On Decoding of Low-Density Parity-Check Codes Over the Binary Erasure Channel

Hossein Pishro-Nik, *Student Member, IEEE,* and Faramarz Fekri, *Senior Member, IEEE*

***Abstract*—This paper investigates decoding of low-density parity-check (LDPC) codes over the binary erasure channel (BEC). We study the iterative and maximum-likelihood (ML) decoding of LDPC codes on this channel. We derive bounds on the ML decoding of LDPC codes on the BEC. We then present an improved decoding algorithm. The proposed algorithm has almost the same complexity as the standard iterative decoding. However, it has better performance. Simulations show that we can decrease the error rate by several orders of magnitude using the proposed algorithm. We also provide some graph-theoretic properties of different decoding algorithms of LDPC codes over the BEC which we think are useful to better understand the LDPC decoding methods, in particular, for finite-length codes.**

***Index Terms*—Bipartite graphs, erasure channel, improved decoding, iterative decoding, low-density parity-check (LDPC) codes, maximum-likelihood (ML) decoding, performance bound.**

I. Introduction

In this paper we study decoding of low-density parity-check (LDPC) codes when they are used over the binary erasure channel (BEC). The application of LDPC codes over BEC has been studied extensively [1]–[5]. When the message-passing algorithm is applied to an LDPC code over the BEC, it results in a very fast decoding algorithm [1]. However, the performance of this decoder is inferior to that of the maximum-likelihood (ML) decoder. Here, we derive some bounds on the performance of the ML decoder over the BEC. Then, we propose a technique to improve the performance of the message-passing decoder while keeping the speed of the decoding fast.

Asymptotic analysis of the performance of LDPC codes has been done successfully [6], [1], [7]. Capacity-achieving degree distributions for the binary erasure channel have been introduced in [1], [4], [3], and [5]. Although using the asymptotic analysis we can find good degree distributions, generating good finite-length LDPC codes has always been a challenge. Finite-length analysis of LDPC codes over the BEC was accomplished in [2]. In that paper, authors also proposed to use finite-length analysis in order to find good finite-length codes for the BEC. Here, we take a different approach. Instead of trying to find good LDPC codes we improve the decoding of the existing codes. The combination of the optimized codes using the finite-length analysis and the improved decoding algorithm that we present in this paper can result in good coding schemes over the BEC. Although the method we propose can be applied to any code length, its impact is more important for finite-length codes because for large values of code lengths there exist codes that achieve the capacity of the BEC. Thus, here we concentrate more on the moderate-length and short-length codes. More specifically, we consider the lengths that are less than or equal to 10^4. It is worth noting that the algorithm we present here for the BEC can be generalized for other memoryless binary-input output-symmetric (MBIOS) channels [8].

Throughout the paper, we assume the following terminology. By a graph we mean a simple graph, i.e., a graph with no loops (edges joining a vertex to itself) and no multiple edges (several edges joining the same two vertices). However, a multigraph may have loops or multiple edges. Let A be a subset of the vertices in the graph g. $N(A)$ shows the set of neighbors of A in g. Let D be a subgraph of g such that its vertex set is A. We say D is induced by A if D contains all edges of g that join two vertices in A. Let $e = vw$ be an edge in the graph g. When we say we contract e we mean that we identify the vertices v and w and remove all the resulting loops. Note that unlike the usual definition of contraction, we do not remove the duplicate edges resulting from identifying v and w. Let D be a subgraph of g. If we contract all the edges in D, we say that we have contracted D into a vertex. Let G be a bipartite multigraph with bipartition $V(G)$ and $C(G)$, For any $A \subseteq V \cup C$, we define $I_g(A)$ as the graph induced by the vertices in A and their neighbors. For a set E, 2^E is the set of all subsets of E.

II. Bounds on the Performance of ML Decoding

ML decoding has the best possible bit-error rate (BER). Since we are concerned with improving the iterative decoding of LDPC codes, ML decoding gives us the best possible improvement we may get. Thus, it is useful to study the ML decoder and its properties. Some properties of ML decoding of LDPC codes have been studied before, see, for examples, [9]–[11] and [2]. We first consider the asymptotic capacity of LDPC codes over the BEC under ML decoding. LDPC codes can be defined by their Tanner graphs [12]. Consider the ensemble $g(\lambda, \rho)$ of bipartite graphs defined by their degree distributions. We define the ML capacity (threshold) $\epsilon^\star$ of the ensemble as the supremum value of the parameter ϵ such that a randomly chosen code from the ensemble can achieve an arbitrarily small BER for sufficiently large n almost surely over a BEC with erasure probability ϵ. There is no concentration result known for ML decoding of LDPC codes over general MBIOS channels, however, the ML capacity is well defined

Manuscript received January 23, 2003; revised May 30, 2003. This work was supported in part by the Air Force Office of Scientific Research under Grant F49620-02-1-0053, the National Science Foundation under Grant ANI-0117840, and by the State of Georgia through Yamacraw.

The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA (email: hossein@ece.gatech.edu; fekri@ece.gatech.edu).

Communicated by S. Litsyn, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2004.824918

and is always greater than or equal to the threshold of the iterative decoding. With the above definition, we can find simple lower and upper bounds on the ML capacity of an ensemble. Since these upper and lower bounds are very close to each other (they are practically the same at least for regular codes), they provide an estimate of the ML capacity of the codes. A simple lower bound can be found using the union bound given in [2] and the asymptotic distance distributions of the regular LDPC codes derived in [13]. Note that in [14], the authors have independently derived the lower bound in the context of the error exponent of ML decoding. Let $g(d_v, d_c)$ be the ensemble of the regular LDPC codes with variable nodes and check nodes of degrees d_v and d_c, respectively.

Theorem 1: Let $t(d_c, \theta)$ be the only positive root of

$$\frac{(1+t)^{d_c-1}+(1-t)^{d_c-1}}{(1+t)^{d_c}+(1-t)^{d_c}} = 1-\theta. \tag{1}$$

Define $p(d_v, d_c, \theta)$ as

$$p(d_v,d_c,\theta)= \begin{cases} \frac{d_v}{d_c}\ln\left(\frac{(1+t)^{d_c}+(1-t)^{d_c}}{2t^{\theta d_c}}\right)-d_vH(\theta), & \text{if } d_c \text{ is even or } \theta\in(0,\frac{d_c-1}{d_c}) \\ -\infty, & \text{otherwise.} \end{cases} \tag{2}$$

Then the ML capacity of the ensemble $g(d_v, d_c), d_v > 2$, is lower-bounded by the supremum value of ϵ such that

$$\epsilon H\left(\frac{\theta}{\epsilon}\right)+p(d_v,d_c,\theta)<0, \qquad \text{for all } \theta\in[0,\epsilon].$$

Proof: Let h_i be the ith column of an $m \times n$ parity-check matrix H. Suppose a codeword is transmitted over a BEC channel with the erasure probability ϵ and let ε denote the set of variable nodes that correspond to the erased bits. Further, assume that $\epsilon < \epsilon_l^\star$, where $\epsilon_l^\star$ is the lower bound for the ML capacity given by the theorem. Let B_η be the event that $|\frac{|\varepsilon|}{n}-\epsilon|<\eta$. We have $\mathrm{pr}(B_\eta)=1-o(1)$ for all $\eta>0$. The ML decoder can decode the received word correctly if the columns of H that corresponds to the erased bits are independent. Let I be the set of indexes of the erased bits. We define

$$A_l=\{(h_{i_1},h_{i_2},\ldots,h_{i_l}): h_{i_1}\oplus h_{i_2}\cdots\oplus h_{i_l}=0, \quad i_1<i_2<\cdots<i_l, i_j\in I, j=1,2,\ldots,l\} \tag{3}$$

for $l = 1, 2, \ldots, n$. Let X_l be a random variable defined by $X_l = |A_l|$, where $|.|$ shows the cardinality of a set. The typical minimum distance of LDPC codes from $g(d_v, d_c)$ with $d_v > 2$ increases linearly with the code length. More specifically, for the ensemble $g(d_v, d_c)$ with $d_v > 2$ there exists $\delta(d_v, d_c) > 0$ such that the probability that the minimum distance of a randomly chosen code from the ensemble is less than or equal to $n\delta(d_v, d_c)$ converges to zero as $n \to \infty$ [15], [16]. Thus, we conclude that

$$\mathrm{pr}\left[\exists l\in\{1,2,\ldots,n\delta(d_v,d_c)\}, X_l>0\right]=o(1). \tag{4}$$

Let p_l be the probability that l randomly chosen columns of H sum to zero. Therefore,

$$E(X_l\,|\,|\varepsilon|=e)=\binom{e}{l}p_l. \tag{5}$$

As it is shown in [13]

$$\lim_{n\to\infty}\frac{1}{n}\ln p_{\theta n}=p(d_v,d_c,\theta).$$

Therefore,

$$\lim_{n\to\infty}\frac{1}{n}\ln E(X_{\theta n}|B_\eta)$$

approaches $\epsilon H(\frac{\theta}{\epsilon})+p(d_v,d_c,\theta)$ as η goes to zero. Let

$$c_s=\sup_{\delta(d_v,d_c)\le\theta\le\epsilon+\eta}\left(\epsilon H\left(\frac{\theta}{\epsilon}\right)+p(d_v,d_c,\theta)\right).$$

Using $\epsilon<\epsilon_l^\star$, it is easy to show that for sufficiently small η we have $c_s < 0$. Thus,

$$E(X_{\theta n}|B_\eta)=O(e^{-n|\frac{c_s}{2}|}). \tag{6}$$

Now define $X=\sum_{l=n\delta(d_v,d_c)}^{n(\epsilon+\eta)}X_l$. From the preceding discussion we have

$$E(X|B\eta)\sum_{l=n\delta(d_v,d_c)}^{n(\epsilon+\eta)}EX_l=O(n\epsilon e^{-n|\frac{c_s}{2}|})\to 0 \text{ as } n\to\infty. \tag{7}$$

Since $X \in \{0, 1, \ldots\}$, using $E(X|B_\eta) \to 0$ and the Markov inequality we conclude that $\mathrm{Pr}(X = 0|B_\eta) = 1 - o(1)$. Since

$$\mathrm{pr}(X=0)=\mathrm{pr}(X=0|B_\eta).\mathrm{pr}(B_\eta)+\mathrm{pr}(X=0|B_\eta^c).\mathrm{pr}(B_\eta^c)$$

we obtain $\mathrm{pr}(X = 0) = 1 - \mathrm{o}(1)$. Combining this with (4) we conclude that the ML decoder can decode the received word successfully almost always. □

Let $\epsilon_l^\star$ be the lower bound for the ML capacity given by Theorem 1. As an example, $\epsilon_l^\star$ of the $g(3, 6)$ ensemble is equal to $\epsilon_l^\star = 0.483$ while the capacity under message passing (the threshold found using density evolution) is $\epsilon_0 = 0.429$. Since we always use LDPC codes below their threshold, we conclude that for sufficiently large code lengths, the ML decoder is likely to decode the received word even though the message-passing decoder fails.

Using the distance distributions of the irregular codes, it is possible to generalize the preceding argument for the irregular codes. The distance distributions of irregular codes have been found by several authors [17], [18], and [14]. As shown in [16], if $\lambda_2\rho'(1) < 1$ the minimum distance of the expurgated ensemble increases linearly with the code length. Thus, we find the following bound for the expurgated ensemble.

Theorem 2: Consider the ensemble $g(\lambda, \rho)$ that satisfies $\lambda_2\rho'(1) < 1$.[1] Let $b_\theta = b_\theta(\lambda, \rho)$ be the average distance

[1]As commented by one of the reviewers, the condition $\lambda_2\rho'(1) < 1$ can be relaxed.

distribution as defined in [13]. Then the ML capacity of the expurgated ensemble $g(\lambda,\rho)$ is lower-bounded by the supremum value of ϵ such that

$$\epsilon H\left(\frac{\theta}{\epsilon}\right) - H(\theta) + b_\theta(\lambda,\rho) < 0$$

for all $\theta \in [0,\epsilon]$.

It is useful to find an upper bound for the ML capacity. The bound would obviously be an upper bound for the iterative decoder as well. First, using Markov's inequality, the following lemma can be easily obtained.

Lemma 1: Let $\theta > 0$ be a constant. Let also N be a positive integer such that for all $n > N$, the sequence of random variables Z_n, where $0 \le Z_n \le n$ satisfies $E(Z_n) \le (\theta + o(1))n$. Then, for all $\alpha > 0$ there exist N' and $\delta > 0$ such that for all $n > N'$ we have $\mathrm{pr}\{Z_n < (\theta+\alpha)n\} > \delta$.

Let ξ_i be the fraction of variable nodes of degree i and φ_i be the fraction of check nodes of degree i. Let us define

$$\Psi(x) = 1 - \sum_i \xi_i (1-x)^{i-1}(1+(i-1)x)$$

and

$$\Phi(x) = \sum_i \varphi_i x^i.$$

By a simple observation, we can find the following upper bound for the capacity of LDPC codes over the BEC.

Theorem 3: To have arbitrarily small bit error probability under ML decoding on a BEC with erasure probability ϵ, we must have

$$1 - R \ge \frac{\epsilon[1+\Psi((1-\epsilon)^{d_{c\max}-1})]}{1-\Phi(1-\epsilon)}. \tag{8}$$

Proof: Let us construct a matrix H' from the parity matrix H by selecting each column of H with probability $(1-\epsilon)$ independently and replacing it with the zero vector. The nonzero columns of H' correspond to the erased bits. If we pick a row from H at random, this row is equal to zero in H' with probability $\Phi(1-\epsilon)$. Therefore, on the average, we have $m\Phi(1-\epsilon)$ zero rows in H'. We now prove the following lemma.

Lemma 2: If we peak a random integer j between 1 and n, then with probability at least $\epsilon\Psi((1-\epsilon)^{d_{c\max}-1}) - o(1)$ we have at least two rows in H' whose jth element is 1 and all their other elements are 0.

Proof: Call the required probability p_j. For clarity of exposition, consider the regular ensemble $g(d_v,d_c)$. Let F_j be the event that the jth column in H' be nonzero, thus, $\mathrm{pr}\{F_j\} = \epsilon$. There are d_v rows $i_1, i_2, \ldots, i_{d_v}$ in H whose jth element is one. Let I_j be the set consisting of these rows. Thus, we have $|I_j| = d_v$. Let EV_{i_k} be the event that only the j_kth element of the ith row in H' is equal to 1. Since $d_v < \infty$, with high probability the positions of the 1's in all of the rows in I_j do not overlap except for the jth position. Therefore, given that the jth column is preserved in H', the events EV_{i_k} are independent. Since the probability of any event is less than or equal to one, to obtain p_j it suffices to consider only the case when EV_{i_k}'s are independent and add an $o(1)$ to the result. Now, obviously, we have $\mathrm{pr}\{EV_{i_k}|F_j\} = (1-\epsilon)^{(d_c-1)}$. Considering the above discussion it is easy to show that

$$p_j = \epsilon\Big\{1 - \{[1-(1-\epsilon)^{d_c-1}]^{d_v} + d_v[1-(1-\epsilon)^{d_c-1}]^{d_v-1}(1-\epsilon)^{d_c-1}\} - o(1)\Big\}. \tag{9}$$

For irregular codes using the inequality

$$\prod_{i=1}^{n}(1-x_i) + \sum_{i=1}^{n} x_i \prod_{j\ne i}(1-x_i) \le (1-x_1)^n + nx_1(1-x_1)^{n-1} \tag{10}$$

for $0 \le x_1 \le x_2 \le \cdots x_n \le 1$, we obtain

$$p_j \ge \epsilon\Psi((1-\epsilon)^{d_{c\max}-1}) - o(1). \tag{11}$$

□

Thus, we showed that if we pick a random integer i between 1 and n, with probability at least $\epsilon\Psi((1-\epsilon)^{d_{c\max}-1}) - o(1)$ we have at least two rows in H' whose ith element is 1 and all the other elements are 0. Any such i reduces the rank of H' by at least one. Therefore, if we define $Z_n := \mathrm{rank}(H')$, we have

$$\begin{aligned} E(Z_n) &= E(\mathrm{rank}(H')) \le m - m\Phi(1-\epsilon) \\ &\quad - n\epsilon\Psi((1-\epsilon)^{d_{c\max}-1}) + o(1)n \\ &= n\{(1-R)[1-\Phi(1-\epsilon)] \\ &\quad - \epsilon\Psi((1-\epsilon)^{d_{c\max}-1}) + o(1)\}. \end{aligned} \tag{12}$$

By defining

$$\theta := (1-R)[1-\Phi(1-\epsilon)] - \epsilon\Psi((1-\epsilon)^{d_{c\max}-1}) \tag{13}$$

we have

$$E(Z_n) \le (\theta + o(1))n. \tag{14}$$

Now we show that for an arbitrarily small error probability we must have $\epsilon \le \theta$. Suppose $\epsilon > \theta$. As previously shown

$$E(Z_n) = E(\mathrm{rank}(H')) \le (\theta + o(1))n.$$

Let $0 < \kappa < \epsilon - \theta$ be a constant. Then, by Lemma 1, there exist N' and $\delta > 0$ such that for all $n > N'$ we have $\mathrm{pr}\{Z_n < (\theta+\kappa)n\} > \delta$. Therefore, with a strictly positive probability that is independent of n we have $\mathrm{rank}(H') < (\theta+\kappa)n$. Hence, the decoder can find the value of at most $(\theta+\kappa)n$ erasures. Consequently, at least $(\epsilon-\theta-\kappa)n$ erasures remain after the decoding. This implies that the overall error probability of the decoder is at least $(\epsilon-\theta-\kappa)\delta$.Therefore, for reliable communication we must have $\epsilon \le \theta$. Thus,

$$(1-R)[1-\Phi(1-\epsilon)] - \epsilon\Psi((1-\epsilon)^{d_{c\max}-1}) \ge \epsilon. \tag{15}$$

This completes the proof of the theorem. □

In the preceding argument if we just consider the rows that are zero in H' and omit the discussion about the rows with weight one, we will get a slightly weaker bound as

$$1 - R \ge \frac{\epsilon}{1-\Phi(1-\epsilon)}. \tag{16}$$

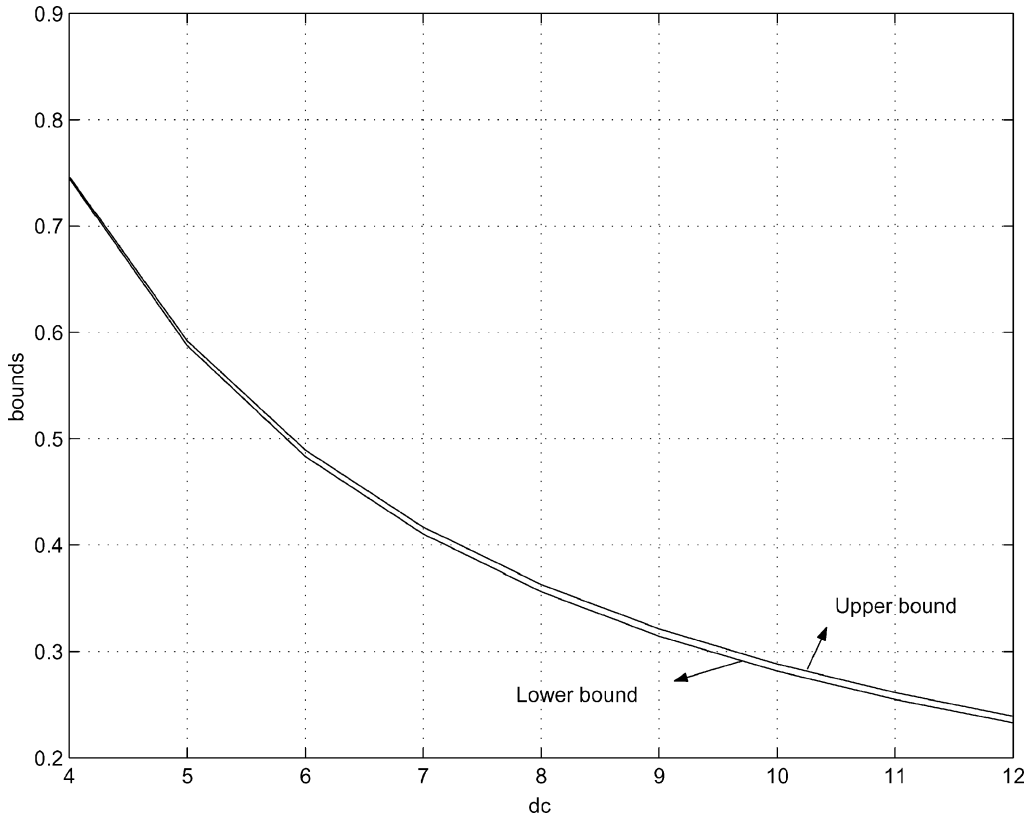

Fig. 1. Lower and upper bounds for the ML capacity of $g(3, d_c)$.

This is the same bound given in [4] for the iterative decoding of the LDPC codes. Note that in [4], the bound is derived for the iterative decoding but the preceding discussion shows that the bound is also valid for ML decoding. It is also noteworthy that examining the proof of Theorem 3, we find that the upper bound given by (16) is valid for any individual code in the ensemble while the one given in (8) is valid for typical codes. This is because we made use of the cycle-free neighborhood assumption. As an example, for the ensemble $g(3,6)$, Theorem 3 gives the upper bound $\epsilon = 0.489$ while (16) provides a slightly weaker bound $\epsilon = 0.491$. Furthermore, the authors in [19] [2] have also obtained the upper bound $\epsilon = 0.491$, that is, again slightly weaker than the one given by Theorem 3. However, the bound in [19] was proven for the individual codes.

Figs. 1 and 2 show the upper bound and the lower bound that are given by Theorems 1 and 3 for $g(3, d_c)$ and $g(4, d_c)$, respectively. As suggested by the figures, the two bounds are practically the same. As an example for irregular graphs, we consider the ensemble of LDPC codes defined by

$$\lambda(x) = 0.142696x + 0.562771x^2 + 0.294532x^{10}, \quad \rho(x) = x^6. \tag{17}$$

Using Theorems 2 and 3, we find that the ML capacity for the given ensemble satisfies $0.4899 \leq \epsilon \leq 0.4948$. It is worth noting that for the calculation of the lower bound we used [17] to find the weight spectrum of the code. Again we conclude that the bounds are sufficiently tight. Therefore, we can approximate the ML capacity from the given bounds. Note that Theorem 3 gives an upper bound that can be easily computed for any degree distributions. However, for the lower bound we need to have the distance distribution of the code.

Since we are interested in the finite-length LDPC codes, it is desirable to choose the codes that satisfy $\lambda_2\rho'(1) < 1$. This is because as shown in [16] and [20], if we have $\lambda_2\rho'(1) > 1$, then the minimum stopping set and the minimum distance will be sublinear with high probability and, therefore, we will have small stopping sets in the graph which is not desirable for finite-length codes. In the above definition, we defined the ML threshold for the bit erasure probability. It is mentioned in [2] that the threshold for bit erasure probability and block erasure probability may be different. Any upper bound for the threshold for bit erasure probability is an upper bound for the threshold for block erasure probability as well. Therefore, the upper bound in Theorem 3 is also an upper bound for the threshold for the block erasure probability. In the ensembles for which $\lambda_2\rho'(1) < 1$, the thresholds for the block erasure probability and the bit erasure probability are the same. This is because for these ensembles there is no codeword with weight less than or equal to $n\delta(d_v, d_c)$ with a high probability. Therefore, if the ML decoder cannot decode the received word, there will be at least $n\delta(d_v, d_c)$ erasures that are left after the decoding is performed. Thus, the BER is at least $\delta(d_v, d_c)$ times the block error rate. The same argument works for the iterative decoder if we replace the minimum distance by the size of the minimum stopping set. Consequently, the lower bounds in Theorems 1 and 2 are also valid for the threshold for block erasure probability. In fact, although we stated the theorem for the bit error probability threshold, in the proof we showed that the block error rate goes to zero.

[2]This paper appeared after the review of our manuscript in *IEEE Transactions on Information Theory*.

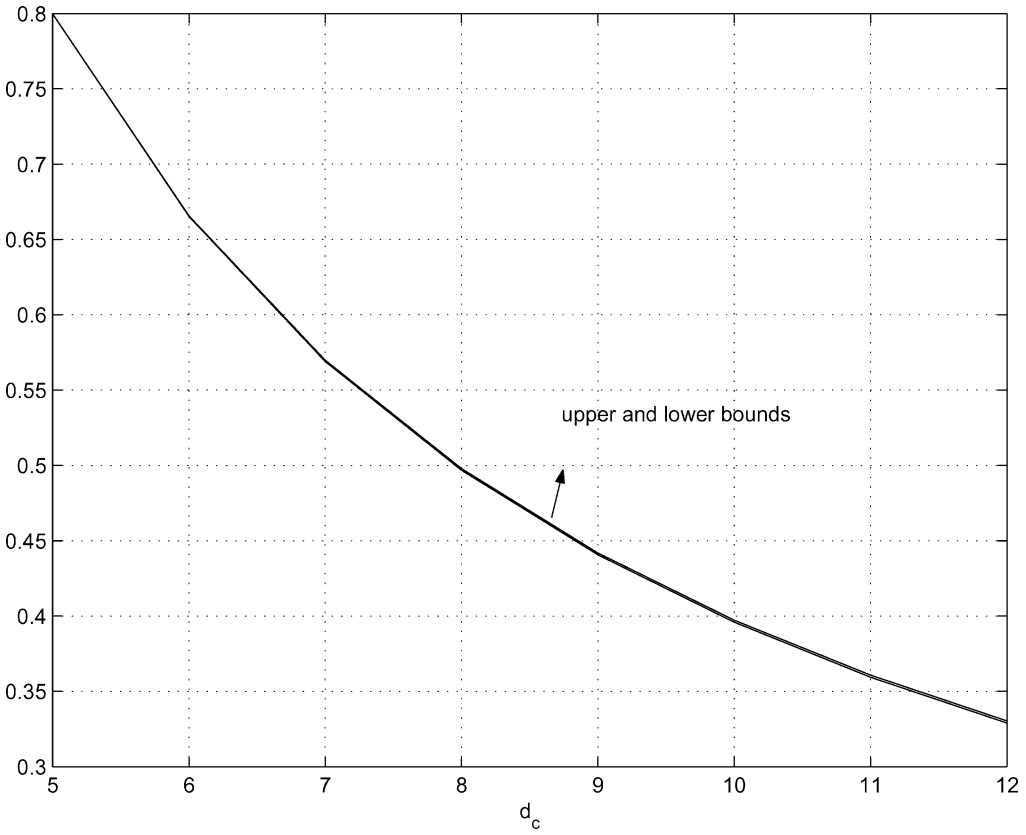

Fig. 2. Lower and upper bounds for the ML capacity of $g(4, d_c)$.

III. Improving the Iterative Decoding

A. Description of Algorithms

The iterative decoding of LDPC codes over the BEC is much faster than the ML decoding. However, it has higher error probability. Our aim in this section is to decrease the error probability while keeping the decoding fast. We mostly focus on moderate and short-length codes. We use the message-passing algorithm with some modifications. Let $V = V(g) = \{v_1, v_2, \ldots, v_n\}$ and $C = C(g) = \{c_1, c_2, \ldots, c_m\}$ be the set of variable and check nodes, respectively. A stopping set S is defined in [2] as a subset of V such that all neighbors of S are connected to S at least twice. Let ε be the subset of the set of variable nodes that is erased by the channel. It is proved in [2] that the iterative decoding fails if and only if ε contains a stopping set. In [2] it is also shown that the set of the remaining erasures when the decoder stops is equal to the unique maximal stopping set of ε.

Let $T_A(n)$ be the average time required for the standard iterative decoding of an LDPC code of length n when it is used over a BEC with the erasure probability ϵ. Let B be an improved decoding method for the same code when used over the same channel. Let $T_B(y, n)$ be the time that Algorithm B needs to decode a received word y and let $T_B(n)$ be the average time of the decoding of the code using Algorithm B. We want to have

$$T_B(n) \leq (1+\gamma) T_A(n) \tag{18}$$

$$\forall y, \;\; T_B(y, n) \leq C T_A(n) \tag{19}$$

where γ is a small constant close to zero and C is a sufficiently small constant. Our simulations show that the algorithm we propose in this section (Algorithm C) will achieve the above inequalities with $\gamma < 0.05$ and $C < 10$.

Theoretically, any LDPC code has a threshold ϵ_{th} such that if $\epsilon > \epsilon_{\text{th}}$ then the error probability of the standard iterative decoding is bounded away from zero by a strictly positive constant. On the other hand, if $\epsilon < \epsilon_{\text{th}}$, an arbitrarily small error probability is attainable if n, the length of the code, is large enough [6], [1]. However, for finite-length codes the situation is deferent. First, we may get an error floor and cannot decrease the error probability as we want. Moreover, to decrease the error probability, for example from 10^{-3} to 10^{-6}, we need to decrease ϵ by a considerable amount. Here we propose a method for decoding LDPC codes over BEC that has the same complexity as the message-passing decoder. However, its error rate is considerably smaller.

The key idea is the following observation. Consider a BEC with an erasure probability ϵ and an LDPC code of length n that has a small enough error probability. If the message-passing decoder fails to decode a received word completely, then there exists a few (usually less than or equal to 3 bits) undecoded bits that if their values are exposed to the decoder, then the decoder can finish the decoding successfully. Note that this is true only when the BER is small enough (for example, less than 10^{-2}). Simulations and intuitive arguments strongly confirm the above statement for different LDPC codes.

Let us recall the message passing decoding of LDPC codes over the BEC [1]. The algorithm can be stated as follows.

- For all unlabeled check nodes do the following. If the values of all but one of the variable nodes connected to the check

node are known, set the missing variable bit to the XOR of the other variable nodes and label that check node 'finished'. If all the variable nodes connected to the check node are known label the check node as finished.The procedure is done sequentially, i.e, one check node at a time.
- Continue the above procedure until all check nodes are labeled as finished or the decoding cannot continue further.

We want to improve the above algorithm. Let us call the above algorithm A and the first improved Algorithm B. For the erasure patterns that Algorithm A finishes the decoding successfully, both algorithms are the same. The difference between the two algorithms is when Algorithm A fails to complete the decoding of a received codeword. In this case, Algorithm B continues the decoding as following. It chooses one of the unknown variable nodes w_1 (we will discuss how to choose this variable node) and guesses its value (for example, by setting its value to zero). Now it continues as follows.

- For all unlabeled check nodes do the following: If the value of all but one of the variable nodes connected to the check node are known, set the missing variable bit to the XOR of the other variable nodes and label it as a finished check node. If all the variable nodes connected to the check node are known then if the check node is satisfied label that check node "finished," otherwise label it "contradicted." The procedure is done sequentially, i.e, one check node at a time.
- Continue the above procedure until all check nodes are labeled or the decoding cannot continue further.

Once the above procedure is finished, if all of the check nodes are labeled and none of them is labeled "contradicted," the decoder outputs the resulting word as the decoded word. If all of the check nodes are labeled but some of them are labeled "contradicted," then it changes the value of w_1, the guessed variable node, and repeats the decoding from there. This time the decoding finishes successfully because we have found the actual value of w_1. But if the decoding stops again (i.e., some of the check nodes are not labeled) we have to choose another unknown variable node w_2 and guess its value to continue the decoding. Again, if some check nodes are labeled as "contradicted," we have to go back and try other values for w_1 and w_2. Obviously, Algorithm B is efficient only if the number of guesses is very small. Fortunately, simulation results show that even if we limit the number of guesses to a very small number, we can decrease the error rate by a considerable amount. Thus, in practice we limit the number of guesses to a maximum value $g_{\max}$. If after $g_{\max}$ guesses the decoding does not finish, we claim a decoding failure. In fact, simulations show that with the right choices of the variable nodes to guess, usually the decoding finishes successfully by one or two guessed variable nodes. Note that Algorithm B does not need any extra computation other than the usual iterative decoding. Thus, the decoding is very fast.

Now let us consider the problem of choosing the variable nodes w_i's that we need to guess their values. One easy method is to choose them from the set of variable nodes with the highest degree. Note that a variable node of degree d is present in d equations. Therefore, when we assume that its value is known, any parity-check equation that has only one unknown variable node other than the guessed variable node will free a variable node. Therefore, intuitively we expect that guessing a high-degree variable node results in freeing more unknown variable nodes. However, we can still improve our method of choosing w_i's as follows. First, we choose a high-degree unknown variable node w_j and examine its neighborhood. If guessing w_j frees at least f unknown variable nodes for a suitable constant f, we accept w_j as one of our guesses. Otherwise, we choose another high-degree variable node. In our simulations we chose the value of f between $0.5d_{c_{\max}}$ and $d_{c\max}$.

Algorithm B has two problems. First, the complexity of the algorithm grows exponentially with the number of guesses. Although the number of guesses is very small, this is undesirable. In fact, if the complexity of the algorithm increased linearly with the number of guesses we could increase $g_{\max}$ and decrease the error probability substantially. Second, it is possible that the algorithm declares a wrong word as the output of the decoding. However, this can happen only if the ML decoder cannot decode the corresponding codeword. Since the ML decoder has a very low error probability, this happens with a very small probability. We now propose Algorithm C that copes with both problems.

Let w_1 be the first variable node that we guess. Let x_1 be the value of w_1. From now on, any variable node whose value is determined by the algorithm can be represented in one of the following forms: x_1, $\overline{x_1} = x_1 \oplus 1$, 1, or 0. In general, if the algorithm makes g guesses, any variable node that is determined after the first guess can be represented as

$$a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \cdots \oplus a_g x_g \tag{20}$$

where $a_j \in \{0, +1\}$. Therefore, any variable node that is determined after the first guess can be represented by $(a_0, a_1, \ldots a_g)$. After the first guess, Algorithm C continues as follows.

- For all unlabeled check nodes do the following: If the values of all but one of the variable nodes that are connected to the check node are known, compute the value of $(a_0, a_1, \ldots, a_g)$ for the missing bit and label that check node as "finished." If all the variable nodes that are connected to the check node are known then label that check node as a "basic equation." The procedure is done sequentially, i.e, one check node at a time.
- Continue the above procedure until all check nodes are labeled or the decoding cannot continue further.

If necessary, Algorithm C makes other guesses. If after the maximum possible number of guesses some of the variable nodes are unlabeled, then we claim decoding failure. Suppose that after $g \leq g_{\max}$ guesses all the check nodes are labeled. Now we have the following.

Lemma 3: The received word is ML decodable if and only if the set of basic equations have a unique solution.

Proof: By the labeling procedure, any choice for the values of $x_1, x_2, \ldots, x_g$ satisfy all the parity-check equations that are labeled "finished." Therefore, decoding is possible if and only if a unique choice of $x_1, x_2, \ldots, x_g$ satisfies all the basic equations. □

Therefore, after all the check nodes are labeled, we examine the set of basic equations. If they have a unique solution, we determine $x_1, x_2, \ldots, x_g$ and then we find the values of all the variable nodes. Otherwise, we claim decoding failure. Note that since g is a very small number, solving the basic equations is a very simple task and can be done quickly. In fact, it is easy to show that Algorithm C has complexity $O(g_{\max}^2 n)$. Sometimes, although the set of basic equations does not have a unique solution, they still determine a subset of $\{x_1, x_2, \ldots, x_g\}$ uniquely. In this case, we can replace the values of these variable nodes in the expressions for unknown variable nodes and, consequently, we may be able to recover some of the bits. This approach is specifically useful when we deal with a code that has error floor due to the small minimum distance. Note that the procedure of finding a variable node for guessing in Algorithm C is the same as Algorithm B. The following lemma determines the number of basic equations.

Lemma 4: Let ε be the subset of the set of variable nodes that are erased by the channel and S be the unique maximal stopping set in ε. Then the number of basic equations is equal to

$$N_B(S) = |N(S)| - |S| + g. \tag{21}$$

Proof: At the beginning of the guessing process there are $|N(S)|$ unlabeled check nodes. Any of these check nodes is either a basic check node or determines exactly one variable node. Since there are $|S| - g$ variable nodes that are determined by the check nodes, $|N(S)| - |S| + g$ check nodes are labeled as "basic equations." □

Note that Algorithm C is equivalent to the ML decoder if we do not limit the maximum number of guesses $g_{\max}$. In fact, in this case Algorithm C is just an efficient implementation of the ML decoder. It is worth noting that the set of basic equations depends on the choice of variable nodes we guess.

An alternative method is to perform ML decoding on the remaining erasures whenever iterative decoding fails. This decoding is much faster than the ordinary ML decoding and has exactly the same performance as ML decoding. However, it has two problems. First, it does not satisfy the requirement in (19), because generally ML decoding of LDPC codes over the BEC has time complexity $\Theta(n^3)$. Second, the ordinary ML decoding of LDPC codes requires $\Theta(n^2)$ space while Algorithms A, B, and C require $O(n)$ space.

B. Bounds on the Number of Guesses in Algorithms B and C

Here we study the required number of guesses by Algorithms B and C based on the properties of the Tanner graph of the code. Instead of providing asymptotic results, we focus on graph-theoretic results that we think are useful in finite-length analysis. To do that we need to extend the iterative decoding to bipartite multigraphs. Let G be a bipartite multigraph with bipartition $V(G)$ and $C(G)$, where $V(G) = \{v_1, v_2, \ldots, v_n\}$ and $C(G) = \{c_1, c_2, \ldots, c_m\}$ are the sets of variable and check nodes, respectively. Iterative decoding works on G as follows. At the beginning, all the erased variable nodes are labeled "unknown" and the following algorithm is repeated in each step.

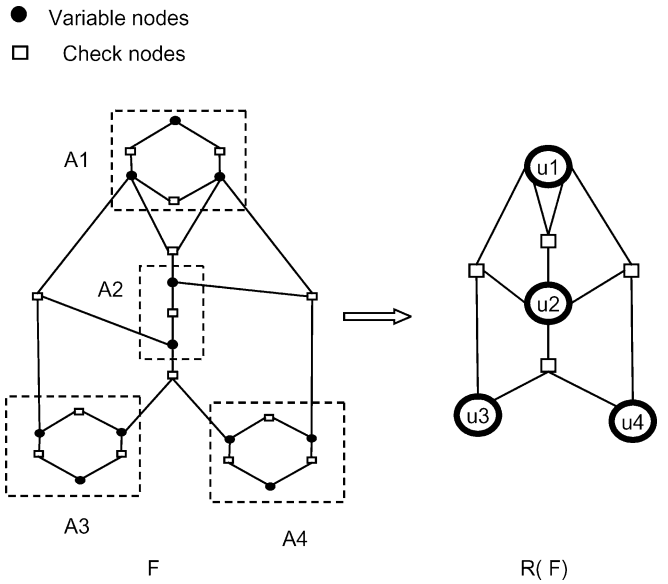

Fig. 3. Construction of $R(F)$.

- If only one of the edges that is connected to a check node is incident with an unknown variable node, label that variable node as "known."

Note that the preceding algorithm is not a decoding algorithm for a real code. We just define it to simplify our discussion. We need here the following definitions. We define a set $B \subset V(G)$ to be sufficient if by knowing the values of the variable nodes in B, the iterative decoder can finish the decoding successfully. A set $B \subset V(G)$ is called unnecessary if $V(G) - B$ is sufficient. In other words, $B \subset V(G)$ is unnecessary if the iterative decoder can determine the values of erased variable nodes, when the variable nodes in B are erased but all the other variable nodes are known. Obviously, a set B is unnecessary if it does not contain a stopping set.

Assume the iterative algorithm A fails to decode a received word on a graph g. Let S be the stoping set that remains after the decoding stops and let $F = I_g(S)$. We define an equivalence relation R on $V(F)$ in the following way. We write vRw if there is a path from v to w on F that does not contain any check node of a degree higher than two. Obviously, R is an equivalence relation. Thus, this relation partitions $V(F)$ into $p(F)$ equivalence classes. Let $A_1, A_2, \ldots, A_{p(F)}$ be the equivalence classes of R. Note that if the value of v is exposed to the decoder, then the decoder can find the values of all variable nodes in the equivalence class $[v]$ of v.

For a bipartite graph F we construct the graph $R(F)$ as follows. For each equivalence class A_i, we contract all the variable nodes in A_i and all the check nodes that do not have any neighbors outside A_i into one vertex u_i. Fig. 3 shows an example of this construction. Note that every check node in $R(F)$ has a degree at least three. Assume the iterative Algorithm A fails to decode a received word Y on a graph g. Let S be the stoping set that remains after the decoding fails and let $F = I_g(S)$. We have the following.

Theorem 4: Let Y be ML decodable and Z be the random variable that is equal to the number of guesses that are required

by Algorithm B to finish the decoding. Let also $M(F)$ be the minimum sufficient set in $R(F)$ and $p(F)$ be the number of equivalent classes in $v(F)$. Then, we have

$$|M(F)| \leq Z \leq p(F). \tag{22}$$

Moreover, the lower bound is always attainable by using the right choices of the guessed variable nodes.

Proof: After each guess in Algorithm B, all the variable nodes in at least one of the equivalence classes will be determined. Therefore, after $p(F)$ guesses the values of all the variable nodes are found. Let $M(F) = \{u_j : j \in I_F\}$ be the minimum cardinality sufficient set in $R(F)$. Suppose, in each step we choose a variable node in one of A_j's such that $j \in I_F$. Then after $|M(F)|$ steps all the variable nodes are determined. Obviously, the number of guesses cannot be less than $|M(F)|$. □

For example, in Fig. 3, $\{A_1\}$ is the minimum cardinality sufficient set in $R(F)$. Therefore, we have $|M(F)| = 1$. If we choose our first guess from the vertices in A_1, only one guess is enough to finish the decoding successfully. Since $p(F) = 4$ the number of guesses satisfies $1 \leq Z \leq 4$. However, it is easy to show that the number of required guesses is always less than or equal to two for this example.

The preceding discussion shows the connection between the graph-theoretic properties and the number of guesses. It would be nice if we could use these arguments to obtain some probabilistic results such as finding the average number of guesses for a specific code and channel. Note that we deal with a finite-length analysis and asymptotic analyses are not useful. In fact, we need some discussion based on the BER of the iterative decoder for a finite-length code to find the probabilistic properties of the given algorithm.

C. Improving Algorithms B and C by Reduction of Number of Guesses

Let g be the Tanner graph of an LDPC code and Z be the number of guesses required by Algorithm B or C when the iterative algorithm A fails. Later we will show that Z is very small (usually $E(Z) < 2$). However, we can still reduce the number of guesses. Here, we introduce one method to reduce Z. We first need to give some results based on the two-edge-connected components of the Tanner graph of LDPC codes.

Assume that all the vertices in g, the Tanner graph of the code, have a degree at least two. For any $A \subseteq V \cup C$ we defined $I_g(A)$ as the graph induced by the vertices in A and their neighbors. The graph $I_g(A)$ is sparser than the graph g in the sense that the degree of each vertex in $I_g(A)$ is less than or equal to the degree of that vertex in g. Let $S \subseteq V$ be a stopping set. Then, obviously $I_g(A)$ has at least one cycle because any vertex in $I_g(A)$ has a degree at least two. Let h^j be the jth row of H. Suppose $x = (x_1, x_2, \ldots, x_n)$ is a codeword and x_j is the bit corresponding to the variable node v_j. Any row h^i can be written as a parity-check equation in the form

$$E^i = \sum_{j=1}^{n} h_{ij} x_j = 0 \tag{23}$$

where h_{ij} is the element in the ith row and the jth column of H. Consider the case that the iterative decoder fails but the ML decoder can decode the received word. Let S_m be the set of variable nodes that the iterative decoder cannot decode. Consider a variable node $v_t \in S_m$. In this case, there exist $I \subseteq \{1, 2, \ldots, m\}$ and $U \subset V \backslash S_m$ such that

$$\sum_{j \in I} E^j = x_t + \sum_{x_j \in U} x_j. \tag{24}$$

We say that the set of parity checks corresponding to $\{h^j : j \in I\}$ frees the variable node v_t and we call this set of parity checks a freeing set for v_t. Let A be a subgraph of g. We define $C(A)$ as the set of parity-check nodes in A. For $k \geq 2$, we say a graph is k-edge-connected if it has at least two vertices and no set of at most $k-1$ edges separates it.

Theorem 5: Assume $S \subset V$ be a nonempty stopping set such that the ML decoder can decode the word when the set S is erased. Suppose we receive a word for which S is the unique maximum stopping set. Then there exists a two-edge-connected subgraph of $I_g(S)$, say g_S, such that the set of parity checks in g_S, i.e., $C(g_s)$, frees an erased variable node.

Proof: Let y be the word constructed from x by marking the variables in S as erasures. Since the ML decoder can decode y, there exists a freeing set for any of the variable nodes in S. Let v_t be an arbitrary variable node in S and F be a minimum freeing set for v_t. Define $D = I_{I_g(S)}(F)$, then D is a connected graph. Otherwise, D has at least two components, A and B. Note that all the variable nodes in D except one (the variable node v_t) have even degrees in D. Thus, in at least one of the two subgraphs A and B (assume A), all the variable nodes have even degrees. Therefore, $F - C_g(A)$ is a freeing set for v_t. This contradicts the assumption that F is minimum (note that $C_g(A) \neq \emptyset$).

If D is two-edge-connected we are done, so we may assume D is not two-edge connected. We consider two cases as shown in Fig. 4.

- Case1: $\deg_D(v_t) > 1$, where $\deg_D(v)$ denotes the degree of the vertex v in the graph D. In this case, the degree of any variable node in D is at least two. By our assumption and definition of D the degree of any check node in D is at least two. Since the degree of each vertex in D is at least two, D contains at least one cycle. Thus, D contains at least one two-edge-connected component. Let D' be a graph obtained by contracting any two-edge-connected component of D to a vertex. Then, D' is a tree. This is because if D' had a cycle then that cycle would be in a two-edge-connected component and would have been contracted to a vertex. Note that D' has at least two vertices otherwise D would be two-edge-connected. Since D' is a tree, it has at least two leaves (vertices of degree one). Moreover, since $\deg_D(v) > 1$ for all $v \in V(D)$, these leaves must correspond to two-edge-connected components C_1 and C_2 in D. Since C_1 and C_2 are disjoint, for at least one of them, say C_1, we have $v_t \notin C_1$. Next we

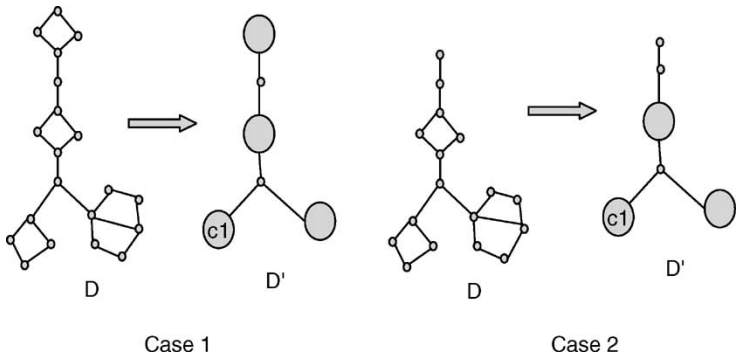

Fig. 4. Construction of the graph D'.

show that $C(C_1)$ is a freeing set for a variable node. Additionally, C_1 is two-edge-connected. The existence of C_1 proves the theorem.

Let $e = uw$ be the only edge that is connected to C_1 in D'. Assume $u \in C$ and $w \notin C$. If w is a variable node then $C(C_1)$ will free w because w has degree one in $I_g(C_1)$ and all the other variable nodes in $I_g(C_1)$ have even degree. This is because the degrees of these variable nodes in D and C_1 are the same. Now if w is a check node, then u is a variable node. Then $C(C_1)$ will free u. This is because by the above assumption $u \neq v_t$. Therefore, $\deg_D(u)$ is even and $\deg_{C_1}(u) = \deg_D(u) - 1$ is an odd number.

- Case2: $\deg_D(v_t) = 1$. We construct the graph D' Similar to Case 1 and v_t will be one of the leaves in D'. But D' has at least one more leaf that corresponds to a two-edge-connected component. Call this leaf C_1. Obviously $v_t \neq C_1$. Therefore, using the argument in Case 1, $C(C_1)$ is a freeing set for a variable node.

Fig. 4 shows the above argument. □

The immediate result of Theorem 5 is the following corollary.

Corollary 1: If we append all the parity-check equations that are formed by adding the parity-check equations in the two-edge-connected subgraphs of g to H, then the iterative decoding on the new H is equivalent to the ML decoding.

Obviously, this is not feasible because there are lots of such equations. However, we will show in the following sections that we can exploit Theorem 5 in order to improve the iterative decoding.

Corollary 2: If we apply the message-passing algorithm to an acyclic graph it will be equivalent to ML decoding.

This is a well-known result that can be proved for the erasure channel as follows. Let S be the stopping set that remains after the iterative decoding. For any set c^* of check nodes in $I_g(S)$, the graph that is induced by the vertices in c^* and their neighbors in $I_g(S)$ has at least two leaves. Since any check node in c^* has a degree at least two in $I_g(S)$, these leaves must be variable nodes. Therefore, the set c^* cannot be a freeing set for any variable node. Note that in any acyclic graph we have at least two variable nodes of degree one (we are assuming that check nodes have always degrees greater than one) and any stopping set in the graph must contain at least two variable nodes of degree one.

Let $C(g) = \{c_1, c_2, \ldots, c_m\}$ be the set of check nodes in g, the Tanner graph of the parity-check matrix H. We define the set $T \subseteq 2^{C(g)}$ as following. For any set

$$R = \{c_j : j \subseteq \{1, 2, \ldots, m\}\}$$

we have $R \in T$ if and only if $I_g(R)$ is a two-edge connected subgraph of g. Let $EQ(H)$ be the set of parity-check equations that are obtained by adding the set of parity-check equations from an element of T. Recall from Theorem 5 that the equations in $EQ(H)$ are sufficient for ML decoding. However, the number of these equations is extremely high and we cannot use all of them in the iterative decoding. Note that all of these equations are redundant because they are obtained by adding some parity-check equations in H. However, it turns out that by using a very small number of suitably chosen equations from this large set of equations, we can reduce the number of guesses in Algorithms B and C. Again we use these equations whenever Algorithm A fails. Note that any two-edge-connected graph is composed of several cycles. A cycle is the simplest two-edge-connected graph. Here, we only consider short cycles. Let C_{2l} be the number of cycles of length $2l$ in $g(d_v, d_c)$. It is shown in the Appendix that the expected value of C_{2l} is equal to

$$E(C_{2l}) = \binom{n}{l}\binom{m}{l}\frac{l!(l-1)!}{2}\left\{\frac{[d_v d_c(d_v-1)(d_c-1)]^l}{E\times(E-1)\cdots(E-2l+1)}\right\} \tag{25}$$

where $m = (1-R)n$ is the number of check nodes and $E = nd_v = md_c$ is the number of edges in the graph. For a constant number l we have

$$\lim_{n\to\infty} E(C_{2l}) = \frac{\left[(d_v-1)(d_c-1)\right]^l}{2l} \tag{26}$$

and for $0 < \theta < \alpha = 1 - R$

$$\begin{aligned} c(\theta) &= \lim_{n\to\infty}\frac{1}{n}\ln(E(C_{2\theta n})) \\ &= H(\theta) + \alpha H\left(\frac{\theta}{\alpha}\right) + \theta\ln[d_v d_c(d_v-1)(d_c-1)] \\ &\quad - d_v H\left(\frac{2\theta}{d_v}\right) - 2\theta\ln 2 \end{aligned} \tag{27}$$

where $H(x) = -x\ln(x) - (1-x)\ln(1-x)$. Therefore, the average number of the finite-length cycles is a constant and does not increase with n. Obviously, the same argument works for irregular codes. In order to reduce the number of guesses in Algorithm B or C we find some parity-check equations that construct short cycles (cycles of lengths four or six) in the Tanner graph and add them together to find new parity-check equations.

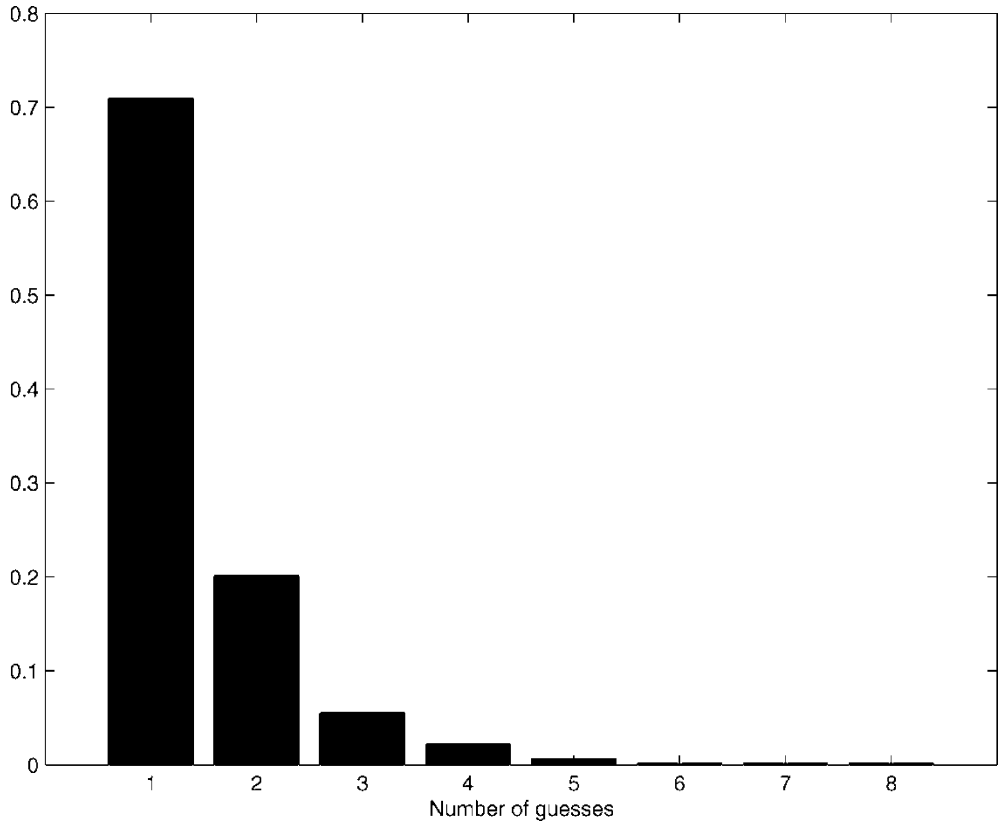

Fig. 5. Distribution of the number of guesses that is required for successful decoding at $\epsilon = 0.36$.

Note that these equations are used only if the iterative decoding (Algorithm A) fails. We will show that using a small number of these equations suffices to reduce the number of guesses. Using (26) we can find the expected number of equations that we need to consider. For example, in $g(3,6)$, if we just take parity-check equations that construct cycles of lengths four or six, on the average we will find around 192 equations.

D. Simulation Results

In this subsection, we provide some empirical results. First, we experimentally verify the main claim that we made in Section III-A (i.e., a very few number of guesses is enough to finish the decoding). We then give simulation results for the LDPC codes of lengths $n = 1000$ and $n = 10\,000$ and evaluate the performances of Algorithms A, B, and C. We compare these algorithms based on the BER, average speed, and the speed of decoding for a specific received word. Notice that the LDPC codes that we use here are not optimized for the corresponding length and rate. We did the simulations for half rate codes and for each length we picked a code with reasonable performance and degree distributions. In fact, we observed that for a fixed length and a maximum degree, the relative performances of the algorithms are roughly independent of the degree distribution of the code.

Let us first study the number of guesses in Algorithms B and C when Algorithm A fails. Again, let us define the random variable Z to be the number of required guesses when the standard iterative decoding (Algorithm A) fails to decode a received word. For the length $n = 1000$, we considered the following degree distribution:

$$\lambda_1(x) = 0.0769x + 0.6923x^2 + 0.2308x^5$$
$$\rho_1(x) = 0.4615x^5 + 0.5385x^6. \tag{28}$$

To evaluate the number of required guesses we set $g_{\max} = \infty$ and decoded 10^{10} bits that were transmitted over the BEC with the erasure probability $\epsilon = 0.36$. Fig. 5 shows the empirical probability density function for the number of required guesses. We note that in more than 70% of the cases for which the iterative decoder fails, only one guess is enough to complete the decoding successfully. We also note that the number of required guesses is always less than or equal to eight. The error rate of Algorithm A is about 10^{-5}. Even if we limit the maximum number of guesses to four, using Algorithms B or C we can improve the error rate by almost two orders of magnitude.

However, the situation changes when we increase ϵ. For example, for $\epsilon = 0.39$, the average BER of the standard iterative decoder (Algorithm A) is 0.0039. Fig. 6 shows the empirical probability density function for the number of required guesses. The figure shows that the number of required guesses increases as we increase ϵ. For example, if we limit the maximum number of guesses $g_{\max}$ to four, we can decrease the BER by only one order of magnitude using Algorithms B or C. However, it is worth noting that even for $\epsilon = 0.39$, the average number of required guesses is still very small ($g_{\text{av}} = 2.23$).

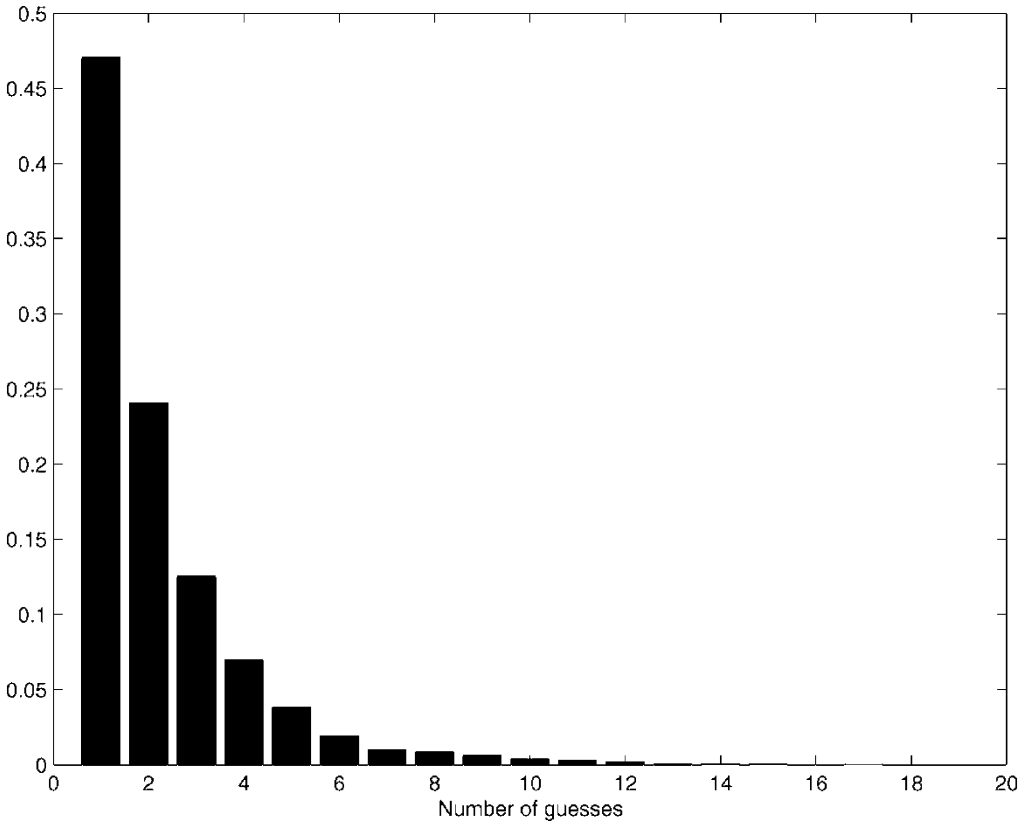

Fig. 6. Distribution of the number of guesses that is required for successful decoding at $\epsilon = 0.39$.

Let us now examine the effect of the code length on the number of guesses. We picked an LDPC code of length 10^4 with the following distributions:

$$\lambda_2(x) = 0.4706x^2 + 0.2353x^7 + 0.2941x^{29}$$
$$\rho_2(x) = 0.7843x^9 + 0.2157x^{10}. \quad (29)$$

Fig. 7 depicts the empirical probability density function of Z for this code. Note again that the number of guesses is largely concentrated on the small values (less than or equal to four). Therefore, we conclude that the number of required guesses is small for most of the practical code lengths. Therefore, Algorithms B and C are efficient. Note that in the preceding example, when the standard iterative decoding fails, there are about 3500 erasures left when the decoding stops. However, in most of the cases by knowing the values of less than or equal to 4 erased bits the decoder can find the value of all the 3500 erasures!

Now we examine the performance of the proposed algorithms. Note that Algorithms B and C have almost the same BER. In all simulations we set $g_{\max}$ to 6. Fig. 8 shows the performance of Algorithms A and C for a code from the ensemble $g(\lambda_1, \rho_1)$ with the length 1000. The figure shows that the gap between the BER of the two algorithms increases as ϵ decreases. At $\epsilon = 0.4$, the BER of Algorithm A is 20 times bigger than the BER of Algorithm C. This gap increases to about three orders of magnitude when ϵ is reduced to 0.36. This suggests that Algorithm C can alleviate the error floor problem in LDPC codes. Specifically, this algorithm can be very useful when very small BER is required. On the other hand, for the large values of ϵ, the improvement due to Algorithm C becomes negligible. Our simulations show that for the values of ϵ that the BER of the iterative decoder is less than or equal to 10^{-2}, a good improvement is possible in the BER by applying Algorithm C.

There are several stopping sets in a Tanner graph of an LDPC code. Some stopping sets are weak in the sense that if the values of one or two bits in the stoping set is exposed to the iterative decoder, the decoder can finish the decoding successfully. For these stopping sets, $|M(F)| = |M(I_g(s))|$ is a very small number. On the other hand, some stopping sets are very strong and their $|M(F)|$ is a large number. When ϵ is close to 1, we usually face with strong stopping sets. For example, at $\epsilon = 1$ the strongest stopping set (i.e., $V(g)$) occurs. As we decrease ϵ, the strong stopping sets become less probable while the weak sopping sets become more probable. Therefore, a smaller number of guesses is required to finish the decoding. This discussion explains why the gap between the BERs of Algorithms A and C increases as ϵ decreases.

Table I shows the average number of required guesses by Algorithm C for the received blocks that Algorithm A fails. The table suggests that the average number of guesses is very small. Note that the values in Table I are slightly smaller than the average values obtained by Figs. 5 and 6 because for those diagrams we have $g_{\max} = \infty$, but Table I is obtained for $g_{\max} = 6$.

Fig. 9 shows the performance of Algorithms A and C for a code from the ensemble $g(\lambda_2, \rho_2)$ with a length of 10^4. We see that the results are similar to that of the code with the length 1000. The above results are obtained when $g_{\max} = 6$.

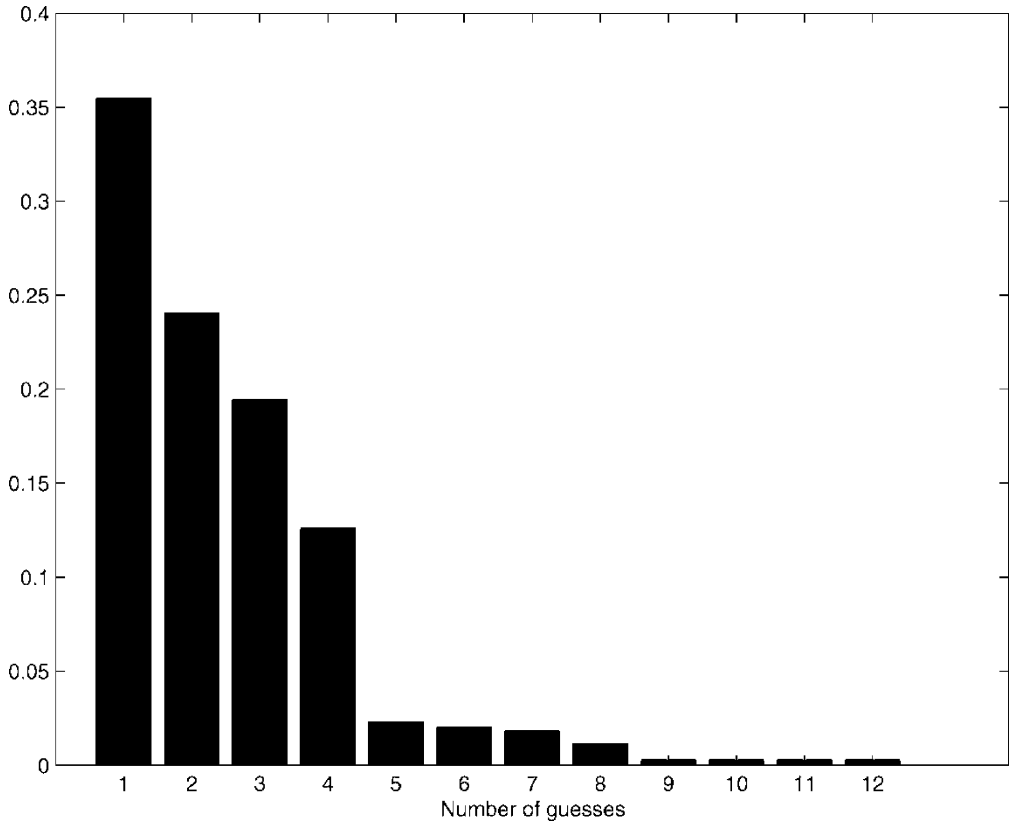

Fig. 7. Distribution of the number of guesses that is required for successful decoding at $\epsilon = 0.39$ and code length $= 10^4$.

TABLE I
THE AVERAGE NUMBER OF GUESSES FOR THE LDPC CODE OF LENGTH 1000

ϵ	Average number of guesses
.36	1.38
.37	1.59
.38	1.78
.39	2.11

TABLE II
THE AVERAGE NUMBER OF GUESSES FOR THE LDPC CODE OF LENGTH 10 000 THAT HAS AN ERROR FLOOR

ϵ	Average number of guesses
.450	2.34
.455	2.78
.460	3.12
.465	3.67

We note that, in both ensembles $g(\lambda_1, \rho_1)$ and $g(\lambda_2, \rho_2)$, the minimum distance grows linearly with the codes lengths. In fact, when we generated codes from these ensembles, we made sure that these codes did not have small stopping sets that could cause an error floor. In order to see the effect of the error floor on Algorithms B and C, we generated a code of length 10^4 from the ensemble defined by

$$\lambda_3(x) = 0.2223x + 0.3884x^2 + 0.1934x^7 + 0.1959x^{14}$$
$$\rho_3(x) = 0.88x^6 + 0.22x^7. \tag{30}$$

We note that for this ensemble we have $\lambda_3'(0)\rho'(1) = 1.5154 > 1$. Thus, with high probability, a code generated from this ensemble has a small minimum distance and shows an error floor. Fig. 10 shows the performance of Algorithms A and C for the code. We observe that for high BERs, Algorithm C shows some improvement over Algorithm A. However, as we approach the error floor region, the improvement decreases. This is because, for this code, even the ML decoder shows an error floor (because of small minimum distance) and thus, using improved decoding methods, does not help much in the error floor region. Table II shows the average number of guesses for this code.

Now we present some experimental results for the running time of the algorithms. Clearly, these results are dependent on the specific computer program and the platform we use. However, a relative timing comparison can be made from these simulations. We give the results for a code of length 1000 from the ensemble $g(\lambda_1, \rho_1)$ and a code of length 10^4 from the ensemble $g(\lambda_2, \rho_2)$. The erasure probability of the channel ϵ is chosen such that the BER of Algorithm A is 10^{-3}. In all cases, we decoded 10^{10} bits and measured the average running time and the maximum running time for the decoding of received blocks. Let $T_A(n)$, $T_B(n)$, and $T_C(n)$ show the average time of decoding of the LDPC code of length n from the given ensembles using Algorithms A, B, and C, respectively. Table III shows the relative average time of Algorithms B and C with respect to the standard iterative decoding. From the table we conclude that the average running time of all the above algorithms are almost the same.

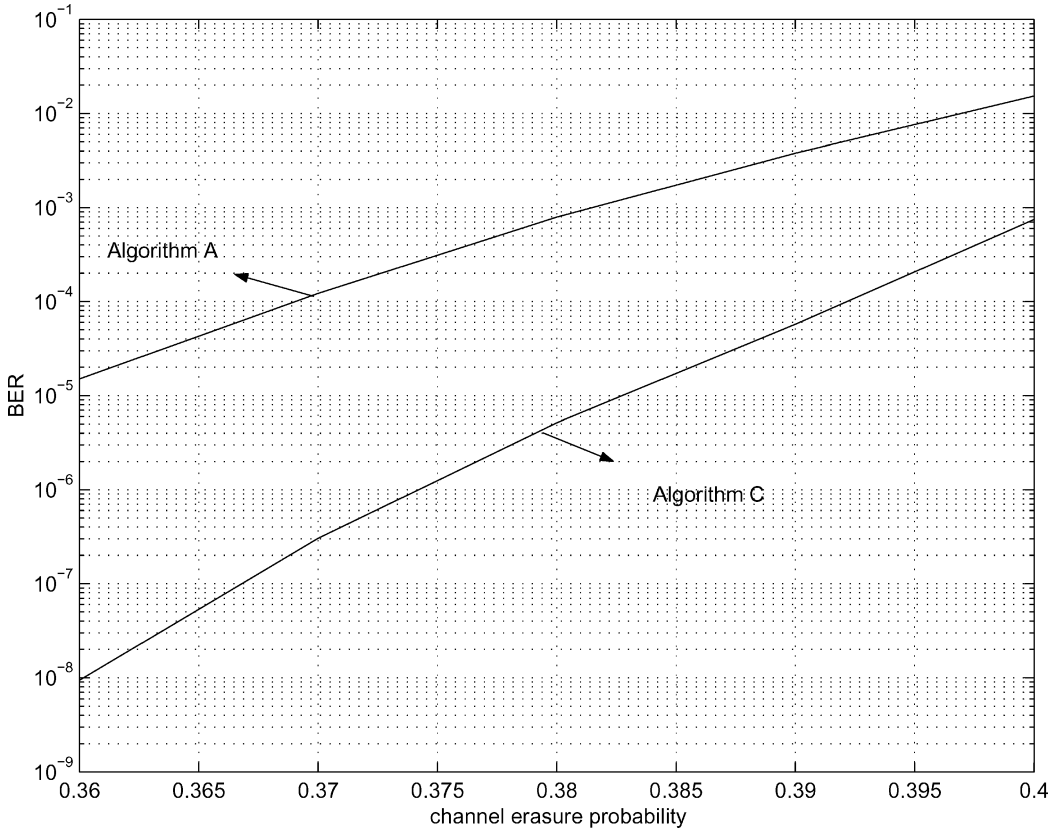

Fig. 8. Comparisons of the BERs of Algorithms A and C for code length $n = 10^3$.

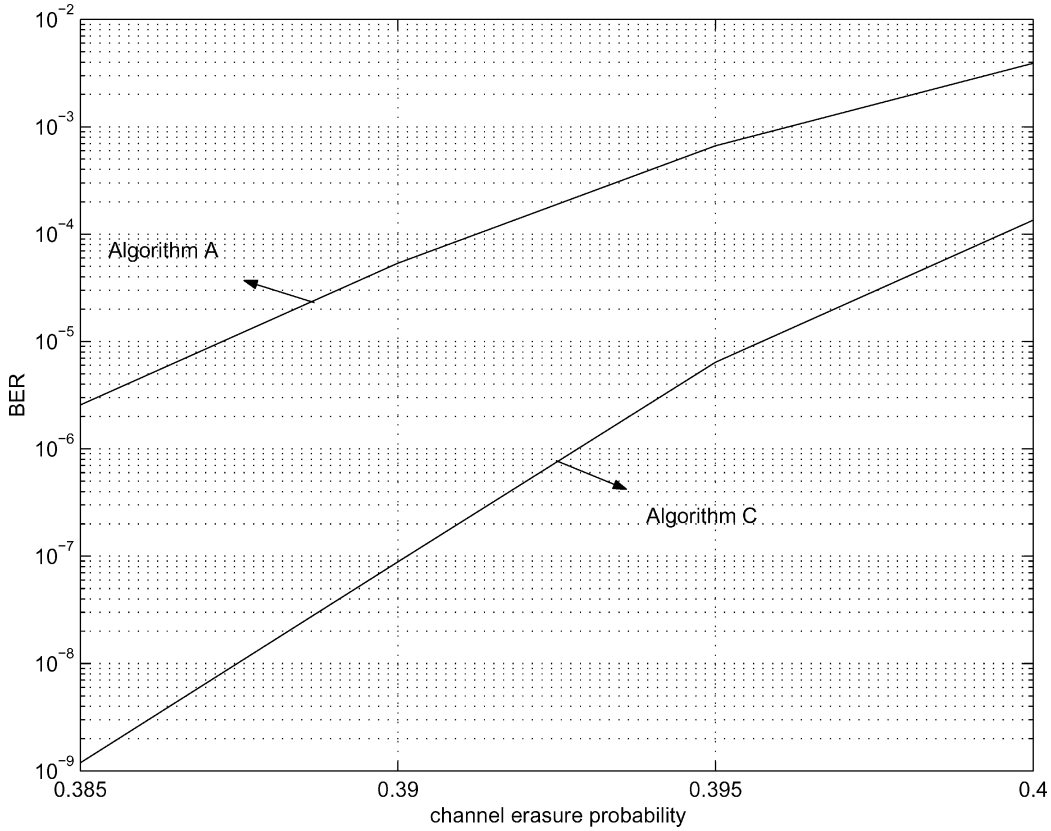

Fig. 9. Comparisons of the BERs of Algorithms A and C for a code of length $n = 10^4$.

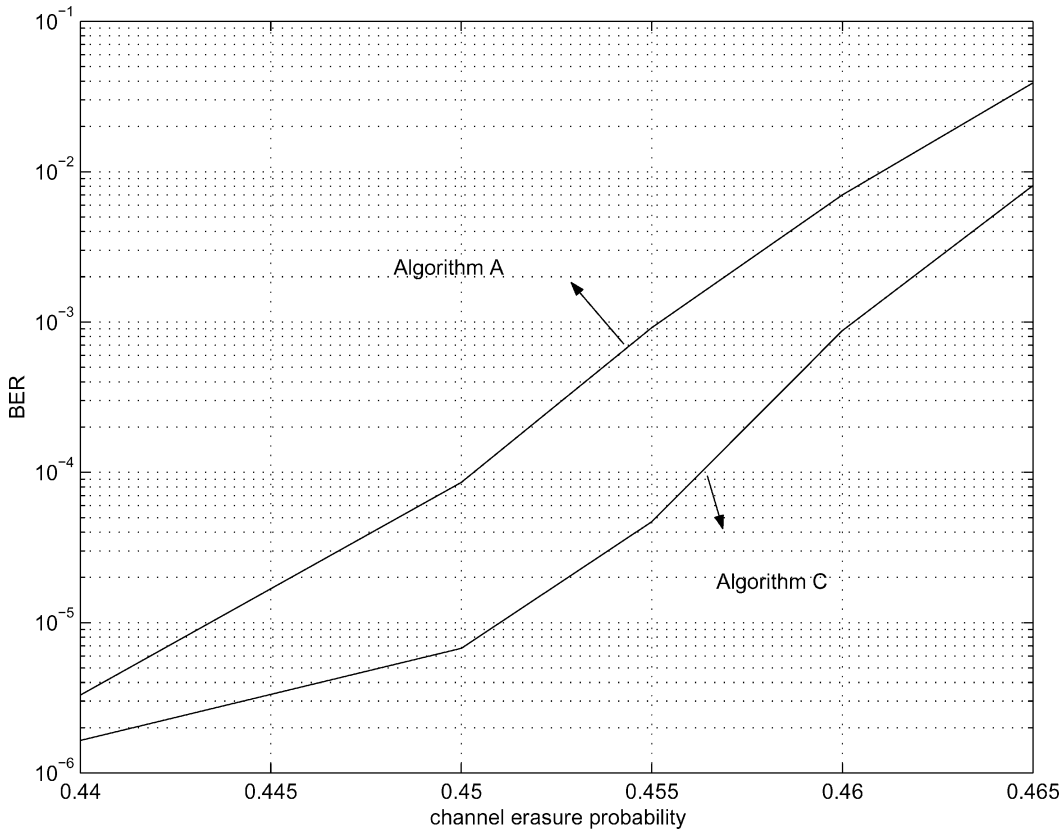

Fig. 10. Comparisons of the BERs of Algorithms A and C for a code of length $n = 10^4$ that has an error floor.

Recall that we defined $T_B(y, n)$ as the time that Algorithm B needs to decode a received word y. Let $T_B^m(n)$ show the maximum value of $T_B(y, n)$ over all the received blocks. We define

$$R_B(n) = \frac{T_B^m(n)}{T_A(n)}.$$

$R_C(n)$ is defined similarly. Table IV shows the values of $R_B(n)$ and $R_C(n)$. The table suggests that although Algorithm B has a good average running time, it can be very slow for some specific received blocks. However, the running time of Algorithm C for any received block is always less than 10 times the running time of the standard iterative decoder. Combining this with the fact that the average running time of Algorithm C is almost the same as that of the iterative decoder we conclude that Algorithm C is efficient in terms of running time. Since Algorithm C is fast and has a BER smaller than the standard iterative decoder, it can be considered as an efficient way of decoding LDPC codes over the BEC.

As we have already mentioned, it is possible to reduce the average number of guesses using some redundant equations. Again, we chose a code of length 1000 from the ensemble $g(\lambda_1, \rho_1)$. As we discussed in the previous section, we looked for cycles of lengths 4 and 6 in the Tanner graph of the code. For any of these cycles we added the rows of H corresponding to the parity-check equations in the cycle and put these parity-check equations as rows of a matrix H'. In total, we chose 374 equations. Hence, H' had 374 rows. We set $g_{\max} = \infty$ and decoded 10^{10} bits. In the first experiment, we used Algorithm C. In the second experiment we used the same algorithm. However, we also used the parity-check equations in H' whenever the iterative decoder failed and we needed to perform the guessing procedure. Let us call the second algorithm C$'$. Table V shows the results. In this table, g_{av} and g'_{av} are the average numbers of required guesses when we needed to perform the guessing process in Algorithms C and C$'$. The table shows that the average number of guesses is substantially

TABLE III
COMPARISON OF AVERAGE RUNNING TIME OF DIFFERENT ALGORITHMS

	$n = 1000$	$n = 10^4$
$\frac{T_B(n)}{T_A(n)}$	1.06	1.08
$\frac{T_C(n)}{T_A(n)}$	1.01	1.02

TABLE IV
MAXIMUM RATIO OF THE RUNNING TIMES OF ALGORITHMS B AND C TO THE RUNNING TIME OF ALGORITHM A

	$n = 1000$	$n = 10^4$
$R_B(n)$	67.2	71.3
$R_C(n)$	6.2	9.7

TABLE V
THE AVERAGE NUMBER OF REQUIRED GUESSES

	g_{av}	g'_{av}
$\epsilon = .39$	2.3	1.56
$\epsilon = .38$	2.06	1.27
$\epsilon = .37$	1.74	.94

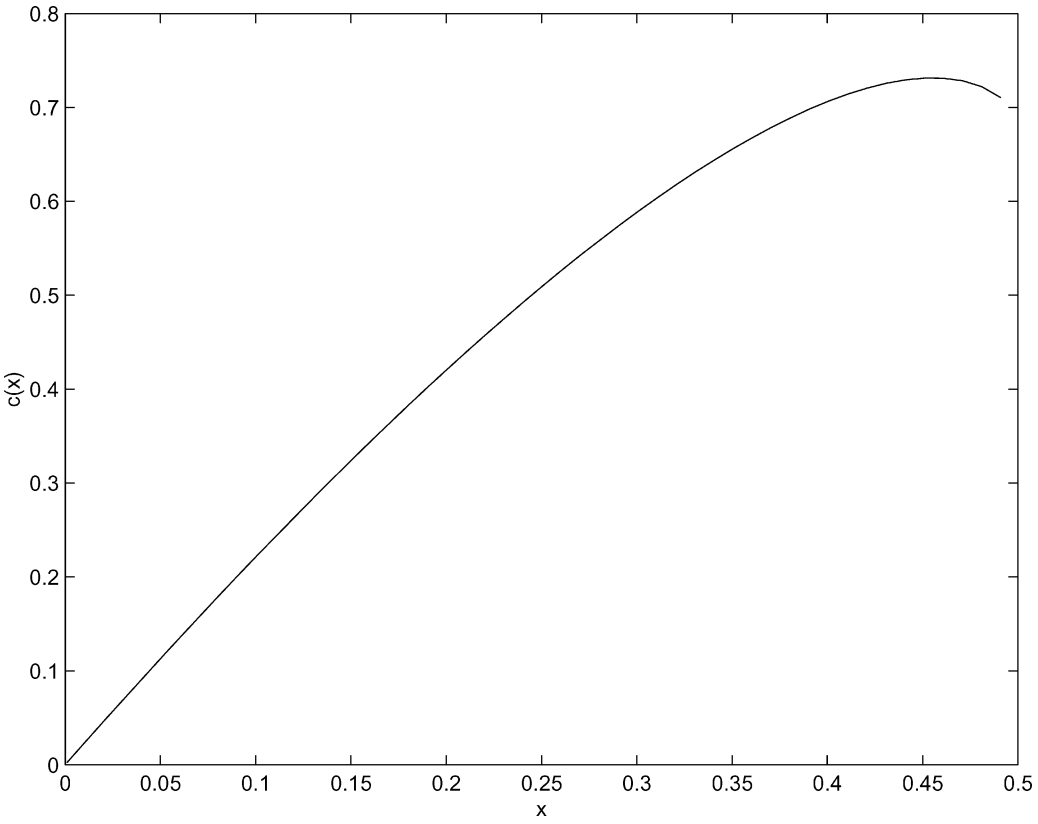

Fig. 11. Cycle distribution in $g(d_v, d_c)$.

smaller in Algorithm C$'$. Note also that the average number of guesses at $\epsilon = 0.37$ is less than 1. This is because sometimes the parity-check equations in H' are sufficient to successfully finish the decoding and we do not need any guesses.

IV. Conclusion

In this paper, we studied some properties of the ML and iterative decoding of LDPC codes when they are used over the BEC. We derived both lower and upper bounds on the ML capacity of the ensembles of LDPC codes. The tightness of the bounds was depicted for regular codes by using some examples. We proposed two algorithms for decoding LDPC codes over the BEC. It was shown by simulations that the proposed algorithms had a better BER than the standard iterative decoder. More specifically, the improvement up to three orders of magnitude was obtained in the BER. It was also demonstrated that the proposed algorithm (Algorithm C) has almost the same running time as the iterative decoder. Therefore, we conclude that Algorithm C can be considered as an efficient method for decoding LDPC codes over the BEC. Since finding good finite-length LDPC codes is still a challenging problem, the decoding scheme presented in this paper may compensate for this problem. We also provided some graph-theoretic results that are useful for bounding the complexity of the decoding algorithms and improving them. Although we demonstrated the superiority of Algorithm C for standard LDPC codes, we expect that the proposed algorithm can also improve the performance of other ensembles such as codes based on cascaded bipartite graphs [1].

Appendix
Number of Cycles

In this appendix, we calculate the average number of cycles in a Tanner graph of LDPC codes. For clarity of exposition we perform the computations for regular graphs. The generalization to irregular graphs is trivial. Here we use the same ensemble described in [6]. To each variable or check node we assign d_v or d_c sockets, respectively. We label the variable nodes and check nodes separately with the set $\{1, 2, \ldots, nd_v\}$. We then pick a random permutation π on $E = nd_v$ letters. For each i, we put an edge between the socket i and $\pi(i)$. Let $v_1, v_2, \ldots, v_l$ be l arbitrary variable nodes and $c_1, c_2, \ldots, c_l$ be l arbitrary check nodes. Let $s_{v_1}, s_{v_2}, \ldots, s_{v_l}$ be l sockets such that s_{v_j} belongs to v_j and let $s_{c_1}, s_{c_2}, \ldots, s_{c_l}$ be l sockets such that s_{c_j} belongs to c_j. Then the probability that these sockets construct the cycle $v_1 - c_1 - v_2 - c_2 \cdots - c_l - v_1$ is equivalent to

$$\frac{1}{E} \times \frac{1}{E-1} \times \cdots \times \frac{1}{E-2l+1}. \tag{31}$$

Since any variable node has d_v sockets and any check node has d_c sockets, the probability of having the cycle $v_1 - c_1 - v_2 - c_2 \cdots - c_l - v_1$ in the Tanner graph is

$$\left\{ \frac{[d_v d_c (d_v - 1)(d_c - 1)]^l}{E \times (E-1) \cdots (E-2l+1)} \right\}. \tag{32}$$

Therefore, the probability that there exists a cycle of length l with vertices $v_1, v_2, \ldots, v_l$ and $c_1, c_2, \ldots, c_l$ is

$$\frac{l!(l-1)!}{2} \left\{ \frac{[d_v d_c (d_v - 1)(d_c - 1)]^l}{E \times (E-1) \cdots (E-2l+1)} \right\}. \tag{33}$$

This proves (25). Evaluating (25) for a constant value of l results in (26). Using the following equations:

$$\lim_{n\to\infty}\frac{1}{n}\ln\binom{n}{n\theta}=H(\theta)$$

$$\lim_{n\to\infty}\left[\ln(n)-\frac{1}{n}\ln(n!)\right]=1$$

$$\lim_{n\to\infty}\left[\frac{1}{n}\ln[n(n-1)\cdots(n-n\theta+1)]-\theta\ln(n)\right]=H(\theta)+\theta\ln\left(\frac{\theta}{e}\right) \quad (34)$$

we get (27). Fig. 11 shows the function $c(\theta)$ for $g(3,6)$. Note that in the above model double edges are allowed. Therefore, we may have double cycles. However, one can show that the probability of having a double cycle is very small.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their helpful comments and suggestions. The authors are also grateful to Simon Litsyn, the Associate Editor, for handling the review.

REFERENCES

[1] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 569–584, Feb. 2001.

[2] C. Di, D. Proietti, İ. E. Telatar, T. Richardson, and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inform. Theory*, vol. 48, pp. 1570–1579, June 2002.

[3] M. A. Shokrollahi, "Capacity-achieving sequences," *IMA Volumes in Mathematics and its Applications*, vol. 123, pp. 153–166, 2000.

[4] ——, "New sequences of linear time erasure codes approaching the channel capacity," *AAECC*, pp. 65–76, 1999.

[5] P. Oswald and M. A. Shokrollahi, "Capacity-achieving sequences for the erasure channel," *IEEE Trans. Inform. Theory*, vol. 48, pp. 3019–3028, Dec. 2002.

[6] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 599–618, Feb. 2001.

[7] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619–637, Feb. 2001.

[8] H. Pishro-Nik and F. Fekri, "Improved decoding algorithms for low-density parity-check codes," in *Proc. 3rd Int. Symp. Turbo Codes and Related Topics*, Brest, France, Sept. 2003.

[9] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, Mar. 1999.

[10] G. Miller and D. Burshtein, "Bounds on the maximum-likelihood decoding error probability of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 2696–2710, Nov. 2001.

[11] D. Burshtein, M. Krivelevich, S. Litsyn, and G. Miller, "Upper bounds on the rate of ldpc codes," *IEEE Trans. Inform. Theory*, vol. 48, pp. 2437–2449, Sept. 2002.

[12] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 533–547, Sept. 1981.

[13] S. Litsyn and V. Shevelev, "On ensembles of low-density parity-check codes: Asymptotic distance distributions," *IEEE Trans. Inform. Theory*, vol. 48, pp. 887–908, Apr. 2002.

[14] D. Burshtein and G. Miller, "Asymptotic enumeration method for analyzing LDPC codes," *IEEE Trans. Inform. Theory*, submitted for publication.

[15] R. G. Galleger, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[16] C. Di, T. Richardson, and R. Urbanke, "Weight distributions: How deviant can you be?," in *Proc. 2001 IEEE Int. Symp. Information Theory*, Washington, DC, 2001, p. 50.

[17] R. Urbanke. Unpublished research. [Online]. Available: http://lthcwww.epfl.ch/research/ldpcopt/

[18] S. Litsyn and V. Shevelev, "Distance distributions in ensembles of irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, submitted for publication.

[19] I. Sason and R. Urbanke, "Parity-check density versus perfprmance of binary linear block codes over memoryless symmetric channels," *IEEE Trans. Inform. Theory*, vol. 49, pp. 1611–1635, July 2003.

[20] T. Richardson, A. Shokrollahi, and R. Urbanke, "Finite-length analysis of low-density parity-check ensembles for the binary erasure channel," in *Proc. 2002 IEEE Int. Symp. Information Theory*, Lausanne, Switzerland, June/July 2001, p. 1.