A Finite-Length Algorithm for LDPC Codes Without Repeated Edges on the Binary Erasure Channel

Sarah J. Johnson, *Member IEEE*

***Abstract*—This paper considers the performance, on the binary erasure channel, of low-density parity-check (LDPC) codes without repeated edges in their Tanner graphs. A modification to existing finite-length analysis algorithms is presented for these codes.**

***Index Terms*—binary erasure channels, finite-length analysis, iterative decoding, low-density parity-check codes, message-passing decoding.**

[1]

I. Introduction

A low-density parity-check (LDPC) code is a block code defined by a sparse parity-check matrix, H. LDPC codes are typically decoded iteratively with message-passing decoding [1]. The *Tanner graph*, T, of an LDPC code is a bi-partite graph which represents the parity-check matrix. Each bit in the codeword corresponds to a column of H, and a *bit vertex* of T, and each parity-check equation corresponds to a row of H, and a *check vertex* of T. The (j, i)-th entry of H is '1', and an edge connects the i-th bit vertex and j-th check vertex of T, if the i-th codeword bit is included in the j-th parity-check equation. The LDPC code is (l, r)-regular if every bit vertex is degree l and every check vertex is degree r. An irregular LDPC code has vertices with varying degrees, given by its *degree distribution*.

For a given degree distribution, an LDPC code can be defined by the permutation, Π, of Tanner graph edges between the check and bit vertices. An LDPC *ensemble* is the set of all such permutations for a particular degree distribution. For a given degree distribution, an *expurgated* ensemble is the subset of all edge permutations which also satisfy a particular constraint, such as not allowing certain edge configurations.

Using the concept of ensembles, the performance of infinite-length LDPC codes has been well studied and infinite-length irregular LDPC ensembles have been shown to be capacity-approaching [2]. In the special case of the binary erasure channel (BEC), irregular LDPC codes are in fact capacity-achieving [3]. Using the BEC, some progress has also been made on understanding the performance of finite-length (FL) LDPC codes. Using a combinatorial characterization of decoding failures, expressions for the exact average bit and block erasure probabilities are derived in [4]–[6] for various LDPC ensembles, when decoded iteratively with message-passing decoding. This finite-length analysis provides both a method to predict the average performance of the codes in an LDPC ensemble, without the need for time-consuming simulations, and a context for better understanding the performance of message-passing decoding of finite-length LDPC codes.

On the binary erasure channel a transmitted symbol is either received correctly or completely erased, the latter with probability ϵ. If only one of the bits in any given code parity-check equation is erased, the erased bit can be determined exactly by choosing the value which satisfies that parity-check equation. Message-passing iterative decoding of an LDPC code, transmitted over a binary erasure channel, is a process of finding parity-check equations which check on only one erased bit. In a decode iteration all such parity-check equations are found and the erased bits corrected. Due to the correction of these bits, new parity-check equations checking on only one erased bit may be created and these are then used to correct further erased bits in the subsequent iteration.

A *stopping set*, $\mathcal{S}$, is a subset of bit vertices in T for which every check vertex connected by a graph edge to a bit vertex in $\mathcal{S}$ is connected to at least one other bit vertex in $\mathcal{S}$. A stopping set is thus a set of codeword bits with the property that every parity-check equation checking on a bit in the stopping set checks on at least two bits in the set. Consequently, if all of the bits in a stopping set are erased, the message-passing decoder will be unable to correct any of the erased stopping set bits. Indeed, the collection of stopping sets in an LDPC Tanner graph determines exactly the erasure patterns for which the message-passing decoding algorithm will fail [4]. A stopping set containing v bit vertices is said to be a *size-v* stopping set.

In [4] the FL performance of regular LDPC ensembles is considered. This is extended in [6] to allow for irregular bit degrees and in [5] to allow for expurgated ensembles. The finite-length ensembles considered in [4]–[6], which we will call *traditional ensembles*, include graphs with repeated edges. That is, permutations are allowed which include more than one edge between the same two bit and check vertices. Since repeated edges cannot be represented in a binary parity-check matrix, LDPC codes are not constructed with repeated edges in practice. As a consequence, the performance predicted by traditional finite-length analysis algorithms is more pessimistic than the average performance of LDPC codes obtained, for example, by Monte Carlo simulation of randomly constructed LDPC codes with the required degree distribution.

[1]This work is supported by the Australian Research Council under grants DP0449627 and DP0665742. S. Johnson is with the School of Electrical Engineering and Computer Science, The University of Newcastle, Callaghan 2308, NSW Australia (email: sarah.johnson@newcastle.edu.au) An earlier version of this work was presented in part at the 2005 Asia Pacific Conference on Communications (APCC), Perth, Australia, October 2005.

In this paper a modified finite-length analysis is presented for new LDPC ensembles, defined as the set of all permutations, Π, except those which include repeated edges. The finite-length analysis algorithm presented in this paper builds upon the approach in [5, Section 3.2] and so this algorithm is outlined in Section II, with a small initialization modification presented in Section II-A. Section III then details the proposed FL algorithm and compares its performance to Monte Carlo simulation results.

II. Traditional Finite-length analysis

The ensembles considered in [5, Section 3.2], and in the remainder of this section, are the set of regular Tanner graphs with M check vertices of degree at most r, and N bit vertices of degree l. We will say that the check vertices have r 'sockets' available for graph edges to connect to. As discussed above, this ensemble definition allows a single bit vertex to be connected to a single check vertex by more than one edge.

In general, FL analysis calculates the word erasure probability for an LDPC ensemble, given that v bits are erased, by calculating the total number of constellations on a fixed set of v bit vertices, $T(v)$, and the number of those constellations which contain stopping sets, $B(v)$. Then $\frac{B(v)}{T(v)}$ gives the conditional probability of unsuccessful decoding given that v erasures occur.

In [5, 3.2] $T(v)$ and $B(v)$ are calculated combinatorially with $B(v)$ found by iteratively building up graphs one bit vertex at a time and counting the total number of ways that this can be done. We outline this process here as it forms the basis for our modifications in the following sections.

Firstly, a v-bit subgraph, $\mathcal{T}$, of a Tanner graph contains v bit vertices, all of the vl edges emanating from them, and all of the check vertices connected to one or more of these edges. The total number, $T(v)$, of all possible subgraphs on a fixed set of v bit vertices is given by [5]:

$$T(v) = (vl)!\binom{Mr}{vl}, \tag{1}$$

as there are in total Mr available sockets to choose for the vl graph edges, which can be done in $\binom{Mr}{vl}$ ways, and $(vl)!$ possible permutations of these edges.

The total number of possible subgraphs on a fixed set of v bit vertices which are a stopping set is given by [5]

$$coef(g(x,r,t),vl)(vl)!, \tag{2}$$

where

$$g(x,r,t) = ((1+x)^r - 1 - rx)^t,$$

and $\mathrm{coef}(g(x),i)$ denotes the coefficient of x^i in $g(x)$. The term $coef(g(x,r,t),vl)$ gives the number of ways that vl edges can be allocated to t check vertices with r sockets such that each check vertex receives two or more edges. The term $(vl)!$ counts all the possible permutations of these edges.

Erasing the v bits corresponding to a stopping set will cause the message passing decoder to fail, but so too will erasing v bits such that some subset of the v bits erases a stopping set. To incorporate these potential decoding failures, $B(v)$ is found by starting with the known stopping sets in (2) and iteratively building up larger subgraphs by adding one bit vertex at a time and counting the total number of ways that this can be done. To track stopping sets, we track the number of degree-1 check vertices in $\mathcal{T}$, denoted by s, and the number of degree ≥ 2 check vertices in $\mathcal{T}$, denoted by t. Using this notation, the subgraphs, $\mathcal{T}$, which are stopping sets are easily seen to be the subgraphs with $s = 0$. The term $A(v,t,s,X)$ is used to denote the number of stopping sets of size $\leq v$ within a subgraph $\mathcal{T}$ containing v unordered degree-l bit vertices, t check vertices with degree ≥ 2 and s degree-1 check vertices. $B(v)$ is then found by summing the contribution for a given v over every possible s and t.

$A(v,t,s,X)$ is initialized by the total number of possible stopping sets with size exactly v, from (2) [5]:

$$A(v,t,0,X) = coef(g(x,r,t),vl)\frac{(vl)!}{v!(l!)^v}\left(\frac{v}{N}\right)^X, \tag{3}$$

where the factor $\left(\frac{v}{N}\right)^X$ calculates the weighted number stopping sets, which is necessary if the bit error rate ($X = 1$), rather than the word error rate ($X = 0$), is required. For expurgated ensembles without stopping sets of size smaller than $S_{\min}$, or greater than $S_{\max}$, these stopping sets are discounted by setting $A(v,t,0,X) = 0$ if $v < S_{\min}$ or $v > S_{\max}$ [5].

Next, $A(v,t,s,X)$ can be calculated for all values of s and t, by examining every way an extra bit vertex can be added to each size $v-1$ subgraph and tracking the values for s and t for each of the newly formed size-v subgraphs.

When adding the new degree-l bit vertex, the l new edges can either: i) create Δs new degree-1 check vertices, increasing s by Δs; ii) add edges to σ of the existing degree-1 check vertices, decreasing s by σ and increasing t by σ (called *covering* the degree-1 check vertices); iii) add Δt new degree-2 check vertices, increasing t by Δt; iv) add τ edges to cover free slots in the existing check vertices with degree ≥ 2; v) add ϕ extra edges to free slots in the $\Delta t + \sigma$ newly created check vertices with degree ≥ 2; or some combination of the above where

$$l = \Delta s + \sigma + 2\Delta t + \tau + \phi.$$

For example, Fig. 1 shows a possible addition of bit vertices to an existing subgraph. Note that v, s, and t will denote the final number of each vertex type after the extra bit vertex has been added. The number of each vertex type in the original subgraph, to which a bit vertex is being added, are thus $v' = v-1$, $s' = s + \sigma - \Delta s$, and $t' = t - \Delta t - \sigma$.

If $\Delta s + \Delta t$ new check vertices are created there are $\binom{t+s}{\Delta t+\Delta s}\binom{\Delta t+\Delta s}{\Delta t}$ choices of location for them. For the Δs new degree-1 check vertices there are r ways to choose the check vertex socket which can be done in $r^{\Delta s}$ ways. Which of the $s + \sigma - \Delta s$ original degree-1 vertices to cover can be chosen in $\binom{s+\sigma-\Delta s}{\sigma}$ ways. Then the $\sigma + 2\Delta t + \phi = l - \Delta s - \tau$ edges allocated to the $\Delta t + \sigma$ new degree-2 vertices must be added so that each check vertex has at least two sockets filled. This can be done in

$$coef(f(x), l - \Delta s - \tau)$$

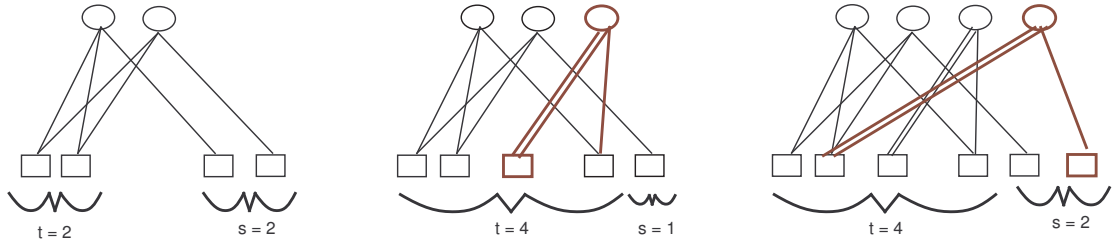

Fig. 1. Addition of two degree $l = 3$ vertices, in bold, to a subgraph which originally contained $v = 2$ bit vertices, $s = 2$ degree-1 check vertices, and $t = 2$ degree ≥ 2 check vertices. The three extra edges of the fist additional vertex are divided as $\sigma = 1$ and $\Delta t = 2$, to give a subgraph with $t = 4$ and $s = 1$. The three extra edges of the second additional vertex are divided as $\tau = 2$ and $\Delta s = 1$, to give a final subgraph with $t = 4$ and $s = 2$.

$$A(v,t,s,X) = \sum_{\Delta s=1}^{l} \sum_{\sigma=0}^{l-\Delta s} \sum_{\Delta t=0}^{\lfloor (l-\Delta s-\sigma)/2 \rfloor} \sum_{\tau=0}^{l-\Delta s-\sigma-2\Delta t} A(v-1, t-\Delta t-\sigma, s+\sigma-\Delta s, X) \binom{t+s}{\Delta t+\Delta s} \binom{\Delta t+\Delta s}{\Delta t} coef(f(x), l-\Delta s-\tau) \binom{r(t-\Delta t-\sigma)-\omega}{\tau} \binom{s+\sigma-\Delta s}{\sigma} r^{\Delta s} \frac{\Delta s}{s} \quad (4)$$

where $\omega = (v-1)l - (s+\sigma-\Delta s)$ [5].

ways where

$$f(x) = ((1+x)^{r-1} - 1)^{\sigma}((1+x)^{r} - 1 - rx)^{\Delta t}.$$

Lastly, the extra τ edges are allocated to remaining slots in the original degree ≥ 2 check vertices. There are $(t - \Delta t - \sigma)r$ slots in the $t - \Delta t - \sigma$ original check vertices with degree ≥ 2, however the existing $v - 1$ bit vertices are using

$$\omega = (v-1)l - (s + \sigma - \Delta s)$$

of these slots. Thus the free slots can be chosen in

$$N_{\tau} = \binom{r(t-\Delta t-\sigma)-\omega}{\tau}$$

ways. Finally, assuming that an extension with parameter Δs is counted Δs times, the same constellation, with parameter s, can be constructed in s different ways giving the $\frac{\Delta s}{s}$ term [5].

Putting this altogether and recursively adding one bit vertex at a time in (4) calculates (for a given v, s, and t) the contribution of all the possible stopping set constellations including those stopping sets on some subset of the v bits.

Finally, $B(v, X)$ sums the contribution for a given v over every possible s and t. Thus $B(v, 0)$ denotes the number of stopping sets of size v or less, on a fixed set of v degree-l bit vertices [5]:

$$B(v,X) = v!(l!)^{v} \sum_{t=0}^{M} \sum_{s=0}^{M-t} \binom{M}{t+s} A(v,t,s,X). \quad (5)$$

The ensemble average bit, $P(\epsilon, X = 1)$, and word, $P(\epsilon, X = 0)$, erasure rates following decoding is the probability that an erasure of size e will occur multiplied by the probability that such an erasure will include a stopping set [5]:

$$P(\epsilon, X) = \sum_{e=0}^{N} \binom{N}{e} \epsilon^{e} (1-\epsilon)^{N-e} \frac{B(e,X)}{T(e)}. \quad (6)$$

A. A small initialization modification

The process described above may count subgraphs that, while not the expurgated stopping sets themselves, contain them as subgraphs. For example, initialization via (3) can expunge graphs of size $v < S_{\min}$ which are stopping sets. However, the expression (2) may include graphs of size $v \geq S_{\min}$ which are themselves stopping sets but which also contain as subgraphs stopping sets of size less than $S_{\min}$. In this first modification $A(v, t, s, X)$ and $T(v)$ are initialized recursively to avoid the inclusion of these expurgated stopping sets.

We define a new function $C(v, t, 0)$ in (7) to calculate the number of possible stopping sets on v bits recursively so that stopping sets on v bits which contain subgraphs which are stopping sets with size $v' < S_{\min}$ are not included. $C(v, t, s)$ is initialized with the single empty graph, $C(0, 0, 0) = 1$, and graphs are built up one bit vertex at a time, in a similar manner to (4), with the exception that subgraphs with both $s = 0$ and $v \leq S_{\min}$ or $\geq S_{\max}$, i.e. the expurgated stopping sets, are not included.

Then $C(v, t, 0)$ is used to initialize $A(v, t, s, X)$:

$$A(v,t,0,X) = \frac{C(v,t,0)}{v!} \left(\frac{v}{n}\right)^{X}. \quad (8)$$

Similarly, $T(v)$ counts the total number of ways a subgraph on v message vertices can be constructed as:

$$T(v) = v!(l!)^{v} \sum_{t=0}^{M} \sum_{s=0}^{M-t} \binom{M}{t+s} \frac{C(v,t,s)}{v!}. \quad (9)$$

Again $T(v)$ is calculated using $C(v, t, s)$ to avoid the inclusion of graphs, which, while not expurgated graphs themselves, contain them as subgraphs. Equations (4)-(6) are unchanged.

Initialization : $C(v,t,s) = 0$ except for $C(0,0,0) = 1$,

Then : $\forall\ v \in \{0,\dots N\},\ t \in \{0,\dots,M\}$ and $s \in \{0,\dots,M-t\}$: $s > 0$ and/or $S_{\min} \le v \le S_{\max}$

$$C(v,t,s) = \sum_{\Delta s=0}^{l} \sum_{\sigma=0}^{l-\Delta s} \sum_{\Delta t=0}^{\lfloor (l-\Delta s-\sigma)/2 \rfloor} \sum_{\tau=0}^{l-\Delta s-\sigma-2\Delta t} C(v-1, t-\Delta t-\sigma, s+\sigma-\Delta s) \binom{t+s}{\Delta t+\Delta s} \binom{\Delta t+\Delta s}{\Delta t} coef(f(x), l-\Delta s-\tau) \binom{r(t-\Delta t-\sigma)-\omega}{\tau} \binom{s+\sigma-\Delta s}{\sigma} r^{\Delta s} \quad (7)$$

III. Finite-length analysis for ensembles without repeated edges

The new ensembles considered in this section are the set of regular Tanner graphs with M check vertices of degree at most r, and N bit vertices of degree l, with the condition that a single bit vertex can be connected to a single check vertex by at most one edge. This ensemble definition more closely maps to the LDPC codes commonly used in practice, such as those from [7], which are never constructed with repeated edges.

The explicit addition of repeated edges in FL analysis can be avoided by setting Δt and ϕ to zero, however repeated edges may still be added if $r > 3$ when τ is greater than 1 (consider the second vertex added in Fig. 1). This is because the number of ways these τ edges can be added is counted by choosing τ of the free slots in the degree ≥ 2 check vertices even if two of the free slots are contained in the same vertex. Here upper and lower bounds on the number of ways these τ edges can be added, without allowing repeated edges, are derived.

The upper bound is derived by assuming that the existing edges are evenly distributed amongst the $t-\sigma$ check vertices and then counting the number of ways τ edges can be added to τ different check vertices. An uneven distribution of free slots in the check vertices will give fewer options for allocating the τ edges.

In the existing graph on $v-1$ bit vertices there are $(v-1)l$ edges, however, $s+\sigma-\Delta s$ of them are allocated to the $s+\sigma-\Delta s$ degree-1 check vertices and so there are

$$\omega = (v-1)l - s - \sigma + \Delta s$$

edges into the existing $t-\sigma$ degree ≥ 2, check vertices. Assuming an even distribution of the w edges into the existing $t-\sigma$ check vertices, each vertex has $(r-\frac{\omega}{t-\sigma})$ free slots. Which τ check vertices to cover use can be chosen in $\binom{t-\sigma}{\tau}$ ways and the slot to use for each check chosen in $(r-\frac{\omega}{t-\sigma})^{\tau}$ ways giving

$$N_\tau \le \binom{t-\sigma}{\tau} \left(r - \frac{\omega}{t-\sigma}\right)^{\tau}. \quad (10)$$

N_τ can be lower bounded by assuming the most uneven distribution of existing edges to the $t-\sigma$ check vertices. Given ω edges into $t-\sigma$ check vertices, and at least 2 edges into each check vertex, the maximum number, a, of check vertices which can be full are

$$a = \left\lfloor \frac{\omega - 2(t-\sigma)}{r-2} \right\rfloor.$$

The remaining check vertices have only 2 edges each, except for a single check vertex which has any extra remaining extra edges allocated to it to give

$$b = 2 + (\omega - 2(t-\sigma) - (r-2)a)$$

of its slots taken. Thus in total there are a full check vertices, one half full check vertex, and the remaining $t-\sigma-a-1$ check vertices that have only two slots taken.

Choosing τ of the $t-\sigma-a$ check vertices with only two slots taken, to add one edge to each, can be done in

$$\binom{t-\sigma-a-1}{\tau}$$

ways while choosing one of the free slots in each of the chosen vertices can be done in $(r-2)^{\tau}$ ways. However, one of the τ edges could also be allocated to the half empty vertex. In this case that slot can be chosen in $(r-b)$ ways and the remaining $\tau-1$ edges allocated to the degree-2 check vertices in

$$\binom{t-\sigma-a-1}{\tau-1}(r-2)^{\tau-1}$$

ways. Thus in total

$$N_\tau \ge \binom{t-\sigma-a-1}{\tau-1}(r-2)^{\tau-1}(r-b) + \binom{t-\sigma-a-1}{\tau}(r-2)^{\tau}. \quad (11)$$

The recursions for the new FL analysis are given in (12) and (13). The recursions in (12) and (13) are actually less computationally demanding than the original computations in (4) and (7) because there are half as many summations.

For an upper bound on $C(v,t,s)$, N_τ from (10), is substituted into (12) while for a lower bound on $C(v,t,s)$ N_τ from (11) is substituted into (12). Similarly, for an upper bound on $A(v,t,s,X)$, N_τ from (10), is substituted into (12) and (13) and while for a lower bound on $A(v,t,s,X)$ N_τ from (11) is substituted into (12) and (13). An upper bound on $P(\epsilon,X)$ in (6) is found by using the upper bound on N_τ when calculating $B(v)$ and the lower bound when calculating $T(v)$. Similarly a lower bound on $P(\epsilon,X)$ in (6) is found by using the lower bound on N_τ when calculating $B(v)$ and the upper bound when calculating $T(v)$. Applying this modification gives FL analysis which is both a more accurate predictor of

Initialization : $C(v,t,s) = 0$ except for $C(0,0,0) = 1$,

Then : $\forall\ v \in \{0,\dots N\},\ t \in \{0,\dots,M\}$ and $s \in \{0,\dots,M-t\}$: $s > 0$ and/or $S_{\min} \le v \le S_{\max}$

$$C(v,t,s) = \sum_{\Delta s=0}^{l} \sum_{\sigma=0}^{l-\Delta s} C(v-1, t-\sigma, s+\sigma-\Delta s) \binom{t+s}{\Delta s} r^{\Delta s} \binom{s+\sigma-\Delta s}{\sigma} (r-1)^{\sigma} N_{\tau} \quad (12)$$

$$A(v,t,0,X) = \frac{C(v,t,0)}{v!} \left(\frac{v}{n}\right)^{X} \quad \text{then } \forall\ v \in \{0,\dots N\},\ t \in \{0,\dots,M\} \text{ and } s \in \{1,\dots,M-t\}$$

$$A(v,t,s,X) = \sum_{\Delta s=1}^{l} \sum_{\sigma=0}^{l-\Delta s} A(v-1, t-\sigma, s+\sigma-\Delta s, X) \binom{t+s}{\Delta s} r^{\Delta s} \binom{s+\sigma-\Delta s}{\sigma} (r-1)^{\sigma} N_{\tau} \frac{\Delta s}{s} \quad (13)$$

where $\tau = l - \sigma - \Delta s$.

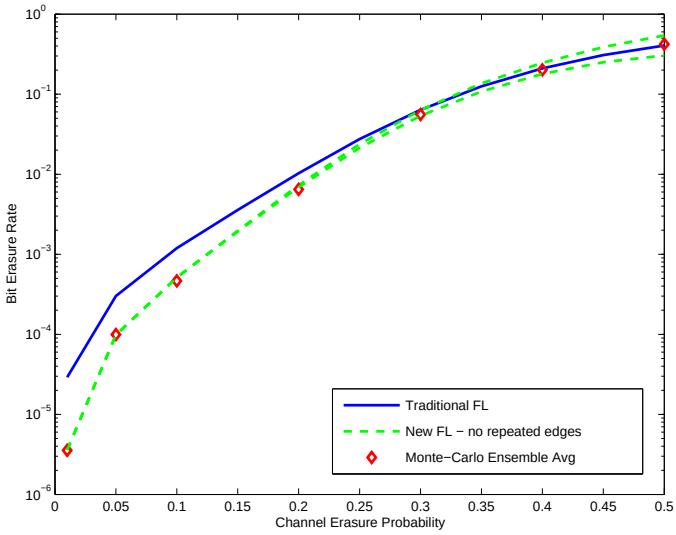

Fig. 2. The ensemble average erasure correction performance of (3,6)-regular, rate-$1/2$, length-32 LDPC codes on a binary erasure channel. Shown is the results of: traditional FL (solid curves); modified FL (dashed curves) and Monte-Carlo simulation (no curve).

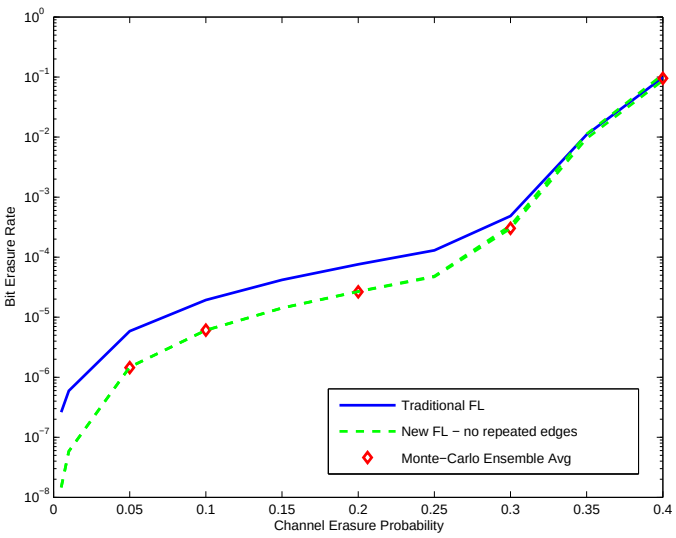

Fig. 3. The ensemble average erasure correction performance of (3,6)-regular, rate-$1/2$, length-200 LDPC codes on a binary erasure channel. Shown is the results of: traditional FL (solid curves); modified FL (dashed curves) and Monte-Carlo simulation (no curve).

the performance of typical LDPC codes and computationally simpler. This prediction improvement is a result of choosing an ensemble definition which more accurately models the LDPC codes used in practice, while the computation simplification is because fewer parameters are required for the modified FL analysis.

Figs. 2-4 show the ensemble average performance of finite-length, rate-$1/2$, LDPC code ensembles on a binary erasure channel. Both the upper and lower bounds for the new FL algorithm are plotted, however, the bounds are sufficiently tight that the difference between the two can only be seen at large channel erasure probabilities. Monte Carlo simulation results, averaged over randomly chosen codes from the ensemble, constructed using the method from [7], [8], are compared to the FL analysis results. The modified FL analysis algorithm presented in this paper accurately predicts the performance of regular LDPC ensembles.

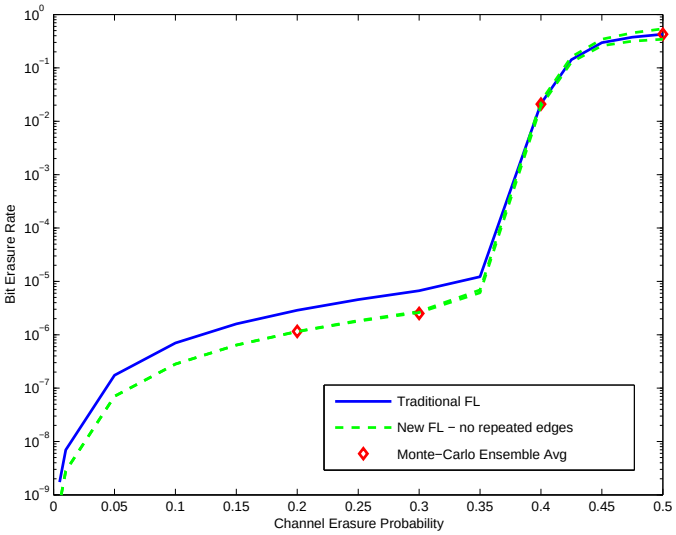

Fig. 4. The ensemble average erasure correction performance of (3,6)-regular, rate-$1/2$, length-1000 LDPC codes on a binary erasure channel. Shown is the results of: traditional FL (solid curves); modified FL (dashed curves) and Monte-Carlo simulation (no curve).

IV. Conclusion

In this paper we have presented a modified finite-length analysis for the performance of LDPC codes without repeated edges on the binary erasure channel. Comparison to Monte Carlo simulation results shows that the new finite-length analysis algorithms accurately predict the performance of regular LDPC ensembles on the binary erasure channel.

Acknowledgement

The author would like to thank Prof. Rudiger Urbanke for helpful comments on finite-length analysis of expurgated ensembles and for pointing out that the FL initialization process required modification. I am also very grateful to the Associate Editor, Prof. T. Richardson, for his many helpful suggestions.

References

[1] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, vol. IT-8, no. 1, pp. 21–28, January 1962.

[2] 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, no. 2, pp. 619–637, February 2001.

[3] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 569–584, February 2001.

[4] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inform. Theory*, vol. 48, no. 6, pp. 1570–1579, June 2002.

[5] T. J. Richardson and R. L. Urbanke, "Finite-length density evolution and the distribution of the number of iterations on the binary erasure channel," unpublished manuscript, available at ⟨http://lthcwww.epfl.ch/papers/RiU02.ps⟩.

[6] H. Zhang and A. Orlitsky, "Finite-length analysis of LDPC codes with large left degrees," in *Proc. International Symposium on Information Theory (ISIT'2002)*, Lausanne, Switzerland, June 30 - July 5 2002, p. p. 3.

[7] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 399–431, March 1999.

[8] R. M. Neal, ⟨www.cs.toronto.edu/∼radford/⟩.

Sarah Johnson received the B.E. degree in electrical engineering (with honours and university medal) in 2000, and PhD in 2004, both from the University of Newcastle, Australia. She then held a postdoctoral position with the Wireless Signal Processing Program, National ICT Australia. Currently she is with the University of Newcastle, Australia. Sarah's research interests are in the field of information theory and error correction, and in particular the area of low-density parity-check and repeat-accumulate codes.