中文摘要

在本報告中，我們分別對於低密度檢測矩陣（Low-Density Parity Check, LDPC）及高密度檢測矩陣（High-Density Parity Check, HDPC）提出了數種不同的解碼方法。

針對低密檢測矩陣碼，我們提出了一系列新穎的退火式（Annealing-Type）信度傳遞（Belief-Propagation）演算法來改善傳統信度傳遞演算法的收斂狀況。信度傳遞演算法的收斂點（Points of Convergence)在熱力學的觀點中等同於其系統的局部最小自由能（Local Minimal Free Energy）；因模擬退火(Simulated Annealing）演算法有助於計算上述系統之全域最小自由能，藉由信度傳播計算及自由能計算之間的相互關係，我們將模擬退火演算法中的溫度(Temperature）參數引入信度傳遞演算法中成為該演算法中一系列的全新參數，並在解碼過程中做等同於「降溫」的動態參數調整來改善傳統演算法的收斂狀況。在此報告中我們總共提出了三種退火式演算法，而實驗結果可證明在運算複雜度與收斂速度相等的情況下，新的退火式信度傳遞演算法可提供比傳統演算法更佳的錯誤更正能力。

除了針對低密檢測矩陣碼外，我們亦研究了數種關於具有高密度位元檢測矩陣之線性方塊碼的隨機解碼法。這些方法可被視為一具有可移動中心的隨機球體解碼法，它會根據一球體對稱的機率分佈來選取在中心向量附近的候選碼。此機率分佈中心向量的更新是根據一被稱為交錯熵（Cross-Entropy）方法的蒙地卡羅法來實現。在每一次的交錯熵方法遞迴過程中，一隨機樣本集合被產生並轉化為合法碼。根據這些隨機產生的合法碼與接收向量間的歐幾理得距離，我們選擇 E 個較佳的候選碼組成一菁英集合並用來修正機率分佈進而影響往後遞迴中產生的隨機樣本。為了確保新產生的隨機樣本會越來越集中在正確的傳送碼附近，不僅中心向量將會移動到傳送碼，其蘊含的機率分佈最終亦會退化為只在傳送碼有值的奇異函數。此外，每次遞迴被更新的機率分佈參數應該要促使新的機率分佈與最佳分佈間的庫柏克萊不勒（Kullback-Leibler）距離越來越接近。在本報告中我們提出了三類隨機解碼法。前兩類是特別針對（n,k）里得所羅門碼所提

出的設計。在第一種解碼法中，被產生的隨機樣本代表了一隨機錯誤指標向量集合，其中每一個向量都指出了接收字碼中 $n-k$ 個應該被擦拭的位置。我們將接收字元中被指定的相對位置擦拭後即可利用只具擦拭（Erasures-Only）解碼器還原成候選碼。在第二種方法中，n 維的實數隨機向量被產生並代表著接收字碼的可靠度向量，而其中 $n-k$ 個最不可靠的座標會假設為應該要被擦拭。針對每個隨機樣本，我們將其 k 個最可靠的座標做硬式決策（hard-decision）並利用只具擦拭解碼器將其還原為合法碼。第三種演算法利用一連續位元翻轉演算法來將隨機樣本向量轉換為合法碼。值得一提的是前兩種演算法只對可最大距離分離（Maximum-Distance Separable）碼有用而第三種演算法則沒有這個限制。這些演算法相對於信度傳遞演算法與部分現有的代數演算法提供了性能與複雜度上的改善，尤其是針對具有高碼率、高密度位元檢測矩陣的方塊碼。

關鍵字：低密度檢測矩陣碼、自由能、退火式信度傳遞演算法、高密度檢測矩陣、交錯熵、庫柏克萊不勒距離、里得所羅門碼、隨機解碼法。

Abstract

This report documents our effort in carrying out the NSC-supported project entitled "Advanced Error-Control Coding Technologies and Their Applications" under grant NSC 972221E009082MY3. Our main results are the developments of novel decoding algorithms for both high-density parity-check (HDPC) and low-density parity-check (LDPC) codes.

For the former class we propose a series of novel annealing-type belief propagation (BP) algorithms to improve the convergent behavior of the conventional BP for decoding LDPC codes. By incorporating a dynamic temperature into the free energy formulation, message passing is performed on a dynamic surface of energy cost. The proposed cooling process helps BP converge to a stable fixed point with lower energy value, which gives more accurate estimation of the transmitted codeword. Both the computational complexity and the convergence rate of our algorithms are nearly equal to the conventional BP algorithm. Furthermore, we derive the message passing rules for general binary networks consisting of parity check functions. Simulation results indicate that decoding LDPC and turbo codes using the annealed BP algorithms gives improved performance.

For latter class of codes, we develop several novel stochastic decoding algorithms. Our approach can be regarded as a randomized sphere decoding with moving center that selects candidate codewords around a center vector according to a sphere-symmetric probability distribution. The center (median) vector of the distribution is updated according to Monte Carlo based approach called the Cross-Entropy (CE) method. The CE method produces, in every iteration, a set of random samples which can be transformed into valid codewords. Based on the Euclidean distances between the received word and the random codewords, we select the best E candidates to form the elite set which is then used to modify the probability distribution that govern the generations of the random samples in the ensuing iteration. To ensure that the newly generated samples are concentrated more and more on a small neighborhood of the correct codeword and either

the median vector will move to or the underlying distribution will eventually degenerate to a singularity at the transmitted codeword, the parameters of the updated distribution should be such that the new distribution is closest to the optimal distribution in the sense of the Kullback-Leibler distance (i.e CE).

We propose three classes of stochastic decoding algorithms. The first two are specifically designed for decoding (n, k) Reed-Solomon (RS) codes. For the first decoder, the random samples represent a set of random error locator vectors, each indicates $n - k$ possible erasure positions within the received word. We associate each error locator vector with a candidate codeword by erasures-only (EO) decoding the received word, assuming that erasure locations are those indicated by the error locator vector. The n-dimensional real random vectors in the second algorithm represent reliability vectors whose least reliable $n - k$ coordinates are assumed to be erasures. For each sample, we make component-wise hard-decisions on the most reliable k coordinates and EO-decoding the resulting binary vector. The third algorithm uses a sequential bit flipping algorithm to convert each random sample into a legitimate codewords. The first two algorithms are valid for MDS codes only while the third algorithm can be used for decoding any linear block code. Our algorithms offer both complexity and performance advantage over BP and some existing algebraic decoding algorithms, especially for high rate linear block codes with HDPC matrices and short or medium lengths.

Index Terms-Low-density parity-check codes, free energy, annealing belief-propagation algorithm, high-density parity-check matrix, cross-entropy, Kullback-Leibler distance, Reed-Solomon codes, stochastic decoding algorithm.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Belief propagation algorithms have been recognized as efficient and powerful inference tools in computer vision, artificial intelligence, error-correcting code, and digital communications. Lately, the BP type algorithms, e.g. BCJR for turbo decoding and sum-product algorithm (SPA) for LDPCC, have received significant attention for their near Shannon-limit error performance. Recent studies show that fixed points of BP algorithms corresponding to the stationary points of the Bethe approximation of the free energy for a factor graph [1]. However, loopy BP algorithms do not promise the convergence when the graph containing cycles. Even if BP converges, it is only guaranteed to converge in a local minimum of Bethe free energy. Some new algorithms attempt to solve the problem by directly minimizing the free energy [2, 3] through using the conventional optimization schemes. These algorithms, however, are often slower than the original BP algorithm.

Inspired by the deterministic annealing methods used in optimization problems [4], we propose a category of annealed BP algorithms to alleviate the ill-convergent effect. By incorporating a dynamic temperature into the formulation of free energy, belief networks start the inference task at higher temperature, which leads to a smoother energy cost surface. According to the results of statistical physics, when the temperature is above the critical point, there is only one local minimum of the free energy function. Thus, BP algorithms can always converge to the unitary minimum at that temperature. However,

inference results at high temperature may be poor due to the unfaithful modelling of actual free energy. On the other hand, there may be more than one local minimum at low temperature. When searching on the energy cost surface at low temperature, we can easily get stuck in one of these local extreme points. However, at low temperature, inference results related with the global minimal free energy turn out to be much more precise than that at the high temperature. Hence, the idea is to start the inference job at a high temperature and smoothly decrease the temperature using some cooling strategies to track the minimum point. Finally, when the temperature approaches to zero, stop the algorithm and output the estimation results. Such a cooling strategy prevents BP algorithms from sticking in some local minimum quickly, and helps the algorithms to converge to the global minimum value with larger probability.

Linear block codes are popular forward error-correcting (FEC) codes due to their simple structures and satisfactory FEC performance. For instance, Reed-Solomon (RS) codes [5] are used in a wide variety of commercial applications, most prominently in CDs, DVDs and Blue-ray discs, in data transmission technologies such as DSL and WiMAX, in broadcast systems such as DVB and ATSC, and in computer applications such as RAID 6 systems. Low density parity-check (LDPC) codes [6] form another class of linear block codes which offer FEC capability close to the theoretical maximum–the Shannon limit [7]. In recent years, LDPC codes have been adopted by several digital broadcast and communication standards such as the DVB-S2 [8], the IEEE 802.3an (10GBASE-T) [9], the IEEE 802.16e (WiMAX) [10], and the IEEE 802.11n (WiFi) [11]. Although many decoding algorithms for block codes are available, more efficient decoding algorithms which can provide performance enhancement and complexity reduction are still of high demand.

Most hard-decision decoding algorithms are bounded-distance decoders (BDD). They select the codeword $\mathbf{c}$, if exits, whose Hamming distance (HD) to the hard-limiting received word $\mathbf{z}$, say $d(\mathbf{z}, \mathbf{c})$, is less than or equal to $\lfloor (d_{min} - 1)/2 \rfloor = t_{min}$, where d_{min} is

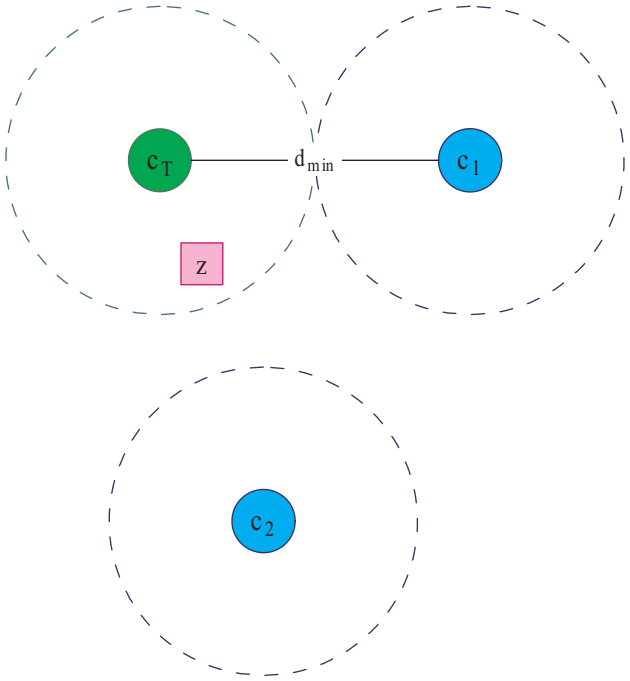

Figure 1.1: A correctly decoding example of a bounded distance decoder.

the minimum distance of the code $\mathbf{C}$. As shown in Fig. 1.1, if $\mathbf{z}$ is within the decoding sphere centered at the transmitted codeword $\mathbf{c}_T$ and then the BDD can correctly output $\mathbf{c}_T$. Syndrome decoding for Hamming codes [16], the Berlekamp-Massey (BM) algorithm [17] and the Euclidean algorithm [18] for RS codes all belong to the class of BDDs. When $\mathbf{z}$ falls into another decoding sphere, e.g., a sphere centered at other legitimate codeword $\mathbf{c}_1$ as shown in Fig. 1.2, a BDD will make an incorrect decision such that a decoding error occurs. A decoding failure is declared if $\mathbf{z}$ does not belong to any decoding sphere of radius t_{min}.

In general, a BDD can only correct up to $\lfloor (d_{min} - 1)/2 \rfloor$ errors while a maximum likelihood soft-decision based decoding algorithm can easily correct beyond t_{min} at the expense of much higher complexity. There are two general approaches to improve the performance without incurring too much complexity increase. The first one is trying to enlarge the decoding sphere (see Fig. 1.4) in order to correct errors beyond t_{min}. For RS codes, the errors-and-erasures decoding [16], Forney's generalized minimum distance

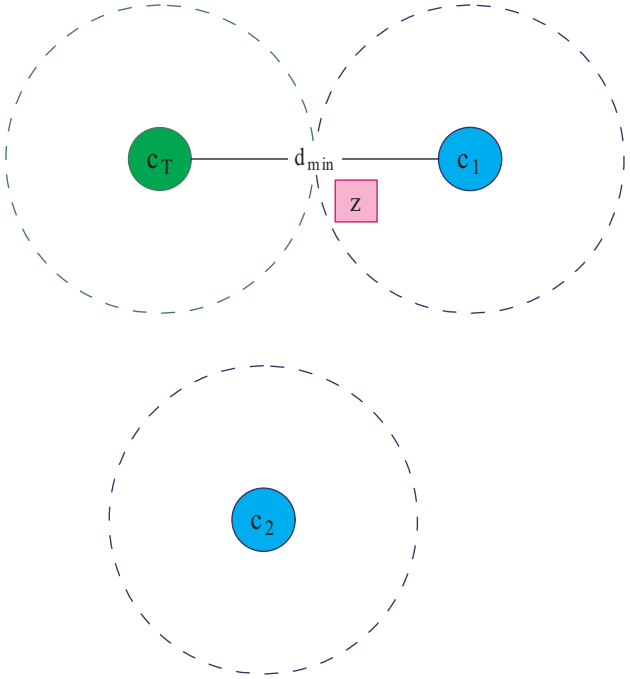

Figure 1.2: An example of erroneous decoding for a bounded distance decoder.

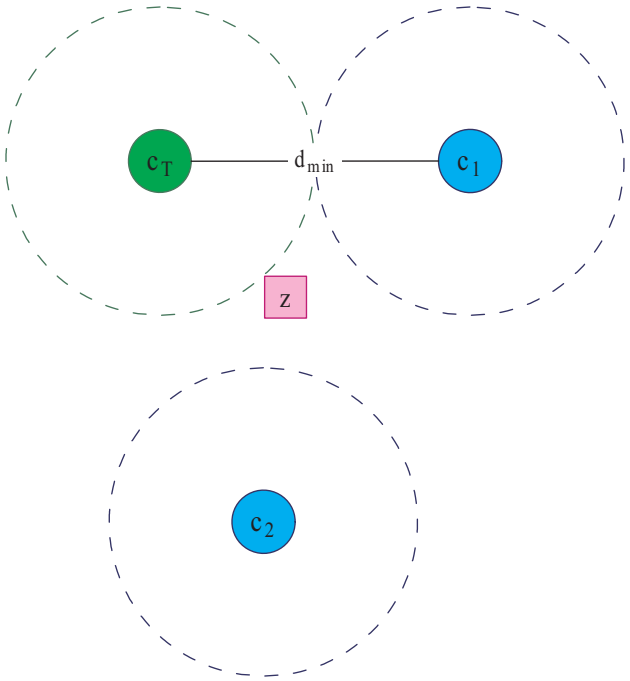

Figure 1.3: Decoding failure by a bounded distance decoder.

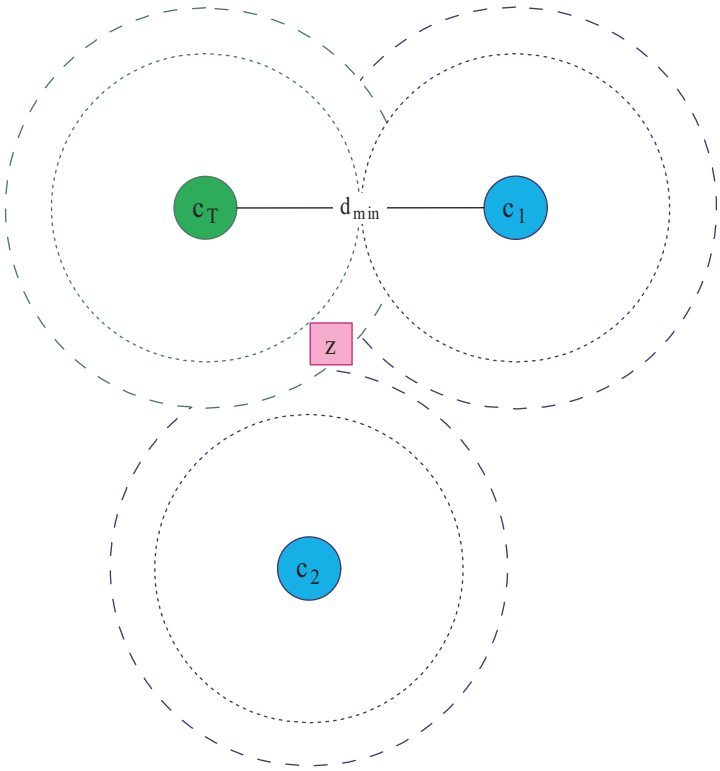

Figure 1.4: Decoding beyond FEC bound by enlarging the decoding sphere.

(GMD) decoding [19], the algebraic list decoding algorithm invented by Guruswami and Sudan (GS) [20] and the algebraic soft decision decoding (SDD) algorithm proposed by Koetter and Vardy (KV) [21] belong to this category. Note that the latter three algorithms are also members of the so-called list decoding algorithms because the enlarged decoding sphere may include more than one codewords.

Another idea for performance enhancement is to sequentially modify and move $\mathbf{z}$ from its original position so that the new location becomes closer and closer to $\mathbf{c}_T$. Decoding methods based on this idea include the Chase II algorithm [22], and the combined Chase II-GMD algorithm [23]. The belief propagation (BP) based algorithms such as the sum product algorithms (SPA) or its less-complex approximation, the min-sum algorithms (MSA) [24] and their variations are also members of this category. A successful decoding based on BP algorithm will gradually update the estimated soft output and move the modified received vector toward the true transmitted codeword

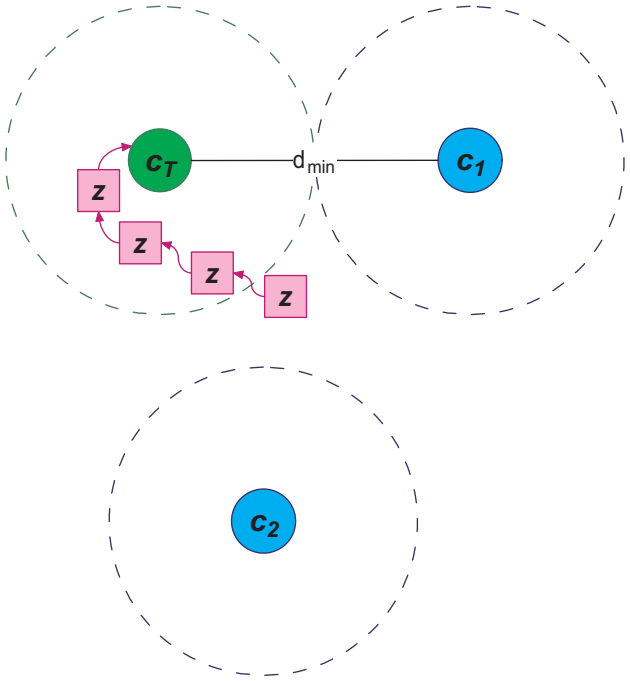

Figure 1.5: Belief propagation - successful decoding.

$\mathbf{c}_T$; see Fig. 1.5. Unfortunately, the BP process may be trapped in some local minimum and the modified received vector coincides with a pseudo codeword $\mathbf{c}_p$ as is shown in Fig. 1.6. This phenomenon can be prevented by using some modification of the BP algorithms such as the annealed BP algorithm [25]. Another possible solution combines the BP algorithm with the BDD such as the algorithms proposed in [12] and [26]. If the pseudo codeword $\mathbf{c}_p$ belongs to the decoding sphere of $\mathbf{c}_T$, successful decoding is achieved although the BP algorithm makes $\mathbf{z}$ coincides with $\mathbf{c}_p$.

We investigate a novel idea of iterative decoding which is a randomized sphere decoding with moving center. If statistical information about possible locations of the transmitted codeword $\mathbf{c}_T$ around the received word $\mathbf{z}$ is given, the order of search should follow the most possible direction. However, we don't have such information usually and hence we search follow a probability distribution which is learned by random sampling. Each sample is transformed into a valid codeword and we choose samples whose corresponding code words having smaller Euclidean distance (ED) to $\mathbf{z}$ to modify the

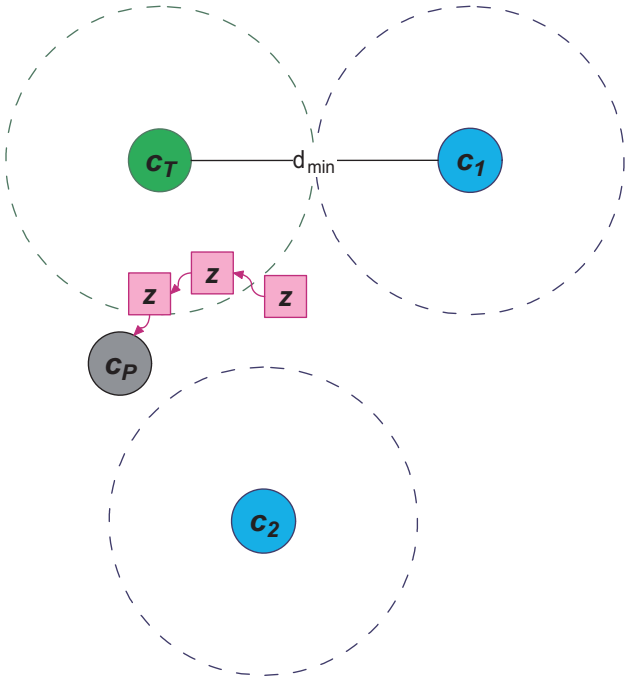

Figure 1.6: Belief propagation - trapped in a pseudo codeword.

distribution and update (move) $\mathbf{z}$. As the iteration goes by, newly generated samples are concentrated more and more on a small neighborhood of the correct codeword. The modified distribution becomes closer in the Cross-Entropy (CE) sense to the optimal (Dirac) distribution centered at the true transmitted codeword. The center thus move closer to $\mathbf{c}_T$ accordingly. This concept is implemented by the CE method [33] which has the following two phases:

1. Explore possible directions pointing the shortest way to the transmitted codeword $\mathbf{c}_T$ via a set of random samples generated from a specific random mechanism.

2. Choose better directions to update the parameters of the random mechanism in order to find better direction in next iteration.

Fig. 1.7 and Fig. 1.8 illustrate the basic principle of the above idea.

The rest of this thesis is organized as follows. Chapter 2 introduces the CE method which is an elegant practical principle for efficiently simulating rare events and can be

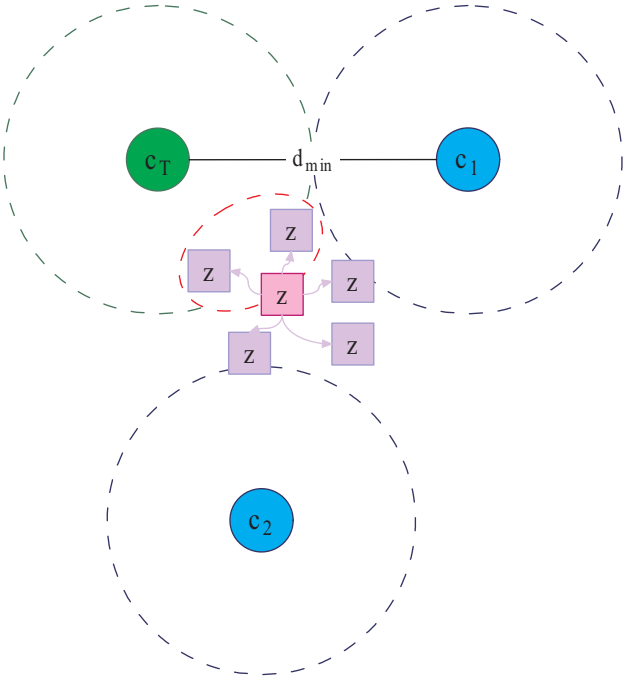

Figure 1.7: A set of random samples are generated and the random samples in the small dash circle are better directions we want.

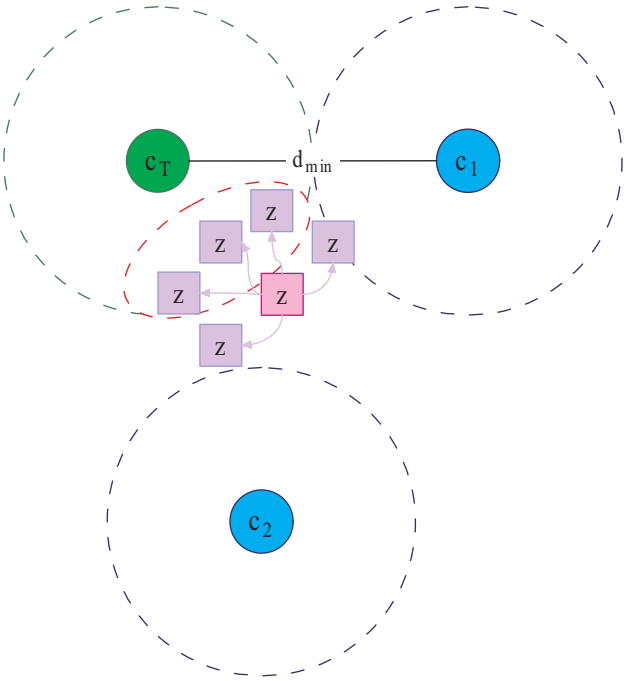

Figure 1.8: After updating the parameter of the random mechanism, the new set of generated random samples points the correct way more often.

converted into an optimization solver. A stochastic erasure-only list decoding (SEOLD) algorithm uses the extended CE method for optimization problem by considering an optimal event as a rare event is illustrated in Chapter 3. In Chapter 4, we investigate another stochastic list decoding algorithm based on a novel sequential bit flipping procedure. Finally, we summarize our major contributions and suggest some future works in Chapter 5.

Chapter 2

Constraint Relaxation and Annealed Belief Propagation for Binary Networks

In section I, we first formulate the free energy of a belief network by taking into consideration the temperature effect. Next, we conduct the BP algorithms using the parameterized free energy in section II. Section III develops several cooling strategies when dealing with binary belief networks, which are commonly used when decoding error correcting codes especially for turbo-like codes. Section IV discusses several cooling schedules and their performances. In section V, we gives the simulation results of the proposed annealing algorithms applied to error correcting codes. Finally, we conclude the works.

2.1 Minimum Free Energy under Bethe Approximation

It is well known that the conventional SPA can be derived from the minimization of Bethe free energy when the temperature equals to one [1]. When developing the annealing type BP algorithms, we begin with the Bethe free energy formulation, which retains the temperature parameter. Considering a factor graph illustrated in Fig. 1, the Bethe free

energy can be written as

$$F_{\text{Bethe}} = U - TH. \tag{2.1}$$

U in (2.1) is the variational average energy

$$U = \langle E_\mu \rangle_{\mathbf{x}_\mu} + \langle E_k \rangle_{x_k}, \tag{2.2}$$

where $\langle E_\mu \rangle_{\mathbf{x}_\mu}$ and $\langle E_k \rangle_{x_k}$ are the average energy functions associated with factor node μ and variable node k respectively,

$$\langle E_\mu \rangle_{\mathbf{x}_\mu} \stackrel{def}{=} -\sum_{\mu} \sum_{\mathbf{x}_\mu} b_\mu(\mathbf{x}_\mu) \ln f_\mu(\mathbf{x}_\mu) \tag{2.3}$$

$$\langle E_k \rangle_{x_k} \stackrel{def}{=} -\sum_{k} \sum_{x_k} b_k(x_k) \ln g_k(x_k). \tag{2.4}$$

$f_\mu(\mathbf{x}_\mu)$ denotes the factor function with its domain $\mathbf{x}_\mu : \{x_k \mid x_k \in \text{neighbors of } f_\mu\}$. $g_k(x_k)$ is the local function associated with variable nodes x_k.

The second term H in (2.1) denotes the variational entropy under the Bethe approximation,

$$H = -\sum_{\mu} \sum_{\mathbf{x}_\mu} b_\mu(\mathbf{x}_\mu) \ln b_\mu(\mathbf{x}_\mu) - \sum_{k} c_k \sum_{\mathbf{x}_k} b_k(x_k) \ln b_k(x_k). \tag{2.5}$$

In (2.5), c_k is defined as 1−(numbers of neighbors of x_k).

Parameter T in (2.1) tends to mimic the temperature effect in the field of statistic physics. Under the marginalization and normalization constraints which state that

$$\sum_{\mathbf{x}_\mu \setminus x_k} b_\mu(\mathbf{x}_\mu) = b_k(x_k) \tag{2.6}$$

$$\sum_{\mathbf{x}_\mu} b_\mu(\mathbf{x}_\mu) = 1 \tag{2.7}$$

$$\sum_{x_k} b_k(x_k) = 1, \tag{2.8}$$

the Bethe free energy minimization problem can be translated to the Lagrangian L_{Bethe}

$$\begin{aligned} L_{\text{Bethe}} = F_{\text{Bethe}} &+ \sum_{k} \sum_{\mu \in \mathcal{N}(k)} \sum_{x_k} \lambda_{\mu k}(x_k) \left(\left[b_k(x_k) - \sum_{\mathbf{x}_\mu \setminus x_k} b_\mu(\mathbf{x}_\mu) \right] \right) \\ &+ \sum_{\mu} \gamma_\mu \left[1 - \sum_{\mathbf{x}_\mu} b_\mu(\mathbf{x}_\mu) \right] + \sum_{k} \gamma_k \left[1 - \sum_{x_k} b_k(x_k) \right], \end{aligned} \tag{2.9}$$

where $\lambda_{\mu k}$, γ_μ, and γ_k are the Lagrange multipliers associated with constraints (2.6),(2.7), and (2.8) respectively. $\mathcal{N}(k)$ denotes the set of neighbors of x_k.

To find the minimum value of L_{Bethe}, we first take derivatives with respect to beliefs $b(\mathbf{x}_\mu)$ and $b(x_k)$, and set the results to zero. We have

$$\frac{\partial L_{\text{Bethe}}}{\partial b_\mu(\mathbf{x}_\mu)} = -\ln f_\mu(\mathbf{x}_\mu) + T\left[1+\ln b_\mu(\mathbf{x}_\mu)\right] - \gamma_\mu - \sum_{k\in\mathcal{N}(\mu)} \lambda_{\mu k}(x_k) = 0. \quad (2.10)$$

$$\frac{\partial L_{\text{Bethe}}}{\partial b_k(x_k)} = -\ln g_k(x_k) + Tc_k\left[1+\ln b_k(x_k)\right] - \gamma_k + \sum_{\mu\in\mathcal{N}(k)} \lambda_{\mu k}(x_k) = 0 \quad (2.11)$$

Solving (2.10) and (2.11), we have the optimum values of beliefs

$$b_\mu^{(*)}(\mathbf{x}_\mu) = f_\mu^\beta(\mathbf{x}_\mu)\exp\left\{\beta\left[\gamma_\mu + \sum_{k\in\mathcal{N}(\mu)}\lambda_{\mu k}(x_k)\right] - 1\right\} \quad (2.12)$$

$$b_k^{(*)}(x_k) = g_k^{\frac{\beta}{c_k}}(x_k)\exp\left\{\frac{\beta}{c_k}\left[\gamma_k - \sum_{\mu\in\mathcal{N}(k)}\lambda_{\mu k}(x_k)\right] - 1\right\}, \quad (2.13)$$

where $\beta \stackrel{def}{=} 1/T$ is a parameter used to model the temperature effect.

To conduct the BP algorithms, we have to derive the Lagrange duality function of the Bethe free energy (2.9). The dual function G_{Bethe} is defined as

$$G_{\text{Bethe}} \stackrel{def}{=} \inf_{b_\mu(\mathbf{x}_\mu), b_k(x_k)} L_{\text{Bethe}} \quad (2.14)$$

Substituting (2.12) and (2.13) back to L_{Bethe}, we have

$$\begin{aligned} G_{\text{Bethe}} &= L_{\text{Bethe}}(b_\mu^{(*)}(\mathbf{x}_\mu), b_k^{(*)}(x_k), \lambda_{\mu k}(x_k), \gamma_\mu, \gamma_k) \\ &= -\sum_\mu \sum_{\mathbf{x}_\mu} b_\mu^{(*)}(\mathbf{x}_\mu)\ln f_\mu(\mathbf{x}_\mu) - \sum_k \sum_{x_k} b_k^{(*)}(x_k)\ln g_k(x_k) \\ &\quad + T\sum_\mu \sum_{\mathbf{x}_\mu} b_\mu^{(*)}(\mathbf{x}_\mu)\ln b_\mu^{(*)}(\mathbf{x}_\mu) + T\sum_k c_k \sum_{x_k} b_k^{(*)}(x_k)\ln b_k^{(*)}(x_k) \\ &\quad + \sum_k \sum_{\mu\in\mathcal{N}(k)} \sum_{x_k} \lambda_{\mu k}(x_k)\left[b_k^{(*)}(x_k) - \sum_{\mathbf{x}_\mu\setminus x_k} b_\mu^{(*)}(\mathbf{x}_\mu)\right] \\ &\quad + \sum_\mu \gamma_\mu\left[1 - \sum_{\mathbf{x}_\mu} b_\mu^{(*)}(\mathbf{x}_\mu)\right] + \sum_k \gamma_k\left[1 - \sum_{x_k} b_k^{(*)}(x_k)\right] \end{aligned} \quad (2.15)$$

Combining (2.15) with (2.12) and (2.13), the dual energy function can be rewritten as

$$
\begin{aligned}
G_{\text{Bethe}} &= -T\sum_{\mu}\sum_{\mathbf{x}_\mu} b_\mu^{(*)}(\mathbf{x}_\mu) - T\sum_{k}\sum_{x_k} c_k b_k^{(*)}(x_k) + \sum_{\mu}\gamma_k + \sum_{k}\gamma_k \qquad (2.16)\\
&= \sum_{\mu}\gamma_\mu + \sum_{k}\gamma_k - T\sum_{\mu}\sum_{\mathbf{x}_\mu} f_\mu^{\beta}(\mathbf{x}_\mu)\exp\left[\beta\left(\gamma_\mu + \sum_{k\in\mathcal{N}(\mu)}\lambda_{\mu k}(x_k)\right) - 1\right]\\
&\quad -T\sum_{k} c_k \sum_{x_k} g_k^{\frac{\beta}{c_k}}(x_k)\exp\left[\frac{\beta}{c_k}\left(\gamma_k - \sum_{\mu\in\mathcal{N}(k)}\lambda_{\mu k}(x_k)\right) - 1\right]. \qquad (2.17)
\end{aligned}
$$

Taking derivative of (2.17) with respect to $\lambda_{\mu k}$, γ_μ, and γ_k gives the following optimum equations,

$$
\begin{aligned}
\frac{\partial G_{\text{Bethe}}}{\partial \lambda_{\mu k}} &= \sum_{\mathbf{x}_\mu\setminus x_k}\left\{f_\mu^{\beta}(\mathbf{x}_\mu)\exp\left[\beta\left(\gamma_\mu + \sum_{j\in\mathcal{N}(\mu)}\lambda_{\mu j}(x_j)\right) - 1\right]\right\}\\
&\quad -g_k^{\frac{\beta}{c_k}}(x_k)\exp\left[\frac{\beta}{c_k}\left(\gamma_k - \sum_{\xi\in\mathcal{N}(k)}\lambda_{\xi k}(x_k)\right) - 1\right] = 0 \qquad (2.18)\\
\frac{\partial G_{\text{Bethe}}}{\partial \gamma_\mu} &= 1 - \sum_{\mathbf{x}_\mu} f_\mu^{\beta}(\mathbf{x}_\mu)\exp\left[\beta\left(\gamma_\mu + \sum_{k\in\mathcal{N}(\mu)}\lambda_{\mu k}(x_k)\right) - 1\right] = 0 \qquad (2.19)\\
\frac{\partial G_{\text{Bethe}}}{\partial \gamma_k} &= 1 - \sum_{x_k} g_k^{\frac{\beta}{c_k}}(x_k)\exp\left[\frac{\beta}{c_k}\left(\gamma_k - \sum_{\mu\in\mathcal{N}(k)}\lambda_{\mu k}(x_k)\right) - 1\right] = 0. \qquad (2.20)
\end{aligned}
$$

(2.19) and (2.20) are normalization equations that give the two partition functions

$$
\begin{aligned}
\sum_{\mathbf{x}_\mu}\left\{f_\mu^{\beta}(\mathbf{x}_\mu)\exp\left[\beta\sum_{k\in\mathcal{N}(\mu)}\lambda_{\mu k}(x_k)\right]\right\} &= \exp\left(1-\beta\gamma_\mu\right), \qquad (2.21)\\
\sum_{x_k}\left\{g_k^{\frac{\beta}{c_k}}(x_k)\exp\left[-\frac{\beta}{c_k}\sum_{\mu\in\mathcal{N}(k)}\lambda_{\mu k}(x_k)\right]\right\} &= \exp\left(1-\frac{\beta}{c_k}\gamma_k\right). \qquad (2.22)
\end{aligned}
$$

From (2.18), we define $m_{\mu k}$ as message passing from factor node μ to variable node

k, and $n_{k\mu}$ as message passing from variable node k to factor node μ,

$$m_{\mu k}(x_k) \overset{def}{=} g_k^{\frac{\beta}{c_k}}(x_k) \exp\left[-\beta\lambda_{\mu k}(x_k) - \frac{\beta}{c_k}\sum_{\xi\in\mathcal{N}(k)}\lambda_{\xi k}(x_k)\right] \tag{2.23}$$

$$\overset{(2.18)}{=} \sum_{\mathbf{x}_\mu\setminus x_k}\left\{f_\mu^\beta(\mathbf{x}_\mu)\exp\left[\beta\left(\gamma_\mu + \sum_{j\in\mathcal{N}(\mu)\setminus k}\lambda_{\mu j}(x_j)\right) - \frac{\beta}{c_k}\gamma_k\right]\right\} \tag{2.24}$$

$$n_{k\mu}(x_k) \overset{def}{=} \exp\left[\beta\lambda_{\mu k}(x_k)\right] \tag{2.25}$$

Combining (2.24) and (2.25), we have

$$m_{\mu k}(x_k) \propto \sum_{\mathbf{x}_\mu\setminus x_k}\left\{f_\mu^\beta(\mathbf{x}_\mu)\prod_{j\in\mathcal{N}(\mu)\setminus k} n_{\mu j}(x_j)\right\}. \tag{2.26}$$

Also, from (2.23) and (2.25), message $n_{k\mu}$ can be updated by

$$g_k^\beta(x_k)\cdot\prod_{\xi\neq\mu,\xi\in\mathcal{N}(k)} m_{\xi k}(x_k) = \exp\left(\beta\lambda_{\mu k}(x_k)\right) \tag{2.27}$$

$$= n_{k\mu}(x_k). \tag{2.28}$$

Equations (2.26) and (2.28) recover the original BP algorithm when β equals to 1.

According to (2.13) and (2.23), belief of the variable node can be estimated as

$$b_k^{(*)}(x_k) \propto m_{\mu k}(x_k)\cdot\exp(\beta\lambda_{\mu k}(x_k)) \tag{2.29}$$

$$= g_k^\beta(x_k)\cdot m_{\mu k}\cdot\prod_{\xi\neq\mu,\xi\in\mathcal{N}(k)} m_{\xi k}(x_k) \tag{2.30}$$

$$= g_k^\beta(x_k)\prod_{\mu\in\mathcal{N}(k)} m_{\mu k} \tag{2.31}$$

From (2.26) and (2.28), we conclude that the BP algorithm can be generalized by incorporating a temperature parameter T, and replace the factor function $f_\mu(\mathbf{x}_\mu)$ by $f_\mu^\beta(\mathbf{x}_\mu)$, and local function $g_k(x_k)$ by $g_k^\beta(x_k)$.

2.2 Annealing Belief Propagation for Binary Network

From the discussions in section II, we know that the first step to implement the annealing type BP algorithm is to parameterize the original local and factor functions with an

artificial parameter β, i.e., $(1/T)$. In the following, we develop the parameterization methodologies for decoding error correcting codes. It can also be extended to many other fields, for example, signal processing, speech processing and pattern recognition.

2.2.1 Local Function Modelling

Considering a low density parity check code (LDPCC) in Gaussian environment using BPSK modulation, the local likelihood function $g_k(x_k)$ associated with each code node x_k is modelled as,

$$g_k(x_k) = p(x_k|y_k) = \sqrt{\frac{1}{2\pi}} \exp\left(\frac{-(y_k - x_k)^2}{2\sigma^2}\right). \tag{2.32}$$

The likelihood ratio of x_k is

$$l_k = \frac{g_k(x_k = -1)}{g_k(x_k = 1)} = \frac{\exp\left(-\frac{(y_k+1)^2}{2\sigma^2}\right)}{\exp\left(-\frac{(y_k-1)^2}{2\sigma^2}\right)} = \exp\left(-\frac{2y_k}{\sigma^2}\right). \tag{2.33}$$

According to (2.28), the generalized likelihood function for Gaussian channel can be rewritten as,

$$g_k^\beta(x_k) \propto \exp\left(\beta \cdot \frac{-(y_k - x_k)^2}{2\sigma^2}\right). \tag{2.34}$$

Hence, the generalized likelihood ratio becomes

$$l_k^\beta = \frac{g_k^\beta(x_k = -1)}{g_k^\beta(x_k = 1)} = \frac{\exp\left(-\beta\frac{(y_k+1)^2}{2\sigma^2}\right)}{\exp\left(-\beta\frac{(y_k-1)^2}{2\sigma^2}\right)} = \exp\left(-\beta\frac{2y_k}{\sigma^2}\right). \tag{2.35}$$

When β changes, (2.35) converges to three interesting limiting points,

$$l_k^\beta = \begin{cases} 1 & \text{,when } \beta \to 0 \text{ (i.e., } T \to +\infty) \\ l_k & \text{,when } \beta \to 1 \text{ (i.e., } T \to 1) \\ \phi(y_k < 0) \cdot \infty & \text{,when } \beta \to +\infty \text{ (i.e., } T \to 0),\ y_k \neq 0. \end{cases} \tag{2.36}$$

$\phi(\cdot)$ denotes the indicator function, where

$$\phi(A) = \begin{cases} 1 & \text{, if event } A \text{ is true,} \\ 0 & \text{, if event } A \text{ is false.} \end{cases} \tag{2.37}$$

Equation (2.36) generalizes the statistical modelling of variable function. Obviously, the conventional likelihood ratio l_k is only a special case when $T \to 1$. As $T \to 0$, which means the code node is in a low temperature state. The generalized likelihood ratio l_k^β approaches an impulse function and gives a deterministic decision of y_k. Otherwise, when $T \to +\infty$, the likelihood ratio tends to be a constant. That means every code node is equally probable when the artificial temperature T approaches to infinity. In other words, by changing the temperature parameter T, we actually model the local likelihood ratio l_k^β in different quantization precisions.

2.2.2 Factor Function Modelling

When decoding an error correcting code, factor function $f_\mu(\mathbf{x}_\mu)$ can be described by an indicator function of parity check equation associated with vector $\mathbf{x}_\mu$, i.e.,

$$f_\mu(\mathbf{x}_\mu) = \delta(\{\oplus_{j\in\mathcal{N}(\mu)} c_j\}), \text{ where } c_j = \begin{cases} 0 & ,\text{if } x_j = 1 \\ 1 & ,\text{if } x_j = -1 \end{cases}. \tag{2.38}$$

$\oplus$ is the exclusive-or operation, and $\delta(\cdot)$ is the delta function

$$\delta(y) = \begin{cases} 1 & ,\text{if } y = 0 \\ 0 & ,\text{otherwise} \end{cases}. \tag{2.39}$$

Since a delta function will not changed by any power other than zero, i.e., $\delta(\cdot) = \delta^\beta(\cdot)$ for $\beta > 0$, it is meaningless to replace the conventional factor function $f_\mu(\mathbf{x}_\mu)$ by $f_\mu^\beta(\mathbf{x}_\mu)$.

An alternative way to parameterize the factor function is to approximate a delta function by the limit of a Gaussian density of a real variable. Alternatively, the Dirac delta function can be rewritten as

$$\delta(y) = \lim_{\beta\to\infty} \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2}y^2\right). \tag{2.40}$$

In (2.40), since the delta function is just a limit case when $\beta \to \infty$, we can generalize it by defining a new function $\delta_\beta(\cdot)$ as

$$\delta_\beta(y) \stackrel{def}{=} \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2}y^2\right). \tag{2.41}$$

Under such an approximation, we naturally incorporate the temperature effect β into the modelling of the factor function $f_\mu^\beta(\mathbf{x}_\mu)$. Significantly, when $0 \leq \beta < +\infty$, $\delta_\beta(y)$ has non zero value for any y of finite value. Hence, in the view of probability, we translate a "deterministic" parity check function $f_\mu(\mathbf{x}_\mu) = \delta(y)$ into a "stochastic" parity check function $f_\mu^\beta(\mathbf{x}_\mu) = \delta_\beta(y)$.

From (2.26) and (2.41), the message $m_{\mu k,\beta}(x_k)$ passing from factor node $f_\mu^\beta(\mathbf{x}_\mu)$ to variable node x_k can be written as

$$m_{\mu k,\beta}(x_k) \propto \sum_{\substack{\{c_j\} \\ j\in\mathcal{N}(\mu),j\neq k}} \left\{ \exp\left[-\frac{\beta}{2} \left(\left(\oplus_{j\in\mathcal{N}(\mu),j\neq k} c_j \right) \oplus c_k \right)^2 \right] \prod_{j\in\mathcal{N}(\mu),j\neq k} n_{j\mu}(x_k) \right\}, \quad (2.42)$$

and the likelihood ratio of $m_{\mu k}(x_k)$ is expressed as

$$l_{m_{\mu k,\beta}} \overset{def}{=} \frac{m_{\mu k,\beta}(x_k=-1)}{m_{\mu k,\beta}(x_k=1)} \quad (2.43)$$

$$= \frac{\sum_{\substack{\{c_j\} \\ j\in\mathcal{N}(\mu),j\neq k}} \left\{ \exp\left[-\frac{\beta}{2} \left(\left(\oplus_{j\in\mathcal{N}(\mu),j\neq k} c_j \right) \oplus 1 \right)^2 \right] \prod_{j\in\mathcal{N}(\mu),j\neq k} n_{j\mu}(x_k) \right\}}{\sum_{\substack{\{c_j\} \\ j\in\mathcal{N}(\mu),j\neq k}} \left\{ \exp\left[-\frac{\beta}{2} \left(\left(\oplus_{j\in\mathcal{N}(\mu),j\neq k} c_j \right) \oplus 0 \right)^2 \right] \prod_{j\in\mathcal{N}(\mu),j\neq k} n_{j\mu}(x_k) \right\}} \quad (2.44)$$

Similarly, there are numbers of limit points when β changes from 0 to $+\infty$.

$$l_{m_{\mu k,\beta}} = \begin{cases} 1 & ,\text{when } \beta \to 0 \\ l_{m_{\mu k}} & ,\text{when } \beta \to +\infty, \end{cases} \quad (2.45)$$

where

$$l_{m_{\mu k}} \overset{def}{=} \frac{\sum_{\substack{\{c_j\} \\ j\in\mathcal{N}(\mu),j\neq k}} \left\{ \delta\left[\left(\left(\oplus_{j\in\mathcal{N}(\mu),j\neq k} c_j \right) \oplus 1 \right) \right] \prod_{j\in\mathcal{N}(\mu),j\neq k} n_{j\mu}(x_k) \right\}}{\sum_{\substack{\{c_j\} \\ j\in\mathcal{N}(\mu),j\neq k}} \left\{ \delta\left[\left(\left(\oplus_{j\in\mathcal{N}(\mu),j\neq k} c_j \right) \oplus 0 \right) \right] \prod_{j\in\mathcal{N}(\mu),j\neq k} n_{j\mu}(x_k) \right\}} \quad (2.46)$$

denotes the likelihood ratio when the factor node is a delta function, i.e., $f_\mu(\mathbf{x}_\mu) = \delta(\mathbf{x}_\mu)$. Also, when the temperature is high, the probabilities of $x_k = 1$ and $x_k = -1$ are equally likely. In the following, we give an example to further study the proposed idea of "stochastic" parity check node.

Example

Consider a factor node $f_\mu^\beta(\mathbf{x}_\mu)$ associated with a parity check equation

$$c_1 \oplus c_2 \oplus c_3 = 0. \tag{2.47}$$

According to (2.42), we have

$$\begin{aligned} m_{\mu 1,\beta}(x_1 = 1) &\propto n_{2\mu}(x_2 = 1)n_{3\mu}(x_3 = 1)e^{-\frac{\beta}{2}\cdot 0} + n_{2\mu}(x_2 = -1)n_{3\mu}(x_3 = 1)e^{-\frac{\beta}{2}\cdot 1} \\ &+ n_{2\mu}(x_2 = 1)n_{3\mu}(x_3 = -1)e^{-\frac{\beta}{2}\cdot 1} + n_{2\mu}(x_2 = -1)n_{3\mu}(x_3 = -1)e^{-\frac{\beta}{2}\cdot 0} \quad (2.48) \\ m_{\mu 1,\beta}(x_1 = -1) &\propto n_{2\mu}(x_2 = 1)n_{3\mu}(x_3 = 1)e^{-\frac{\beta}{2}\cdot 1} + n_{2\mu}(x_2 = -1)n_{3\mu}(x_3 = 1)e^{-\frac{\beta}{2}\cdot 0} \\ &+ n_{2\mu}(x_2 = 1)n_{3\mu}(x_3 = -1)e^{-\frac{\beta}{2}\cdot 0} + n_{2\mu}(x_2 = -1)n_{3\mu}(x_3 = -1)e^{-\frac{\beta}{2}\cdot 1} \quad (2.49) \end{aligned}$$

In (2.48) and (2.49), we find that when $\beta \neq +\infty$, i.e., $T \neq 0^+$, there is a non-zero probability $e^{-\frac{\beta}{2}}$ accompanied with the terms that disappear in the calculation of $m_{\mu 1}$ in the deterministic case. Such a probability explains the idea of a "stochastic" parity check, since we allow the existence of non-zero parity check value with a specified probability.

Interestingly, when we decrease the temperature T from $+\infty$ to 0^+, stochastic parity check will reach two limit states. First, when $T = +\infty$ (i.e., $\beta = 0^+$), both $e^{-\frac{\beta}{2}\cdot 0}$ and $e^{-\frac{\beta}{2}\cdot 1}$ equal to 1 such that $m_{\mu 1}(x_k = 1) = m_{\mu 1}(x_k = -1)$. Hence, the message provided by check node μ at $T = +\infty$ is equally probable. That means we cannot gain any information about code node x_1 from f_μ^β.

On the other hand, as $T = 0^+$ (i.e., $\beta = +\infty$), $e^{-\frac{\beta}{2}\cdot 0} = 1$ and $e^{-\frac{\beta}{2}\cdot 1} = 0$. Message $m_{\mu 1,\beta}(x_k)$ will become

$$\begin{aligned} m_{\mu 1,\beta=0}(x_1 = 1) &\propto n_{2\mu}(x_2 = 1)n_{3\mu}(x_3 = 1) + n_{2\mu}(x_2 = -1)n_{3\mu}(x_3 = -1) \quad (2.50) \\ m_{\mu 1,\beta=0}(x_1 = -1) &\propto n_{2\mu}(x_2 = -1)n_{3\mu}(x_3 = 1) + n_{2\mu}(x_2 = 1)n_{3\mu}(x_3 = -1), \quad (2.51) \end{aligned}$$

which gives exactly the deterministic parity check function. Hence, as $T \to 0$, the stochastic parity check will approach the deterministic case. This can also be explained as the natural result of (2.40).

2.2.3 Trellis State Modelling

2.3 Annealing Belief Propagation Algorithm

The basic principle of annealing type algorithms is to start the inference task at high temperature and iteratively decrease the temperature according to some chosen annealing schedules. In the literatures, there are two main categories when applying the annealing type algorithms. First, the simulated annealing (SA), based on the Markov Chain Monte Carlo (MCMC) method, is quite useful in non-convex optimization problems. In theory, the simulated annealing method generates a sequence of random walks to reach a new state with an acceptance probability that depends on the current state. Intuitively, SA directly simulates the system dynamics. Although SA can theoretically converge to the global optimum value, convergence is only guaranteed under a proper annealing schedule, which starts at a high enough initial temperature. In theory, exponential-time execution is needed for SA to converge to the global optimum. However, such a long annealing time is often not realistic for many applications in practice.

On the other hand, deterministic annealing (DA), the second category of annealing methods, has deterministic processing time. Instead of simulating the stochastic dynamics of the system, DA directly minimizes the expected free energy while avoiding many poor local minima of the cost. According to [4], the DA method performs annealing as it maintains the free energy at the thermal equilibrium while gradually lowering the temperature. Since the processing time of DA is deterministic and controllable, we can take DA as one of the candidate solutions if the implementation of annealing algorithms must be constrained below a fixed processing delay. Lately, DA has already been successfully employed to solve numbers of optimization problems in source coding, pattern recognition, pattern classification, and many other fields.

In this paper, when decoding error correcting codes, we adopt DA to anneal the belief network for numbers of reasons. First, communication systems normally have

constraints on the decoding delay in order to maintain the total system latency. Second, DA combined with the proposed annealing belief propagation (ABP) algorithms needs only minor modification of the conventional BP. In fact, ABP generalizes BP while keeping most of the implementations details, including schedules of message passing, the way that message exchanges, or even total iteration numbers. We describe below the basic ABP framework.

1. Initialize :

   Set initial temperature T_{init}, minimum temperature T_{min}.

   Set maximum iteration number i_{max}, iteration index $i = 1$, temperature index $T_i = T_{init}$, and $\beta_i = \frac{1}{T_i}$.

2. Update :

   Calculate the local function $g_k^{\beta_i}(x_k)$ of variable node x_k for all k.

   Calculate the factor function $f_\mu^{\beta_i}(\mathbf{x}_\mu)$ of the μ-th factor node for all $\mathbf{x}_\mu$.

   Calculate messages $m_{\mu k,\beta_i}(x_k)$ and $n_{k\mu,\beta_i}(x_k)$ for all $\mu,\ k$.

3. Check Temperature : If $T_i \leq T_{\min}$, set $T_i = 0$.

4. Cooling Step : Calculate T_{i+1} and β_{i+1} by decreasing T_i according to the specific annealing schedule.

5. Check Status : Set $i = i+1$. Stop the algorithm when $i > i_{\max}$ or some convergent check is passed.

6. Go to 2).

Following the basic framework of ABP, we propose three types of ABP algorithms: local function annealing (LFA), factor function annealing (FFA), and joint annealing (JA). For different types of ABP algorithms, the function nodes are generalized according to different models described in section III.

2.3.1 Local Function Annealing

Local function annealing can also be called code node annealing when applying ABP to decode error correcting codes. At first, we model the local function by its generalized model. For LDPCC, local functions of code nodes are set to the generalized Gaussian model described in Section III-A. The factor nodes, however, retain the deterministic definition, which means that we still use deterministic parity checks for LDPCC when applying local function annealing. In this case, the exchanged message $m_{\mu k,\beta_i}$ equals to $m_{\mu k,0}$ and $n_{k\mu,\beta_i}$ equals to $n_{k\mu,0}$. Local function annealing follows the basic framework of ABP: start at a high temperature, perform message exchange, and finally check the convergence and decrease the temperature further. Since the ways message exchanges at factor nodes are the same for LFA and BP, the only difference between them is that the former changes the local functions in iterations, while the latter keeps them unchanged. Because LFA only modifies the exponent of local function, the computational complexity is almost the same for LFA and BP.

Based on three previous observations in (2.36), we propose two schedule rules for LFA. First, the initial temperature can be set as high as needed. Second, the lower limit of temperature is bounded by one. An empirical annealing schedule which satisfies both rules is suggested as

$$T_{local,(j+1)\eta} = T_{local,init}^{\gamma^{j+1}} = T_{local,j\eta}^{\gamma}, \tag{2.52}$$

$$\text{and } T_{local,i} = T_{local,j\eta} \text{ for } j\eta \leq i < (j+1)\eta. \tag{2.53}$$

j is the update index and $0 < \gamma \leq 1$ is a constant used to control the annealing rate. The initial temperature is defined as $T_0 \stackrel{def}{=} T_{local,init}$. η denotes the update period of annealing process, and i is the iteration index of ABP algorithms. The artificial parameter η fixes the temperature $T_i = T_{local,j\eta}$ during $j\eta \leq i < (j+1)\eta$ to ensure the ABP algorithm converging to a lower energy state at that temperature. As γ equals to 1, $T_{local,i}$ equals to $T_{local,init}$ for any iteration i. When γ is small, $T_{local,i}$ approaches 1, the low temperature

limit, in few iterations.

2.3.2 Factor Function Annealing

Similarly, factor function annealing is also called check node annealing when applying ABP to decode error correcting codes. The reason for the name "factor function annealing" is that we only perform annealing on the factor nodes. For LDPCC, we replace the deterministic parity check function by the stochastic one described in Section III-B. Local function $g_k(x_k)$ is kept constant during the iteration, i.e.,

$$g_k(x_k) = \delta\left(p(x_k|y_k) = \sqrt{\frac{1}{2\pi}}\exp\left(\frac{-(y_k - x_k)^2}{2\sigma^2}\right)\right). \tag{2.54}$$

According to (2.45), the generalized parity check function reaches two limits when $T = +\infty$, and $T = 0$. Hence, we suggest the following geometric schedule for FFA

$$T_{factor,(j+1)\eta} = \alpha^{j+1} \cdot T_{factor,init} = \alpha \cdot T_{factor,j\eta}, \tag{2.55}$$

$$\text{and } T_{factor,i} = T_{factor,j\eta} \text{ for } j\eta \leq i < (j+1)\eta. \tag{2.56}$$

Similar to LFA, j is the update index and η is the corresponding update period of the annealing process. i denotes the iteration index of ABP algorithms and the initial condition is set as $T_0 \stackrel{def}{=} T_{factor,init}$. $0 < \alpha \leq 1$ is a scaling constant that controls the annealing rate. As $\alpha = 1$, temperature $T_{factor,i}$ equals to $T_{factor,init}$ for all iteration. On the other way, $T_{factor,i}$ approaches 0 in few iterations when α is small. Empirically, the geometric scaling factor α is set between 0.8 and 0.99 [29].

A stopping criterion of the LFA and FFA can be determined either when the maximum iteration limit is reached or all the deterministic parity check functions are satisfied.

2.3.3 Joint Annealing

Combining LFA and FFA, joint annealing anneals both the local functions and the factor functions. Temperature T_i is decreased according to (2.52) and (2.55) for local functions and factor functions respectively. Intuitively, there are several ways to combine the LFA and FFA. We suggest the following two approaches.

Single Loop Joint Annealing (SLJA)

Single loop joint annealing (SLJA) method updates the temperature parameters of LFA and FFA at the same time. The algorithm is summarized as follows:

(1) Set parameters: initial temperature $T_{factor,init}, T_{local,init}$, update period η, local function annealing rate γ, and factor function annealing rate α, update index $j = 0$, and iteration index $i = 0$, minimum temperature $T_{factor,min}$, and $T_{local,min}$.

(2) Set $T_{factor,i} = T_{factor,j\eta}$ and $T_{local,i} = T_{local,j\eta}$ for iteration $i = j\eta$. If $T_{factor,i} < T_{factor,min}$, set $T_{factor,i} = 0$. If $T_{local,i} < T_{local,min}$, set $T_{local,i} = 1$.

(3) Run FFA on factor nodes and LFA on variable nodes.

(4) If $i \neq (j+1)\eta$, set $i \leftarrow i+1$, and $j \leftarrow j$; else set $i \leftarrow i+1$, and $j \leftarrow j+1$.

(5) Lower temperature according to (2.52) and (2.55).

(6) If $i < i_{\max}$, goto step 2, ; else, stop and output the results.

In our experiment, we observe that the performance of SLJA highly depends on the selection of annealing parameters, especially the annealing rates of factor and local functions. The annealing process yields a sequence of solutions at a controlled level of decreasing entropy. It is suggested in [29] that more complicated annealing schedules, such as adaptive annealing, can be adopt to improve the performance further.

Double Loop Joint Annealing (DLJA)

(1) Set parameters: initial temperature $T_{factor,init}, T_{local,init}$, update period η_{factor}, η_{local} where $\eta_{local} = c \cdot \eta_{factor}$ for $c \geq 1, c \in \mathbb{Z}$, local function annealing rate γ, and factor function annealing rate α, update index $j = 0$, $k = 0$, and iteration index $i = 0$, minimum temperature $T_{factor,min}$, and $T_{local,min}$.

(2) Set $T_{local,i} = T_{local,k\eta}$ for iteration $i = k\eta_{local}$. If $T_{local,i} < T_{local,min}$, set $T_{local,i} = 1$.

(3) Set $T_{factor,i} = T_{factor,j\eta}$ for iteration $i = j\eta_{factor}$. If $T_{factor,i} < T_{factor,min}$, set $T_{factor,i} = 0$.

(4) Run FFA on factor nodes and LFA on variable nodes.

(5) Set $i \leftarrow i + 1$.

(6) If $i \neq (j+1)\eta_{factor}$ and $i \neq (k+1)\eta_{local}$, go to step (4). If $i = (j+1)\eta_{factor}$ and $i \neq (k+1)\eta_{local}$, set $j \leftarrow j + 1$, lower temperature of factor functions according to (2.55). and goto step 3; else if $i = (k+1)\eta_{local}$, goto step 7.

(7) Set $k \leftarrow k + 1$, and lower temperature of local functions according to (2.52).

(8) If $i < i_{\max}$, goto step 2; else, stop and output the results.

DLJA is consisted of two main loops. Inner loop, step 3 to step 6, performs FFA and outer loop, i.e., step 2 to step 8, performs LFA. In step 2, the beginning of outer loop, we can also re-heating $T_{factor,i}$ to further escape the local minimum and possibly find better solutions. In DLJA algorithm, we can also run LFA as inner loop and FFA as outer loop. In fact, there are lots of configurations of joint annealing methods, which need to be investigate further.

From the simulation results in the latter section, we observe that performance of FFA, LFA and JA is mainly affected by the settings of initial temperature and annealing rate. That means we can get improvement of the conventional BP even by using the simplest LFA scheme.

2.4 Experimental Results and Discussions

In this section, we report the results of the proposed annealing algorithms on binary networks. We perform ABP to decode LDPCC of different rates and code lengths. Some test codes are chosen for their inherent short cycles, while others are chosen arbitrarily.

Several settings of initial temperature and annealing rate are tested to demonstrate the importance of annealing schedule.

We consider first the (23,12) Golay code, which has cycles of length 4 to prevent the BP algorithm from converging to maximum likelihood decoding (MLD) performance [28].

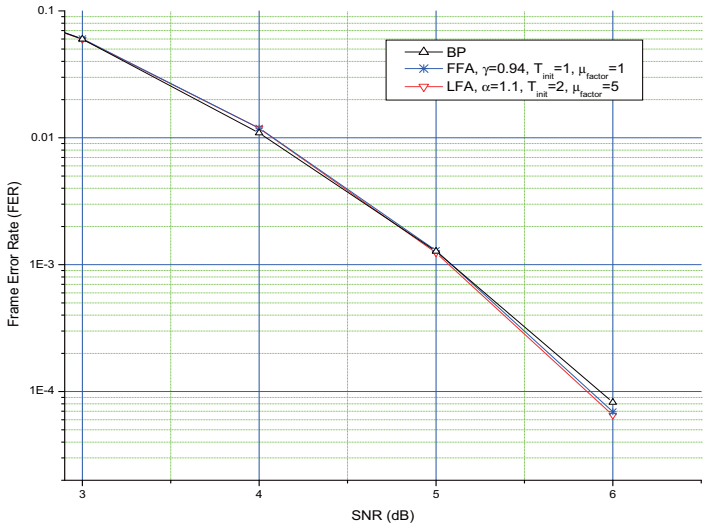

Figure 2.1: Golay Code N=23 K=12, Code Rate: 0.522, Max Iteration Number: 200, Frame Error Count: 200.

In the following tests, we illustrate the results of LDPCC established in database [30]. First, considering the 816.33.164 code of rate 0.5, FFA with initial temperature 1.0 and $\alpha = 0.94$ performs the best, following by the LFA with initial temperature 2.0 and $\gamma = 1.1$. Both of the annealing methods have better bit error rate (BER) performance than the conventional BP algorithm. The convergence advantage of annealing methods becomes more remarkable with the increase of SNR.

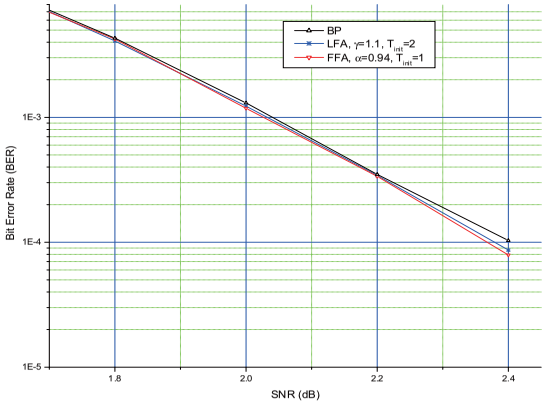

Figure 2.2: Mackay 816.33.164, Code Rate: 0.5, Max Iteration Number: 200, Frame Error Count: 200.

Chapter 3

The Cross-Entropy Method

The cross-entropy (CE) method which was originally developed as an adaptive algorithm for rare-event simulation based on variance minimization [31]. It was soon modified to a randomized optimization technique [32], where the original variance minimization program was replaced by an associated CE minimization problem. We summarize the basic concept of this simple, efficient, and general method in this chapter and more detailed investigations can be found in [33].

3.1 Introduction

In the field of rare-event simulation, the CE method is used in conjunction with importance sampling (IS), a well-known variance reduction technique in which the system is simulated under a different set of parameters, called the reference parameters (or different probability distribution) so as to make the occurrence of the rare event more likely. A major drawback of the conventional IS technique is that the optimal reference parameters to be used in IS are usually very difficult to obtain. Traditional techniques for estimating the optimal reference parameters [34] typically involve time consuming variance minimization programs. The advantage of the CE method is that it provides a simple and fast adaptive procedure for estimating the optimal reference parameters in the IS.

In the field of optimization problems (combinatorial or continuous), the CE method

can be readily applied by first translating the underlying optimization problem into an associated estimation problem, named associated stochastic problem (ASP), which typically involves rare-event estimation. Estimating the rare-event probability and the associated optimal reference parameter for the ASP via the CE method translates effectively back into solving the original optimization problem.

In general, the CE algorithm is an iterative procedure that consists of the following two phases in each iteration.

- Generate samples from the specified importance density given by the parameters from the previous iteration.
- Update the parameters for next iteration according to the order of the score values associated with the drawn samples and the minimizing CE criterion.

The significance of the CE concept is that it defines a precise mathematical framework for deriving fast and good updating/learning rules.

3.2 The CE Method for Rare-Event Simulation

In this section, the basic idea behind the CE algorithm for rare event simulation is illustrated. Let $\mathbf{x}$ be a random vector taking values in some space $\mathcal{X}$. Let $\{f(\cdot;\mathbf{v})\}$ be a family of probability density functions (pdfs) on $\mathcal{X}$, with respect to some base measure μ where $\mathbf{v}$ is a real-valued parameter (vector). Therefore,

$$\mathbb{E}[H(\mathbf{x})] = \int_{\mathcal{X}} H(\mathbf{x}) f(\mathbf{x};\mathbf{v}) \mu(d\mathbf{x}), \tag{3.1}$$

for any function H. For simplicity, for the rest of this section we take $\mu(d\mathbf{x}) = d\mathbf{x}$ because of μ is either a continuous measure or the Lebesgue measure in most cases.

Let S be some real function on $\mathcal{X}$. Suppose we are interested in the probability that $S(\mathbf{x})$ is greater than or equal to some real number γ under $f(\mathbf{x};\mathbf{u})$. This probability can be expressed as

$$\ell = P_{\mathbf{u}}(S(\mathbf{x}) \geq \gamma) = \mathbb{E}_{\mathbf{u}}[I_{\{S(\mathbf{x})\geq\gamma\}}]. \tag{3.2}$$

If this probability is very small, say smaller than 10^{-5}, we call $\{S(\mathbf{x}) \geq \gamma\}$ a rare event.

A straightforward way to estimate ℓ is to use crude Monte-Carlo simulation: Draw a random sample $\mathbf{x}_1, \cdots, \mathbf{x}_N$ from $f(\mathbf{x}; \mathbf{v})$; then

$$\hat{\ell} = \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\mathbf{x}_i) \geq \gamma\}} \tag{3.3}$$

is an unbiased estimator of ℓ. However this poses serious problems when $\{S(\mathbf{x}) \geq \gamma\}$ is a rare event since a large simulation effort is required to estimate ℓ accurately, that is, with a small relative error or a narrow confidence interval.

An alternative is based on importance sampling: take a random sample $\mathbf{x}_1, \cdots, \mathbf{x}_N$ from an importance sampling density g on $\mathcal{X}$, and estimate ℓ using the likelihood ratio (LR) estimator

$$\hat{\ell} = \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\mathbf{x}_i) \geq \gamma\}} \frac{f(\mathbf{x}_i; \mathbf{u})}{g(\mathbf{x}_i)}. \tag{3.4}$$

The best way to estimate ℓ is to use the change of measure with density

$$g^*(\mathbf{x}) = \frac{I_{\{S(\mathbf{x}) \geq \gamma\}} f(\mathbf{x}; \mathbf{u})}{\ell}. \tag{3.5}$$

By using this change of measure we have in (3.4)

$$I_{\{S(\mathbf{x}_i) \geq \gamma\}} \frac{f(\mathbf{x}_i; \mathbf{u})}{g^*(\mathbf{x}_i)} = \ell, \tag{3.6}$$

for all i. Since ℓ is a constant, the estimator (3.4) has zero variance, and we need to produce only $N = 1$ sample.

The obvious difficulty is that g^* depends on the unknown parameter ℓ. Moreover, it is often convenient to choose a g in the family of densities $\{f(\cdot; \mathbf{v})\}$. The idea now is to choose the reference parameter $\mathbf{v}$ such that the distance between the density g^* above and $f(\mathbf{x}; \mathbf{v})$ is minimal. A particularly convenient measure of distance between two densities g and h is the Kullback-Leibler (KL) distance defined as

$$\mathcal{D}(g, h) = \mathbb{E}_g \left[\ln \frac{g(\mathbf{x})}{h(\mathbf{x})} \right] = \int g(\mathbf{x}) \ln g(\mathbf{x}) d\mathbf{x} - \int g(\mathbf{x}) \ln h(\mathbf{x}) d\mathbf{x} \tag{3.7}$$

which is also termed the cross-entropy (CE) between g and h.

Minimizing the Kullback-Leibler distance between g^* in (3.5) and $f(\mathbf{x};\mathbf{v})$ is equivalent to solve the maximization problem

$$\max_{\mathbf{v}} \int g^*(\mathbf{x}) \ln f(\mathbf{x};\mathbf{v}) d\mathbf{x} \tag{3.8}$$

Substituting g^* from (3.5) into (3.8) we obtain the maximization program

$$\max_{\mathbf{v}} \int \frac{I_{\{S(\mathbf{x})\geq\gamma\}} f(\mathbf{x};\mathbf{u})}{\ell} \ln f(\mathbf{x};\mathbf{v}) d\mathbf{x} \tag{3.9}$$

which is equivalent to the program

$$\max_{\mathbf{v}} D(\mathbf{v}) = \max_{\mathbf{v}} \mathbb{E}_{\mathbf{u}} \left[I_{\{S(\mathbf{x})\geq\gamma\}} \ln f(\mathbf{x};\mathbf{v}) \right] \tag{3.10}$$

where D is implicitly defined above. Again using importance sampling, with a change of measure $f(\mathbf{x};\mathbf{w})$ we can rewrite (3.10) as

$$\max_{\mathbf{v}} D(\mathbf{v}) = \max_{\mathbf{v}} \mathbb{E}_{\mathbf{w}} \left[I_{\{S(\mathbf{x})\geq\gamma\}} W(\mathbf{x};\mathbf{u},\mathbf{w}) \ln f(\mathbf{x};\mathbf{v}) \right], \tag{3.11}$$

for any reference parameter $\mathbf{w}$, where

$$W(\mathbf{x};\mathbf{u},\mathbf{w}) = \frac{f(\mathbf{x};\mathbf{u})}{f(\mathbf{x};\mathbf{w})} \tag{3.12}$$

is the likelihood ratio between $f(\mathbf{x};\mathbf{u})$ and $f(\mathbf{x};\mathbf{w})$. The optimal solution of (3.11) can be written as

$$\mathbf{v}^* = \arg\max_{\mathbf{v}} \mathbb{E}_{\mathbf{w}} \left[I_{\{S(\mathbf{x})\geq\gamma\}} W(\mathbf{x};\mathbf{u},\mathbf{w}) \ln f(\mathbf{x};\mathbf{v}) \right]. \tag{3.13}$$

We may estimate $\mathbf{v}^*$ by solving the following stochastic program

$$\max_{\mathbf{v}} \hat{D}(\mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^{N} \left[I_{\{S(\mathbf{x}_i)\geq\gamma\}} W(\mathbf{x}_i;\mathbf{u},\mathbf{w}) \ln f(\mathbf{x}_i;\mathbf{v}) \right], \tag{3.14}$$

where $\mathbf{x}_1, \cdots, \mathbf{x}_N$ is a random sample from $f(\mathbf{x};\mathbf{w})$. In typical applications the function $\hat{D}$ in (3.14) is convex and differentiable with respect to $\mathbf{v}$, in which case the solution of (3.14) may be readily obtained by solving the following system of equations:

$$\frac{1}{N} \sum_{i=1}^{N} \left[I_{\{S(\mathbf{x}_i)\geq\gamma\}} W(\mathbf{x}_i;\mathbf{u},\mathbf{w}) \nabla \ln f(\mathbf{x}_i;\mathbf{v}) \right] = 0. \tag{3.15}$$

The advantage of this approach is that the solution of (3.15) can often be calculated analytically. In particular, this happens if the distributions of the random variables belong to a natural exponential family (NEF).

We have to note that the CE program (3.14) or (3.15) are useful only if the probability of the target event $\{S(\mathbf{x}) \geq \gamma\}$ is not too small under $\mathbf{w}$, say greater than 10^{-5}. For rare-event probabilities, due to the rareness of the events $\{S(\mathbf{x}_i) \geq \gamma\}$, most of the indicator random variables $I_{\{S(\mathbf{x}_i)\geq\gamma\}},\ i = 1, \cdots, N$, will be zero, for moderate N. It makes the program (3.14) and (3.15) difficult to carry out. A multilevel algorithm can be used to overcome this difficulty. The basic idea is to construct a sequence of reference parameters $\{\mathbf{v}_t,\ t \geq 0\}$ and a sequence of levels $\{\gamma_t,\ t \geq 1\}$, and iterate in both $\mathbf{v}_t$ and γ_t.

We initialize by choosing a not very small ϱ, say $\varrho = 10^{-2}$ and by defining $\mathbf{v}_0 = \mathbf{u}$. Next, we let γ_1 ($\gamma_1 < \gamma$) be such that, under the original density $f(\mathbf{x}; \mathbf{u})$, the probability $\ell_1 = \mathbb{E}_{\mathbf{u}} I_{\{S(\mathbf{x}_i)\geq\gamma_1\}}$ is at least ϱ. We then let $\mathbf{v}_1$ be the optimal CE reference parameter for estimating ℓ_1, and repeat the last two steps iteratively with the goal of estimating the pare $\{\ell, \mathbf{v}^*\}$. In other words, each iteration of the algorithm consists of two main phases. In the first phase γ_t is updated, in the second $\mathbf{v}_t$ is updated. Specifically, starting with $\mathbf{v}_0 = \mathbf{u}$ we obtain the subsequent γ_t and $\mathbf{v}_t$ as follows:

1. **Adaptive updating of** γ_t For a fixed $\mathbf{v}_{t-1}$, let γ_t be a $(1-\varrho)$-quantile of $S(\mathbf{x})$ under $\mathbf{v}_{t-1}$. That is, γ_t satisfies

$$P_{\mathbf{v}_{t-1}}(S(\mathbf{x}) \geq \gamma_t) \geq \varrho, \tag{3.16}$$

$$P_{\mathbf{v}_{t-1}}(S(\mathbf{x}) \leq \gamma_t) \geq 1 - \varrho, \tag{3.17}$$

   where $\mathbf{x} \sim f(\mathbf{x}; \mathbf{v}_{t-1})$.

   A simple estimator $\hat{\gamma}_t$ of γ_t can be obtained by drawing a random sample $\mathbf{x}_1, \cdots, \mathbf{x}_N$ from $f(\mathbf{x}; \mathbf{v}_{t-1})$, calculating the performances $S(\mathbf{x}_i)$ for all i, ordering them from smallest to biggest: $S_{(1)} \leq \cdots \leq S_{(N)}$ and finally, evaluating the sample $(1-\varrho)$-

quantile as

$$\hat{\gamma_t} = S_{(\lceil (1-\varrho)N \rceil)} \tag{3.18}$$

Note that $S_{(j)}$ is called the j-th order-statistic of the sequence $S(\mathbf{x}_1), \cdots, S(\mathbf{x}_N)$. Note also that $\hat{\gamma}_t$ is chosen such that the event $\{S(\mathbf{x}) \geq \hat{\gamma}_t\}$ is not too rare (it has a probability of around ϱ), and therefore updating the reference parameter via a procedure such as (3.18) is not void of meaning.

2. **Adaptive updating of $\mathbf{v}_t$** For fixed γ_t and $\mathbf{v}_{t-1}$, derive $\mathbf{v}_t$ from the solution of the following CE program

$$max_{\mathbf{v}} D(\mathbf{v}) = \mathbb{E}_{\mathbf{v}_{t-1}} \left[I_{\{S(\mathbf{x}) \geq \gamma_t\}} W(\mathbf{x}; \mathbf{u}, \mathbf{v}_{t-1}) \ln f(\mathbf{x}; \mathbf{v}) \right]. \tag{3.19}$$

The stochastic counterpart of the above equation is as follows: for fixed $\hat{\gamma}_t$ and $\hat{\mathbf{v}}_{t-1}$, derive $\hat{\mathbf{v}}$ from the solution of following program

$$\max_{\mathbf{v}} \hat{D}(\mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^{N} \left[I_{\{S(\mathbf{x}_i) \geq \hat{\gamma}_t\}} W(\mathbf{x}_i; \mathbf{u}, \hat{\mathbf{v}}_{t-1}) \ln f(\mathbf{x}_i; \mathbf{v}) \right]. \tag{3.20}$$

Thus, at the first iteration, starting with $\hat{\mathbf{v}}_0 = \mathbf{u}$, to get a good estimate for $\hat{\mathbf{v}}_1$, the target event is artificially made less rare by (temporarily) using a level $\hat{\gamma}_1$ which is chosen smaller than γ. The value of $\hat{\mathbf{v}}_1$ obtained in this way will (hopefully) make the event $\{s(\mathbf{x}) \geq \gamma\}$ less rare in the next iteration, so in the next iteration a value $\hat{\gamma}_2$ can be used which is closer to γ itself. The algorithm terminates when at some iteration t a level is reached which is at least γ and thus the original value of γ can be used without getting too few samples.

The above rationale results in the following algorithm:

1. Define $\hat{\mathbf{v}}_0 = \mathbf{u}$. Set $t = 1$.

2. Generate a sample $\mathbf{z}_1, \cdots, \mathbf{x}_N$ from the density $f(\mathbf{x}; \mathbf{v}_{t-1})$ and compute the sample $(1-\varrho)$-quantile $\hat{\gamma}_t$ of the performances according to (3.18), provided $\hat{\gamma}_t$ is less than γ. Otherwise set $\hat{\gamma}_t = \gamma$.

3. Use the same sample $\mathbf{x}_1, \cdots, \mathbf{x}_N$ to solve the stochastic program (3.20). Denote the solution by $\hat{\mathbf{v}}_t$.

4. If $\hat{\gamma}_t < \gamma$, set $t = t + 1$ and reiterate from Step 2. Else proceed with Step 5.

5. Estimate the rare-event probability ℓ using the LR estimate

$$\hat{\ell} = \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\mathbf{x}_i) \geq \gamma\}} W(\mathbf{x}_i; \mathbf{u}, \hat{\mathbf{v}}_T) \tag{3.21}$$

where T denotes the final number of iterations.

3.3 The CE-Method for Optimization Problem

Consider the following general maximization problem: Let $\mathcal{X}$ be a finite set of states, and let S be a real-valued performance function on $\mathcal{X}$. We wish to find the maximum of S over $\mathcal{X}$ and the corresponding state at which this maximum is attained. Let us denote the maximum by γ^*. Thus,

$$S(\mathbf{x}^*) = \gamma^* = \max_{\mathbf{x} \in \mathcal{X}} S(\mathbf{x}). \tag{3.22}$$

The starting point in the methodology of the CE method is to associate with the optimization problem (3.22) a meaningful estimation problem. To this end we define a collection of indicator functions $\left\{I_{\{S(\mathbf{x}) \geq \gamma\}}\right\}$ on $\mathcal{X}$ for various levels $\gamma \in \mathbb{R}$. Next, let $\{f(\cdot; \mathbf{v}), \mathbf{v} \in \mathcal{V}\}$ be a family of (discrete) probability densities on $\mathcal{X}$, parameterized by a real-valued parameter (vector) $\mathbf{v}$. For a certain $\mathbf{u} \in \mathcal{V}$ we associate with (3.22) the problem of estimating the number

$$\ell(\gamma) = P_{\mathbf{u}}(S(\mathbf{x}) \geq \gamma) = \sum_{\mathbf{x}} I_{\{S(\mathbf{x}) \geq \gamma\}} f(\mathbf{x}; \mathbf{u}) = \mathbb{E}_{\mathbf{u}} I_{\{S(\mathbf{x}) \geq \gamma\}}, \tag{3.23}$$

where $P_{\mathbf{u}}$ is the probability measure under which the random state $\mathbf{x}$ has probability density function (pdf) $f(\mathbf{x}; \mathbf{u})$, and $\mathbb{E}_{\mathbf{u}}$ denotes the corresponding expectation operator. We will call the estimation problem (3.23) the *associated stochastic problem* (ASP). To indicate how (3.23) is associated with (3.22), suppose for example that γ is equal to γ^*

and that $f(\mathbf{x};\mathbf{u})$ is the uniform density on $\mathcal{X}$. Note that, typically, $\ell(\gamma^*) = f(\mathbf{x}^*;\mathbf{u}) = 1/|\mathcal{X}|$ where $|\mathcal{X}|$ denotes the number of elements in $\mathcal{X}$ is a very small number. Thus, for $\gamma = \gamma^*$ a natural way to estimate $\ell(\gamma)$ would be to use the LR estimator (3.21) with reference parameter $\mathbf{v}^*$ given by

$$\mathbf{v}^* = \arg\max_{\mathbf{v}} \mathbb{E}_{\mathbf{u}} \left[I_{\{S(\mathbf{x})\geq\gamma\}} \ln f(\mathbf{x};\mathbf{v}) \right]. \tag{3.24}$$

This parameter could be estimated by

$$\hat{\mathbf{v}}^* = \arg\max_{\mathbf{v}} \frac{1}{N} \left[I_{\{S(\mathbf{x}_i)\geq\gamma\}} \ln f(\mathbf{x}_i;\mathbf{v}) \right] \tag{3.25}$$

where the $\mathbf{x}_i$ are generated from pdf $f(\mathbf{x};\mathbf{u})$. It is plausible that, if γ is close to γ^*, that $f(\mathbf{x};\mathbf{v}^*)$ assigns most of its probability mass close to $\mathbf{x}^*$, and thus can be used to generate an approximate solution to (3.22). However, it is important to note that the estimator (3.25) is only of practical use when $I_{\{S(\mathbf{x})\geq\gamma\}} = 1$ for enough samples. This means for example that when γ is close to γ^*, $\mathbf{u}$ needs to be such that $P_{\mathbf{u}}(S(\mathbf{x}) \geq \gamma)$ is not too small. Thus, the choice of $\mathbf{u}$ and γ in (3.22) are closely related. On the one hand we would like to choose γ as close as possible to γ^*, and find (an estimate of) $\mathbf{v}^*$ via the procedure above, which assigns almost all mass to state(s) close to the optimal state. On the other hand, we would like to keep γ relative large in order to obtain an accurate estimator for $\mathbf{v}^*$.

The situation is very similar to the rare-event simulation case. The idea is to adopt a two-phase multilevel approach in which we simultaneously construct a sequence of levels $\hat{\gamma}_1, \hat{\gamma}_2, \cdots, \hat{\gamma}_T$ and parameter (vectors) $\hat{\mathbf{v}}_0, \hat{\mathbf{v}}_1, \cdots, \hat{\mathbf{v}}_T$ such that $\hat{\gamma}_T$ is close to the optimal γ^* and $\hat{\mathbf{v}}_T$ is such that the corresponding density assigns high probability mass to the collection of states that give a high performance.

This strategy is embodied in the following procedure:

1. Define $\hat{\mathbf{v}}_0 = \mathbf{u}$. Set $t = 1$.

2. Generate a sample $\mathbf{x}_1, \cdots, \mathbf{x}_N$ from the density $f(\mathbf{x};\mathbf{v}_{t-1})$ and compute the sample $(1-\varrho)$-quantile $\hat{\gamma}_t$ of the performance according to (3.18).

3. Use the same sample $\mathbf{x}_1, \cdots, \mathbf{x}_N$ and solve the stochastic program (3.20) with $W = 1$. Denote the solution by $\hat{\mathbf{v}}_t$.

5. If for some $t \geq d$, say $d = 5$,

$$\hat{\gamma}_t = \hat{\gamma}_{t-1} = \cdots = \hat{\gamma}_{t-d}, \tag{3.26}$$

then stop (let T denote the final iteration); otherwise set $t = t + 1$ and reiterate from Step 2.

Note that the initial vector $\hat{\mathbf{v}}_0$, the sample size N, the stopping parameter d, and the number ϱ have to be specified in advance.

The above procedure can, in principle, be applied to any discrete and continuous optimization problem. For each individual problem two essential ingredients need to be supplied:

1. We need to specify how the samples are generated. In other words, we need to specify the family of densities $\{f(\cdot; \mathbf{v})\}$.

2. We need to calculate the updating rules for the parameters, based on cross-entropy minimization.

In general there are many ways to generate samples from $\mathcal{X}$, and it is not always immediately clear which way of generating the sample will yield better results or easier updating formulas.

3.4 Updating Rules of Some Useful Densities

In this section we will derive the updating rules for two pdfs which are commonly used for the CE method. The first one is the Bernoulli distribution and the second is the Gaussian distribution.

Suppose the random vector $\mathbf{x}_i = (x_{i1}, \cdots, x_{in}) \sim Ber(\mathbf{p})$ where $Ber(\mathbf{p})$ is Bernoulli distribution with parameter $\mathbf{p} = (p_1, \cdots, p_n)$. Consequently, the pdf is

$$f(\mathbf{x}_i; \mathbf{p}) = \prod_{j=1}^{n} p_i^{x_{ij}} (1 - p_i)^{1 - x_{ij}}, \tag{3.27}$$

and since each x_{ij} can only be 0 or 1,

$$\begin{aligned} \frac{\partial}{\partial p_j} \ln f(\mathbf{x}_i; \mathbf{p}) &= \frac{x_{ij}}{p_j} - \frac{1 - x_{ij}}{1 - p_j} \\ &= \frac{1}{(1 - p_j) p_j} (x_{ij} - p_j). \end{aligned} \tag{3.28}$$

Now we can find the maximum in (3.20) (with $W = 1$) by setting the first derivatives with respect to p_j equal to zero, for $j = 1, \cdots, n$:

$$\frac{\partial}{\partial p_j} \sum_{i=1}^{N} I_{\{S(\mathbf{x}_i) \geq \gamma\}} \ln f(\mathbf{x}_i; \mathbf{p}) = \frac{1}{(1 - p_j) p_j} \sum_{i=1}^{N} I_{\{S(\mathbf{x}_i) \geq \gamma\}} (x_{ij} - p_j) = 0. \tag{3.29}$$

Thus, we get the updating rule

$$p_j = \frac{\sum_{i=1}^{N} I_{\{S(\mathbf{x}_i) \geq \gamma\}} x_{ij}}{\sum_{i=1}^{N} I_{\{S(\mathbf{x}_i) \geq \gamma\}}}. \tag{3.30}$$

Next, consider the Gaussian density

$$f(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2} \frac{(x-\mu)^2}{\sigma^2}}, \ x \in \mathbb{R}. \tag{3.31}$$

The optimal solution of (3.20) (with $W = 1$) follows from minimization of

$$\frac{1}{\sigma^2} \sum_{i=1}^{N} I_i (x_i - \mu)^2 + \ln(\sigma^2) \sum_{i=1}^{N} I_i, \tag{3.32}$$

where $I_i = I_{\{S(x_i) \geq \gamma\}}$. It is easily seen that this minimum is obtained at $(\hat{\mu}, \hat{\sigma}^2)$ given by

$$\hat{\mu} = \frac{\sum_{i=1}^{N} I_i x_i}{\sum_{i=1}^{N} I_i} \tag{3.33}$$

and

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{N} I_i (x_i - \hat{\mu})^2}{\sum_{i=1}^{N} I_i} \tag{3.34}$$

Chapter 4

Stochastic Erasure-Only List Decoding of RS codes

In this chapter, we apply the Cross-Entropy (CE) method [33] to develop a Monte Carlo based iterative SDD algorithm which renders an improved algebraic SDD decoding performance. The CE method is an elegant practical principle for simulating rare events which approximates the probability of the rare event by means of a family of parameterized probabilistic models. Our stochastic erasure-only list decoding (SEOLD) algorithm uses the extended CE method for optimization problem by considering an optimal event as a rare event.

4.1 Preliminary

Let $\mathbf{C}$ be an (n, k) RS code over GF(2^m) with minimum Hamming distance $d_{min} = n - k + 1$. Let $\mathbf{c} = (c_0, \cdots, c_{n-1})$ be a codeword in $\mathbf{C}$. For binary transmission, every code symbol must be expanded into binary with symbols from GF(2) $= \{0, 1\}$. Let α be primitive in GF(2^m), then the ith symbol c_i can be uniquely represented by the binary m-tuple $c_i^{(b)} = (c_{i,0}, \cdots, c_{i,m-1})$ where $c_i = c_{i,0}\alpha^0 + \cdots + c_{i,m-1}\alpha^{m-1}$, $\forall c_{i,j} \in$ GF(2). Therefore, the codeword $\mathbf{c}$ can be uniquely mapped into the binary expansion vector $\bar{\mathbf{c}} = (c_0^{(b)}, c_1^{(b)}, \cdots, c_n^{(b)}) = (\bar{c}_0, \bar{c}_1, \cdots, \bar{c}_{nm-1})$.

Using binary phase-shift-keying (BPSK), the transmitter maps the binary imaged

codeword $\bar{\mathbf{c}}$ into the bipolar vector

$$\Psi(\bar{\mathbf{c}}) = \bar{\mathbf{x}} = (\bar{x}_0, \cdots, \bar{x}_{nm-1}),\ \bar{x}_j = \Psi(\bar{c}_j) = (-1)^{\bar{c}_j} \tag{4.1}$$

and sends it over an additive white Gaussian noise (AWGN) channel with zero mean and power spectral density $N_0/2$. The received sequence at the output of the matched filter is $\bar{\mathbf{y}} = (\bar{y}_0, \cdots, \bar{y}_{nm-1})$ where $\bar{y}_j = \bar{x}_j + \bar{w}_j$ and $\bar{w}_j$'s are statistically independent Gaussian random variables with zero mean and variance $N_0/2$.

Let $\bar{\mathbf{z}} = (\bar{z}_0, \cdots, \bar{z}_{nm-1})$ be the hard decision binary vector of the received bit sequence $\bar{\mathbf{y}}$, i.e.,

$$\bar{z}_j = \begin{cases} 0, & \bar{y}_j > 0 \\ 1, & \text{otherwise} \end{cases} \tag{4.2}$$

and $\mathbf{z} = (z_0, \cdots, z_{n-1})$ be the corresponding symbol vector. Denoted by $\bar{\Gamma} = (\bar{\gamma}_1, \cdots, \bar{\gamma}_{nm-1})$ the reliability vector of $\bar{\mathbf{y}}$ in which $\bar{\gamma}_j$ is the magnitude of the log-likelihood ratio (LLR) associated with the corresponding hard-limited bit $\bar{z}_j$

$$L\left(\bar{c}_j\right) = \log \frac{P\left(\bar{c}_j = 0 | \bar{\mathbf{y}}\right)}{P\left(\bar{c}_j = 1 | \bar{\mathbf{y}}\right)}, \tag{4.3}$$

and define the symbol reliability vector $\Gamma = (\gamma_0, \cdots, \gamma_{n-1})$ of $\mathbf{z}$ by

$$\gamma_i = \min_j \bar{\gamma}_j, \quad j \in \{im, \cdots, (i+1)m - 1\} \tag{4.4}$$

Assume that the ith symbol c_i of $\mathbf{c}$ is uniformly distributed over GF(2^m) and the n received symbols are independent and uniformly drawn from GF(2^m). Then $P(c_i = \beta|\bar{\mathbf{y}})$, the probability that $c_i = \beta$ was transmitted given the observation $\bar{\mathbf{y}}$ can be easily evaluated [21]:

$$\begin{aligned} P(c_i = \beta|\bar{\mathbf{y}}) &= P(c_i = \beta|\bar{\mathbf{y}}_i) \\ &= \frac{P(\bar{\mathbf{y}}_i|c_i = \beta)P(c_i = \beta)}{\sum_{\omega \in \mathrm{GF}(2^m)} P(\bar{\mathbf{y}}_i|c_i = \omega)P(c_i = \omega)} \\ &= \frac{P(\bar{\mathbf{y}}_i|c_i = \beta)}{\sum_{\omega \in \mathrm{GF}(2^m)} P(\bar{\mathbf{y}}_i|c_i = \omega)} \end{aligned} \tag{4.5}$$

where

$$\begin{aligned} \bar{\mathbf{y}}_i &= (\bar{y}_{im}, \bar{y}_{im+1}, \cdots, \bar{y}_{im+m-1}), \\ P(\bar{\mathbf{y}}_i | c_i = \beta) &= \prod_{j=0}^{m-1} P\left(\bar{y}_{im+j} \,|\bar{c}_{im+j} = \beta_j\right), \quad \beta = \beta_0 \alpha^0 + \cdots + \beta_{m-1}\alpha^{m-1}. \end{aligned}$$

The $q \times n$ matrix $\mathbf{R} = [\mathbf{R}_{\beta i} = P(c_i = \beta|\bar{\mathbf{y}})]$, $q = 2^m$, will be referred to as the reliability matrix of the received vector $\bar{\mathbf{y}}$.

4.2 Stochastic List Decoding Algorithm

4.2.1 Algebraic Erasures-Only (EO) Decoding

It is well-known that RS codes are maximum-distance separable (MDS) which implies that any k coordinates (symbols) in an RS codeword can be used to determine the remaining $n-k$ symbols. Hence it is sufficient to decide k correct (message) or $n-k$ incorrect (error) coordinates of a codeword. Let $\mathbf{E}_L$ be the collection of all combinations of $n-k$ error coordinates,

$$\mathbf{E}_L = \left\{ \mathbf{s} = (s_0, \cdots, s_{n-1}) \,\middle|\, s_i \in \{0,1\}, \sum_i s_i = n-k \right\} \tag{4.6}$$

where $s_i = 1$ if the ith coordinate is in error. Then a straightforward decoding schedule is given as below:

(a). For all $\mathbf{s} \in \mathbf{E}_L$, erase the corresponding $n-k$ error coordinates of the received word $\mathbf{z}$ and decode by the erasures-only (EO) decoder. The resulting codeword set is denoted by $\mathbf{C}_\mathbf{z}$.

(b). Choose the codeword from $\mathbf{C}_\mathbf{z}$ with the best score, e.g., the one whose Euclidean distance from the received word is the smallest, as the decoder output.

The basic idea of the above procedure is shown in Fig. 4.1. It can be easily confirmed that for any $\mathbf{c} \in \mathbf{C}_\mathbf{z}$, $d_H(\mathbf{c}, \mathbf{z}) \leq n-k$, where $d_H(\mathbf{c}, \mathbf{z})$ is the Hamming distance between $\mathbf{c}$ and $\mathbf{z}$. Therefore, the transmitted codeword belongs to $\mathbf{C}_\mathbf{z}$ if the number of error

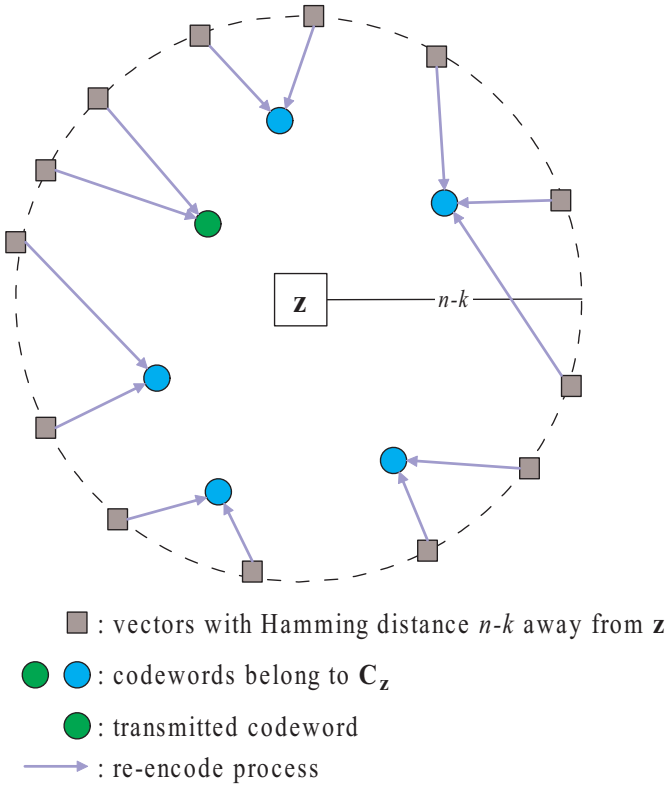

Figure 4.1: Idea of the algebraic erasures-only decoding.

symbols is less than $d_{\min}$. Furthermore, **(b)** is equivalent to the following minimization problem

$$\arg\min_{\mathbf{c}} d\left(\Psi(\bar{\mathbf{c}}), \bar{\mathbf{y}}\right) \text{subject to } \mathbf{c} \in \mathbf{C_z} \tag{4.7}$$

where $\Psi(\cdot)$ is defined by (1) and $d(\bar{\mathbf{a}}, \bar{\mathbf{b}})$ is the Euclidean distance (ED) between the nm-ary real vectors $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$.

4.2.2 A Stochastic List Decoding Idea

Each *error locator vector* (ELV) $\mathbf{s} \in \mathbf{E}_L$ represents a particular set of $n-k$ possible error coordinates and has a corresponding codeword $\mathbf{c_s}$ that belongs to $\mathbf{C_z}$. We denote the latter relationship by $\mathbf{s} \mapsto \mathbf{c_s}$. Although more than one ELV may be associated with the same codeword, the complexity of searching for the optimal solution $\mathbf{c}^*$ in the error location domain $\mathbf{E}_L$ is still extremely high because the cardinality of $\mathbf{E}_L$ is $\binom{n}{k}$ and only a few (or one) elements in $\mathbf{E}_L$, depending on the number and locations of the received errors, can be used to reconstruct $\mathbf{c}^*$.

Suppose we model the selection of the ELV $\mathbf{s}$ from $\mathbf{E}_L$ as a stochastic (vector-valued) experiment governed by a family of parameterized distributions $\{f(\mathbf{s};\mathbf{u})\}$ with $\mathbf{u} \in \nu$ being a real-valued parameter vector. Usually $f(\mathbf{s};\mathbf{u})$ is assumed to be uniformly distributed due to the lack of priori information whence the search in (4.7) is exhaustive unless some algebraic properties of the code are used. One way to solve (4.7) efficiently is to find a parameter $\mathbf{v}^*$ such that $f(\mathbf{s};\mathbf{v}^*) = \delta(\mathbf{s}-\mathbf{s}^*)$ where $\mathbf{s}^* \mapsto \mathbf{c}^*$. Then drawing one sample from $f(\mathbf{s};\mathbf{v}^*)$ is sufficient to obtain the optimal solution $\mathbf{c}^*$. To get around the difficulty that $\mathbf{c}^*$ is not known, one notices that the optimization problem (4.7) is related to the estimation of the probability $P(d(\Psi(\bar{\mathbf{c}}_\mathbf{s}), \bar{\mathbf{y}}) \leq \eta | \mathbf{s} \mapsto \mathbf{c_s})$, which is a rare event when $\eta = \eta^* = d(\Psi(\bar{\mathbf{c}}^*), \bar{\mathbf{y}})$. The connection comes from the fact that efficient estimation of a rare event can be achieved by the method of importance sampling and in this case the optimal importance density is $f(\mathbf{s};\mathbf{v}^*)$. Without the knowledge of the threshold η^*, we start with a proper importance density $f(\mathbf{s};\hat{\mathbf{v}})$ to generate samples of $\mathbf{s}$

and compute an initial estimate $\hat{\eta}$ for η. Ideally, we can use those drawn samples which satisfy $d(\Psi(\bar{\mathbf{c}}_{\mathbf{s}}), \bar{\mathbf{y}}) \leq \hat{\eta}$ to obtain new parameter value $\hat{\mathbf{v}}'$ such that $f(\mathbf{s}; \hat{\mathbf{v}}')$ is closest to $f(\mathbf{s}; \mathbf{v}^*)$ in the Kullback-Leibler (KL) sense, i.e., the CE between $f(\mathbf{s}; \hat{\mathbf{v}}')$ and $f(\mathbf{s}; \mathbf{v}^*)$ is minimized. Since $\mathbf{v}^*$ is unknown, we choose $\hat{\mathbf{v}}'$ such that $f(\mathbf{s}; \hat{\mathbf{v}}')$ is closest to the empirical distribution of $\mathbf{s}$ in those samples that are generated by $f(\mathbf{s}; \hat{\mathbf{v}})$ and satisfy $d(\Psi(\bar{\mathbf{c}}_{\mathbf{s}}), \bar{\mathbf{y}}) \leq \hat{\eta}$ for this empirical distribution is likely to be a good approximation of $f(\mathbf{s}; \mathbf{v}^*)$. New samples of $\mathbf{s}$ are then produced by the updated importance density $f(\mathbf{s}; \hat{\mathbf{v}}')$ to compute new estimate $\hat{\eta}'$. This iterative procedure continues until $|\hat{\eta} - \hat{\eta}'|$ is less than a predetermined threshold.

The above method is known as the CE method [33] which is an iterative procedure that consists of the following two phases in each iteration.

- Generate samples from the specified importance density given by the parameters from the previous iteration.
- Update the parameters for next iteration according to the order of the score values associated with the drawn samples and the minimizing CE criterion.

Based on the above discussion, we propose a generic Monte Carlo based SDD algorithm as shown in Fig. 4.2 and in Table 4.1 with some detailed description given in Section 4.3 and 4.4.

4.2.3 Convergence and Complexity

Different convergence conditions and results have been discussed for the deterministic CE method and its extensions in [35] where it is also proved that convergence in distribution or η can be guaranteed but needs a proper tuning of the parameters of the algorithm such as the number of samples N, number of elites E, and smoothing factor ρ. Convergence to the global minimum is ensured only if a large sample size N is used. On the other hand, the computing complexity is related to N and is given by $O(N(n-k)^2)$. It is obvious

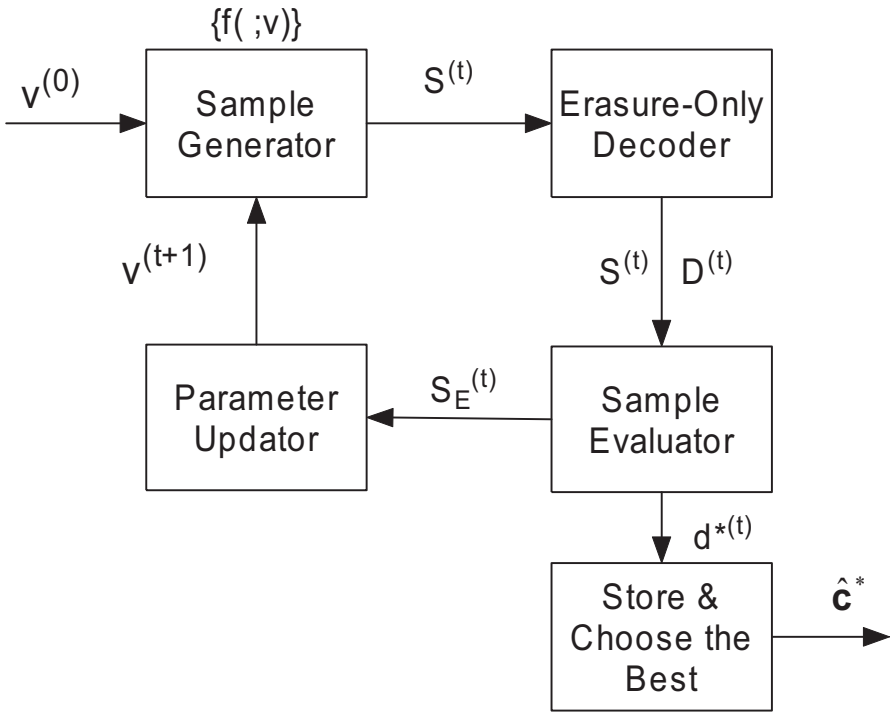

Figure 4.2: Flow chart of a stochastic decoder for RS codes.

1. Define a family of probability densities $\{f(\cdot;\mathbf{v}), \mathbf{v} \in \nu\}$ on the search space $\mathbb{R}^{nm}$. Initialize $\mathbf{v}^{(0)}$. Set $t = 1$.
2. Generate a sample set $\mathbf{S}^{(t)}$ whose N random vector samples are drawn from $f(\cdot;\mathbf{v}^{(t)})$. Regard the magnitudes as bit LLRs and convert them into symbol reliabilities. Erase the $n-k$ least reliable symbols and decode the received word by EO decoding.
3. Evaluate Euclidean distances between the decoded codewords and the received word. Select the E vector samples with best metrics as the new elite set $\mathbf{S_E}^{(t)} \subset \mathbf{S}^{(t)}$ and store the best decoded codeword $\mathbf{d}^{*(t)}$ in $\mathbf{D}^{(t)}$.
4. Evaluate the new parameter $\mathbf{v}^{(t+1)}$ by solving
$$\mathbf{v}^{(t+1)} = \arg\max_{\mathbf{v}} \frac{1}{|\mathbf{S_E}^{(t)}|} \sum_{\mathbf{s}_\ell^{(t)} \in \mathbf{S_E}^{(t)}} \ln f(\mathbf{s}_\ell^{(t)};\mathbf{v}).$$
Update $\mathbf{v}^{(t+1)}$ via $\mathbf{v}^{(t+1)} = \rho\mathbf{v}^{(t+1)} + (1-\rho)\mathbf{v}^{(t)}$ where $0 < \rho < 1$.
5. Terminate decoding if the stopping criterion is met. Choose the best codeword from the list $\{\mathbf{d}^{*(t)};\ \forall\, t\}$, say $\hat{\mathbf{c}}^*$, as the decoder output. Otherwise increase t by 1 and return to step 2.

Table 4.1: A Stochastic List Decoding Algorithm.

that the decoding performance can be improved by using a larger N. As we retain the best sample at the end of each iteration, the decoding performance is also improved by increasing the iteration number T. As an early-stopping at any iteration produces a decoded codeword, we say the algorithm converges if the sequence of decoded codewords converges. With a modest N, we found that the decoded codewords converge to the same codeword within 10 iterations in most cases. Our algorithm yields good performance although uniform convergence in distribution or η within a limited iterations is not guaranteed.

4.3 List Decoding via Erasure Location Estimation

In this section, we propose an novel algorithm to solve the discrete optimization problem described in (4.7) by utilizing the stochastic list decoding idea. This algorithm is named as the first kind stochastic erasure-only list decoding (SEOLD-I) algorithm which is used to efficiently estimate the most possible $d_{min} - 1$ locations of erasures about the received word $\mathbf{z}$.

4.3.1 Importance Density and Sample Format

Suppose the reliability matrix $\mathbf{R}$ is known at the receiver. Define the distrust function, $f_d : \mathrm{GF}(2^m) \mapsto (0, \infty)$, of the ith coordinate z_i of $\mathbf{z}$ as

$$f_d(z_i) = \frac{\sum_\beta \mathbf{R}_{\beta,i}}{\mathbf{R}_{z_i,i}}, \ \beta \in \mathrm{GF}(2^m) \setminus \{z_i\} \tag{4.8}$$

The larger the value of $f_d(z_i)$ is, the higher the probability that z_i should be erased at the decoder.

Let $\mathbf{s} = (s_0, \cdots, s_{n-1})$ be a random vector where $s_0, \cdots, s_{n-1}$ are independent Bernoulli random variables with success probabilities $p_0, \cdots, p_{n-1}$, i.e., $P(s_i = 1) = 1 - P(s_i = 0) = p_i$. We write $\mathbf{s} \sim \mathrm{Ber}(\mathbf{p})$, where $\mathbf{p} = (p_0, \cdots, p_{n-1})$. In our case, $s_i = 1$ represents the ith symbol z_i of $\mathbf{z}$ should be erased. On the other hand, z_i will be reserved because of higher reliability when $s_i = 0$.

The initial parameters $\mathbf{p}^{(0)} = \left(p_0^{(0)}, \cdots, p_{n-1}^{(0)}\right)$ are defined as

$$p_i^{(0)} = \begin{cases} 1-\epsilon, & f_d(z_i) > 1-\epsilon \\ f_d(z_i), & \text{otherwise} \end{cases} \tag{4.9}$$

where ϵ is an arbitrary real value between (0,1).

At the tth iteration, let $\mathbf{s}_1^{(t)}, \mathbf{s}_2^{(t)}, \cdots, \mathbf{s}_N^{(t)}$ be N trials drawn from Ber $\left(\mathbf{p}^{(t)}\right)$ which satisfy

$$\sum_{j=0}^{n-1} s_{\ell,j}^{(t)} = d_{min} - 1, \quad \forall \ell \in \{1, \cdots, N\} \tag{4.10}$$

where $\mathbf{s}_\ell^{(t)} = (s_{\ell,0}^{(t)}, \cdots, s_{\ell,n-1}^{(t)})$. We then collect samples from these N trials to form a sample set $\mathbf{S}^{(t)} = \{\mathbf{s}_1^{(t)}, \cdots, \mathbf{s}_N^{(t)}\}$.

4.3.2 Update Parameters

Let $\mathbf{d}_1^{(t)}, \cdots, \mathbf{d}_N^{(t)}$ be the output codewords of the EO decoder. We compute the ED between each candidate codeword and the received word $\mathbf{y}$ and sort the corresponding random vectors according to the descending order of their associated EDs. Define the elite set $\mathbf{S_E}^{(t)}$ at the tth iteration to be the E vectors with the smallest EDs to $\mathbf{y}$, i.e., the corresponding codewords are more likely to have been transmitted. We always store the best one in $\mathbf{S_E}^{(t)}$ up to the current iteration for the final decision when the maximum number of iteration is reached.

Suppose the parameters used to generate samples at the tth iteration are $\mathbf{p}^{(t)} = \left(p_0^{(t)}, \cdots, p_{n-1}^{(t)}\right)$. The parameters for the next iteration are updated by considering the information provided by both $\mathbf{p}^{(t)}$ and $\mathbf{S_E}^{(t)}$. More precisely, the ith parameter $p_i^{(t+1)}$ is obtained by [33]

$$p_i^{(t+1)} = (1-\rho)p_i^{(t)} + \rho \cdot \frac{\sum_{\ell \in \mathbf{S_E}^{(t)}} s_{\ell,i}^{(t)}}{E} \tag{4.11}$$

where ρ is a smoothing factor with real value between $(0, 1)$.

4.4 List Decoding via Virtual Received Words

In Step 2 of Table 4.1 we try to find the most likely message/error coordinates such that the associated EO-decoded codeword is closest to the received vector. Note that the random samples are used to determine the erasure locations only, and the searching sphere of the algebraic list decoding described in Section 4.2.1 is always centered at the hard-limited received word $\mathbf{z}$ with radius equals to $n-k$. To increase our search range and improve decoding performance, we include some extra codewords which lie statistically in a small neighborhood of the received word in our expanded search, such that some of them may in fact be closer in ED to the true transmitted codeword $\mathbf{c}$; see Fig. 4.3. More specifically, the expansion is accomplished by eliminating the requirement that the search be centered at $\mathbf{z}$. Instead, we randomized the search center by EO-decoding the hard-decision versions of the drawn sample vectors which we refer to as virtual received words. If the importance density does converge to the desired density $\delta(\mathbf{s}-\mathbf{s}^*)$, such an expanded search will eventually contract and converge to the true transmitted codeword.

4.4.1 Importance Density and Sample Format

Let $\bar{\mathbf{s}} = (\bar{s}_0, \cdots, \bar{s}_{nm-1})$ be a random vector where $\bar{s}_0, \cdots, \bar{s}_{nm-1}$ are independent Gaussian random variables with means $\mu_0, \cdots, \mu_{nm-1}$ and variances $\sigma_0^2, \cdots, \sigma_{nm-1}^2$. We write $\bar{\mathbf{s}} \sim \mathcal{N}(\vec{\mu}, \vec{\sigma})$, where $\vec{\mu} = (\mu_0, \cdots, \mu_{nm-1})$ and $\vec{\sigma} = (\sigma_0, \cdots, \sigma_{nm-1})$ are initialized by

$$\mu_j^{(0)} = \bar{\gamma}_j \tag{4.12}$$

$$\sigma_j{}^{(0)} = \sqrt{|\bar{\gamma}_j|} \tag{4.13}$$

At the tth iteration, N random samples $\bar{\mathbf{s}}_1^{(t)}, \bar{\mathbf{s}}_2^{(t)}, \cdots, \bar{\mathbf{s}}_N^{(t)}$ are drawn from $\mathcal{N}\left(\vec{\mu}^{(t)}, \vec{\sigma}^{(t)}\right)$ to form the sample set $\bar{\mathbf{S}}^{(t)}$. Each sample vector represents the bit reliabilities of an associated virtual received word. By using (4.4) to convert the bit reliabilities into symbol reliability, the $n-k$ coordinates with smallest symbol reliabilities are erased; the

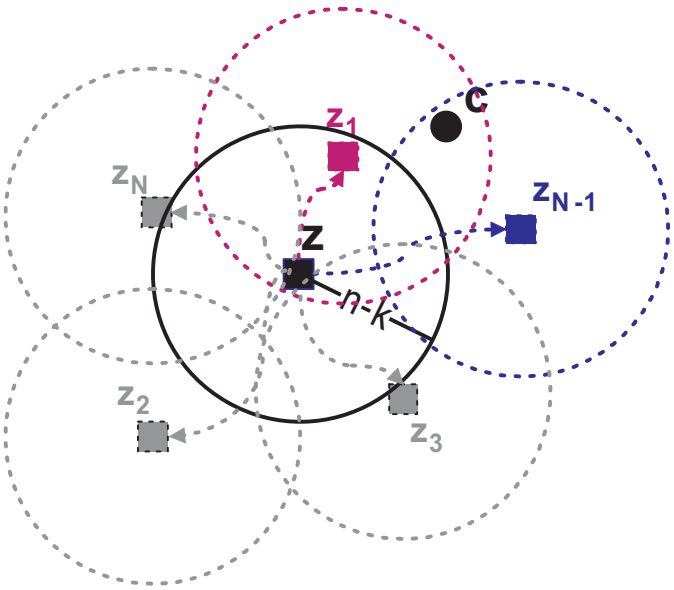

Figure 4.3: Virtual received words are generated around the received LLR vector $\bar{\Gamma}$ by hard-limiting the sample vectors generated by an importance probability density whose parameter values evolved according to the CE principle.

remaining bit positions are hard-limited, mapped into symbol decisions and the resulting virtual received word is then decoded by an EO decoder.

4.4.2 Update Parameters

Let $\mathbf{d}_1^{(t)}, \cdots, \mathbf{d}_N^{(t)}$ be the output codewords of the EO decoder. We compute the ED between each candidate codeword and the received word $\mathbf{y}$ and sort the corresponding random vectors according to the descending order of their associated EDs. Define an elite set $\bar{\mathbf{S}}_{\mathbf{E}}^{(t)}$ which includes E vectors with the smallest EDs to $\mathbf{y}$, i.e., the corresponding codewords are more likely to have been transmitted. We always store the best one in $\bar{\mathbf{S}}_{\mathbf{E}}^{(t)}$ up to the current iteration for the final decision when the maximum number of iteration is reached.

Then the two sets of parameters $\vec{\mu}^{(t+1)}$ and $\vec{\sigma}^{(t+1)}$ are updated by [33]

$$\mu_j^{(t+1)} = (1-\lambda)\mu_j^{(t)} + \lambda \cdot \frac{\sum_{\bar{\mathbf{s}}_\ell^{(t)} \in \bar{\mathbf{S}}_{\mathbf{E}}^{(t)}} \bar{s}_{\ell,j}^{(t)}}{E} \tag{4.14}$$

and

$$\sigma_j^{(t+1)} = (1-\eta)\sigma_j^{(t)} + \eta \cdot \frac{\sum_{\bar{\mathbf{s}}_\ell^{(t)} \in \bar{\mathbf{S}}_{\mathbf{E}}^{(t)}} \left(\bar{s}_{\ell,j}^{(t)} - \mu_j^{(t+1)} \right)^2}{E} \tag{4.15}$$

where λ and η are real values between $(0, 1)$ used to smooth the variation of these parameters. The algorithm described in this section is called the second kind stochastic erasures-only listing decoding (SEOLD-II) algorithm.

4.5 Experimental Results and Discussions

In this section, some simulated performance of two SEOLD algorithms (SEOLD-I and SEOLD-II) are presented and compared with that of other well known RS decoding algorithms. A standard binary input AWGN channel is assumed over which the BPSK modulated codewords are transmitted. We model the receive matched filter output as the sum of a $\pm 1-$valued sequence and Gaussian sequence with zero-mean i.i.d. components. The average performance bounds on the ML error probability of RS codes over an AWGN channel developed in [36] are used as the performance lower limits.

Due to the complexity and the decoding delay considerations, the SEOLD algorithms will not terminate until convergence is assured. Instead, we limit our decoding procedure to T iterations in all simulations.

Fig. 4.4 shows the codeword error rate (CER) performance of the (15,11) RS code over an AWGN channel. HDD-BM refers to the performance of a hard decision bounded minimum distance decoder such as the BM algorithm. GMD and KV refer to the performance of the GMD algorithm proposed by Forney and the algebraic soft decision decoding algorithm proposed by Koetter and Vardy, respectively. Note that the KV algorithm concerned here is infinite interpolation costs, i.e., the complexity is also infinite. For both SEOLD-I and SEOLD-II, the size of the sample set N and the size of the elite set E at every iteration are set to be 20 and 6, respectively. After 10 iterations, SEOLD-I has about 0.5 dB and 0.3 dB coding gain over GMD and KV at a CER of 10^{-5},

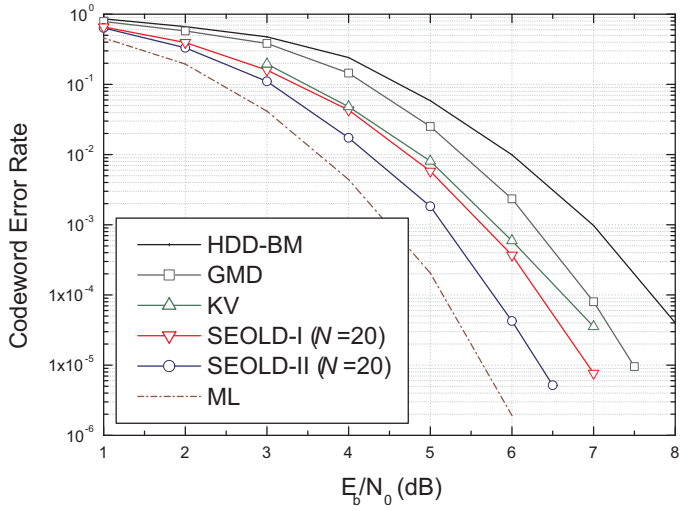

Figure 4.4: Codeword error probability performance of the (15,11) Reed-Solomon code; 10 iterations.

respectively. On the other hand, SEOLD-II outperforms all the previous algorithms with a performance gain of about 1.2 dB and 1.0 dB over GMD and KV at a CER of 10^{-5}.

In Fig. 4.5, SEOLD-I has a near KV performance when the $N = 100$ and $E = 10$ after 10 iterations. At the same condition, SEOLD-II still outperforms the other algorithms with reasonable complexity. The SEOLD-II has about 0.6 dB and 1.0 dB coding gain over KV where N is equal to 100 and 500, respectively. In conclusion, the proposed decoding algorithms are capable of offering good performance with modest complexity for short high rate RS codes. Its performance can be further improved by increasing the sample size N and/or the maximum iteration number T at the cost of increased decoding complexity.

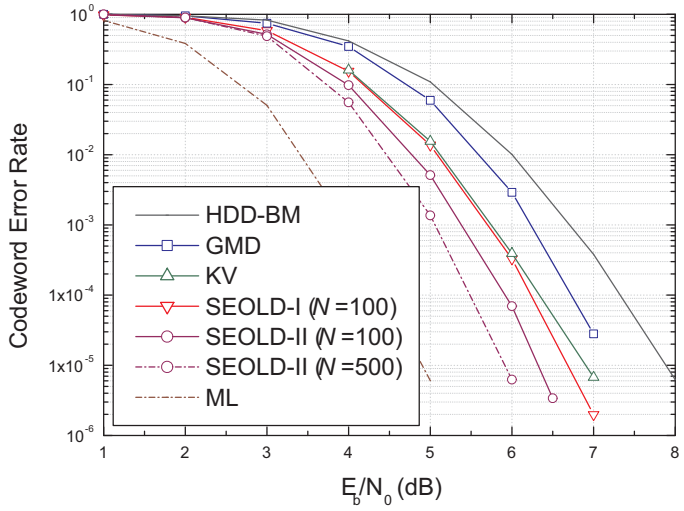

Figure 4.5: Codeword error probability performance of the (31,25) Reed-Solomon code; 10 iterations.

Chapter 5

Stochastic List Decoding of Linear Block Codes

In the previous chapter, we focus on decoding RS codes of short to medium length. The MDS character of RS codes is exploited to reduce the complexity of locating near-by codewords. Such an approach cannot be applied to general linear codes. We thus present a new decoding method which is valid for arbitrary linear codes but is more effective for codes with small girth.

5.1 Preliminary

Let $\mathbf{C}$ be a binary (N, K) linear block code with minimum distance d_{min} and $M \times N$ parity-check matrix $\mathbf{H}$. As the rows of $\mathbf{H}$ may be dependent, we have $M > N - K$. Let $I = \{1, \cdots, N\}$ and $J = \{1, \cdots, M\}$ be the sets of column indices and row indices of $\mathbf{H}$, respectively. We denote the set of bits n that participate in check m by $\mathcal{N}(m) = \{n : \mathbf{H}_{mn} = 1\}$. Similarly, we define the set of checks in which bit n participates as $\mathcal{M}(n) = \{j : \mathbf{H}_{mn} = 1\}$. We denote a set $\mathcal{N}(m)$ with bit n excluded by $\mathcal{N}(m)\backslash n$, and a set $\mathcal{M}(n)$ with parity check m excluded by $\mathcal{M}(n)\backslash m$. The cardinality of $\mathcal{N}(m)$ and $\mathcal{M}(n)$ are denoted by $|\mathcal{N}(m)|$ and $|\mathcal{M}(n)|$, respectively. Let $\mathbf{e}_n$ be a $1 \times N$ elementary vector with 1 at position n and 0 at other entries.

An $1 \times N$ vector $\mathbf{c}$ is a codeword of $\mathbf{C}$ if and only if $\mathbf{c}\mathbf{H}^T = \mathbf{0}$ where $\mathbf{H}^T$ is the

transpose of $\mathbf{H}$ and $\mathbf{0}$ is a $1 \times M$ zero vector. For each row $\mathbf{h}_m$ of $\mathbf{H}$, $m \in J$, let

$$\mathbf{C}_m = \{\mathbf{c} \in \{0,1\}^N : \mathbf{c}\mathbf{h}_m^T = 0 \bmod 2\}, \tag{5.1}$$

then

$$\mathbf{C} = \bigcap_{m=1}^{M} \mathbf{C}_m. \tag{5.2}$$

Using the binary phase-shift-keying (BPSK) signal, the transmitter maps a codeword $\mathbf{c}$ into the bipolar vector

$$\Psi(\mathbf{c}) = \mathbf{x} = (x_0, \cdots, x_{N-1}),\ x_n = \Psi(c_n) = (-1)^{c_n} \tag{5.3}$$

and sends it over an additive white Gaussian noise (AWGN) channel with zero mean and power spectral density $N_0/2$ W/Hz. The received sequence at the output of the matched filter is given by $\mathbf{y} = (y_0, \cdots, y_{N-1})$, where $y_n = x_n + w_n$ and w_n's are statistically independent Gaussian random variables with zero mean and variance $N_0/2$.

Let $\mathbf{z} = (z_0, \cdots, z_{N-1})$ be the hard decision version of the received sequence $\mathbf{y}$, i.e.,

$$z_n = \begin{cases} 0, & y_n > 0 \\ 1, & \text{otherwise} \end{cases} \tag{5.4}$$

For $m \in J$, we define σ_m as the result of check sum-m based on the hard-decision vector $\mathbf{z}$:

$$\sigma_m = \left[\sum_{n \in \mathcal{N}(m)} z_n \mathbf{H}_{mn} \right] \pmod 2 \tag{5.5}$$

and define $\Sigma = (\sigma_1, \cdots, \sigma_M)$ as the syndrome vector.

Denoted by $\Gamma = (\gamma_n, \cdots, \gamma_{N-1})$ the reliability vector of $\mathbf{y}$ in which γ_n is the magnitude of the log-likelihood ratio (LLR) associated with the corresponding hard-limited bit z_n

$$L_n = \log \frac{P(c_n = 0 | \mathbf{y})}{P(c_n = 1 | \mathbf{y})}. \tag{5.6}$$

We also denote $\mathcal{L} = (L_1, \cdots, L_N)$ as the LLR vector of the received word.

Let λ_m be the reliability of check sum m which is defined as

$$\lambda_m = \min_{n \in \mathcal{N}(m)} \gamma_n \tag{5.7}$$

Then we first sort $\{\lambda_m : m \in J\}$ and let $m_1, m_2, \cdots, m_M$ denote the position of the check sums in terms of descending order of $\{\lambda_m : m \in J\}$, i.e., the check sum m_1 is the most reliable and m_M is the least reliable.

Define $q_n = P(z_n \neq c_n|\mathbf{y})$ as the *a posteriori* probability that bit n is in error based on $\mathbf{y}$. Then we have the following lemmas.

Lemma 5.1 *For the AWGN channel model considered, the probability q_n can be expressed as*

$$q_n = \frac{1}{1 + e^{\gamma_n}} \tag{5.8}$$

Proof :

See Appendix A.

Lemma 5.2 *The probability that for check sum $m \in \mathcal{M}(n)$, the sum of all bits $n' \in \mathcal{N}(m)\backslash n$ mismatches the transmitted bit n', say r_{mn}, is*

$$r_{mn} = \frac{1}{2}\left(1 - \prod_{n' \in \mathcal{N}(m)\backslash n} (1 - 2q_{n'})\right). \tag{5.9}$$

Proof :

See [37].

Note that r_{mn} represents the probability of having an odd number of errors $\mathcal{N}(m)\backslash n$. Define $\tilde{q}_n$ as the *a posteriori* probability that bit n is in error based on the results of the check sums intersecting in position-n. Then we obtain the following useful theorem.

Theorem 5.1 *Given the received word* $\mathbf{y}$ *and the syndrome set* $\Sigma_n \equiv \{\sigma_m : m \in \mathcal{M}(n)\}$, *the logarithm of the bit correctness probability ratio for bit* n, *say* ξ_n, *is*

$$\begin{aligned} \xi_n &= \log\left[\frac{1-\tilde{q}_n}{\tilde{q}_n}\right] = \log\left[\frac{P(z_n = c_n|\mathbf{y}, \Sigma_n)}{P(z_n \neq c_n|\mathbf{y}, \Sigma_n)}\right] \\ &\cong \gamma_n + \sum_{m\in\mathcal{M}(n)} \left[(1-2\sigma_m)\left(\min_{n'\in\mathcal{N}(m)\backslash n} \gamma_{n'}\right)\right] \end{aligned} \tag{5.10}$$

Proof :

See Appendix B.

5.2 Sequential Bit-Flipping Algorithm

In this section, we introduce a single-run sequential bit-flipping (SBF) algorithm for transforming $\mathbf{z}$ into a valid codeword. This procedure has a special constraint about the parity-check matrix $\mathbf{H}$ that $\mathbf{H}$ has to be a systematic form. First of all, consider the rows of the parity-check matrix $\mathbf{H}$ are linearly independent, i.e., $M = N - K$. Using appropriate row operations, $\mathbf{H}$ can be transformed into a systematic form, say $\tilde{\mathbf{H}} = [\mathbf{I}_M \mathbf{P}]$, where $\mathbf{I}_M$ is an $M \times M$ identity matrix and $\mathbf{P}$ is an $M \times (N-M)$ binary matrix. Note that both $\mathbf{H}$ and $\tilde{\mathbf{H}}$ are the null space of $\mathbf{C}$, hence we can decode the received word by using $\tilde{\mathbf{H}}$ instead of $\mathbf{H}$.

However, it is impossible to have this transformation when the rows of $\mathbf{H}$ are linearly dependent, i.e., $M > N - K$. Fortunately, we can remove $M - N + K$ rows of $\mathbf{H}$ which can be represented by the linear combination of the remain rows to get a $(N-K) \times N$ sub-matrix $\mathbf{H}'$ where $\mathbf{H}'$ has its systematic form $\tilde{\mathbf{H}}'$.

Example 5.1 *Consider the following parity check matrix:*

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_5 \end{bmatrix} \tag{5.11}$$

Since $\mathbf{h}_5 = \mathbf{h}_1 + \cdots + \mathbf{h}_4$*, we can remove* $\mathbf{h}_1$ *from* $\mathbf{H}$ *and get the following sub-matrix*

$$\mathbf{H}' = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{5.12}$$

where $\mathbf{H}'$ *is itself a systematic form.*

Note that it is easy to confirm that $\tilde{\mathbf{H}}'$ is still the null space of $\mathbf{C}$. Therefore, without lost of generality, we assume that $\mathbf{H}$ is always a systematic form in this chapter for simplicity.

Remind that $\mathbf{C} = \bigcap_{m \in J} \mathbf{C}_m$, i.e., a codeword $\mathbf{c}$ also belongs to subcodes $\mathbf{C}_m$ for all $m \in J$. The idea of the SBF algorithm is to modify $\mathbf{z}$ sequentially such that the final result is a valid codeword. Specifically, the SBF algorithm separates the original problem into M sub-problems and solves these sub-problems sequentially in terms of an arbitrary order of $\{1, \cdots, M\}$, denoted as $\mathbf{o} = (o(1), \cdots, o(M))$. The procedure must ensure that the solution of the m-th sub-problem also satisfy the constraints of previous $(m-1)$ sub-problems. Along the process of the procedure, a sequence of vectors $\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_M$ are produced where

$$\mathbf{d}_t \in \cap_{m=1}^{t} \mathbf{C}_{o(m)}, \ 1 \leq t \leq M. \tag{5.13}$$

and $\mathbf{d}_M$ is obviously a valid codeword. In general, the SBF algorithm needs to input a predetermined order $\mathbf{o}$ and the LLR vector $\mathcal{L}$ at the beginning. At the end of the procedure, a valid codeword $\mathbf{d} = (d_1, \cdots, d_N)$ and an associated new LLR vector $\hat{\mathcal{L}} = (\hat{L}_1, \cdots, \hat{L}_N)$ are the outputs. The difference between $\hat{\mathcal{L}}$ and $\mathcal{L}$ is given by

$$\begin{cases} \hat{L}_n = L_n & \text{if } d_n = z_n \\ \hat{L}_n = -L_n & \text{if } d_n \neq z_n \end{cases} \tag{5.14}$$

where $\hat{\mathcal{L}}$ is useful for the stochastic decoding algorithm described in Section 5.5.

Next, we formulate the detailed procedure of the SBF algorithm as below:

1. Let $\mathbf{d}_0$ be the hard limiting vector of $\mathcal{L}$, $\hat{\mathcal{L}} = \mathcal{L}$, and $I_0 = \{\phi\}$. Set $t = 0$.

2. Let $I_t = I_{t-1} \cup \mathcal{N}(o(t))$. If $\mathbf{d}_{t-1} \in \mathbf{C}_{o(t)}$, let $\mathbf{d}_t = \mathbf{d}_{t-1}$. Otherwise, find the solution, say n^*, of

$$\arg \min_{n \in \{I_t \backslash I_{t-1}\}} \xi_n \tag{5.15}$$

where ξ_n is evaluated by (5.10). Let

$$\begin{aligned} \mathbf{d}_t &\leftarrow \mathbf{d}_{t-1} + \mathbf{e}_{n^*} \pmod 2, & (5.16)\\ \hat{L}_{n^*} &\leftarrow -\hat{L}_{n^*}, & (5.17)\\ \sigma_m &\leftarrow \sigma_m + 1 \pmod 2 \ \forall m \in \mathcal{M}(n^*), & (5.18)\\ t &\leftarrow t + 1. \end{aligned}$$

3. If $t = M$, stop the procedure and output both $\mathbf{d} = \mathbf{d}_M$ and $\hat{\mathcal{L}}$. Otherwise, go to Step 2.

We denote the relationship between the inputs and outputs of the above procedure as $(\mathbf{d}, \hat{\mathcal{L}}) = \Omega(\mathbf{o}, \mathcal{L})$ for simplicity.

Remark 5.1 *We have to mention that once the number of error bits in $\{I_t \backslash I_{t-1}\}$ is greater than or equal to 2, the output codeword $\mathbf{d}_M$ won't be the transmitted codeword.*

Example 5.2 *Consider a (8,4) linear block code with parity check matrix:*

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{5.19}$$

Suppose all zero codeword is transmitted and the received word $\mathbf{y}$ is given by

$$\mathbf{y} = (1.83, 2.07, 2.36, -0.21, 1.05, 1.91, -0.09, 1.63).$$

Then the hard limiting vector $\mathbf{z}$ is

$$\mathbf{z} = (0, 0, 0, 1, 0, 0, 1, 0),$$

and an example of the SRSBFP is shown in Fig. 5.1 where the order of check sums are $3 \to 2 \to 1 \to 4$.

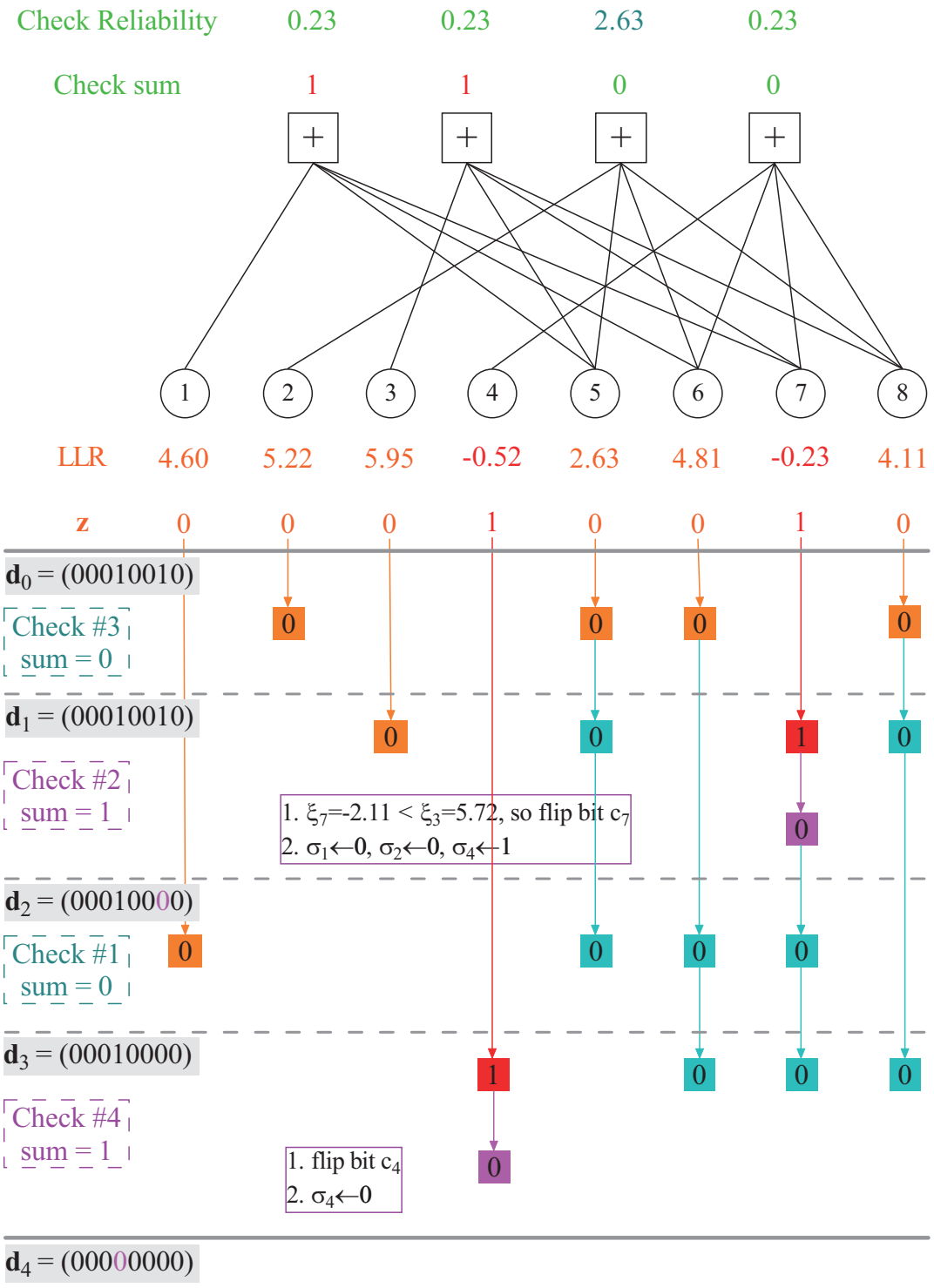

Figure 5.1: An example of the SRSBFP.

5.3 Predicament of Decoding via SBF algorithm

We have introduced a sequential bit flipping procedure, say SRF algorithm, in previous section. It is a simple unified framework for transforming the hard limiting vector $\mathbf{z}$ into a codeword in $\mathbf{C}$. Note that different order may induce different codeword to be produced. For instance, the output codewords in Example 5.2 and 5.3 are different because of different orders although they face the same received word $\mathbf{y}$.

Example 5.3 *Consider the same case described in Example 5.2. If we change the order from* $3 \to 2 \to 1 \to 4$ *to* $4 \to 3 \to 2 \to 1$*, the output codeword* $\mathbf{d}$ *will become* $(1, 0, 1, 1, 0, 0, 1, 0)$.

We observe that output codeword $\mathbf{d}$ in Example 5.2 is equal to the transmitted codeword but in Example 5.3 is not. It is because the order used in Example 5.3 meets the situation described in Remark 5.1. Obviously, it is a big problem if we want to decode by SBF algorithm. Therefore, we try to solve this problem by the following two ideas:

1. Find appropriate order to avoid the situation described in Remark 5.1.
2. Correct some error bits in advance such that the number of orders which can decode the correct codeword increases.

Note that the first idea is impractical because the complexity of finding appropriate order grows quickly as M increases. Besides, the hardware implementation is inefficient if the order changes frequently. Consequently, we propose two modified methods for decoding based on the SBF algorithm with a fixed order. The first one is designed for cyclic codes that we apply the SBF algorithm to transform all of the cyclic shifted received word into valid codewords. Note that cyclic shifting the received word is similar to decode in different order even though we don't change the order actually. The another method is to implement the second idea based on the concept of the randomized sphere decoding with moving center which can correct errors iteratively.

5.4 SBF Algorithm with Cyclic Shifts

Assume a codeword $\mathbf{c}_T$ belongs to a cyclic code $\mathbf{C}$ is transmitted and let $\mathcal{L} = (L_1, \cdots, L_N)$ be the LLR vector of the received word. Define $\mathcal{L}^\nu = (L_{\nu-1}, L_{\nu-2}, \cdots, L_\nu)$ as the cyclic shifted version of $\mathcal{L}$ by ν positions. Then we can obtain a set of candidates by the following algorithms:

1. Determine an order $\mathbf{o}$ for the SBF algorithm.

2. For all $\nu \in J$, apply the SBF algorithm for $\mathcal{L}^\nu$ and $\mathbf{o}$ to obtain a set of candidates $D = \{\mathbf{d}^1, \cdots, \mathbf{d}^N\}$ where
$$(\mathbf{d}^\nu, \hat{\mathcal{L}}^\nu) = \Omega(\mathbf{o}, \mathcal{L}^\nu).$$

The transmitted codeword is then estimated by

$$\hat{\mathbf{c}}_T = \arg\min_{\mathbf{c}\in D} d(\Psi(\mathbf{c}), \mathbf{y}), \tag{5.20}$$

where $d(\mathbf{a}, \mathbf{b})$ is the Euclidean distance between $\mathbf{a}$ and $\mathbf{b}$.

5.5 Stochastic Sequential Bit Flipping Algorithm

Ideally, we can transform $\mathbf{z}$ into the transmitted codeword through the SBF algorithm if the appropriate order is found. In fact, such order is hard to find, especially when the LLRs are unreliable. Therefore, we don't want to decode based on the original LLR vector $\mathcal{L}$ at all times but hope to gradually change $\mathcal{L}$ such that its hard limiting vector is more and more close to the transmitted codeword. In order to implement this idea, we use the similar method illustrated in Section 4.4 which is an iterative procedure with the following two phases:

1. Generate N_s virtual LLR vectors around the original one according to a specific random mechanism. Then decode them by the SBF algorithm to get N_s candidates.

2. Update the parameters of the random mechanism based on E_s better candidates in order to generate better virtual LLR vectors in next iteration.

Note that this is the basic idea of our randomized sphere decoding with moving center. Next, we will illustrate the random mechanism further in next two subsections.

5.5.1 Importance Density and Sample Format

Let $\mathbf{s} = (s_1, \cdots, s_N)$ be a random vector where $s_1, \cdots, s_N$ are independent Gaussian random variables with means $\mu_1, \cdots, \mu_N$ and variances $\rho_1^2, \cdots, \rho_N^2$. We write $\mathbf{s} \sim \mathbb{N}(\vec{\mu}, \vec{\rho})$, where $\vec{\mu} = (\mu_1, \cdots, \mu_N)$ and $\vec{\rho} = (\rho_1, \cdots, \rho_N)$ are initialized by

$$\mu_n^{(0)} = L_n \tag{5.21}$$

$$\rho_n^{(0)} = \frac{4}{N_0} \tag{5.22}$$

At the tth iteration, N_s random samples $\mathbf{s}_1^{(t)}, \mathbf{s}_2^{(t)}, \cdots, \mathbf{s}_{N_s}^{(t)}$ are drawn from $\mathbb{N}\left(\vec{\mu}^{(t)}, \vec{\rho}^{(t)}\right)$ to form the sample set $\mathbf{S}^{(t)}$. Each sample vector represents the LLRs of an associated virtual received word. We decode them by the SBF algorithm based on an pre-designed order and obtain sets of candidates $\mathbf{d}_\ell^{(t)}$ and associated LLR vectors $\hat{\mathbf{s}}_\ell^{(t)} = (\hat{s}_{\ell,1}, \cdots, \hat{s}_{\ell,N})$ for $1 \leq \ell \leq N_s$.

5.5.2 Update Parameters

Let $\mathbf{d}_1^{(t)}, \cdots, \mathbf{d}_{N_s}^{(t)}$ be the output codewords of the SBF algorithm. We compute the ED between each candidate codeword and the received word $\mathbf{y}$ and sort the corresponding random vectors according to the descending order of their associated EDs. Define an elite set $\mathbf{E}^{(t)}$ which includes E_s vectors with the smallest EDs to $\mathbf{y}$, i.e., the corresponding codewords are more likely to have been transmitted. We always store the best one in $\mathbf{E}^{(t)}$ up to the current iteration for the final decision when the maximum number of iteration is reached.

Then the two sets of parameters $\vec{\mu}^{(t+1)}$ and $\vec{\rho}^{(t+1)}$ are updated by [33]

$$\mu_n^{(t+1)} = (1-\delta)\mu_n^{(t)} + \delta \cdot \frac{\sum_{\hat{\mathbf{s}}_\ell^{(t)} \in \mathbf{E}^{(t)}} \hat{s}_{\ell,n}^{(t)}}{E_s} \tag{5.23}$$

and

$$\rho_n^{(t+1)} = (1-\varepsilon)\rho_n^{(t)} + \varepsilon \cdot \frac{\sum_{\hat{\mathbf{s}}_\ell^{(t)} \in \mathbf{E}^{(t)}} \left(\hat{s}_{\ell,n}^{(t)} - \mu_n^{(t+1)}\right)^2}{E_s} \tag{5.24}$$

where δ and ε are real values between $(0, 1)$ used to smooth the variation of these parameters.

5.5.3 Stochastic Sequential Bit Flipping Algorithm

The detailed stochastic sequential bit flipping algorithm (SSBFA) is summarized as follows.

1. Initialize $\vec{\mu}^{(0)}$ and $\vec{\rho}^{(0)}$ by (5.21) and decide an order $\mathbf{o}$. Set $t = 0$.

2. Generate a set of random samples $\mathbf{S}^{(t)} = \{\mathbf{d}_1^{(t)}, \cdots, \mathbf{d}_{N_s}^{(t)}\}$ from $\mathbb{N}\left(, \vec{\mu}^{(t)}, \vec{\rho}^{(t)}\right)$.

3. For each sample, we have $(\mathbf{d}_\ell^{(t)}, \hat{\mathbf{d}}_\ell^{(t)}) = \Omega(\mathbf{o}, \mathbf{d}_\ell^{(t)})$.

4. Evaluate Euclidean distances between the $\mathbf{d}_\ell^{(t)}$ and the received word $\mathbf{y}$. Select the E_s samples with best metrics as the new elite set $\mathbf{E}^{(t)} \subset \mathbf{S}^{(t)}$ and store the best decoded codeword $\mathbf{d}^{*(t)}$ in $\mathbf{D}^{(t)}$.

5. Evaluate the new parameters $\vec{\mu}^{(t+1)}$ and $\vec{\rho}^{(t+1)}$ by (5.23) and (5.24), respectively.

6. If $t = T$ of for some $t \geq c$, say $c = 3$,

$$\mathbf{d}^{*(t)} = \mathbf{d}^{*(t-1)} = \cdots = \mathbf{d}^{*(t-c)}, \tag{5.25}$$

   then stop; otherwise set $t = t + 1$ and reiterate from Step 2.

5.6 Experimental Results and Discussions

In the first part of this section, some simulated performance of the proposed algorithm, say SSBFA, are presented and compared with that of traditional bounded-distance decoding (BDD) and the sum-product algorithm (SPA). A standard binary input AWGN channel is assumed over which the BPSK modulated codewords are transmitted. We model the receive matched filter output as the sum of a $\pm 1-$valued sequence and Gaussian sequence with zero-mean i.i.d. components.

The maximum iteration number of both SPA and SSBFA are set to be 50. But we will stop the proposed algorithm earlier if the best of the output candidates are the same for consecutive 5 iterations. In our simulations, SSBFA terminates quickly and the average iteration number of is slightly more than five. Besides, the sample size N_s is set to be 10 and the elite size E_s is 1. Therefore, the computational complexity of both algorithms in our simulation is approximately at the same level.

Fig. 5.2 - 5.6 are simulation results of five high rate block codes with HDPC matrix which is in order (15,11) Hamming code, (7,5) RS code, (22,16) single error correction (SEC) code, (39,32) SEC code, and (72,64) SEC code. The SSBFA has about 0.5 dB - 0.8 dB coding gain over SPA at a bit error rate (BER) of 10^{-4} under approximately same complexity.

Next, we consider two examples of cyclic codes, (31,26) BCH codes and (15,11) RS codes. For the (31,26) BCH code, we compare our SBF algorithm, SBF with cyclic shifts (CSSBF) and SSBFA with BDD and SPA. As shown in Fig. 5.7, the performance of the SBF algorithm with a fixed order is worse than BDD and SPA because of the phenomenon described in Remark 5.1 may happen frequently. However, two kinds of modified algorithms, SSBFA and CSSBF, have almost the same improved decoding performance and outperform the other decoding methods. In other words, these modified algorithms can greatly reduce the phenomenon described in Remark 5.1. For the (15,11) RS code, similar decoding performance can be observed in Fig. 5.8. Our proposed

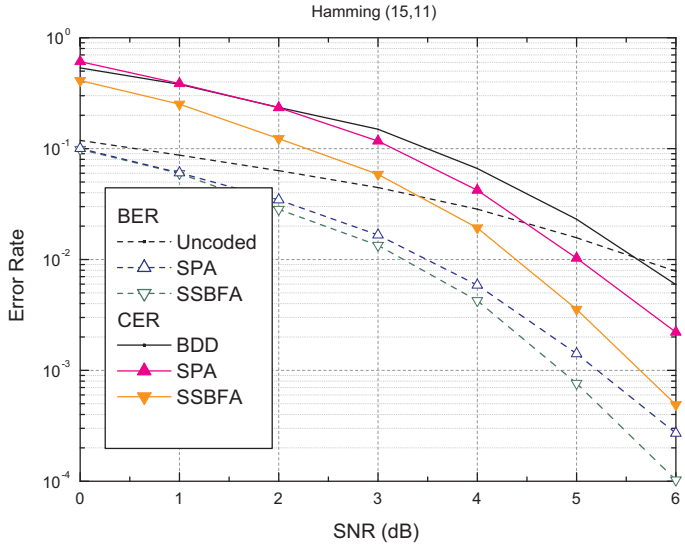

Figure 5.2: Error rate performance of the (15,11) Hamming Code; $N_s = 10$, $E_s = 1$

algorithms still outperform the other decoding methods including the BDD, SPA, Chase-II algorithm with 16 test patterns, and the KV algorithm with infinite multiplicity. Note that the number of test patterns of the Chase-II algorithm is set to 16 due to our CSSBF for (15,11) RS code has 15 cyclic shifted LLR vectors.

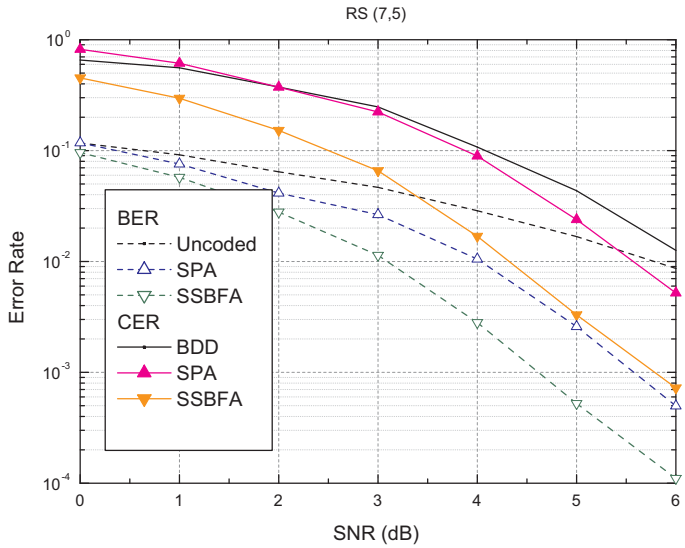

Figure 5.3: Error rate performance of the (7,5) RS Code; $N_s = 10$, $E_s = 1$

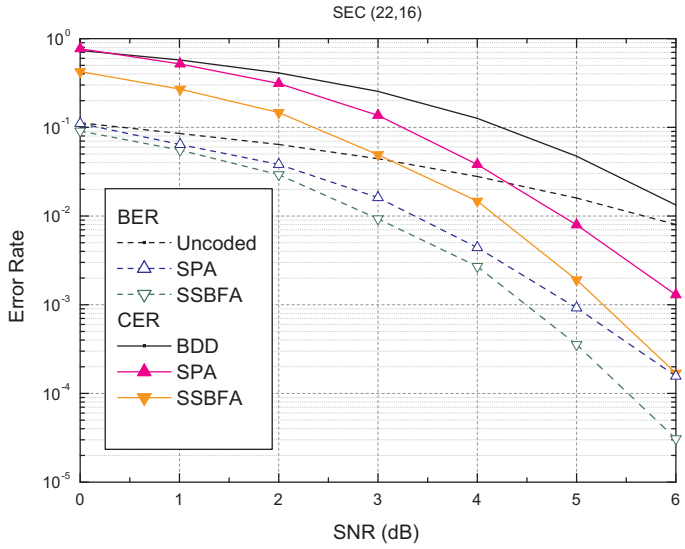

Figure 5.4: Error rate performance of the (22,16) single error correction Code; $N_s = 10$, $E_s = 1$

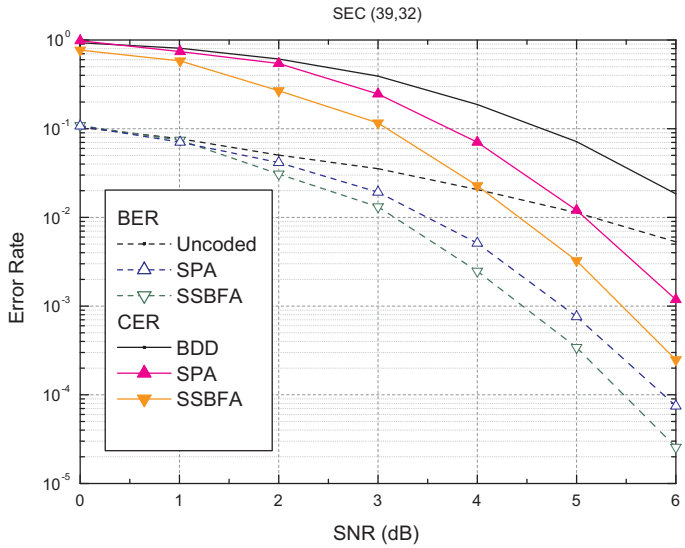

Figure 5.5: Error rate performance of the (39,32) single error correction Code; $N_s = 10$, $E_s = 1$

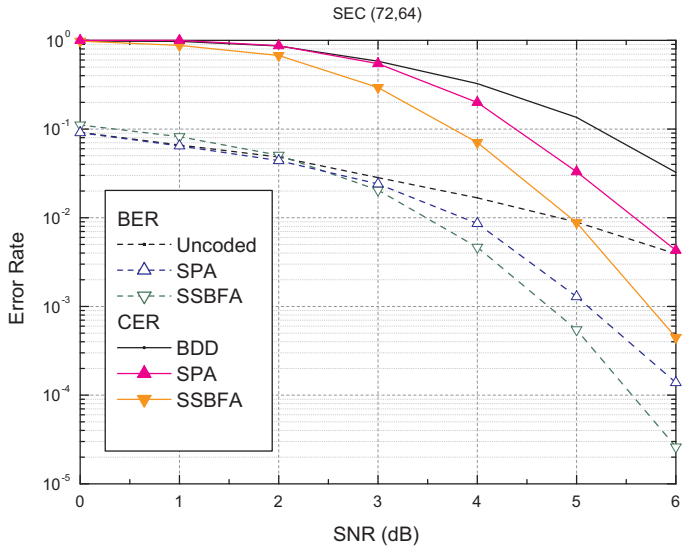

Figure 5.6: Error rate performance of the (72,64) single error correction Code; $N_s = 10$, $E_s = 1$

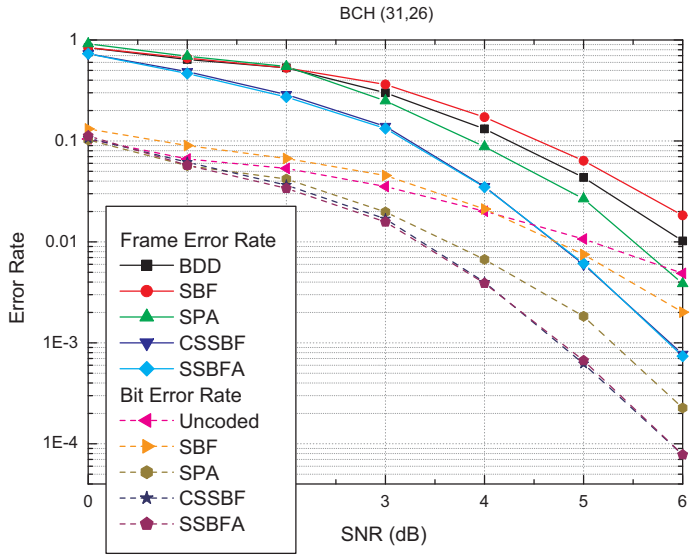

Figure 5.7: Error rate performance of the (31,26) BCH Code.

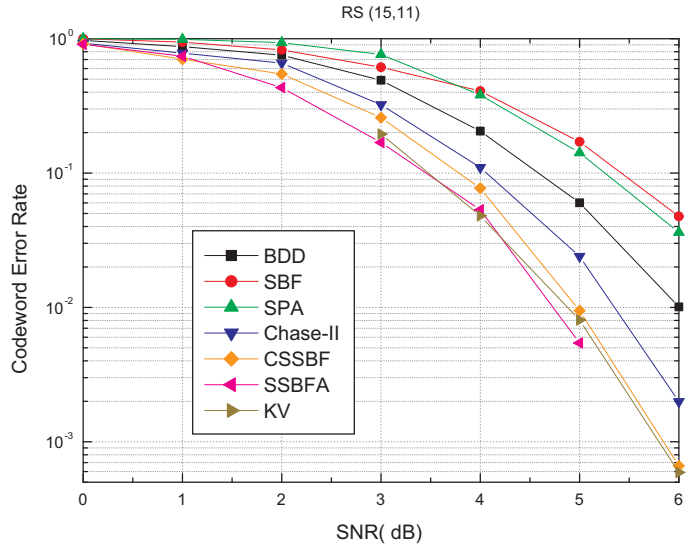

Figure 5.8: Error rate performance of the (15,11) RS Code.

Chapter 6

Conclusions

We have derived generalized message passing rules and developed appropriate schemes to relax the deterministic constraints. Higher order energy approximations and the method proposed in [1] can be extended by the proposed ABP scheme as well. Furthermore, the same relaxation idea can also be used to generalize the trellis structures for turbo decoding algorithms.

We have also presented several novel stochastic list decoding algorithms for linear block codes with high density parity-check (HDPC) matrices. We apply an iterative Monte Carlo based approach called the Cross-Entropy method to produce a list of candidates at each iteration. Hopefully, the set of candidates will converges to the singular ML codeword. Basically, the CE method helps to estimate the probability distribution of the candidate transmitted codeword by examining the current available sample set and find the distribution which is closest to the optimal one in the CE sense. In a sense, we perform sphere decoding with randomized decoding radius. Based on the maximum-distance separable property of the Reed-Solomon (RS) codes, two stochastic erasures-only list decoding algorithms (SEOLD-I and SEOLD-II) are proposed. SEOLD-I estimates the possible erasures locations and then recover the associated codeword by erasures-only decoder (EOD). In order to enhance the performance of SEOLD-I, SEOLD-II utilizes the virtual received words to increase the search radius such that extra candidate codewords and thus candidate elite members can be obtained. There-

fore, SEOLD-II outperforms SEOLD-I at the cost of increased complexity. Simulation results verify that the performance of both algorithms is better than that of the GMD algorithm and the KV algorithm.

In Chapter 5, we try to extend the concept of randomized sphere decoding to decode general linear block codes which are not MDS codes. We first investigate a novel sequential bit-flipping (SBF) algorithm which can transform the hard-limited reliability vector into a valid codeword with low complexity. When cyclic codes are in consideration, we can cyclic shift the reliability vector and decode by SBF algorithm to form a set of candidates where the one with smallest ED to the received word is chosen as the decoder output. On the other hand, we induce the concept of the randomized sphere decoding with moving center when the codes are not cyclic. A set of random virtual reliability vectors are generated and then decode by SBF algorithm. Again, the elite set of candidates is used to modify the random mechanism such that the center of the associated sphere will gradually move more and more close to the transmitted codeword iteratively. The proposed stochastic sequential bit flipping algorithm (SSBFA) outperforms the SPA for linear block codes with HDPC matrix. Although the proposed algorithms give satisfactory performance, especially for high rate linear block codes with HDPC matrix. There are several issues that require more research efforts. We mention just three to conclude this dissertation.

1. For long and/or low rate codes, the required complexity is still too high. Reducing the sampling dimension without compromising performance is a very challenging problem.

2. A stochastic decoding algorithm for decoding low density parity-check (LDPC) codes with complexity much lower than BP-based algorithms is most welcome.

3. Establish a firm theoretical foundation of the randomized sphere decoder and apply the same concept to solve other problems, e.g., MIMO detection.

Appendix A

The Proof of Lemma 5.1

Assume c_n is *a priori* equally likely to be 0 or 1, then the *a posteriori* probabilities on c_n is

$$P(c_n = 0|y_n) = \frac{1}{1 + e^{-L(c_n)}}, \tag{A.1}$$

$$P(c_n = 1|y_n) = \frac{1}{1 + e^{L(c_n)}}. \tag{A.2}$$

By the following fact:

$$\begin{cases} L(c_n) < 0, & \text{if } y_n < 0 \\ L(c_n) > 0, & \text{if } y_n > 0 \end{cases} \tag{A.3}$$

and the definition in (**??**),

$$q_n = \begin{cases} P(c_n = 0|y_n), & \text{if } y_n < 0 \\ P(c_n = 1|y_n), & \text{if } y_n > 0 \end{cases} \tag{A.4}$$

$$= \frac{1}{1 + e^{|L(c_n)|}} \tag{A.5}$$

$$= \frac{1}{1 + e^{\gamma_n}}. \tag{A.6}$$

Appendix B

The Proof of Theorem 5.1

From Bayes rule:

$$\frac{P(z_n = c_n|\mathbf{y}, \Sigma_n)}{P(z_n \neq c_n|\mathbf{y}, \Sigma_n)} = \frac{P(z_n = c_n|\mathbf{y})}{P(z_n \neq c_n|\mathbf{y})} \frac{P(\Sigma_n|z_n = c_n, \mathbf{y})}{P(\Sigma_n|z_n \neq c_n, \mathbf{y})}. \tag{B.1}$$

From Lemma 5.1, the first part of the right hand side of (B.1) is given by

$$P(z_n \neq c_n|\mathbf{y}) = q_n = 1 - P(z_n = c_n|\mathbf{y}). \tag{B.2}$$

Assume the check sums in Σ_n are statistically independent. The probability $P(\Sigma_n|z_n = c_n, \mathbf{y})$ is given by

$$P(\Sigma_n|z_n = c_n, \mathbf{y}) = \prod_{m \in \mathcal{M}(n)} P(\sigma_m|z_n = c_n, \mathbf{y}), \tag{B.3}$$

where

$$P(\sigma_m|z_n = c_n, \mathbf{y}) = \begin{cases} 1 - r_{mn} & \text{if } \sigma_m = 0 \\ r_{mn} & \text{if } \sigma_m = 1 \end{cases}. \tag{B.4}$$

Similarly,

$$P(\Sigma_n|z_n \neq c_n, \mathbf{y}) = \prod_{m \in \mathcal{M}(n)} P(\sigma_m|z_n \neq c_n, \mathbf{y}), \tag{B.5}$$

where

$$P(\sigma_m|z_n \neq c_n, \mathbf{y}) = \begin{cases} r_{mn} & \text{if } \sigma_m = 0 \\ 1 - r_{mn} & \text{if } \sigma_m = 1 \end{cases}. \tag{B.6}$$

From (B.3) - (B.6), we have

$$\frac{P(\Sigma_n|z_n = c_n, \mathbf{y})}{P(\Sigma_n|z_n \neq c_n, \mathbf{y})} = \prod_{m\in\mathcal{N}(m)} \left[(1-2\sigma_m)\left(\frac{1-r_{mn}}{r_{mn}}\right)\right]. \tag{B.7}$$

Using the approximation

$$\prod_{n\in\mathcal{N}(m)} (1-2q_n) \cong 1 - 2 \max_{n\in\mathcal{N}(m)} q_n, \tag{B.8}$$

we obtain

$$\begin{aligned} r_{mn} &= \frac{1}{2}\left(1 - \prod_{n'\in\mathcal{N}(m)\backslash n} (1-2q_{n'})\right) && \text{(B.9)} \\ &\cong \max_{n'\in\mathcal{N}(m)\backslash n}\left[\frac{1}{1+e^{\gamma_{n'}}}\right] && \text{(B.10)} \\ &= \frac{1}{1+e^{\min_{n'\in\mathcal{N}(m)\backslash n}\gamma_{n'}}}. && \text{(B.11)} \end{aligned}$$

Then

$$\begin{aligned} \log\left(\frac{1-\tilde{q}_n}{\tilde{q}_n}\right) &= \log\left(\frac{1-q_n}{q_n}\right) + \sum_{m\in\mathcal{N}(m)}\left[(1-2\sigma_m)\log\left(\frac{r_{mn}}{1-r_{mn}}\right)\right] && \text{(B.12)} \\ &\cong \gamma_n + \sum_{m\in\mathcal{N}(m)}\left[(1-2\sigma_m)\left(\min_{n'\in\mathcal{N}(m)\backslash n}\gamma_{n'}\right)\right]. && \text{(B.13)} \end{aligned}$$

Bibliography

[1] J. S. Yedidia, "Constructing Free-Energy Approximations and Generalized Belief Propagation Algorithms," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp.2282-2312, July 2005.

[2] M. Welling and Y. W. Teh, "Belief optimization for binary networks: A stable alternative to belief propagation," *Proc. Conf. Uncertainty in Artificial Intelligence*, Seattle, WA, Aug. 2001, pp. 554-561.

[3] A. Yuille, "CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagtion," *Neural Computation*, vol. 13, pp. 1691-1722.

[4] K. Rose, "Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems," *Proceedings, IEEE*, vol. 86, no. 11, pp. 2210-2239, November 1998.

[5] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math*, vol. 8, pp. 300-304, R.

[6] G. Gallager, *Low Density Parity Check Codes*, Monograph, M.I.T. Press, 1963.

[7] C. E. Shannon, "Communication in the presence of noise", *Proc. Institute of Radio Engineers*, vol. 37 (1): 10V21, Jan. 1949.

[8] The Digital Video Broadcasting Standard [Online]. Available: www.dvb.org

[9] The IEEE P802.3an 10GBASE-T Task Force [Online]. Available: http://www.ieee802.org/3/an

[10] The 802.16 Working Group [Online]. Available: http://www.ieee802.org/16/

[11] The IEEE 802.11n Working Group [Online]. Available: http://www.ieee802.org/11/

[12] J. Jiang and K. R. Narayanan, "Iterative soft-input soft-output decoding of Reed-Solomon codes by adapting the parity-check matrix", *IEEE Trans. Inform. Theory*, vol. 52, No. 8, 00. 3746-3756, Aug. 2006.

[13] T. Hehn, J. B. Huber, S. Laendner and O. Milenkovic, "Multiple-Bases Belief-Propagation for Decoding of Short Block Codes", *ISIT2007*, Nice, France, June, 2007.

[14] T. R. Halford and K. M. Chugg, "Random Redundant Iterative Soft-in Spft-out Decoding", *IEEE Trans. Commun.*, vol. 56, No. 4, pp. 513-517, Apr. 2008.

[15] I. Dimnil and Y. Be'ery, "Improved Random Redundant Iterative HDPC Decoding", *IEEE Trans Commun.*, vol. 57, no. 7, pp. 1-6, July 2009.

[16] Stephen B. Wicker, *ERROR CONTROL SYSTEMS for Digital Communication and Storage*, New Jersey: Prentice Hall, 1995.

[17] E. B. Berlekamp, *Algebraic Coding Theory*, New York: McGraw-Hill, 1968.

[18] Y. Sugiyama, M. kasahara, S.Hirasawa, and T. Namekawa, "A method for solving key equation for decoding goppa codes," *Inform. contr.*, vol. 27, pp. 87-99, 1975.

[19] G. D. Forney Jr., "Generalized minimum distance decoding," *IEEE Trans. Inform. Theory*, vol. IT-12, pp.125-131, Apr. 1966.

[20] V. Guruswami and M. Sudan, "Decoding of Reed-Solomon codes beyond the error-correction bound," *J. complexity*, vol. 13, pp. 180-193, 1997.

[21] R. Köetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Inform. Theory*, vol. 49, no. 11, pp. 2809-2825, Nov. 2003.

[22] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 170-182, Jan. 1972.

[23] H. Tang, Y. Liu, M. Fosorier, and S. Lin, "On combining chase-2 and GMD decoding algorithms for nonbinary block codes," *IEEE Commun. Lett.*, vol. 5, no. 5, pp. 209-211, May 2001.

[24] N. Wiberg, "Codes and Decoding on General Graphs", Ph. D. dissertation, Electrical Engineering Dept., Linkopin Univ., Linkoping, Sweden, 1996.

[25] Y. C. Chen and Y. T. Su, "Constraint relaxation and annealed belief propagation for binary networks", *ISIT 2007*, pp. 24-29, June, 2007.

[26] M. E. Khamy and R. J. McEliece, "Iterative algebraic soft decision list decoding of Reed-Solomon Codes", *IEEE J. Sel. Areas in Commun.*, vol. 24, pp. 481-490, Mar., 2006.

[27] P. H. Tan and L. K. Rasmussen, "The Serial and Parallel Belief Propagation Algorithms,"

[28] S. Lin and D. J. Costello, "Error Control Coding : Fundamentals and Applicatioins," *Prentice Hall*, 2004.

[29] E. H. L. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines.* New York: Wiley, 1989.

[30] Mackay's Encyclopedia of Sparse Graph Codes, http://www.inference.phy.cam.ac.uk/mackay/codes/data.html.

[31] R. Y. Rubinstein, "Optimization of computer simulation models with rare events", *European of Operational Research*,99:89-112, 1997.

[32] R. Y. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization", *Methodology and Computing in Applied Probability*, 2:127-190, 1999.

[33] R. Y. Rubinstein, D. P, Kroese, *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*, Springer, 2004.

[34] R. Y. Rubinstein and B. Melamed, *Modern Simulation and Modeling*, Wiley Series in Probability and Statistics, New York, 1998.

[35] F. Dambreville, "Cross-Entropy method: convergence issues for extended implementation,"
http://www.FredericDambreville.com

[36] M. EI-Khamy and R. J. McEliece, "Bounds on the average binary minimum distance and the maximum likelihood performance of Reed-Solomon codes," *in 42nd Allerton Conf. on Communication, Control, and Computing*, 2004.

[37] J. L. Massey, *Threshold Decoding*. Canbridge, MA: M.I.T. Press, 1963.