LDPC Coded M-ary Orthogonal Signaling with Noncoherent Detection
Supawat Supakwong, Stephen G. Wilson
Department of Electrical and Computer Engineering
University of Virginia
Charlottesville, VA 22904

Abstract —

We study the combination of LDPC coding and M-ary orthogonal modulation with noncoherent detection on AWGN and Rayleigh fading channels, focusing upon the role of alphabet size M, blocklength N, and the role of iterative demapping of the channel-to-bit message data in the iterative message passing decoder.

I. INTRODUCTION

Use of M-ary orthogonal modulation, say with FSK, together with noncoherent detection represents a logical communication strategy in situations where phase coherence of the demodulator is impossible or difficult to establish. Applications include fast frequency hopping scenarios, and fast channel fading conditions.

In this paper, we study the performance of coded M-ary orthogonal modulation, where the coding is produced by a binary LDPC code, with code bits mapped in natural manner to the modulator symbols. Much has been written recently on LDPC codes, known to be capable of approaching channel capacity at least with binary, coherent transmission, provided the LDPC code degree profiles are carefully selected and the code blocklength becomes large. Our interest here is in performance for noncoherent transmission with moderate blocklengths more typical of practice, and in assessing degradations relative to coherent detection of the same code, and relative to channel capacity. We study both non-fading (AWGN) and fully-interleaved Rayleigh flat fading, and slow Rayleigh fading channel models.

II. PROBLEM DEFINITION

We assume a binary message of length L bits is encoded by a rate R LDPC code, producing a coded vector of length \( N = L/R \) bits. This code is described completely by the parity check matrix for the code, \( \mathbf{H} \), a sparse \( (N-L) \times N \) binary matrix. This matrix defines connections between variable nodes (code bits) and check nodes in a bipartite graph. In our present work, the LDPC code is chosen to have rate \( R = 1/2 \), as suggested by channel capacity arguments given below, and also the code is regular, i.e. the degree of all variable nodes and check nodes are 3 and 6 respectively, values that have been found to be 'best' for moderate blocklength codes. Specific construction of the code follows Gallager’s procedure [1] for constructing regular codes with good graph properties. For background reading, references [2] and [3] represent recent treatments of LDPC codes and their performance.

To determine appropriate code rates, in bits/modulator symbol, one approach is to find the channel capacity of the orthogonal, noncoherent channel model as a function of \( E_b/N_0 \), then translate this into a plot of minimum \( E_b/N_0 \) versus \( R \) to locate best code rates. This has been done for example by Wilson [4], where it is found that for noncoherent detection there is a rather broad optimum range around \( \frac{1}{2} \log_2 M \) for all typical \( M \), and for both channel models considered here. Thus, we adopt rate 1/2 coding (the rate is for the binary code itself). Furthermore there is a significant noncoherent penalty relative to coherent detection in the coded regime. Readers are also referred to Stark [6] for related material on the capacity of these channel models.

The \( N \) code bits are mapped in sequential order (without further interleaving) to the \( M \)-ary modulator inputs, where \( M = 2^m \). The number of transmitted orthogonal symbols is \( S = N/m = N/\log_2 M \), and we assume \( N \) is chosen to make this an integer. With orthogonal signaling there are no issues of bit-to-symbol mapping as there are with, say, \( M \)-QAM modulation. The modulator produces one of \( M \) orthogonal signals for transmission over the channel. The noncoherent matched filter receiver outputs a vector of \( M \) real variables in response to each transmitted symbol, designated \( y_i = \{ y_{i0}, y_{i1}, \ldots, y_{i(M-1)} \} \), where \( i \in \{0,1,\ldots,S-1\} \). In the noncoherent detection model, one of the elements of each vector will be Rician distributed, while the remaining variables will be Rayleigh-distributed. These variables are jointly independent within a vector, and from vector to vector. In the case of Rayleigh fading, we assume either that the fading strength is known by the receiver, or that the fading amplitude is unknown (no channel state information). The detection algorithm below is the same for non-fading and fading scenarios, and the issues attached to the channel model only affect the demapping operation.

Figure 1 illustrates a factor graph [5] for the coded system that depicts the coded variable nodes, parity check nodes, the modulator mapping functions, and the channel observable data. Variable nodes are represented by solid dots and factor expressions by boxes. To perform decoding, we adopt standard message passing in the factor graph of Figure 1, following the recipes found in the literature (see [3] and [5] for example). Message passing provides a convenient distributed decoding algorithm that correctly marginalizes the probabilities of all variables in graphs without cycles. Codes of interest, however, ultimately have cycles of some girth, and message passing must be regarded as slightly suboptimal, relative to ML decoding. Much simulation experience of course suggests this is only a small degradation in practice.

The primary new issue here is how to produce soft information for the \( m \) code bits associated with a symbol \( s_i \), given that we have information about the received symbol vector \( y_i \), and any \textit{a priori} information about these code bits. That is, how should we “demap” the \( M \)-ary probability messages from the channel to binary probability messages to be sent up to the LDPC layer? Let \( v_i \) denote the vector of \textit{a priori} probability messages for the \( m \) code bits to the demapper \( i \). Suppose \( y_i \) is the received signal vector for coded symbol \( i \), and \( y_{i \sim j} \) is the portion of \( v_i \) excluding \( v_j \). The demodulator computes...
With an assumption of a cycle-free factor graph, we can define a log-likelihood ratio for each code bit $c_i$, meaning code bit $j$ of symbol $i$, in the form

$$L(c_i) = \log \frac{P(c_i = 0 | y_i, v_i(\sim j))}{P(c_i = 1 | y_i, v_i(\sim j))}$$

(1)

which is equivalent to

$$L(c_i) = \log \frac{\sum_{k \in \{0, 1, \ldots, M-1\}} P(s_k | y_i, v_i(\sim j))}{\sum_{k \in \{0, 1, \ldots, M-1\}, c_i=1} P(s_k | y_i, v_i(\sim j))}$$

(2)

To compute each term in the summation of (2), first apply Bayes’ rule:

$$P(s_k | y_i, v_i(\sim j)) = \frac{f(y_i | s_k, v_i(\sim j)) P(s_k | v_i(\sim j))}{f(y_i | v_i(\sim j))}$$

(3)

With an assumption of a cycle-free factor graph, we can assume that the messages $v_i(\sim j)$ are generated by using information from other symbols and are therefore independent of $y_i$. As a consequence, $f(y_i | s_k, v_i(\sim j)) = f(y_i | s_k)$. Also $P(v_i(\sim j) | s_k) = \prod_{p \neq j} P(c_p | s_k)$. This simplifies (3) to

$$P(s_k | y_i, v_i(\sim j)) = \frac{f(y_i | s_k, v_i(\sim j)) \prod_{p \neq j} P(c_p | s_k) P(s_k)}{f(y_i | v_i(\sim j)) P(v_i(\sim j))}$$

(4)

Thus,

$$f(y_i | s_k, v_i(\sim j)) \prod_{p \neq j} P(c_p | s_k) P(s_k)$$

$$L(c_i) = \log \frac{\sum_{k \in \{0, 1, \ldots, M-1\}, c_i=1} f(y_i | s_k, v_i(\sim j)) \prod_{p \neq j} P(c_p | s_k) P(s_k)}{\sum_{k \in \{0, 1, \ldots, M-1\}} f(y_i | s_k, v_i(\sim j)) \prod_{p \neq j} P(c_p | s_k) P(s_k)}$$

(5)

This equation can be further simplified by making the assumption that the $s_k$ are equal probable, and canceling some common terms. The log-extrinsic message for code bit $c_i$ is then given by

$$L(c_i) = \log \frac{\sum_{k \in \{0, 1, \ldots, M-1\}, c_i=1} \{f(y_i | s_k) \prod_{p \neq j} P(c_p | s_k)\}}{\sum_{k \in \{0, 1, \ldots, M-1\}} \{f(y_i | s_k) \prod_{p \neq j} P(c_p | s_k)\}}$$

(6)

Note the extrinsic message formulated here utilizes the channel data in the form of a likelihood function and the prior probabilities of all input bits other than the bit being calculated. The likelihood functions are straightforward to express, given the independent Rice-Rayleigh structure described above, and some cancelation of common terms simplifies the expression to one involving sums of modified Bessel functions.

$$L(c_i) = \log \frac{\sum_{k \in \{0, 1, \ldots, M-1\}, c_i=1} \{I_0(\frac{|y_i| s_k}{\sigma^2}) \prod_{p \neq j} P(c_p | s_k)\}}{\sum_{k \in \{0, 1, \ldots, M-1\}} \{I_0(\frac{|y_i| s_k}{\sigma^2}) \prod_{p \neq j} P(c_p | s_k)\}}$$

(7)

This completes the description of the demapping operation. The iterative decoding in the LDPC layer follows the standard LDPC formulas given in Ryan [7], for example. Summary of the complete decoding procedure is given here.

1. Generate the binary probability message, $\{L(c_i)\}$, to be sent to the LDPC layer by using

$$L(c_i) = \log \frac{\sum_{k \in \{0, 1, \ldots, M-1\}, c_i=1} \{I_0(\frac{|y_i| s_k}{\sigma^2}) \prod_{p \neq j} P(c_p | s_k)\}}{\sum_{k \in \{0, 1, \ldots, M-1\}} \{I_0(\frac{|y_i| s_k}{\sigma^2}) \prod_{p \neq j} P(c_p | s_k)\}}$$

(8)

Initially, $P(c_i | s_k)$ is set to 0.5.

2. After obtaining $\{L(c_i)\}$ from the modulator, update the log-probability of the up messages for the next iterative cycle denoted $\{L(q_i^{(k-1)}-k)\}$, to be sent from variable node $j$ of symbol $i$ to neighboring check node $k$ by using

$$L(q_i^{(k-1)-k}) = L(c_i) + \sum_{k' \in C_i \setminus k} L(q_i^{(k'-1)-j})$$

(9)

$L(q_i^{(k'-1)-j})$ is set to zero initially.
3. At the checknode $k$, calculate the log-probability of the down message, \( L(r_i^{k-j}) \), from check node $k$ to variable node $j$ of symbol $i$ using
\[
L(r_i^{k-j}) = \prod_{j' \in V_k \setminus j} \alpha_{j'}^{j-k} \phi(\sum_{j' \in V_k \setminus j} \beta_{j'}^{j-k})
\] (10)
where $\alpha_{j'}^{j-k} = \text{sign}[L(q_{j'}^{k-j})]$, $\beta_{j'}^{j-k} = |L(q_{j'}^{k-j})|$, and $\phi(x) = \log\left(e^x + 1 \over 2\right)$.

4. Update \( L(Q^i_j) \) using equation
\[
L(Q^i_j) = L(C^i_j) + \sum_{k' \in C_k} L(r_k^{k-j})
\] (11)

At this point, the decoder has completed one iteration beginning with sending a priori information about the code bits down to the demodulator, coupled with channel information to obtain \( \{L(c_i^j)\} \) (step 1). This message is sent back up to the LDPC layer where it is used to update the up message, \( L(q_i^{k-j}) \), for the next iteration (step 2). Then this obtained \( L(q_i^{k-j}) \) will be used to calculate \( L(r_i^{k-j}) \), which is the down message from the check node back to the variable node where the iteration starts (step 3). \( L(Q^i_j) \) is calculated to update the log-likelihood ratio of that code bit after completing an iteration. A hard decision is made as follows, for \( k = 0, 1, \ldots, N-1 \):
\[
\hat{c}_i^j = \begin{cases} 
1, & \text{if } L(Q^i_j) < 0 \\
0, & \text{otherwise}
\end{cases}
\] (12)

The decoding stops when a valid codeword is found, i.e. $\hat{c}_H^T = 0$, or a maximum number of iterations is exceeded, else returns to step 1. As described each iteration involves updating of all messages in the graph, including the demapping operation.

III. RESULTS

Performance Analysis
We provide simulation results for several situations:
- alphabet size $M = 4$ or $M = 8$;
- code blocklength $N = 1002$ or $N = 5010$;
- iterative demapping, or non-iterative demapping;
- AWGN, Rayleigh fading with full independence, and Rayleigh slow-fading;
- Reduced complexity by using hybrid schedules, max-log and min-sum approximations

We used a fixed, regular (3,6) rate 1/2 code design for all cases, and the simulation was run for 10000 blocks or until 10000 bit errors had been located. The all-zeros code vector was selected for transmission, with no loss of generality. Iterative decoding results are presented in Figure 2 and 3 for $M = 4$ and $M = 8$ respectively, showing decoded bit error probability for the AWGN (non-fading) model. We also show with a vertical line the limiting $E_b/N_0$ implied by the channel capacity calculations, [4]

Several conclusions can be developed, with no real surprises. First, the effect of increasing blocklength makes the performance characteristic shift toward the capacity limit, and the curve becomes steeper. Second, comparing Figures 2 and 3, we see a roughly 0.8 dB advantage for the 8-ary choice, a
result accurately predicted by channel capacity calculations. (Of course, the demapping complexity is higher for this case.) Finally, and perhaps of most interest, is the effect of iterative demapping versus one-time, or non-iterative demapping. Non-iterative demapping is one way to keep the decoder complexity low by iterating only in the LDPC layer and never updating messages from the demapper. Simulation results show that iterative demapping buys only a small improvement, roughly 0.2 dB for $M = 4$ and about 0.4 dB for $M = 8$. We suggest the performance gains are somewhat smaller here than found in QAM modulation due to the orthogonal signal constellation. With iterative demapping and the longer blocklength, the gap from capacity is about 1.5 dB for the 4-ary modulation, and about 2.0 dB for 8-ary modulation. These are typical of results for comparable codes combined with binary PSK and coherent demodulation.

Results for Rayleigh fast and slow fading are shown in Figure 4 for 8-ary modulation, with $N = 1002$. For reference the AWGN performance is given, and fast fading performance is shown for both perfect channel state information and unknown channel state information, while slow fading performance is shown only for the perfect channel state information case. We observe a penalty of several dB is exacted by fast fading, but the performance curve remains steep due to the large diversity attached to fast fading. As we expect, slow fading performs much worse than the fast fading case, with the slope of -1 on this log-log display.

Complexity Reduction

Computational time and complexity are among the important issues to keep in mind when implementing the decoder. In this paper, we present three practical approaches to reduce this complexity.

1. **Hybrid schedules**
   As shown in Figure 2 and 3, fully iterative message passing improves bit error performance by a small fraction of a dB as compared to the non-iterative demapping. The demapper complexity, however, grows as $O(M \log_2 M/R)$ per information bit, per demapping iteration. Referring to equation (8), $\log_2 M$ multiplications are needed for each of the $M$ terms in the summation to compute $L(c_i)$. This can dominate the total complexity per iteration. For example with a rate 1/2 code, column weight $w_c = 3$, and row weight $w_r = 6$, the demapping complexity is half of the total complexity. A possible design tradeoff involves “hybrid” schedules, that is, we schedule the decoder to iterate in the LDPC layer for $T$ cycles before a next demapping. Results when $T = 4, 6$, and 9 are given in Figure 5 for the AWGN case. With $T = 4$ and 6, the performances are just as good as the fully-iterative case, while the computational demapping complexity is reduced by the factor of $T$.

2. **Min-sum approximations**
   The second complexity issue occurs in the LDPC layer at the check node when generating the down message $L(r_k^{-1})$ by using equation (10). In this expression, we need to calculate $\alpha_j^{i-k} \phi(\sum_{j' \in V_k \setminus j} \phi(\beta_j^{i-k}))$, which $\phi$ is not a simple function to calculate. Moreover, $w_r - 2$ addition operations are needed to compute the summation. This complexity could be significantly reduced by using the min-sum approximation proposed in [8].

---

**Figure 4:** Performance of coded 8-ary modulation, Rayleigh fading, $N=1002$

**Figure 5:** Effect of scheduling
The idea is that the \( \phi \) - function of the minimum \( \beta \) usually dominates. Thus, by approximating the sum of all terms to just a minimum term, the summation is unnecessary. This reduces the equation to
\[
\phi(\sum_{j \in V_k \cup \tilde{V}} \phi(\beta_j^{1-k})) \approx \phi(\min_{j \in V_k \cup \tilde{V}} \beta_j^{1-k}) \tag{13}
\]
Since \( \phi^{-1}(x) = \phi(x) \), we further have
\[
\phi(\sum_{j \in V_k \cup \tilde{V}} \phi(\beta_j^{1-k})) \approx \min_{j \in V_k \cup \tilde{V}} \beta_j^{1-k} \tag{14}
\]
Results shown in Figure 6 demonstrate that while this approximation considerably reduces the decoder complexity, it degrades the bit error performance by only about 0.2 dB.

![Figure 6: Performance of coded 8-ary modulation with min-sum and max-log approximations](image)

3. **Max-log approximation**

A final complexity reduction is performed at the demapper when computing \( L(c_i') \) by using equation (8). Notice that the number of terms in each of the numerator/denominator sums is \( M/2 \) for each of \( \log_2 M \) bits per symbol. The overall computation of the iterative demapping operation, per information bit decoded, grows as \( O(M \log_2 M/R) \), as already shown in the previous subsection. This complexity can be reduced by using the max-log approximations suggested in [9]. The idea is that the log of the sum of two exponential terms can be approximated as the maximum of the two exponents, i.e.
\[
\log(a^A + a^B) \approx \max(A, B) \tag{15}
\]
This can be generalized to the case of approximating the log of the sum of more than two exponential terms as
\[
\log(\sum_{a \in A} a^{k_a}) = \log(a^{k_1} + a^{k_2} + \ldots + a^{k_n}) \approx \max_{i \in \{1, \ldots, n\}} (A_i) \tag{16}
\]
To apply this to our problem, we rearrange equation (8) as
\[
L(c_i') = \log \left( \sum_{\substack{a \in A \\mid \ c_i'^a \neq 0}} e_{a} \log I_a(S|c_i') \right) \approx \log \left( \sum_{\substack{a \in A \\mid \ c_i'^a \neq 0}} e_{a} \log P(C_i'|S_k) \right)
\]
\[
= \log \left( \sum_{\substack{a \in A \\mid \ c_i'^a \neq 0}} e_{a} \log I_a(S|c_i') \right) = \max_{i \in \{1, \ldots, n\}} \left( A_i \right)
\]
Now applying the max-log approximation from equation (16):
\[
L(c_i') \approx \max_{i \in \{1, \ldots, n\}} \left( A_i \right) + \sum_{p \neq j} \log P(C_i'|S_k)
\]
There are several advantages in using this approximation. First, only the maximum terms from the numerator/denominator are needed for the computation. Comparators are used to find the maximum terms, instead of using adders to find the total sum. Second, each term in the original summations can now be computed by summing the log-values, instead of multiplying all the terms together. It is well known that operations in the log-domain gives better numerical stability. Finally, notice that in the previous formulation when we compute \( L(c_i') \) we need to find \( P(c_i') \) for each code bit by using \( L(Q_i') = \log P(C_i'|S_k)/P(C_i' = 0)/P(C_i' = 1) \) and also \( P(c_i' = 0) + P(c_i' = 1) = 1 \). In other words, we have to get out of the log domain every time we update \( L(c_i') \). Further simplifications not presented here show that the max-log approximation allows us to stay in the log-domain for the entire process. Simulation results shown in Figure 6 indicate there is less than 0.1 dB penalty with this max-log approximation. The results also show that when both min-sum and max-log approximations are used, the degradation is dominated by the min-sum approximation.
IV. Conclusions

We have studied iterative decoding of binary LDPC codes coupled with $M$-ary orthogonal signals and noncoherent detection. The complete message passing algorithm has been presented for this case with focus on the demapping operation. Results for AWGN and Rayleigh fading channels support a hybrid scheduling algorithm that performs demapping every $T$ cycles. We conclude with several simplifications to the basic decoding algorithm that impose only a small SNR penalty.

References