Serial Schedules for Belief-Propagation: Analysis of Convergence Time

Jacob Goldberger Haggai Kfir

***Abstract*—Low-Density Parity-Check (LDPC) codes are usually decoded by running an iterative belief-propagation algorithm over the factor graph of the code. In the traditional message-passing schedule, in each iteration all the variable nodes, and subsequently all the factor nodes, pass new messages to their neighbors. Recently several studies show that serial scheduling, in which messages are generated using the latest available information, significantly improves the convergence speed in terms of number of iterations. It was observed experimentally in several studies that the serial schedule converges in exactly half the number of iterations compared to the standard parallel schedule. In this correspondence we provide a theoretical motivation for this observation by proving it for single-path graphs.**

***Index Terms*—LDPC codes, iterative decoding, message-passing scheduling.**

I. Introduction

The belief propagation (BP) algorithm [9] also known as the sum-product algorithm is a popular method for computing approximate marginal probabilities in graphical models. The BP algorithm is an exact inference algorithm for singly connected graphs. Although inference in graphs with loops is known to be NP-hard, the loopy belief propagation, can often lead to good approximations of the marginals. Excellent approximation performance have been reported for BP decoding of LDPC codes [2]. The BP algorithm is based on passing messages between the nodes of the graphical model. However, the belief-propagation paradigm does not imply a specific order of passing the messages. The standard message-passing schedule for LDPC decoding is the parallel (flooding) schedule [5], in which in each iteration all the variable nodes, and subsequently all the factor nodes, pass new messages to their neighbors. Even though the flooding schedule is popular, it is known to be non-optimal.

Several recent studies [13], [3], [6], [10] reported empirical results that convergence of serial scheduling is about twice as fast as parallel schduling, without loss of performance. In all previous works, this gain was explained by the fact that serial schedules utilize the most updated information at every calculation. However, no analytic support has been provided to this effect, and especially to the question where does the factor $1/2$ originate from, and whether it can be further improved. Recently the Density-Evolution technique was used to prove this fact for certain LDPC codes over the BEC channel [11].

In this study we prove that random serial schedules converge twice as fast as flooding schedule for all single path graphs.

J. Goldberger is with the Bar-Ilan University, Ramat-Gan 52900, Israel. E-mail: goldbej@eng.biu.ac.il

H. Kfir is with the Israel Aerospace Industries. E-mail: hkfir@iai.co.il

The paper proceeds as follows. In Section II we summarize the possible message-passing schedules. A concentration theorem for Eulerian numbers is given in Section III, and its application to the analysis of serial schedules is presented in Section IV.

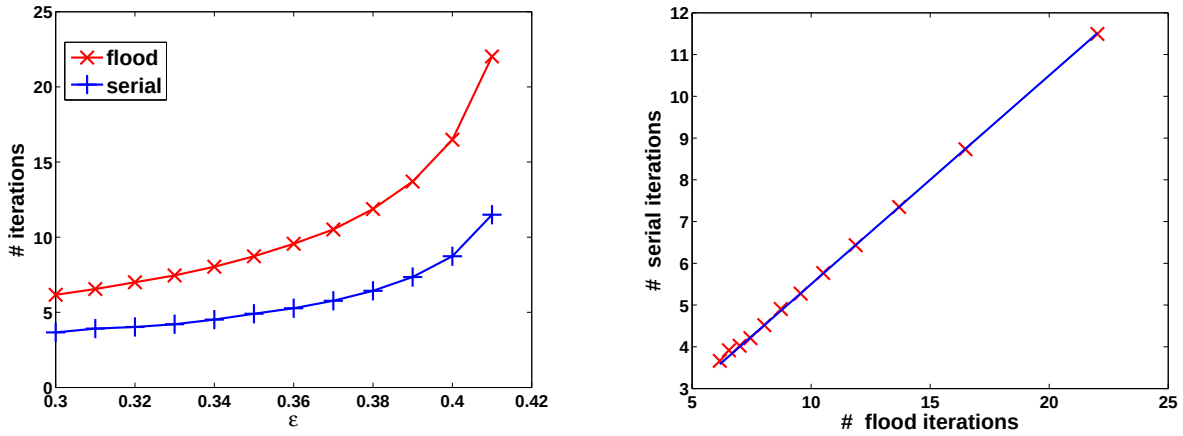
II. Serial Scheduling

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 [9],[2] with Log-Likelihood Ratio (LLR) messages. We denote the LLR 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 LLR message sent from c to v by R_{cv}. The BP algorithm utilizes the following message computation rules:

$$\begin{aligned} Q_{vc} &\leftarrow P_v + \sum_{c' \in N(v) \setminus c} R_{c'v} \\ R_{cv} &\leftarrow 2\text{atanh}\left(\prod_{v' \in N(c) \setminus v} \tanh(\frac{Q_{v'c}}{2}) \right) \end{aligned}$$

where $N(c)$ denotes the variables that are neighbors of c.

The order of passing the messages between the nodes is referred to as an updating rule or a schedule. 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 factor graph. In case of cycle-free graphs we can even construct an optimal schedule [5] 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 factor 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 [2], is the flooding schedule, in which



(a) (b)

Fig. 1. (3,6)-LDPC code over a BEC(ϵ) channel. (a) Number of decoding iterations for the two schedules as a function of the channel parameter. (b) The number of serial-iterations versus the number of flood-iterations and a plot of the function $f(x) = (x+1)/2$.

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* [13], [3] and *serial-C* [10] 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 the following messages are sent:

1) Send all R_{cv} messages into the node v.
2) Send all Q_{vc} messages from the node v.

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 the following messages are sent:

1) Send all Q_{vc} messages into the node c.
2) Send all R_{cv} messages from node c.

Fig. 1 shows simulation results of flooding and serial-C schedules for (3,6) regular LDPC code of length 5000 over a BEC channel. Each point is based on averaging of 1000 decoding sessions. Fig. 1(a) shows the number of iterations for the two schedules as a function of the erasure channel parameter. Fig. 1(b) shows the same information in a different way. It plots the number of serial iterations as a function of the number flood-iterations for same values of ϵ shown in Fig. 1(a). Fig. 1(b) also shows the plot of the function $f(x) = (x+1)/2$ and as it can be seen, the points are almost on the line. This linear relation between the convergence times of serial and flood schedules, is theoretically motivated in the next two sections.

III. Concentration of Eulerian Numbers

In this section we develop the probabilistic tools we need to analyze serial schedules. We prove a concentration theorem on the number of runs in a random permutation. A set of ascending values in a permutation is called a *run*. The identity permutation consists of a single run whereas the reverse permutation on n elements consists of n runs. We use the notation $(\sigma_1, \sigma_2, ..., \sigma_n)$ for a permutation $\sigma \in S_n$, where S_n is the set of all the $n!$ permutations. Using this notation, the permutation $(314526) \in S_6$ contains the following 3 runs: 3, 145 and 26. The Eulerian number $A(n,k)$ is the number of permutations of length n with exactly k runs (e.g. $A(n,1) = A(n,n) = 1$). A thorough introduction on Eulerian numbers can be found in [4].

Define the random variable $f(\sigma)$ as the number of runs in a uniformly selected permutation $\sigma \in S_n$. The expected number of runs arises from a symmetry condition: if a permutation $(\sigma_1, \sigma_2, ..., \sigma_n)$ has k runs, the reflected permutation, $(\sigma_n, ..., \sigma_2, \sigma_1)$ will have $n+1-k$ runs, hence $A(n,k) = A(n, n+1-k)$. Therefore the expected number of runs is: $E(f) = \frac{n+1}{2}$.

Next we utilize the Azuma inequality [7] to show that $f(\sigma)$ is concentrated around its expectation. Define the following sequence of random variables:

$$Z_i(\sigma) = E(f|\sigma_1, ..., \sigma_i) \qquad i = 0, ..., n \qquad (1)$$

Z_i is the conditional expectation of f, knowing how the permutation operates only on the first i elements, i.e. $Z_i(\sigma)$ is a random variable obtained by uniformly sampling a permutation σ and averaging the number of runs over all the permutations that coincide with σ on the first i elements. It is easily verified from Equation (1) that $E(Z_{i+1}|Z_i) = Z_i$, i.e. the sequence $Z_0(= E(f)), Z_1, ..., Z_{n-1}, Z_n(= f)$ is a martingale. Actually, the sequence forms a permutation-element exposing Doob-martingale [1]. To apply the Azuma inequality we need to show that the martingale differences are bounded.

Lemma 1: For every $\sigma \in S_n$ and for each $i = 0, ..., n-1$, the martingale elements satisfy: $|Z_{i+1}(\sigma) - Z_i(\sigma)| \leq 1$.

Proof: Fix i and σ. Denote by $A(x)$ the set of all the permutations π that coincide with σ on the first i elements and $\pi_{i+1} = x$. To prove the lemma, we first define for every $x, y \in \{\sigma_{i+1}, ..., \sigma_n\}$ a 1-1 correspondence between $A(x)$ and $A(y)$ such that the number of runs in each pair differs at most by 1. Assume, without loss of generality, that $x < y$. Given a permutation $\pi \in A(x)$, we define $\pi' \in A(y)$ as follows:

$$\pi'_j = \begin{cases} \pi_j & j \leq i \\ y & j = i+1 \\ \pi_j & j > i+1 \text{ and } (\pi_j < x \text{ or } y < \pi_j) \\ \max_{\{k>i | \pi_k < \pi_j\}} \pi_k & j > i+1 \text{ and } x < \pi_j \leq y \end{cases}$$

For example, let $\sigma = (625431), i = 2, x = 1$ and $y = 5$. Then $\pi = (621453) \in A(1)$ corresponds to $\pi' = (625341) \in A(5)$. It can be easily verified that $\pi' \in A(y)$ and that for each $j > i+1$ it satisfies:

$$\pi'_j < \pi'_{j+1} \quad \text{if and only if} \quad \pi_j < \pi_{j+1}$$

i.e. the run profiles in the last $n-i-1$ elements of π and π' are identical. Hence, a difference in the run profile between π and π' can only occur just before or just after the $i+1$ element of the permutations which implies that $|f(\pi) - f(\pi')| \leq 2$. If $|f(\pi) - f(\pi')| = 2$, then either

$$\pi_i < x < \pi_{i+2} \quad \text{and} \quad \pi_i > y > \pi'_{i+2} \tag{2}$$

or

$$\pi_i > x > \pi_{i+2} \quad \text{and} \quad \pi_i < y < \pi'_{i+2} \tag{3}$$

Since $x < y$ then the construction of π' implies that $\pi'_{i+2} \leq \pi_{i+2}$. Hence, neither of the situations (2) nor (3) can occur. Therefore $|f(\pi) - f(\pi')| \leq 1$.

Denote $B(x) = E(f(\pi) \,|\, \pi \in A(x))$. The correspondence between $A(x)$ and $A(y)$ yields that

$$|B(x) - B(y)| = \frac{1}{(n-i)!} \Big| \sum_{\pi \in A(x)} (f(\pi) - f(\pi')) \Big| \leq 1 \tag{4}$$

Observing that $B(\sigma_{i+1}) = Z_{i+1}(\sigma)$, we obtain:

$$|Z_{i+1}(\sigma) - Z_i(\sigma)| = \Big| \frac{1}{n-i} \sum_x B(x) - B(\sigma_{i+1}) \Big| \tag{5}$$

$$\leq \frac{1}{n-i} \sum_x |B(x) - B(\sigma_{i+1})| \leq 1$$

where the summation is performed over $x \in \{\sigma_{i+1}, ..., \sigma_n\}$. □

We note in passing that the bound 1 on the martingale differences that appears in Lemma 1 can be replaced with one-half and this is a tight bound.

Utilizing Lemma 1, the concentration theorem for the number of runs in a random permutation, follows directly from the Azuma inequality which extends the Chernoff bound to the case of martingales with bounded differences.

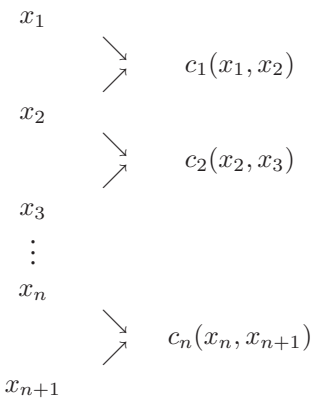
Azuma Inequality [7]: Let $Z_0, ..., Z_n$ be a martingale such that for each i: $|Z_{i+1} - Z_i| \leq 1$, then $p(|Z_n - Z_0| > \epsilon) \leq 2\exp(-\frac{\epsilon^2}{2n})$.

Theorem 1: Let $f(\sigma)$ be the number of runs in a uniformly selected permutation on n elements, then $E(f) = \frac{n+1}{2}$ and for all $\epsilon > 0$:

$$p\left(\,|f - E(f)| > \epsilon\,\right) \leq 2\exp(-\frac{\epsilon^2}{2n}) \tag{6}$$

IV. BP Applied to a Chain Graph

In this section we analyze the simple case of belief-propagation applied to the following single chain that consists of n factors.



(7)

The factors $c_i(x_i, x_{i+1})$ are defined on pairs of consecutive variables. The probabilistic model described by the factor graph (7) is precisely the hidden Markov model (HMM) when the latent variables are discrete and a linear dynamical system when latent and observed random variables are jointly Gaussian. In the case of LDPC codes the factors are parity-check nodes, namely $c_i(x_i, x_{i+1}) = 1 - (x_i \oplus x_{i+1})$.

For this cycle free graph, the BP will converge to the exact marginal (posterior) probabilities, after a message from variable x_1 has arrived at node x_{n+1} and a message from variable x_{n+1} has arrived at node x_1. The optimal schedule for this case will be the forward-backward algorithm which can be implemented in a single iteration by sending the messages forward : $x_1 \rightarrow c_1(x_1, x_2) \rightarrow x_2, ..., \rightarrow x_{n+1}$ and backward $x_{n+1} \rightarrow c_n(x_n, x_{n+1}) \rightarrow x_n, ..., \rightarrow x_1$.

Note that in the optimal algorithm the messages in the two directions can be interlaced in any order. The worst updating rule will be flooding which requires exactly n iterations.

Consider the case of propagating a message from x_1 to x_{n+1} according to a randomly selected factor-based (Serial-C) schedule. A random schedule is described by a permutation $\sigma \in S_n$ that dictates the order in which the checks are updated. It can be verified that the number of iterations needed to propagate a message from x_1 to x_{n+1} according to the check-based schedule σ is exactly the number of runs in σ. For example, the permutation $\sigma = (314526)$ defines a schedule where the check-node c_2 is updated first, check-node c_5 is updated second and the check-node c_6 is updated last. Since σ has 3 runs, we need 3 iterations to propagate a message from x_1 to x_7. On the first iteration the message from x_1 is propagated to x_2. Since according to σ, c_2 is updated before

c_1, the message from x_1 is not further propagated on the first iteration. On the second iteration the message is propagating from x_2 to x_5, and on the third and last iteration the message reaches x_7. The connection between the number of runs in a permutation and the number of iterations of the schedule induced from the permutation yields a concentration theorem for the serial schedule. The following theorem states that in the case of HMM, the number of iterations required until the BP algorithm converges, when we use a random serial schedule, is concentrated around half of the number of iterations needed by the the flooding schedule.

Theorem 2: Let $c_1, ..., c_n$ be the pairwise factors of the HMM $x_1, ..., x_{n+1}$. Let σ be a randomly selected permutation on the n factors. Let $g(\sigma)$ be the number of iterations needed until convergence of the BP algorithm applied to HMM, according to the schedule σ. The following concentration property holds for the random variable g:

$$p(|\, g(\sigma) - \frac{n+1}{2} \,| > \epsilon) \leq 2\exp(-\frac{\epsilon^2}{2n}) \tag{8}$$

Proof: Fix a permutation σ on the n factors $c_1, ..., c_n$. The BP algorithm converges when a message from x_1 arrives to x_{n+1} and a message from x_{n+1} arrives to x_1. Hence the number of iterations until convergence is $g(\sigma) = \max(g_{forward}(\sigma), g_{backward}(\sigma))$, where $g_{forward}$ is number of iterations needed for a message from x_1 to approach x_{n+1} according to the schedule defined by σ and $g_{backward}$ is defined in a similar way. By definition $g_{forward}(\sigma)$ is exactly the number of runs in σ which was denoted in the previous section as $f(\sigma)$ and $g_{backward}$ is $n+1-f(\sigma)$. Note that

$$|f(\sigma) - \frac{n+1}{2}| > \epsilon \quad \text{iff} \quad |n+1-f(\sigma) - \frac{n+1}{2}| > \epsilon$$

Hence

$$p(|\, g(\sigma) - \frac{n+1}{2} \,| > \epsilon) = p(|f(\sigma) - \frac{n+1}{2}| > \epsilon) \tag{9}$$

Combining expressions (9) and (6) we obtain the the concentration result for a serial-schedule BP applied to a single path graph.

V. Conclusions

The theorems and observations in this paper provide an explanation to the opening question presented: why serial schedules converge twice as fast as the parallel schedules for the loopy BP algorithm. We proved that for single path graphs random serial schedules converge on average in half the number of iterations that parallel schedules do. We proved that the distribution is highly concentrated about this mean, and showed that in simulations these results are also applicable to graphs with loops. We introduced the martingale concentration theory to the area of message-passing scheduling. A theoretical generalization of the proposed serial-schedule analysis to the general case of loopy graphs is left for future research. We strongly believe, however, that this paper is a step in this direction. Serial schedules require less memory than parallel ones [3], and they can be implemented inplace. In this sense, the relation between parallel and serial schedules highly resembles the relation that exists between the Jacobi and the Gauss-Seidel iterative methods for solving linear equations. The serial method can be applied in a semi-parallel fashion that allows for several variables to be updated in the same time step [13], [10]. Therefore, it seems that serial schedules should be a considerable choice for BP on loopy graph applications, such as LDPC.

References

[1] N. Alon, J. Spencer and P. Erdös. The probabilistic method. John Wiley and sons, 2000. 2

[2] R. G. Gallager, Low-density parity-check codes, *IRE Trans. Info. Theory*, vol. IT-8, pp. 21–28, 1962. 1

[3] H. Kfir and I. Kanter. Parallel vs. sequential updating for belief propagation decoding. Physica A 330, pp. 259-270, 2003. 1, 2, 4

[4] D. E. Knuth. The art of computer programming. vol 3, chap 5.1.3, Addison-Wesley publishing company, 1973. 2

[5] F. R. Kschischang, B. J. Frey and H. Loeliger. Factor Graphs and the sum-product algorithm. *IEEE Trans. on Info. Theory*, 2001. 1

[6] D.E. Hocevar, A reduced complexity decoder architecture via layered decoding of LDPC codes, *IEEE Workshop on Signal Proc. Sys.*, 2004. 1

[7] W. Hoeffding. Probability inequalities for sums of bounded variables. J. Amer. Statist. Ass. 58, pp. 13-30, 1963. 2, 3

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

[9] J. Pearl. Probabilistic reasoning in intelligent systems. San Mateo CA: Morgan Kaufman, 1988. 1

[10] E. Sharon, S. Litsyn and J. Goldberger. An efficient message-passing schedule for LDPC decoding. The 23rd IEEE convention of Electrical Engineering, Israel, 2004. 1, 2, 4

[11] E. Sharon, S. Litsyn and J. Goldberger. Efficient serial message-passing schedule for LDPC decoding. *IEEE Trans. on Info. Theory*, pp. 4076–4091, 2007. 1

[12] Y. Xiao and A. H. Banihashemi, Graph-based message-passing schedules for decoding LDPC codees, *IEEE Trans. Commm.*, vol. 52 (12) , pp. 1208-2105, 2004.

[13] J. Zhang and M.P.C. Fossorier. Shuffled iterative decoding. *IEEE Transactions on Communications*. Volume 53, Issue 2, pp. 209– 213, 2005. 1, 2, 4

Jacob Goldberger has received the M.Sc. degree in mathematics and the Ph.D. degree in electrical engineering in 1989 and 1998, respectively both from Tel-Aviv University, Israel. He held a post doctoral position at the Weizmann institute vision group (2000) and at the University of Toronto machine learning group (2003). In 2004 he joined the School of Engineering, Bar-Ilan University where he is currently a faculty member. His research interests include machine learning, information theory, computer vision and speech recognition.

Haggai Kfir has received the B. Sc. in physics and computer science and Ph. D. in physics in 2000 and 2005, respectively, both from Bar Ilan University, Israel. His thesis explores various relations between statistical mechanics and error correction codes, decoding of correlated sequences and joint source-channel coding. currently he is with the Israel Aerospace Industries (IAI).