行政院國家科學委員會專題研究計畫成果報告

先進錯誤控制技術及其應用之研究 (2/3)

Advanced Error-Control Coding Technologies and Their Applications (2/3)

計畫編號：NSC 97-2221-E-009-082-MY3

執行期間：98 年 8 月 1 日至 99 年 7 月 31 日

主持人：蘇育德教授　　國立交通大學電機工程學系

計畫參與人員： 謝宗達、李昌明、詹芝雯、劉人仰、鄭乾君、王俊鋐、黃泓錫

中文摘要

我們針對線性方塊碼提出了兩種基於可靠度的低複雜度解碼演算法。此方法應用一新穎的連續位元翻轉(sequential bit-flipping)演算法來將經過硬限定的可靠度向量轉變為一合法的碼。兩個演算法先分別基於循環移位後的可靠度向量集合或者隨機產生的虛擬可靠度向量集合，來產生一組可能的候選碼集合，再選擇具有與接收向量最小歐幾理得距離的碼來當作解碼器輸出。針對具有短到中等碼長的線性方塊碼，以此概念設計的演算法相對於部分現有的演算法提供了性能與複雜度上的改善。

關鍵字：基於可靠度之解碼法、線性方塊碼、連續位元翻轉演算法。

Abstract

We present two low complexity reliability-based code-search algorithms for decoding linear block codes. The presented methodology utilizes a novel sequential bit-flipping (SBF) algorithm which can transform the hard-limited reliability vector into a valid codeword. Based on a set of cyclic shifted reliability vectors or random virtual reliability vectors respectively, both algorithms produce a set of candidates where the one with smallest Euclidean distance (ED) to the received is chosen as the decoder output. The proposed algorithms offer both complexity and performance advantages over some existing soft-decision decoding algorithms for linear block codes with short to medium code length.

Keywords: reliability-based decoding、linear block codes、sequential bit-flipping algorithm。

I. Introduction

Classical algebraic codes of short block length have large minimum distance and efficient hard-decision decoding (HDD) algorithms. Consequently, these codes represent a good choice for low-delay applications where high transmission reliability is required. An algebraic decoder only deals with hard decisions of the received sequence and its performance is not as good as soft decision decoding (SDD) since information is lost by considering hard decisions only. To improve the performance while keeping low decoding complexity, some hybrid algorithms have been devised. Forney's generalized minimum distance (GMD) decoding algorithm [1], the Chase II algorithm [2], and the combined Chase II-GMD algorithm [3] are three of them. These algorithms give a moderate performance improvement over HDD solutions with reasonable complexity.

Guruswami and Sudan (GS) [4] invented an algebraic list decoding algorithm which corrects beyond half the minimum distance. Koetter and Vardy (KV) [5] proposed an algebraic SDD algorithm based on a multiplicity assignment scheme to improve the GS algorithm. The KV algorithm and other improved interpolation-based approaches [6] - [9] can significantly outperform HDD for low rate RS codes. However, to achieve large coding gain, the complexity can be prohibitively large.

In this work, we try to investigate low complexity reliability-based code-search algorithms for decoding linear block codes. A novel sequential bit-flipping (SBF) algorithm which can transform the hard-limited reliability vector into a valid codeword with low complexity is presented. 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 Euclidean distnce (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 [12] 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 rest of this report is organized as follows. Some preliminaries are given in Section II. In Section III, the SBF algorithm and its limitation for decoding are introduced. Two of the proposed reliability-based decoding algorithms based on SBF algorithm are presented in Section IV and Section V, respectively. Some simulation results and discussions are presented in Section VI.

II. 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 $M \geq N-K$. 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 the superscript T is the transpose operation and $\mathbf{0}$ is a $1 \times M$ zero vector. For each row $\mathbf{h}_m$ of $\mathbf{H}$, $m \in J$, let

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

then

$$\mathbf{C} = \bigcap_{m=1}^{M} \mathbf{C}_m. \tag{2}$$

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

$$\Psi(\mathbf{c}) = \mathbf{x} = (x_1, \cdots, x_N),\ x_n = \Psi(c_n) = (-1)^{c_n} \tag{3}$$

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 $\mathbf{y} = (y_1, \cdots, y_N)$, 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 $\mathbf{z} = (z_1, \cdots, z_N)$ be the hard decision version of the received sequence $\mathbf{y}$, i.e.,

$$z_n = \begin{cases} 0, & y_n > 0 \\ 1, & \text{otherwise} \end{cases} \tag{4}$$

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

$$\sigma_m = \left[\sum_{n \in \mathcal{N}(m)} z_n \mathbf{H}_{mn} \right] \pmod 2 \tag{5}$$

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

Denoted by $\Gamma = (\gamma_1, \cdots, \gamma_{N-1})$ 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

$$L_n = \log \frac{P\left(c_n = 0 | \, \mathbf{y}\right)}{P\left(c_n = 1 | \, \mathbf{y}\right)}. \tag{6}$$

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

$$\lambda_m = \min_{n \in \mathcal{N}(m)} \gamma_n \tag{7}$$

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 $\mathbf{y}$. For the AWGN channel model considered, the probability q_n can be expressed as

$$q_n = \frac{1}{1 + e^{\gamma_n}} \tag{8}$$

Then 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 given by [10]

$$r_{mn} = \frac{1}{2} \left(1 - \prod_{n' \in \mathcal{N}(m)\backslash n} (1 - 2q_{n'}) \right). \tag{9}$$

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. 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, can be approximated as [11]

$$\begin{aligned} \xi_n &= \log \left[\frac{1 - \tilde{q}_n}{\tilde{q}_n} \right] = \log \left[\frac{P(z_n = c_n | \mathbf{y}, \Sigma_n)}{P(z_n \neq c_n | \mathbf{y}, \Sigma_n)} \right] \\ &\cong \gamma_n + \sum_{m \in \mathcal{M}(n)} \left[(1 - 2\sigma_m) \left(\min_{n' \in \mathcal{N}(m)\backslash n} \gamma_{n'} \right) \right] \end{aligned} \tag{10}$$

III. 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 works only on the parity-check matrix $\mathbf{H}$ with systematic form. As we know, the span of $\{\mathbf{h}_1, \cdots, \mathbf{h}_M\}$ forms the dull space of $\mathbf{C}$, say $\mathbf{C}^\perp$. For any $\mathbf{H}$, one can find $\tilde{\mathbf{H}} = [\mathbf{I}_{N-K}\mathbf{P}]$, where $\mathbf{I}_M$ is an $M \times M$ identity matrix and $\mathbf{P}$ is an $K \times (N-K)$ binary matrix such that $\{\tilde{\mathbf{h}}_1, \cdots, \tilde{\mathbf{h}}_{N-K}\}$ also spans $\mathbf{C}^\perp$. Therefore, without lost of generality, we assume that $M = N - K$ and $\mathbf{H}$ is always a systematic form in this paper 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

$$\mathbf{d}_t \in \cap_{m=1}^{t} \mathbf{C}_{o(m)}, \ 1 \leq t \leq M. \tag{11}$$

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

$$\begin{cases} \hat{L}_n = L_n & \text{if } d_n = z_n \\ \hat{L}_n = -L_n & \text{if } d_n \neq z_n \end{cases} \tag{12}$$

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

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

$$\arg \min_{n \in \{I_t \setminus I_{t-1}\}} \xi_n \tag{13}$$

where ξ_n is evaluated by (10). Let

$$\begin{aligned} \mathbf{d}_t &\leftarrow \mathbf{d}_{t-1} + \mathbf{e}_{n^*} \text{ (mod 2)}, \quad (14) \\ \hat{L}_{n^*} &\leftarrow -\hat{L}_{n^*}, \quad (15) \\ \sigma_m &\leftarrow \sigma_m + 1 \text{ (mod 2) } \forall m \in \mathcal{M}(n^*), \quad (16) \\ t &\leftarrow t+1. \end{aligned}$$

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 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 two, the output codeword $\mathbf{d}_M$ won't be the transmitted codeword.

Example 1: Consider a (8,4) linear block code with parity check matrix:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \quad (17)$$

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 SBF algorithm is shown in Fig. 1 where the order of check sums is $3 \to 2 \to 1 \to 4$.

A. Predicament of Decoding via SBF algorithm

The SBF algorithm 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 1 and 2 are different because of different orders although they face the same received word $\mathbf{y}$.

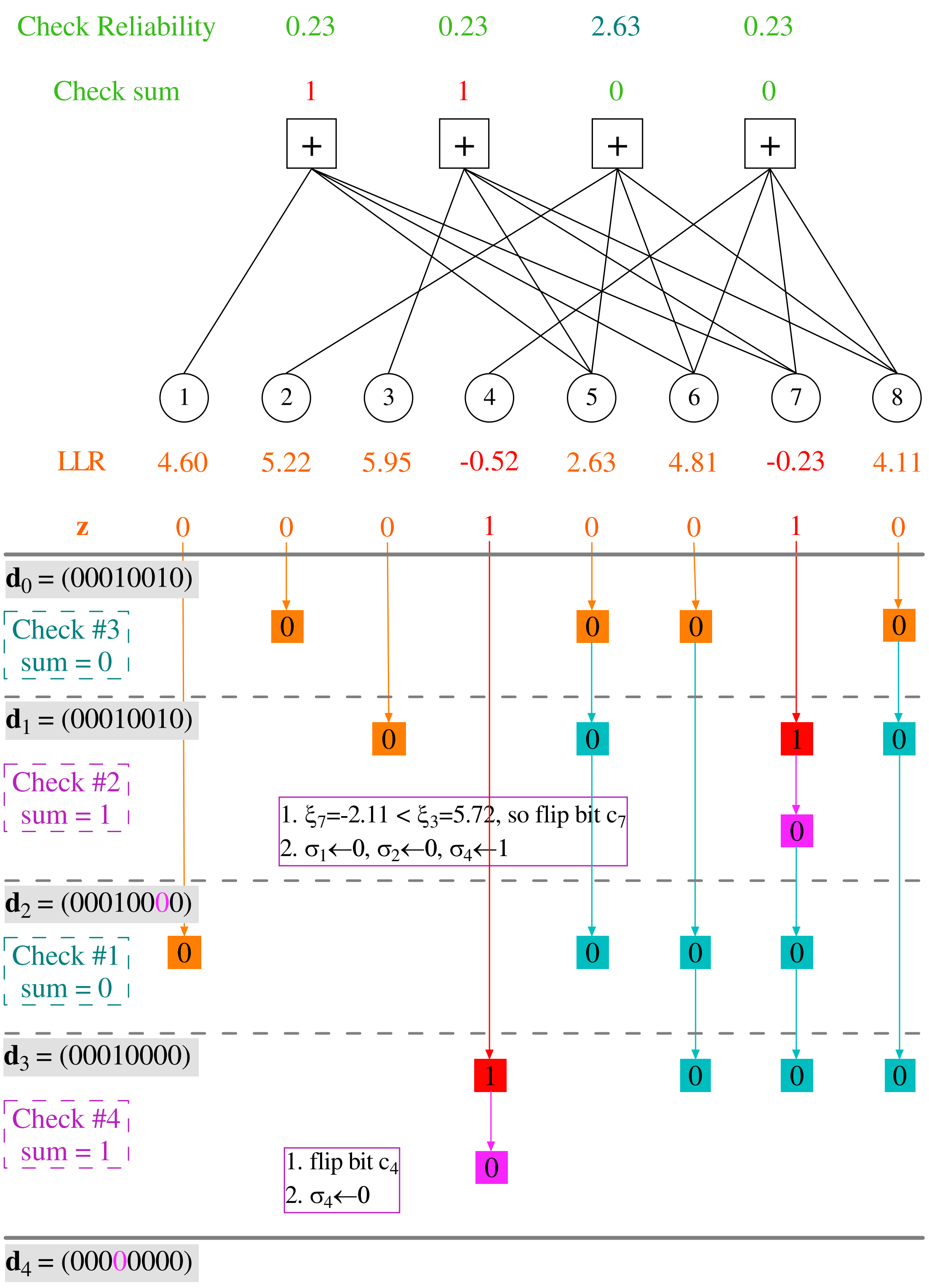
Check Reliability
0.23
0.23
2.63
0.23
Check sum
1
1
0
0
LLR
4.60
5.22
5.95
-0.52
2.63
4.81
-0.23
4.11
z
0 0 0 1 0 0 1 0
$\mathbf{d}_0 = (00010010)$
Check #3 sum = 0
$\mathbf{d}_1 = (00010010)$
Check #2 sum = 1
1. ξ_7=-2.11 < ξ_3=5.72, so flip bit c_7
2. $\sigma_1 \leftarrow 0$, $\sigma_2 \leftarrow 0$, $\sigma_4 \leftarrow 1$
$\mathbf{d}_2 = (00010000)$
Check #1 sum = 0
$\mathbf{d}_3 = (00010000)$
Check #4 sum = 1
1. flip bit c_4
2. $\sigma_4 \leftarrow 0$
$\mathbf{d}_4 = (00000000)$

Fig. 1. An example of the SBF algorithm.

Example 2: Consider the same case described in Example 1. 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 1 is equal to the transmitted codeword but in Example 2 is not. It is because the order used in Example 2 meets the situation described in Remark 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 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 under a fixed order. 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 described in Section V which can correct errors iteratively.

IV. 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 $\mathcal{L}^\nu = (L_{\nu+1}, L_{\nu+2}, \cdots, L_N, L_1, \cdots, L_\nu)$ 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 $\nu \in I$, 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

$$\hat{\mathbf{c}}_T = \arg\min_{\mathbf{c} \in D} d(\Psi(\mathbf{c}), \mathbf{y}), \tag{18}$$

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

V. 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 [12] 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.

A. 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

$$\mu_n^{(0)} = L_n \tag{19}$$

$$\rho_n^{(0)} = \frac{4}{N_0} \tag{20}$$

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 $1 \le \ell \le N_s$.

B. 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 [**?**]

$$\mu_n^{(t+1)} = (1-\delta)\mu_n^{(t)} + \delta \cdot \frac{\sum_{\hat{\mathbf{s}}_\ell^{(t)} \in \mathbf{E}^{(t)}} \hat{s}_{\ell,n}^{(t)}}{E_s} \tag{21}$$

and

$$\rho_n^{(t+1)} = (1-\varepsilon)\rho_n^{(t)} + \varepsilon \cdot \frac{\sum_{\hat{\mathbf{s}}_\ell^{(t)} \in \mathbf{E}^{(t)}} \left(\hat{s}_{\ell,n}^{(t)} - \mu_n^{(t+1)}\right)^2}{E_s} \tag{22}$$

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

C. *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 (19) and decide an order $\mathbf{o}$. Set $t = 0$.
2. Generate a set of random samples $\mathbf{S}^{(t)} = \{\mathbf{s}_1^{(t)}, \cdots, \mathbf{s}_{N_s}^{(t)}\}$ from $\mathbb{N}\left(\vec{\mu}^{(t)}, \vec{\rho}^{(t)}\right)$.
3. For each sample, we have $(\mathbf{d}_\ell^{(t)}, \hat{\mathbf{s}}_\ell^{(t)}) = \Omega(\mathbf{o}, \mathbf{s}_\ell^{(t)})$.
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 (21) and (22), respectively.
6. If $t = T$ of for some $t \geq c$, say $c = 3$,

$$\mathbf{d}^{*(t)} = \mathbf{d}^{*(t-1)} = \cdots = \mathbf{d}^{*(t-c)}, \tag{23}$$

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

VI. Simulation 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. 2 - 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. 7, the performance of the SBF algorithm with a fixed order is worse than BDD and SPA because of the phenomenon described in Remark 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 1. For the (15,11) RS code, similar decoding performance can be observed in Fig. 8. Our proposed 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.

References

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

[2] 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.

[3] 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.

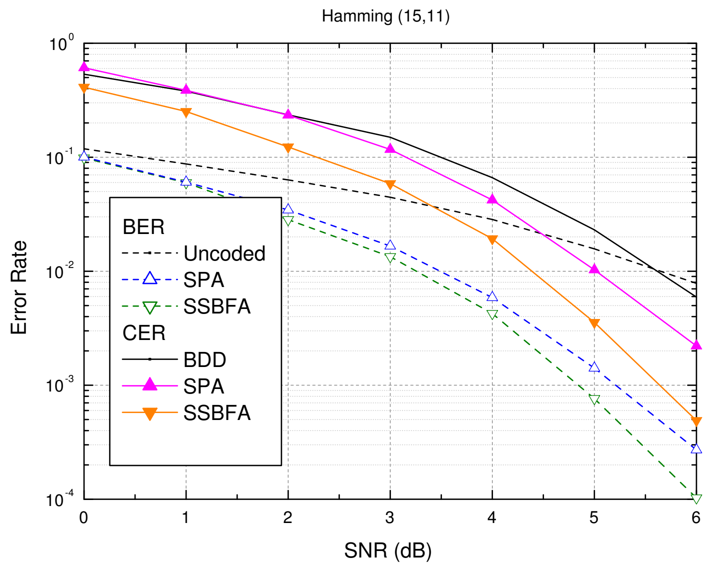

Fig. 2. Error rate performance of the (15,11) Hamming Code; $N_s = 10$, $E_s = 1$

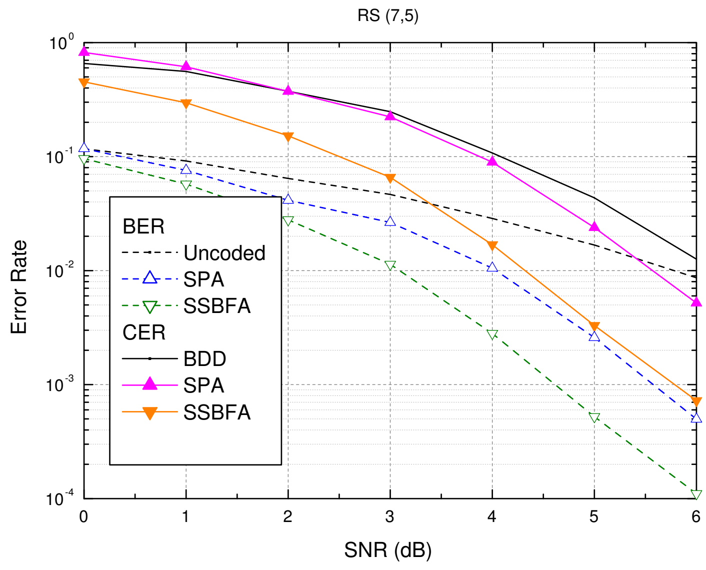

Fig. 3. Error rate performance of the (7,5) RS Code; $N_s = 10$, $E_s = 1$

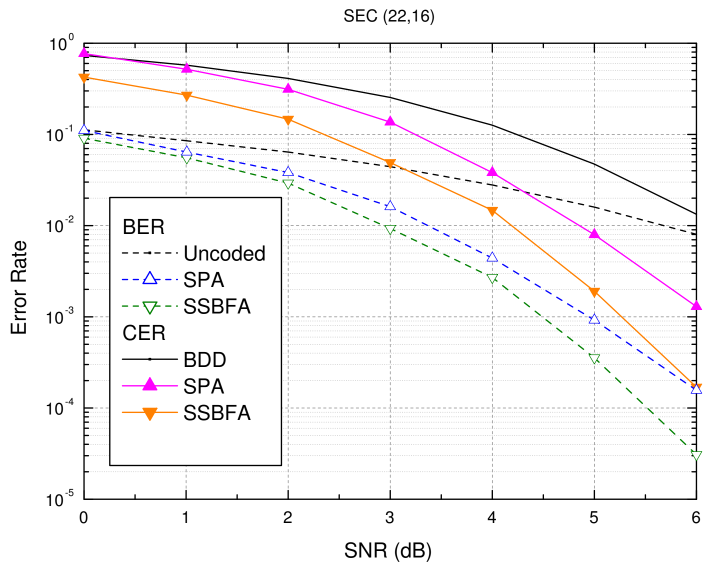

Fig. 4. Error rate performance of the (22,16) single error correction Code; $N_s = 10$, $E_s = 1$

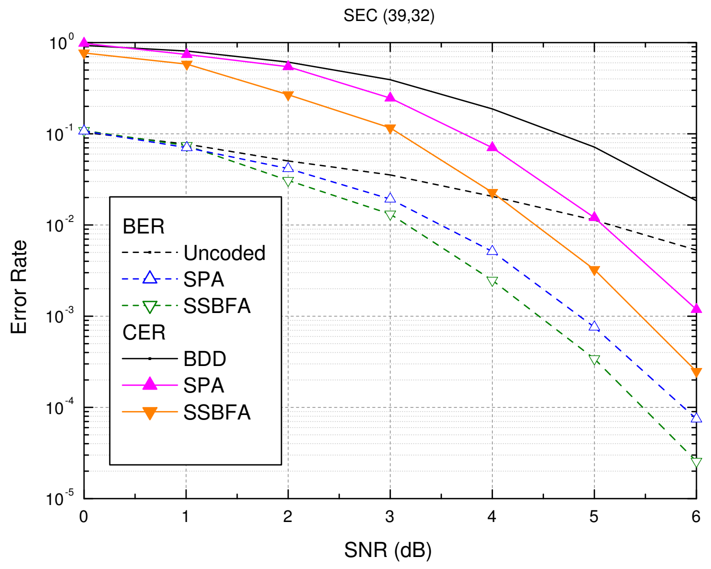

Fig. 5. Error rate performance of the (39,32) single error correction Code; $N_s = 10$, $E_s = 1$

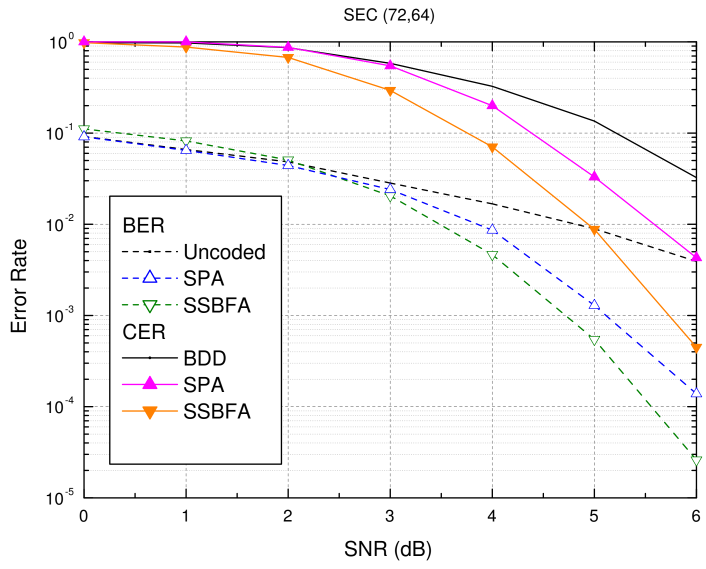

Fig. 6. Error rate performance of the (72,64) single error correction Code; $N_s = 10$, $E_s = 1$

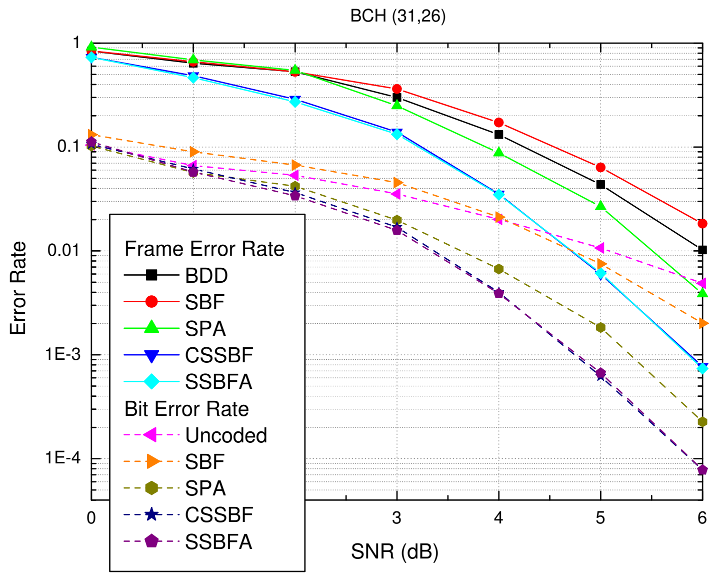

Fig. 7. Error rate performance of the (31,26) BCH Code.

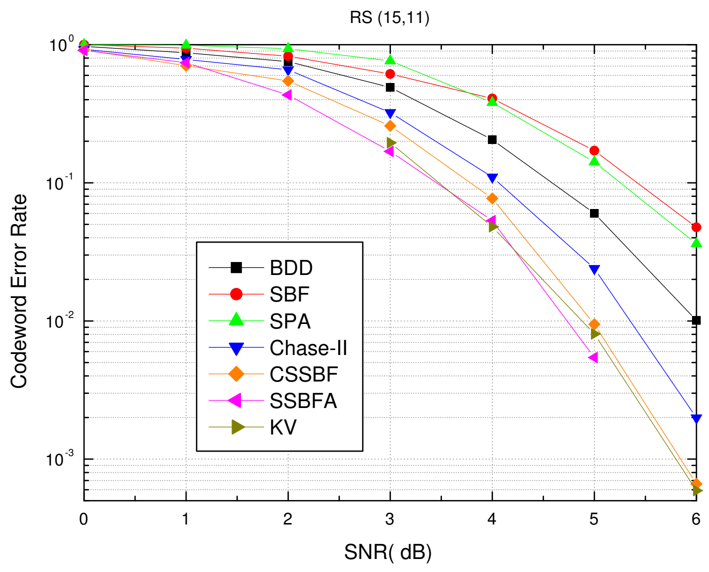

Fig. 8. Error rate performance of the (15,11) RS Code.

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

[5] R. K$\ddot{o}$etter and A. Vardy, “Algebraic soft-decision decoding of Reed-Solomon codes,” *IEEE Trans. Inform. Theory*, vol. 49, no. 11, pp. 2809-2825, Nov. 2003.

[6] W. J. Gross, F. R. Kschischang, R. Koetter and P. G. Gulak, “A VLSI architecture for interpolation in soft-decision list decoding of Reed- Solomon codes,” *in Proc. IEEE Workshop Signal Process. Syst.*, pp. 39-44, 2002.

[7] J. Bellorado and A. Kavcic, “A low-complexity method for Chase-type decoding of Reed-Solomon codes,” *in Proc. IEEE Int. Symp. Inf. Theory*, pp. 2037-2041, 2006.

[8] H. Xia, C. Zhong and J. R. Cruz, “A Chase-type algorithm for softdecision Reed-Solomon decoding on Rayleigh fading channels,” *in Proc. IEEE Global Commun. Conf.*, vol. 3, pp. 1751-1755, 2003.

[9] H. Xia and J. R. Cruz, “Reliability-based forward recursive algorithms for algebraic soft-decision decoding of Reed-Solomon codes,” *IEEE Trans. Commun.*, vol. 55, no. 7, pp.1273-1278, July 2007.

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

[11] M. P. C. Fossorier, M. Mihaljević, and H. Imai, “Reduced Complexity Iterative Decoding of Low-Desity Parity Check Codes Based on Belief Propagation”, *IEEE Trans. Commun.*, vol. 47, pp. 673-680, May 1999.

[12] Chang-Ming Lee and Yu T. Su,“Stochastic Erasure-Only List Decoding Algorithms for Reed-Solomon Codes”, *IEEE Signal Process. Lett.*, vol. 16, pp. 691-694, Aug. 2009.