but cannot do so without making sufficiently many queries. The *yes case* distribution is the deterministic distribution on all-zeros input (which is a lattice point). Given an input from this distribution, the algorithm should accept. The *no case* distribution would consists of the all-zeros vector but with a uniformly random set S of $2\epsilon n$ coordinate positions changed from 0 to 1/2. Indeed, all vectors on the support of this distribution are ϵ-far from the lattice. Assuming that $q \leq n/2$ (otherwise there is nothing to prove), each time the algorithm queries a position that has not been queries before, there is at most a 4ϵ chance that it hits any position in S (even conditioned on the past query outcomes). Thus the probability that the algorithm ever succeeds in finding a position in S is at most $1-(1-4\epsilon)^q$, which, on the other hand by the soundness condition, should be at least $1-s$. Therefore, the soundness error is at least $s \geq (1-4\epsilon)^q$ or, in other words, in order to achieve a given s we must have $q = \Omega(\frac{1}{\epsilon}\log(\frac{1}{s}))$. ■

In the case of code formula lattices generated from Reed-Muller codes of order r, we note that $n = 2^r$. We need the following known lower bound on the query complexity of testing Reed-Muller codes. For completeness we reproduce the exact statement that we need in this work and include a proof.

Theorem 5.4.6 ([90]) *Let $T(\epsilon, c, s, q)$ be a (possibly 2-sided and adaptive) tester for the code $RM(k, r)$ where $k \leq r/(2\log r)$, $\epsilon < 1/2 - \Omega(1)$, and $c + s < 1 - \Omega(1)$ (where $\Omega(1)$ hides arbitrarily small positive absolute constants). Then, $q \geq 2^k$.*

Proof Using the reduction from 2-sided, adaptive testers to non-adaptive, 1-sided tests for any linear code (applicable to RM codes) of [86], it is sufficient to focus on

the latter tests. Let $\mathcal{C}$ be the code $RM(k, r)$. First we note that the length of the code is $R := 2^r$ and its dimension is

$$\log |\mathcal{C}| = \sum_{i=0}^{k} \binom{r}{i} \leq 1 + kr^k.$$

Therefore, noting that $k \leq r/(2 \log r)$,

$$|\mathcal{C}| = O(2^{kr^k}) = O(2^{r^{k+1}}) = O(2^{(\log R)\sqrt{R}}) = 2^{o(R)}.$$

Let V be the number of points in a Hamming ball of radius ϵR in $\{0,1\}^R$. Thus we have $V \leq 2^{h(\epsilon)R}$, where $h(\cdot)$ denotes the binary entropy function. Let S be the set of points in $\{0,1\}^R$ that have Hamming distance at most ϵr with the code. Of course we have

$$|S| \leq V|\mathcal{C}| = O(2^{(h(\epsilon)+o(1))R}).$$

Since $\epsilon < 1$ is a fixed constant, this implies $|S|/2^R = o(1)$. Now we run the tester with the following two input distributions:

Case 1. The tester is given a uniformly random string in $\{0,1\}^R$ as the input.

Case 2. The tester is given a uniformly random codeword of $\mathcal{C}$ as the input.

Since the dual distance of $\mathcal{C}$ is 2^{k+1}, a standard coding theoretic fact implies that a uniformly random codeword of $\mathcal{C}$ is t-wise independent for $t = 2^{k+1}$; i.e., any local view of up to t coordinates of the random codeword is exactly the uniform distribution. Therefore, since the tester makes no more than t queries, its output distribution is exactly the same in the above two cases. Let p be the acceptance probability of the tester with respect to the (common) output distribution. In order to satisfy completeness, the tester should accept with probability at least $1 - c$ in the second case. Therefore, we must have $p \geq 1 - c$.

On the other hand, a uniform random string in $\{0,1\}^R$ is ϵ-far from the code with probability $1 - o(1)$ according to the above bound on $|S|$. In the conditional world where this string actually becomes ϵ-far from the code, the acceptance probability of the code would thus remain within $p(1 \pm o(1))$. However, in this case the soundness

implies that the tester should accept with probability at most s, and thus, $p \leq s(1+o(1))$. Thus the two distributions provided by the above two cases would violate requirements of the local tester assuming that $c + s \leq 1 - \Omega(1)$. ■

The proof of Corollary 5.2.5 then follows from this Claim.

Proof of Corollary 5.2.5 Let $T(\epsilon, c, s, q)$ be a tester for $L(\langle RM(k_i, r)\rangle_{i=0}^{m-1})$. By Theorem 5.2.2, we have

$$q \geq \max_{i=0,1,\ldots,m-1} q_i(\epsilon, c, s).$$

By Theorem 5.4.6, it follows that $q = \Omega(2^{k_{m-1}})$. ■

5.5 Tolerant Testing Code Formula Lattices

We first give a tolerant tester for testing membership in $\mathbb{Z}^n$. We will use this tester in the design of a tolerant tester for testing membership in lattices obtained from the code formula construction.

Lemma 5.5.1 *Let $\epsilon_1, \epsilon_2, c, s > 0$ such that $\epsilon_2 > \epsilon_1$ and $\gamma = \min\{c, s\}$. There exists a tolerant tester $T_Z(\epsilon_1, \epsilon_2, c, s, q_Z)$ for $\mathbb{Z}^n$ with query complexity*

$$q_Z = O(1/(\epsilon_2 - \epsilon_1)^2 \cdot \log(\frac{1}{\gamma})).$$

Proof The tester estimates the distance of the input from $\mathbb{Z}^n$ by querying $O(1/(\epsilon_2 - \epsilon_1)^2 \log(\frac{1}{\gamma}))$ coordinates uniformly at random. If the estimated distance is at least $\frac{(\epsilon_1+\epsilon_2)}{2}n$, then it rejects, otherwise it accepts. The correctness and soundness follow from Chernoff bounds. We describe the test formally as follows:

1. Query $q := C/(\epsilon_2 - \epsilon_1)^2 \cdot \log(\frac{1}{\gamma})$ coordinates of the input t uniformly at random, for some constant C to be determined later. Let $I \subseteq [n]$ be the indices of the queried coordinates.
2. Let $\delta := \frac{\sum_{i \in I} |t_i - \lfloor t_i \rceil|}{q}$.
3. If $\delta \leq \frac{\epsilon_1+\epsilon_2}{2}$ then Accept.

4. Else Reject.

Suppose $d(t, \mathbb{Z}^n) \leq \epsilon_1 n$, then $d(t, \mathbb{Z}^n)/n = \sum_{i=1}^{n} |t_i - \lfloor t_i \rceil|/n \leq \epsilon_1$. Therefore, $\mathbb{E}[\delta] \leq \epsilon_1$. By a Chernoff bound, it follows that $\Pr[\delta - \epsilon_1 > \frac{\epsilon_2 - \epsilon_1}{2}] \leq e^{-q(\epsilon_2-\epsilon_1)^2/2} \leq c$ for $q \geq C/(\epsilon_2 - \epsilon_1)^2 \cdot \log(\frac{1}{\gamma})$ and a constant $C > 0$.

Now suppose $d(t, \mathbb{Z}^n) > \epsilon_2 n$. Then, $d(t, \mathbb{Z}^n)/n = \sum_{i=1}^{n} |t_i - \lfloor t_i \rceil|/n \geq \epsilon_2$. Again, by a Chernoff bound, and suitable choice of the constant C, it follows that $\Pr[\epsilon_2 - \delta \geq \frac{\epsilon_2 - \epsilon_1}{2}] \leq e^{-q(\epsilon_2-\epsilon_1)^2/4} \leq s$ for q chosen as above.

■

We now describe a tolerant tester for code formula lattices.

Proof of Theorem 5.2.7 We use the tolerant testers T_i for the codes C_i and the tolerant tester T_Z for $\mathbb{Z}^n$ to construct a tolerant tester for L.

Let $\lfloor t \rceil$ denote the vector obtained by rounding each coordinate of t to its nearest integer and for any vector x, let $x(j)$ denote the j^{th} coordinate of x. Let $t_0, ..., t_{m-1} \in \{0,1\}^n$ where $t_i(j)$ is the $(i+1)^{th}$ least significant bit in the binary decomposition of the j^{th} coordinate of $\lfloor t \rceil$. Define $t_m = \frac{1}{2^m}(t - \sum_{i=0}^{m-1} 2^i t_i) \in \mathbb{R}^n$. Therefore, t can be written as $t = t_0 + 2t_1 + \cdots + 2^{m-1}t_{m-1} + 2^m t_m$, where $t_i \in \{0,1\}^n$ for all $i \in [m-1]$ and $t_m \in \mathbb{R}^n$. Moreover, $t_m \in \mathbb{Z}^n$ if and only if $t \in \mathbb{Z}^n$.

The tolerant tester $T(\epsilon_1, \epsilon_2, c, s, q)$ on input $t \in \mathbb{R}^n$ now proceeds as follows: Run $T_Z(\epsilon_1, \frac{\epsilon_2}{2}, \frac{c}{m+1}, s, q_Z)$ on t and $T_i(2\epsilon_1, \frac{\epsilon_2}{m2^{i+1}}, \frac{c}{m+1}, s, q_i)$ on t_i for all $i \in \{0, 1, \ldots, m-1\}$. Accept if and only if all tests accept. The query complexity of $T(\epsilon_1, \epsilon_2, c, s, q)$ is therefore:

$$q(\epsilon_1, \epsilon_2, c, s) = \sum_{i=0}^{m-1} q_i + q_Z,$$

where we recall that q_Z is the query complexity of $T_Z(\epsilon_1, \frac{\epsilon_2}{2}, \frac{c}{m+1}, s, q_Z)$. We know from Lemma 5.5.1, that $q_Z = O(\frac{1}{(\epsilon_2 - 2\epsilon_1)^2} \log(\frac{1}{\gamma}))$, where $\gamma = \min\{\frac{c}{m+1}, s\}$. We now analyze the soundness and completeness of this test.

Soundness: Suppose $d(t, L) \geq \epsilon_2 n$. We first show that either t is far from $\mathbb{Z}^n$ or the closest integer vector to t is far from the lattice.

Claim 5.5.2 $d(t, L) \leq d(\lfloor t \rceil, L) + d(t, \mathbb{Z}^n)$

Proof Let u be the closest lattice vector to $\lfloor t \rceil$. Then

$$d(t, L) \leq \|t - u\|_1 = \|(t - \lfloor t \rceil) + (\lfloor t \rceil - u)\|_1 \leq \|t - \lfloor t \rceil\|_1 + \|\lfloor t \rceil - u\|_1$$

Since $\|t - \lfloor t \rceil\|_1 = d(t, \mathbb{Z}^n)$, it follows that $d(t, L) \leq d(\lfloor t \rceil, L) + d(t, \mathbb{Z}^n)$. ■

Therefore, if $d(t, L) \geq \epsilon_2 n$, then from Claim 5.5.2, either $d(\lfloor t \rceil, L) \geq \epsilon_2 n/2$ or $d(t, \mathbb{Z}^n) \geq \epsilon_2 n/2$. If $d(t, \mathbb{Z}^n) \geq \epsilon_2 n/2$, then T_Z rejects t with probability at least $1 - s$. If $d(\lfloor t \rceil, L) \geq \epsilon_2 n/2$, then from Claim 5.4.1 proved in Section 5.4 we can conclude that there exists some $i \in \{0, 1, \ldots, m-1\}$ such that $2^i d(t_i, C_i) \geq \epsilon_2 n/2m$, and $T_i(t_i, 2\epsilon_1, \epsilon_2/m2^{i+1})$ will reject t_i with probability at least $1 - s$. Thus, if $d(t, L) \geq \epsilon_2 n$, then T rejects t with probability with at least $1 - s$.

Completeness: Suppose $d(t, L) \leq \epsilon_1 n$. Then $d(t, \mathbb{Z}^n) \leq \epsilon_1 n$, since $L \subseteq \mathbb{Z}^n$. So, $T_Z(\epsilon_1, \frac{\epsilon_2}{2}, \frac{c}{m+1}, s, q_Z)$ will accept t with probability at least $1 - \frac{c}{m+1}$. We now show that each t_i is also close to the corresponding linear code C_i.

For the sake of contradiction, suppose $d(t_i, C_i) > 2\epsilon_1 n$ for some $i \in [m-1]$. We will show that $d(t, L) > \epsilon_1 n$. We do this in two steps. First we show in Lemma 5.5.3 that $d(\lfloor t \rceil, L) > 2\epsilon_1 n$. Then by Claim 5.5.5 and the fact that $d(t, \mathbb{Z}^n) \leq \epsilon_1 n$, we have that $d(t, L) > \epsilon_1 n$, a contradiction.

Lemma 5.5.3 *If $d(t_i, C_i) > 2\epsilon_1 n$ for some $i \in [m-1]$, then $d(\lfloor t \rceil, L) > 2\epsilon_1 n$.*

Proof Let $v = \sum_{i=0}^{m-1} 2^i v_i + 2^m v_m \in L$ be the closest lattice vector to $\lfloor t \rceil$. By definition of the lattice, each $v_i \in C_i$ for $i \in [m-1]$ and $v_m \in \mathbb{Z}^n$. Consider the vectors $t_0, t_1, \ldots, t_m$ as defined above (for which $\lfloor t \rceil = \sum_{i=0}^{m-1} 2^i t_i + 2^m t_m$). So, each $t_i \in \{0, 1\}^n$ and $t_m \in \mathbb{Z}^n$. The following property of vectors with bounded entries will be used to prove the claim.

Claim 5.5.4 *Let $a_0, a_1, \ldots, a_{m-1} \in \{-1, 0, +1\}^n$ and $a_m \in \mathbb{Z}$. Define $u = a_0 + 2a_1 + \cdots + 2^m a_m$. If there exists some $k \in [m-1]$ such that $\|a_k\|_1 > s$, then $\|u\|_1 > s$.*

Proof Since $\|a_k\|_1 > s$, and $a_k \in \{-1, 0, +1\}^n$, there exist at least s coordinates such that $|a_k(i)| = 1$. Let S be the set of those indices, $S = \{i \in [n] \colon |a_k(i)| = 1\}$. Since $\|a_k\|_1 > s$, we know that $|S| > s$. We now show that for all $i \in S, |u(i)| \geq 1$. Therefore, $\|u\|_1 > s$.

Let $i \in S$. For each such coordinate, we can express $u(i) = \sum_{j=0}^{m} 2^j a_j(i)$. Let $h \in [k]$ be the smallest integer such that $a_h(i) \neq 0$. We know that such h exists since $a_k(i) \neq 0$. Therefore, we know that $u(i) \bmod 2^{h+1} (= a_h)$, is non-zero. Therefore, $u(i)$ is also non-zero. Since $u(i) \in \mathbb{Z}$, we have that $|u(i)| \geq 1$.

Therefore, $|u(i)| \geq 1$ for all $i \in S$. and $\|u\|_1 \geq |S| > s$. ■

Define $a_i = (t_i - v_i)$ for all $i \in [m]$. We note that each $a_i \in \{-1, 0, +1\}^n$ for $i \in [m-1]$ and that $a_m \in \mathbb{Z}^n$. The proof now follows from Claim 5.5.4 for $s = 2\epsilon_1 n$. ■

The next claim is a straightforward application of the triangle inequality.

Claim 5.5.5 $d(t, L) \geq d(\lfloor t \rceil, L) - d(t, \mathbb{Z}^n)$

Proof Let u be the closest lattice vector to t.

$$d(\lfloor t \rceil, L) \leq \|\lfloor t \rceil - u\|_1 = \|\lfloor t \rceil - t + t - u\|_1.$$

By the triangle inequality, we have $\|\lfloor t \rceil - t + t - u\|_1 \leq \|\lfloor t \rceil - t\|_1 + \|t - u\|_1$. Since u is the closest lattice vector to t, $d(t, L) = \|t - u\|_1$. Also, $\|\lfloor t \rceil - t\| = d(t, \mathbb{Z}^n)$, Therefore, $d(\lfloor t \rceil, L) \leq d(t, L) + d(t, \mathbb{Z}^n)$. ■

Thus, if $d(t, L) \leq \epsilon_1 n$, then T_Z accepts t with probability at least $1 - \frac{c}{m+1}$ and each code tester T_i accepts t_i with probability at least $1 - \frac{c}{m+1}$. Therefore, from the union bound, T accepts t with probability at least $1 - \sum_{i=0}^{m} \frac{c}{m+1} = 1 - c$. ■

We next instantiate Theorem 5.2.7 for code-formula lattices obtained from Reed-Muller codes. We first recall a simple observation made in [45] that any local test with individual queries uniformly distributed is also a tolerant test.

Claim 5.5.6 ([45]) *If a length n binary code $C \subseteq \{0,1\}^n$ has a one-sided local test $T(\epsilon, 0, 1/3, q)$ whose queries are each uniformly distributed, then C has a tolerant local test $T(\epsilon_1, \epsilon_2, 1/3, 1/3, q)$, with $\epsilon_1 \leq \frac{1}{3q}$ and $\epsilon_2 \geq \epsilon$.*

Using Claim 5.5.6 and Theorem 5.4.2, and by appropriately amplifying the success probability, we get a tolerant test for Reed-Muller codes.

Corollary 5.5.7 *For any $k, r, c, s > 0$ and $\gamma = \min\{c, s\}$, there exists a tolerant test $T(\epsilon_1, \epsilon_2, c, s, q)$ for $RM(k, r)$ such that $\epsilon_1 \leq c_1 \frac{1}{2^k}$, $\epsilon_2 \geq c_2 \frac{1}{2^k}$ and $q = O(2^k \log(\frac{1}{\gamma}))$, for some $c_1, c_2 > 0$.*

Proof By Theorem 5.4.2 we know that there is a 1-sided tester $T(\epsilon, 0, 1/3, q)$ for $RM(k, r)$ and $\epsilon = O(1/2^k)$ with query complexity $q = O(2^k)$. From Claim 5.5.6, we know that we can obtain a tolerant tester $T(\epsilon_1, \epsilon_2, 1/3, 1/3, q)$ with $O(2^k)$ queries for any $\epsilon_1 \leq c_1/2^k$ and $\epsilon_2 \geq c_2/2^k$. By independently repeating the tester multiple times and taking majority vote to amplify the success probability, for any $0 < c, s \leq 1$ and $\gamma = \min\{c, s\}$ we get a tolerant tester $T(\epsilon_1, \epsilon_2, c, s, q)$ for $RM(n, k)$ with $q = O(2^k \log(\frac{1}{\gamma}))$ queries. ■

Using Corollary 5.5.7 and Theorem 5.2.7 we obtain the following corollary.

Proof of Corollary 5.2.8 From Corollary 5.5.7 we know that every RM_{k_i} has a tolerant local tester T_i $(2\epsilon_1, \frac{\epsilon_2}{m2^{i+1}}, \frac{1}{3(m+1)}, \frac{1}{3}, q_i)$ with query complexity $q_i = O(2^{k_i} \log(m+1))$ for $2\epsilon_1 \leq \frac{c_1}{2^{k_i}}$ and $\frac{\epsilon_2}{m2^{i+1}} \geq \frac{c_2}{2^{k_i}}$ for some constants $c_1, c_2 > 0$.

Using Theorem 5.2.7, we conclude that L has a tolerant tester $T(\epsilon_1, \epsilon_2, \frac{1}{3}, \frac{1}{3}, q)$ with query complexity $q = O(\frac{1}{(\epsilon_2 - 2\epsilon_1)^2} \log(m+1)) + \sum_{i=0}^{m-1} O(2^{k_i} \log(m+1)) = O(2^{k_{m-1}} \log m)$ for $2\epsilon_1 \leq \min_i\{\frac{c_1}{2^{k_i}}\}$ and $\frac{\epsilon_2}{m2^{i+1}} \geq \max_i\{\frac{c_2}{2^{k_i}}\}$. ■

5.6 Canonical Linear Test

In this section we sketch the proof of Theorem 5.2.10. Throughout this section, we focus on full-rank integral lattices. Given a 2-sided adaptive ℓ_p-tester $T(\epsilon, c, s, q)$, with

$q = q_T(\epsilon, c, s)$ for an integral lattice L, we construct a non-adaptive linear ℓ_p-tester $T'(\epsilon, 0, c+s, q)$ with query complexity $q' = q_T(\epsilon/2, c, s) + O((1/\epsilon^p)\log(1/s))$. We reduce the inputs to a bounded set using the following property of integral lattices.

Fact 5.6.1 *[88] Given any full rank integral lattice L, there exists $d \in \mathbb{Z}$ such that $d \cdot \mathbb{Z}^n \subseteq L$. In particular $|\det(L)| \cdot \mathbb{Z}^n \subseteq L$ for any lattice (where $\det(L)$ denotes the determinant of a lattice, a parameter that can be computed given a basis of the lattice). For instance, we can take $d = 2^m$ for the lattices of height m obtained using the code formula construction.*

Let $V = L \bmod d$ embedded in $\mathbb{Z}^n$ (i.e., we treat V as a set of vectors in $\mathbb{Z}^n$ each of which is obtained by taking coordinate-wise modulo d of some lattice vector). Thus, $V \subseteq \mathcal{Z}_d^n$. We will need the following properties of V.

Proposition 5.6.2 *Let $L \subseteq \mathbb{Z}^n$ be a full-rank lattice, $d \in \mathbb{Z}_+$ such that $d\mathbb{Z}^n \subseteq L$, and let $V = L \bmod d \subseteq \mathbb{Z}^n$. Then V satisfies the following properties:*

1. *$v \in L$ if and only if $v \bmod d \in V$.*
2. *$V = L \cap \mathcal{Z}_d^n$.*
3. *$(v + V) \bmod d \subseteq V$ if and only if $v \in L$.*
4. *For any $v \in \mathbb{Z}^n$, $d_p(v, L) = d_p(v \bmod d, L)$.*

Proof

1. If $v \in L$, then $v \bmod d \in V$ by definition. For the opposite direction, let $v \in \mathbb{Z}^n$ be such that $u = v \bmod d \in V$. Then by the definition of V there exists $v' \in L$ such that $v' = u = v \pmod d$. Then $v - v' \in d\mathbb{Z}^n \subseteq L$, and so $v \in L$.

2. By definition $L \cap \mathcal{Z}_d^n \subseteq V$. To show that $V \subseteq L$ note that by 1), if $v \in V$ there exists $v' \in L$ such that $v' = v \bmod d$. As before, this implies that $v \in L$.

3. This statement follows by the fact that $V \subseteq L$ and from the fact that lattices are closed under addition.

4. Note that $d_p(v, L) = \min_{u \in L} d_p(u, v) = \min_{u \in L} \|v - u\|_p$. If $v = dv_1 + v_2$, since $dv_1 \in d\mathbb{Z}^n \subseteq L$, it follows that $\min_{u \in L} \|v - u\|_p = \min_{u \in L} \|v_2 - u\|_p$, since a lattice is closed under addition.

■

Theorem 5.2.10 will immediately follow by combining Lemmas 5.6.3, 5.6.4, 5.6.5, and 5.6.6. We now state the lemmas and prove them in the subsequent subsections.

Lemma 5.6.3 *Suppose a full-rank lattice $L \subseteq \mathbb{Z}^n$ with $d\mathbb{Z}^n \subseteq L$ for $d \in \mathbb{Z}_+$ has an adaptive 2-sided ℓ_p-tester $T(\epsilon, c, s, q)$ for inputs from the domain $\mathcal{Z}_d^n$. Then L has an adaptive linear ℓ_p-tester $T'(\epsilon, 0, c + s, q)$ for inputs from the domain $\mathcal{Z}_d^n$.*

Lemma 5.6.4 *Suppose a full-rank lattice $L \subseteq \mathbb{Z}^n$ with $d\mathbb{Z}^n \subseteq L$ for $d \in \mathbb{Z}_+$ has an adaptive linear ℓ_p-tester $T(\epsilon, 0, s, q)$ for inputs from the domain $\mathcal{Z}_d^n$. Then L has a non-adaptive linear ℓ_p-tester $T'(\epsilon, 0, s, q)$ for inputs from the domain $\mathcal{Z}_d^n$.*

Lemma 5.6.5 *Let $L \subseteq \mathbb{Z}^n$ be a full-rank lattice with $d\mathbb{Z}^n \subseteq L$ for $d \in \mathbb{Z}_+$. Then, L has a non-adaptive linear ℓ_p-tester $T(\epsilon, 0, s, q)$ for inputs from the domain $\mathcal{Z}_d^n$ if and only if L has a non-adaptive linear ℓ_p-tester $T'(\epsilon, 0, s, q)$ for inputs from $\mathbb{Z}^n$.*

Lemma 5.6.6 *Let $T(\epsilon, c, s, q)$ be a non-adaptive ℓ_p-tester for a full-rank lattice $L \subseteq \mathbb{Z}^n$ whose inputs come from the domain $\mathbb{Z}^n$. Then there exists a non-adaptive ℓ_p-tester $T'(\epsilon, c, s, q')$ for inputs in $\mathbb{R}^n$ with query complexity $q' = q(\epsilon/2, c, s) + O((1/\epsilon^p) \log(1/s))$. Moreover, if T is a linear tester, then so is T'.*

The proof of Lemma 5.6.6 uses the following tester for integer lattices which is based on querying a random collection of coordinates and verifying whether all of them are integral.

Lemma 5.6.7 *For every $0 < \epsilon \leq 1$ and every $0 < s \leq 1$, there exists a non-adaptive linear ℓ_p-tester $T_p(\epsilon, 0, s, q_Z)$ for $\mathbb{Z}^n$ with query complexity*

$$q_Z = O\left(\frac{1}{\epsilon^p} \log \frac{1}{s}\right).$$

5.6.1 2-sided to Linear Tester

In this section, we prove Lemma 5.6.3. Given a 2-sided adaptive tester $T(\epsilon, c, s, q)$ for inputs x from the domain $\mathcal{Z}_d^n$, we build an adaptive linear (thus one-sided) test $T'(\epsilon, 0, c+s, q)$ for inputs from the same domain $\mathcal{Z}_d^n$ with the same query complexity as that of T in this section.

For an index set $J \subseteq [n]$ and a vector $w \in \mathcal{Z}_d^n$, let $X(w, J) := \{x \in \mathcal{Z}_d^n \mid (\forall\ i \in J)\ x_i = w_i\ \}$. For a subset of coordinates $J \subseteq [n]$ and a vector $w \in \mathcal{Z}_d^n$, we say that *there exists a dual witness for* $X(w, J)$ if there exists $\alpha \in L_J^\perp$ such that $\langle \alpha, w \rangle \notin \mathbb{Z}$. That is, a dual witness α is a dual vector entirely supported on J that proves none of the vectors in $X(w, J)$ (and thus w) can be in the lattice. Recall that $V := L \bmod d$ where $d\mathbb{Z}^n \subseteq L$.

If the input vector x is from the domain $\mathcal{Z}_d^n$, then each coordinate of the input has d possible choices. Thus, any 2-sided adaptive tester $T(\epsilon, c, s, q)$ for inputs from the domain $\mathbb{Z}_d^n$, can be viewed as a distribution over deterministic decision trees with each leaf being labeled 1 if accepting and 0 if rejecting. Therefore we will express the tester as $T = (\Upsilon_T, D_T)$, where Υ_T is the set of all decision trees (with at most q queries) and D_T is a distribution over Υ_T.

Let l be a leaf of a decision tree. We denote the coordinates queried along the path to l by $\mathrm{var}(l)$. We denote the vector that is consistent with the queried coordinates along the path to l and has zeros in the non-queried coordinates by s_l. Let us define V_l to be the set of lattice vectors u which are consistent with the queries along the path to l. Similarly, let V_l^x be the set of vectors in $(x + V) \bmod d$ which are consistent with the queries along the path to l, i.e., $V_l = X(s_l, \mathrm{var}(l)) \cap V$ and $V_l^x = X(s_l, \mathrm{var}(l)) \cap ((x+V) \bmod d)$. We need the following claim about the sizes of V_l and V_l^x.

Claim 5.6.8 *For every leaf l in the decision tree Γ, if both V_l and V_l^x are non-empty, then $|V_l| = |V_l^x|$.*

Proof Let U denote the set of all the lattice vectors in $\mathcal{Z}_d^n$ which have all 0's in the positions queried along the path to l. We know that U is non-empty because the all zeros vector is in U.

For every $v \in V_l$ and $u \in U$, we have that $(v + u) \bmod d$ is also in V_l since we are only adding 0's at the queried coordinates. Similarly, for every vector $v' \in V_l^x$ and $u \in U$, we have that $(v' + u) \bmod d$ is also in V_l^x. Therefore, we know that $(U + v) \bmod d \subseteq V_l$ for every $v \in V_l$ and similarly, $(U + v') \bmod d \subseteq V_l^x$ for every vector $v' \in V_l^x$.

Further, for every two vectors $u, v \in V_l$, we have that $(u - v) \bmod d$ is in U and since u and v are both consistent along the path to l, the vector $u - v$ has all zeros at the queried coordinates. So, $(u - v) \bmod d \in U$. Therefore, $(V_l - v) \bmod d \subseteq U$ for every $v \in V_l$ and hence $V_l \subseteq (U + v) \bmod d$ for every $v \in V_l$. Similarly, for every vector $v' \in V_l^x$, we have that $(V_l^x - v') \bmod d \subseteq U$ and hence $V_l^x \subseteq (U + v') \bmod d$.

Therefore, if V_l and V_l^x are non-empty, then $(U + v) \bmod d = V_l$ for every vector $v \in V_l$ and $(U + v') \bmod d = V_l^x$ for every vector $v' \in V_l^x$. Hence, $|V_l| = |U| = |V_l^x|$.

■

We now show that if a linear test accepts, then there exists a lattice vector that is consistent with the queried coordinates. In other words, if there is no dual witness then there is a lattice vector that is accepted by the test.

In the following, let $\text{proj}_J(u) \in \mathbb{R}^{|J|}$ denote the projection of vector u to the coordinates in J and $\text{proj}_J(S)$ denote the set of vectors obtained by projecting the vectors in S to the coordinates in J. We note that the projection of a rational lattice to a set of coordinates gives a lattice again.

Proposition 5.6.9 *Let $J \subseteq [n]$, $w \in \mathcal{Z}_d^n$. If $\langle \alpha, proj_J(w) \rangle \in \mathbb{Z}$ for every $\alpha \in proj_J(L_J^{\perp})$, then $V \cap X(w, J) \neq \emptyset$.*

Proof We recall that the dual of a projection of a lattice is the set of vectors in the projected space which have integral dot products with all points in the projected lattice. The following proposition shows that the dual of a projected lattice is the

projection of the set of vectors in the dual lattice whose support is contained in the projection.

Proposition 5.6.10 *Let $J \subseteq [n]$. Then*

$$(proj_J(L))^\perp = proj_J\left(L_J^\perp\right).$$

Proof Let $\alpha_J \in \text{proj}_J(L_J^\perp)$. Let us extend the vector α_J to $\alpha \in \mathbb{R}^n$ by setting the coordinates that are not in J to zero. We note that $\alpha \in L_J^\perp$. Hence $\langle \alpha, x \rangle \in \mathbb{Z}$ for every $x \in L$. Therefore $\langle \alpha_J, x_J \rangle \in \mathbb{Z}$ for every $x_J \in \text{proj}_J(L)$. Thus, $\alpha_J \in (\text{proj}_J(L))^\perp$.

Let $\alpha_J \in (\text{proj}_J(L))^\perp$. Then for every $v_J \in \text{proj}_J(L)$, we have $\langle \alpha_J, v_J \rangle \in \mathbb{Z}$. Consequently for every $v \in L$, we have $\langle \alpha_J, \text{proj}_J(v) \rangle \in \mathbb{Z}$. Let us extend the vector α_J to $\alpha \in \mathbb{R}^n$ by setting the coordinates that are not in J to zero. Then $\langle \alpha, v \rangle \in \mathbb{Z}$ for every $v \in L$. Therefore $\alpha \in L^\perp$ and hence $\alpha_J \in \text{proj}_J(L_J^\perp)$. ■

We have that $\langle \alpha, \text{proj}_J(w) \rangle \in \mathbb{Z}$ for every $\alpha \in \text{proj}_J(L_J^\perp)$. Therefore $\text{proj}_J(w) \in (\text{proj}_J(L_J^\perp))^\perp$. By Proposition 5.6.10, we have that $\text{proj}_J(w) \in \text{proj}_J(L)$. Hence, there exists $x \in X(w, J) \cap L = X(w, J) \cap V$. ■

Note that it is possible to determine if there exists a dual witness for $X(w, J)$ and if so, find one efficiently as shown in Proposition 5.6.11.

Proposition 5.6.11 *Given $w \in \mathcal{Z}_d^n$ and $J \subseteq [n]$, we can find a dual witness for $X(w, J)$ if one exists or confirm that no dual witness for $X(w, J)$ exists in time $O(|J|^\omega)$, where $O(m^\omega)$ is the time to compute the inverse of a $m \times m$ real matrix.*

Proof A basis for $\text{proj}_J(L)$ can be obtained by projecting the basis for L. Now a basis for the dual of the projected lattice, namely $\text{proj}_J(L)^\perp = \text{proj}_J(L_J^\perp)$, can be computed in time $O(|J|^\omega)$. We observe that for every $\alpha \in L_J^\perp$, we have $\langle \alpha, w \rangle \in \mathbb{Z}$ if and only if for every basis vector b of $\text{proj}_J(L_J^\perp)$, we have $\langle b, \text{proj}_J(w) \rangle \in \mathbb{Z}$. Hence it is sufficient to only verify the inner product of $\text{proj}_J(w)$ with the basis vectors of $\text{proj}_J(L_J^\perp)$. ■

We now have the ingredients needed to prove Lemma 5.6.3.

Proof of Lemma 5.6.3 We first relabel the decision tree according to the rule required for a linear test: Given a decision tree Γ for the tester T, we say that it is *optimally labeled* if the label of any leaf l is 0 whenever there exists a dual witness for $X(s_l, \mathrm{var}(l))$ and 1 otherwise. We denote the tree obtained from Γ by optimally relabeling to be Γ_{OPT} (the relabeling for a given leaf of a tree Γ can be done efficiently by Proposition 5.6.11). We build a tester T' as follows:

1. On input $x \in \mathcal{Z}_d^n$, choose a tree Γ according to D_T.
2. Choose a uniformly random vector v in V (recall that $V := L \bmod d$).
3. Answer according to the relabeled decision tree Γ_{OPT} on input $(x + v) \bmod d$.

It is clear that T' is a linear test and has the same query complexity as that of T. We now show that the probability of acceptance by T' of any vector w which is ϵ-far from L, does not exceed $c + s$. Let us define the following for a tester $\bar{T}$:

$$\rho^{\bar{T}} := \operatorname*{avg}_{y \in V} Pr[\bar{T}(y) = 1],$$

$$\rho_x^{\bar{T}} := \operatorname*{avg}_{y \in (x+V) \bmod d} Pr[\bar{T}(y) = 1].$$

Due to the randomness in the choice of the tester T', we have

$$\rho^{T'} = Pr[T'(x) = 1 \mid x \in V],$$

$$\rho_x^{T'} = Pr[T'(x) = 1].$$

Since T' is a 1-sided tester, we have that $\rho^{T'} = 1$. Since T accepts lattice vectors with probability at least $1 - c$, we have $\rho^T \geq 1 - c$. Let $x \in \mathcal{Z}_d^n$ be ϵ-far from L. For every $v \in V$, we have that $(x + v) \bmod d$ is also ϵ-far from L by Proposition 5.6.2. Therefore, $\rho_x^T \leq s$. Using Claim 5.6.12, we have

$$\rho_x^{T'} \leq \rho^{T'} - \rho^T + \rho_x^T \leq 1 - (1 - c) + s = c + s.$$

■

Claim 5.6.12 *For every $x \in \mathcal{Z}_d^n$,*

$$\rho_x^{T'} \leq \rho^{T'} - \rho^T + \rho_x^T.$$

Proof Let x be a vector in $\mathcal{Z}_d^n$. We analyze the effect of relabeling a single leaf l of the decision tree Γ. We show that relabeling l optimally preserves the claim and hence by repeated relabeling, we can deduce the claim.

Case (i). There exists a dual witness for $X(s_l, \text{var}(l))$. Then the leaf l is relabeled from 1 to 0. If input $y \in X(s_l, \text{var}(l))$, then y cannot be a lattice vector (if y is a lattice vector, then there cannot exist a dual witness for $X(s_l, \text{var}(l))$). Therefore, the probability of acceptance of lattice vectors is not changed due to relabeling, i.e., $\rho^{T'} = \rho^T$. If the leaf l is reached for input $y \in \mathcal{Z}_d^n \setminus V$, then T' rejects. Thus, relabeling does not increase the probability of acceptance of non-lattice vectors, i.e., $\rho_y^{T'} \leq \rho_y^T$. Therefore, $\rho_x^{T'} \leq \rho^{T'} - \rho^T + \rho_x^T$ holds for this case.

Case (ii). There does not exist a dual witness for $X(s_l, \text{var}(l))$. Then the leaf l is relabeled from 0 to 1.

The set of vectors in $V_l \cup V_l^x$ were rejected by T and, after optimal relabeling of the leaf l, are now accepted by T'. The rest of the vectors in V and $(x + V) \bmod d$ are rejected/accepted equally by both T and T'.

Now, if y was a lattice vector, then the probability of accepting a lattice vector increases because of the relabeling of l. Among the vectors in V, the vectors in V_l are precisely the ones which were rejected before relabeling and are now accepted after relabeling. Since we average over all possible vectors $y \in V$ in the definition of ρ^T, the fractional change in the acceptance probability given that T' and T chose the decision tree Γ is exactly $|V_l|/|V|$. Therefore,

$$\rho^{T'} = \rho^T + D_T(\Gamma)\frac{|V_l|}{|V|}.$$

Among the vectors in $(x + V) \bmod d$, the vectors in V_l^x are the only vectors which were rejected before relabeling and are now accepted after relabeling. Thus,

the fractional change in the acceptance probability of $(x+v) \bmod d$ given that T' and T chose the decision tree Γ is exactly $|V_l^x|/|V|$. Therefore,

$$\rho_x^{T'} = \rho_x^T + D_T(\Gamma)\frac{|V_l^x|}{|V|}.$$

Combining the two equations, we get

$$\rho_x^{T'} = \rho^{T'} - \rho^T + \rho_x^T + \frac{D_T(\Gamma)}{|V|}(|V_l^x| - |V_l|).$$

Using Claim 5.6.8, we know that $|V_l^x| \leq |V_l|$ if V_l is non empty. Since there does not exist a dual witness for l, by Proposition 5.6.9, we have that V_l is non-empty. Hence the claim follows. ■

5.6.2 Adaptive to Non-adaptive

In this section we show that given an adaptive linear tester for a lattice, we can construct a non-adaptive linear tester from it without increasing the query complexity or the acceptance probability of non-lattice vectors.

Proof of Lemma 5.6.4 Let $T(\epsilon, 0, s, q)$ be an adaptive linear tester for inputs from the domain $\mathcal{Z}_d^n$ with query complexity q. We construct a non-adaptive linear tester $T'(\epsilon, 0, s, q)$ for inputs from the domain $\mathcal{Z}_d^n$ as follows:

1. On input $x \in \mathcal{Z}_d^n$, choose a random vector $v \in V$.
2. Run T on input v. Let J denote the set of coordinates that are queried.
3. Query x on all the coordinates in J.
4. Reject if and only if there exists a dual witness for $X(x, J)$.

We note that T' is a linear test and the query complexity of T' is the same as the query complexity of T. Since the queries depend only on a random $v \in V$ and not on the input x, the test T' is non-adaptive. It remains to bound the acceptance probability of non-lattice vectors by T'. We will show that there is no dual witness for

$X(x, J)$ if and only if there exists a vector $y \in (x + V) \bmod d$ that is consistent with the queried coordinates of v. As a consequence, we will show that the probability that T' accepts x is identical to the average acceptance probability of $x + v$ for random vectors $v \in V$ by T. Before analyzing the acceptance probability, we introduce a few notations and observations.

For a decision tree $\Gamma \in \Upsilon_T$, we denote the set of leaves of Γ which are labeled 1 by $l_1(\Gamma)$. For a leaf l of Γ and a vector $x \in \mathcal{Z}_d^n$, let I_l^x be a boolean (indicator) variable which takes a value of 1 if and only if $\langle \alpha, x \rangle \in \mathbb{Z}$ for every $\alpha \in L_{var(l)}^{\perp}$.

Let $\bar{\Gamma}$ be the decision tree chosen by the tester T' on input x. The random vector $v \in V$ chosen by T' corresponds to a leaf labeled 1 in $\bar{\Gamma}$. This is because T is a linear test and hence a lattice vector v cannot have any dual witness. Therefore, $v \in V_{\bar{l}}$ for some $\bar{l} \in l_1(\bar{\Gamma})$. Since T' is a linear test it is clear that T' accepts x if and only if $I_{\bar{l}}^x = 1$.

Claim 5.6.13 *Let l be a leaf of a decision tree $\Gamma \in \Upsilon_T$, $x \in \mathcal{Z}_d^n$ and $y \in (x + V) \bmod d$. We have that $I_l^x = 1$ if and only if $I_l^y = 1$.*

Proof If $y \in (x+V) \bmod d$, then $x-y \in L$. If $x, y \in \mathbb{Z}^n$ belong to the same coset of L, then for every $a \in L^{\perp}$, we have that $\langle x, a \rangle \in \mathbb{Z}$ if and only if $\langle y, a \rangle \in \mathbb{Z}$. Therefore, there exists $\alpha \in L_{var(l)}^{\perp}$ such that $\langle \alpha, x \rangle \notin \mathbb{Z}$ if and only if there exists $\alpha \in L_{var(l)}^{\perp}$ such that $\langle \alpha, y \rangle \notin \mathbb{Z}$. Hence $I_l^x = 1$ if and only if $I_l^y = 1$ for every $y \in (x + V) \bmod d$. ■

Claim 5.6.14 *Let $x \in \mathcal{Z}_d^n$ and l be a leaf of a decision tree $\Gamma \in \Upsilon_T$ such that $l \in l_1(\Gamma)$. Then $|V_l^x| = I_l^x |V_l|$.*

Proof We know that for every leaf l which is labeled 1, the set V_l is non-empty since T is a linear tester (using Proposition 5.6.9). By Claim 5.6.8 we know that $|V_l^x| = |V_l|$ if V_l^x is also non-empty. Therefore it is sufficient to show that $I_l^x = 1$ if and only if V_l^x is non-empty

If V_l^x is non-empty, then by definition, there is a vector $y \in (x + V) \bmod d$ which is consistent with all the queries along the path to l. Since l is labeled 1, we know

that T accepts y. Since T is a linear tester, this implies that there does not exist an $\alpha \in L^{\perp}_{\mathrm{var}(l)}$ such that $\langle \alpha, x \rangle \notin \mathbb{Z}$. Hence $I^y_l = 1$. By Claim 5.6.13, we know that I^x_l is also 1.

If $I^x_l = 1$, then for every $\alpha \in L^{\perp}_{\mathrm{var}(l)}$, we have $\langle \alpha, x \rangle \in \mathbb{Z}$. By Proposition 5.6.9, there exists a vector $v \in V \cap X(x, \mathrm{var}(l))$. Hence, we have a vector $v \in V$ whose entries are identical to that of x at the coordinates in $\mathrm{var}(l)$. We observe that the vector $(x - v) \bmod d$ has all 0 entries at the coordinates in $\mathrm{var}(l)$. Further, V_l is non-empty since l is labeled 1. Let $u \in V_l$. Then $((x - v) + u) \bmod d$ is consistent with all queries along the path to l, and is in $(x + V) \bmod d$. Therefore V^x_l is non-empty. ■

We now show that the acceptance probability of T' is equal to the average acceptance probability of T. Let

$$\rho_x := \operatorname*{avg}_{v \in V} \Pr[T((x + v) \bmod d) = 1].$$

We note that this quantity is 1 if $x \in V$ and is at most s if x is ϵ-far from the lattice L. The following claim shows that T' accepts an input vector x with probability 1 if $x \in V$ and with probability at most s if x is ϵ-far from the lattice L. ■

Claim 5.6.15 *Let $x \in \mathcal{Z}^n_d$. Then $\Pr[T'(x) = 1] = \rho_x$.*

Proof The average acceptance probability of T can be viewed as follows: we pick a decision tree Γ according to D_T. Then we pick a leaf l labeled 1 with probability proportional to the fraction of vectors in $x + V \bmod d$ that are consistent with the queries along the path to l. Therefore,

$$\rho_x = \sum_{\Gamma \in \Upsilon_T} D_T(\Gamma) \left(\sum_{l \in l_1(\Gamma)} \frac{|V^x_l|}{|V|} \right).$$

We have seen that $T'(x) = 1$ if and only if for the random vector $v \in V$ chosen by T', and a leaf $\bar{l} \in l_1(\bar{\Gamma})$ such that $v \in V_{\bar{l}}$, we have $I^x_{\bar{l}} = 1$. Thus the execution of T' can be treated as follows: First, pick a decision tree $\Gamma \in \Upsilon_T$ according to $D_T(\Gamma)$, then choose a leaf l labeled 1 in Γ with probability proportional to the fraction of vectors

in V that are consistent with the queries to the coordinates in l. Finally, query x on the variables in $\mathrm{var}(l)$ and accept if and only if $I_l^x = 1$. Therefore, the acceptance probability of T is given by

$$\Pr[T(x) = 1] = \sum_{\Gamma \in \Upsilon_T} D_T(\Gamma) \left(\sum_{l \in l_1(\Gamma)} \frac{|V_l|}{|V|} \cdot I_l^x \right).$$

By Claim 5.6.14, we see that $\rho_x = \Pr[T'(x) = 1]$. ■

5.6.3 Handling Real-Valued Inputs

In this section, we build a tester for real-valued inputs using a tester for bounded integral inputs. We first show how to handle all integral inputs using a tester for integral inputs from a bounded domain.

Proof of Lemma 5.6.5 If we have a tester $T'(\epsilon, c, s, q)$ for integral inputs, then the same tester can be applied to inputs in $\mathcal{Z}_d^n$ with the same completeness and soundness parameters and the same query complexity. Given a tester $T(\epsilon, c, s, q)$ for inputs from the domain $\mathcal{Z}_d^n$, we construct the tester $T'(\epsilon, c, s, q)$ for arbitrary integral inputs as follows: On input $x \in \mathbb{Z}^n$ run $T(\epsilon, c, s, q)$ on $w := x \bmod d$, and output the result.

If x is a lattice vector, then from Proposition 5.6.2, we know that w is also a lattice vector, and therefore T' accepts x with probability at least $1 - c$. If x is ϵ-far from the lattice, then again from Proposition 5.6.2, we know that w is also ϵ-far from the lattice and T' will accept x with probability at most s. We note that the query complexity of T' is identical to that of T. ■

To address the case of real inputs, we will design a tester for the integer lattice.

Proof of Lemma 5.6.7 The test queries $O((1/\epsilon^p)\log(1/s))$ coordinates of the input uniformly at random and accepts iff all the queried coordinates are integral.

If the input is in the lattice, then all the queried coordinates will be integral, and hence the tester will accept. If the input w is at ℓ_p distance at least $\epsilon \cdot \|1^n\|_p$, then at least $\epsilon^p n$ coordinates of the input are non-integral. Thus the tester will reject with probability at least $1 - s$.

We note that the tester is a linear test: the test described can be viewed as picking independent uniform random standard basis vectors $e_i \in \mathbb{Z}^n \subseteq L^\perp$ (where e_i is the indicator vector of the index i), for $i \in [n]$, and testing if the input w satisfies $\langle w, e_i \rangle \in \mathbb{Z}$.

■

Proof of Lemma 5.6.6 Suppose we have a ℓ_p-tester $T(\epsilon, c, s, q)$ for integer inputs. We can build a tester T' for real valued inputs as follows:

1. On input $x \in \mathbb{R}^n$, run the ℓ_p-tester $\bar{T}(\epsilon/2, 0, s, q_Z)$ for $\mathbb{Z}^n$ from Lemma 5.6.7 on input x. If the tester $\bar{T}$ rejects, then reject.

2. Else, run $T(\epsilon/2, c, s, q')$ on x where $q' = q(\epsilon/2, c, s)$ and reject if any of the coordinates queried are not integers; otherwise output the result of T.

If x is a lattice vector, then the acceptance probability of T' is the same as that of T since the tester used in step 1 is a linear tester. If $d_p(x, L) \geq \epsilon \cdot \|1^n\|_p$, then $d_p(x, \lfloor x \rceil) + d_p(\lfloor x \rceil, L) \geq d_p(x, L) \geq \epsilon \cdot \|1^n\|_p$ and therefore either $d_p(x, \lfloor x \rceil)$ or $d_p(\lfloor x \rceil, L)$ is at least $\frac{1}{2}\epsilon \cdot \|1^n\|_p$. If $d_p(x, \lfloor x \rceil) \geq \frac{1}{2}\epsilon \cdot \|1^n\|_p$, then $d_p(x, \mathbb{Z}^n) = d_p(x, \lfloor x \rceil) \geq \frac{1}{2}\epsilon \cdot \|1^n\|_p$ and therefore step 1 rejects with probability at least $1 - s$. If $d_p(\lfloor x \rceil, L) \geq \frac{1}{2}\epsilon \cdot \|1^n\|_p$, then step 2 rejects with probability at least $1 - s$.

The number of queries made by the tester T' is $q(\epsilon/2, c, s) + O((1/\epsilon^p) \log(1/s))$. We note that since the tester used in step 1 is a non-adaptive linear tester, T' would be a non-adaptive linear tester if T is a non-adaptive linear tester. ■

Remark We note that the test described in the proof of Theorem 5.2.1 is not a linear test by definition. We now describe a linear test for the code-formula lattice which is equivalent to the test described in Section 5.4.1.

Let T_p denote the tester for $\mathbb{Z}^n$. We assume that each code tester T_i for the code C_i is linear [86] (i.e T_i queries the input $t_i \in \{0,1\}^n$ at $I_i = \{i_1, \cdots, i_q\} \subseteq [n]$ coordinates according to some distribution and accepts it if and only if $\langle t_i, v \rangle \equiv 0 \bmod 2$ for every

$v \in C_{I_i}^{\perp}$). Consider the following variant of the test, that we call T_{linear}, which by definition is a linear test:

1. Let each T_i query $I_i \subseteq [n]$ coordinates and let T_p query I_p coordinates.
2. Let $I = \cup_i I_i \cup I_p$.
3. Accept t if $\langle t, x \rangle \in \mathbb{Z}$ for all $x \in (L^{\perp})_I$
4. Reject otherwise.

Note that T_{linear} makes at most as many queries as T.

If the input is a lattice vector; i.e., $t \in L$, then by definition, the inner product of t with every dual lattice vector would be an integer. Therefore, the test is 1-sided.

We now show that T_{linear} rejects all inputs t which are rejected by T and hence, T_{linear} performs at least as well as T. If T rejects t, then there is some $i \in \{0, 1, \ldots, m-1\}$ such that t_i is rejected by T_i or t is rejected by T_p. We note that each code tester T_i and also T_p are linear. Therefore, if T_i rejects t_i, then there are no codewords of C_i which agree with t_i on the coordinates I_i queried by T_i. So, for the set I which contains I_i, there are no codewords of C_i which agree with t_i on the coordinates in I. If T_p rejects t, then there is some non-integral coordinate in I_p and hence in I. By definition of the code formula construction, t is a lattice vector if and only if for each $i = 0, \ldots m-1$, t_i is a codeword in C_i and $t \in \mathbb{Z}^n$. Hence, no lattice vector of L agrees with t on those set of coordinates. Therefore, there exists a dual lattice vector supported on I, which does not have an integral inner product with t. Therefore, T_{linear} also rejects t (thus, has at least as good a soundness as the original test T).

5.7 Testing Inputs Outside the Span of the Lattice

In this section we prove Theorem 5.2.11, Theorem 5.2.12, Corollary 5.2.13 and Theorem 5.2.14. We first recall the definitions. Let L be a rank k lattice in $\mathbb{Z}^n$. Let S denote the $span(L)$ and $S^{\perp}$ be the subspace orthogonal to S. Let $U =$

$[u_1, \cdots, u_{n-k}]^T \in \mathbb{R}^{(n-k)\times n}$ be an orthonormal basis for $S^\perp$. Let $P \subseteq [n]$ be the set of coordinates that support the vectors in $S^\perp$ i.e.,

$$P := \bigcup_{i\in[n-k]} supp(u_i).$$

Proof of Theorem 5.2.11 To show the $\Omega(|P|)$ lower bound, we use Yao's principle: we setup a distribution $\mathcal{D}$ on far inputs such that every deterministic algorithm requires $\Omega(|P|)$ queries to distinguish whether the input is $0 \in L$ or is far from L. We define $\mathcal{D}$ as follows: pick j uniformly at random from P, and set $t^{(j)} := De_j$, where $D \geq \frac{\epsilon\cdot\|1^n\|_p}{\min\limits_{i,j:u_{i,j}\neq 0}|u_{i,j}|}$. The following claim shows that the distance of each such $t^{(j)}$ from L is at least $\epsilon \cdot \|1^n\|_p$.

Claim 5.7.1 *$d_p(t^{(j)}, L) \geq \epsilon \cdot \|1^n\|_p$ for every $j \in P$.*

Proof It is sufficient to show that $t^{(j)}$ is far from S, since $L \subseteq S$. Let $t^{(j)} = t^{(j)\|} + t^{(j)\perp}$, where $t^{(j)\|}$ is the component of $t^{(j)}$ in S and $t^{(j)\perp}$ is the component of t in $S^\perp$. By definition,

$$t^{(j)\perp} = proj_{S^\perp}(t^{(j)}) = \sum_{\ell\in[n-k]} \langle t^{(j)}, u_\ell\rangle u_\ell.$$

Since U is an orthonormal basis of $S^\perp$,

$$\|t^{(j)\perp}\|_p^p = \sum_{\ell\in[n-k]} |\langle t^{(j)}, u_\ell\rangle|^p = \sum_{\ell\in[n-k]} (Du_{\ell,j})^p \geq (\epsilon \cdot \|1^n\|_p)^p.$$

The last inequality follows from the choice of D and the fact that there exists at least one $u_{\ell,j} \neq 0$ since $j \in P$. Therefore, the distance of $t^{(j)}$ from S and hence from L, is at least $\epsilon \cdot \|1^n\|_p$. ■

By the choice of the distribution, every deterministic test fails on inputs drawn from $\mathcal{D}$ with probability $1/|P|$. Thus any randomized test requires $\Omega(|P|)$ queries in order to succeed with constant probability. ■

Proof of Theorem 5.2.12 Let $T(\epsilon, c, s, q)$ be an ℓ_p-tester for L for inputs in $span(L)$ with query complexity $q = q(\epsilon)$. We now design a tester $T'(\epsilon', c', s', q')$ for

L for inputs $t = (t_1, t_2, \cdots, t_n) \in \mathbb{R}^n$. By making an additional $|P|$ queries, T' can compute the coordinates of the projection of t onto S. If t is far from L, then either (i) t is far from S or (ii) t is close to S but far from L. The coordinates in P would identify if t is far from S and enable rejection. If t is close to S but far from L, the tester T would reject the projection of t onto S thus enabling rejection. We now formalize this intuition.

Let $t = (t_1, t_2, \cdots, t_n) \in \mathbb{R}^n$ be the input to the tester T'. We compute the projection of t on $span(L)$ by querying all the coordinates in P. Let $t^\perp$ be the projection of t onto $S^\perp$. Since U is an orthonormal basis for $S^\perp$, we have

$$t^\perp = \sum_{\ell \in [n-k]} \langle t, u_\ell \rangle u_\ell.$$

Each inner product in this expression can be computed using only the coordinates in P and therefore, $t^\perp$ can be computed from t by querying just $|P|$ coordinates. If $\|t^\perp\|_p \geq \epsilon'/2 \cdot \|1^n\|_p$, then T' rejects t immediately. So we now assume $\|t^\perp\|_p < \epsilon'/2 \cdot \|1^n\|_p$ The projection of t onto S is:

$$t_j^{\|} = \begin{cases} t_j & \text{if } j \notin P \\ t_j - t_j^\perp & \text{if } j \in P \end{cases}$$

Now we run the tester for T on input $t^{\|}$ for distance parameter $\epsilon = \epsilon'/2$ and accept t if and only if T accepts.

If $t \in L$, then $t^\perp = 0$ and T would accept with probability at least $1 - c$. If $d_p(t, L) \geq \epsilon' \cdot \|1^n\|_p$, then

$$d_p(t^{\|}, L) \geq \epsilon' \cdot \|1^n\|_p - d_p(t^\perp, S) \geq \epsilon'/2 \cdot \|1^n\|_p = \epsilon \cdot \|1^n\|_p.$$

Therefore, T would reject with probability at least $1 - s$. Finally, note that $q'(\epsilon') \leq q(\epsilon'/2) + |P|$. ■

5.7.1 Testing Knapsack Lattices

Proof of Corollary 5.2.13 We note that L has rank $n - 1$ and the vector $(a_1, a_2, \ldots, a_{n-1}, -1)$ generates the subspace orthogonal to $span(L)$, hence the set

P of elements in the support of this space has size $|P| = n$, and the lower bound follows from Theorem 5.2.11. ■

We now prove Theorem 5.2.14, namely that knapsack lattices can be tested with a constant number of queries if the inputs come from the span of the lattice. In fact, we will show that testing such lattices simply reduces to testing membership in $\mathbb{Z}^n$.

Proof of Theorem 5.2.14 Let $L = L_{a_1,\ldots,a_{n-1}}$. Let $w \in \text{span}(L)$ denote the input. Any vector $w \in \mathit{span}(L)$ is of the form

$$w = \left(\alpha_1, \cdots, \alpha_{n-1}, \sum_{i=1}^{n-1} a_i \alpha_i\right)$$

for some real values $\alpha_1, \ldots, \alpha_{n-1}$. Let $w' \in \mathbb{R}^{n-1}$ denote the projection of w on the first $n-1$ coordinates. Let $T_p(\epsilon', 0, s', q')$ denote the ℓ_p-tester for $\mathbb{Z}^{n-1}$, where $q' = O\left(\left(\frac{1}{\epsilon'^p}\right) \log \frac{1}{s'}\right)$.

The tester proceeds as follows: Run the tester $T_p(\epsilon' = \epsilon/(M+1)^{1/p}, 0, s, q = O\left(\left(\frac{M}{\epsilon^p}\right) \log \frac{1}{s}\right))$ on input w'. Accept if and only if the tester T_p accepts.

The query complexity of the tester is immediate. If $w \in L$, then each coordinate is integral. Therefore the test accepts w with probability 1. We use the following claim to analyze the soundness of the test.

Claim 5.7.2 *Let $w \in \mathit{span}(L)$, and $w' = (w_1, \cdots, w_{n-1}) \in \mathbb{R}^{n-1}$ then,*

$$d(w, L)^p \leq (M+1) \cdot d(w', \mathbb{Z}^{n-1})^p$$

Proof Let $\lfloor w_i \rceil$ denotes the rounding of w_i the nearest integer. Consider the following vector $v = (\lfloor w_1 \rceil, \cdots, \lfloor w_{n-1} \rceil, \sum_{i=1}^{n-1} a_i \lfloor w_i \rceil) \in L$. We now upper bound the distance of w from L using this lattice vector v.

$$
\begin{aligned}
d(w,L)^p \le d(w,v)^p &= \|w-v\|_p^p \\
&= \sum_{i=1}^{n-1} |w_i - \lfloor w_i \rceil|^p + |w_n - \sum_{i=1}^{n-1} a_i \lfloor w_i \rceil|^p \\
&= \sum_{i=1}^{n-1} |w_i - \lfloor w_i \rceil|^p + |\sum_{i=1}^{n-1} a_i w_i - \sum_{i=1}^{n-1} a_i \lfloor w_i \rceil|^p \\
&\le \sum_{i=1}^{n-1} |w_i - \lfloor w_i \rceil|^p + M \sum_{i=1}^{n-1} |w_i - \lfloor w_i \rceil|^p \\
&= (M+1) \cdot \sum_{i=1}^{n-1} |w_i - \lfloor w_i \rceil|^p \\
&= (M+1) \cdot d(w', \mathbb{Z}^{n-1})^p
\end{aligned}
$$

■

It remains to bound the soundness error probability. If $d(w,L) \ge \epsilon \|1^n\|_p$, then from Claim 5.7.2, we get that $d(w', \mathbb{Z}^{n-1}) \ge (\epsilon/(M+1)^{1/p}) \|1^n\|_p$. Therefore, the tester T_p rejects w with probability at least $1-s$.

■

5.8 Discussion and Open Questions

In this chapter we defined a notion of local testing for a new family of objects: point lattices. Our results demonstrate connections between lattice testing and the ripe theory of locally testable codes, and raises several open questions that merit further study:

1. (Questions 5.1.2) An intriguing direction is to construct lattices with various trade-offs between the rank, dimension and the number of queries necessary for testing. This is also a well-studied and active research direction in the study of error-correcting codes.

2. (Questions 5.1.3) Another related question is to understand necessary and sufficient conditions that make a lattice testable with constant number of queries. In the case of testing codes, the "symmetry" of the code leads to well-studied necessary and sufficient conditions (see, e.g., [91] for a survey). In this work we show that some lattices obtained from Reed-Muller codes are testable. These lattices inherit the large set of symmetries of the constituent Reed-Muller codes. It would be interesting to understand the role of symmetry in testing lattices.

3. *Local decodability* is a notion similar to local testability of error-correcting codes. Here the goal is to decode one bit of the message using only a few bits of the received word. For locally decodable codes, a well studied problem is to understand the tradeoffs between rate, dimension and query complexity (see survey by Yekhanin [92]). This problem can be immediately transferred to *locally decodable lattices*, where one may hope to achieve better tradeoffs. Moreover, it would be interesting to explore connections between private information retrieval schemes [93] and locally decodable lattices. Such connections have been well-established for locally decodable codes.

4. Finally, one may explore questions similar to the ones studied here by varying the testing model depending on the application of interest. For instance, in applications like IP and cryptography, it is natural to ask for a notion of tester that ensures that scaling the lattice does not change the query complexity. An alternate definition of ϵ-far based on the *covering radius* of the lattice could be helpful to achieve this property. In order to have a tester notion where scaling preserves query complexity, we may define a vector as being ϵ-far from the lattice, if the distance of the vector to every lattice point is at least ϵ times the covering radius of the lattice. This definition is essentially equivalent to the current Definition 5.1.1 if the covering radius of the lattice is $\Theta(n)$.

REFERENCES

REFERENCES

[1] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[2] Richard W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950.

[3] John William Scott Cassels. *An Introduction to the Geometry of Numbers*. Springer Science & Business Media, 2012.

[4] A.K. Lenstra, H.W. Lenstra Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[5] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, August 1987.

[6] H.W. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

[7] Andrew M. Odlyzko. The rise and fall of knapsack cryptosystems. *Cryptology and Computational Number Theory*, 42:75–88, 1990.

[8] Antoine Joux and Jacques Stern. Lattice reduction: A toolbox for the cryptanalyst. *Journal of Cryptology*, 11(3):161–185, 1998.

[9] Phong Q. Nguyen and Jacques Stern. The two faces of lattices in cryptology. In *Cryptography and Lattices*, pages 146–180. Springer, 2001.

[10] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of 28th Annual ACM Symposium on the Theory of Computing*, pages 99–108, 1996.

[11] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C.A. Van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[12] Peter van Emde Boas. *Another NP-Complete Partition Problem and the Complexity of Computing Short Vectors in a Lattice*. Universiteit van Amsterdam. Mathematisch Instituut, 1981.

[13] Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences*, 54(2):317–331, 1997.

[14] Irit Dinur, Guy Kindler, Ran Raz, and Shmuel Safra. Approximating CVP to within almost-polynomial factors is NP-hard. *Combinatorica*, 23(2):205–243, 2003.

[15] Michael Szydlo. Hypercubic lattice reduction and analysis of GGH and NTRU signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 433–448. Springer, 2003.

[16] Ishay Haviv and Oded Regev. On the lattice isomorphism problem. In *Proceedings of 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 391–404, 2014.

[17] Wilhelm Plesken and Bernd Souvignier. Computing isometries of lattices. *Journal of Symbolic Computation*, 24(3/4):327–334, 1997.

[18] H.W. Lenstra Jr. and A. Silverberg. Revisiting the Gentry-Szydlo algorithm. In *Advances in Cryptology*, volume 8616 of *Lecture Notes in Computer Science*, pages 280–296. Springer, 2014.

[19] H.W. Lenstra Jr. and A. Silverberg. Lattices with symmetry. *CoRR*, abs/1501.00178, 2015.

[20] László Babai. Automorphism groups, isomorphism, reconstruction. In *Handbook of Combinatorics*, volume 27, pages 1447–1540. North-Holland, 1996.

[21] Mathieu Dutour Sikiric, Achill Schürmann, and Frank Vallentin. Complexity and algorithms for computing voronoi cells of lattices. *Mathematics of Computation*, 78(267):1713–1731, 2009.

[22] J. Conway, N.J.A. Sloane, and E. Bannai. *Sphere Packings, Lattices and Groups*. A series of comprehensive studies in mathematics. Springer, 1999.

[23] W. Wesley Peterson. Encoding and error-correction procedures for the Bose-Chaudhuri codes. *IRE Transactions on Information Theory*, 6(4):459–470, 1960.

[24] Lloyd R. Welch and Elwyn R. Berlekamp. Error correction for algebraic block codes, 1986. US Patent 4,633,470.

[25] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997.

[26] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45:1757–1767, 1999.

[27] V. Guruswami and A. Vardy. Maximum-likelihood decoding of Reed-Solomon codes is NP-hard. *IEEE Transactions on Information Theory*, 51(7):2249–2256, 2005.

[28] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[29] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[30] Oded Goldreich. Short locally testable codes and proofs: A survey in two parts. In *Property Testing – Current Research and Surveys*, pages 65–104, 2010.

[31] Venkata Gandikota, Badih Ghazi, and Elena Grigorescu. On the NP-hardness of bounded distance decoding of Reed-Solomon codes. In *IEEE International Symposium on Information Theory*, pages 2904–2908, 2015.

[32] Venkata Gandikota, Badih Ghazi, and Elena Grigorescu. NP-hardness of Reed-Solomon decoding and the Prouhet-Tarry-Escott problem. In *IEEE 57th Annual Symposium on Foundations of Computer Science*, pages 760–769, 2016.

[33] Karthekeyan Chandrasekaran, Venkata Gandikota, and Elena Grigorescu. Deciding orthogonality in Construction-A lattices. *SIAM Journal on Discrete Mathematics*, 31(2):1244–1262, 2017.

[34] Karthekeyan Chandrasekaran, Mahdi Cheraghchi, Venkata Gandikota, and Elena Grigorescu. Local testing for membership in lattices. In *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, 2016.

[35] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[36] Richard Singleton. Maximum distance q-nary codes. *IEEE Transactions on Information Theory*, 10(2):116–118, 1964.

[37] Ravindran Kannan and Achim Bachem. Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM Journal on Computing*, 8(4):499–507, 1979.

[38] John Leech and N.J.A. Sloane. Sphere packings and error-correcting codes. *Canadian Journal of Mathematics*, 23(4):718–745, 1971.

[39] Wittawat Kositwattanarerk and Frédérique E. Oggier. Connections between Construction-D and related constructions of lattices. *Designs, Codes and Cryptography*, 73(2):441–455, 2014.

[40] G. D. Forney. Coset codes-I: Introduction and geometrical classification. *IEEE Transactions on Information Theory*, 34(5):1123–1151, 1988.

[41] Philippe Gaborit and Gilles Zémor. On the construction of dense lattices with a given automorphisms group. In *Annales de l'institut Fourier*, volume 57, pages 1051–1062, 2007.

[42] Ralph C. Merkle and Martin E. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, 24(5):525–530, 1978.

[43] Adi Shamir. A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. In *Advances in Cryptology*, pages 279–288. Springer, 1983.

[44] Dorit Aharonov and Oded Regev. Lattice problems in NP ∩ co-NP. *Journal of the ACM*, 52(5):749–765, 2005.

[45] M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 72(6):1012–1042, 2006.

[46] Venkatesan Guruswami and Atri Rudra. Tolerant locally testable codes. In *Proceedings of RANDOM*, pages 306–317, 2005.

[47] Swastik Kopparty and Shubhangi Saraf. Tolerant linearity testing and locally testable codes. In *Proceedings of RANDOM*, pages 601–614, 2009.

[48] Jean Bernard Lasserre. *Moments, Positive Polynomials and their Applications*, volume 1. World Scientific, 2009.

[49] Eugéne Prouhet. Mémoire sur quelques relations entre les puissances des nombres. *Comptes Rendus de l'Académie des Sciences*, 33(225):1851, 1851.

[50] Leonard Eugene Dickson. *History of the Theory of Numbers, Volume II: Diophantine Analysis*, volume 2. Courier Corporation, 2013.

[51] E. M. Wright. Prouhet's 1851 solution of the Tarry-Escott problem of 1910. *The American Mathematical Monthly*, 66(3):199–201, 1959.

[52] Godfrey Harold Hardy and Edward Maitland Wright. *An Introduction to the Theory of Numbers*. Oxford Science Publications. Clarendon Press, 1979.

[53] Loo Keng Hua. *Introduction to Number Theory*. Springer, 1982.

[54] P. Erdos and G. Szekeres. On the product $\prod_{k=1}^{n}(1-z^a k)$. *Serbe Publications de l'Institut Mathématique*, 13:29–34, 1959.

[55] P. Borwein and C. Ingalls. The Prouhet-Tarry-Escott problem revisited. *L'Enseignement Mathématique*, 40:3–27, 1994.

[56] Peter Borwein, Petr Lisonek, and Colin Percival. Computational investigations of the Prouhet-Tarry-Escott problem. *Mathematics of Computation*, 72(244):2063–2070, 2003.

[57] Oded Goldreich, Ronitt Rubinfeld, and Madhu Sudan. Learning polynomials with queries: The highly noisy case. *SIAM Journal on Discrete Mathematics*, 13(4):535–570, 2000.

[58] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

[59] Parikshit Gopalan, Subhash Khot, and Rishi Saket. Hardness of reconstructing multivariate polynomials over finite fields. *SIAM Journal on Computing*, 39(6):2598–2621, 2010.

[60] Farzad Parvaresh and Alexander Vardy. Correcting errors beyond the Guruswami-Sudan radius in polynomial time. In *Proceedings of 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 285–294, 2005.

[61] Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on Information Theory*, 54(1):135–150, 2008.

[62] Ralf Koetter and Alexander Vardy. Algebraic soft-decision decoding of Reed-Solomon codes. *IEEE Transactions on Information Theory*, 49(11):2809–2825, 2003.

[63] Atri Rudra and Mary Wootters. Every list-decodable code for high noise has abundant near-optimal rate puncturings. In *Annual ACM Symposium on the Theory of Computing*, pages 764–773, 2014.

[64] Alexander Vardy. Algorithmic complexity in coding theory and the minimum distance problem. In *Proceedings of 29th Annual ACM Symposium on the Theory of Computing*, pages 92–109, 1997.

[65] Irit Dinur, Guy Kindler, Ran Raz, and Shmuel Safra. Approximating CVP to within almost-polynomial factors is NP-hard. *Combinatorica*, 23(2):205–243, 2003.

[66] Ilya Dumer, Daniele Micciancio, and Madhu Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Transactions on Information Theory*, 49(1):22–37, 2003.

[67] Uriel Feige and Daniele Micciancio. The inapproximability of lattice and coding problems with preprocessing. *Journal of Computer and System Sciences*, 69(1):45–67, 2004.

[68] Oded Regev. Improved inapproximability of lattice and coding problems with preprocessing. *IEEE Transactions on Information Theory*, 50(9):2031–2037, 2004.

[69] Qi Cheng. Hard problems of algebraic geometry codes. *IEEE Transactions on Information Theory*, 54(1):402–406, 2008.

[70] R. P. Stanley. *Enumerative Combinatorics*, volume 2. Cambridge University Press, 1999.

[71] J. Li and D. Wan. On the subset sum problem over finite fields. *Finite Fields and Their Applications*, 14(4):911–929, 2008.

[72] T. J. Schaefer. The complexity of satisfiability problems. In *Annual ACM Symposium on the Theory of Computing*, pages 216–226, 1978.

[73] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2009.

[74] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, pages 781–793, 2004.

[75] Swastik Kopparty and Shubhangi Saraf. Local list-decoding and testing of random linear codes from high error. *SIAM Journal on Computing*, 42(3):1302–1326, 2013.

[76] André Weil. *Sur les Courbes Algébriques et les Variétés quis' en Déduisent*. Number 1041. Hermann, 1948.

[77] P. Deligne. Applications de la formule des traces aux sommes trigonométrigues. In *Cohomologie Étale*, pages 168–232. Springer, 1977.

[78] Rudolf Lidl and Harald Niederreiter. *Finite Fields*, volume 20. Cambridge University Press, 1997.

[79] H.C. Chan, C.A. Rodger, and Jennifer Seberry. On inequivalent weighing matrices. *Ars Combinatoria*, 21-A:299–333, 1986.

[80] T. Kaufman and M. Sudan. Algebraic property testing: The role of invariance. In *Annual ACM Symposium on the Theory of Computing*, pages 403–412, 2008.

[81] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a Symposium on the Complexity of Computer Computations*, pages 85–103, 1972.

[82] Laurence A Wolsey and George L Nemhauser. *Integer and Combinatorial Optimization*. John Wiley & Sons, 2014.

[83] Yi-Kai Liu, Vadim Lyubashevsky, and Daniele Micciancio. On bounded distance decoding for general lattices. In *Proceedings of RANDOM*, pages 450–461, 2006.

[84] Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev. ℓ_p-testing. In *Annual ACM Symposium on the Theory of Computing*, pages 164–173, 2014.

[85] Uri Erez, Simon Litsyn, and Ram Zamir. Lattices which are good for (almost) everything. *IEEE Transactions on Information Theory*, 51(10):3401–3416, 2005.

[86] E. Ben-Sasson, P. Harsha, and S. Raskhodnikova. Some 3-CNF properties are hard to test. *SIAM Journal on Computing*, 35(1):1–21, 2005.

[87] Arnab Bhattacharyya, Swastik Kopparty, Grant Schoenebeck, Madhu Sudan, and David Zuckerman. Optimal testing of Reed-Muller codes. In *Proceedings of 51st Annual IEEE Symposium on Foundations of Computer Science*, pages 488–497, 2010.

[88] Daniele Micciancio. Lecture notes on lattice algorithms and applications, Winter 2012, Lecture 2, 2012.

[89] A-C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science*, pages 222–227, 1977.

[90] Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing Reed-Muller codes. *IEEE Transactions on Information Theory*, 51(11):4032–4039, 2005.

[91] Madhu Sudan. Guest column: testing linear properties: some general theme. *SIGACT News*, 42(1):59–80, 2011.

[92] Sergey Yekhanin. Locally decodable codes. *Foundations and Trends in Theoretical Computer Science*, 6(3):139–255, 2012.

[93] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *Journal of the ACM*, 45(6):965–981, November 1998.