7 A Unified Framework for List Decoding of Algebraic Codes

Be wise! Generalize!
Piccayune [sic] Sentinel

7.1 Introduction

In the previous chapter we presented list decoding algorithms for two widely-studied families of algebraic codes: Reed-Solomon codes and AG-codes. Owing to the importance of these codes, these results can be viewed as providing strong evidence to the general utility of list decoding as an algorithmic notion. Indeed, as we shall see in future chapters, they set the stage for a whole body of results about list decoding.

The reader might have already noticed a great deal of similarity between the general structure of the decoding algorithms for Reed-Solomon codes and AG-codes. Since Reed-Solomon codes are a special instance of AG-codes, the decoding algorithm for AG-codes is just a generalization of the Reed-Solomon decoding algorithm, and this should explain the great deal of similarity between the algorithms. In this chapter, we will present a further generalization of the decoding algorithm by presenting a unified algorithm for soft decoding a general family of algebraic codes (which we call *ideal-based codes*). The decoding algorithms for Reed-Solomon and AG-codes are then just special cases of this general paradigm. Such a unified framework for list decoding is important for two reasons. Firstly, such unifications are elegant and highlight the essence of the idea without any vagaries that might result from a specific situation. Secondly, it reduces the list decoding problem for specific instantiations of ideal-based codes, including the Reed-Solomon and AG-codes we studied in the previous chapter, to the efficient implementation of certain core algorithmic steps when applied to the specific context in question. To illustrate this point, after developing the general list decoding algorithm, we will apply it to a "new" situation, namely to list decoding Chinese Remainder codes (henceforth, CRT codes).

Recall that CRT codes, also called Redundant Residue codes, are the number-theoretic analog of Reed-Solomon codes. They are defined by picking n relatively prime integers $p_1 < p_2 < \cdots < p_n$. The messages m of the code are integers in the range $0 \leq m < \prod_{i=1}^{k} p_i$ for some k, $1 \leq k < n$. A message m is encoded by its residues modulo all the p_i's, i.e., $m \mapsto \langle m \bmod p_1, m \bmod p_2, \ldots, m \bmod p_n \rangle$. By the Chinese Remainder theorem, the message m is uniquely specified by *any* k of its residues modulo $p_1, p_2, \ldots, p_n$, and hence the above forms a redundant encoding of the message m. Indeed, this argument shows that two codewords (corresponding to encodings of m_1, m_2 with $m_1 \neq m_2$) differ in at least $(n - k + 1)$ positions. Hence, the distance of the code can be shown to equal $(n - k + 1)$.

There has been a lot of interest in decoding CRT codes [133, 134, 72, 31], but all these works fall short of list decoding CRT codes up to the Johnson radius, and in fact even fall short of decoding to half the minimum distance in general.[1]

Our general weighted list decoding algorithm for ideal-based codes, when applied to the case of CRT codes with a specific choice of weights (the exact choice ends up being a non-trivial guess), almost immediately gives an improvement to the prior results and decodes up to close to the Johnson bound. In fact, by choosing the parameters in the algorithm appropriately, the algorithm can decode up to the corresponding "weighted" Johnson bound (see Theorem 7.10) for *every* choice of weights. We also give a more efficient algorithm based on the Generalized Minimum Distance (GMD) decoding, to decode CRT codes up to half the minimum distance. GMD decoding was first discovered by Forney [60], who applied it to the soft decoding of Reed-Solomon codes.

We should mention here that by the very nature of the topic, the contents of this chapter are somewhat heavy on algebra. The results of this chapter put the algorithms from the previous chapter in a unified context and thus elucidate them better, but they are not necessary to the understanding of the results in the following chapters.

7.1.1 Overview

We begin in the next section by discussing the necessary preliminaries and terminology from commutative algebra concerning rings and ideals. These will be necessary for the definition of ideal-based codes and in the development of the list decoding algorithm for ideal-based codes. In Section 7.3 we give a formal definition of ideal-based codes and explain how Reed-Solomon codes, AG-codes and CRT codes can all be obtained as specific examples of ideal-based codes. In Section 7.4 we enlist some basic assumptions about the underlying rings and ideals, and prove the basic distance property of ideal-based codes. We add some further assumptions and develop a general

[1]This limitation is for the Hamming metric of measuring distance between the received word and the codewords. Indeed, the result of [72] provides a list decoding algorithm to decode up to the Johnson bound for a certain "natural" weighting of codeword positions of the CRT code.

weighted (soft) list decoding algorithm for ideal-based codes in Section 7.5. We then apply the results to the specific context of CRT codes in Section 7.6 and obtain a polynomial time soft decoding algorithm for CRT codes. We then apply it to specific interesting choices of weights to deduce results for CRT codes that decode up to the Johnson bound. Finally, in Section 7.7, we discuss the GMD decoding algorithm to decode CRT codes up to half the minimum distance.

7.2 Preliminaries

We quickly recall the basic algebraic definitions necessary for this chapter. If necessary, the reader can find further details and examples in any of the standard algebra texts (eg. [14]).

Rings: A *ring* is an algebraic structure $(R, +, \cdot)$ consisting of a set R together with two binary operations $(+, \cdot)$, normally called addition and multiplication respectively, which satisfy the following axioms:

- R is an abelian group under the operation $+$, with identity denoted by 0. This abelian group is denoted by R^+.
- R is closed under the operation $\cdot$, and $\forall\ x, y, z \in R$ we have
  - $x \cdot y = y \cdot x$ (Commutativity)
  - $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ (Associativity)
  - $x \cdot (y + z) = x \cdot y + x \cdot z$ (Distributive property of $\cdot$ over $+$)
- There exists an identity element for multiplication, denoted by 1, which satisfies $1 \cdot x = x \cdot 1 = x$ for every $x \in R$.

The terminology relating to rings is not completely standardized. In some texts, rings are defined without the requirement of the commutativity of multiplication and/or the existence of the multiplicative identity 1. In their terminology, the above definition will correspond to a subclass of rings called *commutative rings with identity.* We will work exclusively with commutative rings with identity, and hence we included these axioms in our definition of rings.

A ring is said to be an *integral domain* if $a \cdot b = 0$ implies that either $a = 0$ or $b = 0$ or both. All rings we deal with will be integral domains.

A *field* is a ring together with the additional property that for every non-zero element $x \in R$, there exists a unique inverse $x^{-1} \in R$ such that $x \cdot x^{-1} = x^{-1} \cdot x = 1$. In other words, a field is a ring whose non-zero elements form an abelian group under the multiplication operation.

Ideals:

An *ideal* I of a ring R is, by definition, a subset of R with the following properties:

(i) I is a subgroup of R^+.
(ii) If $a \in I$ and $r \in R$, then $r \cdot a \in I$.

In any ring, the set of multiples of a particular element a forms an ideal called the *principal ideal* generated by a, and is denoted (a). The set consisting of 0 alone is always an ideal called the *zero ideal*, and is denoted (0). Likewise, the whole ring R is also an ideal (generated by the element 1), called the *unit ideal*, and is denoted (1).

One can define sum, product and intersection operations on ideals as follows. The intersection of ideals I, J is simply their intersection as subsets of R. The sum of I, J is defined as $I + J = \{a + b : a \in I \text{ and } b \in J\}$. The product of I and J, denoted $I \cdot J$ (or, IJ), is defined to be all finite linear combinations of the form $a_1b_1 + a_2b_2 + \ldots + a_mb_m$ where each $a_i \in I$ and each $b_i \in J$. In other words, IJ is the smallest ideal which contains all elements of the form ab where $a \in I$ and $b \in J$. It is easily checked that if I, J are ideals of R, then so are $I \cap J$, $I + J$ and IJ. Note that for every pair of ideals I and J, $IJ \subseteq I \cap J$. For an ideal I, the power ideal I^n, for $n \geq 1$, is defined in the obvious way as: $I^n = I$ if $n = 1$, and $I^n = I \cdot I^{n-1}$ if $n > 1$.

Quotient rings: Let I be an ideal of a ring R. Consider the relation on R defined by $a \sim b$ if $a - b \in I$. It is easily checked that $\sim$ is an equivalence relation, and therefore it partitions R into equivalence classes. These equivalence classes are called the *cosets* of the ideal I. For $a \in R$, we denote by a/I the coset to which a belongs. The set of cosets of I themselves form a ring, denoted R/I, by inheriting the addition and multiplication operations from R. Specifically, one defines $(+, \cdot)$ for R/I by: $a/I + b/I \stackrel{\text{def}}{=} (a+b)/I$ and $a/I \cdot b/I \stackrel{\text{def}}{=} (a \cdot b)/I$. It is easy to check that these operations are well-defined and that R/I forms a ring under these operations. The ideals of R/I are in one-one correspondence with the ideals of R that contain I.

As an example, if $R = \mathbb{Z}$ and $I = (n)$ is the ideal generated by n, then $R/I = \mathbb{Z}/(n)$ is the ring of integers modulo n.

Prime and Maximal Ideals:

An ideal I of a ring R is a *prime ideal* if $a \cdot b \in I$ implies that at least one of a, b belongs to I. This is equivalent to the condition that the quotient ring R/I is an integral domain. The terminology "prime ideal" comes from the fact that if R is the ring of integers $\mathbb{Z}$ and $I = (m)$ is the ideal generated by an integer m, then I is a prime ideal if and only if m is a prime number.

An ideal I is a *maximal ideal* if $I \neq R$ and $I \not\subseteq J$ for any ideal $J \neq I, R$. An equivalent definition is that I is maximal iff the quotient ring R/I is a field.

Two ideals I, J of R are said to be *coprime* if $I + J = R$ (i.e., if $1 \in I + J$). The terminology comes from the fact that if the ring $R = \mathbb{Z}$ and $I = (m)$ and $J = (n)$ for integers m, n, then I, J are coprime ideals if and only if m, n are coprime integers. For coprime ideals I, J, we have $IJ = I \cap J$.[2]

[2]The easy proof of this fact goes as follows. Since $IJ \subseteq I \cap J$, we only have to prove that if $f \in I \cap J$ and $I + J = R$, then $f \in IJ$. Let $a \in I$ and $b \in J$ be such

7.3 Ideal-Based Codes

We now describe the basic principle that underlies the construction of several families of algebraic error-correcting codes, including Reed-Solomon codes, Algebraic-geometric codes, Chinese Remainder codes (and also Number field codes [127, 77]).

An algebraic error-correcting code is defined based on an underlying ring R (assume it is an integral domain), whose elements r come equipped with an appropriate notion of "size", denoted $\mathsf{size}(r)$. For example, for Reed-Solomon codes, the ring is the polynomial ring $\mathbb{F}[X]$ over a (large enough) finite field $\mathbb{F}$, and the "size" of $f \in \mathbb{F}[X]$ is related to its degree as a polynomial in X. Similarly, for the CRT code, the ring is $\mathbb{Z}$, and the "size" is the usual absolute value.

The messages of the code are the elements of the ring R whose size is at most a parameter B (this parameter governs the rate of the code). The encoding of a message $m \in R$ is given by

$$m \mapsto \mathsf{Enc}(m) = \langle m/I_1, m/I_2, \cdots, m/I_n \rangle \ ,$$

where I_j, $1 \le j \le n$ are n pairwise coprime ideals of R (we will assume that each of the quotient rings R/I_j is finite). Here m/I_j denotes the residue of m modulo the ideal I_j, and will belong to a finite alphabet whose size equals $|R/I_j|$. The formal definition follows:

Definition 7.1. *Let R be an integral domain; let $I_1, I_2, \ldots, I_n$ be n pairwise coprime ideals in R such that each R/I_j is finite, and let B be an arbitrary positive real. Further assume that there is a non-negative function* $\mathsf{size} : R \to \mathbb{R}^+$ *that associates a non-negative size with each element of the ring R. Then, the* "ideal-based" code $\mathbf{C}[R; I_1, I_2, \ldots, I_n; \mathsf{size}, B]$ *is defined to be the set of codewords*

$$\{\langle m/I_1, m/I_2, \ldots, m/I_n \rangle : m \in R \ \wedge \ \mathsf{size}(m) \le B\} \qquad (7.1)$$

7.3.1 Examples of Ideal-Based Codes

Chinese Remainder codes (CRT codes): Taking $R = \mathbb{Z}$; $I_j = (p_j)$, the principal ideal generated by the n mutually coprime integers $p_1, p_2, \ldots, p_n$; and $\mathsf{size}(m) = |m|$, the absolute value of m, we get the definition of CRT codes from the above definition.

Reed-Solomon codes: We get the Reed-Solomon code from the above definition by taking $R = \mathbb{F}_q[X]$ where $\mathbb{F}_q$ is a finite field with at least n elements (i.e. $q \ge n$), and $I_j = (X - \alpha_j)$ — the ideal generated by the polynomial

that $a + b = 1$. Now, $f = f \cdot (a + b) = f \cdot a + f \cdot b$. Now, clearly both $f \cdot a$ and $f \cdot b$ belong to IJ. Hence $f \in IJ$, as desired.

$(X - \alpha_j)$ — for $1 \leq j \leq n$, where $\alpha_1, \ldots, \alpha_n$ are *distinct* elements of $\mathbb{F}_q$. The notion of size is defined by $\mathsf{size}(p) = q^{\deg(p)}$. In other words, the messages are polynomials in $\mathbb{F}_q[X]$ of degree at most k, for some parameter k.

Algebraic-geometric codes: We now describe how the AG-codes from the previous chapter can also be obtained as a special case of ideal-based codes. Let $K/\mathbb{F}_q$ be a function field and P_0 be any fixed place of $K/\mathbb{F}_q$. For $i \geq 0$, let $\mathcal{L}(iP_0)$ be the set of functions in K which have no poles outside P_0 and have at most i poles at P_0. To specify an AG-code in the above ideal-theoretic language, we take the ring $R = \bigcup_{i \geq 0} \mathcal{L}(iP_0)$, and the ideal I_j to be a place P_j such that $P_1, P_2, \ldots, P_n$ and P_0 are all distinct places. (Recall from the previous chapter that a place P is by definition the unique maximal ideal of the ring $\mathcal{O}_P$ of regular functions at P, and since clearly $\mathcal{O}_P \subseteq R$ if $P \neq P_0$, such a place can also be viewed as an ideal of R.) The notion of size we use is related to the pole order at the place P_0; specifically we set $\mathsf{size}(x) = q^{-v_{P_0}(x)}$. Hence the set $\{x \in R : \mathsf{size}(x) \leq q^\alpha\}$ equals $\mathcal{L}(\alpha P_0)$, as with the usual definition of AG-codes.

7.4 Properties of Ideal-Based Codes

We now develop a set of axioms/assumptions about the ring R which will allow us to quantify the distance properties of the ideal-based code defined in Equation (7.1) above. We will later add a few further assumptions which will allow us to specify a unified list decoding algorithm for ideal-based codes and perform a quantitative analysis of its error-correction capabilities.

7.4.1 Axioms and Assumptions

Let R be an integral domain (a commutative ring where $a \cdot b = 0$ implies either $a = 0$ or $b = 0$). We assume the following properties for the ring R:

1. [Size of Elements]: There exists a function $\mathsf{size} : R \to \mathbb{R}$ such that for all $x, y \in R$:
   (S1) $\mathsf{size}(x) \geq 0$, and $\mathsf{size}(x) = 0 \Leftrightarrow x = 0$, and $\mathsf{size}(1) = \mathsf{size}(-1) = 1$.
   (S2) There exists an integer $1 \leq a \leq 2$ such that $\mathsf{size}(x + y) \leq a \cdot \max\{\mathsf{size}(x), \mathsf{size}(y)\}$; in other words, size satisfies a certain kind of "triangle" inequality.[3]
   (S3) $\mathsf{size}(xy) \leq \mathsf{size}(x)\mathsf{size}(y)$
2. [Size of Ideals]: There exists a function Δ that maps each non-zero ideal I of R to a positive real number $\Delta(I)$ such that
   (I1) If x is a non-zero element of an ideal I, then $\Delta(I) \leq \mathsf{size}(x)$.
   (I2) For every pair of coprime ideals I, J, $\Delta(IJ) \geq \Delta(I)\Delta(J)$.

[3]We point out that it is a well-known fact that if the stated inequality holds for some $a \leq 2$, then the "regular" archimedean triangle inequality $\mathsf{size}(x + y) \leq \mathsf{size}(x) + \mathsf{size}(y)$ also holds. Hence the name "triangle inequality" for this property.

The above axioms suffice to define a code and state the distance property that the code will satisfy.

7.4.2 Distance Property of Ideal-Based Codes

Lemma 7.2. *Assume that the assumptions (S1-S3) and (I1, I2) hold. Consider the code $\mathbf{C}[R; I_1, \ldots, I_n; \mathsf{size}, B]$ where the ring R satisfies the above assumptions (S1-S3) and (I1, I2). Assume further that the ideals I_j are ordered so that $\Delta(I_1) \le \Delta(I_2) \le \cdots \le \Delta(I_n)$. Then the minimum (Hamming) distance of this code is at least $(n - t + 1)$ where t is the smallest integer satisfying:*

$$\prod_{i=1}^{t} \Delta(I_i) > a \cdot B \ .$$

Proof: Let two distinct codewords in $\mathbf{C}$ corresponding to messages x, y agree on s residues, and let t be as in the statement of the lemma. We will show that $s < t$. Since $\mathsf{size}(x) \le B$ and $\mathsf{size}(y) \le B$, we have $\mathsf{size}(x - y) \le a \cdot B$ by axiom (S2). On the other hand, $(x - y)$ belongs to at least s ideals, and since the I_j's are pairwise coprime, $(x - y)$ belongs to the product of at least s ideals, say that of $I_{j_1}, \ldots, I_{j_s}$. Then, using axioms (I1) and (I2), we have

$$\mathsf{size}(x - y) \ge \Delta\Big(\prod_{i=1}^{s} I_{j_i}\Big) \ge \prod_{i=1}^{s} \Delta(I_{j_i}) \ge \prod_{i=1}^{s} \Delta(I_i) \ .$$

Together with $\mathsf{size}(x - y) \le aB$, this implies that

$$\prod_{i=1}^{s} \Delta(I_i) \le aB < \prod_{i=1}^{t} \Delta(I_i) \ ,$$

which shows that $s < t$ and completes the proof. □

To quantify the rate of these codes, we need a lower bound on the number of elements of R that have size at most B. We will later add axioms that guarantee this and further properties about the size of ideals that we will need to argue about the performance of our list decoding algorithm. We now turn to the specification of our list decoding algorithm.

7.5 List Decoding Ideal-Based Codes

We directly tackle the general "weighted" list decoding problem which is described below. We use the notation from the previous section and focus on list decoding an ideal-based code $\mathbf{C}[R; I_1. \ldots, I_n; \mathsf{size}, B]$ with message space $\mathcal{M} = \{x \in R : \mathsf{size}(x) \le B\}$.

Input: A vector $\mathbf{r} = \langle r_1, \ldots, r_n \rangle$ where $r_i \in R/I_i$ for $1 \le i \le n$, non-negative real weights $w_1, w_2, \ldots, w_n$, and agreement parameter W.

Required Output: A list of all $m \in \mathcal{M}$ such that $\sum_{i=1}^n w_i a_i > W$ where a_i is defined to be equal to 1 if $m/I_i = r_i$ and 0 otherwise.

To describe our list decoding algorithm, we assume the weights are some appropriate integers $z_1, z_2, \ldots, z_n$. Our algorithm will then output all codewords that satisfy a certain weighted condition in terms of the z_i's. The description of how to pick the z_i's to get useful results for specific input weights $w_1, w_2, \ldots, w_n$ will be described later when we apply the general algorithm to the case of the CRT code.

7.5.1 High Level Structure of the Decoding Algorithm

Before formally describing the algorithm, we first give some intuition on how it is designed based on the earlier Reed-Solomon decoding algorithm. Recall that our goal is to efficiently find a list of all $m \in R$ with $\mathsf{size}(m) \le B$ such that $\mathbf{C}(m)$ and the received word $\mathbf{r}$ have sufficient weighted agreement.

Following the Reed-Solomon and AG-codes case, the basic idea will be to "interpolate" a polynomial $c \in R[y]$ (based on the received word $\mathbf{r}$) with the property that every m for which $\mathbf{C}(m)$ has sufficient weighted agreement with the received word must be a *root* of the polynomial $c(y)$ (this polynomial c was called Q in the algorithms of the previous chapter). Then, by finding the roots of $c(y)$ and pruning out the spurious roots, we can recover all the codewords with sufficient weighted agreement with $\mathbf{r}$.

We are able to construct such a polynomial c by pursuing two objectives, which are in turn adaptations of the objectives from the case of decoding Reed-Solomon and AG-codes:

1. To ensure that the polynomial c has the property that for any $m \in R$ that satisfies $m/I_i = r_i$, we have $c(m) \in M_i$, for some suitable sequence of coprime ideals M_i, $i = 1, 2, \ldots, n$. This in turn implies that for any $m \in R$ we have $c(m) \in \prod_i M_i^{a_i}$, where $a_i = 1$ if $m/I_i = r_i$, and $a_i = 0$ otherwise.
2. To ensure that the coefficients c_j of $c(y) = \sum_{j=0}^{\ell} c_j y^j$ are small, i.e., each $\mathsf{size}(c_j)$ is sufficiently small. The aim of this step is to ensure that $\mathsf{size}(c(m))$ is small, say $\mathsf{size}(c(m)) < F$, for every m with $\mathsf{size}(m) \le B$.

By combining Objectives 1 and 2, we see that for any $m \in R$ with $\mathsf{size}(m) \le B$, $c(m)$ on the one hand has size less than F, and on the other hand belongs to $\prod_i M_i^{a_i}$. Hence if, $c(m) \ne 0$, we must have

$$F > \mathsf{size}(c(m)) \ge |R/M_i|^{a_i} \ , \tag{7.2}$$

where the second step uses axioms (I1), (I5). Therefore, if the boolean "agreement" vector $\mathbf{a} = \langle a_1, a_2, \ldots, a_n \rangle$ between $\mathbf{C}(m)$ and $\mathbf{r}$ satisfies the weighted condition

$$\sum_i a_i \log |R/M_i| > \log F \ ,$$

then Condition (7.2) cannot hold, and hence we must have $c(m) = 0$. Naturally, the performance of the algorithm depends on the choices of the ideals M_i and the parameter F. Our algorithm will pick $M_i = I_i^{z_i}$ (where z_i's are the input integer weights), and F to be a sufficiently large integer for which a polynomial $c \in R[y]$ meeting Objectives 1 and 2 exists. Precise details follow in the next section.

7.5.2 Formal Description of the Decoding Algorithm

Before describing the algorithm we need some auxiliary definitions and notation.

– Let $R[y]$ be the ring of polynomials in y with coefficients from R.
– For $1 \leq i \leq n$, let J_i be the ideal in $R[y]$ defined as $\{a(y)(y - r_i) + b(y) \cdot p | a, b \in R[y] \text{ and } p \in I_i\}$. It is readily checked that J_i is an ideal in $R[y]$ and further that if $m \in R$ satisfies $m/I_i = r_i$, then $c(m) \in I_i$ for every $c \in J_i$.

The algorithm is formally described in Figure 7.1. We stress that we do not know efficient implementations of all the steps in the algorithm for a general ideal-based code, but for specific codes like Reed-Solomon codes and AG-codes these do have efficient implementations. We will later show how with a moderate "slack" they can also be implemented in polynomial time for CRT codes.

(Weighted) List-decoding algorithm:

Input: A vector $\mathbf{r} = \langle r_1, \ldots, r_n \rangle$ where $r_i \in R/I_i$ for $1 \leq i \leq n$, non-negative integers $z_1, z_2, \ldots, z_n$ and parameter Z.

Required Output: A list of all $m \in \mathcal{M}$ such that $\sum_{i=1}^n z_i a_i > Z$ (where a_i is defined to be equal to 1 if $m/I_i = r_i$ and 0 otherwise).

1. Pick parameters ℓ, F appropriately.
2. Find a non-zero polynomial $c \in \prod_{i=1}^n J_i^{z_i}$ of degree at most ℓ with the property that $\mathsf{size}(c(m)) \leq F$ for every $m \in R$ with $\mathsf{size}(m) \leq B$.
3. Find all roots of c that lie in R and report those roots ζ such that $\mathsf{size}(\zeta) \leq B$ and the condition $\sum_{i=1}^n z_i a_i > Z$ is satisfied (where a_i is defined to be equal to 1 if $\zeta/I_i = r_i$ and 0 otherwise).

Fig. 7.1. A general list decoding algorithm for ideal-based codes

7.5.3 Further Assumptions on the Underlying Ring and Ideals

In order to analyze the error-correction capability of the algorithm above, we add some further axiomatic assumptions. The following assumptions need to apply only to the ideals $I_1, \ldots, I_n$ specified in the construction of the code.

(I3) For each i, we have $\Delta(I_i^k) \geq \Delta(I_i)^k$ for all positive integers k.
(I4) For each i, we have that $|R/I_i^k| \leq |R/I_i|^k$ for all positive integers k.
(I5) For each i, we have that $\Delta(I_i) \geq |R/I_i|$.

We also add the following assumption on the number of elements in R with bounded size. This is not only critical in order to quantify the rate of the code, but is also used in the analysis of the list decoding algorithm.

(S4) There exists a positive constant α depending only on the ring R such that for all positive integers F, the number of elements x of R with $\mathsf{size}(x) < F$ is at least αF.

Note that for the CRT code ($R = \mathbb{Z}$), we have $\alpha \simeq 2$, while for Reed-Solomon and AG-codes we have $\alpha = 1$.

7.5.4 Analysis of the List Decoding Algorithm

We now specify the parameter choices in the above algorithm for it to output all the "relevant" codewords, and determine the exact condition (specifically the value of the agreement parameter Z) for which the algorithm will succeed in finding all codewords that satisfy $\sum_i a_i z_i > Z$.

The following sequence of lemmas will be used in the analysis.

Lemma 7.3. *If $c \in J_i^{z_i}$, then for every $m \in R$ with $m/I_i = r_i$, we have $c(m) \in I_i^{z_i}$.*

Proof: Every $c \in J_i^{z_i}$ is the sum of a finite number of terms each of the form

$$\prod_{s=1}^{z_i} (a_s(y)(y - r_i) + b_s(y)p_s) \ ,$$

where $a_s, b_s \in R[y]$ and $p_s \in I_i$ for $1 \leq s \leq z_i$. Substituting $y = m$ where $m/I_i = r_i$, we have each of the s terms in the product belongs to I_i, and hence the entire term belongs to $I_i^{z_i}$. Since this is true for each term of $c(m)$, it follows that $c(m)$ itself is in $I_i^{z_i}$, as desired. $\square$

Lemma 7.4. *For each i, $1 \leq i \leq n$, $|R[y]/J_i^{z_i}| \leq |R/I_i|^{\binom{z_i+1}{2}}$.*

Proof: We need to estimate the number of different residues that polynomials in $R[y]$ can have modulo $J_i^{z_i}$. Let $c \in R[y]$ be any polynomial. Expand $c(y)$ in terms of sums of powers of $(y - r_i)$ (i.e., use the change of variable $y' = y - r_i$, and write down $c(y' + r_i)$). Since $(y - r_i)^m \in J_i^{z_i}$ for $m \geq z_i$, to compute the residue of c modulo $J_i^{z_i}$, we can ignore all terms of degree at least z_i. Thus we can assume that

$$c/J_i^{z_i} = \sum_{s=0}^{z_i - 1} \alpha_s (y - r_i)^s \ , \tag{7.3}$$

for suitable coefficients α_s. Now since $\alpha_s(y - r_i)^s \in J_i^{z_i}$ if $\alpha_s \in I_i^{z_i - s}$, it follows that we may assume that α_s is reduced modulo $I_i^{z_i - s}$ in the above, or in other words that $\alpha_s \in R/I_i^{z_i - s}$. Hence the number of possibilities for α_s is at most $|R/I_i^{z_i - s}| \leq |R/I_i|^{z_i - s}$ using assumption (I4). Combining with Equation (7.3), we obtain that the total number of possible residues modulo $J_i^{z_i}$, in other words $|R[y]/J_i^{z_i}|$, is at most

$$\prod_{s=0}^{z_i - 1} |R/I_i|^{z_i - s} = |R/I_i|^{\binom{z_i + 1}{2}} \ ,$$

as claimed. □

Corollary 7.5. *We have*

$$\left|R[y]/\prod_{i=1}^{n} J_i^{z_i}\right| \leq \prod_{i=1}^{n} |R/I_i|^{\binom{z_i + 1}{2}} \ .$$

Proof: First of all, note that since the I_i's are all coprime (i.e., $I_i + I_j = R$ for $i \neq j$), we also have the J_i's to be pairwise coprime. This in turn implies that the ideals $J_i^{z_i}$ are all pairwise coprime. Therefore,

$$\left|R[y]/\prod_{i=1}^{n} J_i^{z_i}\right| = \prod_{i=1}^{n} |R[y]/J_i^{z_i}| \leq \prod_{i=1}^{n} |R/I_i|^{\binom{z_i + 1}{2}}$$

where the second step follows from Lemma 7.4. □

Before stating the next lemma, we need the following notation. Let b_k be the least integer such that for all $x_1, x_2, \ldots x_k \in R$, we have $\mathsf{size}(x_1 + x_2 + \cdots + x_k) \leq b_k \max\{\mathsf{size}(x_1), \ldots, \mathsf{size}(x_k)\}$. We clearly have $b_1 = 1$, $b_2 \leq a$ (recall that a was the parameter used in the "triangle" inequality (S2)). Of course if $a = 1$, then each $b_k = 1$, and one can show that as long as $a \leq 2$, $b_k \leq k$. (This follows because it is a standard exercise to show that $a \leq 2$ implies size satisfies the "familiar" triangle inequality $\mathsf{size}(x + y) \leq \mathsf{size}(x) + \mathsf{size}(y)$, from which of course $b_k \leq k$ follows easily.) Thus, for Reed-Solomon and algebraic-geometric codes, we have $b_k = 1$ for all $k \geq 1$, while for CRT codes we have $b_k = k$ for all $k \geq 1$.

Lemma 7.6. *For positive integers B, F', the number of polynomials $c \in R[y]$ of degree at most ℓ with the property that $\mathsf{size}(c(m)) < F'$ whenever $\mathsf{size}(m) \le B$ is at least*

$$\left(\frac{\alpha F'}{b_{\ell+1} B^{\ell/2}}\right)^{\ell+1} .$$

Proof: Consider polynomial $c(y) = c_0 + c_1 y + \ldots + c_\ell y^\ell$ where each $c_j \in R$ for $0 \le j \le \ell$. We will pick coefficients so that for any m with $\mathsf{size}(m) \le B$, we will have $\mathsf{size}(c_j m^j) < F'/b_{\ell+1}$. Note that this will imply that $\mathsf{size}(c(m)) < F'$ whenever $\mathsf{size}(m) \le B$. This requirement on c_j will be satisfied if $\mathsf{size}(c_j) < F' \cdot B^{-j}/b_{\ell+1}$ (here we are using (S3)). Also, by assumption (S4) there at least $\frac{\alpha F'}{B^j b_{\ell+1}}$ such choices for c_j. Hence the total number of polynomials $c \in R[y]$ with the required property is at least

$$\left(\frac{\alpha F'}{b_{\ell+1}}\right)^{\ell+1} \cdot \prod_{j=0}^{\ell} B^{-j} = \left(\frac{\alpha F'}{b_{\ell+1} B^{\ell/2}}\right)^{\ell+1} ,$$

as claimed. □

We are now ready to prove that for suitable choices of ℓ, F a non-zero polynomial with the desired properties as in Step 2 of the list decoding algorithm exists in $\prod_{i=1}^n J_i^{z_i}$.

Lemma 7.7. *Let ℓ, B, F be positive integers which satisfy the following condition:*

$$F \ge B^{\ell/2} \cdot \left(\frac{a \cdot b_{\ell+1}}{\alpha}\right)\left(\prod_{i=1}^{n} |R/I_i|^{\binom{z_i+1}{2}}\right)^{1/(\ell+1)} . \tag{7.4}$$

Then there exists a non-zero $c \in \prod_{i=1}^n J_i^{z_i}$ which satisfies the property that $\mathsf{size}(c(m)) < F$ for every $m \in R$ with $\mathsf{size}(m) \le B$.

Proof: The proof follows from Corollary 7.5, Lemma 7.6, and the pigeonhole principle. Specifically, if Condition (7.4) is satisfied, then we have

$$\left(\frac{\alpha \cdot F/a}{b_{\ell+1} B^{\ell/2}}\right)^{\ell+1} > \prod_{i=1}^{n} |R/I_i|^{\binom{z_i+1}{2}} ,$$

and thus the number of degree ℓ polynomials $c \in R[y]$ with $\mathsf{size}(c(m)) < F/a$ whenever $\mathsf{size}(m) \le B$ is greater than the total number of residues of polynomials modulo $\prod_{i=1}^n J_i^{z_i}$. Hence, by the pigeonhole principle there must exist two distinct polynomials $c_1, c_2 \in R[y]$ of degree at most ℓ such that $(c_1 - c_2) \in \prod_{i=1}^n J_i^{z_i}$. Since $\mathsf{size}(c_1(m)) < F/a$ and $\mathsf{size}(c_2(m)) < F/a$ for every m with $\mathsf{size}(m) \le B$, we have by assumption (S2) that $\mathsf{size}((c_1 - c_2)(m)) < F$ for each such m. Thus the claim of the lemma is satisfied with $c \overset{\text{def}}{=} (c_1 - c_2)$. □

Lemma 7.8. *Let $c \in \prod_{i=1}^{z_i} J_i^{z_i}$ be such that $\mathsf{size}(c(x)) < F$ for every $x \in R$ with $\mathsf{size}(x) \le B$. Then, any $m \in R$ that satisfies $\mathsf{size}(m) \le B$ and*

$$\prod_{i : m/I_i = r_i} |R/I_i|^{z_i} \ge F \tag{7.5}$$

must be a root of c, i.e., must satisfy $c(m) = 0$.

Proof: Let m be any such element of R. Since $\mathsf{size}(m) \le B$, by the property of c, we have

$$\mathsf{size}(c(m)) < F \; . \tag{7.6}$$

Since $c \in J_i^{z_i}$ for each i, $1 \le i \le n$, by Lemma 7.3, we have $c(m) \in I_i^{z_i}$ for each i such that $m/I_i = r_i$. Hence

$$c(m) \in \prod_{i : m/I_i = r_i} I_i^{z_i} \; .$$

Now, using assumptions (I1), (I2), (I3) and (I5), we have that if $c(m) \ne 0$, then

$$\mathsf{size}(c(m)) \ge \prod_{i : m/I_i = r_i} \Delta(I_i^{z_i}) \ge \prod_{i : m/I_i = r_i} \Delta(I_i)^{z_i} \ge \prod_{i : m/I_i = r_i} |R/I_i|^{z_i} \; . \tag{7.7}$$

From (7.7) and (7.6) it follows that if Condition (7.5) is satisfied, we have a contradiction and therefore must have $c(m) = 0$, as desired. □

7.5.5 Performance of the List Decoding Algorithm

We are now ready to state and prove the main result of this section on the performance of our list decoding algorithm from Section 7.5.1.

Theorem 7.9. *For every set of non-negative integers $z_1, z_2, \ldots, z_n$, for a suitable choice of parameters ℓ, F, the list decoding algorithm on receiving as input a word $\mathbf{r} = \langle r_1, \ldots, r_n \rangle$ with $r_i \in R/I_i$, can find a list of size at most ℓ which includes all messages $m \in R$ with $\mathsf{size}(m) \le B$ that satisfy*

$$\sum_{i=1}^{n} a_i z_i \log q_i > \frac{1}{\ell+1} \sum_{i=1}^{n} \binom{z_i+1}{2} \log q_i + \log(a/\alpha) + \tag{7.8}$$
$$+ \frac{\ell}{2} \log B + \log b_{\ell+1} \; .$$

where we use the shorthand $q_i = |R/I_i|$, and a_i is an indicator variable defined to be 1 if $m/I_i = r_i$ and 0 otherwise.

Proof: The proof follows easily from the statements of Lemma 7.7 and Lemma 7.8. Indeed, one can choose

$$F = \left\lceil B^{\ell/2} \cdot \Big(\frac{a \cdot b_{\ell+1}}{\alpha}\Big)\Big(\prod_{i=1}^{n} |R/I_i|^{\binom{z_i+1}{2}}\Big)^{1/(\ell+1)} \right\rceil , \tag{7.9}$$

and for this choice of F, the algorithm can find a non-zero $c \in \prod_{i=1}^{n} J_i^{z_i}$ with $\mathsf{size}(c(m)) < F$ whenever $\mathsf{size}(m) \le B$ (since by Lemma 7.7 such a c exists). By Lemma 7.8, the algorithm will output a list of all $m \in R$ with $\mathsf{size}(m) \le B$ such that

$$\prod_{i=1}^{n} q_i^{a_i z_i} \ge F .$$

Note that the number of solutions the algorithm outputs is at most ℓ, since it only outputs a subset of the roots of a degree ℓ polynomial over the integers. Also, since both the terms on the right and left hand sides of the above condition are integers, taking logarithms we note that the above condition is implied by the decoding Condition (7.8) stated in the theorem. □

Remark: There is a natural notion of an "approximate solution" for Step 2 in the list decoding algorithm. We know that for F defined as in Equation (7.9), there exists a non-zero polynomial $c \in \prod_{i=1}^{n} J_i^{z_i}$ that satisfies $\mathsf{size}(c(m)) < F$ whenever $\mathsf{size}(m) \le B$. It is conceivable that, in certain contexts, finding such a c for this "optimum" choice of F might be difficult to accomplish efficiently. In such a case, suppose the algorithm only manages to find a non-zero polynomial $c \in \prod_{i=1}^{n} J_i^{z_i}$ with a factor β slack in the size guarantee, namely a polynomial c such that $\mathsf{size}(c(m)) < F'$ for every m with $\mathsf{size}(m) \le B$, where $F' = \beta F$. Then, it is easy to check that such an algorithm can decode under a condition similar to (7.8) with an additional $\log \beta$ term on the right hand side. We will make use of this fact when considering an efficient implementation of the decoding algorithm for the specific context of decoding CRT codes in Section 7.6.2.

7.5.6 Obtaining Algorithms for Reed-Solomon and AG-codes

We now briefly indicate how the Reed-Solomon and AG-code list decoding algorithms from the previous chapter can be obtained from Theorem 7.9 above. Note that the list decoding algorithm of Figure 7.1 is really only a general algorithmic schema, and one needs to implement each of its steps efficiently in order to apply it and get polynomial time list decoding algorithms for specific families of ideal-based codes. Hence, our aim below is only to show that this algorithm gives (more or less) the same parameters as the specific polynomial time algorithms discussed in the previous chapter.

For Reed-Solomon codes over a field $\mathbb{F}_q$, each $q_i = q$, $\alpha = 1$, and $a = 1$ in assumption (S2) (and hence $b_{\ell+1} = 1$ as well). If the code is defined by evaluations of polynomials of degree at most k, then since $\mathsf{size}(p) = q^{\deg(p)}$, we have $B = q^k$. Substituting these we get the algorithm finds all codewords that have "z-weighted" agreement with $\mathbf{r}$ more than

$$\frac{1}{\ell+1}\sum_{i=1}^{n}\binom{z_i+1}{2} + \frac{\ell}{2}k$$

which for large ℓ, is approximately $\sqrt{k\sum_i z_i(z_i+1)}$ which approaches the performance of the soft decoding algorithm for Reed-Solomon codes from Section 6.2.10 (by taking the z_i's to be large multiples of the weights w_i).

For AG-codes over $\mathbb{F}_q$, once again each $q_i = q$, $a = 1$ and $b_{\ell+1} = 1$. If the underlying function field has genus g, then $\alpha = q^{-g}$. Also, $B = q^{\alpha^*}$ if the message space of the AG-code is $\mathcal{L}(\alpha^* P_0)$. Hence Theorem 7.9 implies that one can find all codewords that have "z-weighted" agreement with $\mathbf{r}$ more than

$$\frac{1}{\ell+1}\sum_{i=1}^{n}\binom{z_i+1}{2} + \frac{\ell}{2}\alpha^* + g$$

which for large ℓ again approaches the performance of the soft decoding algorithms for AG-codes from Chapter 6.

We already knew the decoding algorithms for Reed-Solomon codes and AG-codes from the previous chapter, but the above indicates the generality of our decoding algorithm for ideal-based codes. In the next section, we will exploit the generality of our algorithm from Figure 7.1 to devise a list decoding algorithm for Chinese Remainder (CRT) codes. Indeed, it was the design of a good decoding algorithm for CRT codes that motivated us to dig deeper into the algebra underlying the list decoding algorithms and unveil the unified decoding algorithm for ideal-based codes described in Figure 7.1.

7.6 Decoding Algorithms for CRT Codes

In this section, we discuss efficient decoding algorithms for the CRT code. Recall that a CRT code is specified by a sequence $p_1 < p_2 < \ldots < p_n$ of relatively prime integers and an integer $k < n$. Let $K = \prod_{i=1}^{k} p_i$; $N = \prod_{i=1}^{n} p_i$. For easy reference, we say such a CRT codes as being specified by parameters $(p_1, p_2, \ldots, p_n; K)$. We associate to each integer $m \in \{0, 1, \ldots, K-1\}$ the codeword $\langle m_1, m_2, \ldots, m_n\rangle$, where $m_i = m \bmod p_i$. We will abuse notation and refer to both this sequence and m as a *codeword*. We consider a *received word* to be a sequence $\mathbf{r} = \langle r_1, r_2, \ldots, r_n\rangle$ of integers with $0 \le r_i < p_i$ for each $i \in [n]$. For a given sequence of weights $\mathbf{w} = \langle w_1, \ldots, w_n\rangle$, the $\mathbf{w}$-*weighted agreement* (or simply *weighted agreement*, when the weighting we are referring to is clear) between a codeword $m < K$ and a received word $\mathbf{r}$ is the defined to be the quantity $\sum_i a_i w_i$, where $a_i = 1$ if $m_i = r_i$, and $a_i = 0$ otherwise.

Our goal in this section is to efficiently find a list of all non-negative integers $m < K$ such that the encoding of m and the received word $\mathbf{r}$ have sufficient weighted agreement. We note that a simple transformation makes it equivalent for us to find integers m where $|m| \leq K/2$, whose encodings have sufficient agreement with $\mathbf{r}$. It is this version of the problem that we focus on for describing our decoding algorithms.

In this section, we present two efficient decoding algorithms for the CRT code. In the first (which is our main) decoding algorithm, the goal is to efficiently find a list of *all* codewords m such that m and the received word $\mathbf{r}$ have sufficient weighted agreement. In particular, we are able to give an efficient list decoding algorithm which outputs all m with $|m| \leq K/2$ such that $m \bmod p_i = r_i$ for at least $\sqrt{k(n+\varepsilon)}$ values of i (for any ε, with the running time of the algorithm depending polynomially on $1/\varepsilon$). Thus, we are able to efficiently list decode the CRT code up to (essentially) the Johnson bound on list decoding radius (from Corollary 3.3 with distance $d = n-k+1$). This improves the earlier works of [72, 31] which could only find the codewords which agreed with the received word in at least $\Omega(\sqrt{kn \log p_n / \log p_1})$ positions. Our algorithm is obtained by efficient implementations of the steps of the general decoding algorithm of Figure 7.1, specialized for the case of the CRT code. This gives a general weighted decoding algorithm which successfully list decodes as long as a certain "weighted" condition is satisfied. The above claimed bound is then obtained by an appropriate choice of weights in the weighted algorithm (the exact setting of weights turns out to be a non-trivial guess).

For any sequence of positive weights $\boldsymbol{\beta}$, our second decoding algorithm efficiently (in near-quadratic time) recovers the *unique* codeword m with highest $\boldsymbol{\beta}$-weighted agreement with a received word r, as long as there is a codeword whose $\boldsymbol{\beta}$-weighted distance from r is less than half the $\boldsymbol{\beta}$-weighted minimum distance of the code. This is accomplished by adapting the GMD decoding algorithm due to Forney, introduced for Reed-Solomon codes in [60], to CRT codes in Section 7.7. Note that in particular this result gives the *first* polynomial time algorithm to correct up to $(n-k)/2$ errors (i.e., decode up to half the minimum distance) for the CRT code. In view of our more powerful list decoding algorithm, the main role of this result can be viewed as highlighting the role of GMD decoding in the task of decoding the CRT code, plus achieving a simpler, faster algorithm for unique decoding of CRT codes.

7.6.1 Combinatorial Bounds on List Decoding

Before delving into the decoding algorithms, we first state a generalized Johnson-type bound which specifies a fairly general condition under which list decoding using "small" lists can be performed. This result will indicate the kind of performance that we can hope for from our list decoding algorithms for the CRT codes, since in order to efficiently output a list of codewords as possible answers, we need an a priori guarantee that the list size will be small.

The result below gives a generalization of the weighted Johnson bound from Chapter 3 (specifically the result of Corollary 3.7) to the case when the various codeword positions have different contributions towards the distance of the code.

Theorem 7.10. *Let C be a code of blocklength n with the i'th symbol coming from an alphabet of size q_i, for $1 \le i \le n$. Let the distance D_α of the code be measured according to a weighting vector $\boldsymbol{\alpha} = \langle \alpha_1, \ldots, \alpha_n \rangle$. In other words, for any two distinct codewords $c_1, c_2 \in C$, we have $\sum_{i: c_{1i} \ne c_{2i}} \alpha_i \ge D_\alpha$ (assume each $\alpha_i \ge 1$ without loss of generality). For a weighting vector $\boldsymbol{\beta} = \langle \beta_1, \ldots, \beta_n \rangle$ and a received word $\mathbf{r} = \langle r_1, \ldots, r_n \rangle \in [q_1] \times \cdots \times [q_n]$, define the set $S_{\boldsymbol{\beta}}(\mathbf{r}, W)$ to consist of all strings z (in the space $[q_1] \times [q_2] \times \cdots \times [q_n]$) with weighted $\boldsymbol{\beta}$-weighted agreement with $\mathbf{r}$ at least W, i.e., which satisfy $\sum_{i: r_i = z_i} \beta_i \ge W$. Then, for all $\mathbf{r}$, the set $S_{\boldsymbol{\beta}}(\mathbf{r}, W_\beta)$ has at most $\left(2 \sum_{i=1}^n q_i\right)$ codewords from C, provided that:*

$$W_\beta \ge \left[\Big(\sum_{i=1}^n \alpha_i - D_\alpha \Big) \sum_{i=1}^n \frac{\beta_i^2}{\alpha_i} \right]^{1/2} , \tag{7.10}$$

and has at most L codewords from C, provided that

$$W_\beta \ge \left[\Big(\sum_{i=1}^n \alpha_i - D_\alpha + \frac{D_\alpha}{L} \Big) \sum_{i=1}^n \frac{\beta_i^2}{\alpha_i} \right]^{1/2} . \tag{7.11}$$

Remark: A more complicated and stronger bound than the above theorem can be proved by taking into account the size of the alphabets q_i's (akin to the weighted Johnson bound of Theorem 3.6 that took into account the alphabet size). This is, however, not very important for us since we want to use the above bound to only informally indicate the "near-tightness" of the error-correction performance of our list decoding algorithms for the CRT code, and the above bound suffices for this purpose. Moreover, for the CRT code the q_i's are typically large primes, and for large alphabets the difference between the stronger bound and the above bound becomes negligible.

Proof of Theorem 7.10: The proof follows along the lines of Theorem 3.1. Let $\boldsymbol{\beta}$ be a weighting vector and W an agreement parameter. Let $\mathbf{c_1}, \ldots, \mathbf{c_m}$ be *all* the codewords in $S_{\boldsymbol{\beta}}(\mathbf{r}, W_\beta)$, where $\mathbf{r} \in [q_1] \times \cdots \times [q_n]$ is the "received word".

We will embed elements of $[q_1] \times \cdots \times [q_n]$ as vectors in $\mathbb{R}^Q$ where $Q = \sum_{i=1}^n q_i$, with the i'th block being a vector of length q_i corresponding to the i'th symbol. For the received word $\mathbf{r}$, we will let the i'th block (which is of length q_i) have a value of $\beta_i / \sqrt{\alpha_i}$ at position number r_i, and 0's elsewhere. By abuse of notation, we denote the resulting vector in $\mathbb{R}^Q$ also by $\mathbf{r}$. For each of the codewords $\mathbf{c_j}$, $1 \le j \le m$, we will let the i'th block have a value of $\sqrt{\alpha_i}$ at position number $c_{j,i}$ (i.e., the i'th symbol of the codeword $\mathbf{c_j}$), and 0's elsewhere. Once again, since it is convenient to do so, we denote the resulting vectors in $\mathbb{R}^Q$ also by $\mathbf{c_1}, \ldots, \mathbf{c_m}$.

It is easy to see from the above choices that $\langle \mathbf{c_j}, \mathbf{r} \rangle$ equals the $\boldsymbol{\beta}$-weighted agreement between $\mathbf{c_j}$ and $\mathbf{r}$ (i.e., $\langle \mathbf{c_j}, \mathbf{r} \rangle = \sum_{i:c_{j,i}=r_i} \beta_i$), and that $\langle \mathbf{c_j}, \mathbf{c_k} \rangle$ equals the $\boldsymbol{\alpha}$-weighted agreement between $\mathbf{c_j}$ and $\mathbf{c_k}$. Therefore, we have, for every $1 \le j < k \le m$,

$$\langle \mathbf{c_j}, \mathbf{r} \rangle \ge W_\beta \tag{7.12}$$

$$\langle \mathbf{c_j}, \mathbf{c_k} \rangle \le A_\alpha \overset{\text{def}}{=} \Big(\sum_{i=1}^{n} \alpha_i - D_\alpha \Big) \tag{7.13}$$

The idea now is to pick a suitable parameter $\gamma > 0$ such that the pairwise dot products between the vectors $(\mathbf{c_j} - \gamma \mathbf{r})$ are all non-positive. This is similar to the idea used in the proof of Theorem 3.1, and the details are in fact simpler in this case (since the conditions under which we want to show a small list size do not depend on the alphabet sizes q_i).

Equations (7.12) and (7.13) together with the facts that $\langle \mathbf{r}, \mathbf{r} \rangle = \sum_{i=1}^{n} \beta_i^2/\alpha_i$, implies, for $j \ne k$,

$$\langle \mathbf{c_j} - \gamma \mathbf{r}, \mathbf{c_k} - \gamma \mathbf{r} \rangle \le A_\alpha - 2\gamma W_\beta + \gamma^2 \sum_{i=1}^{n} \frac{\beta_i^2}{\alpha_i} \ . \tag{7.14}$$

We will therefore have $\langle \mathbf{c_j} - \gamma \mathbf{r}, \mathbf{c_k} - \gamma \mathbf{r} \rangle \le 0$, provided

$$W_\beta \ge \frac{\gamma}{2} \sum_{i=1}^{n} \frac{\beta_i^2}{\alpha_i} + \frac{A_\alpha}{2\gamma} \ . \tag{7.15}$$

The right hand side is minimized for $\gamma = (A_\alpha)^{1/2} \cdot \big(\sum_i \frac{\beta_i^2}{\alpha_i} \big)^{-1/2}$, and for this choice of γ, Condition (7.15) becomes

$$W_\beta \ge \left[A_\alpha \sum_{i=1}^{n} \frac{\beta_i^2}{\alpha_i} \right]^{1/2} \ . \tag{7.16}$$

Now appealing to the geometric Lemma 3.4, Part (i), we get that the number of codewords m is at most $2Q$ (since the pairwise dot products of the Q-dimensional real vectors $(\mathbf{c_j} - \gamma \mathbf{r})$ are all non-positive). Thus, the number of codewords which lie in $S_{\boldsymbol{\beta}}(\mathbf{r}, W_\beta)$ if Condition (7.16) holds is at most $2Q$, which proves the first assertion of the theorem. The second assertion also follows similarly, by picking the parameter γ such that the pairwise dot products of the unit vectors along $(\mathbf{c_j} - \gamma \mathbf{r})$ is at most $-1/(L-1)$, and then appealing to geometric Lemma 3.5. We omit the details. □

We now apply the result of Theorem 7.10 to specific choices of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ for the CRT code. For a CRT code we have, in the ideal-based language, $R = \mathbb{Z}$ and

$I_i = (p_i)$ for relatively prime integers $p_1 < p_2 < \cdots < p_n$. Hence $q_i = p_i$ for $1 \le i \le n$. Furthermore the "size" of the ideals satisfy $\Delta(I_i) = |R/I_i| = p_i$. If we consider messages to be integers m in the range $-K/2 < m \le K/2$ where $K = \prod_{i=1}^{k} p_i$, then it is easy to prove using ideas in Lemma 7.2 that the $\boldsymbol{\alpha}$-weighted distance of the CRT code is

- at least $(n-k+1)$ for the all-ones weight vector (the case when $\alpha_i = 1$ for every i), and
- at least $\log(N/K)$ for case when $\alpha_i = \log p_i$.

Using these in the bound of Theorem 7.10 we get (roughly) the following conditions under which CRT list decoding is feasible (combinatorially):

$$\sum_i a_i \log p_i > \sqrt{(\log K + \varepsilon)\log N} \tag{7.17}$$

$$\sum_i a_i > \sqrt{(k+\varepsilon)n} \tag{7.18}$$

$$\sum_i a_i \beta_i > \left((\log K + \varepsilon)\sum_i \frac{\beta_i^2}{\log p_i}\right)^{1/2} \tag{7.19}$$

(Note that the third condition above implies the first with the choice of weights $\beta_i = \log p_i$.) In fact, the algorithm of Goldreich, Ron, and Sudan [72] could list decode under the first condition (7.17). In some sense the case $\alpha_i = \beta_i$ is the most natural one for the CRT code. However, neither the algorithm of [72] nor the improvement due to Boneh [31], could work as well for other weightings (including the case $\beta_i = \alpha_i = 1$ from the second condition above). In the next section, we apply our general decoding algorithm for ideal-based codes to the case of CRT codes to remedy this defect of earlier algorithms, and give a weighted list decoding algorithm which, by appropriately choosing the weights, can decode under each of the above conditions.

7.6.2 Weighted List Decoding Algorithm

We now apply Theorem 7.9 to the case of CRT codes and get the following.

Theorem 7.11. *For a CRT code with parameters $(p_1, p_2, \ldots, p_n; K)$, given a received word $\mathbf{r} = (r_1, r_2, \ldots, r_n)$ with $0 \le r_i < p_i$, and non-negative integers ℓ and z_i for $1 \le i \le n$, we can find in time polynomial in $n, \ell, \sum_i \log p_i$ and $\sum_i z_i$, a list of size at most ℓ which includes all codewords m that satisfy*

$$\sum_{i=1}^{n} a_i z_i \log p_i > \log(\ell+1) + \frac{\ell}{2}\log K + \frac{1}{\ell+1}\sum_{i=1}^{n}\binom{z_i+1}{2}\log p_i\ , \tag{7.20}$$

where as usual we define $a_i = 1$ if $m_i = r_i$ and $a_i = 0$ otherwise.

Proof: We will show that the above condition implies the condition under which the decoding algorithm of Figure 7.1, when applied to the CRT case, successfully list decodes the received word. It will then remain to argue that for the CRT code each of the steps of the algorithm can be implemented in polynomial time.

For the CRT code, we have $|R/I_i| = |\mathbb{Z}/(p_i)| = p_i$ for $1 \le i \le n$, $a = 2$, and $b_{\ell+1} = \ell+1$ (since the integers satisfy the "familiar" archimedean triangle inequality). Furthermore, since there are $(2F-1)$ integers of absolute value less than F for any positive integer F, we can assume that $\alpha \ge (2-\gamma)$ for some small $\gamma > 0$ (in fact we can take $\gamma = o(1)$ in the parameters involved). Also since the messages are integers of absolute value at most $K/2$, we have $B = K/2$. Plugging these parameters into the general bound of Equation (7.8) we get the condition

$$\sum_{i=1}^{n} a_i z_i \log p_i > \log(\ell+1) + \frac{\ell}{2}\log(K/2) + \qquad (7.21)$$
$$+\frac{1}{\ell+1}\sum_{i=1}^{n}\binom{z_i+1}{2}\log p_i + \log(2/(2-\gamma)) .$$

Note that in fact the above condition poses a weaker requirement than that of Condition 7.20 stated in the theorem, since we have an $\frac{\ell}{2}\log(K/2)$ term on the right hand side instead of $\frac{\ell}{2}\log K$ as stated in the theorem — the additional $\log(2/(2-\gamma))$ term is of course negligible in comparison. Hence the general algorithm can also decode under the condition stated in the theorem. The reason for the slack in Condition (7.20) is that we now also want a polynomial time implementation of its various steps, and hence can only find an "approximation" to the best polynomial $c \in \mathbb{Z}[y]$ in Step 2 of the algorithm. As discussed in the remark following Theorem 7.9, this necessitates a slight weakening of the error-correction performance. We discuss the details next.

The two non-trivial steps in the algorithm of Figure 7.1, when applied to the CRT code, are (i) finding a non-zero degree ℓ polynomial c with integer coefficients in the ideal $\prod_i J_i^{z_i}$ such that $|c(m)| < F$ for all m with $|m| \le K/2$, and (ii) finding the roots of c and looking for candidate codewords among its roots. The second task can be done in polynomial time using, for instance, the algorithm for factoring polynomials with integer coefficients due to Lenstra, Lenstra and Lovász [126].[4] For the first task, Lemma 7.7 applied to the CRT case implies that for

$$F = \lceil F^* \rceil \quad \text{where} \quad F^* \stackrel{\text{def}}{=} (\ell+1)(K/2)^{\ell/2}\Big(\prod_i p_i^{\binom{z_i+1}{2}}\Big)^{1/(\ell+1)} ,$$

[4]Since the root finding task is easier than a general factorization task, there are faster ways to solve the root finding problem. A brief discussion about this appears in [72].

there exists a non-zero $c \in \prod_i J_i^{z_i}$ with $|c(m)| < F$ whenever $|m| \le K/2$ (in fact the coefficients c_j of c will satisfy $|c_j| < \frac{F}{(\ell+1)(K/2)^j}$ for $0 \le j \le \ell$). We will now prove that for F' which $2^{\ell/2}$ times larger than F, we can find, *in polynomial time*, a $c \in \mathbb{Z}[y]$ that satisfies $|c(m)| < F'$ for every m with $|m| \le K/2$. We do this by reducing this problem to that of finding a short lattice vector in a suitably defined lattice, and then appealing to the well-known approximate shortest lattice vector algorithms due to [126].

We can view degree ℓ polynomials as vectors in $\mathbb{Z}^{\ell+1}$ in the obvious way. Note that the ideal $J = \prod_i J_i^{z_i}$, when restricted to polynomials of degree at most ℓ, can be viewed as an integer lattice, say L, of dimension $(\ell+1)$. Therefore, finding a suitable non-zero polynomial $c \in J$ with small coefficients amounts to finding a short non-zero lattice vector in L. This can be accomplished using the LLL algorithm, provided we can compute a basis for the lattice L. We now demonstrate how this can be done efficiently.

Note that $L = \cap_i L_i$ where L_i is the lattice corresponding to degree ℓ polynomials in $J_i^{z_i}$, for $1 \le i \le n$. Explicit bases for the individual lattices L_i are easily obtained by considering the generating polynomials for $J_i^{z_i}$ restricted to polynomials of degree at most ℓ. Let $\tilde{z}_i = \min\{z_i, \ell\}$. The first $\tilde{z}_i + 1$ vectors in our basis correspond to the generating polynomials $\{p_i^{(z_i-a)}(y-r_i)^a : 0 \le a \le \tilde{z}_i\}$ from the ideal $I_i^{z_i}$. For example, corresponding to $p_i^{z_i-2}(y-r_i)^2$, we add the vector $(r_i^2 \cdot p_i^{z_i-2}, \quad -2r_i \cdot p_i^{z_i-2}, \quad p_i^{z_i-2}, \quad 0, \quad \ldots, \quad 0)$. If $\ell > z_i$, then we also add vectors corresponding to the polynomials $\{y^a \cdot (y-r_i)^{z_i}\}_{a=1}^{\ell-z_i}$. Let $M^{(i)}$ be the $(\ell+1)$ by $(\ell+1)$ matrix whose rows are the vectors from this basis. It is straightforward to check that the integer linear combinations of these vectors correspond exactly to the set of polynomials of degree at most ℓ in the ideal $J_i^{z_i}$.

Thus bases for each L_i can be computed efficiently. Using standard techniques (see the discussion immediately following this proof), given bases for the (full-dimensional) lattices L_i, a basis B for the intersection lattice $L = \cap_i L_i$ can be computed in polynomial time.

With this basis in hand, our goal is to find a short vector in L (intuitively, short vectors in the lattice L correspond to polynomials in $\prod_i J_i^{z_i}$ with small coefficients). We argued earlier that there exists a vector $\mathbf{c} = (c_0, c_1, \ldots, c_\ell) \in L$ with $|c_j| \le \frac{F}{(\ell+1)(K/2)^j}$, and we would like to find a vector in L with components not much bigger than this. To do so, it is convenient to work with a re-scaled version L' of the lattice L where $(v_0, v_1, \ldots, v_\ell) \in L$ iff $(v_0, v_1 \cdot (K/2), \ldots, v_\ell \cdot (K/2)^\ell) \in L'$. The vector corresponding to $\mathbf{c}$ in L' has L_2-norm less than $F/\sqrt{\ell+1}$. Applying the LLL algorithm to the $(\ell+1)$-dimensional lattice L', we can therefore find a non-zero vector $\mathbf{w} = (w_0, \ldots, w_\ell) \in L'$ with L_2-norm $\|\mathbf{w}\|_2 < 2^{\ell/2}F/\sqrt{\ell+1}$ in polynomial time. By Cauchy-Schwartz, we have that the L_1-norm of $\mathbf{w}$ satisfies $\|\mathbf{w}\|_1 \le \sqrt{\ell+1} \cdot \|\mathbf{w}\|_2 < 2^{\ell/2}F$. Clearly this implies that the polynomial $w(y) = w_0 + w_1 y + \ldots + w_\ell y^\ell$ satisfies $|w(m)| < 2^{\ell/2}F$ whenever $|m| \le K/2$.

Thus one can apply Lemma 7.8 with F replaced by $2^{\ell/2}F$. Hence the decoding Condition (7.21) must be modified by adding a $\log(2^{\ell/2}) = \ell/2$ term to the right hand side, and then we will have a polynomial time list decoding algorithm working under the modified condition. We therefore conclude that one can list decode in *polynomial* time and output every m with $|m| \le K/2$ that satisfies

$$\sum_{i=1}^{n} a_i z_i \log p_i > \log(\ell+1) + \frac{\ell}{2}\log K + \frac{1}{\ell+1}\sum_{i=1}^{n}\binom{z_i+1}{2}\log p_i \ ,$$

as claimed in the theorem.[5] For easy reference, the CRT list decoding algorithm is described in Figure 7.6.2. □

```
List Decode($\mathbf{r}, \ell, z_1, z_2, \ldots, z_n$)
 1. Let $I_1^{z_i}$ be the set of polynomials that are integer linear combinations of
    $\{p_i^a (x - r_i)^{(z_i - a)}\}_{a=0}^{z_i}$.
 2. Compute a basis for the lattice $L$ of all degree $\ell$ polynomials belonging to
    $\bigcap_{i=1}^{n} I_i^{z_i}$.
 3. Scale this lattice by multiplying the $i$'th coordinate by $(K/2)^{i-1}$ to produce
    the lattice $L'$.
 4. Run LLL to find a short vector $v'$ in $L'$; let it correspond to a degree $\ell$
    polynomial $c(x) \in \mathbb{Z}[x]$.
 5. Find all integer roots $m$ of $c(x)$ (for example, by factoring $c(x)$ over $\mathbb{Z}[x]$
    using [126]).
 6. For each root $m$ with $|m| \le K/2$, define the vector $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ by
    $a_i = 1$ if $m \equiv r_i (\mathrm{mod} p_i)$, and $a_i = 0$ otherwise. Output $m$ if $\mathbf{a}$ satisfies
    Condition (7.20).
```

Fig. 7.2. The list decoding algorithm for Chinese Remainder codes

Discussion of the assumed lattice algorithm: In the above proof we assumed a subroutine to compute the basis for an intersection lattice given the basis of the individual lattices. We now discuss how this may be done — further details and a more formal treatment may be found in [41, 140].

Let L be any full-dimensional lattice of dimension d, with basis given by the rows of the matrix M. We define the *dual* L^* of the lattice L to be

[5]The astute reader might have noticed and be slightly bothered by the fact that we have ignored the $\log(2/(2-\gamma))$ term from Equation (7.21). This would cause an $o(1)$ difference to the result stated. Nevertheless, the result of Theorem 7.11 is itself accurate in its stated form. This is because, instead of the LLL algorithm, one can use Schnorr's improvement to the LLL algorithm, which finds an $2^{\varepsilon\ell}$ approximation to the shortest lattice vector in polynomial time, for any desired constant $\varepsilon > 0$. In this way, we can in fact weaken the requirement of Condition (7.20) by subtracting $(1/2 - \varepsilon)\ell$ from the right hand side.

$\{u \in \mathbb{R}^d \ : \ u \cdot v \in \mathbb{Z} \text{ for all } v \in L\}$. Note that the rows of $\left(M^{-1}\right)^T$ give a basis for L^*.

Note also that given bases for two lattices L_1 and L_2, a basis for (the closure of) the union of the two lattices, denoted $L_1 \cup L_2$, can be found efficiently using algorithms for computing the Hermite Normal Form of a generating set of vectors. Now, to compute a basis for the intersection of two lattices L_1 and L_2, observe that $L_1 \cap L_2 = (L_1^* \cup L_2^*)^*$. Therefore, by combining the facts above, one obtains an efficient algorithm for computing a basis for the intersection of full-dimensional lattices given bases for the individual lattices.

7.6.3 Applications to "Interesting" Weight Settings

The result of Theorem 7.11 gives a general list decoding algorithm that works as long as a certain "weighted" condition is satisfied. We now get specific results for the CRT code for interesting choices of weights on the coordinate positions, through an appropriate choice of parameters (like ℓ, z_i) in Theorem 7.11. We begin by proving a version of Theorem 7.11 with arbitrary (not necessarily integer) values of z_i. The proof is somewhat technical but the main idea is simple: approximate the z_i's by large integers z_i^*, and pick a large enough "list size" parameter ℓ.

Theorem 7.12. *For list decoding of CRT codes, for any tolerance parameter $\varepsilon > 0$, and non-negative reals z_i, when given as input a received word $\mathbf{r}$, we can in time polynomial in $n, \log N$ and $1/\varepsilon$, find a list of all codewords such that*

$$\sum_{i=1}^{n} a_i z_i \log p_i \geq \sqrt{\log K \Big(\sum_{i=1}^{n} z_i^2 \log p_i + \varepsilon z_{\max}^2 \Big)} \ , \tag{7.22}$$

where the a_i's are defined as earlier.

Proof: We may assume that $z_{\max} = 1$ (note that the condition of (7.22) is invariant under scaling of the z_i's, so this can be ensured by dividing out all weights by $z_{\max}$). We will prove the claimed result by appealing to Theorem 7.11 on a suitably chosen set of *integer* weights z_i^*.

Let A be a large integer to be specified later in the proof. Set $z_i^* = \lceil A z_i \rceil$. By Theorem 7.11, for any positive integer ℓ we can successfully list decode (in poly$(n, \log N, A, \ell)$ time) provided

$$\sum_{i=1}^{n} a_i z_i^* \log p_i > \log(\ell+1) + \frac{\ell}{2} \log K + \frac{1}{\ell+1} \sum_{i=1}^{n} \binom{z_i^*+1}{2} \log p_i .$$

We would like to pick a good choice for ℓ. Since $A z_i \leq z_i^* < A z_i + 1$, the above condition is met whenever

$$\sum_{i=1}^{n} a_i z_i \log p_i \geq \frac{\log(\ell+1)}{A} + \frac{\ell}{2A}\log K + \\ + \frac{A}{2(\ell+1)} \sum_{i=1}^{n} \Big(z_i^2 + \frac{3}{A} z_i + \frac{2}{A^2} \Big) \log p_i \ . \tag{7.23}$$

Define $Z_i = z_i^2 + \frac{3}{A} z_i + \frac{2}{A^2}$ for $1 \leq i \leq n$. Let us pick

$$\ell = \left\lceil A \sqrt{\frac{\sum_{i=1}^{n} Z_i \log p_i}{\log K}} \right\rceil - 1. \tag{7.24}$$

It is not difficult to see that for this choice of ℓ, Condition (7.23) is met whenever

$$\sum_{i=1}^{n} a_i z_i \log p_i \geq \frac{1}{A} \log\Bigg(A \sqrt{\frac{\sum_{i=1}^{n} Z_i \log p_i}{\log K}} + 1 \Bigg) + \\ + \sqrt{\log K \Big(\sum_{i=1}^{n} Z_i \log p_i \Big)} \ . \tag{7.25}$$

For $A \geq \frac{10 \log N}{\varepsilon}$, the right side of Equation (7.25) above is at most

$$O(\frac{\log\log N}{\log N}) + \sqrt{\log K \Big(\sum_{i=1}^{n} z_i^2 \log p_i + \frac{\varepsilon}{2} \Big)} \leq \sqrt{\log K \Big(\sum_{i=1}^{n} z_i^2 \log p_i + \varepsilon \Big)}$$

for large N. Thus, Condition (7.25) is met provided

$$\sum_{i=1} a_i z_i \log p_i \geq \sqrt{\log K \Big(\sum_{i=1}^{n} z_i^2 \log p_i + \varepsilon \Big)},$$

and the proof is complete by noting that $A = O(\frac{\log N}{\varepsilon})$ and $\ell = O(\varepsilon^{-1} \log^{3/2} N)$, and so the overall runtime is polynomial in $n, \log N$ and ε^{-1}. $\square$

Corollary 7.13. *For list decoding of CRT codes, for any tolerance parameter $\varepsilon > 0$, and non-negative real weights β_i, when given as input a received word* $\mathbf{r}$*, we can, in time polynomial in $n, \log N$ and $1/\varepsilon$, find a list of all codewords whose $\boldsymbol{\beta}$-weighted agreement with* $\mathbf{r}$ *satisfies:*

$$\sum_{i=1}^{n} a_i \beta_i \geq \sqrt{\log K \left(\sum_{i=1}^{n} \frac{\beta_i^2}{\log p_i} + \varepsilon \max_j \frac{\beta_j^2}{\log p_j} \right)} \ .$$

Proof: Follows by setting $z_i = \beta_i / \log p_i$ in the result of the above theorem. $\square$

Note that the above corollary implies that we can essentially "match" the combinatorial bound of Condition (7.19). Let us now collect further results for the "usual" uniform weighting of the codeword positions, namely $\beta_i = 1$ for each i.

Theorem 7.14. *For list decoding of CRT codes with parameters $(p_1, p_2, \ldots, p_n; K)$, for any $\varepsilon > 0$, we can in time polynomial in $n, \sum_i \log p_i$ and $1/\varepsilon$, find a list of all codewords which agree with a received word in t places provided $t \geq \sqrt{k(n+\varepsilon)}$.*

Proof: Let us apply Theorem 7.12 with $z_i = 1/\log p_{k+1}$ for $1 \leq i \leq k$, $z_i = 1/\log p_i$ for $k < i \leq n$, and $\varepsilon' = \varepsilon \log p_{k+1}$. This gives that we can decode whenever the number of agreements t is at least

$$k - \frac{\log K}{\log p_{k+1}} + \sqrt{\frac{\log K}{\log p_{k+1}} \left(\frac{\log K}{\log p_{k+1}} + \sum_{i=k+1}^{n} \frac{\log p_{k+1}}{\log p_i} + \varepsilon' \right)} .$$

Define $\Delta \overset{\text{def}}{=} k - \frac{\log K}{\log p_{k+1}}$; clearly $\Delta \geq 0$. Since $\log p_{k+1} \leq \log p_i$ for $i = k+1, \cdots, n$, the above condition is met whenever $t \geq \Delta + \sqrt{(k-\Delta)(n-\Delta+\varepsilon)}$. Now, a simple application of the Cauchy-Schwartz inequality shows that $\Delta + \sqrt{(k-\Delta)(n-\Delta+\varepsilon)} \leq \sqrt{k(n+\varepsilon)}$, and thus our decoding algorithm works whenever $t \geq \sqrt{k(n+\varepsilon)}$. $\square$

Theorem 7.15. *For list decoding of CRT codes with parameters $(p_1, p_2, \ldots, p_n; K)$, for any $\varepsilon > 0$, we can in time polynomial in $n, \sum_i \log p_i$ and $1/\varepsilon$, find a list of all codewords which agree with a received word in t places provided*

$$t \geq \sqrt{\log K \left(\sum_{i=1}^{n} \frac{1}{\log p_i} + \varepsilon \right)} .$$

Proof: This follows from Corollary 7.13 with $\beta_i = 1$ for $1 \leq i \leq n$. $\square$

Note that the result of Theorem 7.14 matches the combinatorial bound of Condition (7.18). The bounds in Theorem 7.14 and Theorem 7.15 are incomparable in general.

7.7 GMD Decoding for CRT Codes

For integers k, n, relatively prime integers $p_1 < p_2 < \cdots < p_n$, $K = \prod_{i=1}^{k} p_i$, and any integer j, $1 \leq j \leq n$, Goldreich, Ron, and Sudan [72] gave a near-linear time algorithm to compute the unique integer m in the range $-K/2 < m \leq K/2$, if any, that satisfies

$$\sum_{i=1}^{j} a_i \log p_i > \frac{1}{2}\left(\sum_{i=1}^{j} \log p_i + \sum_{i=1}^{k} \log p_i\right) \tag{7.26}$$

where a_i is defined in the usual way: $a_i = 1$ if $m = r_i (\text{mod } p_i)$ and $a_i = 0$ otherwise. Note that the above algorithm decodes up to half the minimum $\mathbf{w}$-weighted distance ($\frac{1}{2} \cdot \log(N/K)$) for the "natural" weighting $w_i = \log p_i$ of the CRT code. Using this algorithm as the basic subroutine and running a GMD style algorithm similar to Forney [60] (see also Appendix A), we are able to perform such a decoding for *any* "user-specified" choice of weights $\boldsymbol{\beta} = \langle \beta_1, \beta_2, \ldots, \beta_n \rangle$. In other words, we give a soft decoding algorithm for CRT codes for the case of unambiguous decoding (the result of Theorem 7.11 being for the case of soft decoding with lists of size ℓ). While the list decoding algorithm decodes under a more general condition than the soft decoding algorithm to be discussed here, the advantage of this GMD based decoding algorithm is its simplicity and faster runtime (we will get a near-quadratic time algorithm).

To obtain the claimed decoding algorithm, we prove a more general result that applies to any code, and then apply it to the CRT code. Suppose we have an arbitrary code $\mathbf{C}$ of blocklength n. We show how to use a decoding algorithm designed for *any* weighting $\boldsymbol{\alpha}$ to produce one that works for the *desired* weighting $\boldsymbol{\beta}$. Define $A_\alpha = \sum_{i=1}^{n} \alpha_i - D_\alpha$ where D_α is $\boldsymbol{\alpha}$-weighted distance of the code, so that A_α is the maximum $\boldsymbol{\alpha}$-weighted agreement between two distinct codewords of $\mathbf{C}$. A_β for the weight vector $\boldsymbol{\beta}$ is defined similarly. We are now ready to state and prove the main result of this section:

Proposition 7.16. *Let* $\mathbf{C}$ *be an arbitrary code of blocklength* n*. Let* $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}_+^n$ *be positive real vectors such that* $\frac{\beta_1}{\alpha_1} \geq \frac{\beta_2}{\alpha_2} \geq \cdots \geq \frac{\beta_n}{\alpha_n}$*, and let* A_α, A_β *for the code* $\mathbf{C}$ *defined as described above. Suppose we have a polynomial time algorithm* Alg_α *that when given as input a received word* $\mathbf{r} = \langle r_1, \ldots, r_n \rangle$ *and an index* j *(*$1 \leq j \leq n$*), can find the unique codeword* $\mathbf{c} \in \mathbf{C}$*, if any, whose* $\boldsymbol{\alpha}$*-weighted agreement with* $\mathbf{r}$ *in the first* j *codeword positions is more than* $\frac{1}{2}\big(\sum_{i=1}^{j} \alpha_i + A_\alpha\big)$*. Then, for any vector of positive reals* $\boldsymbol{\beta} = \langle \beta_1, \ldots, \beta_n \rangle$*, there is a polynomial time algorithm* Alg_β *that when given as input a received word* $\mathbf{r}$*, outputs the unique codeword, if any, whose* $\boldsymbol{\beta}$*-weighted agreement with* $\mathbf{r}$ *is at least*

$$\frac{1}{2}\Big(\sum_{i=1}^{n} \beta_i + A_\beta + \beta_{\max}\Big) .$$

Moreover, the run-time of Alg_β *is at most* $O(n)$ *times that of* Alg_α*.*

Proof: Recall that the codeword positions i are ordered so that $\frac{\beta_1}{\alpha_1} \geq \frac{\beta_2}{\alpha_2} \geq \cdots \geq \frac{\beta_n}{\alpha_n}$. Define

$$\tilde{A}_\beta \overset{\text{def}}{=} \max_{\substack{\mathbf{x} \in [0,1]^n \\ \sum \alpha_i x_i \leq A_\alpha}} \left\{ \sum_{i=1}^{n} \beta_i x_i \right\} . \tag{7.27}$$

Note that under the condition $\mathbf{x} \in \{0,1\}^n$, the above would just define A_β; we relax the condition to $\mathbf{x} \in [0,1]^n$ in the above to define $\tilde{A}_\beta$. Clearly $\tilde{A}_\beta \geq A_\beta$. It is also easy to verify that $\tilde{A}_\beta < A_\beta + \beta_{\max}$. We will present an algorithm to find the unique codeword $\mathbf{c} = \langle c_1, c_2, \ldots, c_n \rangle \in \mathbf{C}$, if any, that satisfies

$$\sum_{i=1}^{n} a_i \beta_i > \frac{1}{2}\big(\sum_{i=1}^{n} \beta_i + \tilde{A}_\beta\big) \tag{7.28}$$

(where $a_i = 1$ if $c_i = r_i$ and 0 otherwise), and this will imply the claimed result (since $\tilde{A}_\beta < A_\beta + \beta_{\max}$). We now assume such a codeword $\mathbf{c}$ exists, as otherwise there is nothing to prove.

The algorithm Alg_β will simply run Alg_α for all values of j, $1 \leq j \leq n$, and pick the closest codeword among the (at most n) codewords which the runs of Alg_α returns. If this algorithm fails to find the codeword $\mathbf{c}$ that satisfies Condition (7.28), then, by the hypothesis of the Theorem, the following condition must hold for every j, $1 \leq j \leq n$:

$$2\sum_{i=1}^{j} a_i \alpha_i \leq \sum_{i=1}^{j} \alpha_i + A_\alpha \ . \tag{7.29}$$

Let $\tilde{\mathbf{x}} = \langle 1 \ 1 \ \cdots \ 1 \ \varepsilon \ 0 \ \cdots \ 0 \rangle$ be a vector such that $\sum_{i=1}^{n} \alpha_i \tilde{x}_i = A_\alpha$ (here $0 \leq \varepsilon < 1$). Denote by ℓ the last position where $\tilde{x}_i = 1$ (so that $\tilde{x}_\ell = 1$ and $\tilde{x}_{\ell+1} = \varepsilon$). By our definition (7.27) $\tilde{A}_\beta \geq \sum_i \beta_i \tilde{x}_i$ (in fact by the ordering of the codeword positions it is also true that $\tilde{A}_\beta = \sum \beta_i \tilde{x}_i$, though we will not need this). Now for $j \geq \ell + 1$, $A_\alpha = \sum_{i=1}^{n} \alpha_i \tilde{x}_i = \sum_{i=1}^{j} \alpha_i \tilde{x}_i$. Also, for $1 \leq j \leq \ell$, we have the obvious inequality $\sum_{i=1}^{j} a_i \alpha_i \leq \sum_{i=1}^{j} \alpha_i = \sum_{i=1}^{j} \alpha_i \tilde{x}_i$, which implies

$$2\sum_{i=1}^{j} a_i \alpha_i \leq \sum_{i=1}^{j} \alpha_i + \sum_{i=1}^{j} \alpha_i \tilde{x}_i \ .$$

Combining the above with Equation (7.29) we obtain that the following uniform condition that holds for all j, $1 \leq j \leq n$:

$$2\sum_{i=1}^{j} a_i \alpha_i \leq \sum_{i=1}^{j} \alpha_i + \sum_{i=1}^{j} \alpha_i \tilde{x}_i \ . \tag{7.30}$$

Multiplying the j^{th} inequality above by the non-negative quantity $\big(\frac{\beta_j}{\alpha_j} - \frac{\beta_{j+1}}{\alpha_{j+1}}\big)$ for $1 \leq j \leq n$ (define $\beta_{n+1} = 0$ and $\alpha_{n+1} = 1$), and adding the resulting inequalities, we get

$$2\sum_{i=1}^{n} a_i \beta_i \leq \sum_{i=1}^{n} \beta_i + \sum_{i=1}^{n} \beta_i \tilde{x}_i \leq \sum_{i=1}^{n} \beta_i + \tilde{A}_\beta \ ,$$

which contradicts Condition (7.28). Thus the codeword $\mathbf{c}$ that satisfies (7.28), if any, will indeed be output by the algorithm Alg_β. □

Theorem 7.17. *For the CRT code with parameters $(n, k; p_1, p_2, \ldots, p_n)$, for any received word $\mathbf{r} = \langle r_1, r_2, \ldots, r_n \rangle$, there is a polynomial time (in fact near-quadratic time) algorithm to find the unique codeword $m = (m_1, m_2, \ldots, m_n)$, if any, that agrees with $\mathbf{r}$ in at least $\frac{n+k}{2}$ positions.*

Proof: By the result of [72], we have a near-linear time decoding algorithm for the weighting $\alpha_i = \log p_i$ and $A_\alpha = \log K$ (where $K = p_1 p_2 \cdots p_k$). For $\boldsymbol{\beta}$ equal to the all-ones vector, we have $A_\beta = k - 1$. Therefore, by Proposition 7.16, we can find the unique codeword m that agrees with $\mathbf{r}$ in at least $(n+k)/2$ places, as claimed. $\square$

7.8 Bibliographic Notes

The redundancy property of the Chinese Remainder representation has been exploited often in theoretical computer science. For example, the Karp-Rabin pattern matching algorithm is based on this redundancy [118]. The CRT representation of an integer allows one to reduce computation over large integers to that over small integers. This is also useful in certain complexity-theoretic settings, a notable example being its use in showing the hardness of computing the permanent of 0/1 matrices [191].

The natural error-correcting code (the CRT code) that results from the Chinese Remainder representation has also been studied often in the literature (see [174, 122] and the references there in). The CRT code was proposed as an alternate method for implementing secret sharing [42, 17]. Mandelbaum [133, 134] was the first to consider the basic algorithmic question of decoding the CRT code up to half the minimum distance. He succeeded in giving such a decoding algorithm; however, the runtime of his algorithm was polynomial only when the p_i's are very close to one another, and could be exponential in n otherwise. Goldreich, Ron and Sudan [72] present and analyze a variant of Mandelbaum's algorithm, which can be implemented in near-linear time, and can unique decode the CRT code up to $\frac{(n-k)\log p_1}{\log p_1 + \log p_n}$ errors. This is a close approximation to half the distance when the primes are reasonably close to one another.

Inspired by the success of list decoding algorithms for Reed-Solomon and AG-codes, Goldreich et al [72] considered the list decoding problem for CRT codes. They presented a polynomial time algorithm to list decode CRT codes up to (about) $\left(n - \sqrt{2kn\frac{\log p_n}{\log p_1}}\right)$ errors. For primes which are close to one another and for small values of k/n, this decodes well beyond half the distance of the code. However, this is not the case when the primes vary widely in size and/or the "rate" k/n is large. One of the motivations of the list decoding algorithm in [72] was an application to the average-case hardness of the permanent on certain random matrices — a discussion of this connection appears in the conference version [71] of the same paper. Håstad and Näslund

[100] used the algorithm of [72] to construct new hardcore predicates based on one-way functions.

Subsequent to this, Boneh [31] improved the list decoding algorithm of [72]. His algorithm could correct up to about $\left(n - \sqrt{kn\frac{\log p_n}{\log p_1}}\right)$ errors. One weakness common to all the above results on CRT decoding is their poor(er) performance if the primes vary significantly in size. This can cause the algorithm of Mandelbaum [133] to take exponential time, while it degrades the number of errors that the algorithms of Goldreich et al [72], or Boneh [31] can correct. This weakness is due to an eccentricity of the CRT code: its alphabet size is not uniform, and so the "contribution" of an error is not independent of its location (knowing a residue modulo a larger p_i correctly gives more information than knowing a residue modulo a smaller p_i). Hence one needs to suitably "reweight" the coordinate positions in order to compensate for this inherent disparity between the various positions. This is exactly what the weighted decoding algorithm we discussed in this chapter allows us to do. It thereby permits efficient decoding up to about $(n - \sqrt{kn})$ errors, and thus completely removes the dependence of the number of correctable errors on the size of the p_i's. It was the development of this soft decoding algorithm for CRT codes that caused us to examine in greater detail the algebra underlying the various list decoding algorithms and unveil the unified ideal-theoretic view of decoding presented in this chapter.

The CRT decoding algorithms discussed in this chapter appear in [86]. The general "ideal-theoretic" approach to list decoding algebraic codes was sketched in [86] as an appendix, and it has been further developed and expanded for presentation in this chapter.