

Tabu-List Noisy Gradient Descent Bit Flipping Decoding of LDPC Codes

Li Zhang, Nan Liu, Zhiwen Pan, Xiaohu You

National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China

Email:{220170901, nanliu, pzw, xhyu}@seu.edu.cn

Abstract—The Noisy Gradient Descent Bit Flipping (NGDBF) algorithm surpasses the previous Gradient Descent Bit Flipping (GDBF) and other Bit Flipping (BF) algorithms for decoding Low-Density Parity-Check (LDPC) codes. However, the NGDBF decoder sometimes cannot escape from the local maxima. To further enhance the performance of the NGDBF algorithm, an improved NGDBF algorithm called Tabu-List Noisy Gradient Descent Bit Flipping (TNGDBF) is proposed in this paper. A tabu-list is employed to help the NGDBF decoder escape from the local maxima. It is shown in the simulation results that the TNGDBF algorithm gains about 0.2 dB performance compared with NGDBF at the cost of a small amount of complexity increase.

Index Terms—GDBF, bit flipping, LDPC, tabu-list, iterative decoding.

I. INTRODUCTION

A considerable research attention of Low-Density Parity-Check (LDPC) codes [1] has been gained in the past years. Various kinds of decoding algorithms for LDPC codes provide a good tradeoff between performance and complexity. The soft decision algorithms, referred to as Belief Propagation (BP) algorithm [1], Min-Sum (MS) algorithm [2], and Offset Min-Sum (OMS) algorithm [3], have a wonderful performance on the binary input additive white Gaussian noise (AWGN) channel, but at the cost of high computation complexity. Comparatively, the hard decision algorithms, referred to as Bit-Flipping (BF) algorithms, are easier for hardware implementation and have a lower complexity.

Many variants of BF algorithms which employ soft information have been investigated to improve the performance, such as Weighted Bit Flipping (WBF) algorithm [4] and Modified Weighted Bit Flipping (MWBF) algorithm [5]. Recently, the Gradient Descent Bit Flipping (GDBF) algorithms [6] are proposed. The GDBF algorithms develop a new inversion function derived from a gradient descent formula. The GDBF algorithms outperform the previous BF algorithms, but they tend to be trapped in local maxima and converge on an erroneous codeword. To overcome the drawback of GDBF algorithms, the Noisy Gradient Descent Bit Flipping (NGDBF) algorithms [7] are proposed to improve the performance. To make the decoder get out of the local maxima, every symbol metric employs a random perturbation at each iteration. Unfortunately, the NGDBF algorithms cannot always escape from the local maxima.

In [8], the type of gradient descent bit flipping algorithm is adapted from the AWGN channel to the binary symmetric channel (BSC). The Probabilistic Gradient Descent Bit Flipping (PGDBF) algorithm only uses hard information to decode. In [9], the Tabu-List Random-Penalty Gradient Descent Bit Flipping (TRGDBF) algorithm, which uses a random-penalty and a tabu-list to make the decoder escape from the trapping sets and accelerate convergence, is proposed for the BSC.

In this paper, we propose an improved NGDBF algorithm called Tabu-List Noisy Gradient Descent Bit Flipping (TNGDBF) algorithm on the AWGN channel. Inspired by [9], a tabu-list is used in each iteration to avoid repeated bit flipping between two successive iterations. Tabu-list and random perturbation work together to prevent the decoder from being trapped. Simulation results show tabu-list can help the TNGDBF decoder escape from the local maxima. We also provide the optimal values of noise scale parameter as a function of the SNR value. Only a small amount of computation complexity is needed to gain about 0.2 dB performance compared with the NGDBF algorithm.

II. TABU-LIST NOISY GRADIENT DESCENT BIT FLIPPING DECODING

A. System Model

Let $\mathbf{H}$ denote a binary parity check matrix with $m \times n$ dimensions and $n > m \geq 1$. We consider the set of LDPC codes $\mathcal{C}$ which is represented by $\mathcal{C} \triangleq \{\mathbf{c} \in \mathbb{F}_2^n : \mathbf{H}\mathbf{c} = \mathbf{0}\}$. In the present paper, we define the set of bipolar codes $\hat{\mathcal{C}} \triangleq \{(1 - 2c_1), (1 - 2c_2), \dots, (1 - 2c_n)\}$ corresponding to $\mathcal{C}$. Evidently, $\mathcal{C}$ is mapped to $\hat{\mathcal{C}}$ from binary $(0, 1)$ to bipolar $(+1, -1)$. Assume that the codes are transmitted over a binary input AWGN channel and the transmission is defined by $\mathbf{y} = \hat{\mathbf{c}} + \mathbf{z}$, where $\hat{\mathbf{c}} \in \hat{\mathcal{C}}$ and $\mathbf{z}$ is a white Gaussian noise vector. Each element of $\mathbf{z}$ is an independent and identically distributed Gaussian random variable with zero-mean and variance $N_0/2$, where N_0 is the noise spectral density.

The hard decision vector $\mathbf{x} \in \{+1, -1\}^n$ is obtained by $x_k = \text{sign}(y_k)$ for $k = 1, 2, \dots, n$. Let T be the maximum iteration number which is designed by the algorithm. The vector at the t -th iteration is represented by $\mathbf{x}(t)$, where $t \in [1, T]$. In each iteration, one or more bits may be flipped according to different algorithms. In this paper, we only consider the single BF algorithms [4], [5].

The parity check matrix $\mathbf{H}$ can also be defined by a Tanner graph with m Check Nodes (CNs) and n Variable Nodes (VNs). Let h_{ij} be the (i, j) th element of the parity check matrix $\mathbf{H}$, where $i \in [1, m]$ and $j \in [1, n]$. If $h_{ij} = 1$, the i -th CN is said to be linked with the j -th VN. Let $N(i) \triangleq \{j \in [1, n] : h_{ij} = 1\}$ for $i = 1, 2, \dots, m$ be the VNs linked with the i -th CN. The CNs linked with the j -th VN is represented similarly as $M(j) \triangleq \{i \in [1, m] : h_{ij} = 1\}$ for $j = 1, 2, \dots, n$. The parity check conditions of the codeword can be written as $s_i(t) \triangleq \prod_{j \in N(i)} x_j(t)$ for $i = 1, 2, \dots, m$, which is also called the bipolar syndrome. A codeword is said to be a legitimate codeword if and only if every syndrome component is equal to +1.

B. Noisy Gradient Descent Bit Flipping Algorithm

In [6], the GDBF algorithm is shown by regarding the maximum likelihood (ML) problem as the gradient descent optimization problem of an objective function. Inspired by the ML decoding problem, the objective function of the GDBF algorithm, which employs the syndrome component as a penalty term, is proposed. The objective function is defined by

$$f(\mathbf{x}) = \sum_{k=1}^n x_k y_k + \sum_{i=1}^m s_i. \quad (1)$$

Thus, the codeword to solve the ML problem is also the solution to maximize the objective function. The local inversion function is obtained by taking the partial derivative of a particular symbol x_k , and it is defined by

$$E_k = x_k \frac{\partial f(\mathbf{x})}{\partial x_k} = x_k y_k + \sum_{i \in M(k)} s_i. \quad (2)$$

To increase the objective function (1), the x_k which has the most negative E_k value is flipped. To improve the performance of the GDBF algorithm, a noisy type of GDBF called NGDBF is proposed in [7]. The inversion function of the NGDBF algorithm is defined by

$$E_k = x_k y_k + w \sum_{i \in M(k)} s_i + q_k, \quad (3)$$

where w is a parameter of syndrome weight and q_k is a Gaussian independent and identically distributed random variable which follows $N(0, \eta^2 N_0/2)$, where η is the noise scale parameter. The details of the NGDBF algorithm can be described in Algorithm 1.

In [7], a random perturbation is employed in the inversion function to assist the decoder in getting out of the local maxima in the NGDBF algorithm at the cost of a small amount of complexity increase. However, sometimes random perturbation causes the NGDBF decoder to oscillate around the local maxima, and the GDBF decoder also oscillates most of the time or rarely converges quickly to the transmitted codeword.

Algorithm 1 Single NGDBF (S-NGDBF) Algorithm [7]

- 1: Initialization: Set $T = 0$ and $E_k^0 = x_k y_k$ for $k = 1, 2, \dots, n$.
 - 2: Compute bipolar syndrome $s_i = \prod_{j \in N(i)} x_j$ for $i = 1, 2, \dots, m$. If $s_i = +1$ for all $i \in [1, m]$, output $\mathbf{x}$ and stop; Otherwise, $T = T + 1$.
 - 3: Calculate inversion function at each VN using (3).
 - 4: Find the bit position given by x_k for $k = \arg \min_{k \in \{1, 2, \dots, n\}} E_k$, and then flip the symbol: $x_k = -x_k$.
 - 5: If the number of iterations do not reach the maximum number of iterations T , repeat steps 2 to 4 till a valid codeword is detected; otherwise output $\mathbf{x}$ and exit.
-

C. Tabu-List Noisy Gradient Descent Bit Flipping Algorithm

To further improve the performance of NGDBF algorithm and keep the low complexity, we add a tabu-list to the NGDBF algorithm and propose a new algorithm called Tabu-List Noisy Gradient Descent Bit Flipping (TNGDBF) algorithm. The TNGDBF algorithm can be formally stated as follows:

Algorithm 2 Single TNGDBF (S-TNGDBF) Algorithm

- 1: Initialization: Set $T = 0$, $L^0 = 0$ and $E_k^0 = x_k y_k$ for $k = 1, 2, \dots, n$.
 - 2: Compute bipolar syndrome $s_i = \prod_{j \in N(i)} x_j$ for $i = 1, 2, \dots, m$. If $s_i = +1$ for all $i \in [1, m]$, output $\mathbf{x}$ and stop; Otherwise, $T = T + 1$.
 - 3: Calculate inversion function at each VN using (3).
 - 4: Find the bit position given by x_k for $k = \arg \min_{k \in \{1, 2, \dots, n\} \setminus L^{T-1}} E_k$, and then flip and record the symbol: $x_k = -x_k$, $L^T = k$, where L^T is used to record the flipped position in the T -th iteration.
 - 5: If the number of iterations do not reach the maximum number of iterations T , repeat steps 2 to 4 till a valid codeword is detected; otherwise output $\mathbf{x}$ and exit.
-

As discussed above, once the decoder is trapped in the local maxima, it will repeatedly flip some bits of the codeword in a fixed order and can hardly escape from the local maxima. The increase of the objective function is stopped by the local maxima and the decoder will oscillate. In the TNGDBF algorithm, the tabu-list is used to record the flipped bit position in each iteration and prevent the same bit from being flipped between any two adjacent iterations. This can increase the ability of the decoder to jump out of the local maxima with only a little increase in complexity.

As is shown in Step 3, we use the same inverse function as that in the NGDBF algorithm. The optimal noise scale parameter of the TNGDBF algorithm can be obtained by simulation. In Step 4, we add the bit which is flipped in the current iteration to the tabu-list and remove it in the next iteration. Tabu-list and random perturbation work together to help the decoder get out of the local maxima. The bit which

has the minimum E_k and is not in the tabu-list will be flipped in each iteration.

III. SIMULATION RESULTS

Simulation results are based on PEGReg504 $\times$ 1008 regular (3,6), $\frac{1}{2}$ -rate LDPC code and PEGirReg504 $\times$ 1008 irregular, $\frac{1}{2}$ -rate LDPC code from MacKay's online encyclopedia [10] on the AWGN channel with the Binary Phase Shift Keying (BPSK) modulation. The decoding performance of the TNGDBF algorithm is demonstrated and compared with the GDBF and NGDBF algorithms in this section. In order to demonstrate the ability of the tabu-list to help decoder escape from the local maxima, we also add tabu-list to the GDBF algorithm for comparison and call it TGDBF. As discussed above, the NGDBF algorithm have the ability to get out of the local maxima compared with the GDBF algorithm, and we find that the tabu-list can enhance this ability and accelerate the convergence to the transmitted codeword. The decoding parameters at different SNR values are empirically obtained according to different decoding methods and different codes. Unless otherwise stated, the NGDBF and the TNGDBF algorithms use determined values of $\omega = 0.75$ and $\eta = 1$. In order to further enhance the performance, we also study the optimal noise scale parameter of the TNGDBF algorithm on the PEGReg504 $\times$ 1008 LDPC code at some specific SNR values.

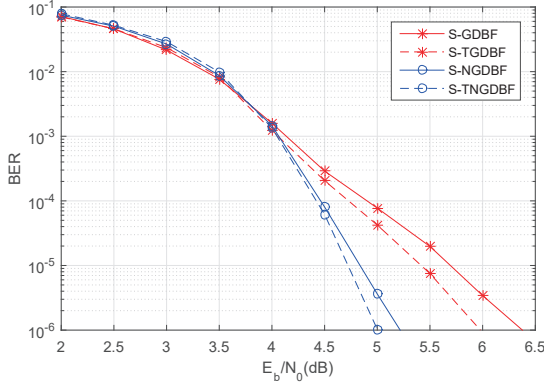


Fig. 1. BER performance for several decoding algorithms on the PEGReg504 $\times$ 1008, rate- $\frac{1}{2}$, regular LDPC code with 100 maximum iterations.

Fig. 1 presents the BER performance for several decoding algorithms on the PEGReg504 $\times$ 1008 LDPC code with 100 maximum iterations. The TGDBF algorithm outperforms by 0.4 dB compared with the GDBF algorithm at $\text{BER} = 10^{-6}$, which shows that tabu-list has the ability to help the GDBF decoder get out of the local maxima, but this ability is not as good as the random perturbation in the NGDBF algorithm. As a comparison, the TNGDBF algorithm outperforms by 0.2 dB compared with the NGDBF algorithm at $\text{BER} = 10^{-6}$, which shows that the random perturbation together with the tabu-list

can further enhance the ability of making the decoder escape from the local maxima.

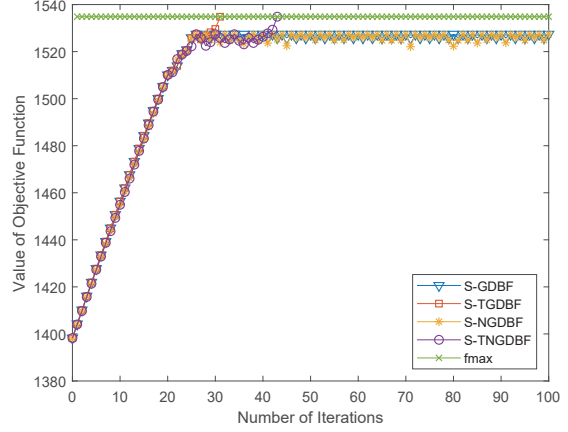


Fig. 2. The $f(\mathbf{x})$ value evaluated for several decoding algorithms as a function of iterations on the PEGReg504 $\times$ 1008, rate- $\frac{1}{2}$, regular LDPC code at $E_b/N_0 = 5.5$ dB.

Fig. 2 shows the value of $f(\mathbf{x})$ in (1) evaluated as a function of iterations at $E_b/N_0 = 5.5$ dB. All variants of the GDBF algorithms are designed to find the global maximum of the objective function $f(\mathbf{x})$ quickly and accurately. By flipping particular bit iteratively, every class of GDBF algorithm attempts to maximize the $f(\mathbf{x})$. Hence, we can observe the objective function value of each iteration for convergence analysis. The transmitted codeword is the ML solution for any ML-decodable problem, and the global maximum of objective function is obtained by introducing the transmitted codeword in (1). As is shown in Fig. 2, the value of objective function gradually increases with the number of iterations, and the global maximum is $f_{max} \approx 1534.84$. After 25 iterations, different algorithms enter different states. The rise of GDBF, which is indicated by the lower triangle line, stops and the $f(\mathbf{x})$ flattens out, and NGDBF, which is indicated by the asterisk line, is the same, but the fluctuation of NGDBF is greater because of the random perturbation. Obviously, the random perturbation tries to help NGDBF get out of the local maximum, but it fails in this example. Note that tabu-list makes TGDBF and TNGDBF escape from the local maximum effectively.

Fig. 3 shows the BER performance for NGDBF and T-NGDBF algorithms on the PEGReg504 $\times$ 1008 LDPC code with different maximum iterations. At $\text{BER} = 10^{-6}$, the TNGDBF algorithm outperforms the NGDBF algorithm by 0.2, 0.2 and 0.25 dB with 100, 200 and 300 iterations, respectively. Notably, the TNGDBF algorithm with 200 iterations has the same performance as the NGDBF algorithm with 300 iterations at $\text{BER} = 10^{-6}$, which means that the tabu-list can efficiently accelerate the convergence.

To further verify the performance of the proposed algorithm, PEGirReg504 $\times$ 1008 LDPC code is simulated. Fig. 4 presents

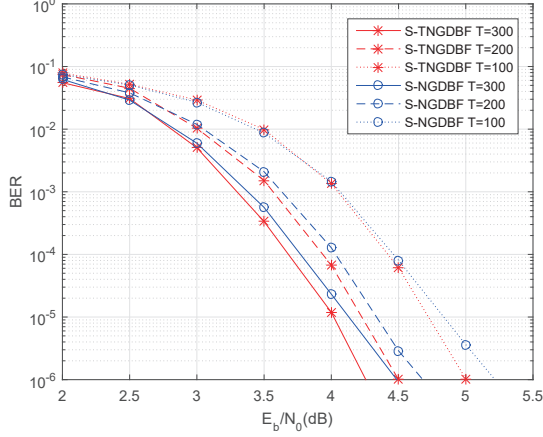


Fig. 3. BER performance for the NGDBF and TNGDBF algorithms on the PEGReg504 $\times$ 1008, rate- $\frac{1}{2}$, regular LDPC code with different maximum iteration numbers.

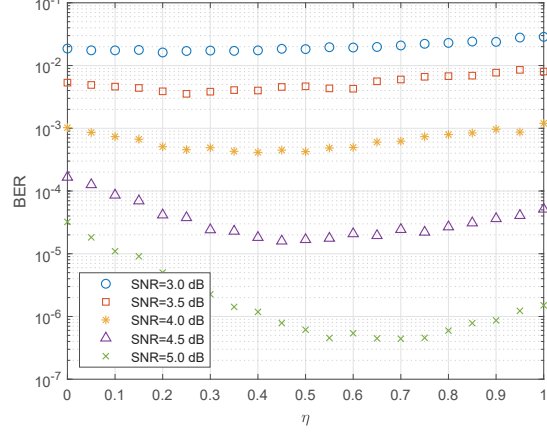


Fig. 5. BER performance for the TNGDBF algorithm on the PEGReg504 $\times$ 1008, rate- $\frac{1}{2}$, regular LDPC code with 100 maximum iterations at some specific SNR values relative to the parameter η .

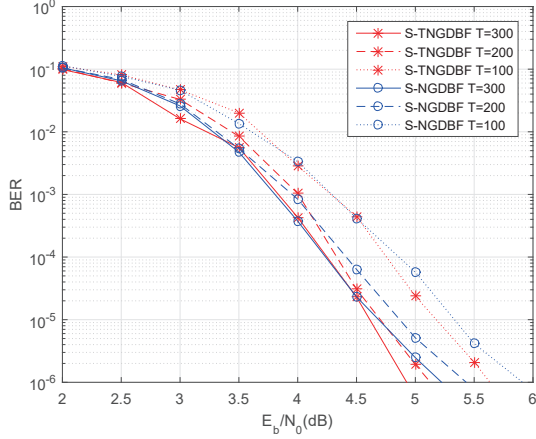


Fig. 4. BER performance for the NGDBF and TNGDBF algorithms on the PEGReg504 $\times$ 1008, rate- $\frac{1}{2}$, irregular LDPC code with different maximum iteration numbers.

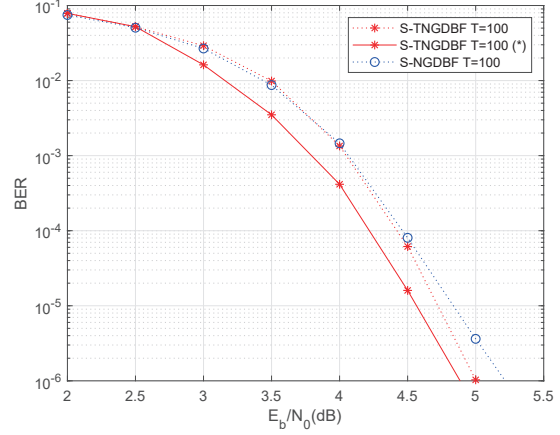


Fig. 6. BER performance for the NGDBF and TNGDBF algorithms on the PEGReg504 $\times$ 1008, rate- $\frac{1}{2}$, regular LDPC code with 100 maximum iterations. The algorithm with optimal η parameter values at some specific SNR values is indicated by an asterisk (*).

the BER performance on the PEGReg504 $\times$ 1008 LDPC code. At BER = 10^{-6} , the TNGDBF algorithm outperforms the NGDBF algorithm by 0.25, 0.35 and 0.35 dB with 100, 200 and 300 iterations, respectively. The gain on PEGReg504 $\times$ 1008 LDPC code is slightly greater than that on PEGReg504 $\times$ 1008 LDPC code.

We also study the optimal noise scale parameter of the TNGDBF algorithm on the PEGReg504 $\times$ 1008 LDPC code with 100 maximum iterations at some specific SNR values. Fig. 5 presents the performance of the TNGDBF algorithm. The TNGDBF algorithm with $\eta = 0.65$ has significantly better performance than that with $\eta = 1$ at 5 dB. The optimal parameter η is obtained empirically by simulation. Moreover, the optimal η values increase with the increase of SNR value, and gradually approach one.

Fig. 6 shows the BER performance for the NGDBF and TNGDBF algorithms on the PEGReg504 $\times$ 1008 LDPC code with 100 maximum iterations. The TNGDBF algorithm with optimal parameter η values is indicated by an asterisk (*). At 2 dB and 2.5 dB, we set the optimal parameter η of the TNGDBF algorithm to 1, because different parameters don't have an obvious gain on performance. The best performance of the TNGDBF algorithm is obtained with $\eta = 0.2$ at 3 dB, $\eta = 0.25$ at 3.5 dB, $\eta = 0.4$ at 4 dB, $\eta = 0.45$ at 4.5 dB and $\eta = 0.65$ at 5 dB. As is shown in Fig. 6, the TNGDBF algorithm with an optimal parameter η has 0.3 dB gain over the NGDBF algorithm and 0.1 dB gain over the TNGDBF algorithm with fixed $\eta = 1$ at BER = 10^{-6} .

IV. CONCLUSION

In this paper, an improved iterative decoding algorithm, denoted as TNGDBF, is proposed based on the NGDBF algorithm. In TNGDBF, a tabu-list is introduced to help the NGDBF decoder escape from the local maxima. With a small amount of complexity increase, the proposed TNGDBF algorithm has about 0.2 dB performance gains compared to the NGDBF algorithm. We also study the relationship between the noise scale parameter and performance. While we only consider the single GDBF algorithms in this paper, the multiple GDBF algorithms [6], [7] are worth looking into as part of the future work.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 61571123, in part by the National Key Research and Development Project 2018YF-B1802400, and in part by the Research Fund of National Mobile Communications Research Laboratory, Southeast University, under Grant No. 2019A03.

REFERENCES

- [1] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, January 1962.
- [2] Jinghu Chen and M. P. C. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," *IEEE Transactions on Communications*, vol. 50, no. 3, pp. 406–414, March 2002.
- [3] Jinghu Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and Xiao-Yu Hu, "Reduced-complexity decoding of ldpc codes," *IEEE Transactions on Communications*, vol. 53, no. 8, pp. 1288–1299, Aug 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 Transactions on Information Theory*, vol. 47, no. 7, pp. 2711–2736, Nov 2001.
- [5] Juntan Zhang and M. P. C. Fossorier, "A modified weighted bit-flipping decoding of low-density parity-check codes," *IEEE Communications Letters*, vol. 8, no. 3, pp. 165–167, March 2004.
- [6] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, "Gradient descent bit flipping algorithms for decoding ldpc codes," *IEEE Transactions on Communications*, vol. 58, no. 6, pp. 1610–1614, June 2010.
- [7] G. Sundararajan, C. Winstead, and E. Boutillon, "Noisy gradient descent bit-flip decoding for ldpc codes," *IEEE Transactions on Communications*, vol. 62, no. 10, pp. 3385–3400, Oct 2014.
- [8] O. A. Rasheed, P. Ivaniš, and B. Vasić, "Fault-tolerant probabilistic gradient-descent bit flipping decoder," *IEEE Communications Letters*, vol. 18, no. 9, pp. 1487–1490, Sep. 2014.
- [9] H. Cui, J. Lin, and Z. Wang, "An improved gradient descent bit-flipping decoder for ldpc codes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–13, 2019.
- [10] Mackay, "Encyclopedia of sparse graph codes," <http://www.inference.org.uk/mackay/codes/data.html>, accessed 20, 2014.