Convergence Analysis of Serial Message-Passing Schedules for LDPC Decoding

Eran Sharon[1], Simon Litsyn[1], Jacob Goldberger[2]

[1] School of Electrical Engineering, Tel-Aviv University, Ramat Aviv 69978, Israel, {eransh,litsyn}@eng.tau.ac.il
[2] School of Engineering, Bar-Ilan University, Ramat Gan 52900, Israel, goldbej@eng.biu.ac.il

Abstract

Serial decoding schedules for low-density parity-check (LDPC) codes are described and analyzed. Conventionally, in each iteration all the variable nodes and subsequently all the check nodes send messages to their neighbors ("flooding schedule"). In contrast, in the considered methods, the updating of the nodes is implemented according to a serial schedule. The evolution of the decoding algorithm's computation tree under serial scheduling is analyzed. The analysis shows that it grows twice as fast in comparison to the flooding schedule's computation tree, indicating that the serial schedule propagates information twice as fast in the code's underlying graph. Furthermore, asymptotic analysis of the serial schedule's convergence rate is done using the Density Evolution (DE) algorithm. Applied to various ensembles of LDPC codes, it shows that when working near the ensemble's threshold, for long enough codes the serial schedule is expected to converge in half the number of iterations compared to the standard flooding schedule. This observation is generally proved for the Binary Erasure Channel (BEC) under some likely assumptions. Finally, an accompanying concentration theorem is proved, justifying the asymptotic DE analysis assumptions.

1 Introduction

Low-density parity-check (LDPC) codes, introduced by Gallager [4] in 1962, are linear error correcting codes defined by sparse parity-check matrices. The most attractive feature of LDPC codes is their ability to achieve a significant fraction of the channel capacity using iterative decoding with relatively low implementation complexity. In this work we analyze message-passing iterative decoding algorithms and in particular the Belief Propagation (BP) algorithm [11]. These algorithms are based on iterative message passing between variable and check nodes in the a bipartite graph representation of the code. The order of passing the messages between the nodes is referred to as updating rule or schedule. The standard message-passing schedule is the *flooding* schedule, where in each iteration all the variable nodes, and subsequently all the check nodes, pass new messages to their neighbors. It was pointed out by Forney [3]: "An open question is whether different schedules could improve iterative decoding algorithms". In this paper we provide an affirmative answer to this question by analyzing schedules which are better than the flooding schedule.

Some alternative message-passing schedules were considered in earlier research. Mao and Banihashemi [9] proposed a probabilistic variant of the flooding schedule, which takes into account the cycles in the code's graph. Zhang and Fossorier [14] and Kfir and Kanter [6] developed a decoding algorithm which is based on a serial schedule involving a sequential update of variable nodes' messages. Both papers provided convincing empirical results indicating a fast convergence of the serial schedule. However, these observations were not supported by a theoretical analysis. Mansour and Shanbhag [10] proposed a turbo decoding algorithm for Gallager codes which utilizes a dual decoding schedule to the one suggested in [14],[6]. Efficient implementations of LDPC decoding algorithm based on serial scheduling were suggested in [13] and [5].

In this work we provide a theoretical analysis confirming the fast convergence of the serial schedules. We show that the decoder's computation tree under serial scheduling grows twice as fast in comparison to the flooding schedule, indicating that the serial schedules propagate information on the code's graph twice as fast. A density evolution (DE) algorithm for asymptotic analysis of the serial schedule convergence rate is derived. We prove that for the Binary Erasure Channel (BEC), asymptotically in the code's length and when working near the decoder's threshold, the serial schedule is expected to converge in half the number of iterations compared to the standard flooding schedule. To complete the asymptotic analysis, an accompanying concentration theorem is proved.

2 Efficient Scheduling

An LDPC code can be defined by a sparse bipartite graph, constructed from its parity-check matrix. The graph consists of two types of nodes, variable and check nodes. We will denote the sets variable nodes, check nodes and edges in the graph by V,C and E respectively. A variable node, denoted by v, represents a bit in the transmitted codeword and a check node, denoted by c, represents a parity check constraint. Each check node is connected by an edge to the variable nodes it checks.

A regular (d_v, d_c)-LDPC code is defined by a regular bipartite graph, such that each variable node (check node) is connected to d_v (d_c) check nodes (variable nodes). The ensemble of regular (d_v, d_c)-LDPC codes of length n is denoted by $\mathcal{C}_n^{(d_v,d_c)}$. Regular LDPC codes can be

generalized to irregular LDPC codes based on irregular bipartite graphs which exhibit better performance under iterative decoding [8]. An ensemble of irregular LDPC codes is defined by the variable nodes' and check nodes' degree distributions: $\lambda(x) = \sum_{i=2}^{d_v} \lambda_i x^{i-1}$ and $\rho(x) = \sum_{i=2}^{d_c} \rho_i x^{i-1}$ where λ_i is the fraction of the edges belonging to degree-i variable nodes, and ρ_i is the fraction of edges belonging to degree-i check nodes. We denote the ensemble of irregular $(\lambda(x), \rho(x))$-LDPC codes of length n by $\mathcal{C}_n^{(\lambda(x),\rho(x))}$.

In each round of the BP algorithm each node in the graph passes messages to its neighbors along the connecting edges. Let Q_{vc} and R_{cv} denote the messages passed from a variable node v to a check node c and from c to v respectively. Let $P_v = \log \frac{p(v=0)}{p(v=1)}$ denote the *a-priori* Log-Likelihood Ratio (LLR) of a received bit. BP defines the following computation rules:

$$Q_{vc} \leftarrow P_v \quad + \sum_{c' \in N(v) \setminus c} R_{c'v}$$

$$R_{cv} \leftarrow \varphi^{-1}\left(\sum_{v' \in N(c) \setminus v} \varphi(Q_{v'c})\right)$$

such that $N(\cdot)$ denotes the set of neighboring nodes in the graph and $\varphi(x) = (\text{sign}(x)\,, -\log\tanh(\frac{|x|}{2}))$ and its computations are defined over the group $G := \mathbb{F}_2 \times [0, \infty]$.

The order of passing the messages between the nodes is called updating rule or *schedule*. The BP paradigm does not require utilizing a specific schedule. We define an *iteration* as message passing on all edges in both directions. In the case of cycle-free graphs the situation is simple. Given any schedule, the BP algorithm will converge after a finite number of iterations to the exact *a-posterior* probabilities. The number of iterations is bounded by $D/2$ where D is the length of the longest path in the graph. In case of cycle-free graph we can even construct an optimal schedule, in which exactly one message passes in each direction on each edge [7]. Hence, a single iteration is needed for convergence. In graphs with cycles the behavior of iterative decoding is more involved and less clear. The algorithm may produce inaccurate *a-posterior* probabilities. In this case, use of a specific schedule not only influences the convergence rate, but can also affect the algorithm convergence at all.

The standard message-passing schedule for decoding LDPC code, described by Gallager [4], is a version of the so-called *flooding* schedule [7], in which in each iteration all the symbol nodes, and subsequently all the check nodes, pass new messages to their neighbors. For LDPC codes, though the flooding schedule is popular, there is no evidence that it is optimal. Actually, on cycle-free graphs the flooding schedule will converge exactly after $D/2$ iterations. Hence, in terms of the number of iterations it can be considered as the worst schedule, converging in the maximum number of iterations.

Serial schedules, in contrast to the flooding schedule, enable immediate propagation of messages in the same iteration, resulting in faster convergence. We identify two serial scheduling strategies referred to as *serial-V* and *serial-C* schedules. The serial-V schedule is based on a serial update of variable nodes' messages. Instead of sending all the messages from the variable nodes to the check nodes and then all the messages from the check nodes to the variable nodes, as done in the flooding schedule, we interleave the two phases. We go over the variable nodes in some order and for each variable node we send all the input messages to the node, followed by sending all the output messages from the node. That is, for each variable node $v \in V$ we send the following messages:

1) R_{cv} for each $c \in N(v)$ (send all R_{cv} messages into the node v).
2) Q_{vc} for each $c \in N(c)$ (send all Q_{vc} messages from the node v).

The shuffled BP decoding algorithms proposed in [14] and [6] utilize the Serial-V schedule. The serial-C schedule is a dual schedule to the serial-V schedule, based on a serial update of the check nodes' messages. It is defined similarly to the serial-V schedule. The turbo decoding algorithm described in [10] utilizes the Serial-C schedule. Efficient implementation of LDPC decoding algorithms based on serial scheduling are described in [13].

Iterations of the flooding schedule can be fully parallelized, i.e. all variable and check node messages can be updated simultaneously. The serial decoding is inherently sequential. Another option is to use a hybrid of the serial schedule and the flooding schedule. We will refer to the schedule involving a serial update of m subsets of nodes, such that the nodes in each subset are updated simultaneously, as a *semi-serial schedule*. For example, let us divide the check nodes into m subsets $B_1, \ldots, B_m$, and perform an iteration by updating all the check nodes in B_1 simultaneously, then updating all the check nodes in B_2 simultaneously, and so on until B_m.

3 Convergence Analysis

Simulation results show that message passing decoding based on serial scheduling converges approximately in half the number of iterations compared to the flooding schedule. In this section we provide an analysis of the serial schedules' convergence rate. We present the analysis for the serial-V schedule. However the same analysis applies to the serial-C schedule as well.

3.1 Evolution of a computation tree under serial scheduling

Consider a message passed from a variable node v to a check node c along the edge $e = (v, c)$ during message-passing decoding. Depending on the iteration number, this message is a function of a sub-tree that is derived from the code's graph by spanning a tree from the edge e towards the variable node v and of the received channel observations for variables that are contained in the sub-tree. We refer to this sub-tree as the computation tree of e, and denote it by $T_e^{(l)}$. We will refer to a computation tree without cycles as an *acyclic* computation tree. Note that the computation tree for a graph with cycles will contain replicated nodes. For example, the computation tree of an arbitrary edge e after the first decoding iteration of a length 4 regular (2,4)-LDPC code under the flooding and the serial-V schedules is shown on Figure 1. Throughout the rest of this subsection we consider acyclic computation trees. It is easy to see that the computation tree of the flooding schedule is always a balanced tree, and its depth increases by 2 at each iteration. On the other hand, the computation tree of a serial schedule is unbalanced and its

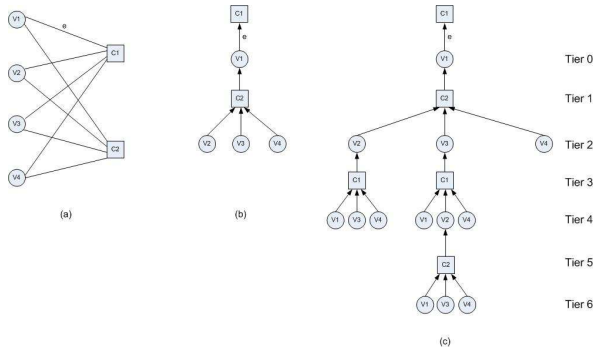

Fig. 1. (a) bipartite graph of an arbitrary length 4 regular (2,4)-LDPC code (b) computation tree of edge e after the first flooding iteration (c) computation tree of edge e after the first serial-V iteration with the serial schedule $\{v_2, v_3, v_1, v_4\}$

depth with respect to each of its leaves grows by at least 2 at each iteration. Thus, the computation tree of an edge under a serial schedule always contains the computation tree of the edge under the flooding schedule. This implies that for cycle-free graphs a serial BP decoder will converge faster (or at least not slower) compared to the flooding BP decoder. Similar arguments were used in [14] to prove this claim. We refine this claim by analyzing exactly how fast the serial schedule computation tree grows compared to the flooding schedule computation tree. For that purpose we would like to determine the probability that a variable v' at graphical distance $2t$ from a root variable node v is contained in the serial schedule computation tree spanned from the edge $e = (v, c)$ after l iterations.

We define the ensembles of serial and semi-serial schedules as follows:

Definition 1.1 (Serial schedules' ensemble):

$$S = \{\sigma : \{1,\ldots,|V|\} \rightarrow \{1,\ldots,|V|\},\ \ \sigma \text{ is a bijection}\}$$

Definition 1.2 (Semi-serial schedules' ensemble):

$$S_m = \{\sigma : \{1,\ldots,|V|\} \rightarrow \{1,\ldots,m\}\}$$

There are $|V|!$ equally probable serial schedules and $m^{|V|}$ equally probable semi-serial schedules. In order to generate a semi-serial schedule we choose *independently* for each variable node a number from $\{1,\ldots,m\}$ randomly with equal probability. This is the number of the subset to which the variable node belongs. The subsets are updated serially according to their subset number, i.e. we first update subset 1, then subset 2, and so on till subset m. We use this definition for the semi-serial schedule since it simplifies the analysis.

Let $T^{(l)}_{flood}$,$T^{(l)}_{serial(\sigma)}$ and $T^{(l)}_{m-serial(\sigma)}$ denote the acyclic computation trees spanned from edge e after l iterations of a flooding schedule, a serial schedule $\sigma \in S$ and a semi-serial schedule $\sigma \in S_m$ respectively (the index e is omitted for brevity). Consider a sequence of numbers $w_1, w_2, \ldots, w_n$, we say that the transition between w_i and w_{i+1} is an ascent if $w_{i+1} \geq w_i$. The following proposition provides a way to test whether a variable node v' belongs to the computation tree $T^{(l)}_{serial(\sigma)}$ spanned from an edge $e = (v, c)$, and whether it is a leaf of $T^{(l)}_{serial(\sigma)}$:

Proposition 1.3: Let w denote a sequence of numbers representing the order of updating the variable nodes along the path $P_v^{v'}$ between v and v' according to serial schedule σ. The variable node v' belongs to the computation tree $T^{(l)}_{serial(\sigma)}$ spanned from edge $e = (v, c)$ iff the number of ascents in w is less than l or the number of ascents in w is equal to l and the last transition in w is an ascent. The variable node v' is a leaf of $T^{(l)}_{serial(\sigma)}$ iff the number of ascents in w is equal to l and the last transition in w is an ascent. □

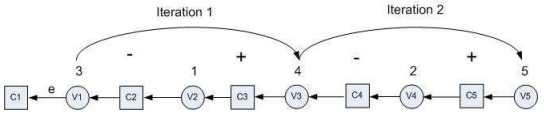

Fig. 2. The advance of the computation tree along the path $P_{v_1}^{v_5}$ for a serial schedule in which the variable nodes $\{v_1, v_2, v_3, v_4, v_5\}$ are updated in the order $\{v_2, v_4, v_1, v_3, v_5\}$

The number of permutations of $\{1, 2, \ldots, n\}$ having k ascents, denoted by $A(n, k)$, is known as Eulerian number [2]. Eulerian numbers can be computed using the recursion:

$$A(n,k) = (k+1)A(n-1,k) + (n-k)A(n-1,k-1)$$
$$A(n,0) = 1, \quad A(n,n-1) = 1, \quad A(n,k>n-1) = 0.$$

Alternatively, they can be computed explicitly by

$$A(n,k) = \sum_{j=0}^{k} (-1)^j \left(\begin{array}{c} n+1 \\ j \end{array} \right) (k-j+1)^n \qquad (1)$$

We define $A_+(n, k)$ as the number of permutations of $\{1, 2, \ldots, n\}$ having k ascents for which the last transition is an ascent.

Proposition 1.4: The numbers $A_+(n, k)$ can be computed using the recursion

$$A_+(n,k) = A(n-1,k-1) + kA_+(n-1,k) + (n-k-1)A_+(n-1,k-1)$$
$$A_+(n,0) = 0, \quad A_+(n,n-1) = 1, \quad A_+(n,k>n-1) = 0 \qquad \square$$

Let $P_L(t, l)$ denote the probability that a variable node at graphical distance $2t$ from the root of $T^{(l)}_{serial}$ is a leaf of $T^{(l)}_{serial}$. Let $P_C(t, l)$ denote the probability that a variable node at graphical distance $2t$ from the root of $T^{(l)}_{serial}$ belongs to $T^{(l)}_{serial}$. Then,

Proposition 1.5:

$$P_L(t,l) = \frac{A_+(t+1,l)}{(t+1)!}$$

$$P_C(t,l) = \frac{\sum_{k=0}^{l-1} A(t+1,k) + A_+(t+1,l)}{(t+1)!}$$

Note that Proposition 1.4 imply that $P_C(t, l) = 1$ for $t \leq l$ and that $P_L(t, l) = 0$ for $t < l$, which is obvious since the computation tree of an edge under a serial schedule always contains the computation tree of the edge under the flooding schedule.

Proposition 1.5 can provide a good indication why the serial schedule converges approximately twice as fast compared to the flooding schedule. To see this we can examine the expected fraction of variable nodes at tier $2t$ which are included in $T^{(\lceil (1+\delta)\frac{l}{2} \rceil)}_{serial}$ compared to the expected fraction of variable nodes at tier $2t$ which are included in $T^{(l)}_{flood}$. As can be seen in Figure 3, for a fixed δ, the expected fraction of variable nodes which belong to $T^{(l)}_{flood} \backslash T^{(\lceil (1+\delta)\frac{l}{2} \rceil)}_{serial}$ tends to zero as l increases.

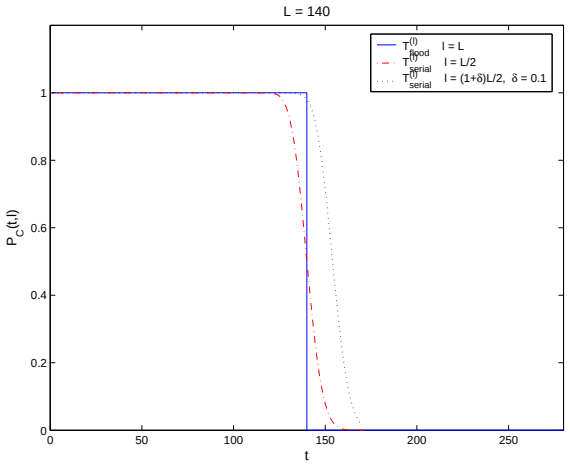

Fig. 3. The expected fraction of variable nodes at tier $2t$ which belong to $T^{(l)}_{flood}$,$T^{(\lceil \frac{l}{2} \rceil)}_{serial}$ and $T^{(\lceil (1+\delta)\frac{l}{2} \rceil)}_{serial}$ for $\delta = 0.1$ and $l = 140$

By considering the asymptotic properties of Eulerian numbers [1], [2], this observation can be quantified

Theorem 1.6: For any $0 < \delta < 1$, there exists l' such that for any $l > l'$ and any integer $t \leq l$, the expected fraction of variable nodes in tier $2t$ that belong to $T^{(\lceil (1+\delta)\frac{l}{2} \rceil)}_{serial}$ is lower bounded by

$$\begin{cases} P_C(t, \lceil (1+\delta)\frac{l}{2} \rceil) > \gamma(l,\delta), & \text{if } \lceil (1+\delta)\frac{l}{2} \rceil < t \leq l \\ P_C(t, \lceil (1+\delta)\frac{l}{2} \rceil) = 1, & \text{if } t \leq \lceil (1+\delta)\frac{l}{2} \rceil \end{cases}$$

$$\gamma(l,\delta) = \max\left\{\tfrac{1}{2}, 1 - \sqrt{\tfrac{3l}{2\pi}} e^{-\frac{3}{2}\delta^2(l+2)}\right\} \longrightarrow_{(l\to\infty)} 1 \qquad \square$$

Note that the analysis does not assume anything about the bipartite graph and applies for both regular and irregular bipartite graphs.

We can perform similar analysis for the evolution of a computation tree of the semi-serial schedule by deriving similar expressions for $P_L(t,l)$ and $P_C(t,l)$ of the semi-serial computation tree. However, the expressions involve cumbersome recursions and are hard to analyze. Instead, we consider the case where the number of subsets m is large with respect to l.

Proposition 1.7:
For any $t \leq 2l$ and $\delta > 0$, if $m > 4l^2 e^{2\delta l}$ then
$P_C(t,l)^{m-serial} \geq (1 - e^{-2\delta l}) P_C(t,l)^{serial}$ $\square$

This is not surprising since the semi-serial schedule becomes closer to the serial schedule as the number of subsets increases.

3.2 Density Evolution

LDPC codes exhibit a threshold phenomenon - as the codeword length tends to infinity, the bit error probability can be made arbitrarily small if the noise level is smaller than some constant threshold. Richardson *et al* developed an algorithm for determining the threshold for various message passing decoders based on the flooding schedule called Density Evolution [12]. The algorithm is based on iterative calculation of the densities of message passed by the decoder in each iteration, assuming the bipartite graph that represents the code is cycle-free. This assumption is justified by a general concentration theorem [12] showing that the decoder's performance on random graphs converges exponentially with the code length to its expected performance. This expected performance in the limit of infinitely long codes can be determined from the corresponding cycle-free bipartite graph.

Note that under the cycle-free graph assumption all possible schedules converge to the exact *a-posterior* value after a finite number of iterations. Hence, the threshold of the flooding and serial schedules is the same and for threshold computation one can use the flooding DE. However, we are interested in the DE algorithm for prediction of the average number of iterations needed for convergence, which depends on the specific schedule even under the cycle-free graph assumption. For that purpose we develop a DE algorithm for the serial schedules. Similarly to the flooding DE, the analysis is based on the assumption of a cycle-free graph, justified by the general concentration theorem proved in the next section.

Consider a semi-serial schedule. Let f_P and $f^{(l)}_{Q_i}$ denote the expected probability density functions (pdf) of a channel message P, and variable-to-check message Q sent from a variable node belonging to subset B_i at the l'th iteration given that the zero codeword is transmitted. Expectation is taken over all graph edges, all tree-like graphs from the ensemble, all semi-serial schedules and all decoder inputs.

Theorem 2.1: Density Evolution of a semi-serial schedule with m subsets for a $(\lambda(x), \rho(x))$-LDPC codes ensemble is described by:
$f^{(l)}_{Q_i} = f_P \bigotimes \lambda(\Gamma^{-1}(\rho(\Gamma(\frac{1}{m}(\sum_{j=1}^{i-1} f^{(l)}_{Q_j} + \sum_{j=i}^{m} f^{(l-1)}_{Q_j})))))$
$f^{(l)}_{Q} = \frac{1}{m}\sum_{i=1}^{m} f^{(l)}_{Q_i}$,
and $f^{(0)}_{Q_i} = f_P \qquad i = 1, \ldots, m.$ $\square$

Here, $\lambda(f) = \sum_{i=2}^{d_v} \lambda_i \bigotimes^{i-1} f$ and $\rho(f) = \sum_{j=2}^{d_c} \rho_j \bigotimes^{j-1} f$ and $\bigotimes$ denotes convolution. The change of measure transform Γ is defined so that for a real random variable X with density f_X, the density of $\varphi(X)$ is $\Gamma(f_X)$. Note that the convolution after the Γ transform is taken over the group $G := \mathbb{F}_2 \times [0, \infty]$.

Unfortunately, developing a DE algorithm for the serial schedule is a much harder task due to statistical dependencies created by the serial scheduling. Consider variable node v at tier t of $T^{(l)}_{serial}$, and assume it is a descendent of variable node v_f at tier $(t-2)$. We know that v has probability $P_{L|C}(t,l) = \frac{P_L(t,l)}{P_C(t,l)}$ to be a leaf of $T^{(l)}_{serial}$. However, once we know it is a leaf, the probability of a neighboring variable node v' at tier t which is a descendent of v_f to be a leaf is higher than $P_{L|C}(t,l)$. Thus, variable nodes which are descendants of the same ancestor are positively correlated. Unfortunately, due to these statistical dependencies between variable nodes it is hard to analyze the ensemble of serial computation trees. Instead we consider a *pseudo-serial* ensemble of computation trees, denoted by $\mathcal{T}^{(l,t)}_{ps}$, which is more amenable to analysis. We show that the pseudo-serial ensemble provides a lower bound on the asymptotic performance of the serial schedule for certain ensembles. We conjecture that this is true for any ensemble over any channel.
We define the pseudo serial ensemble as follows:

Definition 2.2: For a given graph G and an edge e with a cycle-free neighborhood whose depth is at least $2t$, a

pseudo-serial ensemble $\mathcal{T}_{ps}^{(l,t)}$ of acyclic computation trees $T_{ps}^{(l,t)}$ is defined by the following procedure for generating a tree from the ensemble:

1. Span a balanced computation tree T of depth $2t$ from edge e of the graph G.
2. for $j = 1, 2, 3, \ldots, t-1$ do:
   for each variable node v at tier $2j$ set independently with probability $P_{L|C}(j,l)$ if v is a leaf. If so, delete the subtree below v.

Let $p_{ps}^{(l,t)}$ denote the expected fraction of incorrect or erased messages computed by a computation tree in $\mathcal{T}_{ps}^{(l,t)}$. We derive a DE algorithm for computing $p_{ps}^{(l,t)}$ by tracking the expected pdf of messages sent from each tier of variable nodes, starting from tier $2t$ (the farthest from the root) and finishing in tier 0 (which represents the root). Let $f_Q^{(2j)}$ denote the expected pdf of a variable-to-check message sent from tier $2j$ to tier $2j-1$.

Theorem 2.3: Density Evolution of the pseudo-serial ensemble $\mathcal{T}_{ps}^{(l,t)}$ for a $(\lambda(x),\rho(x))$-LDPC codes ensemble: $p_{ps}^{(l,t)} = \int_{-\infty}^{0} f_Q^{(0)}(q)dq$,
where $f_Q^{(0)}$ is computed by the following recursion
$f_Q^{(2j)} = P_{L|C}(j,l) f_P + (1 - P_{L|C}(j,l)) f_P \bigotimes \lambda(\Gamma^{-1}(\rho(\Gamma(f_Q^{(2j+2)}))))$
for $j = t-1, \ldots, 0$ and $f_Q^{(2t)} = f_P$. □

The fraction of erroneous messages as a function of the number of iterations as predicted by the Density Evolution algorithm for the flooding schedule, the semi-serial schedule with various values of m and the pseudo-serial schedule for a (3,4)-LDPC code over a Binary Erasure Channel (BEC) with channel parameter $\epsilon = 0.6466$, and for a (3,6)-LDPC code over a Gaussian channel with channel parameter $Eb/No = 1.12dB$ are shown in Figures 4 and 5. We applied the DE algorithms for analysis of various regular and irregular LDPC ensembles over the BEC and the AWGN channel and obtained similar results for all ensembles. The semi-serial and pseudo-serial ensembles converge in approximately half the number of iterations compared to the flooding schedule, when working near the ensemble's capacity. Furthermore, the semi-serial convergence rate improves as the number of subsets m increases, approaching some bound. We conjecture that this bound is the convergence rate of the serial schedule, since as the number of subsets in the semi-serial schedule increases it becomes closer to the serial schedule as indicated by Proposition 1.7. Finally, the results provide a strong indication that the pseudo-serial can serve as a lower bound on the convergence rate of the serial schedule. For all the ensembles we tested, the convergence rate of the pseudo-serial ensemble was slower than the convergence rate of the semi-serial ensemble with $m = 100$. Unfortunately, we were unable to prove this claim in general. Though, we prove it for a $(x, \rho(x))$-LDPC ensemble over the BEC.

Proposition 2.4: For $(x, \rho(x))$-LDPC ensemble over the BEC and for any $t > 0$: $p_{serial}^{(l)} \leq p_{ps}^{(l,t)}$
where, $p_{serial}^{(l)}$ denotes the expected fraction of incorrect messages passed along an arbitrary edge with an acyclic computation tree at iteration l of the serial BP decoder, averaged over all possible $|V|!$ serial schedules.

Motivated by the DE results and by Proposition 2.4 we conjecture that

Conjecture 2.5: For any integers $l > 0$ and $t > 0$ $p_{serial}^{(l)} \leq p_{ps}^{(l,t)}$ regardless of the graph ensemble and the channel.

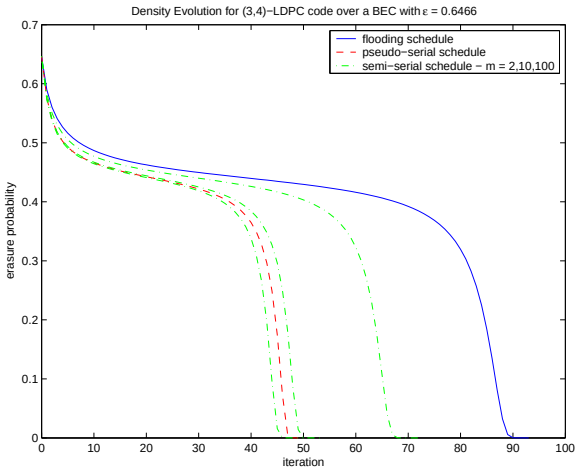

Fig. 4. Density Evolution for a (3,4)-LDPC code over the BEC channel

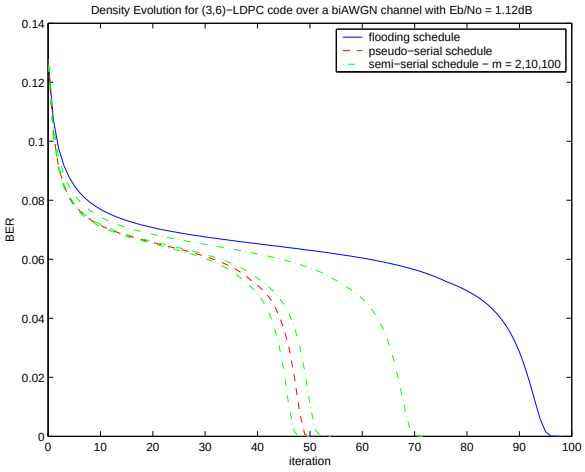

Fig. 5. Density Evolution for a (3,6)-LDPC code over the Gaussian channel

We conclude this section by proving that for BEC, asymptotically in the code length and when working near the decoder's capacity, the pseudo-serial ensemble is expected to converge in half the number of iterations compared to the flooding schedule. Let us consider a set of BEC's, parameterized by the erasure probability α. The BEC is monotone with respect to the BP decoder [12], i.e. convergence of the decoder for a parameter α implies convergence of the decoder for any parameter $\alpha' \leq \alpha$. Let α^* denote the decoder's threshold as predicted by the DE algorithm, which is given by,

$$\alpha^* = \sup_{\alpha} \{\alpha : \alpha \cdot \lambda(1 - \rho(1-x)) < x \quad \text{for } x \in (0, \alpha)\} \quad (2)$$

Note that α^* is independent of the particular scheduling applied by the decoder. Let $p_{fp}(\alpha)$ be defined as

$$p_{fp}(\alpha) = \arg_{x:\ x \in (0,\alpha)} \min\{x - \alpha \cdot \lambda(1 - \rho(1-x))\} \quad (3)$$

$p_{fp}(\alpha^*)$ is the erasure probability of a variable-to-check message at which fixed point of the DE algorithm occurs. Furthermore, let $l(\alpha, p)$ denote the minimal number of iterations required by the decoder to achieve an expected message erasure probability lower or equal to p over an arbitrary edge with acyclic computation tree, when working over a BEC with parameter α.

Theorem 2.6: For any $\epsilon > 0$ and $\delta > 0$, there exists $\alpha' \leq \alpha^*$ such that for any $\alpha \in [\alpha', \alpha^*]$ and for any $p \in (0, p_{fp}(\alpha))$:

$$l_{ps}(\alpha, p+\epsilon) \leq \left\lceil (1+\delta)\frac{l_{flood}(\alpha, p)}{2} \right\rceil$$

Modulo Conjecture 2.5 for any LDPC ensemble, or using Proposition 2.4 for $(x, \rho(x))$-LDPC ensembles we get

Corollary 2.7: For any $\epsilon > 0$ and $\delta > 0$, there exists $\alpha' \leq \alpha^*$ such that for any $\alpha \in [\alpha', \alpha^*]$ and for any $p \in (0, p_{fp}(\alpha))$:

$$l_{serial}(\alpha, p+\epsilon) \leq \left\lceil (1+\delta)\frac{l_{flood}(\alpha, p)}{2} \right\rceil$$

3.3 Concentration Theorem

In the previous section, we saw how the average behavior of a given semi-serially or serially scheduled message passing decoder can be determined, assuming that the graph does not contain cycles. In this section, we follow [12] and prove a general concentration theorem for the semi-serial and serial schedules, showing that for a randomly chosen serial schedule, graph and decoder input the fraction of incorrect messages passed at the l'th decoding step is arbitrarily close to the value predicted by the proposed DE algorithm with probability that approaches 1 as the code length increases.

The depth of a semi-serial computation tree after l iterations is not fixed and depends on the graph, the serial schedule by which the sets of variable nodes are updated and the specific edge. However, it is easy to see that the depth of the computation tree after l iterations is bounded by $D_e^{(l)} \leq 2ml$ (where m is the number of subsets). Since for the semi-serial schedule the computation tree's depth is bounded, a similar concentration theorem to the one proved in [12] for the flood schedule applies to the semi-serial schedule requiring minor changes in its proof for showing concentration over the semi-serial schedules:

Theorem 3.1: (Concentration theorem for the semi-serial schedule). Over the probability space of all graphs $\mathcal{C}_n^{(d_v,d_c)}$, all semi-serial schedules with m variable node subsets and all channel realizations, let Z denote the number of incorrect messages among all $d_v n$ variable-to-check messages sent in the l-th decoding step. Further, let p denote the expected number of incorrect messages passed along an arbitrary edge with an acyclic computation tree at the l-th decoding step. Then there exist positive constants β, γ, such that for any $\epsilon > 0$ and $n > \frac{2\gamma}{\epsilon}$,

$$Pr\left(|Z/(nd_v) - p| > \epsilon\right) \leq 2e^{-\beta\epsilon^2 n} \qquad (4)$$

Unfortunately, since for a serial schedule the number of sets m is not fixed and depends on the code length, the depth of the computation graph of an edge cannot be bounded by a constant even after a single iteration. For this reason, the flooding concentration theorem proved in [12] cannot be applied directly to the serial schedule and some additional arguments are needed to show that the probability that the computation tree is not bounded approaches zero as the code length increases:

Lemma 3.2: . Let e denote an edge in a randomly chosen element of $\mathcal{C}_n^{(d_v,d_c)}$ decoded using a randomly chosen serial schedule and let $T_e^{(l)}$ denote the computation tree of e at the l-th iteration with depth $D_e^{(l)}$. Then for sufficiently large n

$$Pr\left(T_e^{(l)} \text{is cyclic or} D_e^{(l)} \geq \frac{4l\ln(n)}{\ln(\ln(n))}\right) \leq \frac{o(n)}{n} \to 0 \quad (5)$$

Using Lemma 3.2, we are then able to prove the concentration theorem for the serial schedule.

Theorem 3.3: (Concentration theorem for the serial schedule). Over the probability space of all graphs $\mathcal{C}_n^{(d_v,d_c)}$, serial schedules and channel realizations, let Z denote the number of incorrect messages among all $d_v n$ variable-to-check messages sent in the l-th decoding step. Further, let p denote the expected number of incorrect messages passed along an arbitrary edge with an acyclic computation tree at the l-th decoding step. Then for any $\epsilon > 0$ and sufficiently large n,

$$Pr\left(|Z/(nd_v) - p| > \epsilon\right) \leq \frac{2}{n^{1-o(1)}} \qquad (6)$$

References

[1] E. A. Bender, "Central and local limit theorems applied to asymptotic enumeration," *J. Combinatorial Theory*, Ser. A, vol.15 , pp. 91–111, 1973.

[2] L. Carlitz, D. C. Kurtz, R. Scoville, O. P. Stackelberg, "Asymptotic properties of Eulerian numbers," *Z. Wahrscheinlichkeitstheorie und Verw. Gebiete*, vol.23, pp. 47–54, 1972.

[3] G. D. Forney Jr., "On iterative decoding and the two-way algorithm," *Proc. Int. Symp. on Turbo Codes and Related Topics*, pp. 12–15, 1997.

[4] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Info. Theory*, vol. IT-8, pp. 21–28, 1962.

[5] D.E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," *IEEE Workshop on Signal Proc. Sys. (SIPS.04)*, Austin, TX, October 13-15, 2004.

[6] H. Kfir and I. Kanter, "Parallel versus sequential updating for belief propagation decoding," *Physica A*, vol. 330, pp. 259–270, 2003.

[7] F. R. Kschischang, B. J. Frey and H. Loeliger, "Factor Graphs and the sum-product algorithm," *IEEE Trans. on Info. Theory*, vol. 47(2), pp. 498–519, 2001.

[8] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi and D. A. Spielman, "Improved low density parity-check codes using irregular graphs and belief propagation," *Proc. of the IEEE Int. Symp. on Inf. Theory (ISIT)*, p. 117, 1998.

[9] Y. Mao and A. H. Banihashemi, "Decoding low-density parity-check codes with probabilistic scheduling," *Communications Letters*, vol. 5 (10) , pp. 414–416, 2001.

[10] M. M. Mansour and N. R. Shanbhag, "High-Throughput LDPC decoders," *IEEE Trans. on Very Large Scale Integration Systems*, vol. 11, pp. 976–995, 2003.

[11] J. Pearl, *Probabilistic reasoning in intelligent systems: Networks of plausible inference*, Morgan Kaufmann, 1988.

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

[13] E. Sharon, S. Litsyn and J. Goldberger "An Efficient Message-Passing Schedule for LDPC Decoding," in proc. 23rd IEEE Convention in Tel-Aviv, Israel, pp. 223-226, September 6-7, 2004.

[14] J. Zhang and M. Fossorier, "Shuffled belief propagation decoding," *The Proceedings 36th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, USA, vol.1, pp.8–15, 2002.