Efficient Serial Message-Passing Schedules for LDPC Decoding

Eran Sharon, Simon Litsyn, *Senior Member, IEEE*, and Jacob Goldberger

***Abstract*—Conventionally, in each low-density parity-check (LDPC) decoding iteration all the variable nodes and subsequently all the check nodes send messages to their neighbors (flooding schedule). An alternative, more efficient, approach is to update the nodes' messages serially (serial schedule). A theoretical analysis of serial message passing decoding schedules is presented. In particular, the evolution of the computation tree under serial scheduling is analyzed. It shows that the tree grows twice as fast in comparison to the flooding schedule's one, indicating that the serial schedule propagates information twice as fast in the code's underlying graph. Furthermore, an 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 for long codes the serial schedule is expected to converge in half the number of iterations compared to the standard flooding schedule, when working near the ensemble's threshold. This observation is generally proved for the binary erasure channel (BEC) under some natural assumptions. Finally, an accompanying concentration theorem is proved.**

***Index Terms*—Belief propagation, density evolution, factor graph, iterative decoding, low-density parity-check (LDPC) codes, message-passing decoding.**

I. Introduction

The most attractive feature of low-density parity-check (LDPC) codes is their ability to achieve a significant fraction of the channel capacity using iterative decoding at relatively low implementation complexity. In this work, we analyze the convergence rate of message-passing iterative decoding algorithms. These algorithms operate on the bipartite graph representation of the code by iteratively exchanging messages along the graph edges between variable and check nodes.

The order of passing the messages along the graph edges is referred to as an updating rule or schedule. The standard message-passing schedule is the *flooding* schedule [11], where in each iteration all the variable nodes, and subsequently all the check nodes, pass new messages to their neighbors. Even though the flooding schedule is very popular, it is not efficient in terms of convergence rate. Alternative schedules, which are based on a serial updating order have been considered in various papers. Zhang and Fossorier [24] and Kfir and Kanter [10] developed a decoding algorithm which is based on a sequential schedule involving a serial 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 Shnabhag [18] proposed a turbo decoding algorithm for a generalization of Gallager's codes, where the LDPC code is constructed as an intersection of several supercodes. This algorithm utilizes a dual decoding schedule to the one suggested in [24], [10] involving a serial update of the supercodes. The authors [22] and Hocevar [9] presented a similar decoding algorithm generalized for LDPC codes and proposed efficient implementation of the algorithm.

Adaptive scheduling techniques, which change the message updating order based on the graph structure or the specific channel realization were also considered in several papers. Mao and Banihashemi [17] proposed a probabilistic variant of the flooding schedule, which reduces the error floor and the number of undetected errors by taking into account the loop structure of the underlying code's graph. A lazy scheduling approach in which not all messages are updated every iteration, and the probability of updating a message is a function of its reliability and updating history, was described in [13].

In this paper, we consider nonadaptive, fixed scheduling techniques. We use several approaches for analyzing the convergence rate of serial and flooding decoding schedules. The analysis quantifies and confirms empirical observations about the convergence rate of serial and flooding schedules from previous papers.

The paper is organized as follows. Background on LDPC codes and message passing decoding schedules is provided in Section II. In Section III, we analyze the decoding computation tree evolution under serial scheduling and show that it 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. Asymptotic convergence rate analysis of serial scheduling based on the density evolution (DE) algorithm is described in Section IV. Based on it, we prove that under certain likely assumptions, 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, a concentration theorem for serially scheduled decoding is proved.

Manuscript received June 12, 2005; revised May 1, 2007. The material in this paper was presented in part at the 4th International Symposium on Turbo Codes, Munich, Germany, April 2006.

E. Sharon and S. Litsyn are with the School of Electrical Engineering, Tel-Aviv University, Ramat-Aviv 69978, Israel (e-mail: eransh@eng.tau.ac.il; litsyn@eng.tau.ac.il).

J. Goldberger is with the School of Engineering, Bar-Ilan University, Ramat Gan 52900, Israel (e-mail: goldbej@eng.biu.ac.il).

Communicated by T. J. Richardson, Associate Editor for Coding Theory.

Color versions of Figures 1–7 in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2007.907507

II. Background

LDPC codes are linear error-correcting codes defined by sparse parity-check matrices [8]. Alternatively, they can be defined by a sparse bipartite undirected graph consisting of variable and check nodes. We will denote the sets of variable nodes, check nodes, and graph edges 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 a code that is defined by a regular bipartite graph, such that each variable node is connected to d_v check nodes and each check node is connected to d_c variable nodes. The ensemble of regular (d_v, d_c)-LDPC codes of length n is denoted by $\mathcal{C}_n^{(d_v,d_c)}$. An irregular LDPC code [14] is represented by an irregular bipartite graph, where the degrees of variable nodes and check nodes may differ. An ensemble of irregular LDPC codes is defined by the variable nodes' and check nodes' degree distributions. Let $\lambda(x) = \sum_{i=2}^{d_v} \lambda_i x^{i-1}$ and $\rho(x) = \sum_{i=2}^{d_c} \rho_i x^{i-1}$ be the generating functions of the degree distributions for the variable and check nodes, respectively, where λ_i and ρ_i are the fractions of the edges belonging to degree-i variables and check nodes, respectively. We denote the ensemble of irregular $(\lambda(x), \rho(x))$-LDPC codes of length n by $\mathcal{C}_n^{(\lambda(x),\rho(x))}$.

LDPC codes can be efficiently decoded using iterative message-passing decoding algorithms. These algorithms operate on the bipartite graph representation of the code by iteratively exchanging messages between the variable and check nodes along the edges of the graph. Even though the results derived in this paper can be applied to any message passing algorithm, we will mainly concentrate on the belief propagation (BP) algorithm [19], [8] with log-likelihood ratio (LLR) messages. We denote the channel message for a variable node v by P_v, the message sent from a variable node v to a check node c by Q_{vc}, and the message sent from c to v by R_{cv}. The BP algorithm utilizes the following message computation rules:

$$Q_{vc} \leftarrow P_v + \sum_{c' \in N(v)\setminus c} R_{c'v} \tag{1}$$

$$R_{cv} \leftarrow \varphi^{-1}\left(\sum_{v' \in N(c)\setminus v} \varphi(Q_{v'c}) \right) \tag{2}$$

where $N(c)$ denotes the set of neighbors of a node c, and

$$\varphi(x) = \left(\mathrm{sign}(x), -\log\tanh\left(\frac{|x|}{2}\right)\right) \tag{3}$$

$$\varphi^{-1}(\mathrm{sign}, x) = (-1)^{\mathrm{sign}}\left(-\log\tanh\left(\frac{x}{2}\right)\right). \tag{4}$$

Note that $\varphi(x)$ and its computations are defined on the group $G := \mathbb{F}_2 \times [0, \infty]$.

The order of passing the messages between the nodes is referred to as an updating rule or a schedule. The BP paradigm does not require utilizing a specific schedule. In this paper, we consider fixed, nonadaptive, decoding schedules, where in each iteration messages are passed along all edges in both directions. In these schedules, the decoding complexity is proportional to the expected number of iterations required for the decoder to converge to a valid codeword. Hence, it is of interest to analyze the decoder's convergence rate.

In the case of cycle-free graphs, the situation is simple and well understood. Given any iterative schedule, the BP algorithm will converge after a finite number of iterations to the exact *a posteriori* probabilities. The number of iterations is bounded by $D/2$ where D is the diameter (length of the longest path) of the graph. In case of cycle-free graphs, we can even construct an optimal schedule [12] in the sense of the minimum number of messages that have to be sent until the convergence is achieved. The number of messages passed in the optimal schedule is precisely $2|E|$, where $|E|$ is the number of graph edges, and exactly one message will pass in each direction over each edge. Hence, a single iteration is needed for convergence. However, graphs defining good LDPC codes contain cycles. In these graphs, the behavior of iterative decoding is less clear. BP produces inaccurate *a posteriori* probabilities and may even not converge. Nevertheless, it exhibits excellent empirical performance.

The standard message-passing schedule for decoding LDPC code, already presented by Gallager [8], is the *flooding* schedule [11], in which in each iteration all the variable nodes, and subsequently, all the check nodes, pass new messages to their neighbors. Even though the flooding schedule is popular, there is no evidence that it is optimal and provides a good complexity–performance tradeoff. 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, resulting in faster convergence. We consider 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 variable nodes to check nodes and then all the messages from check nodes to variable nodes, as done in the flooding schedule, the two phases are interleaved. The variable nodes are traversed in some order and for each variable node $v \in V$ the following messages are sent:

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

The shuffled BP decoding algorithms proposed in [24] and [10] and the "lazy scheduled" decoding in [13] utilize the Serial-V schedule.

The serial-C schedule is based on a serial update of the check nodes' messages, hence, it can be seen as dual to the serial-V schedule. The check nodes are traversed in some order and for each check node $c \in C$ the following messages are sent:

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

The turbo decoding algorithm [18], the serial-C decoding algorithm [22], and the layered BP decoding [9] use the Serial-C schedule.

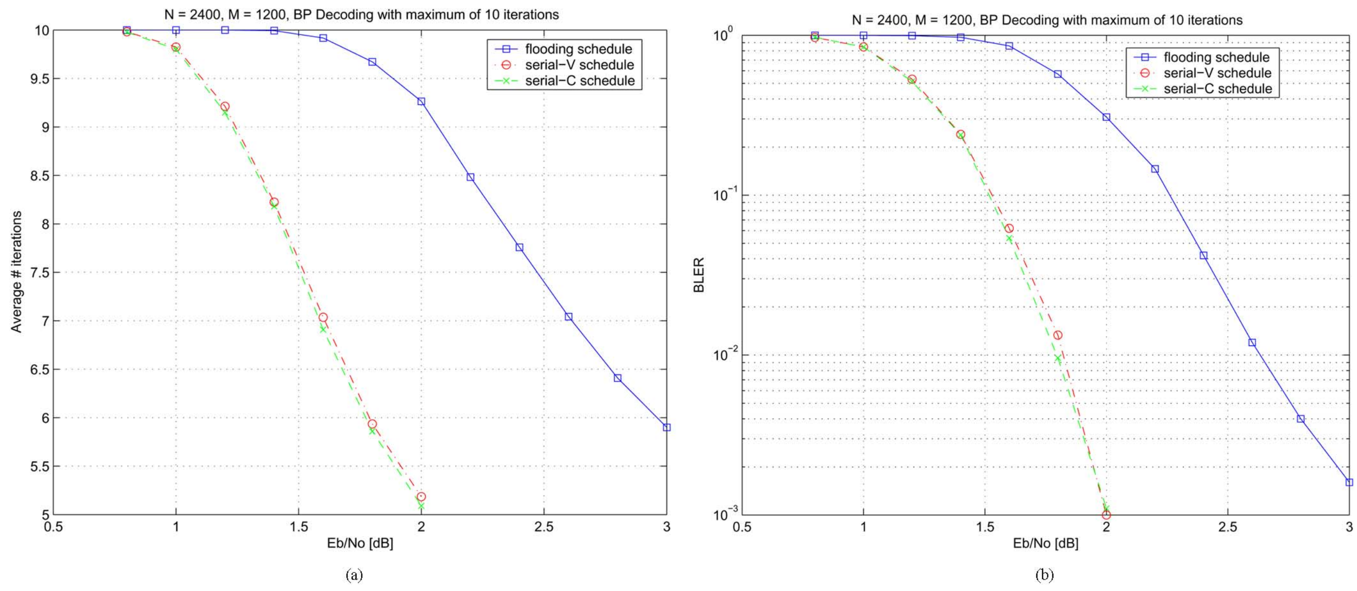


Fig. 1. Simulation results for a length 2400, irregular $(0.1316x + 0.5365x^2 + 0.3319x^7, x^6)$-LDPC code over the Gaussian channel with maximum of 10 iterations: (a) average iterations versus E_b/N_0. (b) Block error rate versus E_b/N_0.

Serial schedules have several important advantages over the flooding schedule. The main advantage is faster convergence. Simulation results indicate that they converge approximately in half the number of iterations (see Fig. 1(a)). This means that a decoder utilizing a serial schedule requires approximately half the processing hardware in order to achieve a given decoding throughput, compared to a flooding-based decoder. Moreover, in practical applications, the maximal number of allowed decoding iterations is usually limited due to latency or throughput constraints. Hence, a serially scheduled decoder provides better error rate performance as shown in Fig. 1(b). Another advantage of serial schedules is that they can be efficiently implemented with significantly lower memory requirements [22], [9], [13].

One might argue that the disadvantage of serial schedules is that they do not allow for a parallelized decoder hardware implementation due to their serial nature, while the flooding schedule can be fully parallelized (i.e., all variable- and check-node messages can be updated simultaneously). However, the serial schedule can be partially parallelized. Messages from sets of nodes can be updated simultaneously. For instance, in the serial-V schedule, if the sequence of variable nodes is divided into the subsets $B_1, \dots, B_m$, such that no two variable nodes in a subset are connected to the same check node, i.e.,

$$\forall\, i \in \{1, \dots, m\}, \quad \forall\, v, v' \in B_i : N(v) \cap N(v') = \phi \quad (5)$$

then there is no difference whether the variable nodes within the subsets are updated serially or in parallel. Since a fully parallel implementation is usually not possible due to the complex interconnect fabric between the memory and computation nodes, the partially sequential nature of the serial schedule is not limiting.

There is also a possibility to use a hybrid schedule of the flooding and serial schedules, in which an iteration involves a serial update of m subsets of nodes, $B_1, \dots, B_m$, such that the nodes in each subset are updated simultaneously. We will refer to this schedule as a *semi-serial schedule*. Note that if $m = 1$, the semi-serial schedule coincides with the flooding schedule and if the check nodes are divided into subsets $B_1, \dots, B_m$ satisfying (5) it coincides with some serial schedule.

Sections III and IV include an analysis of the convergence rate of serial and semi-serial schedules. We present the analysis for serial-V schedules. However, the same analysis applies to serial-C schedules as well.

III. Computation Tree Evolution Under Serial Scheduling

In this section, we examine the evolution of the message-passing algorithm's computation tree under serial scheduling. Consider a message passed from a variable node v to a check node c along the edge $e = (v, c)$ of a given graph G during message-passing decoding. Depending on the iteration number, this message is a function of a subtree that is derived from the code's graph G by spanning a tree from the edge e toward the variable node v and of the received channel observations for variables that are contained in the subtree. We refer to this subtree 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 Fig. 2.

It is easy to see that the computation tree of the flooding schedule is always a balanced tree, and its depth is increased by 2 at each iteration. On the other hand, the computation tree of a serial schedule is not balanced and its 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 [24] to prove this

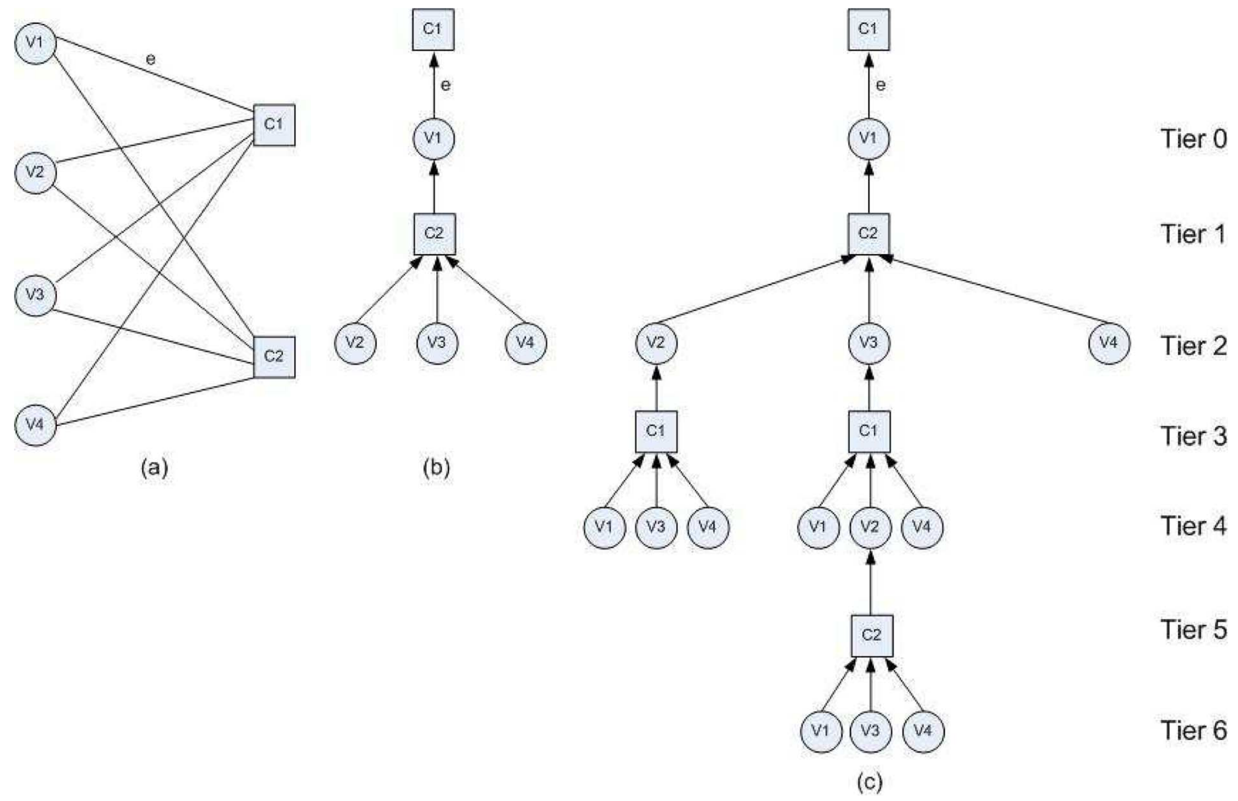


Fig. 2. (a) Bipartite graph of an arbitrary length 4 regular $(2,4)$-LDPC code. (b) Computation tree of edge e after the first flooding iteration. (c) Computation tree of edge e after the first serial-V iteration with the serial schedule $\{v_2, v_3, v_1, v_4\}$.

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.

Throughout the rest of this subsection we consider acyclic computation trees. We define the ensembles of serial-V and semi-serial-V schedules as follows.

Definition III-1 (Serial Schedules' Ensemble):

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

There are $|V|!$ equally probable serial schedules.

Definition III-2 (Semi-Serial Schedules' Ensemble):

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

There are $m^{|V|}$ equally probable semi-serial schedules. In order to generate a semi-serial schedule, we randomly choose for each variable node a number from $\{1,\ldots,m\}$ independently 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 $T_{\text{flood}}^{(l)}$ denote the acyclic computation tree spanned from edge e after l flooding iterations (the index e is omitted for brevity). Let $T_{\text{serial}(\sigma)}^{(l)}$ denote the acyclic computation tree spanned from edge e after l iterations of serial schedule $\sigma \in S$. Let $T_{m\text{-serial}(\sigma)}^{(l)}$ denote the acyclic computation tree spanned from edge e after l iterations of semi-serial schedule $\sigma \in S_m$. 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 method for testing whether a variable node v' belongs to the computation tree $T_{\text{serial}(\sigma)}^{(l)}$ spanned from an edge $e=(v,c)$, and whether it is a leaf of $T_{\text{serial}(\sigma)}^{(l)}$:

Proposition III-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 $T_{\text{serial}(\sigma)}^{(l)}$ 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 $T_{\text{serial}(\sigma)}^{(l)}$ iff the number of ascents in w is equal to l and the last transition in w is an ascent.

Proof: Follows directly from the definition of the serial schedule. In each iteration the computation the tree is extended along $P_v^{v'}$ up to the next ascent in w. □

For example, a path $P_{v_1}^{v_5}$ between two variable nodes v_1 and v_5 is shown in Fig. 3. The order of updating the variable nodes along the path is $w=\{3,1,4,2,5\}$. In each iteration, the computation tree spanned from the edge e along $P_{v_1}^{v_5}$ is extended up to the next ascent in w. Thus, the computation tree will include v_5 after two iterations.

Let $\pi(n,k)$ denote the set of permutations of $\{1,2,\ldots,n\}$ having k ascents. The cardinality of $\pi(n,k)$, denoted by

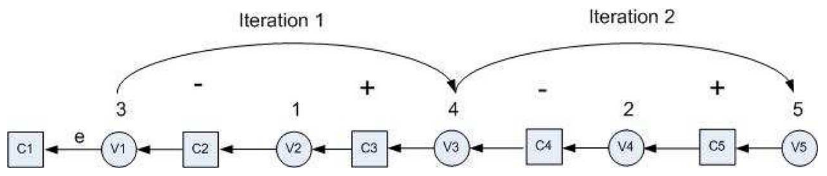


Fig. 3. The advance of the computation tree along the path $P_{v_1}^{v_5}$ for a serial schedule in which the variable nodes $\{v_1, v_2, v_3, v_4, v_5\}$ are updated in the order $\{v_2, v_4, v_1, v_3, v_5\}$.

$A(n,k)$, is known as the Eulerian number [3], [6]. The Eulerian numbers can be computed using the recursion

$$\begin{aligned} A(n,k) &= (k+1)A(n-1,k) + (n-k)A(n-1,k-1) \\ A(n,0) &= 1, \quad A(n,n-1) = 1, \quad A(n,k>n-1) = 0. \end{aligned} \tag{6}$$

Alternatively, they can be computed explicitly by

$$A(n,k) = \sum_{j=0}^{k} (-1)^j \binom{n+1}{j} (k-j+1)^n. \tag{7}$$

By definition, the Eulerian numbers satisfy the equality

$$\sum_{k=0}^{n-1} A(n,k) = n!. \tag{8}$$

Let $\pi_+(n,k)$ denote the set of permutations of $\{1,2,\ldots,n\}$ having k ascents for which the last transition is an ascent. We denote the cardinality of $\pi_+(n,k)$ by $A_+(n,k)$.

Proposition III-4: The numbers $A_+(n,k)$ can be computed using the recursion

$$\begin{aligned} A_+(n,k) &= A(n-1,k-1) + kA_+(n-1,k) \\ &\quad + (n-k-1)A_+(n-1,k-1) \\ A_+(n,0) &= 0, \quad A_+(n,n-1) = 1, \quad A_+(n,k>n-1) = 0. \end{aligned}$$

Proof: All permutations in $\pi_+(n,k)$ can be generated in one of the following ways: 1) add n to a permutation in $\pi_+(n-1,k)$ as the first number or between two consecutive ascending numbers, but not between the last two ascending numbers; 2) add n to a permutation in $\pi_+(n-1,k-1)$ between two consecutive descending numbers; 3) add n as the last number of a permutation in $\pi(n-1,k-1)\backslash\pi_+(n-1,k-1)$.

The sets of permutations generated in each way cannot overlap since n is located either between two ascending numbers, two descending numbers, or as the last number. Thus

$$\begin{aligned} A_+(n,k) &= kA_+(n-1,k) + (n-k)A_+(n-1,k-1) \\ &\quad + (A(n-1,k-1) - A_+(n-1,k-1)). \quad \square \end{aligned}$$

Let $d(v,v')$ denote the graphical distance between variable nodes v and v'. Let $P_L(t,l)$ denote the probability that a variable node at graphical distance $2t$ from the root of $T_{\text{serial}}^{(l)}$ is a leaf of $T_{\text{serial}}^{(l)}$. Let $P_C(t,l)$ denote the probability that a variable node at graphical distance $2t$ from the root of $T_{\text{serial}}^{(l)}$ belongs to $T_{\text{serial}}^{(l)}$. Then, $P_L(t,l)$ and $P_C(t,l)$ are given by the following.

Proposition III-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 III-4 as well as (6) and (8) imply that $P_C(t,l) = 1$ for $t \le l$ 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.

Proof: Let $I(\cdot)$ denote the indicator function. Then

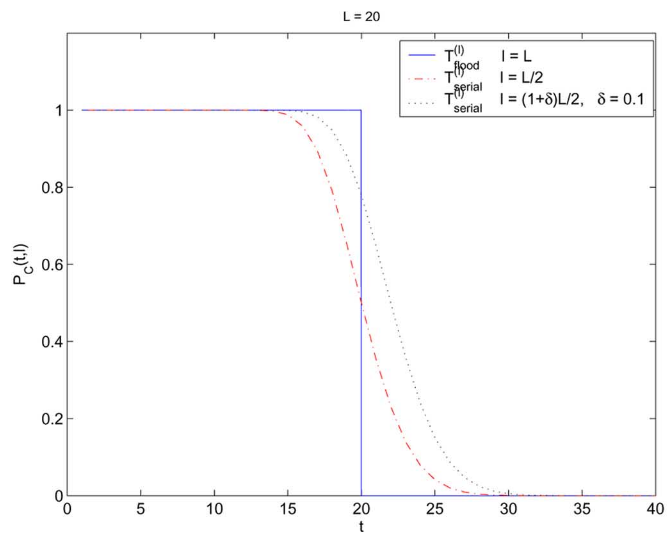
$$\begin{aligned} &P_L(t,l) \\ &= Pr\left(v' \text{ is a leaf of } T_{\text{serial}}^{(l)} | d(v,v') = 2t\right) \\ &= \mathbf{E}_{\sigma\in S}\left(\mathbf{E}_{v':d(v,v')=2t}\left(I\left(v' \text{ is a leaf of } T_{serial(\sigma)}^{(l)}\right)\right)\right) \\ &= \mathbf{E}_{v':d(v,v')=2t}\left(\mathbf{E}_{\sigma\in S}\left(I\left(v' \text{ is a leaf of } T_{serial(\sigma)}^{(l)}\right)\right)\right) \\ &= \mathbf{E}_{v':d(v,v')=2t}\left(\frac{|\pi_+(t+1,l)|}{(t+1)!}\right) \\ &= \frac{A_+(t+1,l)}{(t+1)!}. \end{aligned}$$

The fourth equality follows directly from Proposition III-.3.

Similarly, we have

$$\begin{aligned} P_C(t,l) &= Pr\left(v' \in T_{\text{serial}}^{(l)} | d(v,v') = 2t\right) \\ &= \mathbf{E}_{\sigma\in S}\left(\mathbf{E}_{v':d(v,v')=2t}\left(I\left(v' \in T_{\text{serial}(\sigma)}^{(l)}\right)\right)\right) \\ &= \mathbf{E}_{v':d(v,v')=2t}\left(\mathbf{E}_{\sigma\in S}\left(I\left(v' \in T_{\text{serial}(\sigma)}^{(l)}\right)\right)\right) \\ &= \mathbf{E}_{v':d(v,v')=2t}\left(\frac{\left|\bigcup_{k=0}^{l-1}\pi(t+1,k)\bigcup\pi_+(t+1,l)\right|}{(t+1)!}\right) \\ &= \frac{\sum_{k=0}^{l-1} A(t+1,k) + A_+(t+1,l)}{(t+1)!}. \quad \square \end{aligned}$$

Proposition III-5 shows that the serial schedule's computation tree grows twice as fast compared to the flooding schedule's computation tree. To see this we can examine the expected fraction of variable nodes at tier $2t$ which are included in $T_{\text{serial}}^{\left(\lceil(1+\delta)\frac{l}{2}\rceil\right)}$ compared to the expected fraction of variable nodes at tier $2t$ which are included in $T_{\text{flood}}^{(l)}$. As can be seen in Fig. 4, for a fixed δ, the expected fraction of variable nodes which belong to $T_{\text{flood}}^{(l)}\backslash T_{\text{serial}}^{\left(\lceil(1+\delta)\frac{l}{2}\rceil\right)}$ tends to zero as l increases. This indicates that the serial schedule propagates



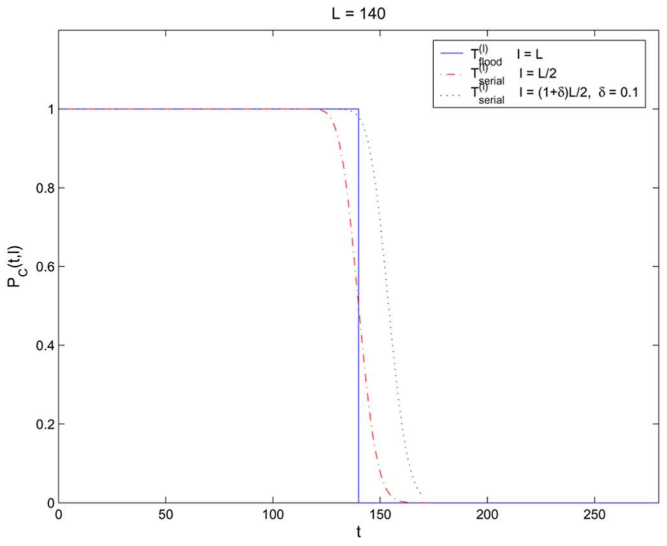


Fig. 4. The expected fraction of variable nodes at tier $2t$ which belong to $T^{(l)}_{\mathrm{flood}}$, $T^{(\lceil \frac{l}{2} \rceil)}_{\mathrm{serial}}$, and $T^{(\lceil (1+\delta)\frac{l}{2} \rceil)}_{\mathrm{serial}}$ for $\delta = 0.1$ and $l = 20, 140$.

information twice as fast on the code's graph, explaining its faster convergence.

By considering the asymptotic properties of Eulerian numbers, this observation can be quantified.

Theorem III-6: For any $0 < \delta < 1$, there exists l' such that for any $l > l'$ and any integer $t \le l$, the expected fraction of variable nodes in tier $2t$ that belong to $T^{(\lceil (1+\delta)\frac{l}{2} \rceil)}_{\mathrm{serial}}$ is lower-bounded by the expression shown at the bottom of the page. Hence

$$P_C\left(t, \left\lceil (1+\delta)\frac{l}{2} \right\rceil\right) > \max\left\{\frac{1}{2}, 1 - \sqrt{\frac{3l}{2\pi}} e^{-\frac{3}{2}\delta^2(l+2)}\right\}$$
$$\Rightarrow \lim_{l\to\infty} P_C\left(t, \left\lceil (1+\delta)\frac{l}{2} \right\rceil\right) = 1.$$

Proof: Asymptotic properties of Eulerian numbers were studied in [2], [4]. Bender [2] proved a local limit theorem, showing that Eulerian numbers have a normal asymptotic behavior

$$\frac{1}{n!}A(n,k) = \frac{e^{-B(\alpha)(n+1)}}{\sqrt{2\pi\sigma^2_\alpha(n+1)}}(1+o(1)) \tag{9}$$

where

$$\frac{k+1}{n+1} = \frac{e^\alpha}{e^\alpha - 1} - \frac{1}{\alpha} \tag{10}$$

$$B(\alpha) = \frac{1 - e^\alpha + \alpha e^\alpha}{e^\alpha - 1} + \log\frac{\alpha}{e^\alpha - 1} \tag{11}$$

$$\sigma^2_\alpha = \frac{1}{\alpha^2} - \frac{e^\alpha}{(e^\alpha - 1)^2}. \tag{12}$$

From (10), we have

$$\lim_{\alpha\to-\infty}\frac{k+1}{n+1} = 0, \qquad \lim_{\alpha\to+\infty}\frac{k+1}{n+1} = 1.$$

The function $\frac{1}{n!}A(n,k)$ attains its maximum at $k = n/2$ for even n and $k = (n\pm1)/2$ for odd n, corresponding to $\alpha \to \pm 0$. It is symmetric, monotonically decreasing around its maximum, and reaches the value $1/n!$ at $k = 0$ and $k = n-1$, corresponding to $\alpha \to \pm\infty$. We are interested in studying the behavior of $\frac{1}{n!}A(n,k)$ near its maximum.

Studying the function $B(\alpha)$ we can see that it has a single minimum at $\alpha = 0$ and that

$$\lim_{\alpha\to 0} B(\alpha) = 0, \qquad \lim_{\alpha\to\pm\infty} B(\alpha) = +\infty.$$

By taking the first two terms in the Taylor series expansion of (10) we can obtain a lower bound on α for $\frac{k+1}{n+1} > \frac{1}{2}$,

$$\alpha \ge 12\frac{k+1}{n+1} - 6. \tag{13}$$

Using (13) we can obtain a simple lower bound on (11) for $\frac{k+1}{n+1} > \frac{1}{2}$ and sufficiently large n as shown in Fig. 5. This is

---

if $\left\lceil (1+\delta)\frac{l}{2} \right\rceil < t \le l$,

$$P_C\left(t, \left\lceil (1+\delta)\frac{l}{2} \right\rceil\right) > \max\left\{\frac{1}{2}, 1 - \left(t - \left\lceil (1+\delta)\frac{l}{2} \right\rceil\right) \frac{e^{-6\left(\frac{\lceil (1+\delta)\frac{l}{2} \rceil + 2}{t+2} - \frac{1}{2}\right)^2 (t+2)}}{\sqrt{\frac{1}{6}\pi(t+2)}}\right\}$$

and

if $t \le \left\lceil (1+\delta)\frac{l}{2} \right\rceil$,

$$P_C\left(t, \left\lceil (1+\delta)\frac{l}{2} \right\rceil\right) = 1.$$

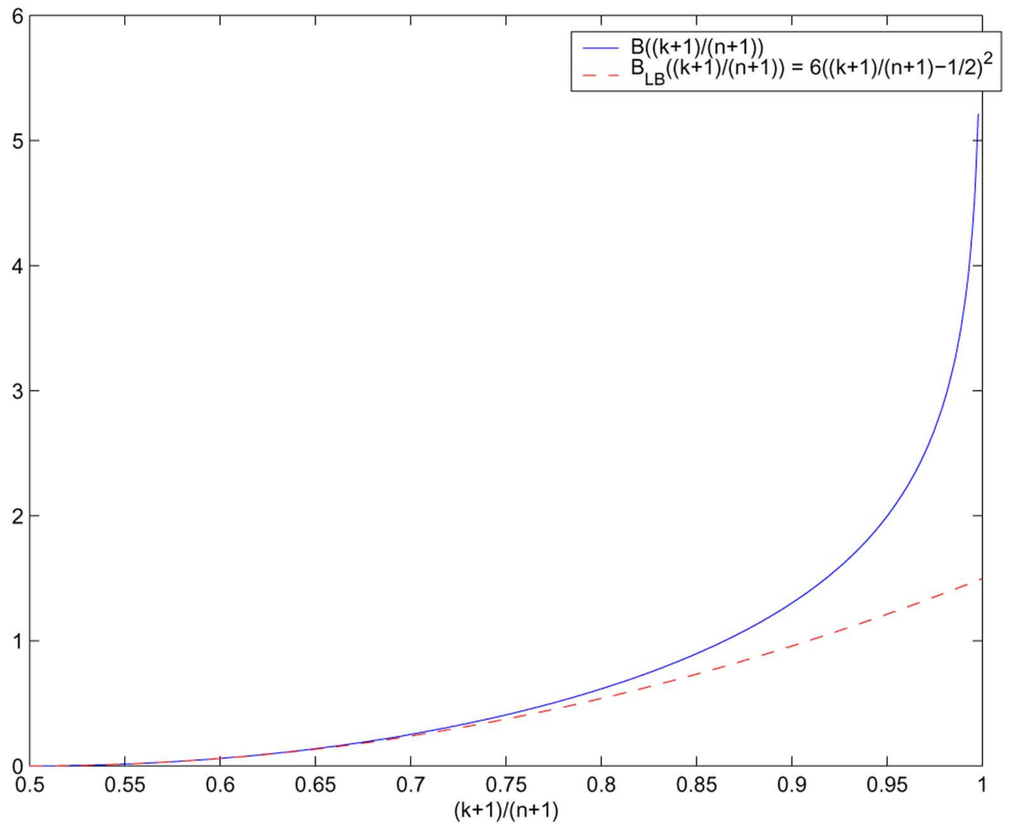


Fig. 5. $B(k,n)$ and its lower bound.

done by taking the first term in the Taylor series expansion of $B(\alpha)$, and substituting (13) instead of α

$$B(k,n) \geq 6\left(\frac{k+1}{n+1}-\frac{1}{2}\right)^2. \qquad (14)$$

We can now use (14) to upper-bound (9) for $\frac{k+1}{n+1} > \frac{1}{2}$ and n sufficiently large. To further simplify the expression, we use the first term in the series expansion of $\sigma_\alpha^2 = \frac{1}{12} + O(\alpha^2)$ around $\alpha = 0$, in order to obtain a tight bound near the maximum of $\frac{1}{n!}A(n,k)$ (i.e., near $k \to n/2$)

$$\frac{1}{n!}A(n,k) \leq \frac{e^{-6\left(\frac{k+1}{n+1}-\frac{1}{2}\right)^2(n+1)}}{\sqrt{\frac{1}{6}\pi(n+1)}}. \qquad (15)$$

Note, that although we prove (15) only for large n, we observe that it is also valid for any $n < 200$ (hence, we conjecture that it is valid for any n).

From Proposition III-5 and (8) we have the expression at the bottom of the page. For $\lceil(1+\delta)\frac{l}{2}\rceil < t \leq l$ we can use the monotonicity of $A(t+1,k)$ and (15) to get

$$\begin{aligned}&\frac{\sum_{k=\lceil(1+\delta)\frac{l}{2}\rceil+1}^{t} A(t+1,k)}{(t+1)!}\\ &\quad< \left(t-\left\lceil(1+\delta)\frac{l}{2}\right\rceil\right)\frac{A\left(t+1,\lceil(1+\delta)\frac{l}{2}\rceil+1\right)}{(t+1)!}\\ &\quad\leq \left(t-\left\lceil(1+\delta)\frac{l}{2}\right\rceil\right)\frac{e^{-6\left(\frac{\lceil(1+\delta)\frac{l}{2}\rceil+2}{t+2}-\frac{1}{2}\right)^2(t+2)}}{\sqrt{\frac{1}{6}\pi(t+2)}}.\end{aligned}$$

We can further simplify the bound on $P_C(t,\lceil(1+\delta)\frac{l}{2}\rceil)$ for $\lceil(1+\delta)\frac{l}{2}\rceil < t \leq l$

$$\begin{aligned}&P_C\left(t,\left\lceil(1+\delta)\frac{l}{2}\right\rceil\right)\\ &\quad> P_C\left(l,\left\lceil(1+\delta)\frac{l}{2}\right\rceil\right)\end{aligned}$$

$$P_C\left(t,\left\lceil(1+\delta)\frac{l}{2}\right\rceil\right) > \max\left\{\frac{1}{2}, 1-\frac{\sum\limits_{k=\lceil(1+\delta)\frac{l}{2}\rceil+1}^{t} A(t+1,k)}{(t+1)!}\right\}, \qquad \text{if} \quad \left\lceil(1+\delta)\frac{l}{2}\right\rceil < t \leq l$$

and

$$P_C\left(t,\left\lceil(1+\delta)\frac{l}{2}\right\rceil\right) = 1, \qquad \text{if} \quad t \leq \left\lceil(1+\delta)\frac{l}{2}\right\rceil.$$

$$> \max\left\{\frac{1}{2}, 1-\frac{l}{2}\frac{e^{-6\left(\frac{\lceil(1+\delta)\frac{l}{2}\rceil+2}{l+2}-\frac{1}{2}\right)^2(l+2)}}{\sqrt{\frac{1}{6}\pi(l+2)}}\right\}$$
$$\simeq \max\left\{\frac{1}{2}, 1-\sqrt{\frac{3l}{2\pi}}e^{-\frac{3}{2}\delta^2(l+2)}\right\} \qquad \square$$

Let $P_{L|C}(j,l)=\frac{P_L(j,l)}{P_C(j,l)}$ denote the expected fraction of leaf variable nodes in tier $2j$ of a tree in $\mathcal{T}^{(l)}_{\text{serial}}$. Next, we derive an upper bound on $P_{L|C}$ that is used in Section IV for asymptotic analysis of the serial schedule's convergence rate.

Lemma III-.7: For any $\delta>0$, there exists l' such that for any $l>l'$ and any $j\le l$, the expected fraction of variable node leaves in tier $2j$ of $T_{\text{serial}}^{(\lceil(1+\delta)\frac{l}{2}\rceil)}$ is upper-bounded by

$$P_{L|C}\left(j, \lceil(1+\delta)\frac{l}{2}\rceil\right) < \frac{e^{-6\left(\frac{\lceil(1+\delta)\frac{l}{2}\rceil+1}{j+2}-\frac{1}{2}\right)^2(j+2)}}{\sqrt{\frac{1}{24}\pi(j+2)}}, \quad \text{if} \quad \left\lceil(1+\delta)\frac{l}{2}\right\rceil\le j\le l$$

and bounded by

$$P_{L|C}\left(j,\left\lceil(1+\delta)\frac{l}{2}\right\rceil\right)=0, \qquad \text{if} \quad j<\left\lceil(1+\delta)\frac{l}{2}\right\rceil.$$

Thus, for any $j\le l$

$$P_{L|C}\left(j,\left\lceil(1+\delta)\frac{l}{2}\right\rceil\right) < \frac{e^{-6\left(\frac{\lceil(1+\delta)\frac{l}{2}\rceil+1}{l+2}-\frac{1}{2}\right)^2(l+2)}}{\sqrt{\frac{1}{24}\pi(l+2)}}$$
$$\triangleq \beta(l,\delta)\simeq\sqrt{\frac{24}{\pi(l+2)}}e^{-\frac{3}{2}\delta^2(l+2)} \to 0.$$

Proof: From Proposition III-5 we have

$$P_{L|C}\left(j,\left\lceil(1+\delta)\frac{l}{2}\right\rceil\right) < \frac{A\left(j+1,\lceil(1+\delta)\frac{l}{2}\rceil\right)}{\sum_{k=0}^{\lceil(1+\delta)\frac{l}{2}\rceil-1} A(j+1,k)}, \quad \text{if} \quad \left\lceil(1+\delta)\frac{l}{2}\right\rceil\le j\le l$$

and

$$P_{L|C}\left(j,\left\lceil(1+\delta)\frac{l}{2}\right\rceil\right)=0, \quad \text{if}\, j<\left\lceil(1+\delta)\frac{l}{2}\right\rceil.$$

Hence, using Theorem III-6 and (15) we conclude that for any $\delta>0$ there exists l' such that for any $l>l'$ and any $\lceil(1+\delta)\frac{l}{2}\rceil\le j\le l$

$$P_{L|C}\left(j,\left\lceil(1+\delta)\frac{l}{2}\right\rceil\right) < \frac{\frac{e^{-6\left(\frac{\lceil(1+\delta)\frac{l}{2}\rceil+1}{j+2}-\frac{1}{2}\right)^2(j+2)}}{\sqrt{\frac{1}{6}\pi(j+2)}}}{\frac{1}{2}}.$$

Finally, since $P_{L|C}\left(j,\lceil(1+\delta)\frac{l}{2}\rceil\right)$ is an increasing function of j, we have for any $j\le l$

$$P_{L|C}\left(j,\left\lceil(1+\delta)\frac{l}{2}\right\rceil\right)\le P_{L|C}\left(l,\left\lceil(1+\delta)\frac{l}{2}\right\rceil\right)$$
$$< \frac{e^{-6\left(\frac{\lceil(1+\delta)\frac{l}{2}\rceil+1}{l+2}-\frac{1}{2}\right)^2(l+2)}}{\sqrt{\frac{1}{24}\pi(l+2)}}$$
$$\triangleq \beta(l,\delta)$$

and

$$\beta(l,\delta)\cong\sqrt{\frac{24}{\pi(l+2)}}e^{-\frac{3}{2}\delta^2(l+2)}\to 0. \quad \square$$

Similar analysis can be performed for the computation tree evolution of the semi-serial schedule. Since the definition of an ascent also includes the case of two identical subsequent numbers, we can use Proposition III-3 in order to test whether a variable node v' belongs to a computation tree $T^{(l)}_{m\text{-serial}(\sigma)}$ spanned from an edge $e=(v,c)$ and whether it is a leaf of $T^{(l)}_{m\text{-serial}(\sigma)}$. Based on Proposition III-3, we can obtain an expression for $P_C(t,l)$ for the semi-serial computation tree, however, the expression involves cumbersome recursions and is hard to analyze. Instead, we show that $P_C(t,l)^{m\text{-serial}}$ tends to $P_C(t,l)^{\text{serial}}$ as the number of subsets m increases, indicating that the semi-serial schedule should have similar convergence rate as the serial schedule. This is not surprising since the semi-serial schedule becomes closer to the serial schedule as the number of subsets increases.

Proposition III-8:

$$\forall\, t\le 2l, \quad P_C(t,l)^{m\text{-serial}}\ge\left(\frac{m-2l}{m}\right)^{2l+1} P_C(t,l)^{\text{serial}} \tag{16}$$

For any $\delta>0$, if $m\ge\frac{2l}{1-(1-e^{-\delta l})^{1/(2l+1)}}$ then

$$\forall\, t\le 2l, \quad P_C(t,l)^{m\text{-serial}}\ge(1-e^{-\delta l})P_C(t,l)^{\text{serial}} \to P_C(t,l)^{\text{serial}}.$$

Proof: Consider a path $P_v^{v'}$ of length $2t$, where $t\le 2l$, in the computation tree $T^{(l)}_{m\text{-serial}(\sigma)}$ spanned from the edge $e=(v,c)$. Let w denote the sequence of numbers representing the subset numbers of the variables along the path $P_v^{v'}$. Let p denote the probability that no number appears in w more than once. Then

$$p=\frac{\prod_{i=0}^{t} m-i}{m^{t+1}}\ge\left(\frac{m-2l}{m}\right)^{2l+1}. \tag{17}$$

Assuming $m\ge\frac{2l}{1-(1-e^{-\delta l})^{1/(2l+1)}}$, we get

$$p\ge 1-e^{-\delta l}. \tag{18}$$

Using (17) we can lower-bound the probability that v' is in $T^{(l)}_{m\text{-serial}(\sigma)}$ by

$$P_C(t,l)^{m\text{-serial}}\ge p\cdot P_C(t,l)^{\text{serial}}+(1-p)\cdot 0$$
$$\ge(1-e^{-\delta l})P_C(t,l)^{\text{serial}}. \qquad \square$$

Note that the analysis in this section has not assumed anything about the bipartite graph which generates the computation tree and applies to both regular and irregular graphs.

IV. Asymptotic Convergence Analysis

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 below some constant threshold. This phenomenon, first observed by Gallager for binary-symmetric channel channels [8], was shown to exist in irregular LDPC codes by Luby *et al.* [14], [15] and was further generalized by Richardson and Urbanke [20] to a large range of symmetric, memoryless, binary-input channels and to various message-passing decoding algorithms. Richardson *et al.* developed an algorithm for determining the threshold for various message-passing decoders based on flood scheduling over various channels called **density evolution** (DE) [20], [21], [5]. The algorithm is based on iterative calculation of the densities of messages passed by the decoder in each iteration. The calculation is simplified tremendously by taking advantage of several assumptions as follows.

- The "local tree assumption"—the local neighborhood of an edge is cycle-free. This assumption is justified by the general concentration theorem that was proved in [20] 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.
- The channel is memoryless. Combined with the local tree assumption, this assumption implies that the messages entering a node are statistically independent.
- The zero codeword was transmitted—as shown in [20], for symmetric channels the decoder's performance is independent of the transmitted codeword.

Let f_P, $f_Q^{(l)}$, and $f_R^{(l)}$ denote the conditional probability density functions (pdf) assuming the zero codeword was transmitted, of a channel message and a variable-to-check and check-to-variable messages at the lth iteration, respectively.

Theorem IV-9: DE of the flooding schedule for a $(\lambda(x),\rho(x))$-LDPC codes ensemble is given by [5], [20], [21]

$$f_Q^{(l)} = f_P \bigotimes \lambda\left(\Gamma^{-1}\left(\rho\left(\Gamma\left(f_Q^{(l-1)}\right)\right)\right)\right)$$

where, $f_Q^{(0)} = f_P$. □

Here

$$\lambda(f) = \sum_{i=2}^{d_v} \lambda_i \bigotimes^{i-1} f \quad \text{and} \quad \rho(f) = \sum_{j=2}^{d_c} \rho_j \bigotimes^{j-1} f.$$

The operator $\bigotimes$ denotes convolution. The operator Γ is defined so that for a real random variable X with density f_X, the density of $\varphi(X)$ is $\Gamma(f_X)$. The DE algorithm is run iteratively, starting with a fixed channel parameter (determining f_P), until either the pdf f_Q tends to "point mass at infinity" (corresponding to zero error probability) or converges to a fixed point of the pdf with error probability greater than zero (defined by the probability of Q being negative). If we are interested in the threshold computation, the flooding DE algorithm can be used for analysis of the serial schedule as well, since under the cycle-free graph assumption all possible schedules converge to the exact *a posteriori* value after a finite number of iterations. Thus, the threshold under the flooding schedule and the serial schedule is the same. 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 used even under the cycle-free graph assumption.

A. Density Evolution Algorithm for the Semi-Serial Schedule

In this subsection, we derive a DE algorithm for asymptotic analysis of the semi-serial schedule. Similarly to the flooding DE, the analysis is based on the "local tree" assumption, justified by the general concentration theorem proved in Section IV-C and on the assumption that the channel is memoryless and symmetric.

According to Definition III-2 of the semi-serial schedules ensemble, a schedule is generated by assigning a subset B_i to each variable node, where i is chosen independently and randomly with equal probability from the set $\{1,\ldots,m\}$ for each node. The subsets are updated serially in each iteration (i.e., first B_1, then B_2, and so on, until B_m). When a subset is updated, messages are simultaneously sent to all of its variable nodes and then messages are simultaneously sent from all of its variable nodes. Let $f_{Q_i}^{(l)}$ denote the expected pdf of a variable-to-check message sent from a variable node belonging to the ith subset at the lth 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. A variable-to-check message from a variable node belonging to the subset B_i is a function of messages that emanated from the subsets that have already been updated $(B_j, j < i)$ and of messages from the subsets that have not been updated yet $(B_j, j \geq i)$. Taking into account that subset assignments are chosen independently and with equal probability and that messages entering a node are statistically independent and identically distributed (due to standard DE assumptions) we get the following algorithm.

Theorem IV-A.1: DE of a semi-serial schedule with m subsets for a $(\lambda(x),\rho(x))$-LDPC codes ensemble is given by the equation at the top of the following page.

B. Density Evolution Algorithm for the Serial Schedule

Let $\mathcal{T}_{\text{serial}}^{(l)} = \{T_{\text{serial}(\sigma)}^{(l)} : \sigma \in S\}$ denote the ensemble of computation trees spanned from edge e of a given cycle-free graph G after l iterations of the serial schedule. Knowing whether a variable node belongs to a computation tree $T_{\text{serial}}^{(l)} = T_{\text{serial}(\sigma)}^{(l)}$ affects the probability that other variable nodes belong to the computation tree. For example, consider variable node v at tier t of $T_{\text{serial}}^{(l)}$, and assume it is a descendant of variable node v_f at tier $(t-2)$. We know that v has probability $P_{L|C}(j,l)$ to be a leaf of $T_{\text{serial}}^{(l)}$. 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 $P_{L|C}(j,l)$ and *vice versa*. Thus, variable nodes which are descendants of the same ancestor are

$$f_{Q_i}^{(l)} = f_P \bigotimes \lambda \left(\Gamma^{-1}\left(\rho\left(\Gamma\left(\frac{1}{m}\left(\sum_{j=1}^{i-1} f_{Q_j}^{(l)} + \sum_{j=i}^{m} f_{Q_j}^{(l-1)}\right)\right)\right)\right)\right), \qquad i = 1,\dots,m$$

$$f_Q^{(l)} = \frac{1}{m}\sum_{i=1}^{m} f_{Q_i}^{(l)},$$

where

$$f_{Q_i}^{(0)} = f_P \quad i = 1,\dots,m. \qquad \square$$

positively correlated. Unfortunately, due to these statistical dependencies between variable nodes it is hard to analyze the computation trees ensemble $\mathcal{T}_{\text{serial}}^{(l)}$. In order to overcome this problem, we consider a *pseudoserial* ensemble of computation trees, denoted by $\mathcal{T}_{ps}^{(l,t)}$, which is more amenable to analysis. We show that the pseudoserial ensemble provides a lower bound on the asymptotic performance of the serial schedule in certain cases. We conjecture that this is true in general, i.e., for any ensemble over any channel.

We define the pseudoserial ensemble as follows.

Definition IV-B.1: 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 $j = 1, 2, 3, \dots, t-1$ do:
   for each undeleted variable node v at tier $2j$ decide independently with probability $P_{L|C}(j,l)$ if v is a leaf. If so, delete the subtree below v.

Note that $P_{L|C}(j,l) = 0$ for $j = 1, 2, \dots, l-1$, so the first $(l-1)$ steps in the procedure for generating a pseudo-serial computation tree can be skipped.

Let $p_{\text{flood}}^{(l)}$ denote the expected fraction of incorrect or erased messages passed along an arbitrary edge with an acyclic computation tree at iteration l of the flooding BP decoder, averaged over the bipartite graph ensemble, the graph edges, and the decoder inputs. Note that the flooding DE algorithm [20], [21], described in the beginning of this section, computes $p_{\text{flood}}^{(l)} = \int_{-\infty}^{0} f_Q^{(l)}(q)dq$. Let $p_{\text{serial}}^{(l)}$ denote the expected fraction of incorrect or erased messages passed along an arbitrary edge with an acyclic computation tree at iteration l of the serial BP decoder, averaged over the bipartite graph ensemble, the graph edges, the $|V|!$ serial schedules, and the decoder inputs. Let $p_{ps}^{(l,t)}$ denote the expected fraction of incorrect or erased messages computed by a pseudo-serial computation tree, averaged over the bipartite graph ensemble, the graph edges, their computation tree ensemble in $\mathcal{T}_{ps}^{(l,t)}$, and the decoder inputs.

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 off from the tree 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 $2j$. Then $f_Q^{(2j)}$ can be computed as a function of $f_Q^{(2j+2)}$ based on Definition IV-B.1 of the ensemble $\mathcal{T}_{ps}^{(l,t)}$.

Theorem IV-B.2: 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:

$$f_Q^{(2j)} = P_{L|C}(j,l)f_P + \left(1 - P_{L|C}(j,l)\right) f_P \bigotimes \lambda\left(\Gamma^{-1}\left(\rho\left(\Gamma\left(f_Q^{(2j+2)}\right)\right)\right)\right), \quad j = t-1,\dots,0$$

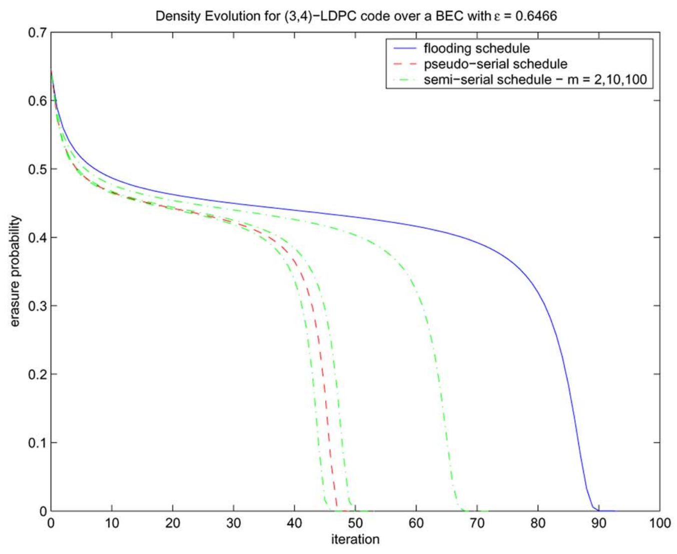
and $f_Q^{(2t)} = f_P$. $\square$

The fraction of erroneous messages as a function of the number of iterations as predicted by the DE 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 BEC with channel parameter $\epsilon = 0.6466$, and for a $(3,6)$-LDPC code over a Gaussian channel with channel parameter $E_b/N_0 = 1.12$ dB are shown in Fig. 6. We applied the DE algorithms for analysis of various regular and irregular LDPC ensembles over the BEC and the additive white Gaussian noise (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 capacity of the ensemble. 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 III-8. Finally, the results provide a strong indication that the convergence rate of the pseudo-serial schedule can serve as a lower bound on the one 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, generally proving this claim is hard. Hence, we prove it for a $(x, \rho(x))$-LDPC ensemble over the BEC.

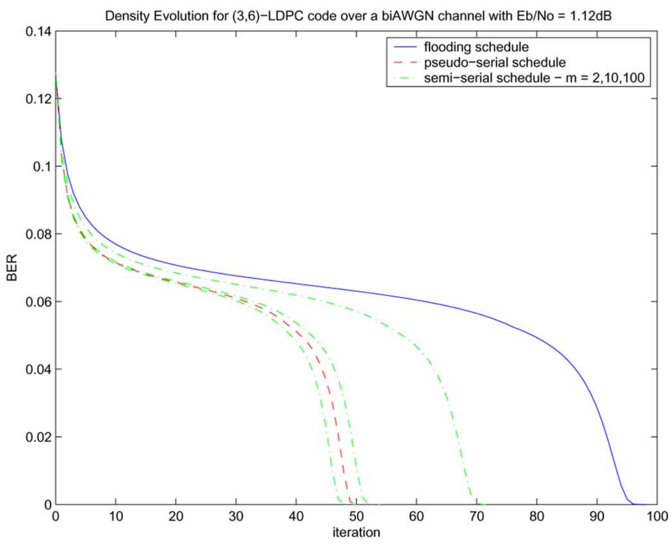
Proposition IV-B.3: For $(x, \rho(x))$-LDPC ensemble over the BEC and for any $t > 0$

$$p_{\text{serial}}^{(l)} \le p_{ps}^{(l,t)}.$$

Proof: Let $T_{\text{trunc}(G,\sigma)}^{(l,t)}$ denote the serial computation tree of the edge $e = (v,c)$ in the graph G and serial schedule σ after l iterations, truncated at tier t. Obviously, $T_{\text{trunc}(G,\sigma)}^{(l,t)}$ is a



(a)



(b)

Fig. 6. (a) Density evolution for a $(3,4)$-LDPC code over the BEC channel. (b) Density evolution for a $(3,6)$-LDPC code over the Gaussian channel.

degraded computation tree compared to the nontruncated serial computation tree $T^{(l)}_{\text{serial}(G,\sigma)}$ with respect to its expected erasure probability for any G and σ. Hence, $p^{(l)}_{\text{serial}} \le p^{(l,t)}_{\text{trunc}}$.

Consider BEC with erasure probability α. Let v' denote a variable node at tier $2j$ of $T^{(l,t)}_{\text{trunc}(G,\sigma)}$ and c' denote the neighboring check node of v' at tier $2j+1$. Let $I(v',G,\sigma)$ denote an indicator function for the event that v' is a leaf of $T^{(l,t)}_{\text{trunc}(G,\sigma)}$. Let $q^{(2j)}$ denote the expected erasure probability of a variable-to-check message sent from tier $2j$ of $T^{(l,t)}_{\text{trunc}(G,\sigma)}$, averaged over the $(x,\rho(x))$-ensemble graphs, the serial schedules and the decoder inputs. Then $q^{(2j)}$ is given by (19) at the bottom of the page. Here $q_i^{(2j+2)}$ is the expected erasure probability of a variable-to-check message sent from the ith descendant of variable node v' at tier $2j+2$, averaged over all decoder inputs. Due to the positive correlation between neighboring variable nodes in the serial schedule we can upper bound (19) by

$$\begin{aligned} q^{(2j)} &\le P_{L|C}(j,l)\alpha + (1-P_{L|C}(j,l))\alpha \\ &\quad \times \sum_{j=1}^{d_c} \rho_j \left(1 - \prod_{i=1}^{j-1}\left(1 - \mathbf{E}_G\mathbf{E}_\sigma\mathbf{E}_{v'}\left[q_i^{(2j+2)}\right]\right)\right) \\ &= P_{L|C}(j,l)\alpha + (1-P_{L|C}(j,l))\alpha \\ &\quad \times \sum_{j=1}^{d_c} \rho_j \left(1 - (1-q^{(2j+2)})^{j-1}\right) \end{aligned} \tag{20}$$

where $q^{(2t)} = \alpha$.

Since (20) describes an identical recursion to the one defined by Theorem IV-B.2, we get

$$p^{(l)}_{\text{serial}} \le p^{(l,t)}_{\text{trunc}} = q^{(0)} \le p^{(l,t)}_{ps} \qquad \square$$

We note that this proof holds only for ensembles with degree 2 variable nodes, for which information combining is performed only at the check nodes.

Motivated by the DE results and by Proposition IV-B.3 we conjecture the following.

Conjecture IV-B.4: For any integers $l>0$ and $t>0$

$$p^{(l)}_{\text{serial}} \le p^{(l,t)}_{ps}$$

regardless of the graph ensemble and the 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 BEC parameterized by erasure probability α. The BEC is monotone with respect to the BP decoder [20], i.e., convergence of the decoder for parameter α implies convergence of the decoder for any parameter $\alpha' \le \alpha$. Let α^* denote the decoder's threshold as predicted by the DE algorithm. The value α^* is given by [14], [15]

$$\alpha^* = \sup_\alpha\{\alpha : f_{\text{DE}}(x) < x \text{ for } x \in (0,\alpha)\} \tag{21}$$

$$\begin{aligned} q^{(2j)} &= \mathbf{E}_G\mathbf{E}_\sigma\mathbf{E}_{v'}\left[I(v',G,\sigma)\alpha + (1-I(v',G,\sigma))\alpha\left(1 - \prod_{i=1}^{|N(c')\backslash v'|}\left(1-q_i^{(2j+2)}\right)\right)\right] \\ &= P_{L|C}(j,l)\alpha + (1-P_{L|C}(j,l))\alpha\sum_{j=2}^{d_c}\rho_j\left(1 - \mathbf{E}_G\mathbf{E}_\sigma\mathbf{E}_{v'}\left[\prod_{i=1}^{j-1}\left(1-q_i^{(2j+2)}\right)\right]\right). \end{aligned} \tag{19}$$

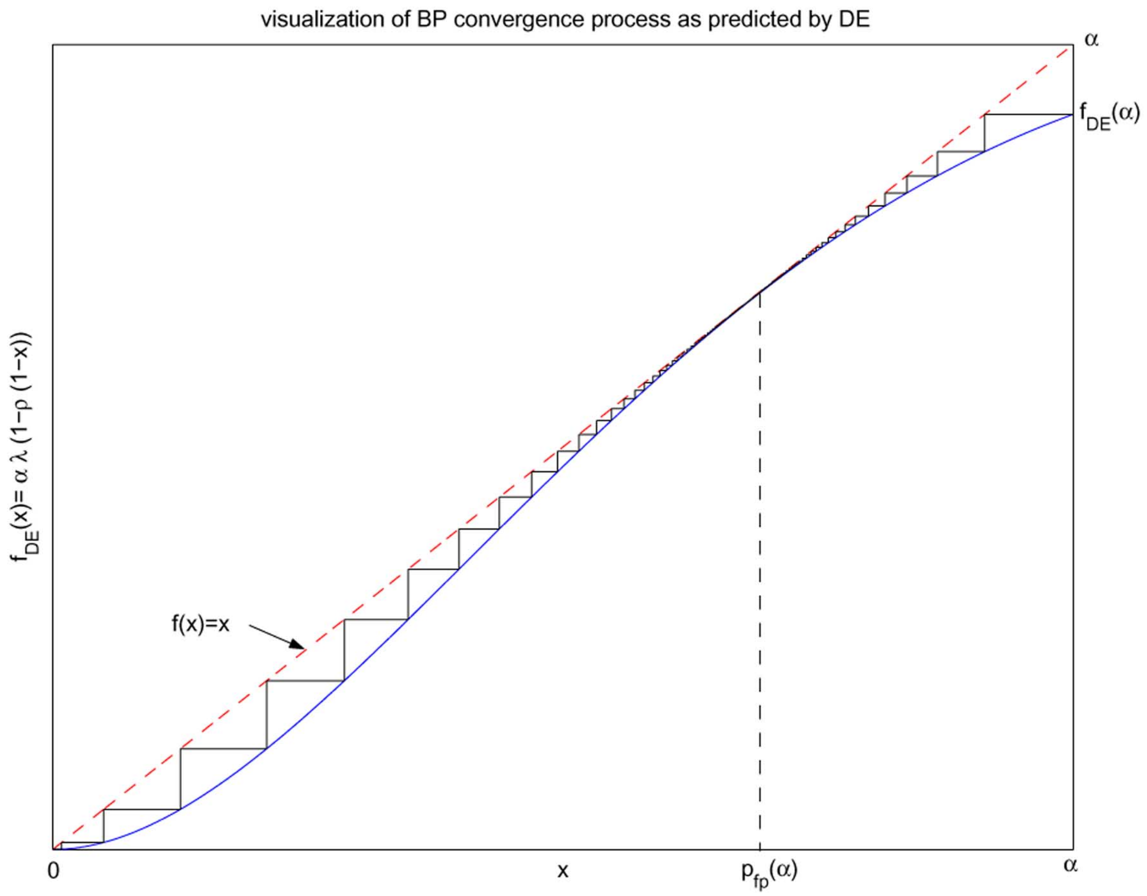


Fig. 7. Visualization of the BP convergence process over a BEC with parameter α as predicted by the DE algorithm.

where, $f_{\mathrm{DE}}(x) = \alpha \cdot \lambda(1-\rho(1-x))$ is the BEC version of the DE recursion function defined in Theorem III-9. Note that α^* is independent of the particular scheduling applied by the decoder, since for cycle-free graphs, the decoder converges to the exact *a posteriori* probabilities after a finite number of iterations for any schedule.

Let $p_{fp}(\alpha)$ be defined as

$$p_{fp}(\alpha) = \arg_{x:x\in(0,\alpha)} \min\{x - \alpha \cdot \lambda(1-\rho(1-x))\}. \quad (22)$$

Then $p_{fp}(\alpha^*)$ is the erasure probability of a variable-to-check message at which fixed point of the DE algorithm (Theorem IV-9) occurs. At this point we have

$$p_{fp}(\alpha^*) - \alpha^* \cdot \lambda(1-\rho(1-p_{fp}(\alpha^*))) = 0. \quad (23)$$

The convergence process of the flooding BP decoder as predicted by the BEC DE algorithm can be visualized on a graph showing the variable-to-check message erasure probability at iteration j as a function of the erasure probability at iteration $(j-1)$ as shown in Fig. 7.

Let $l(\alpha, p)$ denote the minimal number of iterations required by the decoder to achieve an expected message erasure probability less or equal to p over an arbitrary edge with an acyclic computation tree, when working over a BEC with parameter α.

Theorem IV-B.5: For any $\epsilon > 0$ and $\delta > 0$, there exists $\alpha' \leq \alpha^*$ such that for any $\alpha \in [\alpha', \alpha^*]$ and for any $p \in (0, p_{fp}(\alpha))$:

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

For proving Theorem IV-B.5 we need several auxiliary lemmas.

Lemma IV-B.6: For any $p \in (0, p_{fp}(\alpha))$

$$\lim_{\alpha\to\alpha^*} l_{\mathrm{flood}}(\alpha,p) = \infty. \quad (24)$$

Proof: Based on (23) and since $p_{fp}(\alpha)$ is a continuous function of α, we have

$$\lim_{\alpha\to\alpha^*} p_{fp}(\alpha) - \alpha \cdot \lambda(1-\rho(1-p_{fp}(\alpha))) = 0. \quad (25)$$

Assume that at some point the expected variable-to-check message erasure probability is p, then the expected erasure probability will decrease in the following iteration by $p - f_{\mathrm{DE}}(p)$ (see Fig. 7). Since $f_{\mathrm{DE}}(p) = \alpha\lambda(1-\rho(1-p))$ is a continuous function of p and due to (25), near $p_{fp}(\alpha)$ the decrease in erasure probability in each iteration approaches zero as α approaches α^*, and the number of iterations for attaining any $p \in (0, p_{fp}(\alpha))$ tends to infinity. □

Lemma IV-B.7: For any $p \in (0, p_{fp}(\alpha))$ and any $\gamma > 0$

$$\lim_{\alpha\to\alpha^*} \frac{L(\gamma,l)}{l_{\mathrm{flood}}(\alpha,p)} = 0$$

where $L(\gamma,l) = \left|\left\{i \in \{0,\ldots,l\} : f'_{\mathrm{DE}}\left(p^{(i)}_{\mathrm{flood}}\right) > e^{\gamma}\right\}\right|$.

Proof of Lemma IV-B.7: Consider a segment $[x_1, x_2]$ on which $f'_{\mathrm{DE}}(x) > e^{\gamma} > 1$. By definition (22) of the fixed point $x = p_{fp}(\alpha)$, we have $f'_{\mathrm{DE}}(x)\ |_{x=p_{fp}(\alpha)} = 1$. Hence,

$p_{fp}(\alpha) \notin [x_1, x_2]$. Furthermore, if the stability condition [21] is satisfied, i.e., if $f'_{\text{DE}}(x)|_{x=0} = \alpha\lambda'(0)\rho'(1) < 1$, then the fixed point $x = 0$ is also not in $[x_1, x_2]$. Thus, the number of iterations within $[x_1, x_2]$ is bounded (it can be bounded by an explicit function of $\underline{\lambda}$, $\underline{\rho}$, α^*, and γ) and $L(\gamma, l)$ is bounded for any α. Use of Lemma IV-B.6 completes the proof. $\square$

Lemma IV-B.8: For any $\delta > 0$ and $\epsilon > 0$, there exists l' such that for any $l > l'$

$$p_{ps}^{\left(\left\lceil(1+\delta)\frac{l}{2}\right\rceil\right)} \le p_{\text{flood}}^{(l)} + \epsilon.$$

Proof: For a given $(\lambda(x), \rho(x))$-LDPC ensemble, decoded using BP decoder over BEC with erasure probability α, we can compute $p_{ps}^{\left(\left\lceil(1+\delta)\frac{l}{2}\right\rceil, l\right)}$ using the DE algorithm described in Theorem IV-B.2. Let $q_{ps}^{(2j)}$ denote the expected erasure probability of a variable-to-check message sent from tier $2j$ to tier $2j-1$ of $T_{ps}^{\left(\left\lceil(1+\delta)\frac{l}{2}\right\rceil, l\right)}$. Then, $p_{ps}^{\left(\left\lceil(1+\delta)\frac{l}{2}\right\rceil, l\right)} = q_{ps}^{(0)}$ and $q_{ps}^{(0)}$ is computed according to the following recursion:

$$\begin{aligned} q_{ps}^{(2j)} =& P_{L|C}\left(j, \left\lceil(1+\delta)\frac{l}{2}\right\rceil\right)\alpha, \qquad j = l-1, \ldots, 0 \\ &+ \left(1 - P_{L|C}\left(j, \left\lceil(1+\delta)\frac{l}{2}\right\rceil\right)\right) f_{\text{DE}}\left(q_{ps}^{(2j+2)}\right) \quad (26)\end{aligned}$$

and $q_{ps}^{(2l)} = \alpha$.

Similarly, let $q_{\text{flood}}^{(2j)}$ denote the expected erasure probability of a variable-to-check message sent from tier $2j$ to tier $2j-1$ of $T_{\text{flood}}^{(l)}$. Then, $p_{\text{flood}}^{(l)} = q_{\text{flood}}^{(0)}$ and $q_{\text{flood}}^{(0)}$ can be computed using the recursion

$$q_{\text{flood}}^{(2j)} = f_{\text{DE}}\left(q_{\text{flood}}^{(2j+2)}\right), \qquad j = l-1, \ldots, 0 \qquad (27)$$

where $q_{\text{flood}}^{(2l)} = \alpha$.

Let $\Delta_{2j} = q_{ps}^{(2j)} - q_{\text{flood}}^{(2j)}$ denote the difference between the expected erasure probability of variable-to-check messages sent from tier $2j$ of $T_{ps}^{\left(\left\lceil(1+\delta)\frac{l}{2}\right\rceil, l\right)}$ and $T_{\text{flood}}^{(l)}$. Note that $\Delta_{2j} \ge 0$ for any $0 \le j \le l$ and $\Delta_{2l} = 0$. Then

$$\begin{aligned} q_{ps}^{(2j)} &= P_{L|C}\left(j, \lceil(1+\delta)\frac{l}{2}\rceil\right)\alpha \\ &\quad + \left(1 - P_{L|C}\left(j, \left\lceil(1+\delta)\frac{l}{2}\right\rceil\right)\right) f_{\text{DE}}\left(q_{ps}^{(2j+2)}\right) \\ &= P_{L|C}\left(j, \left\lceil(1+\delta)\frac{l}{2}\right\rceil\right)\alpha \\ &\quad + \left(1 - P_{L|C}\left(j, \left\lceil(1+\delta)\frac{l}{2}\right\rceil\right)\right) \\ &\quad \times f_{\text{DE}}\left(q_{\text{flood}}^{(2j+2)} + \Delta_{2j+2}\right) \\ &\le \beta(l,\delta) + f_{\text{DE}}\left(q_{\text{flood}}^{(2j+2)}\right) \\ &\quad + f'_{\text{DE}}\left(q_{\text{flood}}^{(2j+2)}\right)\Delta_{2j+2} + m \\ &= \beta(l,\delta) + q_{\text{flood}}^{(2j)} + f'_{\text{DE}}\left(q_{\text{flood}}^{(2j+2)}\right)\Delta_{2j+2} + m \\ \Rightarrow \Delta_{2j} &\le \beta(l,\delta) + f'_{\text{DE}}\left(q_{\text{flood}}^{(2j+2)}\right)\Delta_{2j+2} + m \end{aligned}$$

where $m = \max_{x\in[0,1]} f''_{\text{DE}}(x)\Delta_{2j+2}^2$.

The third inequality follows from Proposition III-7 and from considering the first two terms in the Taylor series of $f_{\text{DE}}(x)$ around $q_{\text{flood}}^{(2j+2)}$.

The term $\max_{x\in[0,1]} f''_{\text{DE}}(x)$ can be upper-bounded by a constant C_1 depending only on $\underline{\lambda}$ and $\underline{\rho}$. Furthermore, we will show that Δ_{2j+2} can be made arbitrarily small as l increases. Hence, for any $\gamma > 0$ there exists l' such that for any $l > l'$ and any $0 \le j \le l$ we have

$$\Delta_{2j} \le \beta(l,\delta) + e^{\gamma} f'_{\text{DE}}\left(q_{\text{flood}}^{(2j+2)}\right)\Delta_{2j+2}. \qquad (28)$$

The recursion (28) can be unfolded and upper-bounded. Define $\prod_{j\in\emptyset} f(j) \equiv 1$, then for any $\gamma > 0$

$$\begin{aligned} \Delta_0 &= \beta(l,\delta)\left(\sum_{i=0}^{l-1}\prod_{j=1}^{i} e^{\gamma} f'_{\text{DE}}\left(q_{\text{flood}}^{(2j)}\right)\right) \\ &\le l\beta(l,\delta)e^{\gamma l} \prod_{j\in\{1,\ldots,l-1\}: f'_{\text{DE}}\left(q_{\text{flood}}^{(2j)}\right)>1} f'_{\text{DE}}\left(q_{\text{flood}}^{(2j)}\right) \\ &\le l\beta(l,\delta)e^{2\gamma l} \prod_{j\in\{1,\ldots,l-1\}: f'_{\text{DE}}\left(q_{\text{flood}}^{(2j)}\right)>e^{\gamma}} f'_{\text{DE}}\left(q_{\text{flood}}^{(2j)}\right). \end{aligned}$$

The maximal value of the derivative

$$\begin{aligned} f'_{\text{DE}}(x) &= (\alpha\cdot\lambda(1-\rho(1-x)))' \\ &= \alpha\cdot\lambda'(1-\rho(1-x))\cdot\rho'(1-x) \end{aligned}$$

can be bounded by a constant

$$\begin{aligned} &\max_{x\in[0,1]}\{\alpha\cdot\lambda'(1-\rho(1-x))\cdot\rho'(1-x)\} \\ &\quad \le \max_{x\in[0,1]}\{\lambda'(1-\rho(1-x))\cdot\rho'(1-x)\} \\ &\quad \equiv C_2(\underline{\lambda}, \underline{\rho}). \end{aligned}$$

Thus

$$\begin{aligned} \Delta_0 &\le l\beta(l,\delta)e^{2\gamma l} C_2(\underline{\lambda}, \underline{\rho})^{\left|\left\{j\in\{0,\ldots,l\}: f'_{\text{DE}}\left(q_{\text{flood}}^{(2j)}\right)>e^{\gamma}\right\}\right|} \\ &= l\beta(l,\delta)e^{2\gamma l} C_2(\underline{\lambda}, \underline{\rho})^{L(\gamma,l)}. \end{aligned}$$

Choosing $\gamma < \frac{3}{4}\delta^2$ and using Lemmas III-7 and IV-B.7 we get that for any $\delta > 0$ and $\epsilon > 0$ there exist l' such that for any $l > l'$

$$p_{ps}^{\left(\left\lceil(1+\delta)\frac{l}{2}\right\rceil, l\right)} - p_{\text{flood}}^{(l)} = q_{ps}^{(0)} - q_{\text{flood}}^{(0)} \le \epsilon. \qquad \square$$

Proof of Theorem IV-B.5: According to Lemma IV-B.6, for any l' there exists $\alpha' < \alpha^*$ such that for any $\alpha \in (\alpha', \alpha^*)$ and for any $p \in (0, p_{fp}(\alpha))$ we have $l_{\text{flood}}(\alpha, p) > l'$. Furthermore, according to Lemma IV-B.8, for any $\epsilon > 0$ and $\delta > 0$ there exists l' such that for any $l > l'$ we have $p_{ps}^{\left(\left\lceil(1+\delta)\frac{l}{2}\right\rceil\right)} \le p_{\text{flood}}^{(l)} + \epsilon$. Thus, for any $\epsilon > 0$ and $\delta > 0$ there exists $\alpha' < \alpha^*$ such that for any $\alpha \in (\alpha', \alpha^*)$ and for any $p \in (0, p_{fp}(\alpha))$ we have $l_{ps}(\alpha, p+\epsilon) \le \left\lceil(1+\delta)\frac{l_{\text{flood}}(\alpha,p)}{2}\right\rceil$. $\square$

Modulo Conjecture IV-B.4 for any LDPC ensemble, or using Proposition IV-B.3 for $(x, \rho(x))$-LDPC ensembles we get the following.

Corollary IV-B.9: For any $\epsilon > 0$ and $\delta > 0$, there exists $\alpha' \le \alpha^*$ such that for any $\alpha \in [\alpha', \alpha^*]$ and for any $p \in (0, p_{fp}(\alpha))$

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

C. Concentration Theorem

In the previous section, we saw how the expected behavior of a given semi-serially or serially scheduled decoder can be determined using the DE algorithm, by assuming that the graph does not contain cycles. In this section, we follow [20] and prove a concentration theorem for the semi-serial and serial schedules, showing that the behavior of the decoder for a randomly chosen graph, schedule, and decoder input is concentrated around its expected behavior.

For the sake of simplicity, we prove the concentration theorem for the ensemble $\mathcal{C}_n^{(d_v,d_c)}$ of regular (d_v,d_c)-LDPC codes of length n. Consider a message passed along an edge e during the lth decoding iteration. This message is a function of the computation tree $T_e^{(l)}$ and of the received channel observations for the variables contained in it. Let $D_e^{(l)}$ denote the depth of $T_e^{(l)}$. As discussed in Section III, the computation tree depth of any edge for the flooding schedule grows by 2 at each iteration, i.e., $D_e^{(l)} = 2l$. For the semi-serial schedule, the depth of the computation tree after l iterations is not fixed and depends on the graph, the schedule by which the sets of variable nodes are updated, and the specific edge. However, it is easy to see that it is bounded by $D_e^{(l)} \le 2ml$, where m is the number of subsets. Due to the bounded computation tree size a similar concentration theorem to the one proved in [20] for the flooding schedule applies to the semi-serial schedule.

Theorem IV-C.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 be the number of incorrect messages among all $d_v n$ variable-to-check messages sent in the lth decoding step. Furthermore, let p be the expected number of incorrect messages passed along an arbitrary edge with an acyclic computation tree at the lth decoding step, as determined by the DE algorithm. Then there exist positive constants β, γ, such that for any $\epsilon > 0$ and $n > \frac{2\gamma}{\epsilon}$

$$Pr\left(|Z/(nd_v) - p| > \epsilon\right) \le 2e^{-\beta\epsilon^2 n}. \tag{29}$$

Proof: Follows the lines of [20], except the final step, where Azuma's inequality [1] is used for showing concentration. This step is similar to the final step in the proof of the serial schedule's concentration theorem described below. The constant γ can be taken as $M_{ml}^2 + \frac{d_c}{d_v} C_{ml}^2$ where

$$M_{ml} = \sum_{i=0}^{ml} (d_v - 1)^i (d_c - 1)^i$$

and

$$C_{ml} = 1 + (d_v - 1) \sum_{i=0}^{ml-1} (d_v - 1)^i (d_c - 1)^i.$$

The constant β can be taken as $\frac{d_v^2}{(512d_v+32)(d_c d_v)^{2ml}}$. □

Unfortunately, for a serial schedule, the computation tree of an edge cannot be bounded by a constant and can grow with n. As a result, the flooding concentration theorem [20] cannot be applied directly to the serial schedule and some additional arguments are needed. The resulting theorem that we were able to prove is weaker, showing the concentration with a probability approaching 1 inverse polynomially in n (instead of exponentially). However, we believe that an exponential concentration is valid for the serial schedule as well.

We start by proving some auxiliary lemmas.

Lemma IV-C.2: For a code with sufficiently large length n and for a randomly chosen serial schedule

$$Pr\left(\max_e\left\{D_e^{(l)}\right\} \ge \frac{4l\ln(n)}{\ln(\ln(n))}\right) \le \frac{1}{n^{1-o(1)}}. \tag{30}$$

Proof: We consider the first iteration and bound the probability that the computation tree spanned from edge $e = (v, c)$ is of depth greater than or equal to $2r$. Let P_v^u denote a path from v to another variable node u at graphical distance $2r$ from v. The probability that the computation tree $T_e^{(1)}$ contains u is the probability that the $r+1$ variable nodes along the path P_v^u are unique (no repetitions) and are updated one after another starting from u and ending in v. This probability is smaller then $\frac{1}{r!}$. Note, that in case some variable nodes along P_v^u repeat (cyclic computation tree), then $T_e^{(1)}$ will include only the variable nodes in P_v^u up to the first repetition. So, the limiting case is the acyclic computation tree. Using the union bound we get the equation shown at the bottom of the page, where $c = d_v(d_c - 1)$.

Now, if the depth of the computation tree after the first iteration is bounded by $2r$ for *all* edges, then it can increase by at most $2r$ at each iteration, and after l iteration it is bounded by $2rl$. Thus

$$\begin{aligned}\Pr\left(\max_e\left\{D_e^{(l)}\right\} \ge 2rl\right) &\le \Pr\left(\max_e\left\{D_e^{(1)}\right\} \ge 2r\right)\\ &\le nc^r\frac{1}{r!}.\end{aligned}$$

Finally, choosing $r = \frac{\ln(n)}{\frac{1}{2}\ln(\ln(n))}$, we get that for sufficiently large n

$$\begin{aligned}&\Pr\left(\max_e\left\{D_e^{(l)}\right\} \ge \frac{4l\ln(n)}{\ln(\ln(n))}\right)\\ &\quad\le e^{-\ln(n)\left(1-\frac{\ln(\frac{1}{2}\ln(\ln(n)))}{\frac{1}{2}\ln(\ln(n))}-\frac{\ln(C\cdot e)}{\frac{1}{2}\ln(\ln(n))}\right)}\\ &\quad\le \frac{1}{n^{1-o(1)}}.\end{aligned}$$

□

$$\begin{aligned}\Pr\left(\max_e\left\{D_e^{(1)}\right\} \le 2r\right) &\le \sum_{v\in V}\ \sum_{u\in V:d(v,u)=2r} \Pr(\text{variables along } P_v^u \text{ are updated serially})\\ &\le nc^r\frac{1}{r!}.\end{aligned}$$

Next, we use a slightly modified version of a lemma proved in [20] in order to bound the probability that the computation tree of an edge is cyclic.

Lemma IV-C.3: Let e be an edge in a randomly chosen element of $\mathcal{C}_n^{(d_v,d_c)}$ and let $T_e^{(l)}$ be the computation tree of e at the lth iteration with depth $D_e^{(l)}$, then

$$\Pr\left(T_e^{(l)} \text{ is cyclic} \,|\, \text{the depth of } T_e^{(l)} \text{ is } D_e^{(l)}\right) \le \frac{M_{D_e^{(l)}}^2 + \frac{d_c}{d_v} C_{D_e^{(l)}}^2}{n} \quad (31)$$

where

$$M_{D_e^{(l)}} = \sum_{i=0}^{D_e^{(l)}/2} (d_v-1)^i (d_c-1)^i$$

and

$$C_{D_e^{(l)}} = 1 + (d_v-1) \sum_{i=0}^{D_e^{(l-1)}/2-1} (d_v-1)^i (d_c-1)^i.$$

Combining Lemmas IV-C.2 and IV-C.3 we obtain the following result.

Lemma IV-C.4: Let e be an edge in a randomly chosen element of $\mathcal{C}_n^{(d_v,d_c)}$ decoded using a randomly chosen serial schedule and let $T_e^{(l)}$ be the computation tree of e at the lth iteration with depth $D_e^{(l)}$. Then, for a sufficiently large n

$$\Pr\left(T_e^{(l)} \text{ is cyclic or } D_e^{(l)} \ge \frac{4l\ln(n)}{\ln(\ln(n))}\right) \le \frac{o(n)}{n} \to 0. \quad (32)$$

Proof:

$$\begin{aligned}\Pr&\left(T_e^{(l)} \text{ is cyclic or } D_e^{(l)} \ge \frac{4l\ln(n)}{\ln(\ln(n))}\right)\\ &\le \Pr\left(D_e^{(l)} \ge \frac{4l\ln(n)}{\ln(\ln(n))}\right)\\ &\quad + \Pr\left(T_e^{(l)} \text{ is cyclic } | D_e^{(l)} \le \frac{4l\ln(n)}{\ln(\ln(n))}\right) \quad \square\end{aligned}$$

We are now ready to prove the concentration theorem for the serial schedule.

Theorem IV-C.5: (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 be the number of incorrect messages among all $d_v n$ variable-to-check messages sent in the lth decoding step. Furthermore, let p be the expected number of incorrect messages passed along an arbitrary edge with an acyclic computation tree at the lth decoding step. Then for any $\epsilon > 0$ and sufficiently large n

$$\Pr\left(|Z/(nd_v) - p| > \epsilon\right) \le \frac{2}{n^{1-o(1)}}. \quad (33)$$

Proof: As in the flood concentration theorem, we can write

$$\begin{aligned}\Pr\left(|Z - nd_v p| > nd_v\epsilon\right) &\le \Pr\left(|Z - \mathbf{E}[Z]| > nd_v\epsilon/2\right)\\ &\quad + \Pr\left(|\mathbf{E}[Z] - nd_v p| > nd_v\epsilon/2\right) \quad (34)\end{aligned}$$

where $\mathbf{E}[Z]$ is the expected value of Z over all graphs, all serial schedules and all decoder inputs. We bound the terms on the right-hand side of (34), starting with the right term. Let $\mathbf{E}[Z_i], i \in \{1,\ldots,nd_n\}$, be the expected number of incorrect messages passed along edge e_i. From symmetry and expectation linearity we get $\mathbf{E}[Z] = \sum_{i=1}^{nd_v} \mathbf{E}[Z_i] = nd_v\mathbf{E}[Z_1]$ and

$$\begin{aligned}\mathbf{E}[Z_1] = \mathbf{E}&\left[Z_1 | T_{e_1}^{(l)} \text{ is not cyclic}\right] \Pr\left(T_{e_1}^{(l)} \text{ is not cyclic}\right)\\ &+ \mathbf{E}\left[Z_1 | T_{e_1}^{(l)} \text{ is cyclic}\right] \Pr\left(T_{e_1}^{(l)} \text{ is cyclic}\right).\end{aligned}$$

Now, we can lower- and upper-bound $\mathbf{E}[Z_1]$

$$\begin{aligned}&\mathbf{E}\left[Z_1 | T_{e_1}^{(l)} \text{ is not cyclic}\right] \Pr\left(T_{e_1}^{(l)} \text{ is not cyclic and } D_{e_1}^{(l)} < c\right)\\ &\qquad \le \mathbf{E}[Z_1] \le\\ &\mathbf{E}\left[Z_1 | T_{e_1}^{(l)} \text{ is not cyclic}\right] + \Pr\left(T_{e_1}^{(l)} \text{ is cyclic or } D_{e_1}^{(l)} \ge c\right)\end{aligned}$$

where $c = \frac{4l\ln(n)}{\ln(\ln(n))}$.

Further, note that $\mathbf{E}[Z_1 | T_{e_1}^{(l)} \text{ is not cyclic}] = p$ by definition, thus using Lemmas IV-C.2 and IV-C.4 we get

$$nd_v p\left(1 - \frac{o(n)}{n}\right) \le \mathbf{E}[Z] \le nd_v\left(p + \frac{o(n)}{n}\right).$$

Hence, $|\mathbf{E}[Z] - nd_v p| \le d_v o(n)$. It follows that if $\frac{2o(n)}{\epsilon} < n$, which is the case for large enough n, then $\Pr\left(|\mathbf{E}[Z] - nd_v p| > nd_v\epsilon/2\right) = 0$.

Next, we bound the left term on the right-hand side of (34)

$$\begin{aligned}\Pr&(|Z - \mathbf{E}[Z]| > nd_v\epsilon/2)\\ &= \Pr\left(\max_e\left\{D_e^{(l)}\right\} \ge \frac{4l\ln(n)}{\ln(\ln(n))}\right) \cdot P\\ &\quad + \Pr\left(\max_e\left\{D_e^{(l)}\right\} \le \frac{4l\ln(n)}{\ln(\ln(n))}\right) \cdot P\\ &\le \Pr\left(\max_e\left\{D_e^{(l)}\right\} \ge \frac{4l\ln(n)}{\ln(\ln(n))}\right) + P\end{aligned}$$

where

$$P = \Pr\left(|Z - \mathbf{E}[Z]| > nd_v\epsilon/2 \mid \max_e\left\{D_e^{(l)}\right\} \le \frac{4l\ln(n)}{\ln(\ln(n))}\right).$$

P can be tightly bounded using Azuma's inequality [20]. This is done by constructing the following Doob's martingale: $Z_0, Z_1, \ldots, Z_{(d_v+2)n}$, where

$$Z_0 = \left(\mathbf{E}[Z] \mid \max_e\left\{D_e^{(l)}\right\} \le \frac{4l\ln(n)}{\ln(\ln(n))}\right)$$

and

$$Z_{(d_v+2)n} = \left(Z \mid \max_e\left\{D_e^{(l)}\right\} \le \frac{4l\ln(n)}{\ln(\ln(n))}\right)$$

such that in the first nd_v steps we expose the $d_v n$ edges of the graph, in the following n steps we expose the order by which the variable nodes are updated, and in the last n steps we expose the n decoder inputs.

Since by condition the depth of the computation tree is bounded by $D_e^{(l)} \le \frac{4l\ln(n)}{\ln(\ln(n))}$ it can contain at most $2(d_v d_c)^{\frac{2l\ln(n)}{\ln(\ln(n))}}$ distinct edges. Hence, the difference between consecutive random variables in the martingale is bounded by

$$|Z_i - Z_{i-1}| \le 8(d_v d_c)^{\frac{2l\ln(n)}{\ln(\ln(n))}}, \qquad i = 1, \ldots, nd_v.$$

Next, we expose the order by which the variable nodes are updated, such that in each step we reveal the next variable node that is updated. By revealing the serial schedule in that way, an exposed variable node can affect at most the $2(d_v d_c)^{\frac{2l\ln(n)}{\ln(\ln(n))}}$ computation trees it appears in. It does not provide any information on the update order of the remaining variable nodes or on the computation trees that do not contain it. Thus

$$|Z_i - Z_{i-1}| \le 2(d_v d_c)^{\frac{2l\ln(n)}{\ln(\ln(n))}}, \quad i = nd_v + 1, \ldots, n(d_v+1).$$

Finally, we consider the decoder inputs one by one. Again, each exposed decoder input can affect only the computation trees that contain its corresponding variable node, i.e., at most $2(d_v d_c)^{\frac{2l\ln(n)}{\ln(\ln(n))}}$ computation trees

$$|Z_i - Z_{i-1}| \le 2(d_v d_c)^{\frac{2l\ln(n)}{\ln(\ln(n))}}, \qquad i = n(d_v+1)+1, \ldots, n(d_v+2).$$

Since the martingale differences are bounded, Azuma's inequality can be applied to produce

$$\Pr\left(|Z - \mathbf{E}[Z]| > nd_v\epsilon/2 \mid \max_e \left\{D_e^{(l)}\right\} \le \frac{4l\ln(n)}{\ln(\ln(n))}\right) \le 2e^{-\beta\epsilon^2 n^{1-o(1)}}$$

for some positive constant β and sufficiently large n. Finally, we have

$$\Pr(|Z - \mathbf{E}[Z]| > nd_v\epsilon/2) \le \frac{1}{n^{1-o(1)}} + 2e^{-\beta\epsilon^2 n^{1-o(1)}} \le \frac{2}{n^{1-o(1)}}$$

proving the theorem. □

V. Conclusion

We analyzed the convergence rate of the semi-serial and serial schedules. We showed that the computation tree under serial scheduling grows asymptotically in the number of iterations, twice as fast compared to the flooding schedule. We derived DE algorithm for asymptotic analysis of the serial schedule's convergence rate and proved an accompanying concentration theorem. We proved that for BEC under some likely assumptions, asymptotically in the code's length, and when working near the decoder's capacity, the serial decoder is expected to converge in half the number of iterations compared to the flooding decoder.

References

[1] N. Alon, J. Spencer, and P. Erdös, *The Probabilistic Method*. New York: Wiley, 1992.

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

[3] L. Carlitz, "Eulerian numbers and polynomials," *Math. Mag.*, vol. 32, pp. 247–260, 1959.

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

[5] S.-Y. Chung, G. D. Forney, Jr, T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun.Lett*, vol. 5, no. 2, pp. 58–60, Feb. 2001.

[6] L. Comtet, "Permutations by number of rises; Eulerian numbers," in *Advanced Combinatorics: The Art of Finite and Infinite Expansions*. Dordrecht, The Netherlands: Kluwer, 1974, pp. 51 and 240–246.

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

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

[9] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. IEEE Workshop Signal Processing and Systems (SIPS.04)*, Austin, TX, Oct. 2004, pp. 107–112.

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

[11] F. R. Kschischang and B. J. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE J. Select. Areas Commun.*, vol. 16, no. 2, pp. 219–230, Feb. 1998.

[12] F. R. Kschischang, B. J. Frey, and H. Loeliger, "Factor graphs and the sum–product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[13] D. Levin, E. Sharon, and S. Litsyn, "Lazy scheduling for LDPC decoding," *IEEE Commun. Lett.*, vol. 11, no. 1, pp. 70–72, Jan. 2007.

[14] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low density parity-check codes using irregular graphs and belief propagation," in *Proc. IEEE Int. Symp. Information Theory)*, Cambridge, MA, Auf. 1998, p. 117.

[15] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. S. Stemann, "Practical loss-resilient codes," in *Proc. 29th Annu. ACM Symp. Theory of Computing*, El Pao, TX, May 1997, pp. 150–159.

[16] Y. Mao and A. H. Banihashemi, "Decoding low-density parity-check codes with probabilistic scheduling," *IEEE Commun. Lett.*, vol. 5, no. 10, pp. 414–416, Oct. 2001.

[17] Y. Mao and A. H. Banihashemi, "A new schedule for decoding low-density parity-check codes," in *Proc. IEEE GlobeCom*, San Antonio, TX, Nov. 2001, vol. 2, pp. 1007–1010.

[18] M. M. Mansour and N. R. Shanbhag, "Highthroughput LDPC decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 6, pp. 976–995, Dec. 2003.

[19] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.

[20] T. J. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[21] T. J. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity approaching low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[22] E. Sharon, S. Litsyn, and J. Goldberger, "An efficient message-passing schedule for LDPC decoding," in *Proc. 23rd IEEE Conv.*, Tel-Aviv, Israel, Sep. 2004, pp. 223–226.

[23] R. M. Tanner, "A recursive approach to low complextity code," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 533–547, Sep. 1981.

[24] J. Zhang and M. Fossorier, "Shuffled belief propagation decoding," in *Proc 36th Asilomar Conf. Signals, Systems and Computers*, Pacific Grove, CA, Nov. 2002, vol. 1, pp. 8–15.