2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing

Weighted Symbol-Flipping Decoding for Nonbinary LDPC Codes

Bing Liu, Jun Gao, Gaoqi Dou
Dept. of Communication Engineering
Naval University of Engineering
Wuhan, China
Email: liubing5275093@hotmail.com

Wei Tao
System Division
Naval Academy of Armament
Beijing, China
Email: alantao0451@yahoo.com.cn

***Abstract*—In this paper, a new low-complexity symbol-flipping algorithm to decode nonbinary low-density parity-check (LDPC) codes is proposed. The decoding procedure updates iteratively the hard-decision received symbol vector in search of a valid codeword in the symbol vector space. Only one symbol is changed in each iteration, and symbol flipping function combines the number of failed checks and reliability of the received bits and calculated symbols. An optional mechanism to avoid infinite loops in high Galois field search is also proposed. Our studies show that the algorithm achieves an appealing tradeoff between performance and complexity over relatively low Galois field for short to medium code length.**

Keywords- Low-density parity-check (LDPC) codes; nonbinary; quasi-cyclic; symbol-flipping (SF) decoding

I. INTRODUCTION

Error control using low-density parity-check (LDPC) codes is currently the most promising coding technique to achieve Shannon capacity for a wide range of channels. Binary LDPC codes were originally invented by Gallager [1] and rediscovered by Mackay and Neal [2]. Subsequently, a great deal of research was focused on binary LDPC codes. The construction of binary LDPC codes could be extended to the higher Galois field, which forms nonbinary LDPC codes. Nonbinary LDPC codes were first investigated by Davey and Mackay in 1998 [3] . In the probability domain, the algorithm can be efficiently implemented using fast Fourier transform (FFT), which froms FFT Q-ary sum product algorithm (FFT-QSPA) [4]. In log-likelihood ratio (LLR) domain, extended min-sum (EMS) was proposed in [5]. The reduced check-node processing uses the incoming messages concerning only a part of Galois field elements. The aim of these algorithms is to reduce the computational complexity. As we know, weighted bit-flipping algorithm over binary LDPC codes could achieve a tradeoff among performance and complexity [6-8]. So extending bit-flipping algorithm into nonbinary LDPC codes to obtain low complexity is still an open challenge.

The main contribution of the paper is to propose a weighted symbol-flipping (SF) decoding algorithm for nonbinary LDPC codes. While most of the nonbinary LDPC decoding literature [3-5, 9] focuses on the probability or LLR domain decoding, and its downward lower complexity approximations, here we propose a symbol-flipping algorithm, and make full use of the reliability of received sequences. In this context, we would like to achieve low complexity as bit-flipping in binary LDPC codes, and the proposed weighted symbol flipping algorithm shows better performance than RS codes of the same lengths and rates decoded with hard-decision Berlekamp-Massey (BM) algorithm.

The rest of this paper is organized as follows. Notation, definitions, and the core algorithm with loop detection are given in Section II. We calculate the complexity of the algorithm in Section III. In Section IV, we present and discuss some simulation results, and in Section V, we draw a few concluding remarks.

II. SYMBOL-FLIPPING DECODING ALGORITHM

A. Notation and Basic Definitions

Consider a nonbinary (N,K) LDPC code with length N and dimension K and choose an arbitrary codeword $\mathbf{c}=[c_0,c_1,\cdots,c_{N-1}]\in\{\mathrm{GF}(q)\}^N$, where $q=2^b$. The codeword is now mapped to a signaling constellation, Ω, resulting in a bipolar vector $\mathbf{t}=[\mathbf{t}_0,\mathbf{t}_1,\cdots,\mathbf{t}_{N-1}]$, where $\mathbf{t}_n=[t_{nb},\cdots,t_{(n+1)b-1}]$, $0\le n<N$. Here we consider BPSK signaling. Thus, $\psi:\mathrm{GF}(q)\to\Omega^b$, with $\Omega=\{-1,1\}$ such that $\psi(c_n)=[t_{nb},\cdots,t_{(n+1)b-1}]$ and $t_i\in\Omega$, $0\le i<Nb$. Then the channel output is

$$\mathbf{r}=\mathbf{t}+\mathbf{n}, \tag{1}$$

with $\mathbf{n}$ a real vector of independent white Gaussian noise sample with power $\sigma^2=N_0/(2E_bR)$, where E_b/N_0 is the SNR per information bit and $R=K/N$ is the code rate.

For $0\le i<Nb$, the initial hard–limited bit decision

$$x_i=\mathrm{sgn}(r_i), \tag{2}$$

where $\mathrm{sgn}(r)=1$ if $r\ge 1$, and $\mathrm{sgn}(r)=0$ if $r<1$.

A block of Nb bits is converted to a sequence of N $\mathrm{GF}(q)$ symbols according to some mapping, $\varphi:(\mathrm{GF}(2))^b\to\mathrm{GF}(q)$. Then the hard-limited symbol decision is $\mathbf{y}=[y_0,y_1,\cdots,y_{N-1}]$, where $y_n=\varphi([x_{nb},\cdots,x_{(n+1)b-1}])$, $0\le n<N$.

We first define the log-likelihood ratio of nonzero element a over $\mathrm{GF}(q)$:

$$\mathbf{L}_n(a)=[L_n(a=\alpha_1),\cdots,L_n(a=\alpha_{q-1})], \tag{3}$$

where

$$L_n(a=\alpha_i)=\ln\frac{P(a=\alpha_i)}{P(a=0)}=\frac{2}{\sigma^2}\sum_{j:\psi(\alpha_i)_j=+1} r_{nb+j}, \tag{4}$$

with $P(a=\alpha_i)$ denoting the probability that a takes on the value α_i.

We denote the Galois field GF(q) without the zero element by $\mathrm{GF}_0(q)=\{\alpha_1,\cdots,\alpha_{q-1}\}$. For simplicity, the reliability of a on the nth symbol is $\mathbf{L}_{n,a}=[L_{n,\alpha_1},\cdots,L_{n,a},\cdots,L_{n,\alpha_{q-1}}]$, where $L_{n,a}=\sum_{j:\psi(a)_j=+1} r_{nb+j}$. So $L_{n,a}$ is simplified version of $L_n(a=\alpha_i)$ without the constant item $2/\sigma^2$, where $a\in\mathrm{GF}_0(q)$.

A nonbinary (N,K) LDPC code is completely described by its sparse nonbinary parity-check matrix $\mathbf{H}$ that has N columns and $M\geq N-K$ rows. For any tentative symbol decision vector $\mathbf{y}$, the syndrome is the vector $\mathbf{s}=\mathbf{y}\mathbf{H}^T$, where the multiplication operation is over GF(q). $\mathbf{H}$ has column weight d_v and row weight d_c. We define the set $\mathcal{N}(m)$ that contains the indices of the d_c symbols that participate in check m, and the set $\mathcal{M}(n)$ that contains the indices of the d_v checks in which symbol n participates

$$\mathcal{N}(m)\triangleq\{n:\mathbf{H}_{mn}\neq 0\},\ \mathcal{M}(n)\triangleq\{m:\mathbf{H}_{mn}\neq 0\}. \quad (5)$$

B. Loop Detection

For convenience, we first define the vector sum $\mathbf{E}(x)\triangleq\sum_{i=x+1}^{k}\mathbf{e}_{n^{(i)}}$, where $\mathbf{e}_{n^{(i)}}$ denotes the flipping symbol vector in ith iteration. If in the kth iteration, the new symbol vector $\mathbf{y}^{(k)}$ coincides with a previously considered vector $\mathbf{y}^{(k_0)}$, $k_0<k$, then $n^{(k+1)}=n^{(k_0+1)}$, the decoding process is trapped in an infinite loop. Naturally, we reach the conclusion that

$$\mathbf{y}^{(k)}=\mathbf{y}^{(k_0)}\ \text{if and only if}\ \mathbf{E}(k_0)=\mathbf{0}. \quad (6)$$

In the algorithm, we can calculate $\mathbf{E}(l)$ iteratively

$$\mathbf{E}(l-1)=\begin{cases}\mathbf{e}_{n^{(k)}}, & if\ l=k\\ \mathbf{E}(l)+\mathbf{e}_{n^{(l)}}, & if\ l<k\end{cases}. \quad (7)$$

So we can calculate the vectors $\mathbf{E}(k-1),\mathbf{E}(k-2),\cdots\mathbf{E}(0)$ iteratively. If any of these vectors is all zeros, an infinite loop is detected, and the symbol selected to be flipped should be discarded.

C. Algorithm

The proposed algorithm starts from the hard-limited received symbol vector $\mathbf{y}^{(0)}$ and searches iteratively the N-dimensional symbol vector space over GF(q) for a valid codeword vector. Assume that the prior to the kth iteration, $k\geq 1$, the symbol vector under consideration is $\mathbf{y}^{(k-1)}$ with corresponding syndrome vector $\mathbf{s}^{(k-1)}=\mathbf{y}^{(k-1)}\mathbf{H}^T\neq 0$. The objective is to perturb $\mathbf{y}^{(k-1)}$ by one symbol and create a new candidate symbol vector $\mathbf{y}^{(k)}$. In order to choose which symbol of $\mathbf{y}^{(k-1)}$ to flip, and change to which value, here we first calculate the metric value of symbol over $\mathrm{GF}_0(q)$ to determine the flipped symbol, for each symbol and each $\mathrm{GF}_0(q)$ element,

$$E_{n,a}^{(k)}=\sum_{m\in\mathcal{M}(n)}(2s_m-1)w_{n,m,a}-\alpha\left|L_{n,a}\right|, \quad (8)$$

where $w_{n,m,a}=\min_{i\in\mathcal{N}(m)\backslash n}\left|L_{i,a}\right|$, $0\leq n<N$, $0\leq m<M$, $a\in\mathrm{GF}_0(q)$, and α is a weighting factor needed to be determined before decoding.

In order to evaluate the symbol reliability, we calculate the metric value

$$E_n^{(k)}=\sum_{a\in\mathrm{GF}_0(q)}E_{n,a}^{(k)}. \quad (9)$$

So we can determine which symbol should be flipped by the symbol flipping function $E_n^{(k)}$, i.e. the flipped symbol position is determined. But which symbol should be flipped into is still to be selected. So the selected value of the flipped symbol could also depend on the absolute values of the channel output, i.e. $|r_i|$, $nb\leq i<(n+1)b$. Each symbol y_n can be transformed into b bits over GF(q), and there are b corresponding absolute values of r_i that considered as the measurement of the reliability of the received symbols for additive white Gaussian noise (AWGN) channel. The closer $|r_i|$ approaches zero, the less reliable the corresponding bit is. So here we can find out some minimum values $|r_i|$ among the b absolute values of r_i and flip their corresponding bits according to the bit flipping flag value *flagbit*. The flipped bit number of the chosen symbol equals to the bit flipping flag value, and we should flip the minimum *flagbit* values according to $|r_i|$.

So flip the symbol in $\mathbf{y}^{(k-1)}$ with minimum *flagbit* values corresponding bits

$$\mathbf{y}^{(k)}=\mathbf{y}^{(k-1)}\oplus\mathbf{e}_{n^{(k)}}, \quad (10)$$

where $\oplus$ denotes addition over GF(q) and

$$\mathbf{e}_n=[\underbrace{0,\cdots,0}_{n-1},\beta,\underbrace{0,\cdots,0}_{N-n}], \quad (11)$$

in vector $\mathbf{e}_n$, $\mathrm{GF}_0(q)$ element β depends on the received bit values $|r_i|$, $nb\leq i<(n+1)b$.

We update the syndrome vector to $\mathbf{s}^{(k)}$, and get the hard decision syndrome vector

$$s_{h,j}^{(k)}=\begin{cases}0 & if\ s_j^{(k)}=0\\ 1 & if\ s_j^{(k)}\neq 0\end{cases}. \quad (12)$$

If $\mathbf{s}_h^{(k)}=0$, we have a valid codeword and stop the search; if not, we calculate flipping function $E_n^{(k+1)}$, $0\leq n<N$, and continue.

In summary, the complete decoding algorithm with loop detection and prevention is as follows.

Step 1) Initialization: Set iteration counter $k=0$; Calculate $\mathbf{y}^{(0)}$, $\mathbf{L}_{n,a}$ and $w_{n,m,a}$; "Exclusion symbol list" $B\leftarrow\varnothing$; "Bit flipping flag" $flagbit=1$.

Step 2) Calculate hard syndrome $\mathbf{s}_h^{(k)}$; if $\mathbf{s}_h^{(k)}=0$, then return $\mathbf{y}^{(k)}$ and STOP.

Step 3) $k\leftarrow k+1$; if $k>k_{\max}$ (user-specified maximum iteration number), then declare decoding failure and STOP.

Step 4) For $0\leq n<N$, calculate $E_n^{(k)}$.

Step 5) Calculate and sort out the flipped symbol $n^{(k)}=\arg\max_{n\in\mathcal{N}(m),n\notin B}E_n^{(k)}$.

Step 6) For the chosen symbol, flip the bit that corresponding to minimum absolute *flagbit* values of x_i, $nb\leq i<(n+1)b$.

Step 7) Calculate $\mathbf{E}(l)$ by (7), $l=k-1,k-2,\cdots,0$; if $\mathbf{E}(l)=\mathbf{0}$ for any $l=k-1,\cdots,0$, then if $flagbit\neq q$, $flagbit\leftarrow flagbit+1$, go to Step 6); if $flagbit=q$, $B\leftarrow B\bigcup\{n^{(k)}\}$ and $flagbit=1$, go to Step 5).

Step 8) Calculate $\mathbf{y}^{(k)}$ by (10); $B\leftarrow\varnothing$; go step 2).

TABLE I. COMPLEXITY COMPARISON PER SEQUENCE FOR VARIOUS DECODING SCHEMES

Schemes	Preprocess	SF Function Update	Select Symbol to Flip
SF	$N\left[\sum_{i=1}^{b}(i-1)\binom{b}{i}\right]+ (q-1)N\log_2 N$	$(q-1)Nd_v + (q-1)(A_{ni}-1)d_v d_c + (q-2)A_{ni}$	$A_{ni}(b\log_2 b + N - 1)$
FFT-QSPA	$A_{ni}[Nd_v q(2\log_2 q + 2 + d_c + 2d_v) + 2Md_c - M]$		

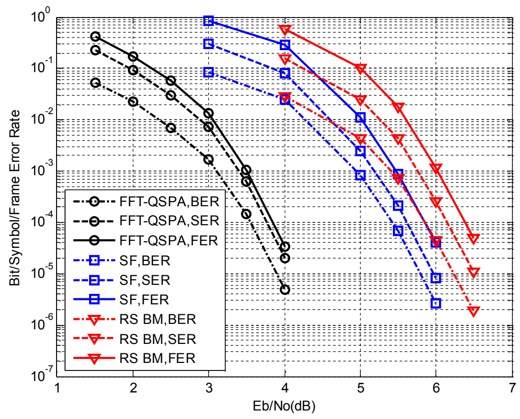

Figure 1. Error performance of the finite geometry (63, 37) LDPC code and (63, 37) RS code over GF(2^6)

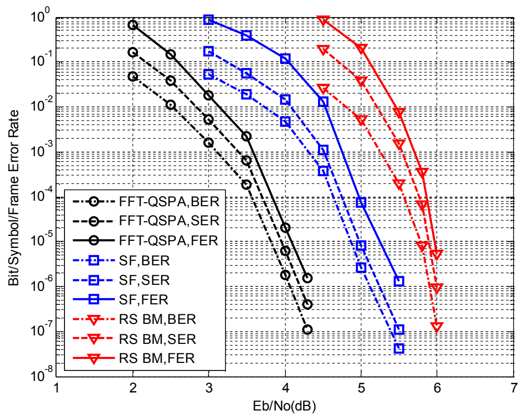

Figure 2. Error performance of the finite field (225, 147) LDPC code and (225, 147) shortened RS code over GF(2^8)

III. COMPUTATIONAL COMPLEXITY

The computational complexity of the decoding algorithm described above can be calculated as follows. For the proposed symbol flipping algorithm, the complexity roughly consists of three phases: preprocessing, symbol flipping function updating and to-be-flipping symbol selecting. Ignoring simple GF(q) operations and a small number of real multiplications or divisions, it suffices to address the dominant real additions for SF decoding scheme, if one real comparison is regarded as one real addition.

To sum up, Table I gives complexity composition for the proposed symbol flipping algorithm and the total complexity of FFT-QSPA, where A_{ni} denotes average number of iterations per decoding. The complexity of SF function update consists of three parts: the first part denotes that each SF function initialization needs $(q-1)d_v$ real additions for one symbol in the first iteration; the second part denotes that the corresponding SF function update for the subsequence iterations; and the last part denotes the addition of (9). But the complexity of FFT-QSPA includes real additions, multiplications and divisions, the complexities of which are $2Nd_vq\log_2q+2Nd_v\ (q-1)+M(d_c-1)$, $Nd_vq(d_c+2d_v-1)+\ Md_c$ and $Nd_v\ (q+2)$, respectively. Real multiplications and divisions are more consumable units than real additions. Loop detection, as described in Section II, requires only symbol operations and comparisons, and the total number of loop detection and prevention operations is independent of the block length, and depends only on the total number of executed iterations, so the cost of loop detection can be safely ignored in the algorithm. Therefore, the complexity of symbol flipping algorithm is much lower than FFT-QSPA.

IV. SIMULATION RESULTS

In this section, we evaluate the bit-error rate (BER), symbol-error rate (SER) and frame-error rate (FER) of two LDPC decoders and a RS decoder as a function of the SNR. We study the following decoders: 1) weighted SF algorithm; 2) FFT-QSPA; 3) RS BM algorithm.

In Fig. 1, we present our findings for (63,37) 64-ary Euclidean geometry (EG)-LDPC code with rate 0.5873 [10], [11]. The parity-check matrix **H** of this code consists of a single 63×63 circulant with both column and row weights 8. The weighting factor is set to 1.6. The performance of this code decoded with the proposed algorithm (200 iterations) as described in Section II is shown in Fig. 1. For comparison, we also include the performance of the FFT-QSPA (50 iterations) for LDPC code and BM algorithm for RS code. At FER of 10^{-4}, it achieves about 0.53dB coding gain over (63, 37) RS code over GF(2^6) decoded with the BM algorithm. At the same time, the proposed algorithm is just about 2dB away from FFT-QSPA decoding.

In Fig.2, we present the performance of (225,147) 16-ary finite field (FF)-LDPC code with rate 0.6533 [12]. The parity-check matrix **H** of this code is a 225×225 circulant with both column and row weights 14. The weighting factor is fixed at 1.4. At FER of 10^{-5}, it achieves a 0.72dB coding gain over (225,147) shortened RS code over GF(2^8), and the proposed algorithm (100 iterations) is just about 1.16dB away from FFT-QSPA decoding (50 iterations). In view of the results in Fig. 1 and 2, we argue that the proposed symbol flipping algorithm offer the tradeoff points between performance and computational cost.

V. Conclusion

In this paper, we propose the weighted symbol flipping decoding algorithm for nonbinary LDPC codes, in which one symbol is updated in each iteration during the decoding process. A loop detection procedure with minimal computational overhead is also designed that protects the generated symbol-search sequence from falling into infinite loop traps. Simulation results and complexity comparison show that compared to RS codes a noticeable improvement in decoding performance over relatively low Galois field for short to medium code length is observed, while the computational complexity is very low. Therefore, we can expect that it is very competitive for design of high throughput communication equipments.

References

[1] R. G. Gallager, *Low-Density Parity-Check Codes*: Cambridge, MA: M.I.T. Press, 1963.

[2] D. J. C. Mackay and R. M. Neal, "Near Shannon limit performance of low-density parity-check codes," *Electron. Lett.,* vol. 32, pp. 1645-1646, Aug. 1996.

[3] M. C. Davey and D. MacKay, "Low-density parity check codes over GF(q)," *IEEE Commun. Lett.,* vol. 2, no. 6, pp. 165-167, Jun. 1998.

[4] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over GF(2^q)," in *IEEE Information Theory Workshop*, Paris, France, Mar. 2003, pp. 70-73.

[5] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Trans. Commun.,* vol. 55, no. 4, pp. 633-643, Apr. 2007.

[6] Z. Liu and D. A. Pados, "A decoding algorithm for finite-geometry LDPC codes," *IEEE Trans. Commun.,* vol. 53, no. 3, pp. 415-421, Mar. 2005.

[7] T. M. N. Ngatched, F. Takawira and M. Bossert, "An improved decoding algorithm for finite-geometry LDPC codes," *IEEE Trans. Commun.,* vol. 57, no. 2, pp. 302-306, Feb. 2009.

[8] X. Wu, C. Ling, M. Jiang, E. Xu, C. Zhao and X. You, "New insights into weighted bit-flipping decoding," *IEEE Trans. Commun.,* vol. 57, no. 8, pp. 2177-2180, Aug. 2009.

[9] H. Wymeersch, H. Steendam and M. Moeneclaey, "Log-domain decoding of LDPC codes over GF(q)," in *IEEE International Conference on Communications*, Paris, France, Jun. 2004, pp. 772-776.

[10] L. Zeng, L. Lan, Y. Y. Tai, B. Zhou, S. Lin and K. A. S. Abdel-Ghaffar, "Construction of nonbinary cyclic, quasi-cyclic and regular LDPC codes: a finite geometry approach," *IEEE Trans. Commun.,* vol. 56, no. 3, pp. 378-387, Mar. 2008.

[11] B. Zhou, J. Kang, Y. Y. Tai, S. Lin and Z. Ding, "High performance non-binary quasi-cyclic LDPC codes on Euclidean geometries," *IEEE Trans. Commun.,* vol. 57, no. 5, pp. 1298-1311, May 2009.

[12] L. Zeng, L. Lan, Y. Y. Tai, S. Song, S. Lin and K. Abdel-Ghaffar, "Constructions of nonbinary quasi-cyclic LDPC codes: a finite field approach," *IEEE Trans. Commun.,* vol. 56, no. 4, pp. 545-554, Apr. 2008.