Symbol Flipping Algorithm With Self-Adjustment Strategy for LDPC Codes Over GF(q)

Bin Dai, Rongke Liu, *Member, IEEE*, Chenyu Gao, and Zhen Mei

***Abstract*—Compared with the non-prediction algorithm, the prediction-based symbol flipping decoding algorithm of non-binary, low-density, parity-check codes over GF(q) can significantly improve the error performance. To escape from undesirable local maximum, this paper proposes a self-adjustment strategy to redesign the flipping metric. Combined with the prediction mechanism, our proposed algorithm provides a good trade-off between the complexity and the error correction performance. Simulation results for the additive white Gaussian noise channel and the binary symmetric channel verify the effectiveness of the proposed algorithm.**

***Index Terms*—NB-LDPC codes, self-adjustment strategy, iterative decoding, symbol flipping decoder.**

I. Introduction

Non-Binary low-density parity-check (NB-LDPC) codes have attracted researchers' attention due to their superior performance compared with their binary counterparts, especially for high rate and short length codes [1], [2]. However, this performance improvement is at the cost of a higher computation complexity of the message-passing decoding algorithm.

The q-ary sum-product algorithm (QSPA) [1], among different message-passing decoding algorithms, provides the best error performance. However, this algorithm is not effective in practical applications, since it has the highest complexity which needs $O(q^2)$ operations to update each check node (CN). Hence, two main branches of the QSPA algorithm are studied to reduce the complexity: 1) the fast Fourier transform SPAs (FFT-QSPA [3] and LOG-FFT-QSPA [4]), which need $O(q\log_2 q)$ operations to update each CN. 2) The extended Min-Sum (EMS) algorithm [5] and the Min-Max algorithm [6], which use n_m ($n_m < q$) most reliable messages for each symbol and reduce the complexity to $O(n_m \log_2 n_m)$. In addition, the trellis-based EMS (T-EMS) algorithm [7] is also presented to reduce the complexity. However, these message-passing decoding algorithms still have high complex CN updating processing.

Different from the message-passing decoding algorithms, the majority-logic decoding based algorithm (MLGD) [8] and symbol flipping decoding (SFD) algorithms [9], [10] present much lower decoding complexity. The MLGD algorithm only considers the most reliable field element for each symbol at the decoding processing. From the view of hardware implementation, the SFD algorithm has a much lower complexity, since it uses the hard reliability at the decoding processing. Most SFD algorithms flip the symbols according to the flipping metric. In the SFD algorithm, the oscillation phenomenon will cause the decoder trapping into the local maximum which leads to a decoding failure. Recently, a prediction mechanism with hard reliability has been considered in the SFD algorithm and can be abbreviated as the SFDP algorithm [10]. The SFDP algorithm considers the information before and after symbol flipping. It does not flip one symbol in two successive iterations to avoid the decoder being trapped into the oscillation. However, this strategy cannot completely avoid the oscillation and even cannot vanish the cycle oscillation, in which several variable nodes (VNs) are changed circularly in subsequent iterations.

To resolve the above problem, we consider redesigning the flipping metric of the SFDP algorithm by using a self-adjustment strategy. We record the number of times (NoTs) a symbol is flipped for each symbol with different symbol values. The NoTs can help the design of the flipping metric to escape the local maximum. The effectiveness of the proposed strategy will be evaluated on the AWGN channel and the BSC, followed by the complexity analysis.

Manuscript received October 3, 2018; revised January 27, 2019 and March 12, 2019; accepted April 11, 2019. Date of publication May 9, 2019; date of current version July 16, 2019. This work was supported by the National Natural Science Foundation of China under Grants 91438116 and 61871009. The review of this paper was coordinated by Prof. H.-F. Lu. *(Corresponding author: Rongke Liu.)*

B. Dai, R. Liu, and C. Gao are with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China (e-mail: daibinok@buaa.edu.cn; rongke_liu@buaa.edu.cn; gaochenyu@buaa.edu.cn).

Z. Mei is with the Science and Maths Cluster, Singapore University of Technology and Design, Singapore 487372 (e-mail: mei_zhen@sutd.edu.sg).

Digital Object Identifier 10.1109/TVT.2019.2915802

II. Background

A. Notation

The NB-LDPC code C is defined by the null space of an $m \times n$ row-column (RC)-constrained parity check matrix $\boldsymbol{H}$, with elements $h_{i,j} \in \mathrm{GF}(q)$, where $i = 0, 1, \ldots, m-1$ and $j = 0, 1, \ldots, n-1$. In this work, we consider $q = 2^d$. The index set of VNs connected to the i-th CN is defined as $N_i = \{j : 0 \le j < n, h_{i,j} \neq 0\}$, and the index set of CNs connected to the j-th VN is defined as $M_j = \{i : 0 \le i < m, h_{i,j} \neq 0\}$. $\boldsymbol{c} = (c_0, c_1, \ldots, c_{n-1})$, where $c_j \in GF(q)$, is a codeword and it satisfies $\boldsymbol{c} \cdot \boldsymbol{H}^T = \boldsymbol{0}$. The j-th coded symbol's binary tuple is defined as $c_j = (c_{j,0}, c_{j,1}, \ldots, c_{j,d-1})$. After the binary phase-shift keying (BPSK) modulation, each $c_{j,t}$ is mapped as $1 \to +1$ and $0 \to -1$, for $0 \le t < d$.

We denote $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ as the modulated symbol sequence. After the transmission over the AWGN channel or the BSC, the received symbol sequence is defined as $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n-1})$. We denote $\boldsymbol{z} = (z_0, z_1, \ldots, z_{n-1})$ as the hard-decision vector of each decoding iteration. The syndrome $\boldsymbol{s} = (s_0, s_1, \ldots, s_{m-1})$ corresponding to $\boldsymbol{z}$ is calculated by $\boldsymbol{s} = \boldsymbol{z} \cdot \boldsymbol{H}^T$. Note that if and only if $\boldsymbol{s} = \boldsymbol{0}$, $\boldsymbol{z}$ is a valid codeword.

B. The SFDP Algorithm

For binary LDPC codes, the gradient descent bit-flipping algorithm [11] follows the maximum likelihood (ML) decoding rule to obtain the gradient descent optimization of an objective function. The objective function of the SFDP algorithm can be defined as [10]

$$F^{(k)} = \sum_{j=0}^{n-1} z_j^{(k)} \odot y_j + \sum_{i=0}^{m-1} \omega(s_i^{(k)}), \tag{1}$$

where the operator $\odot$ is defined as

$$z_j^{(k)} \odot y_j = \sum_{t=0}^{d-1} (2z_{j,t}^k - 1) y_{j,t}. \tag{2}$$

And $\omega(s_i^{(k)})$ is the check-based message which measures the hard reliability in the SFDP algorithm. However, it is difficult to obtain the derivative of the objective function for non-binary LDPC codes. The difference between one symbol and its change in value can be indicated as the reliability of this symbol. Hence, this difference can be seen as the flipping metric.

To make full use of the check information, the SFDP algorithm [10] introduces hard reliability to the flipping metric. Furthermore, $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_d]$ and $\boldsymbol{\eta} = [\eta_0, \eta_1, \ldots, \eta_\gamma]$ are the extrinsic weighting coefficients vectors corresponding to the binary Hamming distance and plurality logic, which can measure the hard reliability. The SFDP algorithm uses the binary Hamming distance to measure the hard reliability, which can be referred to as the D-SFDP algorithm. Hence, its flipping metric is expressed as

$$E_j^{(k)}(z_j^{*(k)}, z_j^{(k)})_{\text{D-SFDP}} = z_j^{*(k)} \odot y_j + \sum_{i \in M_j} \theta_{\underset{d}{\rightarrow}(z_j^{*(k)}, \sigma_{i,j}^{(k)})} - z_j^{(k)} \odot y_j - \sum_{i \in M_j} \theta_{\underset{d}{\rightarrow}(z_j^{(k)}, \sigma_{i,j}^{(k)})}, \quad (3)$$

where $z_j^{*(k)}$ is the change of the j-th symbol and $\underset{d}{\rightarrow}(z_j^{(k)}, \sigma_{i,j}^{(k)})$ denotes the binary Hamming distance between $z_j^{(k)}$ and $\sigma_{i,j}^{(k)}$ with their binary representations in the k-th iteration. In addition, $\sigma_{i,j}^{(k)}$ denotes the extrinsic information-sum (EXI) passed from the i-th CN to the j-th VN

$$\sigma^{(k)}_{i,j} = h_{i,j}^{-1} \sum_{j' \in N(i) \backslash j} h_{i,j'} z_{j'}^{(k)}, \quad (4)$$

where $j \in N(i)$ and $0 \le i < m$.

Similarly, the SFDP algorithm combines the plurality logic to the flipping metric, referred to as the P-SFDP algorithm. The flipping metric of the P-SFDP algorithm is

$$E_j^{(k)}(z_j^{*(k)}, z_j^{(k)})_{\text{P-SFDP}} = z_j^{*(k)} \odot y_j + \eta_{n(z_j^{*(k)})} - z_j^{(k)} \odot y_j - \eta_{n(z_j^{(k)})}. \quad (5)$$

where $n(z_j^{(k)})$ is the number of times $z_j^{(k)}$ occurs among the EXIs passed from all CNs connected to the j-th VN.

III. Symbol Flipping Algorithm With Self-Adjustment Strategy

A. The Self-Adjustment Strategy

By using the prediction mechanism, the SFDP algorithms provide a significant improvement compared with other SFD algorithms. However, as a class of gradient descent (GD) algorithms, the SFDP algorithms still suffer from the problem that the decoding may be trapped into a local maximum. The straightforward way to jump out of the trap is to adjust the step size of the GD algorithm. Unfortunately, since the value of VNs is equal to one element of $GF(q)$, the step size is limited. As a result, we should find another way to jump out of the trap instead of adjusting the step size.

When the algorithm is trapped into a local maximum, the VNs may drop into an oscillation. In the extreme case, several VNs are changed circularly in subsequent iterations, which can be regarded as the cycle oscillation. To avoid the cycle oscillation, we consider reducing the probability of the symbol being flipped over repeatedly. Hence, we record NoTs of each flipped symbol with a symbol value and then redesign the flipping metric $E_j^{(k)}(z_j^{*(k)}, z_j^{(k)})$ by subtracting the weighted NoTs of $z_j^{*(k)}$. We call it the self-adjustment strategy. Denote $t_{z_j^*}^{(k)}$ as the NoTs of $z_j^{*(k)}$. To jump out of the local maximum, the flipping metric of the D-SFDP algorithm with the self-adjustment strategy (referred to as the D-SA-SFDP algorithm) is given as

$$E_j^{(k)}(z_j^{*(k)}, z_j^{(k)})_{\text{D-SA-SFDP}} = E_j^{(k)}(z_j^{*(k)}, z_j^{(k)})_{\text{D-SFDP}} - \omega_t \cdot t_{z_j^*}^{(k)}, \quad (6)$$

The flipping metric for the P-SFDP algorithm with the self-adjustment strategy (referred to as the P-SA-SFDP algorithm) is

$$E_j^{(k)}(z_j^{*(k)}, z_j^{(k)})_{\text{P-SA-SFDP}} = E_j^{(k)}(z_j^{*(k)}, z_j^{(k)})_{\text{P-SFDP}} - \omega_t \cdot t_{z_j^*}^{(k)}, \quad (7)$$

where ω_t is a weighted value with different positive values for different codes. If there exists a cycle oscillation, there must be a VN which has been chosen more than once. However, by adding the self-adjustment strategy, the more times the VN is flipped, the more difficult it is to be selected, and the cycle oscillation can be broken. Hence, the proposed self-adjustment strategy can vanish the oscillation near a local maximum. The details of the proposed strategy are summarized in Algorithm 1.

Algorithm 1: The D-SA-SFDP and P-SA-SFDP Algorithms.

1: Initialization: Set $k = 0$ and $t_{z_j}^{(0)} = 0$ for $j = 1, \ldots, n$ and $z_j = 1, \ldots, q$. Obtain the hard-decision vector $z^{(0)}$ by $z_{j,t}^{(0)} = 1$ if $y_{j,t} > 0$. Otherwise, $z_{j,t}^{(0)} = 0$ for $j = 1, \ldots, n$ and $t = 1, \ldots, d$.
2: Compute the flipping metric of the D-SA-SFDP algorithm by (6), or compute the flipping metric of the P-SA-SFDP algorithm by (7).
3: Determine $p^{(k)}$ as $E_{p^{(k)}} = \max_{z_j^{*(k)} \in \Gamma_j^k} E_j^k(z_j^{*(k)}, z_j^{(k)})$ and flip the $p^{(k)}$-th symbol to $v_p^{(k)} = \arg\max_{z_j^{*(k)} \in \Gamma_j^k} E_j^k(z_j^{*(k)}, z_j^{(k)})$, where $v_p^{(k)}$ is the corresponding flipped value. Then $k = k + 1$ and $t_{v_p}^{(k+1)} = t_{v_p}^{(k)} + 1$.
4: If $z^k \cdot H^T = 0$ or the maximum number of iterations is reached, output and stop. Otherwise, return to step 2.

B. Complexity Analysis

To facilitate the complexity analysis, we consider the regular NB-LDPC code which has constant row weight ρ and column weight γ. Compared with the SFDP (D-SFDP and P-SFDP) algorithms, the SA-SFDP (D-SA-SFDP and P-SA-SFDP) algorithms introduce a self-adjustment strategy to the flipping metric. Hence, the computational complexity of the SA-SFDP algorithms can be divided into two steps: 1) the computations of the SFDP algorithms which are shown in [10]. 2) the computation of the self-adjustment strategy. If and only if the p-th symbol has been flipped to its corresponding flipped value v_p before the k-th iteration, $t_{v_p}^{(k)}$ is not equal to 0. In addition, we can ignore the operation of multiplying zero or adding zero. Thus, the second step is only needed for the flipped symbols. The maximum number of flipped symbols to their corresponding values can be $\min(k, nq)$. Hence, no more

TABLE I
COMPUTATIONAL COMPLEXITY OF VARIOUS DECODERS IN EACH ITERATION

Decoders	Number of Computation					Memory Computation
	GM	GA	IM/RM	IA/RA	IC/RC	
D-SFDP [10]	C_1	$2\gamma(\rho-1)$ $+C_2\gamma(d+\gamma)$	$--$	$(\gamma^2+4d+2\gamma d+2\gamma)C_2$	$2(n-1)+(\gamma+d-1)C_2$	$ndb+(2n+n\gamma)d$ $+(q+n+\gamma)b+\delta$
P-SFDP [10]	C_1	$2\gamma(\rho-1)$	$--$	$(\gamma d+3\gamma+5d+1)C_2$	$2(n-1)+(\gamma+d-1)C_2$	$ndb+(2n+n\gamma)d$ $+(q+n)b+\delta+q\beta$
D-SA-SFDP	C_1	$2\gamma(\rho-1)$ $+C_2\gamma(d+\gamma)$	$\min(k,nq)$	$(\gamma^2+4d+2\gamma d+2\gamma)C_2$ $+\min(k,nq)+1$	$2(n-1)+(\gamma+d-1)C_2$	$n(d+q)b+(2n+n\gamma)d$ $+(q+n+\gamma+1)b+\delta$
P-SA-SFDP	C_1	$2\gamma(\rho-1)$	$\min(k,nq)$	$(\gamma d+3\gamma+5d+1)C_2$ $+\min(k,nq)+1$	$2(n-1)+(\gamma+d-1)C_2$	$n(d+q)b+(2n+n\gamma)d$ $+(q+n+1)b+\delta+q\beta$

GA/GM: Galois Field Addition/Multiplication IC/IA/IM: Integer Comparison/Addition/Multiplication RC/RA/RM: Real Comparison/Addition/Multiplication, $C_1=\gamma(2\rho-1), C_2=(\gamma\rho+1-\gamma), \delta=\lceil\log_2 n\rceil, \beta=\lceil\log_2\gamma\rceil$.

than $\min(k,nq)$ real additions and real multiplications are required for the computation of the self-adjustment strategy. In each iteration, one integer addition is needed to compute $t_{v_p}^{(k+1)}=t_{v_p}^{(k)}+1$. Compared with the SFDP algorithms, the proposed SA-SFDP algorithms only add a small number of real multiplications and real additions. The increased computation is small compared with the overall complexity. The computational complexity of different algorithms is shown in Table I. The memory consumption of the proposed algorithms is also analyzed. Similar to the SFDP algorithms, assume that each Galois field element requires d bits to store and each reliability metric needs b bits. It requires nqb bits and b bits to store $t_{z_j^*}^{(k)}$ and ω_t. The proposed SA-SFDP algorithms need more memory consumption than the SFDP algorithms, while the increased memory consumption can be negligible compared with the overall memory consumption. The overall memory consumption of various decoders is shown in Table I.

IV. SIMULATION RESULTS

The BER results of the proposed SA-SFDP algorithms are compared with the SFDP algorithms [10] over the AWGN channel and the BSC. The maximum number of iterations $k_{\max}$ for all the decoding algorithms is set to 100. Since the parameters optimization depends on many factors including the degree distribution and the Galois field order, the parameters are obtained experimentally by computer search for a particular NB-LDPC code. Through the Monte Carlo simulations, we first optimize the $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ for the D-SA-SFDP and P-SA-SFDP algorithms with $\omega_t=0$. Then, based on the optimized $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$, we obtain the optimal ω_t by simulating the BER results of different ω_t values. As shown in [10], the coefficients $\theta_u=\theta_2$ for $3\le u\le d$ and $\eta_\gamma=\eta_{\gamma-1}$ for $\gamma\ge 3$. The BER performance comparisons of three codes are shown below.

A. Code 1

The first code is a (204, 102) NB-LDPC code [12] over GF(2^4), in which the row weight is 6 and the column weight is 3. Fig. 1 shows the BER performance of the D-SA-SFDP algorithm under different coefficients $(\theta_1,\theta_2,\theta_3)$ with $\omega_t=0$. Then, we choose the coefficients $\boldsymbol{\theta}$ that lead to the best BER performance from a series of parameters through the Monte Carlo simulations. Similarly, we can optimize $\boldsymbol{\eta}$ for the P-SA-SFDP algorithm in the same way. We omit the figure to show the BER performance of the P-SA-SFDP algorithm with different coefficients $(\eta_1,\eta_2.\eta_3,\eta_4)$, since it is difficult to visualize the optimization process of four coefficients. The optimal $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ of the D-SA-SFDP and P-SA-SFDP algorithms are [3, 1.4, 1.2] and [0, 2, 4,6], respectively. Fig. 2 shows the BER performance of different ω_t at $E_b/N_0=5$ dB. From Fig. 2, it can be seen that adding the weighted value ($\omega_t>0$) has an great effect on the performance of the D-SA-SFDP and P-SA-SFDP algorithms. However, when ω_t is greater than 0.4, the change of ω_t value has little effect on the performance. The ω_t of the D-SA-SFDP and the P-SA-SFDP algorithms are both optimized as 1.2. The BER results of various symbol flipping decoding algorithms with code 1 on the AWGN channel are shown in Fig. 3. At BER $=10^{-6}$, the proposed SA-SFDP algorithms present similar performance, which are about 0.7 dB better than the SFDP algorithms.

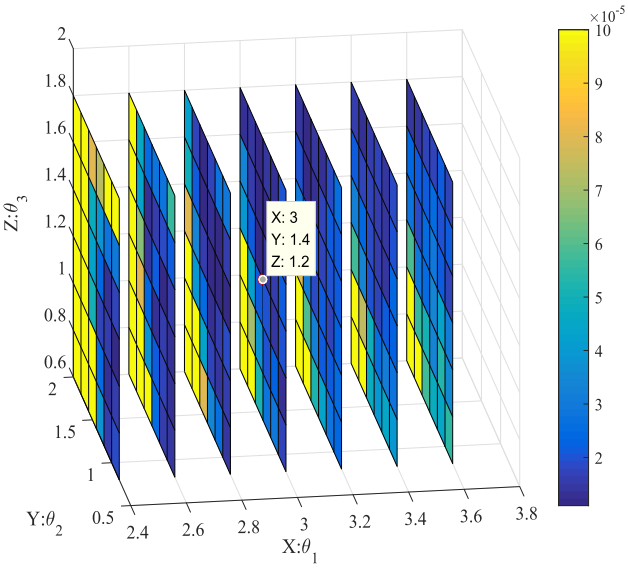

Fig. 1. The BER performance for different θ on the (204, 102) NB-LDPC code at $E_b/N_0=5$ dB.

Fig. 4 shows the BER results for different algorithms as a function of maximum number of iterations (MNI) at $E_b/N_0=4.5$ dB. At each MNI, 10^8 frames are simulated. It can be seen that all algorithms perform similar when MNI <60. As the MNI increases, the SFDP algorithms occur a decoding error floor. While the BER of the proposed SA-SFDP algorithms can reach 10^{-5} at MNI $=100$. This result shows that the proposed self-adjustment strategy can effectively solve the oscillation problem.

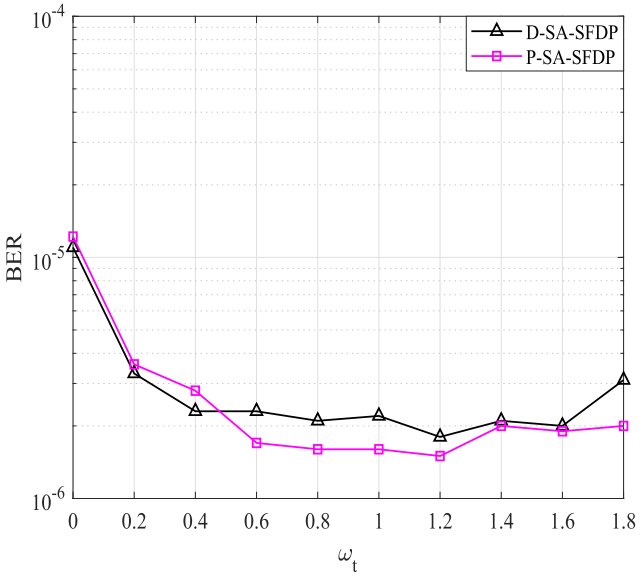

Fig. 2. The BER performance for different ω_t on the (204, 102) NB-LDPC code at $E_b/N_0 = 5$ dB.

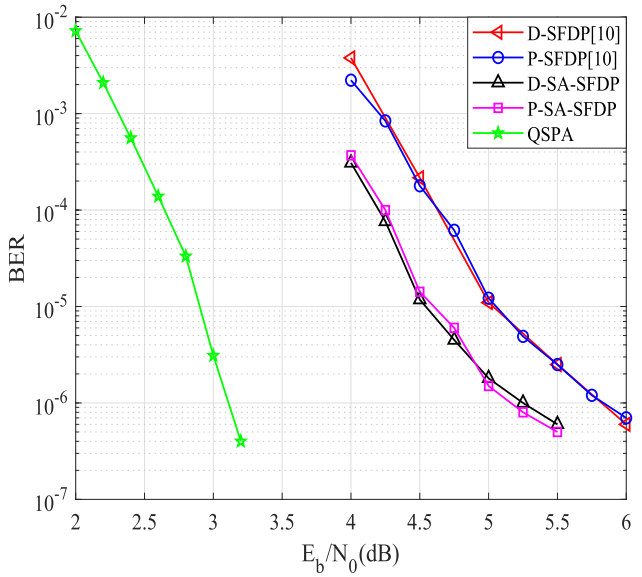

Fig. 3. BER performance for various decoding algorithms on the (204, 102) NB-LDPC code.

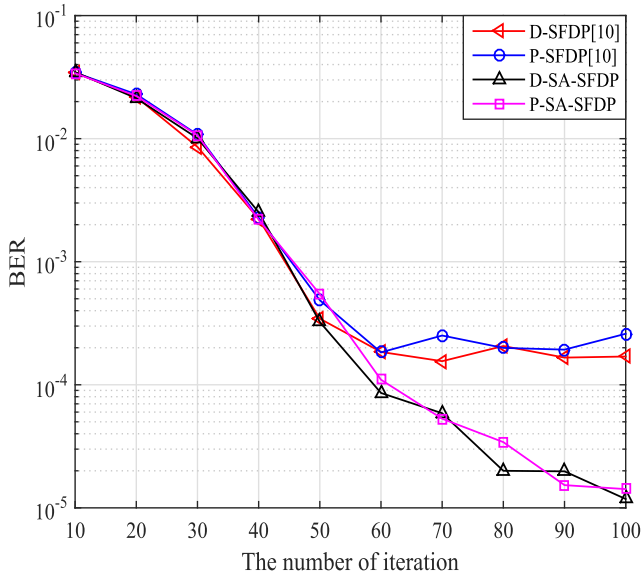

Fig. 4. The BER performance for different algorithms as a function of maximum number of iterations on the (204, 102) NB-LDPC code at $E_b/N_0 = 4.5$ dB.

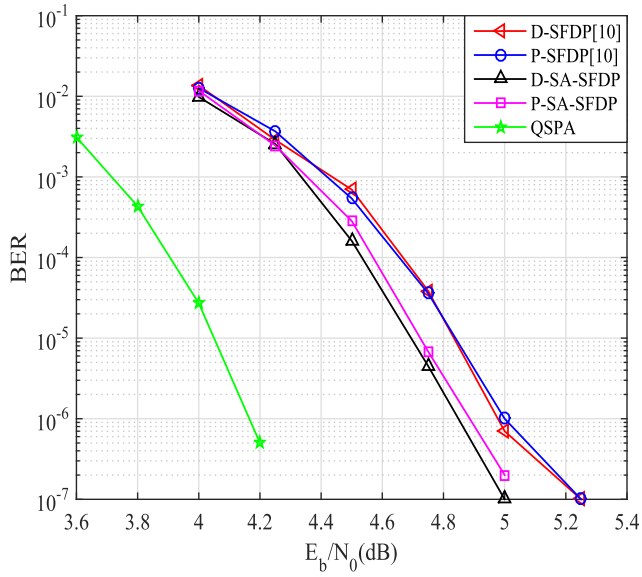

Fig. 5. BER performance for various decoding algorithms on the (837, 726) NB-LDPC code.

B. Code 2

The second code is a (837, 726) NB-LDPC code [13] over GF(2^5), in which the row weight is 27 and the column weight is 4. The optimal $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ of the D-SA-SFDP and P-SA-SFDP algorithms are [1.1, 0.3, 0.3] and [0, 0.9, 1.7, 3], respectively. Note that the parameters optimization process for the D-SA-SFDP and P-SA-SFDP algorithms on code 2 and code 3 are the same as code 1. The ω_t of the D-SA-SFDP and the P-SA-SFDP algorithms are both optimized as 0.4. The BER results of various symbol flipping decoding algorithms on code 2 over the AWGN channel are shown in Fig. 5. The D-SA-SFDP algorithm performs similarly to the P-SA-SFDP algorithm. At BER $= 10^{-7}$, the D-SA-SFDP algorithm achieves about 0.1 dB gain compared with the D-SFDP algorithm. Meanwhile, the P-SA-SFDP algorithm outperforms its corresponding P-SFDP algorithm with about 0.25 dB gain.

C. Code 3

The third code is a (256, 128) NB-LDPC code [14] over GF(2^8), in which the row weight is 8 and the column weight is 4. The optimal $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ of the D-SA-SFDP and P-SA-SFDP algorithms are [3.5, 1.0, 1.0] and [0, 2.8, 6.4, 8.4], respectively. The ω_t of the D-SA-SFDP and the P-SA-SFDP algorithms are both optimized as 1. Fig. 6 shows the BER

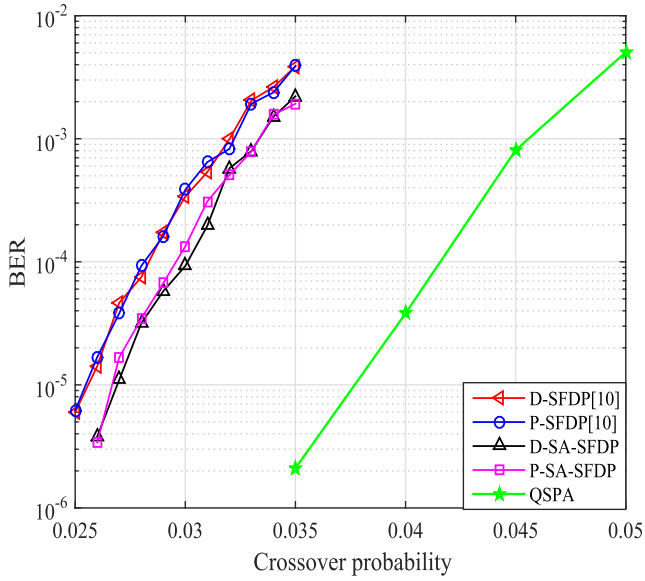

Fig. 6. BER performance for various decoding algorithms on the (256, 128) NB-LDPC code.

results of code 3 on the BSC. It can be also observed that the proposed SA-SFDP algorithms perform better than the SFDP algorithms.

V. Conclusion

We have presented a modified symbol flipping algorithm by adding a self-adjustment strategy to the flipping metric. The new flipping metric not only takes advantage of the SFDP algorithms, but also effectively avoids the cycle oscillation. Hence, the proposed SA-SFDP algorithms can obtain a better performance compared with the SFDP algorithms with a negligible complexity increase. The simulation results have been presented to show the validity of our proposed SA-SFDP algorithms over two channels (the AWGN channel and the BSC). The proposed algorithms can achieve $0.1 \sim 0.25$ dB gain for code 1 and 0.7 dB gain for code 2 over the SFDP algorithms. For the BSC, the proposed algorithms also present obvious better performance compared with the SFDP algorithms.

Acknowledgment

The authors would like to thank Prof. Q. Huang for providing them NB-LDPC codes that were used in simulation.

References

[1] 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.
[2] L. Zeng, L. Lan, 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.
[3] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over GF(2^q)," in *Proc. Inf. Theory Workshop*, Mar. 2003, pp. 70–73.
[4] H. Song and J. Cruz, "Reduced-complexity decoding of q-ary LDPC codes for magnetic recording," *IEEE Trans. Magn.*, vol. 39, no. 2, pp. 1081–1087, Mar. 2003.
[5] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low complexity decoding for non-binary LDPC codes in high order fields," *IEEE Trans. Commun.*, vol. 58, no. 5, pp. 1365–1375, May 2010.
[6] V. Savin, "Min-max decoding for non-binary LDPC codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2008, pp. 960–964.
[7] E. Li, D. Declercq, and K. Gunnam, "Trellis-based extended min-sum algorithm for non-binary LDPC codes and its hardware structure," *IEEE Trans. Commun.*, vol. 61, no. 7, pp. 2600–2611, Jul. 2013.
[8] C. Y. Chen, Q. Huang, C. C. Chao, and S. Lin, "Two low-complexity reliability-based message-passing algorithms for decoding non-binary LDPC codes," *IEEE Trans. Commun.*, vol. 58, no. 11, pp. 3140–3147, Nov. 2010.
[9] C.-C. Huang, C.-J. Wu, C.-Y. Chen, and C.-C. Chao, "Parallel symbol flipping decoding for non-binary LDPC codes," *IEEE Commun. Lett.*, vol. 17, no. 6, pp. 1228–1231, Jun. 2013.
[10] S. Wang, Q. Huang, and Z. Wang, "Symbol flipping decoding algorithms based on prediction for non-binary LDPC codes," *IEEE Trans. Commun.*, vol. 65, no. 5, pp. 1913–1924, May 2017.
[11] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, "Gradient descent bit flipping algorithms for decoding LDPC codes," *IEEE Trans. Commun.*, vol. 58, no. 6, pp. 1610–1614, Jun. 2010.
[12] N. Q. Nhan, T. M. N. Ngatched, O. A. Dobre, P. Rostaing, K. Amis, and E. Radoi, "Multiple-votes parallel symbol-flipping decoding algorithm for non-binary LDPC codes," *IEEE Commun. Lett.*, vol. 19, no. 6, pp. 905–908, Jun. 2015.
[13] J. O. Lacruz, F. Garcia-Herrero, J. Valls, and D. Declercq, "One minimum only trellis decoder for non-binary low-density parity-check codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 1, pp. 177–184, Jan. 2015.
[14] J. Li, K. Liu, S. Lin, and K. Abdel-Ghaffar, "A matrix-theoretic approach to the construction of non-binary quasi-cyclic LDPC codes," *IEEE Trans. Commun.*, vol. 63, no. 4, pp. 1057–1068, Apr. 2015.