RZ 3312 (#93358) **11/20/00**
Engineering & Technology **8 pages**

Research Report

A Reduced-Complexity Decoding Algorithm for Low-Density Parity-Check Codes

Evangelos Eleftheriou, Thomas Mittelholzer and Ajay Dholakia

IBM Research
Zurich Research Laboratory
8803 RüschlikonSwitzerland

A Reduced-Complexity Decoding Algorithm for Low-Density Parity-Check Codes

Evangelos Eleftheriou, Thomas Mittelholzer and Ajay Dholakia

IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland
E-mail: {adh,ele,tmi}@zurich.ibm.com

***Abstract*:** A new reduced-complexity decoding algorithm for low-density parity-check codes that operates entirely in the log-likelihood domain is presented. The computationally expensive check-node updates of the sum-product algorithm are simplified by using a difference-metric approach on a two-state trellis and by employing the dual-max approximation. The dual-max approximation is further improved by using a correction factor that allows the performance to approach that of full sum-product decoding.

Introduction

A binary low-density parity-check (LDPC) code [1, 2] is a linear (N, K) code described by an $M \times N$ parity-check matrix H. Iterative decoding of binary LDPC codes [1] has become a topic of increasing interest after it was recognized that these codes can achieve very good performance on binary-input memoryless channels such as the binary symmetric channel or the additive white Gaussian noise channel [2, 3]. The most commonly used method for decoding LDPC codes is the so-called "sum-product" algorithm (SPA) (cf. [2] and references therein), which is best described by using the bipartite graph associated with the parity-check matrix H . This graph has two kinds of nodes: N symbol nodes, corresponding to each bit in the codeword $\underline{x}$, and M check nodes, corresponding to the parity checks $pc_m(\underline{x})$, $1 \le m \le M$, represented by the rows of the matrix H. Each symbol node is connected to the check nodes it participates in, and each check node is connected to the symbol nodes it checks. The SPA operates by passing messages between symbol nodes and check nodes. The messages themselves can be *a posteriori* probabilities (APP) or log-likelihood ratios (LLR). A convenient and commonly used message-passing schedule alternately computes updates of all symbol node and of all check node messages.

Recently, a simplification of the SPA has been proposed [3] that reduces its high computational cost at the price of some loss in performance. In this report, a novel simplified SPA is presented that operates entirely in the LLR domain and offers a substantial reduction in complexity with essentially the same performance as the full SPA. Such an algorithm can be used for iteratively decoding LDPC encoded data over the AWGN channel but also in applications in which LLRs are used to exchange soft reliability information between the inner and the outer decoders in a serially concatenated system [4].

The Sum-Product Algorithm (SPA)

Following the notation in [2, 3], let $N(m) = \{n : H_{m,n} = 1\}$ be the set of bits that participate in check m, and let $M(n) = \{m : H_{m,n} = 1\}$ be the set of checks in which bit n participates. The exclusion of an element n from $N(m)$ or m from $M(n)$ is denoted by $N(m)\backslash n$ or $M(n)\backslash m$, respectively, and H^T is the transpose of H. Finally, we denote by $\underline{y} = [y_1, ..., y_N]$ the received sequence that corresponds to the transmitted codeword $\underline{x} = [x_1, ..., x_N]$. The inputs to the LDPC decoder consist of LLRs $\ln(P(x_n = 1| y_n)/P(x_n = 0| y_n))$ or, equivalently, of APPs $P(x_n = 1| y_n)$ and $P(x_n = 0| y_n)$, which are determined by the channel statistics. The SPA is an iterative algorithm using this soft information as inputs. In the initial step of the SPA, each symbol node $n$generates messages $q_{m,n}(x)$ for $x = 0, 1$, and passes them to all check nodes m that are connected to symbol node n, i.e.,

Initialization: $q_{m,n}(x) = P(x_n = x \mid y_n)$ for $x = 0, 1$.

The core of the SPA consists of two local update rules for check and symbol nodes [1], [2]. The update of the messages sent from check node m to all symbol nodes $n \in N(m)$ is given by

Step 1 (check-node update): For each m and $n \in N(m)$, and for $x = 0, 1$, compute

$$r_{m,n}(x) = \sum_{\{x_{n'}: n' \in N(m)\setminus n\}} P(pc_m(\underline{x}) = 0 \mid x_n = x, \{x_{n'}: n' \in N(m)\setminus n\}) \prod_{n' \in N(m)\setminus n} q_{m,n'}(x_{n'}),$$

where the conditional probability in the summation is an indicator function that indicates whether the *m*-th check-sum is satisfied given the hypothesized values for x_n and $\{x_{n'}\}$.

The update of the messages sent from symbol node n to all check nodes $m \in M(n)$ is given by

Step 2 (symbol-node update): For each n, and $m \in M(n)$, and for $x = 0, 1$, update

$$q_{m,n}(x) = \lambda_{m,n} P(x_n = x \mid y_n) \prod_{m' \in M(n)\setminus m} r_{m',n}(x),$$

where the constant $\lambda_{m,n}$ is chosen such that $q_{m,n}(0) + q_{m,n}(1) = 1$. The SPA then computes for each n and for $x = 0, 1$, the "pseudoposterior probabilities" $q_n(\cdot)$ as

$$q_n(x) = \lambda_n P(x_n = x \mid y_n) \prod_{m \in M(n)} r_{m,n}(x), \tag{1}$$

where the constant λ_n is chosen such that $q_n(0) + q_n(1) = 1$.

In the final decoding step the SPA performs the following procedure:

Step 3: (a) Quantize $\hat{\underline{x}} = [\hat{x}_1, \ldots, \hat{x}_N]$ such that $\hat{x}_n = 1$ if $q_n(1) > 0.5$, and $\hat{x}_n = 0$ if $q_n(1) \le 0.5$.
(b) If $\hat{\underline{x}} H^T = \underline{0}$, then stop and $\hat{\underline{x}}$ is the decoder output; otherwise go to Step 1.
(c) Declare a failure if the algorithm does not halt within some maximum number of iterations.

In general, each check-sum $pc_m(\underline{x})$ can be viewed as a single-parity check code on the $k = |N(m)|$ symbols it checks. The node messages $r_{m,n}(x)$ of Step 1 can be regarded as extrinsic information for x_n given the statistics $q_{m,n}(\cdot)$. These messages can be computed by the forward-backward algorithm [2] on the two-state trellis of the single-parity check code. In the probability domain, the *i*th branch metrics of the two-state trellis corresponding to check node m is given by $q_{m,i}(x)$. The metrics $\alpha_i(s)$ of states $s = 0, 1$ at time i in the forward recursion of the BCJR are updated by (where $\oplus$ denotes addition modulo 2):

forward recursion: For $i = 1, \ldots, k-1$ and $x = 0,\ 1$

$$\alpha_i(x) = \alpha_{i-1}(0) q_{m,i}(x) + \alpha_{i-1}(1) q_{m,i}(x \oplus 1) \quad ; \tag{2}$$

with initial condition $\alpha_0(0) = 1$, $\alpha_0(1) = 0$. For a fixed time i, the state metric $\alpha_i(\cdot)$ is a probability distribution, viz., $\alpha_i(s)$ is the probability of reaching state s in the forward recursion, when each branch is chosen according to the branch metric (probability) $q_{m,n}(x)$. Similar update rules hold for the state metrics $\beta_i(s)$ of the backward recursion, which again have an interpretation as probabilities. In particular the backward recursion of the two-state BCJR algorithm is updated by

backward recursion: For $i=(k-1),\cdots,1$ and $x=0,\ 1$

$$\beta_i(0)=\beta_{i+1}(0)q_{m,i+1}(x)+\beta_{i+1}(1)q_{m,i+1}(x\oplus 1) \quad ;$$

with the initial condition $\beta_k(0)=1,\ \ \beta_k(1)=0.$ The combining pass of the BCJR algorithm yields the messages that are sent from check node m to symbol node $n \in N(m)$ and is given by

combining recursion: For $i=1,\cdots,k$ and $x=0,\ 1$

$$r_{m,i}(x)=\alpha_{i-1}(0)\beta_i(x)+\alpha_{i-1}(1)\beta_i(x\oplus 1) \quad .$$

Simplified SPA Using Log-likelihood Ratios

It is possible to use LLRs as messages instead of APPs. This allows us to replace multiplications in Step 2 of the SPA with additions. Step 3 can also be easily adapted for LLRs. The simplifications derived in this paper allow one to efficiently use LLRs also in Step 1 without converting between LLRs and APPs.

In the LLR domain, we define

$$\delta A_i \triangleq \ln\tfrac{\alpha_i(1)}{\alpha_i(0)} \quad \text{and} \quad \delta B_i \triangleq \ln\tfrac{\beta_i(1)}{\beta_i(0)}\,.$$

We will also make use of the following LLRs:

$$\lambda_{m,i} \triangleq \ln\tfrac{q_{m,i}(1)}{q_{m,i}(0)} \quad \text{and} \quad \Lambda_{m,i} \triangleq \ln\tfrac{r_{m,i}(1)}{r_{m,i}(0)}\,.$$

Note that the LLRs $\delta A_i=\ln\,\alpha_i(1)-\ln\,\alpha_i(0)$ and $\delta B_i=\ln\beta_i(1)-\ln\beta_i(0)$ can be viewed as the forward and backward difference metrics in the log domain, respectively. The application of a difference-metric approach to the dual-max detector for partial-response class IV channels has been proposed in [5]. In this report, we consider the two-state parity-check trellis and use the difference of state metrics, i.e., the difference of logarithms, which is merely the LLR of the probabilities. Using the difference metric definition δA_i , and the standard approximation

$$\ln\sum_j \exp a_j \approx \max_j a_j\ ,$$

the forward recursion (2) can be rewritten as

$$dA_i = \ln \frac{a_{i-1}(0)q_{m,i}(1)+a_{i-1}(1)q_{m,i}(0)}{a_{i-1}(0)q_{m,i}(0)+a_{i-1}(1)q_{m,i}(1)}$$

$$= \ln\{\exp(k_{m,i}) + \exp(dA_{i-1})\} - \ln\{1 + \exp(k_{m,i} + dA_{i-1})\} \qquad (3)$$

$$\text{l} \ \max\{k_{m,i},\ dA_{i-1}\} - \max\{0,\ k_{m,i} + dA_{i-1}\}$$

$$= \begin{cases} -\mathrm{sgn}(dA_{i-1})k_{m,i} & \text{if } |dA_{i-1}| > |k_{m,i}| \\ -\mathrm{sgn}(k_{m,i})dA_{i-1} & \text{otherwise,} \end{cases} \qquad (4)$$

where sgn($) is the sign function. The backward and the combining recursions can be formulated in a similar way. By replacing dA_i and dB_i with da_i and db_i , respectively, the following LLR version of the forward-backward algorithm is obtained:

initialization: $da_0 =$ o and $db_k =$ o

forward recursion: For $i = 2 ¬ \ k-1$

$$da_i = \begin{cases} -\mathrm{sgn}(da_{i-1})k_{m,i} & \text{if } |da_{i-1}| > |k_{m,i}| \\ -\mathrm{sgn}(k_{m,i})da_{i-1} & \text{otherwise} \end{cases} \qquad (5)$$

backward recursion: For $i = k-1 ¬ \ 1$

$$db_i = \begin{cases} -\mathrm{sgn}(db_{i+1})k_{m,i+1} & \text{if } |db_{i+1}| > |k_{m,i+1}| \\ -\mathrm{sgn}(k_{m,i+1})db_{i+1} & \text{otherwise} \end{cases} \qquad (6)$$

combining recursion For $i = 1 ¬ \ k$

$$L_i = \begin{cases} -\mathrm{sgn}(da_{i-1})db_i & \text{if } |da_{i-1}| > |db_i| \\ -\mathrm{sgn}(db_i)da_{i-1} & \text{otherwise} \end{cases} \qquad (7)$$

Correction Factor for the Dual-Max Approximation

The simplified SPA that results from using Eqs. (5) to (7) for the check node updates will be called the LLR-SPA because it operates entirely in the LLR domain. The LLR-SPA has a somewhat lower performance than the full SPA. Following [6, 7], we can apply a correction factor obtained from the Jacobian logarithm to improve the dual-max approximation in Eqs. (3) to (4) while maintaining the low complexity. The Jacobian logarithm is given by

$$\ln\{\exp(x)+\exp(y)\} \;=\; \max\{x,y\}+\ln\{1+\exp(-|x-y|)\}.$$

One can show that the approximation error, i.e., Eq. (3) minus Eq. (4), is given by the bivariate function

$$f(u,v)=\ln\frac{1+\exp(-|\,u-v|)}{1+\exp(-|\,u+v|)} \quad ,$$

where $u = dA_{i-1}$ and $v = k_{m,i}$. Figure 1 shows a plot of the generic bivariate function $f(u,v)$. In practice, $f(u,v)$ can be approximated by using a single constant term c , i.e.,

$$f(u,v)\; l \; \{ \begin{array}{ll} c & \text{if } |u+v|>2|u-v| \text{ and } |u-v|<2 \\ -c & \text{if } |u-v|>2|u+v| \text{ and } |u+v|<2 \\ 0 & \text{otherwise .} \end{array} \qquad (8)$$

A similar correction factor applies to the approximations in the backward and combining recursions. The constant c can be selected to maximize the performance gains in the region of interest with respect to bit-error rate or signal-to-noise ratio. Figure 2 shows a plot of the simplified bivariate function as given in (8) with $c=0.5$. Finally, Figure 3 shows the performance of the *LLR-SPA with correction factor* $c=0.5$ for the additive white Gaussian noise channel using the same rate-1/2 LDPC code with $N=504$ as in [3]. For comparison, the performance of the full SPA and LLR-SPA is also shown. The number of iterations for the two sets of curves shown is at most 10 and 200, respectively. It can be seen that *LLR-SPA with correction factor* performs within less than 0.05 dB of the full SPA. It can also be seen that the performance difference between the full SPA and the low-complexity derivatives thereof decreases as the signal-to-noise ratio increases.

Acknowledgments: We thank D. J. C. MacKay for providing the LDPC matrix for the code used in this paper.

References

1 R. G. Gallager, "Low-density parity-check codes", *IRE Trans. Info. Theory*, 1962, IT-8, pp. 21-28

2 D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices", *IEEE Trans. Info. Theory*, 1999, 46(2), pp. 399-431

3 M. P. C. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation", *IEEE Trans. Commun.*, 1999, 47(5), pp. 673-680

4 T . Mittelholzer, A. Dholakia, and E. Eleftheriou, "Reduced-complexity decoding of LDPC codes for generalized partial response channels," *IEEE Trans. Magn.*, Jan. 2001 (in press).

5 S. Ölçer and G. Ungerboeck, “Reed-Muller coding for partial response channels”. *Proc. 1993 IEEE Int’l. Symp. on Information Theory*, San Antonio, TX (IEEE, Piscataway, 1992), p. 243

6 A. J. Viterbi, “An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes”, *IEEE J. Sel. Areas Commun.*, 1998, 16(2), pp. 260-264

7 W. J. Gross and P. G. Gulak, “Simplified MAP algorithm suitable for implementation of turbo decoders”, *Electron. Lett.*, 1998, 34(16), pp. 1577-1578

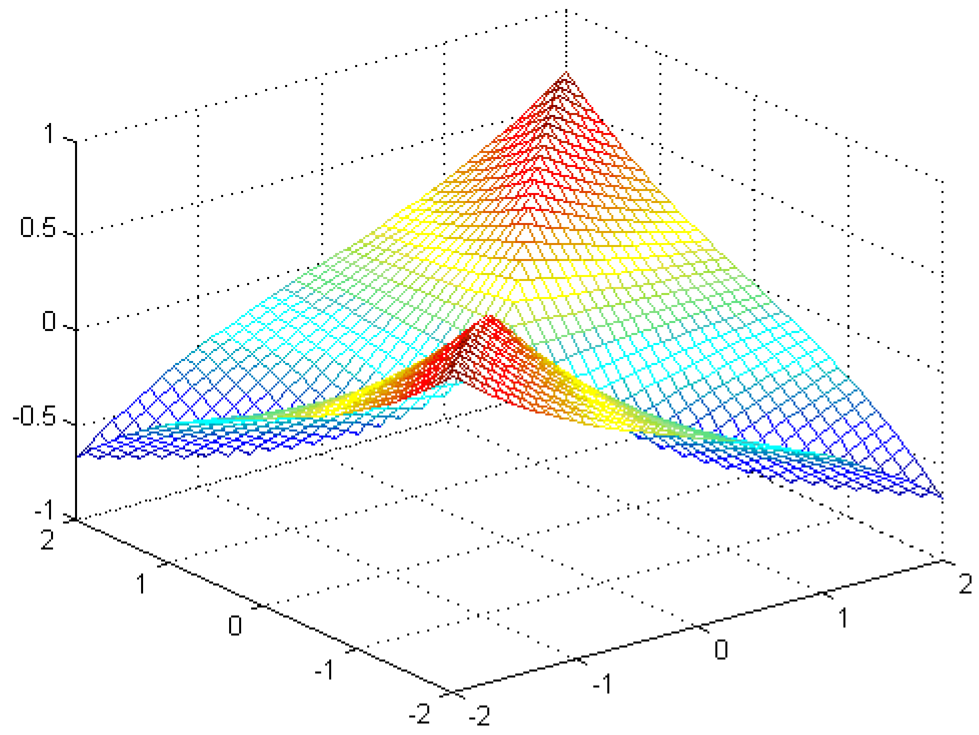

Figure 1: Generic bivariate function $f(u, v)$.

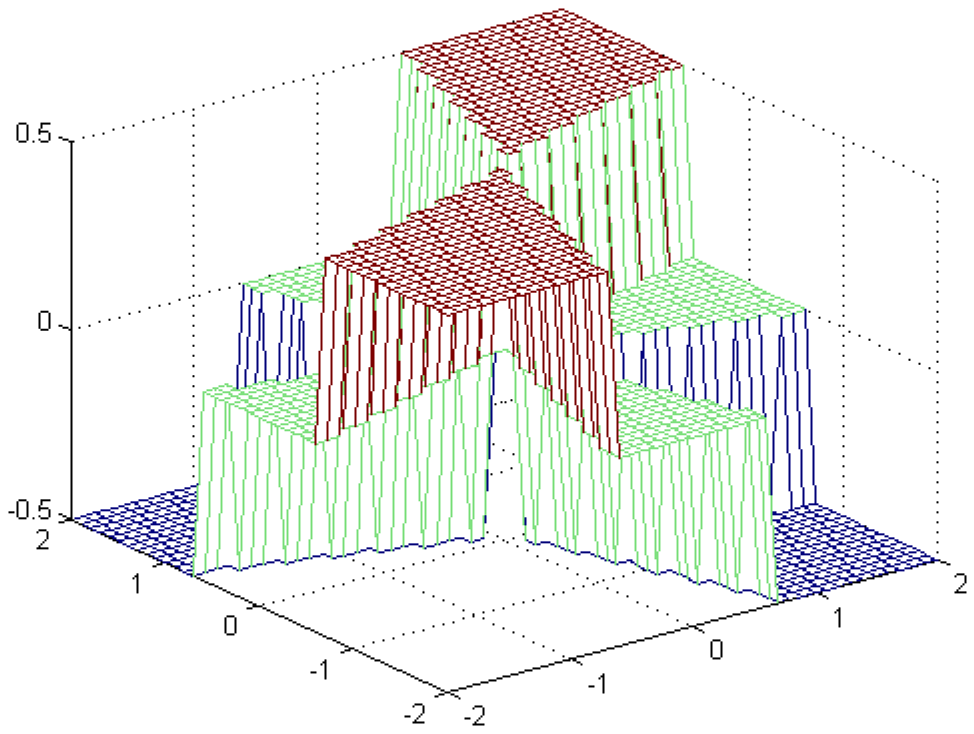

Figure 2: Simplified bivariate function $f(u, v)$.

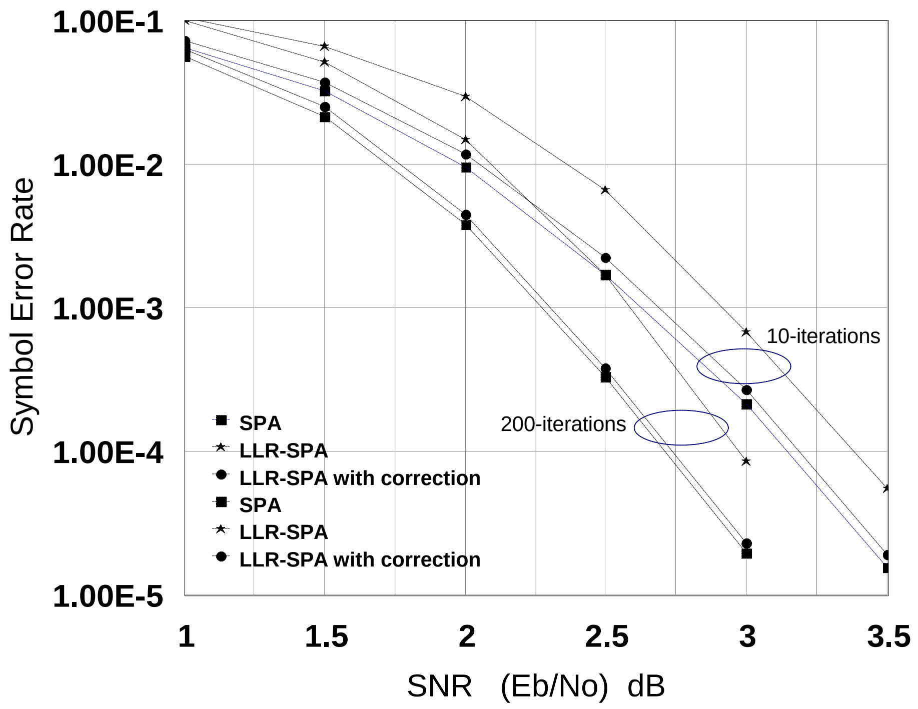

Figure 3. Performance of rate-1/2 LDPC code with N=504.