High-Performance, Low-Complexity Decoding of Generalized Low-Density Parity-Check Codes

Tong Zhang and Keshab K. Parhi
Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN 55455, USA

Abstract—**A class of pseudo-random compound error-correcting codes, called** *Generalized Low-Density (GLD) Parity-Check* **codes, has been proposed recently. As a generalization of Gallager's Low-Density Parity-Check (LDPC) codes, GLD codes are also asymptotically good in the sense of minimum distance criterion and can be effectively decoded based on iterative soft-input soft-output (SISO) decoding of individual constituent codes. The code performance and decoding complexity of GLD code are heavily dependent on the employed SISO decoding algorithm. In this paper, we show that Max-Log-MAP is an attractive SISO decoding algorithm for GLD coding scheme, considering the trade-off between performance and complexity in the practical implementations. A normalized Max-Log-MAP is presented to improve the GLD code performance significantly compared with using conventional Max-Log-MAP. Moreover, we propose two techniques, decoding task scheduling and reduced search Max-Log-MAP, to effectively reduce the decoding complexity without any performance degradation.**

I. Introduction

Low-Density Parity-Check (LDPC) codes, first introduced by Gallager [1], have recently received a lot of attention because of their excellent performance and many new developments have been brought in this area. As a direct generalization of Gallager's LDPC codes, GLD codes were introduced by Lentmaier [2] and Boutros [3], independently. GLD codes are constructed by replacing each single parity check in Gallager's LDPC codes with the parity check matrix of a small linear block code called the *constituent code*. It has been shown that GLD codes are asymptotically good in the sense of minimum distance and exhibit an excellent performance over both AWGN and Rayleigh channels [2][3][4]. Moreover, Pothier [4] demonstrates that GLD codes also can be considered as a generalization of product codes, and because of their higher flexibility on the selection of code length, GLD codes turn out to be a promising alternative to product codes in many applications.

GLD codes can be effectively decoded based on iterative SISO decoding of individual constituent codes, where the code performance and decoding complexity are heavily dependent on the employed SISO decoding algorithm. Exploiting the fact that the constituent code usually has a small code length and high code rate, the trellis-based MAP (maximum a posteriori probability) algorithm [5], also referred as BCJR, or its low-complexity modification, Max-Log-MAP algorithm [6], can be used to obtain high error-correcting performance with reasonable decoding complexity.

This research was supported by the Army Research Office by grant number DA/DAAD19-01-1-0705.

In this paper, by introducing several techniques to improve performance and reduce decoding complexity, we show that Max-Log-MAP is an attractive SISO decoding algorithm for GLD coding scheme. The paper is organized as follows: In section II, short reviews of GLD code and Max-Log-MAP algorithm are given and an earlier stopping criterion for GLD decoding is presented. By introducing a scaling factor, in section III, we present a normalized Max-Log-MAP algorithm to improve the GLD code performance. In section IV, we propose two techniques, *task scheduling* and *reduced search* Max-Log-MAP algorithm, to reduce the decoding complexity without any performance degradation. The conclusions are drawn in section V.

II. Background

A. GLD Codes

In the following, according to [4], we briefly describe the construction of GLD codes and their iterative decoding scheme. As a generalization of LDPC codes, GLD codes are also defined by a sparse parity check matrix H, constructed by replacing each row in LDPC parity check matrix with $(n-k)$ rows including one copy of the parity check matrix H_0 of constituent code $C_0(n,k)$, a k-dimensional linear code of length n. The structure of the GLD parity check matrix H is depicted in Fig. 1. We divide H into J submatrices, $H^1 \cdots H^J$, each containing a single column of constituent parity check matrix H_0 in each column. H^1 is a *block diagonal* matrix and produces the direct sum of N/n constituent codes, where N is the GLD code length. All other submatrices are constructed as: $H^j = \pi_{j-1}(H^1)$ for $j = 2 \cdots J$, where π_{j-1} represents a random column permutation. A (N, J, n) GLD code C can be considered as the intersection of J *super-codes* $C^1, \cdots, C^J$, whose parity check matrices are the J submatrices, $H^1, \cdots, H^J$, respectively.

GLD codes can be effectively decoded using the following iterative decoding scheme: For each bit, we compute its probability given its received sample considering that it belongs to the super-code C^1. We use N/n SISO decoders working in parallel on the N/n independent constituent codes of C^1. This step generates for each coded bit an *a posteriori probability* (APP) and an *extrinsic* probability. The latter one, as an *a priori* information, is fed to the SISO decoders working

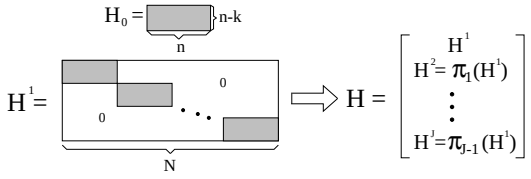

Fig. 1. Structure of GLD parity check matrix H

on the N/n constituent codes of super-code C^2. This process is iterated on each super-code: $C^1 \rightarrow C^2 \rightarrow \cdots \rightarrow C^J \rightarrow C^1 \rightarrow \cdots$, until the preset maximum iteration number is reached.

It has been shown that binary GLD codes with only $J = 2$ levels is asymptotically good. Furthermore, GLD codes with 2 levels have the highest code rate and simple decoder structure. Thus, in this work, we only consider the decoding of $(N, 2, n)$ binary GLD codes.

B. Max-Log-MAP Decoding of Linear Binary Block Codes

Consider a binary (n, k) linear block code C with a minimal bit-level *trellis* T [7]. Each path through the trellis T represents one distinct codeword in C. Let $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ be a codeword in C. Assume $\mathbf{u}$ be modulated by BPSK and transmitted over AWGN channel with noise spectrum density N_0. Denote the received noisy sequence as $\mathbf{r} = (r_1, r_2, \ldots, r_n)$.

Let σ_k denote a state in trellis T at time-k and $B_k(C)$ denote the set of all branches (σ_{k-1}, σ_k) that connect the states at time-$(k-1)$ and time-k in T. Let $B_k^0(C)$ and $B_k^1(C)$ denote the two disjoint subsets of $B_k(C)$ that correspond to the output code bits $u_k = 0$ and $u_k = 1$, respectively. The MAP rule provides us the log-likelihood ratio (LLR) associated with each bit u_k:

$$L_R(u_k) = \underbrace{L_c \cdot r_k + L(u_k)}_{L_{in}(u_k)} + \underbrace{\log \frac{\sum_{(\sigma',\sigma)\in B_k^1(C)} \alpha_{k-1}(\sigma') \cdot \beta_k(\sigma)}{\sum_{(\sigma',\sigma)\in B_k^0(C)} \alpha_{k-1}(\sigma') \cdot \beta_k(\sigma)}}_{L_e(u_k)} \quad (1)$$

where the channel probability $L_c = \frac{4E_c}{N_0}$ (E_c denotes the coded bit energy), $L(u_k)$ is the *a priori* value of $\log \frac{P(u_k=1)}{P(u_k=0)}$ and the sum of $L_c \cdot r_k$ and $L(u_k)$, denoted as $L_{in}(u_k)$, is called *intrinsic* information. The third term $L_e(u_k)$ is called *extrinsic* information in which forward recursion metric $\alpha_k(\sigma)$ and backward recursion metric $\beta_k(\sigma)$ are recursively calculated as

$$\alpha_k(\sigma) = \sum_{(\sigma',\sigma)\in B_k(C)} \alpha_{k-1}(\sigma') \cdot e^{(x_k \cdot L_{in}(u_k))},$$

$$\beta_{k-1}(\sigma') = \sum_{(\sigma',\sigma)\in B_k(C)} \beta_k(\sigma) \cdot e^{(x_k \cdot L_{in}(u_k))},$$

where $x_k = \frac{2l_k - 1}{2}$ and $l_k \in \{0, 1\}$ represents the label of the branch (σ', σ). In the iterative decoding, only the extrinsic information $L_e(u_k)$ will be fed to the successive decoder as *a priori* value $L(u_k)$. Let $\bar{\alpha}_k(\sigma) = \log(\alpha_k(\sigma))$ and $\bar{\beta}_k(\sigma') = \log(\beta_k(\sigma'))$, and introduce the approximation $\log(e^a + e^b) \approx \max(a, b)$, we easily obtain the Max-Log-MAP realization of MAP rule as

$$L_R(u_k) = L_{in}(u_k) + \max_{(\sigma',\sigma)\in B_k^1(C)} (\bar{\alpha}_{k-1}(\sigma') + \bar{\beta}_k(\sigma)) - \max_{(\sigma',\sigma)\in B_k^0(C)} (\bar{\alpha}_{k-1}(\sigma') + \bar{\beta}_k(\sigma)) \quad (2)$$

where

$$\bar{\alpha}_k(\sigma) = \max_{(\sigma',\sigma)\in B_k(C)} \left(\bar{\alpha}_{k-1}(\sigma') + x_k \cdot L_{in}(u_k) \right), \quad (3)$$

$$\bar{\beta}_{k-1}(\sigma') = \max_{(\sigma',\sigma)\in B_k(C)} \left(\bar{\beta}_k(\sigma) + x_k \cdot L_{in}(u_k) \right). \quad (4)$$

At the beginning of iterative decoding, we usually assume $P(u_k = 1) = P(u_k = 0)$. Thus, it can be shown from (2), (3), and (4) that, throughout the entire iterative decoding process, all the extrinsic information and intrinsic information will contain the channel probability L_c as a factor. Therefore, we can eliminate the multiplication of L_c without any performance degradation and intrinsic information $L_{in}(u_k)$ becomes $r_k + L(u_k)$, where the $L(u_k)$ is a scaled *a priori* value: $\frac{1}{L_c} \log \frac{P(u_k=1)}{P(u_k=0)}$.

C. An Earlier Stopping Criterion for GLD decoding

For $(N, 2, n)$ GLD codes, after the decoding of each super-code (C^1 or C^2), we will obtain the hard decision $\hat{c}_i$ of each bit. Because GLD codes have good distance properties (the minimum distance scales linearly with the block length), it's intuitive that we can use the parity check result as an earlier stopping criterion: if $H \cdot \hat{\mathbf{c}} = 0$, then the decoder stops and $\hat{\mathbf{c}}$ is output as the decoding result. Theoretically, using such stopping criterion will incur some *undetected errors*. Because of their good distance properties, the rate of occurrence of undetected errors is extremely small. In all our simulations with GLD code of block length greater than 1000, undetected errors have never occurred when the stopping criterion is being used. Therefore, if the GLD code length is large enough, we may believe that such adaptive approach empirically doesn't make undetected errors.

Moreover, it has been shown in [6] that Max-Log-MAP algorithm makes the same hard decisions as the well-known Viterbi algorithm, in another word, Max-Log-MAP performs the maximum likelihood (ML) decoding. So if we use Max-Log-MAP algorithm to decode N/n constituent codes of each super-code, the hard decision $\hat{\mathbf{c}}$ obtained after decoding each super-code (C^1 or C^2) is a valid super-codeword ($H^1 \cdot \hat{\mathbf{c}} = 0$ or $H^2 \cdot \hat{\mathbf{c}} = 0$). Thus, to check whether $\hat{\mathbf{c}}$ is a valid GLD

code, we only need to multiply $\hat{\mathbf{c}}$ with the parity check matrix of another super-code (H^2 or H^1) in stead of the entire parity check matrix H.

We note that in this paper, under the assumption of BPSK modulation and AWGN channel transformation, the GLD codes simulations are presented for two different GLD code configurations, *i.e.*, (a) $(4035, 2, 15)$ GLD code with Hamming $(15, 11)$ constituent code, the GLD code rate is 0.467, and (b) $(4061, 2, 31)$ GLD code with Hamming $(31, 26)$ constituent code, the GLD code rate is 0.677. The earlier stopping criterion is used in all the simulations and undetected errors have never occurred.

III. Normalized Max-Log-MAP Decoding

When a Max-Log-MAP decoder is fed with Gaussian distributed input, the output extrinsic information also approximate Gaussian distribution. Using the similar analysis method presented in [8], we have: The probability density function of the Max-Log-MAP output extrinsic information v given input bits $u = 0$ can be approximately described as

$$p(v|u=0) = \frac{1}{\sqrt{2\pi}\sigma_v} e^{-\frac{1}{2\sigma_v^2}(v-\mu_v)^2} \tag{5}$$

where μ_v and σ_v^2 are the mean and variance of v. Let $L_u = \log \frac{P(u=1|v)}{P(u=0|v)}$ denote the conditional LLR, given the observation of Max-Log-MAP output v, and assume $P(u = 1) = P(u = 0)$. Using Bayes' rule, we have

$$L_u = \log \frac{P(v|u=1)}{P(v|u=0)} = -\mu_v \frac{2}{\sigma_v^2} v. \tag{6}$$

The above result indicates that a scaling factor $c = -\mu_v \frac{2}{\sigma_v^2}$ should be used to normalize the output extrinsic information of Max-Log-MAP decoder in order to obtain the more accurate *a priori* value for the next decoding step (unless $c = 1$). Moreover, as discussed in section II-B, in the practical implementation of Max-Log-MAP, the multiplication by the channel probability L_c is typically eliminated and the extrinsic information provides a scaled *a priori* value: $\frac{1}{L_c} \log \frac{P(v|u=1)}{P(v|u=0)}$. So the scaling factor for extrinsic information in this case should be

$$c = -\frac{1}{L_c} \mu_v \frac{2}{\sigma_v^2}. \tag{7}$$

When Max-Log-MAP is used in the iterative decoding of GLD code, we found that, during the first few iterations, the extrinsic information is Gaussian distributed in good approximation. But with the continuation of the iterative decoding, the extrinsic information can't be approximated as Gaussian distribution anymore, as shown in Fig. 2.

Therefore, during first several iterations we could calculate the expected scaling factor using (7) to normalize the extrinsic information. For practical implementation, we may calculate

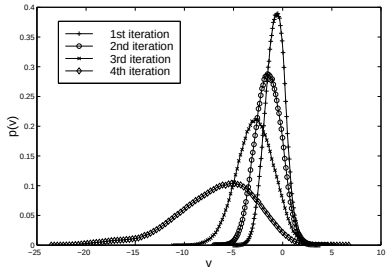

Fig. 2. Probability density function of Max-Log-MAP output at different number of iterations in (4035,2,15) GLD decoding.

the expected scaling factors based on simulations and then use them as the preset values in the real decoding. With the continuation of iteration, although the extrinsic information can't be well approximated as Gaussian distribution, we found that multiplying the extrinsic information with a scaling factor still improves the performance. However, in such cases, have to apply the near-exhaustive trials to identify the suitable preset scaling factors.

Theoretically, the preset scaling factors should include a series of scaling factors corresponding to different SNR values, different GLD code configurations and different iteration numbers, which will make things quite complicated. From our simulations, we observed that nearly all the expected scaling factors lie between 0.7 and 0.9, and small modification of these values won't affect the performance much. Thus, for the sake of efficient implementation, we propose to use a fixed scaling factor 0.75 throughout the entire GLD decoding, which is referred as *normalized Max-Log-MAP algorithm* in the following. The performance improvement gained by using this normalized Max-Log-MAP is shown in Fig. 3. We can see that, in both cases, the performance difference between Log-MAP and normalized Max-Log-MAP is always less than 0.1 dB. Compared with conventional Max-Log-MAP, by introducing a fixed normalization scaling factor, normalized Max-Log-MAP improves the performance significantly. Moreover, Fig. 4 depicts the average number of iterations, where maximum iteration number equals to 5 for both cases.

IV. Complexity Reduction Techniques

In this section, based on the fact that Max-Log-MAP performs ML decoding and there is an earlier stopping criterion in GLD decoding, we present two techniques to reduce the GLD decoding complexity without any performance degradation. We first introduce the following denotations: The decoding of each super-code consists of N/n Max-Log-MAP decoding tasks, each task is denoted as $Dec^{(i)}$, $1 \le i \le N/n$. Let $\mathbf{L}_{in}^{(i)}$ denote the intrinsic information fed to $Dec^{(i)}$ and $\mathbf{c}_{in}^{(i)}$ denote the hard decision on $\mathbf{L}_{in}^{(i)}$. The hard decision on output LLR $L_R(u_k)$ of each $Dec^{(i)}$ is denoted as $\mathbf{c}_{out}^{(i)}$. If $\mathbf{c}_{in}^{(i)}$ is a valid

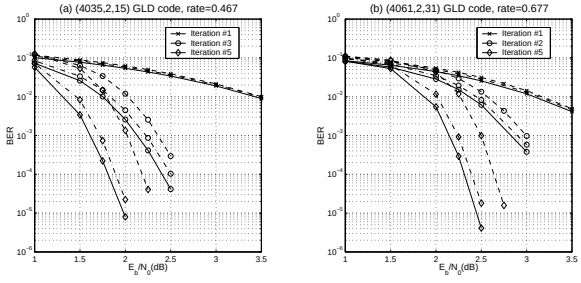

Fig. 3. Performance comparison. In both cases, solid lines correspond to Log-MAP, dash dot and dash lines for normalized Max-Log-MAP and conventional Max-Log-MAP, respectively.

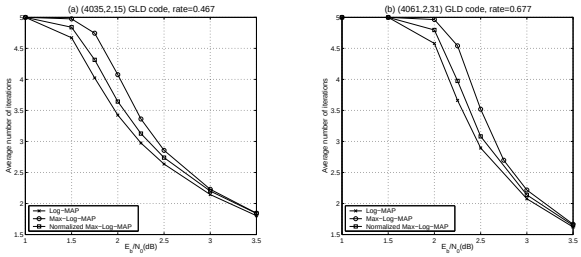

Fig. 4. Average of iteration numbers.

constituent codeword, that is $H_0 \cdot \mathbf{c}_{in}^{(i)} = 0$, the decoding task $Dec^{(i)}$ is referred as P-type decoding task, otherwise, $Dec^{(i)}$ is referred as F-type decoding task.

A. Task Scheduling

The first technique, *task scheduling*, can effectively eliminate all the *unnecessary* decoding tasks in the final decoding step. The final decoding step means the decoding of one super-code after which we obtain a valid GLD codeword. Since each constituent code decoding task $Dec^{(i)}$ is either F-type or P-type, we can put all N/n tasks in the final decoding step into two disjoint sets $\mathbf{P}$ and $\mathbf{F}$ which contain all the P-type and F-type tasks, respectively.

From the discussion in section II-C, we know that Max-Log-MAP makes the hard decision representing a ML path through the trellis. For each $Dec^{(i)} \in \mathbf{P}$, its $\mathbf{c}_{in}^{(i)}$ is already a valid constituent codeword and we may easily prove that $\mathbf{c}_{out}^{(i)}$ produced by each $Dec^{(i)} \in \mathbf{P}$ is always identical to $\mathbf{c}_{in}^{(i)}$. Therefore in the final decoding step, only performing the F-type decoding tasks is sufficient to obtain the final valid GLD codeword. With the convergence of GLD decoding, it's reasonable to expect that the number of F-type decoding tasks in the final decoding step is very small compared with the total decoding task number N/n. Moreover, as shown in Fig. 4, the average number of iterations in GLD decoding is not large. So if we only perform the F-type decoding tasks in the final decoding step, the decoding complexity will be reduced significantly. However, the crucial problem here is that before completing each super-code decoding, we never know whether this step is the final one or not.

To solve the above problem, we can simply be so optimistic that we expect each super-code decoding will be the final step. Thus prior to each super-code decoding, we always put N/n decoding tasks into $\mathbf{P}$ and $\mathbf{F}$ which contain all the P-type and F-type tasks, respectively. Let $R_F = |\mathbf{F}|$ and $R_P = |\mathbf{P}|$. First we perform the R_F F-type decoding tasks, then, if we don't get a valid GLD codeword, perform the remaining R_P P-type decoding tasks. Using such task scheduling technique, the corresponding flow diagram of each super-code decoding becomes as shown in Fig. 5(b).

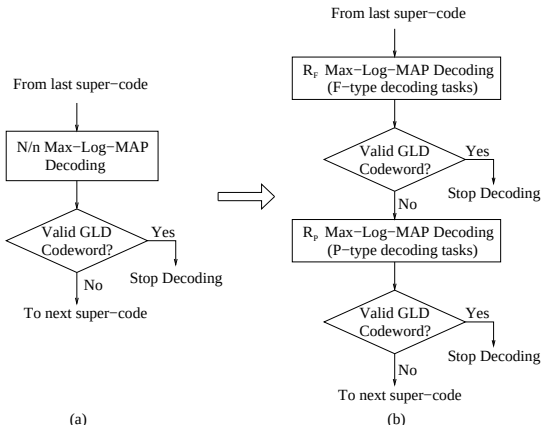

Fig. 5. One super-code decoding flow diagram (a) original (b)task scheduling.

It's clear that, except in the final decoding step, both original approach (Fig. 5(a)) and task scheduling approach (Fig. 5(b)) perform N/n decoding tasks. For the latter, we also need to perform N/n $H_0 \cdot \mathbf{c}_{in}^{(i)}$ operations to distinguish between F-type tasks and P-type tasks and perform one more parity check, $H^1 \cdot \hat{\mathbf{c}}$ or $H^2 \cdot \hat{\mathbf{c}}$, which will incur some extra implementation complexity and make the control mechanism more complicated. Since all these extra operations only involve simple logic computations, *i.e.*, XOR and AND, it's reasonable to expect their complexities are ignorable compared with each constituent code decoding task. Through our simulations, the above task scheduling technique turns out to be a very effective approach to reduce the entire GLD decoding complexity. Table I shows the average decoding task savings of task scheduling approach over original approach without scheduling. Obviously, all the savings are achieved in the final decoding steps.

B. Reduced Search Max-Log-MAP

Another technique, called *reduced search* Max-Log-MAP, is used to reduce the complexity of P-type decoding task. As shown in (2), Max-Log-MAP produces the extrinsic informa-

TABLE I
THE AVERAGE DECODING TASK SAVINGS FOR (4035,2,15) GLD CODE AND (4061,2,31) GLD CODE, WHERE NORMALIZED MAX-LOG-MAP IS USED AND MAXIMUM ITERATION NUMBER IS 5.

SNR	2.0 dB	2.5 dB	3.0 dB	3.5 dB
(4035,2,15) code	12.4%	15.7%	20.0%	23.9%
(4061,2,31) code	3.1%	12.6%	18.7%	24.9%

tion $L_e(u_k)$ based on two paths *per step*: the best with bit zero and the best with the bit one at each time-k, one of these two paths will always be the ML path [6]. Therefore one of the two terms for computing $L_e(u_k)$ in (2) will equal to $(\bar{\alpha}_{k-1}(\sigma'_{ML}) + \bar{\beta}_k(\sigma_{ML}))$, where branch $(\sigma'_{ML}, \sigma_{ML})$ is on the ML path. If we have known the ML path before computing the extrinsic information, one term can always be computed using only one addition, thus the computations associated with computing extrinsic information can be reduced by nearly $1/2$. From (2), (3) and (4), we also know that the complexities of computing $\bar{\alpha}_k$, $\bar{\beta}_k$ and $L_e(u_k)$ are nearly equal. So we may expect that, if we know the ML path prior to computing $L_e(u_k)$, the entire computation complexity can be roughly reduced by $1/6$. We refer such approach as reduced search Max-Log-MAP.

In general cases, in order to find out the ML path before computing $L_e(u_k)$, we have to complete the two recursions (for computing all $\bar{\alpha}$ and $\bar{\beta}$) first, then by tracing back $\bar{\alpha}$ or $\bar{\beta}$ along the trellis T to obtain the ML path (just like in the Viterbi algorithm), finally calculate extrinsic information L_e using the reduced search Max-Log-MAP. Since each $L_e(u_k)$ at time-k can be calculated as soon as both $\bar{\alpha}_{k-1}$'s and $\bar{\beta}_k$'s are available, the above approach actually trades decoding speed for computation complexity and will lead to longer decoding. In the following, we show that we can apply the reduced search Max-Log-MAP to P-type decoding task without incurring extra delay.

Let $Dec^{(i)}$ be a P-type decoding task. As discussed earlier, we know that $\mathbf{c}_{out}^{(i)}$ will be identical to $\mathbf{c}_{in}^{(i)}$, which means that $\mathbf{c}_{in}^{(i)}$ corresponds to the ML path in current Max-Log-MAP decoding. Therefore we can simply use $\mathbf{c}_{in}^{(i)}$ to identify the ML path before the execution of P-type decoding task. Furthermore, in the P-type decoding task, the complexity of computing recursion metrics also can be slightly reduced: In stead of using (3) and (4), the computations of $\bar{\alpha}_k(\sigma)$ and $\bar{\beta}_k(\sigma)$ of the states along the ML path can be performed as:

$$\begin{aligned} \bar{\alpha}_k(\sigma_{ML}) &= \bar{\alpha}_{k-1}(\sigma'_{ML}) + x_k \cdot L_{in}(u_k), \\ \bar{\beta}_{k-1}(\sigma'_{ML}) &= \bar{\beta}_k(\sigma_{ML}) + x_k \cdot L_{in}(u_k). \end{aligned}$$

Based on our simulation results as shown in Table II, the occurrence rates of P-type decoding task during the entire GLD decoding process are very high, which indicates that applying the reduced search Max-Log-MAP to only P-type decoding tasks is an effective approach to reduce the computation complexity without incurring extra delay. We note that, as shown in Table II, when reduced search Max-Log-MAP is used together with task scheduling, the savings will be smaller compared with only using reduced search Max-Log-MAP. The reason is that, when using task scheduling, we don't count the P-type decoding tasks in the final decoding step since such tasks won't be actually executed in this case.

TABLE II
THE OCCURRENCE RATE OF P-TYPE DECODING TASK IN DECODING OF (4035,2,15) GLD CODE AND (4061,2,31) GLD CODE, WHERE NORMALIZED MAX-LOG-MAP IS USED AND MAXIMUM ITERATION NUMBER IS 5.

SNR		2.0 dB	2.5 dB	3.0 dB	3.5 dB
(4035,2,15) GLD code	I	47.6%	52.7%	57.2%	61.5%
	II	36.0%	37.0%	37.4%	37.6%
(4061,2,31) GLD code	I	26.2%	42.0%	49.2%	55.5%
	II	23.7%	29.5%	30.7%	30.6%

[1]I: without task scheduling; II: with task scheduling

V. CONCLUSIONS

This paper has presented several techniques to improve performance and reduce complexity for practical implementations of GLD decoder when Max-Log-MAP is used to decode constituent codes. A normalized Max-Log-MAP algorithm is proposed to improve the performance of GLD codes up to less than 0.1 dB inferior to GLD decoding based on Log-MAP algorithm. Moreover, two effective techniques have been proposed to effectively reduce the GLD decoding complexity without any performance degradation. With the good performance and significantly reduced decoding complexity, Max-Log-MAP turns out to be a very attractive constituent code decoding algorithm for the practical GLD decoder implementation.

REFERENCES

[1] R. G. Gallager, *Low-Density Parity-Check Codes,* M.I.T Press, 1963.

[2] M. Lentmaier and K. S. Ziganfirov, "Iterative decoding of generalized low-density parity-check codes", *in Proceedings of IEEE International Symposium on Information Theory*, p. 149, 1998.

[3] J. Boutros, O. Pothier, and G. Zemor, "Generalized low density (Tanner) codes", *in Proceedings of ICC'99*, pp. 441–445, Vancouver, June 1999.

[4] O. Pothier, "Compound codes based on graphs and their iterative decoding", Ph.D. thesis, Ecole Nationale Suprieure des Tlcommunications, Jan. 2000.

[5] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate", *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, March 1974.

[6] P. Robertson, E. Villeburn, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain", *in IEEE International Conference on Communications, ICC'95*, vol. 2, pp. 1009–1013, Seattle, 1995.

[7] S. Lin, T. Kasami, T. Fujiwara, and M. Fossorier, *Trellises and Trellis-Based Decoding Algorithms for Linear Block Codes*, Kluwer Academic Publishers, 1998.

[8] L. Papke, P. Robertson, and E. Villebrun, "Improved decoding with the SOVA in a parallel concatenated (turbo-code) scheme", *in IEEE International Conference on Communications, ICC'96*, vol. 1, pp. 102–106, 1996.