Gradient Descent Bit Flipping Algorithms for Decoding LDPC Codes

Tadashi Wadayama, Keisuke Nakamura, Masayuki Yagita,
Yuuki Funahashi, Shogo Usami, and Ichi Takumi

***Abstract*—A novel class of bit-flipping (BF) algorithm for decoding low-density parity-check (LDPC) codes is presented. The proposed algorithms, which are referred to as *gradient descent bit flipping (GDBF) algorithms*, can be regarded as simplified gradient descent algorithms. The proposed algorithms exhibit better decoding performance than known BF algorithms, such as the weighted BF algorithm or the modified weighted BF algorithm for several LDPC codes.**

***Index Terms*—LDPC code, bit-flipping algorithm, gradient descent algorithm.**

I. Introduction

BIT-FLIPPING (BF) algorithms for decoding low-density parity-check (LDPC) codes [1] have been investigated extensively and numerous variants of BF algorithms, such as the weighted BF (WBF) algorithm [4], the modified weighted BF (MWBF) algorithm [8], and other variants [2], [3], [5], have been proposed.

Although the bit error rate (BER) performance of the BF algorithm is inferior to that of the sum-product algorithm or the min-sum algorithm, in general, the BF algorithm enables the design of a much simpler decoder, which is easier to implement. Thus, bridging the performance gap between BF decoding and belief propagation (BP) decoding is an important technical challenge.

In the present letter, a novel class of BF algorithm for decoding LDPC codes is proposed. The proposed algorithms, which are referred to as *gradient descent bit flipping (GDBF) algorithms*, can be regarded as bit-flipping gradient descent algorithms. The proposed algorithms are naturally derived from a simple gradient descent formulation. The behavior of the proposed algorithms can be explained from the viewpoint of the optimization of a non-linear objective function.

II. Preliminaries

A. Notation

Let H be a binary $m \times n$ parity check matrix, where $n > m \geq 1$. The binary linear code C is defined by $C \stackrel{\triangle}{=} \{\boldsymbol{c} \in F_2^n : H\boldsymbol{c} = \boldsymbol{0}\}$, where F_2 denotes the binary Galois field. In the present letter, a vector is assumed to be a column vector. By convention, we introduce the bipolar codes $\tilde{C}$ corresponding to C as follows: $\tilde{C} \stackrel{\triangle}{=} \{(1-2c_1, 1-2c_2, \ldots, 1-2c_n) : \boldsymbol{c} \in C\}$. Namely, $\tilde{C}$, which is a subset of $\{+1,-1\}^n$, is obtained from C using binary $(0,1)$ to bipolar $(+1,-1)$ conversion.

In the present letter, we assume the binary-input AWGN channel, which is defined as $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{z}$ $(\boldsymbol{c} \in \tilde{C})$. The vector $\boldsymbol{z} = (z_1, \ldots, z_n)$ is a white Gaussian noise vector, where $z_j (j \in [1,n])$ is an i.i.d. Gaussian random variable with zero mean and variance σ^2. The notation $[a,b]$ denotes the set of consecutive integers from a to b.

Let $N(i)$ and $M(j) (i \in [1,m], j \in [1,n])$ be $N(i) \stackrel{\triangle}{=} \{j \in [1,n] : h_{ij} = 1\}$ and $M(j) \stackrel{\triangle}{=} \{i \in [1,m] : h_{ij} = 1\}$, where h_{ij} is the ij-element of the parity check matrix H. Using this notation, we can write the parity condition as: $\prod_{j \in N(i)} x_j = 1 (\forall i \in [1,m])$ which is equivalent to $(x_1, \ldots, x_n) \in \tilde{C}$. The value $\prod_{j \in N(i)} x_j \in \{+1,-1\}$ is referred to as the i-th *bipolar syndrome* of $\boldsymbol{x}$.

B. Brief review of known BF algorithms

A number of variants of BF algorithms have been developed. The BF algorithms can be classified as either single bit flipping (single BF) algorithms or multiple bits flipping (multi BF) algorithms. In the decoding process of the single BF algorithm, only one bit is flipped according to its bit flipping rule. On the other hand, the multi BF algorithm allows multiple bit flipping during each iteration of a decoding process. In general, although the multi BF algorithm exhibits faster convergence than the single BF algorithm, the multi BF algorithm suffers from the oscillation behavior of a decoder state, which is not easy to control.

In a decoding process of a BF algorithm, bit flip positions are determined based on the values of an *inversion function*. The inversion functions of WBF [4] are defined by

$$\Delta_k^{(WBF)}(\boldsymbol{x}) \stackrel{\triangle}{=} \sum_{i \in M(k)} \beta_i \prod_{j \in N(i)} x_j, \tag{1}$$

where $\boldsymbol{x} = (x_1, \ldots, x_n) \in \{+1,-1\}$. The values $\beta_i (i \in [1,m])$ are the *reliabilities* of bipolar syndromes defined by $\beta_i \stackrel{\triangle}{=} \min_{j \in N(i)} |y_j|$. In this case, the inversion function $\Delta_k^{(WBF)}(\boldsymbol{x})$ provides a measure of the invalidness of symbol assignment on x_k, which is given by the sum of the weighted bipolar syndromes. Although the inversion function of the MWBF algorithm [8] has a similar form to the inversion function of the WBF algorithm, the inversion function of the MWBF algorithm contains a term corresponding to a received symbol. The inversion function of the MWBF algorithm is

Paper approved by F. Fekri, the Editor for LDPC Codes and Applications of the IEEE Communications Society. Manuscript received January 22, 2009; revised September 15, 2009.

T. Wadayama, K. Nakamura, M. Yagita, and I. Takumi are with the Nagoya Institute of Technology, Nagoya, 466-8555, Japan (e-mail: wadayama@nitech.ac.jp).

Y. Funahashi and S. Usami are with Meijo University.

A part of this letter was presented at the International Symposium on Information Theory and its Applications, Auckland, Dec. 2008, and was posted to ArXiv (indicator:0711.0261v1).

Digital Object Identifier 10.1109/TCOMM.2010.06.090046

given by

$$\Delta_k^{(MWBF)}(\boldsymbol{x}) \stackrel{\triangle}{=} \alpha|y_k| + \sum_{i\in M(k)} \beta_i \prod_{j\in N(i)} x_j, \qquad (2)$$

where the parameter α is a positive real number.

The framework of the single BF algorithm is summarized as follows:

Single BF algorithm

Step 1 For $j \in [1,n]$, let $x_j := \text{sign}(y_j)$, where $\text{sign}(x) \stackrel{\triangle}{=} +1$ if $x \geq 0$. Otherwise $\text{sign}(x) \stackrel{\triangle}{=} -1$. Let $\boldsymbol{x} \stackrel{\triangle}{=} (x_1, x_2, \ldots, x_n)$.

Step 2 If the parity equation $\prod_{j\in N(i)} x_j = +1$ holds for all $i \in [1,m]$, output $\boldsymbol{x}$, and then exit.

Step 3 Find the flip position given by $\ell := \arg\min_{k\in[1,n]} \Delta_k(\boldsymbol{x})$ and then flip the symbol: $x_\ell := -x_\ell$. The function $\Delta_k(\boldsymbol{x})$ is referred to as an inversion function.

Step 4 If the number of iterations is less than the maximum number of iterations L_{max}, then return to Step 2; otherwise output $\boldsymbol{x}$ and exit.

In a decoding process of the single BF algorithm, hard decision decoding for a given $\boldsymbol{y}$ is first performed, and $\boldsymbol{x}$ is initialized to the hard decision result. The minimum of the inversion function $\Delta_k(\boldsymbol{x})$ for $k \in [1,n]$ is then found[1]. An inversion function $\Delta_k(\boldsymbol{x})$ can be considered to be a measure of the invalidness of bit assignment on x_k. The bit x_ℓ, where ℓ gives the smallest value of the inversion function, is then flipped.

III. Gradient Descent Formulation

A. Objective function

It seems natural to consider the dynamics of a BF algorithm as a minimization process of a hidden objective function. This observation leads to a gradient descent formulation of BF algorithms.

The maximum likelihood (ML) decoding problem for the binary AWGN channel is equivalent to the problem of finding a (bipolar) codeword in $\tilde{C}$, which gives the largest correlation to a given received word $\boldsymbol{y}$. Namely, the MLD rule can be written as $\hat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}\in\tilde{C}} \sum_{j=1}^n x_i y_i$.

Based on this correlation decoding rule, we define the following objective function[2]:

$$f(\boldsymbol{x}) \stackrel{\triangle}{=} \sum_{i=1}^{n} x_j y_j + \sum_{i=1}^{m} \prod_{j\in N(i)} x_j. \qquad (3)$$

The first term of the objective function corresponds to the correlation between a bipolar codeword and the received word, which should be maximized. The second term is the sum of the bipolar syndromes of $\boldsymbol{x}$. The second term has a maximum value $\sum_{i=1}^m \prod_{j\in N(i)} x_j = m$ if and only if $\boldsymbol{x} \in \tilde{C}$. Thus, this term can be considered to be a *penalty term*, which forces $\boldsymbol{x}$ to be a valid codeword.

Note that this objective function is a non-linear function and has many local maxima. These local maxima become a major source of sub-optimality of the GDBF algorithm presented later herein.

B. Gradient descent BF algorithm

For the numerical optimization problem of a differentiable function, such as (3), the gradient descent method is a natural choice for the first attempt. Namely, we consider the following maximization problem:

$$\text{maximize } f(\boldsymbol{x})$$

for the decoding of $\tilde{C}$ and apply the gradient ascent method to find a local maximum.

The partial derivative of $f(\boldsymbol{x})$ with respect to the variable $x_k (k \in [1,n])$ can be immediately derived from the definition of $f(\boldsymbol{x})$:

$$\frac{\partial}{\partial x_k} f(\boldsymbol{x}) = y_k + \sum_{i\in M(k)} \prod_{j\in N(i)\backslash k} x_j. \qquad (4)$$

Let us consider the product of x_k and the partial derivative of x_k in $\boldsymbol{x}$, namely,

$$x_k \frac{\partial}{\partial x_k} f(\boldsymbol{x}) = x_k y_k + \sum_{i\in M(k)} \prod_{j\in N(i)} x_j. \qquad (5)$$

For a small real number s, we have the first-order approximation: $f(x_1, \ldots, x_k + s, \ldots, x_n) \simeq f(\boldsymbol{x}) + s\frac{\partial}{\partial x_k} f(\boldsymbol{x})$. When $\frac{\partial}{\partial x_k} f(\boldsymbol{x}) > 0$, we need to choose $s > 0$ in order to obtain $f(x_1, \ldots, x_k + s, \ldots, x_n) > f(\boldsymbol{x})$. On the other hand, if $\frac{\partial}{\partial x_k} f(\boldsymbol{x}) < 0$ holds, we should choose $s < 0$ in order to obtain the above inequality. Therefore, flipping the kth symbol $(x_k := -x_k)$ is highly likely to increase the objective function value if (i) $x_k \frac{\partial}{\partial x_k} f(\boldsymbol{x}) < 0$ holds and (ii) flipping induces a relatively small change in a search point[3].

One reasonable way to find a flipping position is to choose the position at which the absolute value of the partial derivative is largest. This flipping rule is closely related to the steepest descent algorithm based on ℓ_1-norm also known as the *coordinate descent algorithm*[4]. According to this observation, we use the following rule to choose the flipping position.

Definition 1 (Inversion function of the GDBF algorithm): The single BF algorithm based on the inversion function

$$\Delta_k^{(GD)}(\boldsymbol{x}) \stackrel{\triangle}{=} x_k y_k + \sum_{i\in M(k)} \prod_{j\in N(i)} x_j \qquad (6)$$

is referred to as the gradient descent BF (GDBF) algorithm[5]. Thus, the decoding process of the GDBF algorithm can be seen as the minimization process of $-f(\boldsymbol{x})$ (it can be

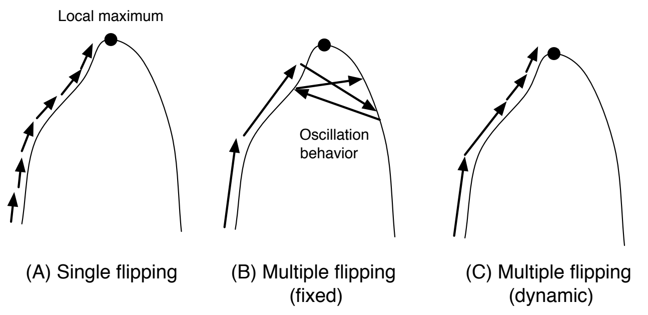
[1]When $\Delta_k(\boldsymbol{x})$ is an integer-valued function, a tie-break rule is required in order to resolve a tie.

[2]This objective function has been inspired by the correlation MLD rule. There are several possible objective functions for mimicking the correlation MLD rule. In the present letter, we investigate one of the simplest objective functions in such a class.

[3]The possibility exists that the objective function value decreases because the step size is fixed.

[4]Note that an algorithm based on coordinate descent approximation of the min-sum and LP decoder was proposed by Vontobel and Koetter [7].

[5]Strictly speaking, the algorithm should be referred to as the gradient *ascent* BF algorithm. However, based on the equivalent minimization problem, we adopt the name gradient descent BF (GDBF) algorithm.


Local maximum
Oscillation behavior
(A) Single flipping
(B) Multiple flipping (fixed)
(C) Multiple flipping (dynamic)

(A) converging but slow, (B) not converging but fast, (C) converging and fast

Fig. 1. Convergence behavior of a search point in GDBF decoding process.

considered as the *energy* of the system) based on the *bit-flipping* gradient descent method.

The combination of the objective function $\tilde{f}(\boldsymbol{x})$ defined by

$$\tilde{f}(\boldsymbol{x}) \overset{\triangle}{=} \alpha \sum_{i=1}^{n} x_j |y_j| + \sum_{i=1}^{m} \beta_i \prod_{j \in N(i)} x_j \qquad (7)$$

and the argument on gradient descent presented above gives the inversion functions of conventional algorithms such as the WBF algorithm (1) and the MWBF algorithm (2).

C. Multi GDBF algorithm

A decoding process of the GDBF algorithm can be regarded as a maximization process of the objective function (3) in a gradient ascent manner. Thus, we can use the objective function value to observe the convergence behavior. For example, it is possible to monitor the value of the objective function for each iteration. In the first several iterations, the value increases as the number of iterations increases. However, the value eventually ceases to increase when the search point arrives at the nearest point in $\{+1,-1\}^n$ to the local maximum of the objective function. We can easily detect such convergence to a local maximum by observing the value of the objective function.

Both the BF algorithms reviewed in the previous section and the GDBF algorithm flip only one bit for each iteration. In terms of the numerical optimization, in these algorithms, a search point moves toward a local maximum with a very small step (i.e., 1 bit flip) in order to avoid oscillation around the local maximum (See Fig.1(A)). However, the small step size leads to slower convergence to a local maximum.

The multi bit flipping algorithm is expected to converge faster than the single bit flipping algorithm because of its larger step size. If the search point is close to a local maximum, a fixed large step is not suitable for finding a (nearby) local maximum point and leads to oscillation in a multi-bit flipping BF algorithm (Fig.1(B)). It is necessary to adjust the step size dynamically from a large step size to a small step size in an optimization process (Fig.1(C)).

The objective function provides a guideline for adjusting the step size (i.e., number of flipping bits). The *multi GDBF algorithm* is a GDBF algorithm that incorporates the multi-bit flipping concept. In the following, we assume the inversion function $\Delta_k^{(GD)}(\boldsymbol{x})$ defined by (6) (the inversion function for the GDBF algorithm). The flow of the multi GDBF algorithm is approximately the same as that of the previously presented GDBF algorithm. When it is necessary to clearly distinguish between two decoding algorithms, the GDBF algorithm presented in the previous sub-subsection is referred to as the single GDBF algorithm.

In order to define the multi GDBF algorithm, we need to introduce the parameters θ and μ. The parameter θ is a negative real number, which is called the *inversion threshold*. The binary (0 or 1) variable μ, which is called the *mode flag*, is set to 0 at the beginning of the decoding process. Step 3 of the BF algorithm should be replaced with the following multi-bit flipping procedure:

Step 3 Evaluate the value of the objective function, and let $f_1 := f(\boldsymbol{x})$. If $\mu = 0$, then execute Sub-step 3-1 (multi-bit mode), else execute Sub-step 3-2 (single-bit mode).

- 3-1 Flip all bits satisfying $\Delta_k^{(GD)} < \theta \quad (k \in [1,n])$. Reevaluate the value of the objective function, and let $f_2 := f(\boldsymbol{x})$. If $f_1 > f_2$ holds, then let $\mu = 1$.
- 3-2 Flip a single bit at the jth position where $j \overset{\triangle}{=} \arg\min_{k \in [1,n]} \Delta_k^{(GD)}$.

IV. Behavior of the GDBF Algorithms

In this section, the behavior and decoding performance of (single and multi) GDBF algorithms obtained from computer simulations are presented. The objective function (3) was used in these experiments.

Figure 2 presents objective function values $f(\boldsymbol{x})$ (3) as a function of the number of iterations in the single and multi GDBF processes. Throughout the present letter, a regular LDPC code with $n = 1008$ and $m = 504$ (called PEGReg504x1008 in [6]) is assumed. The column weight of the code is 3. In both (single and multi) cases, we tested the same noise, and both algorithms output the correct codeword (i.e., the decoding was successful).

In the case of the single GDBF-algorithm, the value of the objective function gradually increases as the number of iterations grows increases in the first 50–60 iterations. Eventually, the value of the objective function is no longer incremented, and a flat section of the curve that corresponds to a local maximum appears. The curve corresponding to the multi GDBF algorithm exhibits much faster convergence as compared to the single GDBF algorithm. The multi GDBF algorithm requires only 15 iterations for the search point to closely approach the local maximum point.

Figure 3 presents the bit error curves of single and multi GDBF algorithms ($L_{max} = 100, \theta = -0.6$). As references, the curves for the WBF algorithm ($L_{max} = 100$), the MWBF algorithms ($L_{max} = 100, \alpha = 0.2$), and the normalized min-sum algorithm ($L_{max} = 5$, scale factor 0.8) are included as well. The parameter L_{max} denotes the maximum number of iterations for each algorithm.

The GDBF algorithms perform much better than the WBF and MWBF algorithms. For example, at BER $= 10^{-6}$, the multi GDBF algorithm has approximately 1.6 dB gain over

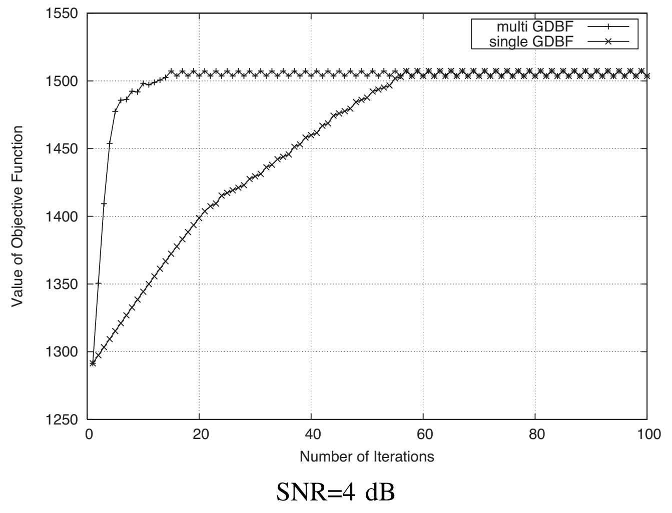


Fig. 2. Objective function values in GDBF processes as a function of the number of iterations (single GDBF v.s. multi GDBF) : regular LDPC code (PEGReg504x1008[6]).

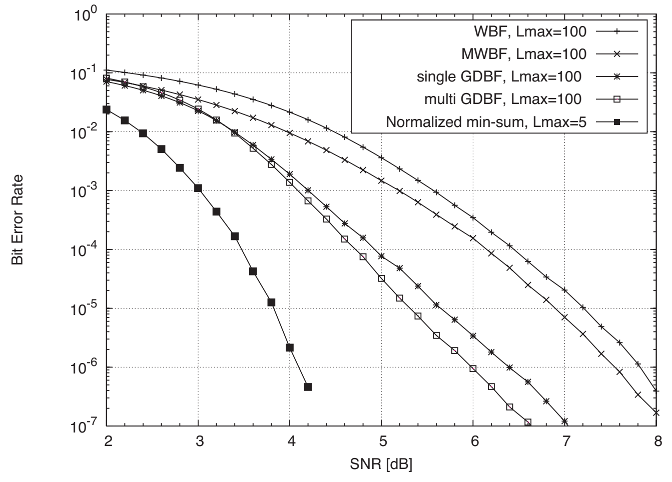


Fig. 3. Bit error rate of GDBF algorithms and other BF algorithms: regular LDPC code (PEGReg504x1008[6]).

the MWBF algorithm. Compared with the single GDBF algorithm, the error curve of the multi GDBF algorithm has a steeper slope. According to these results, the single GDBF algorithm appears to have no advantage, in either complexity or performance, over the multi GDBF algorithm.

Unfortunately, there is still a large performance gap between the error curves of the normalized min-sum algorithm and the GDBF algorithms. The GDBF algorithm fails to decode when a search point is attracted to an undesirable local maximum of the objective function. This large performance gap suggests the existence of some local maxima relatively close to a bipolar codeword, which degrades the BER performance.

V. Escape from a Local Maximum

A. GDBF algorithm with the escape process

A decoding failure occurs when a search point is captured by a local maximum, which is not a transmitted codeword. Since the weight of the final position of a search point is so small, a small perturbation of a captured search point appears to be helpful for the search point to escape from an undesirable local maximum. Such a perturbation process can be expected to improve the BER performance of BF algorithms.

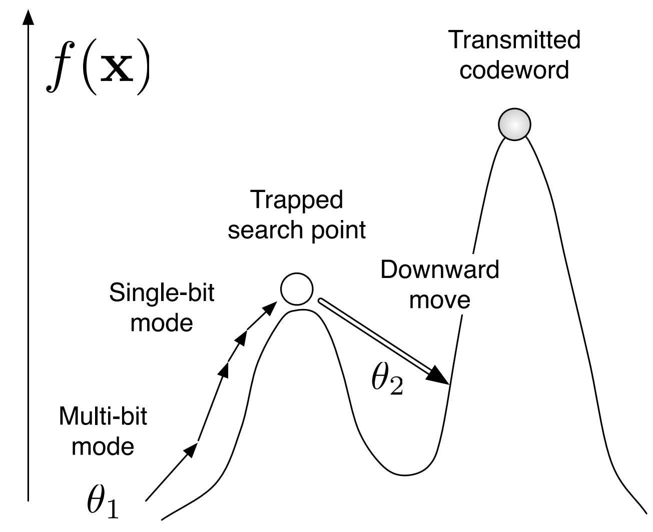


Fig. 4. Idea of escape process. A trapped search point can move to another hill by multi-bit flipping based on θ_2.

One of the simplest ways to add a perturbation to a trapped search point is to forcibly switch the flip mode from the single-bit mode to the multi-bit mode with an appropriate threshold when the search point arrives at a non-codeword local maximum. This additional process is referred to as the *escape process*. In general, the escape process reduces the object function value, i.e., the search point moves downward in the energy landscape. After the escape process, the search point again begins to climb a hill, which may be different from that of the trapped search point.

Here, we modify the multi GDBF algorithm by incorporating two thresholds: θ_1 and θ_2. The threshold θ_1 is the threshold constant used in the multi-bit mode at the beginning of the decoding process. After several iterations, the multi-bit mode is changed to the single-bit mode, and the search point may then eventually arrive at the non-codeword local maximum. In such a case, the decoder changes to the multi-bit mode (i.e., $\mu = 0$) with threshold θ_2. Thus, the threshold θ_2 can be regarded as the threshold for *downward movement*. Although θ_2 can be a constant value, in terms of the BER performance, it is advantageous to choose randomly[6]. In other words, θ_2 can be a random variable. After the downward move (only one iteration), the decoder changes the threshold back to θ_1. The above process continues until the parity check condition holds or the number of iterations becomes L_{max}. Figure 4 illustrates the concept of the escape process.

B. Simulation results

Figure 5 shows the BER curve for such a decoding algorithm (labeled 'Multi GDBF with escape'). In this simulation, we used the parameters $\theta_1 = -0.7$ and $\theta_2 = 1.7 + \alpha$, where α is a Gaussian random number with mean zero and variance

[6]This conclusion is based on experimental observations.

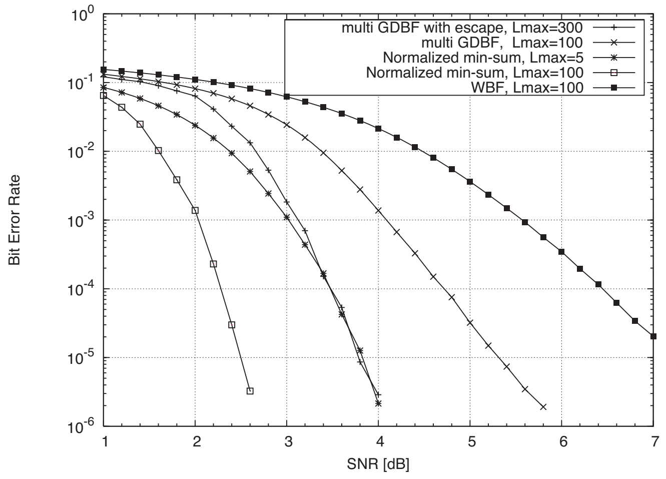


Fig. 5. Bit error rate of the GDBF algorithm with the escape process: regular LDPC code (PEGReg504x1008[6]).

0.01. Unfortunately, we have no quantitative theory predicting the effect of the escape process. The parameters θ_1 and θ_2 were chosen based on the results of preliminary experiments, i.e., these parameters were obtained by an ad hoc optimization at SNR = 4 dB.

The BER curve of the multi GDBF with escape (with $L_{max} = 300$) is much steeper than that of the naive multi GDBF algorithm. At a BER of 10^{-5}, the multi GDBF with escape gives approximately 1.5 dB gain over the naive multi GDBF algorithm. The average number of iterations of the multi GDBF with escape is approximately 25.6 at SNR = 4 dB. This result implies that the perturbation can actually reduce the number of trapped search points in order to converge to the desirable local maximum corresponding to the transmitted codeword. Optimization of the flipping schedule to narrow the performance gap between the min-sum algorithm and the GDBF algorithm remains an open problem.

VI. Conclusion

The present letter proposed a class of BF algorithms based on the gradient descent algorithm. The proposed algorithms show better decoding performance than known BF algorithms, such as the weighted BF algorithm and the modified weighted BF algorithm, for several LDPC codes.

The GDBF algorithms can be regarded as a maximization process of the object function using the bit-flipping gradient descent method (i.e., bit-flipping dynamics, which minimizes the energy $-f(\boldsymbol{x})$). The dynamics of the decoder is naturally derived by differentiating the energy function, and the gradient descent formulation introduces an energy landscape of the state-space of the BF-decoder. This formulation provides a new understanding of the convergence behaviors of BF algorithms.

Acknowledgement

The author would like to express his thanks to the anonymous reviewers of ISITA 2008 and IEEE Transactions on Communications for their comments, which helped to improve the quality of the manuscript. The present study was supported in part by the Ministry of Education, Science, Sports, and Culture of Japan through a Grant-in-Aid for Scientific Research on Priority Areas No. 180790091 and by a research grant from the Storage Research Consortium.

References

[1] R. G. Gallager, “Low-density parity-check codes," in Research Monograph Series. Cambridge, MA: MIT Press, 1963.
[2] F. Guo and H. Henzo, “Reliability ratio based weighted bit-flipping decoding for low-density parity-check codes," *IEEE Electron. Lett.*, vol. 40, no. 21, pp. 1356-1358, 2004.
[3] M. Jiang, C. Zhao, Z. Shi, and Y. Chen, “An improvement on the modified weighted bit flipping decoding algorithm for LDPC codes," *IEEE Commun. Lett.*, vol. 9, no. 9, pp. 814-816, 2005.
[4] Y. Kou, S. Lin, and M. P. C Fossorier, “Low-density parity-check codes based on finite geometries: a rediscovery and new results," *IEEE Trans. Inf. Theory*, pp. 2711–2736, vol. 47, Nov. 2001.
[5] C. H.Lee, and W.Wolf, “Implmentation-efficient reliability ratio based weighted bit-flipping decoding for LDPC codes," *IEEE Electron. Lett.*, vol. 41, no. 13, pp. 755-757, 2005.
[6] D. J. C. MacKay, “Encyclopedia of sparse graph codes." [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html
[7] P. O. Vontobel and R. Koetter, “Towards low-complexity linear programming decoding," in *Proc. 4th Int. Symp. Turbo Codes Related Topics*, Munich, Germany, Apr. 2006.
[8] J. Zhang, and M. P. C. Fossorier, “A modified weighted bit-flipping decoding of low-density parity-check codes," *IEEE Commun. Lett.*, pp. 165-167, vol. 8, Mar. 2004.