An Efficient Maximum Likelihood Decoding of LDPC Codes Over the Binary Erasure Channel

David Burshtein and Gadi Miller
School of Electrical Engineering
Tel-Aviv University
Tel-Aviv 69978, Israel
Email: burstyn@eng.tau.ac.il

***Abstract*— We propose an efficient maximum likelihood decoding algorithm for decoding low-density parity-check codes over the binary erasure channel. We also analyze the computational complexity of the proposed algorithm.**

***Index Terms*— Low-density parity-check (LDPC) codes, Binary erasure channel (BEC), Iterative decoding, Maximum likelihood (ML) decoding.**

I. Introduction

The binary erasure channel (BEC) model was proposed by Elias [7] in 1955. The BEC has been recently used for modelling the transmission of information over the Internet. Luby *et al.* [13] proposed an iterative algorithm for decoding low-density parity-check (LDPC) codes over the BEC and showed that the proposed scheme can approach channel capacity arbitrarily close. The iterative decoding algorithm proposed in [13] is equivalent to Gallager's soft decoding algorithm [8] when applied to the BEC.

Although iterative decoding can achieve channel capacity for an arbitrary BEC, there is still a gap between the threshold erasure probability of iterative decoding and optimal (maximum likelihood, ML) decoding for any fixed code structure. This gap can sometimes be significant [2]. In fact, in order to achieve reliable communication over a given BEC, the averaged left and right degrees of the LDPC ensemble are typically much larger when iterative decoding is applied rather than ML. In this paper we extend the iterative decoding algorithm and propose efficient ML decoding. To this end we apply techniques presented in [18] in the context of efficient encoding of LDPC codes, and propose practical algorithms for solving sparse linear equations over $GF(2)$ for efficient decoding of LDPC codes over the BEC.

Efficient algorithms for solving sparse linear equations over finite fields have been proposed in the context of calculating discrete logarithms and factoring integers in cryptographic applications, e.g. [3]-[5], [10], [15]-[17], [21]-[22] and references therein. Three families of algorithms were proposed. The first is structured Gaussian elimination [16] whose purpose is to convert the given sparse linear equations to a new system with smaller dimensions, which is hopefully still sparse, and then solve this new system using another method. The second family includes the conjugate gradient and Lanczos algorithms [16], [5]. These algorithms are the finite field variants of the standard conjugate gradient [9] and Lanczos [11] algorithms for solving linear equations over the reals. A disadvantage associated with the finite field variants of these algorithms is that they are not guaranteed to produce a solution when the linear system is solvable [16], [21]. Improvements to these algorithms that avoid this problem were also proposed [16], [21]. Both algorithms require about $O(N^2)$ operations to solve a linear system of L sparse equations in N variables when both L and the number of non-zero elements in the system matrix are proportional to N. The third family is the Wiedemann [22] algorithm and its derivatives. This algorithm uses the Berlekamp-Massey algorithm and is guaranteed to provide a solution in about $O(N^2)$ operations. In [10] the performances of the various algorithms were compared on actual problems of integer factorization and discrete logarithm computations. It was noted that the Wiedemann algorithm was about as efficient as the conjugate gradient and the Lanczos algorithms, but the Wiedemann algorithm was more complicated to program. The paper [10] recommends on using structured Gaussian elimination in the initial processing stage. Parallel versions of both the Lanczos and the Wiedemann algorithms were proposed in [3] and [4]. A parallel version of the Lanczos algorithm that incorporates structured Gaussian elimination was proposed in [15].

In this paper we propose simple practical probabilistic algorithms for decoding LDPC codes over the BEC which are similar to the structured Gaussian elimination approach for solving sparse linear equations [16]. However the probabilistic nature of our algorithms enable us to evaluate their computational complexity, when decoding LDPC codes over the BEC, analytically using the differential equation techniques that were presented in [18]. Our algorithms can be viewed as a natural extension of the standard iterative decoding algorithm of LDPC codes over the BEC, for the case were we are willing to pay some additional computational cost (that can be adjusted by the user) in order to improve the performance.

The paper is organized as follows. In Section II we provide brief background information on LDPC codes and their iterative decoding over the BEC. In Section III we describe the straightforward Gaussian elimination way of performing ML decoding over the BEC, and in Section IV we present methods which are more computationally efficient to obtain the same result. We also analyze the computational complexity of the

Manuscript received August, 2002; revised July, 2004.

proposed algorithms. In Section V we present some examples. Section VI concludes the paper.

II. Iterative decoding of LDPC codes over the BEC

A. LDPC codes and graph representations

We assume the following ensemble of irregular LDPC codes [13]. The ensemble is described in terms of the Tanner graph representation [20] of the code. The Tanner graph is a bipartite graph with left and right nodes. The left nodes are associated with the variables. The right nodes are associated with the parity-check equations. The ensemble is characterized by two probability vectors,

$$\boldsymbol{\lambda} = (\lambda_2, \ldots, \lambda_c) \qquad \boldsymbol{\rho} = (\rho_2, \ldots, \rho_d)$$

where λ_l is the fraction of edges with left degree l, and ρ_l is the fraction of edges with right degree l. Let $\mathcal{E}$ denote the total number of edges in the graph. We now assign left and right sockets, such that each variable node with degree l contributes l left sockets, and each parity-check node with degree l contributes l right sockets. The set of left and right sockets are then matched by a random permutation of size $\mathcal{E}$ that is chosen with uniform probability among the set of all permutations of this size. The resulting Tanner graph defines a $(\boldsymbol{\rho}, \boldsymbol{\lambda})$-irregular code in the ensemble, such that the k, l'th element in the parity check matrix of the code is set to one if and only if the number of edges between the l'th variable node and the k'th check node is odd. Otherwise the k, l'th element in the parity check matrix is set to zero.

For convenience we define the polynomials

$$\lambda(x) = \sum_{l=2}^{c} \lambda_l x^{l-1} \qquad \rho(x) = \sum_{l=2}^{d} \rho_l x^{l-1}$$

The blocklength (number of variable nodes), N, is then given by

$$N = \mathcal{E} \sum_{l=2}^{c} \frac{\lambda_l}{l} = \mathcal{E} \int_0^1 \lambda(x) dx$$

Similarly, the number of parity check equations, L, is given by

$$L = \mathcal{E} \sum_{l=2}^{d} \frac{\rho_l}{l} = \mathcal{E} \int_0^1 \rho(x) dx$$

The planned rate of the code is

$$R \triangleq 1 - \frac{L}{N} = 1 - \frac{\int_0^1 \rho(x) dx}{\int_0^1 \lambda(x) dx}$$

The actual rate of the code is lower bounded by R due to a possible degeneracy in the parity check equations.

Let $\tilde{\boldsymbol{\lambda}} = (\tilde{\lambda}_2, \ldots, \tilde{\lambda}_c)$, where $\tilde{\lambda}_l$ is the fraction of left nodes with degree l. Similarly, let $\tilde{\boldsymbol{\rho}} = (\tilde{\rho}_2, \ldots, \tilde{\rho}_d)$, where $\tilde{\rho}_l$ is the fraction of right nodes with degree l. It is sometimes convenient to use the node perspective distribution $(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\rho}})$ rather than the edge perspective $(\boldsymbol{\lambda}, \boldsymbol{\rho})$. The left edge and node perspective distributions, $\boldsymbol{\lambda}$ and $\tilde{\boldsymbol{\lambda}}$ are related by

$$\lambda_i = \frac{i\tilde{\lambda}_i}{\sum_j j\tilde{\lambda}_j} \tag{1}$$

Similarly, the right edge and node perspective distributions, $\boldsymbol{\rho}$ and $\tilde{\boldsymbol{\rho}}$ are related by

$$\rho_i = \frac{i\tilde{\rho}_i}{\sum_j j\tilde{\rho}_j} \tag{2}$$

A special case of the $(\boldsymbol{\lambda}, \boldsymbol{\rho})$-irregular ensemble is the (c, d)-regular ensemble. In this case $\lambda(x) = x^{c-1}$ and $\rho(x) = x^{d-1}$. The planned rate is therefore $R = 1 - c/d$.

B. Iterative decoding over the BEC

Luby *et al.* [13], [12] showed that LDPC codes can be used for reliable communication over the BEC, under iterative decoding, at transmission rates arbitrarily close to channel capacity. The iterative decoding algorithm can be either Gallager's soft decoding algorithm, which utilizes message passing along edges, or the iterative matrix triangulation algorithm described in [18]. Both algorithms are equivalent in the sense that they produce the same result. More precisely, both succeed if and only if the set of erasures does not contain a stopping set [18]. Otherwise, both algorithms will fail decoding the bits in the largest stopping set which is a subset of the erasures.

Let the proportion of erasure messages (going from left to right) in the l-th iteration of Gallager's soft decoding algorithm be p_l. It can be shown [13], [12] that for N sufficiently large,

$$p_l = \delta\lambda\left(1 - \rho(1 - p_{l-1})\right) \tag{3}$$

where δ is the probability of channel erasure. The algorithm can correct a δ fraction of losses (erasures) in the channel if

$$\delta\lambda\left(1 - \rho(1 - x)\right) < x \tag{4}$$

for $x \in (0, \delta]$. In this case we say that the decoding succeeds. The largest value of δ satisfying (4) for given ensemble parameters $(\boldsymbol{\lambda}, \boldsymbol{\rho})$ is called the *threshold erasure probability* and is denoted $\delta^*(\boldsymbol{\lambda}, \boldsymbol{\rho})$. The quantity $\delta^*(\boldsymbol{\lambda}, \boldsymbol{\rho})$ can be calculated exactly using the techniques in [1].

III. ML decoding over the BEC: Straightforward approach

Denote the transmitted bit vector by $\mathbf{x} = (x_1, ..., x_N)$ and the received (corrupted) vector by $\mathbf{y} = (y_1, ..., y_N)$ where $x_i \in \{0, 1\}$ and $y_i \in \{0, 1, E\}$ for $i = 1, ..., N$ (E denotes erasure). The $L \times N$ parity-check matrix H satisfies $H\mathbf{x}^T = \mathbf{0}^T$ where $\mathbf{0}$ is of length L. Denote by $\mathcal{K}$ the set of the indices of known bits in $\mathbf{y}$, i.e. $\mathcal{K} = \{i : y_i \neq E\}$. Similarly, denote by $\overline{\mathcal{K}}$ the set of erasures, i.e. $\overline{\mathcal{K}} = \{i : y_i = E\}$.

Denote by $H_{\mathcal{K}}$ ($H_{\overline{\mathcal{K}}}$, respectively) the matrix obtained by taking from H only the columns corresponding to $\mathcal{K}$ ($\overline{\mathcal{K}}$). Similarly, $\mathbf{x}_{\mathcal{K}}$ and $\mathbf{x}_{\overline{\mathcal{K}}}$ are the vectors obtained by taking from $\mathbf{x}$ only the components corresponding to $\mathcal{K}$ and $\overline{\mathcal{K}}$. Since $\mathbf{0} = H\mathbf{x}^T = H_{\mathcal{K}}\mathbf{x}_{\mathcal{K}}^T + H_{\overline{\mathcal{K}}}\mathbf{x}_{\overline{\mathcal{K}}}^T$ we see that

$$H_{\overline{\mathcal{K}}}\mathbf{x}_{\overline{\mathcal{K}}}^T = H_{\mathcal{K}}\mathbf{x}_{\mathcal{K}}^T = H_{\mathcal{K}}\mathbf{y}_{\mathcal{K}}^T = \mathbf{z}^T \tag{5}$$

where $\mathbf{z}$ is a (length L) known vector. Thus, ML decoding over the BEC sums up to solving the linear system (5). Denoting the probability of erasure by δ, the weak law of large numbers dictates that $|\overline{\mathcal{K}}| = N(\delta + o(1))$ with probability 1. As long as

ML decoding is possible, the system (5) has a unique solution, which is the case if and only if the columns of $H_{\overline{\mathcal{K}}}$ are linearly independent [18].

Equation (5) is a linear system of L equations and $M_1 = (\delta + o(1))N$ variables. We assume that the solution uses Gaussian elimination. Hence the solution involves $\beta M_1^2 L + \gamma M_1^3$ operations, where β and γ are constants that depend on the specifics of the algorithm chosen to perform the elimination. This is so because, in general, the solution of the linear system can be divided to two parts: finding M_1 independent equations, and solving the set of M_1 linearly independent equations with M_1 variables. Since we may independently choose any method to do each of these two parts, we represent the overall complexity involved as the sum of the two parts. The overall complexity of the straightforward approach is hence

$$\left((1-R)\beta + \gamma\delta\right)\delta^2 N^3 \tag{6}$$

It should be noted, though that there are faster methods to solve a linear system of equations. The first fast method was proposed by Strassen [19] and it requires $O(N^{2.81})$ operations. As noted in [10] this method is practical for N on the order of several hundred. The later methods, of which [6] that requires $O(N^{2.376})$ is currently the fastest, are impractical.

IV. ML DECODING OVER THE BEC: IMPROVED APPROACH

In this section we describe ways in which the ML decoding complexity remains $O(N^3)$, but in which the constants are significantly reduced, providing a substantial practical improvement over the straightforward approach.

Suppose that in addition to the $N(1-\delta)+o(N)$ known bits received from the channel, we declare αN of the remaining bits as *reference variables*. Later we specify three methods for choosing these reference variables and determine the typical value of α in each case.

We now describe a variation on the belief propagation algorithm, expressing the $(\delta - \alpha)N$ unknown bits as (non-homogeneous) linear combinations of the reference variables. As explained shortly, if $\delta - \alpha$ is sufficiently small (depending on the code parameters), the iterative decoding procedure will almost surely terminate with all $(\delta - \alpha)N$ unknown variables expressed as combinations of the reference bits. Moreover, the complexity of this stage is $O(N^2)$.

More specifically, the procedure is simplest to explain in terms of the approximate triangulation process described in [18]. This process is depicted in Figures 1 and 2. Denoting the original low density matrix by $H_{L\times N}$, the algorithm proceeds as follows. The columns of the parity check matrix are first permuted so that the $(1-\delta)$ fraction of known variables, as well as the α fraction of reference variables, form the first columns. We assume a data structure in which the rows and the columns of the parity check matrix are addressed using some permutation, so that the complexity of the above described column permutation is $O(N)$.

We now perform a *diagonal extension step* [18]. This means that we check for degree-one rows in the residual matrix. At the beginning of the algorithm, this is the submatrix that consists of the $(\delta - \alpha)N$ rightmost columns in the parity check matrix. Since the matrix H is sparse, assuming a data structure in which the location of the '1' bits of each row is stored, the complexity of performing this step is $O(N)$. We then perform row and column permutations to bring the found bits to a diagonal form (Figure 1). Again, this involves $O(N)$ operations.

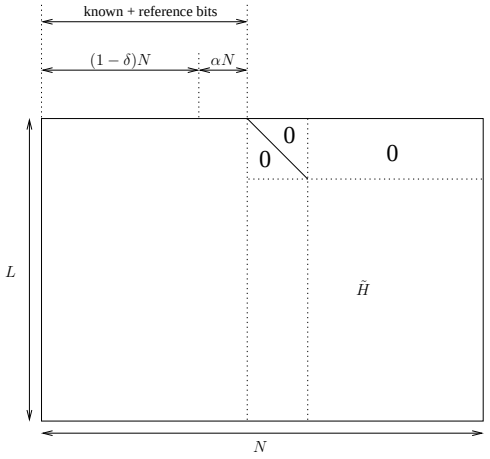

Fig. 1. The approximate triangulation procedure after a single diagonal extension step.

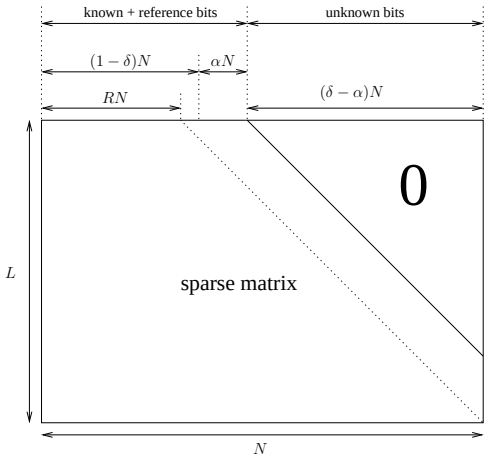

Fig. 2. The result of the approximate triangulation.

Next we apply diagonal extension steps on $\tilde{H}$ repeatedly until no weight one row remains. We assume that α is sufficiently large so that in the end of the process the matrix has been brought to the form shown in Figure 2. Each diagonal extension step involves $O(N)$ operations. Since each step extends the lower triangular part of the matrix by at least one more row, there can not be more than N such steps. Hence the approximate triangulation complexity is indeed $O(N^2)$.

In this paper we propose three probabilistic algorithms for selecting the αN reference variables. The first makes this selection once, before the iterative decoding (diagonal extension steps). The other two algorithms combine the (random) selection of the reference variables with the iterative decoding. All three algorithms require $O(N)$ operations for selecting the reference variables.

Once the triangulation is done, expressing all $(\delta - \alpha)N$ unknown bits as affine combinations of reference bits proceeds

recursively from the leftmost unknown bit to the rightmost bit in Figure 2. When we start the l'th step of this procedure, the first (from the left) $l-1$ unknown bits are already expressed as affine combinations of αN reference bits. Now, since the matrix is sparse, the l'th unknown bit is determined by the sum of a fixed number of affine combinations that are already known at the l'th step of the recursion. The l'th step consumes $O(N)$ operations. Since there are $O(N)$ unknown bits, there are $O(N)$ recursion steps and their overall complexity is also $O(N^2)$.

So far we have expressed the N code bits as linear combinations of αN reference variables. Now, the L original homogeneous *sparse* parity check equations (5) translate into $L-(\delta-\alpha)N$ non-homogeneous non-sparse equations in the reference variables, simply by substituting each original variable by the corresponding combination of reference variables (with a free element). Hence we have obtained a consistent system of $L-(\delta-\alpha)N$ equations and αN variables. Since the original equations are sparse, the complexity of expressing the equations in terms of the reference variables is also $O(N^2)$.

Expressing the complexity of solving this system again as $\beta M_2^2[L-(\delta-\alpha)N]+\gamma M_2^3$, where $M_2=\alpha N$ is the number of variables, and L is the number of equations, we see that the complexity has now been reduced to

$$\left((1-R-\delta+\alpha)\beta+\gamma\alpha\right)\alpha^2N^3 \tag{7}$$

The final stage comprises substituting the reference variables in the original variables, which takes $O(N^2)$ operations.

Comparing (6) and (7) we see that the complexity has reduced by a factor of at least $(\delta/\alpha)^2$. Hence, from complexity considerations, α should be as small as possible. The problem is how to choose as few reference variables as possible, while still successfully completing the triangulation process.

We now present three methods for choosing the reference variables and calculate the typical α in each case.

A. Method A

According to this method the reference variables are chosen at random from the $N(1+o(1))\delta$ unknown (erased) bits.

The maximum proportion of unknown bits which still guarantees that the triangulation process does not terminate prematurely with probability $1-o(1)$ is just the threshold probability $\delta^*(\boldsymbol{\lambda},\boldsymbol{\rho})$ since in this case the corresponding iterative decoding algorithm is successful with probability $1-o(1)$. We thus need α to satisfy

$$\delta-\alpha<\delta^*(\boldsymbol{\lambda},\boldsymbol{\rho})$$

Let α_A be the smallest possible value of α which guarantees a successful decoding using decoding method A (with probability $1-o(1)$). In other words, α_A is the difference between the actual erasure probability (which we assume is decodable using ML decoding) and the (smaller) erasure probability that would enable a successful iterative decoding, i.e.

$$\alpha_A=\delta-\delta^*(\boldsymbol{\lambda},\boldsymbol{\rho})$$

B. Method B

Instead of choosing the reference variables in advance, we can randomly choose an ϵ-fraction from the remaining unknown variables each time the triangulation process terminates with the diagonal not reaching the right edge of the matrix (as in Figure 2). This procedure is repeated until finally the approximate triangular form is obtained. We are concerned with the limit as $\epsilon\to 0^+$.

We now claim that α_B, the fraction of reference variables at the end of this process satisfies $\alpha_B\le\alpha_A$ (with probability $1-o(1)$). This is because in Method B the new reference variables are chosen at random each time the triangulation terminates (prematurely). Had we permitted also using variables which were already "diagonalized", that is, variables already expressed as linear combinations of reference variables, we would be at exactly the same setting as in method A. Since choosing diagonalized variables as reference variables does not assist the triangulation process, we see that indeed $\alpha_B\le\alpha_A$.

The calculation of α_B proceeds in two stages. In the first stage we investigate the properties of the residual graph which remains after performing an iterative decoding using the $(1-\delta)N$ bits revealed by the channel. We then use this degree distribution as a starting point to the second stage, in which we calculate the number of reference variables required to finish off the work. For reasons of convenience we now use the bipartite graph terminology and not the matrix one.

We begin by calculating the residual graph degree distribution at the end of the first stage above. This can be done by first solving for the fixed point, p_*, of (3) (i.e. the value $p_l=p_{l-1}=p_*$), which is the erasure probability of rightbound messages when the first stage terminates. Now p_* can be used to determine the residual distribution of the check nodes. The residual distribution of the variable nodes can be determined by first translating p_* to the erasure probability of leftbound messages when the first stage terminates. However, we prefer to calculate the residual distribution by a different approach, since the same equations will be used to analyze the second stage.

Note that the residual graph distribution is independent of the transmitted codeword. Thus without loss of generality we can assume that the all-zero codeword was transmitted. Furthermore, it is easy to see that the residual distribution of the following procedure (B1) yields the sought-for residual distribution. In this procedure each left node in the graph has one of three possible states: unknown, revealed and decoded where unknown corresponds to erasure, and by the all-zero transmitted codeword assumption, the value of all revealed and decoded bits is actually zero.

Procedure B1

1) [Initialization] Declare all (variable) nodes *unknown*.
2) [Reveal] Declare each node which is not *revealed* to be *revealed* with probability ϵ.
3) [Decode] Perform iterative decoding on the graph. Any decoded node (which was *unknown*) is declared *decoded*. At the end of this step, any *edge* emanating from a variable node that is not *unknown* is deleted.
4) [Finish] If no *unknown* nodes remain or if the proportion

of the *revealed* nodes is greater than $1-\delta$ then terminate. Otherwise go to 2.

When ϵ is sufficiently small, Procedure B1 is identical to an iterative decoding of the output of a BEC with parameter δ, in the sense that the remaining unknown subgraph will have exactly the same statistical properties. In the new formulation we perform numerous infinitesimal contributions which are susceptible to analysis via the differential equation approach presented in [13], [18].

Denoting the proportion of nodes that are *not revealed* by Δ, we see that Δ is initially 1, and it evolves as:

$$\Delta \leftarrow (1-\epsilon)\Delta + o(1) \tag{8}$$

where $o(1) \to 0$ as $N \to \infty$ by the law of large numbers. The proportion of erasures among messages going right *on the residual graph* converges to

$$1 - K\epsilon + O(\epsilon^2) + o(1) \tag{9}$$

To find K we substitute (9) in (3) (with p_l and p_{l-1} substituted by (9) and with $\delta = 1-\epsilon$), to obtain

$$\begin{aligned} & 1 - K\epsilon + O(\epsilon^2) + o(1) \\ &= (1-\epsilon)\lambda\left(1-\rho\left(K\epsilon + O(\epsilon^2) + o(1)\right)\right) \\ &= (1-\epsilon)\lambda\left(1-\rho_2 K\epsilon + O(\epsilon^2) + o(1)\right) \\ &= (1-\epsilon)\left[\lambda(1) - \lambda'(1)\rho_2 K\epsilon + O(\epsilon^2) + o(1)\right] \end{aligned}$$

In the sequel we neglect to write the $o(1)$ term. Using $\lambda(1) = 1$ and equating the coefficients of ϵ we get

$$K = \frac{1}{1-\rho_2\lambda'(1)} \tag{10}$$

Denoting the proportion of degree i right nodes by $\tilde{\rho}_i$, then on the one hand (2) holds, and on the other hand for any $i \geq 2$, the new value of $\tilde{\rho}_i$ at the end of the step is given by

$$\tilde{\rho}_i \leftarrow \frac{\nu_i}{\sum_{j\geq 2}\nu_j} \tag{11}$$

where

$$\begin{aligned} \nu_i &= \sum_{j\geq i}\tilde{\rho}_j\binom{j}{i}(1-K\epsilon)^i(K\epsilon)^{j-i} \\ &= \tilde{\rho}_i\binom{i}{i}(1-K\epsilon)^i + \tilde{\rho}_{i+1}\binom{i+1}{i}(1-K\epsilon)^i(K\epsilon) \\ &\quad + O(\epsilon^2) \\ &= (1-iK\epsilon)\tilde{\rho}_i + (i+1)K\epsilon\tilde{\rho}_{i+1} + O(\epsilon^2) \end{aligned} \tag{12}$$

To calculate the evolution of the left degrees we must first find the erasure probability among messages going left. Using (9) we have, for this probability

$$\begin{aligned} \text{erasure probability going left} &= 1-\rho\left(K\epsilon + O(\epsilon^2)\right) \\ &= 1-\rho_2 K\epsilon + O(\epsilon^2) \end{aligned} \tag{13}$$

Denoting the proportion of degree i left nodes by $\tilde{\lambda}_i$, recall that (1) holds. Now, a degree i left node survives a single step of the procedure if at the end of the step it receives only erasure messages, both from the channel and from its neighbor right nodes (this means, in particular, that a left-regular graph will always stay this way). Thus using (13) for $i \geq 2$ we have

$$\tilde{\lambda}_i \leftarrow \frac{\mu_i}{\sum_{j\geq 2}\mu_j} \tag{14}$$

where

$$\begin{aligned} \mu_i &= \tilde{\lambda}_i(1-\epsilon)\left(1-\rho_2 K\epsilon\right)^i + O(\epsilon^2) \\ &= \tilde{\lambda}_i\left(1-\epsilon\left(1+i\rho_2 K\right)\right) + O(\epsilon^2) \end{aligned} \tag{15}$$

We also calculate the evolution of Γ, the proportion of *unknown* variables, which is initially 1. A variable that was *unknown* at the beginning of the step stays so only if it was not declared as *revealed* (with probability $(1-\epsilon)$) and if it receives only erasure messages. Using the probability of erasures going left (13), we thus have

$$\Gamma \leftarrow \Gamma\,(1-\epsilon)\sum_{i\geq 2}\tilde{\lambda}_i(1-\rho_2 K\epsilon)^i + O(\epsilon^2)$$

A little algebra gives us

$$\Gamma \leftarrow \Gamma\left(1-\epsilon-\rho_2 K\epsilon\sum_{i\geq 2} i\tilde{\lambda}_i + O(\epsilon^2)\right) \tag{16}$$

If we express the relevant quantities as a function of a time variable t, we have, using (11), (12), (14) and (15),

$$\frac{d\tilde{\rho}_i(t)}{dt} = K(t)\left[(i+1)\tilde{\rho}_{i+1}(t) - i\tilde{\rho}_i(t) + \tilde{\rho}_i(t)2\tilde{\rho}_2(t)\right] \tag{17}$$

and

$$\frac{d\tilde{\lambda}_i(t)}{dt} = \rho_2(t)K(t)\tilde{\lambda}_i(t)\left(\sum_{j\geq 2} j\tilde{\lambda}_j(t) - i\right) \tag{18}$$

where K is given by (10), and where $\tilde{\boldsymbol{\lambda}}$ and $\boldsymbol{\lambda}$ (or $\tilde{\boldsymbol{\rho}}$ and $\boldsymbol{\rho}$) are related through (1) (or (2)). Similarly, from (16) and (8) we have

$$\frac{d\Gamma(t)}{dt} = -\Gamma(t)\left(1+\rho_2(t)K(t)\sum_{i\geq 2} i\tilde{\lambda}_i(t)\right) \tag{19}$$

and

$$\frac{d\Delta(t)}{dt} = -\Delta(t) \tag{20}$$

Using the boundary condition $\Delta(0) = 1$, the solution to (20) is:

$$\Delta(t) = e^{-t} \tag{21}$$

Recall that the stopping condition for Procedure B1 is $\Delta(t) = \delta$. Hence, the proportion of unknown nodes in the graph and the degree distributions of the residual graph remaining after an application of Gallager's belief propagation to the messages arriving from the channel are given by the solution to the system (17), (18) and (19) at $t = -\ln\delta$. Of course, in practice, one may utilize (8), (11), (14) and (16) directly to numerically solve the coupled differential equations.

Note that we have implicitly assumed that $\rho_2\lambda'(1) < 1$ (e.g., see (10)). Suppose that this condition holds initially. In this case it will hold as long as $\Gamma > 0$, since $\rho_2\lambda'(1) \to 1$ implies $K \to \infty$. The claim now follows by (19). If initially $\rho_2\lambda'(1) > 1$ then (9) does not hold since in this case there is a linear

size connected component of degree-two right nodes. When one of this nodes is revealed the entire connected component collapses with it and thus an ϵ fraction of revealed nodes leads to a linear number of nodes revealed by the iterative algorithm. In order to obtain the resulting degree distribution, one should first obtain the erasure probability at equilibrium, by solving for the fixed point of (3).

Before we go on to calculate α_B, suppose for a moment that we continued applying Procedure A not until $\Delta(t) = \delta$, but until $\Gamma(t) = 0$. Let us denote the corresponding termination time by τ. Then by the definition of $\Gamma(\cdot)$, $\Delta(\cdot)$ and the threshold probability δ^* (see Section II-B) we have

$$\Gamma(\tau) = 0 \tag{22}$$

and

$$\Delta(\tau) = \delta^* \qquad \text{(in the above-described context)} \tag{23}$$

From (21), (22) and (23) we have

$$\Gamma\left(-\ln \delta^*\right) = 0 \tag{24}$$

We are now ready for the second stage of the calculation of α_B. Consider applying the following procedure to the result of Procedure B1 (with the states of the variable nodes preserved from the end of the previous stage, hence there need not be an initialization part here).

Procedure B2

1) [Reveal] Declare each (variable) node which is *unknown* to be *revealed* with probability ϵ.
2) [Decode] Perform iterative decoding on the graph. Any decoded node (which was *unknown*) is declared *decoded* and at the end of this step, any edge emanating from a node that is not *unknown* is deleted.
3) [Finish] If no *unknown* nodes remain then terminate. Otherwise go to 1.

The main difference between Procedures B1 and B2 is that when declaring nodes to be *revealed*, in Procedure B2 we only consider *unknown* nodes, whereas in Procedure B1 we consider all nodes that are not *revealed*. The proportion of *revealed* nodes (=$1-\Delta$) at the end of this procedure is equal to $1-\delta+\alpha_B$ (there are $(1-\delta)N$ revealed nodes from the end of the previous stage, and new $\alpha_B N$ revealed nodes, which are the reference variables), i.e.

$$\Delta = \delta - \alpha_B \tag{25}$$

Since declaring a *decoded* node as *revealed* has no contribution to the iterative decoding (it does not affect the decoded nodes), the evolution of $\boldsymbol{\rho}$, $\boldsymbol{\lambda}$ and Γ, as expressed by (11) , (14) and (16) does not alter upon switching from Procedure B1 to B2. On the other hand, (8) becomes:

$$\Delta \leftarrow \Delta - \epsilon\Gamma + O(\epsilon^2)$$

which yields the following differential equation:

$$\frac{d\Delta(t)}{dt} = -\Gamma(t) \qquad , \qquad t > -\ln\delta \tag{26}$$

Thus, for $0 < t < -\ln\delta$ we use (17) – (20), and for $t > -\ln\delta$ we use (17) – (19) and (26). Since the time evolution of $\Gamma(t)$ remains in Procedure B2 as in Procedure B1, (24) remains valid. By (24) and (25),

$$\alpha_B = \delta - \Delta(-\ln\delta^*) \tag{27}$$

(recall that $\Delta(t)$ is the value of Δ at time t).

Now, α_B is a function of δ (note that $\Delta(\cdot)$ is implicitly dependent on δ through the stopping criterion of Procedure B1). The following claim lists some of its interesting properties.

Claim 1: The function $\alpha_B(\delta)$ possesses the following properties:

1) $\alpha_B(\delta^*) = \alpha_B'(\delta^*) = \alpha_B''(\delta^*) = 0$.
2) It is convex $\cup$.

Proof: From (21) we have

$$\Delta(-\ln\delta) = \delta \tag{28}$$

Equations (26), (27) and (28) yield

$$\begin{aligned}\alpha_B(\delta) &= \delta - \Delta(-\ln\delta^*) \\ &= \int_{-\ln\delta}^{-\ln\delta^*} \Gamma(t)dt\end{aligned}$$

Thus, $\alpha_B(\delta^*) = 0$. Differentiating with respect to δ we have

$$\frac{d\alpha_B(\delta)}{d\delta} = \frac{\Gamma(-\ln\delta)}{\delta} \tag{29}$$

From (29) and (24) we have $\alpha_B'(\delta^*) = 0$. A second differentiation gives

$$\begin{aligned}\frac{d^2\alpha_B(\delta)}{d\delta^2} &= -\frac{\Gamma(-\ln\delta) + \Gamma'(-\ln\delta)}{\delta^2} \\ &= \frac{1}{\delta^2}K(-\ln\delta)\Gamma(-\ln\delta)\rho_2(-\ln\delta) \times \\ &\quad \sum_{i\geq 2} i\tilde{\lambda}_i(-\ln\delta)\end{aligned} \tag{30}$$

where to derive (30) we have used (19). Since all terms in (30) are non-negative, we see that $\alpha_B(\delta)$ is indeed convex $\cup$. Furthermore, (30) and (24) imply $\alpha_B''(\delta^*) = 0$. $\square$

C. Method C

Instead of choosing the ϵ-fraction of new reference variables at random as in Method B, we may try choosing them in some more educated manner. In particular, we may choose the strategy proposed in [18] ("Greedy Algorithm C" there, p. 648) for efficient encoding of LDPC codes. In this method we first specify a vector $(\omega_2, \omega_3, ...)$ such that for any $i \geq 2$, $0 \leq \omega_i \leq 1$. Each time the triangulation process gets stuck we choose (from the residual "undiagonalized" submatrix) each row with probability $\epsilon\omega_i$, where i is the residual degree of that row and $\epsilon > 0$, as before, is arbitrarily small. For each chosen row (of degree i) we randomly select $i-1$ out of the i columns corresponding to the '1' components of the chosen row. Note that this is not a generalization of Method B, in the sense that no choice of vector $\boldsymbol{\omega}$ in Method C coincides with Method B.

Denote by α_C the fraction of reference variables needed to complete the approximate triangulation procedure for some values of δ and $\boldsymbol{\omega}$. As in the case of Method B, we divide the calculation of α_C to two stages. The first stage is identical

to the first stage in the analysis of Method B (procedure B1), and results in the degree distribution of the nodes in the bipartite graph remaining after applying iterative decoding to the channel output, as well as the fraction of remaining unknown left-vertices. To analyze the second part of the procedure, we first present it formally:

Procedure C2

1) [Reveal] Choose each right node of degree $i \geq 2$ with probability $\epsilon\omega_i$. If a right node has been chosen, randomly declare $i-1$ of its i left neighbors to be *revealed*.
2) [Decode] Perform iterative decoding on the graph. Any decoded variable node (which was *unknown*) is declared *decoded* and at the end of this step, any edge emanating from a variable node that is not *unknown* is deleted.
3) [Finish] If no *unknown* variable nodes remain then terminate. Otherwise go to 1.

This algorithm was analyzed in [18]. The right degrees, ρ_i, are updated by using

$$\rho_i \leftarrow \rho_i \left[1+\epsilon\left(\sum_j \rho_j\omega_j - \omega_i\right)\right] + \left[i\rho_{i+1} - (i-1-\rho_2)\,\rho_i\right]\frac{\lambda'(1)\sum_j \rho_j\omega_j}{1-\lambda'(1)\rho_2}\epsilon + O(\epsilon^2)$$

for $i \geq 2$. The left degrees, λ_i, are updated by using

$$\lambda_i \leftarrow \lambda_i\left[1+\frac{\left(\sum_j \rho_j\omega_j\right)\left(\sum_j j\lambda_j - i\right)}{1-\rho_2\lambda'(1)}\epsilon + O(\epsilon^2)\right]$$

for $i \geq 2$. The update formula for the proportion of nodes that are *unknown* is

$$\Gamma \leftarrow \Gamma\left(1-\epsilon\frac{\sum_j \rho_j\omega_j}{[1-\rho_2\lambda'(1)]\sum_j \lambda_j/j}\right) + O(\epsilon^2)$$

The update formula for the proportion of nodes that are *not revealed* is

$$\Delta \leftarrow \Delta - \epsilon\Gamma\frac{\sum_j \rho_j\omega_j(j-1)/j}{\sum_j \lambda_j/j} + O(\epsilon^2)$$

Just as for Procedure B, we have here $\alpha_C = \delta - \Delta(\tau)$, where τ is defined by $\Gamma(\tau) = 0$.

V. Examples

In this section we compare the performance of the three methods for two ensembles, the $(3,6)$ regular ensemble and the $(3,5)$ regular ensemble.

Figures 3 and 4 show α_A, α_B and α_C as a function of δ for the $(3,6)$ and the $(3,5)$ regular ensembles respectively. For method C, the vector $\boldsymbol{\omega}$ which was chosen is $\omega_2 = 1$ and the other components some very small positive numbers.

For the $(3,6)$ regular code $\delta^*(x^2, x^5) = 0.429$. Hence, working over a channel with erasure probability $\delta = 0.47$ we see that we need to choose $\alpha_A = 0.041$. The same value of δ gives $\alpha_B = 0.0278$ and $\alpha_C = 0.0236$. Hence, method A is at least $(\delta/\alpha_A)^2 = 131$ times more efficient than the naive approach, method B is at least 286 times more efficient and method C beats the naive approach by a factor of more than 397 in this case.

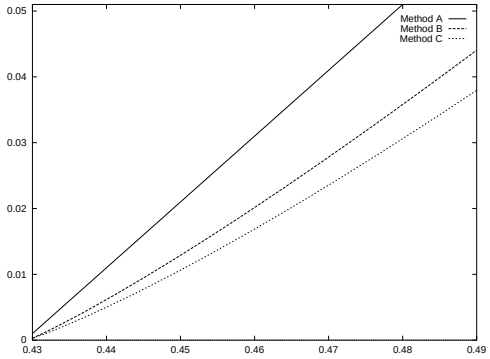

Fig. 3. α as a function of δ for the $(3,6)$ regular ensemble using methods A, B and C.

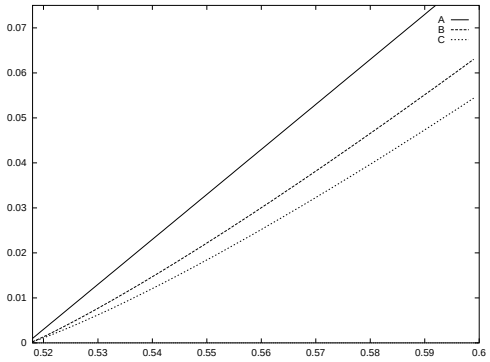

Fig. 4. α as a function of δ for the $(3,5)$ regular ensemble using methods A, B and C.

Note that Figures 3 and 4 can also be used to determine the threshold erasure probability of the channel as a function of the computational complexity of the decoding algorithm. This is useful when the computational complexity of the decoding algorithm is fixed beforehand. Consider for example the $(3,6)$ regular code and suppose that method C is used. Then the threshold erasure probability of an algorithm that permits a fraction of up to $\alpha = 0.01$ reference variables is 0.45, while the threshold erasure probability of the standard iterative decoding algorithm (which corresponds to $\alpha = 0$) is 0.429.

VI. Conclusion

We presented simple practical probabilistic algorithms for efficient ML decoding of LDPC codes over the BEC and analyzed their computational complexity. These algorithms can be viewed as a generalization of the standard iterative decoding algorithm, since standard iterative decoding corresponds to $\alpha = 0$ (i.e. no reference variables are used). In fact the user can adjust the value of α so as to determine the desirable tradeoff between the performance improvement and the increase in computational complexity compared to plain iterative decoding.

As a topic for further research we would like to note the comparison of our algorithms to plain iterative decoding and to the advanced methods for solving sparse linear equations over finite fields in practical scenarios of decoding LDPC codes

over the BEC (similar to the comparison in [10] between methods for solving sparse linear equations in the context of integer factoring and discrete logarithm computations).

REFERENCES

[1] L. Bazzi, T. Richardson and R. Urbanke, "Exact thresholds and optimal codes for the binary symmetric channel and Gallager's decoding algorithm A", submitted for publication, *IEEE Trans. Inform. Theory*, available at http://cm.bell-labs.com/cm/ms/who/tjr/pub.html.

[2] D. Burshtein and G. Miller, "Bounds on the Performance of Belief Propagation Decoding", *IEEE Trans. Inform. Theory*, vol. 48, pp. 112–122, January 2002.

[3] D. Coppersmith, "Solving linear equations over GF(2): block Lanczos algorithm," *Linear Algebra Applications*, vol. 192, pp. 33–60, 1993.

[4] D. Coppersmith, "Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm," *Mathematics of Computation*, vol. 62, no. 205, pp. 333–350, January 1994.

[5] D. Coppersmith, A. Odlyzko and R. Schroeppel, "Discrete logarithms in GF(p)," *Algorithmica*, vol. 1, pp. 1–15, 1986.

[6] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," in *Proc. 19'th ACM Symp. Theory Comp.*, pp. 1-6, 1987.

[7] P. Elias, "Coding for two noisy channels," in *Information Theory, 3rd London Symposium*, pp. 61–76, 1955.

[8] R. G. Gallager, *Low Density Parity Check Codes*, M.I.T Press, Cambridge, Massachusetts, 1963.

[9] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Res. Nat. Bureau of Standards*, vol. 49, pp. 409–436, 1952.

[10] B. A. LaMacchia and A. M. Odlyzko, "Solving large sparse linear systems over finite fields", in *Advances in Cryptology: CRYPTO '90*, A. Menezes, S. Vanstone, eds., *Lecture Notes in Computer Science*, vol. 537, pp. 109–133, Springer-Verlag, 1991.

[11] C. Lanczos, "Solution of systems of linear equations by minimized iterations," *J. Res. Nat. Bureau of Standards*, vol. 49, pp. 33–53, 1952.

[12] M. Luby, M. Mitzenmacher and M. A. Shokrollahi, "Analysis of random processes via and-or tree evaluation," in *Proc. of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 364–373, 1998.

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

[14] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inform. Theory*, vol. 47, pp. 585–598, February 2001.

[15] P. L. Montgomery, "A block Lanczos algorithm for finding dependencies over GF(2)," in *Advances in Cryptology: EUROCRYPT '95*, L. C. Guillou and J. J. Quisquater, eds., *Lecture Notes in Computer Science*, vol. 921, pp. 106–120, Springer-Verlag, 1995.

[16] A. M. Odlyzko, "Discrete logarithms in finite fields and their cryptographic significance," in *Advances in Cryptology: Proceedings of Eurocrypt '84*, T. Beth, N. Cot, I. Ingemarsson, eds., *Lecture Notes in Computer Science*, vol. 209, pp. 224–314, Springer-Verlag, 1985.

[17] A. Odlyzko, "Discrete logarithms: the past and the future", *Designs Codes and Cryptography*, vol. 19, no. 2-3, pp. 129–145, March 2000.

[18] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 638–656, February 2001.

[19] V. Strassen, "Gaussian elimination is not optimal," *Numerische Math.*, vol. 13, pp. 354–356, 1969.

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

[21] J. Teitelbaum, "Euclid's algorithm and the Lanczos method over finite fields", *Mathematics of Computation*, vol. 67, no. 224, pp. 1665–1678, October 1998.

[22] D. Wiedemann, "Solving sparse linear equations over finite fields," *IEEE Trans. Inform. Theory*, vol. IT-32, no. 1, pp. 54–62, January 1986.

David Burshtein (M'92 – SM'99) received the B.Sc. and Ph.D. degrees in Electrical Engineering in 1982 and 1987, respectively from Tel-Aviv University.

During 1988-1989 he was a research staff member in the speech recognition group of IBM, T. J. Watson Research Center. In 1989 he joined the School of Electrical Engineering, Tel-Aviv University, where he is presently associate professor. His research interests include information theory, codes on graphs and iterative decoding algorithms and signal processing.

Gadi Miller received the B.Sc. degree in Mathematics and Physics, the M.Sc. degree in Physics and the Ph.D. degree in Electrical Engineering, all from Tel-Aviv university, Tel-Aviv, Israel, in 1995, 1997 and 2002 respectively. He has done his post-doctorate at the E.N.S.T., Paris in 2002-2003. He is currently with Algotec systems.