CS6782 Probabilistic Graphical Models
Decoding LDPC Codes

Skand Hurkat (sh953)

December 13, 2013

Abstract

In this project, I study the application of two techniques to the belief-propagation based decoding of LDPC codes. The first technique, min-sum decoding was implemented in the 1980s by Tanner as a method to reduce computational complexity. The other technique, residual belief propagation optimises the order in which message updates are scheduled in an informed manner, leading to faster and better convergence. The two methods are combined and experimentally evaluated.

1 Introduction

Low density parity check (LDPC) codes were invented by Gallager in the 1960s[8]. Impractical to implement at the time, they were essentially forgotten for over three decades until they were rediscovered in the 1990s[13, 14] as a compelling alternative to Turbo codes as approaching the Shannon limit. The development of the iterative belief propagation algorithm led to decoding methods that were linear in time to the block length of the code; and since the 1990s, there has been a flurry of effort in the design and analysis of LDPC codes (eg: [4, 5, 20, 6]). These codes are one of the few for which an analytical framework exists that fives guarantees on asymptotic performance for large block lengths, as well as for evaluating the performance compared to the Shannon capacity. As a consequence, it is possible to design LDPC codes which outperform turbo codes for large block lengths, and this has led to the proliferation of LDPC codes into multiple standards, such as DVB-S2[1] used in satellite television, ITU-T G.hn[2] home network, and 802.11n WiFi[11].

This project tries to combine research in LDPC codes with research in belief propagation. Specifically, it explores the effect of the min-sum decoding algorithm first used by Tanner[18] and the residual based belief propagation algorithm proposed by Elidan et al[7]. The min-sum algorithm was proposed as a method to allow for simpler decoding on embedded devices including digital signal processors (DSPs) and field programmable gate arrays (FPGAs). The residual based approach works on the premise that not all message updates are equally important, and tries to prioritise message updates to achieve faster convergence, at the cost of adding some bookkeeping overhead. This project aims to combine the two techniques and compare performance, both in terms of decoding performance, and time complexity.

This report is organised as follows: section 2 describes some of the basic concepts in information and coding theory, including some basics of linear block codes and LDPC codes. Section 3 describes the decoding of LDPC codes in detail, including the Tanner graph representation of LDPC codes, and the iterative belief propagation algorithm. Section 4 discusses the two techniques, namely the min-sum decoding algorithm and the residual belief propagation algorithm. Section 5 describes the experimental setup used in this project. Section 6 discusses the results obtained.

2 Information and Coding Theory Basics

One of the most fundamental theorems in information theory is the Shannon-Hartley theorem, which relates the information capacity of the network to the signal to noise ratio. In the baseband domain, this theorem can be related as [15]

$$I = \frac{1}{2}\log_2\left(1 + \frac{S}{N}\right) \tag{1}$$

where I is the information that can be sent (in terms of bits per symbol, with error correction) and $\frac{S}{N}$ is the signal to noise ratio (SNR) for the channel. A capacity approaching code is one that reaches the Shannon limit, i.e. the error rate falls off dramatically on approaching the Shannon SNR.

An (n,k) block binary code is a mapping of k information bits to an n bit long codeword. For the purposes of this discussion, we shall denote the set of codewords by $\mathscr{C}$.[1]

A binary *linear* code satisfies the property that

$$\{\mathbf{x}_1, \mathbf{x}_2\} \in \mathscr{C} \Rightarrow \mathbf{x}_1 + \mathbf{x}_2 \in \mathscr{C} \tag{2}$$

where addition is defined over the binary field so it corresponds to the XOR operation.

Generator matrix A binary linear code $\mathscr{C}$ forms a k dimensional subspace within an n dimensional space over the binary field. We can decompose the subspace into k basis vectors $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$ so that

$$\mathbf{x} = \sum_i u_i \mathbf{v}_i \tag{3}$$

where $\{u_1, u_2, \ldots, u_k\}$ are the coefficients of basis expansion with respect to $\{\mathbf{v}_i\}$. Generally, $\{u_i\}$ is chosen to represent the information bits.

This operation can be rewritten in matrix form as

$$\mathbf{x} = \mathbf{uG} \tag{4}$$

where

$$\mathbf{G} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_k \end{bmatrix} \tag{5}$$

is called the generator matrix for the code $\mathscr{C}$.

Dual code and parity check matrix Given that $\mathscr{C}$ is a k dimensional subspace within an n dimensional space, there will be an $(n-k)$ dimensional subspace $\mathscr{C}^\perp$ which will be orthogonal to every codeword in $\mathscr{C}$. $\mathscr{C}^\perp$ can be thought of as an $(n, n-k)$ binary linear code, which is the *dual code* for $\mathscr{C}$. So, we have

$$\mathbf{x} \in \mathscr{C}, \mathbf{y} \in \mathscr{C}^\perp \Rightarrow \langle \mathbf{x}, \mathbf{y} \rangle = 0 \tag{6}$$

If $\mathbf{H}$ denotes the generator matrix for $\mathscr{C}^\perp$, i.e. the rows of H form a basis for $\mathscr{C}^\perp$, we have

$$\mathbf{Hx}^T = \mathbf{0} \tag{7}$$

Hence, each row of $\mathbf{H}$ forms a *parity check equation* for the codeword $\mathbf{x}$, which is why $\mathbf{H}$ is often referred to as the *parity check matrix* for $\mathscr{C}$.

It is interesting to note that neither the parity check matrix nor the generator matrix are unique for a code $\mathscr{C}$, as any row operations on either matrix will not change the subspace $\mathscr{C}$ or $\mathscr{C}^\perp$.

[1]A major part of this and the next section is covered in section 7.3 of the book *Fundamentals of Digital Communication*[16].

Systematic codes A code is termed *systematic* if information bits can be directly read from codeword bits. Conversely, if not all information bits can be read from codeword bits, the code is termed *non-systematic*.

For a systematic binary linear code, the generator matrix generally takes the form [12]

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_k & \mathbf{P} \end{bmatrix} \tag{8}$$

where $\mathbf{I}_k$ is a $k \times k$ identity matrix and $\mathbf{P}$ corresponds to the set of equations representing the *parity bits* in the codeword. This can be seen as

$$\mathbf{x} = \mathbf{uG} = \mathbf{u}\begin{bmatrix} \mathbf{I}_k & \mathbf{P} \end{bmatrix} = \begin{bmatrix} \mathbf{u} & \mathbf{uP} \end{bmatrix} \tag{9}$$

When the generator matrix is represented in such a systematic form, the parity check matrix can also be represented in a systematic form as

$$\mathbf{H} = \begin{bmatrix} \mathbf{P}^T & \mathbf{I}_{n-k} \end{bmatrix} \tag{10}$$

so that

$$\mathbf{HG}^T = \mathbf{P}^T + \mathbf{P}^T = \mathbf{0} \tag{11}$$

A codeword $\mathbf{x}$ is decoded through the following algorithm. First, a parity check is run. If the parity check is satisfied ($\mathbf{Hx}^T = \mathbf{0}$), then the information bits $\mathbf{u}$ are recovered by a reverse lookup table mapping $\mathbf{x} \in \mathscr{C}$ to $\mathbf{u}$. If the parity check is not satisfied, an error correction algorithm is run until the parity check is satisfied, and then the reverse lookup is executed.

It can be shown (though not in this report) that the performance of a binary linear code $\mathscr{C}$ is determined by the *minimum Hamming distance* between its codewords and not by the form of the generator matrix. Hence, the performance of the code $\mathscr{C}$ does not depend on whether the code was systematic or not. As systematic codes are easier to decode (they do not involve a $\Theta(2^k)$ search over the set of codewords to find the information bits, rather the information bits can be directly read from the codeword), this project considers only the systematic form of the generator matrix and the parity check matrix.

Low density parity check codes are so called because both the generator and parity check matrices for the codes are sparse. However, this does not mean that *any* parity check matrix for the given code $\mathscr{C}$ is sparse, rather it means that the parity check matrix we use for decoding is sparse. Other parity check matrices corresponding to the code need not be sparse. (The sparseness of the parity check matrix is an important factor in the iterative decoding algorithm discussed in the next section.) Gallger showed[8] that the linear (Hamming) distance between the codes typically increases linearly with block length, which leads asymptotically to the Shannon capacity for large block lengths.

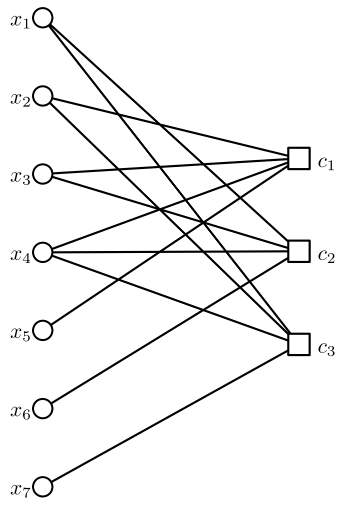

Figure 1: Tanner graph for Hamming code given in Equation 12

3 Decoding LDPC Codes

This section discusses the Tanner graph representation of LDPC codes, code construction, and decoding using iterative belief propagation.

3.1 Tanner Graph Representation

The Tanner graph is an important concept in decoding binary linear codes. Given a parity check matrix **H**, the Tanner graph is a bipartite graph with variable nodes on the left hand side, and parity check nodes on the right hand side, one check node for each row of the parity check matrix. Denoting the variable nodes by x_j and the check nodes by c_i, x_j is connected to c_i if $H_{ij} = 1$.

Given the standard $(7,4)$ Hamming code (single error correcting, double error detecting), the parity check matrix is given by

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

the Tanner graph representing this code is given in Figure 1.

Regular LDPC codes Regular LDPC codes have exactly d_v ones in each column and d_c ones in each row. So, the Tanner graph for a regular LDPC code has exactly d_v edges emanating from each variable node, and d_c edges emanating from each check node. Gallager provided a recipe for generating regular LDPC codes in his original proposal[8], however this report shall not discuss the creation of regular LDPC codes.

Irregular LDPC codes Irregular LDPC codes do not have any uniform structure. Both variable nodes and check nodes can have a range of degrees. These codes are generally represented by two polynomials

$$\lambda(x) = \sum_i \lambda_i x^{i-1} \tag{13}$$

$$\rho(x) = \sum_i \rho_i x^{i-1} \tag{14}$$

where

$$\lambda_i = P(\text{edge is incident on variable node of degree } i)$$
$$\rho_i = P(\text{edge is incident on check node of degree } i)$$

Construction of irregular LDPC codes given the polynomials $\lambda(x)$ and $\rho(x)$ is done as follows: find the number of variable nodes of degree j as $n\lambda_j$ and the number of check nodes of degree j as $(n-k)\rho_j$, randomly distribute the variable and check nodes so that the degree constraints are satisfied, then create a permutation Tanner network connecting variable nodes to check nodes. Discard obviously bad choices (such as those with redundant parity checks), and choose the best performing code by simulation.

This project does not follow all these steps to generate LDPC codes given the limited time allowed for the project and the extensive time taken for a simulation run. This project employs a much less robust method to generate a LDPC code (details are listed in section 5).

3.2 Iterative Decoding and Density Evolution

Gallager's algorithm Gallager proposed a simple bit-flipping algorithm that can be thought of as a precursor to the more powerful belief propagation algorithm.

First, all check nodes pass messages to variable nodes as

$$r_{ij} = \sum_{\hat{j} \in \{C_i \setminus j\}} x_{\hat{j}} \tag{15}$$

where C_i represents the set of all neighbours of check node i

In the next step, variable nodes pass messages to check nodes as

$$q_{ji} = \begin{cases} \bar{x_j} & \text{if } \sum_{\hat{i} \in \{V_j \setminus i\}} \delta(r_{\hat{i}j}, \bar{x_j}) > T \\ x_j & \text{otherwise} \end{cases} \tag{16}$$

where V_j represents the set of all neighbours of variable node j, and T is a threshold decided based upon the probability of a bit being flipped in the binary symmetric channel.

The next step involves passing messages from check nodes to variable nodes

$$r_{ij} = \sum_{\hat{j} \in \{C_i \setminus j\}} q_{\hat{j}i} \tag{17}$$

Steps in equations 16 and 17 are repeated until all parity checks are satisfied. Though a crude decoding algorithm, the message passing steps are very similar to the modern belief propagation algorithm discussed hereafter.

Belief propagation and soft decoding If we have access to probabilistic estimates of demodulated bits, we can improve the performance of the decoding algorithm. This method is termed *soft* decoding. The belief propagation based algorithm also operates on the Tanner graph but works with log-likelihood ratios instead of hard demodulated bits.

Let x_i be the noisy input received when bit b_j is transmitted. The log-likelihood ratio is defined as

$$\mathrm{LLR}(x_j) = \log \frac{l(x_j|b_j = 0)}{l(x_j|b_j = 1)} = \log \frac{P(b_j = 0|x_j)}{P(b_j = 1|x_j)} \tag{18}$$

Here, we assume that 0 and 1 are equally likely a-priori. If this were not true, then the entropy of the source would not be 1, and Shannon's limit would not hold.

The message received from the demodulator is the log-likelihood ratio for each demodulated bit. Any message on an edge incident on a variable node specifies the log-likelihood ratio for that variable/bit. Denoting the message received from the channel by c_j, and the same convention for messages as in Gallager's algorithm (equations 15 and 17), we have the following update at the variable node

$$q_{ji} = c_j + \sum_{\hat{i} \in \{V_j \setminus i\}} r_{\hat{i}j} \tag{19}$$

Which is to say that the likelihood ratio for a certain node is the product of likelihood ratios given the evidence from the channel and from the other check nodes.

The update at the check nodes is more complex. In order to generalise the form of the update function, it is more convenient to work in terms of $P(b_j = 0) - P(b_j = 1)$.

Consider

$$m = \log \frac{P(b = 0)}{P(b = 1)} \tag{20}$$

We have

$$P(b = 0) - P(b = 1) = \frac{e^m - 1}{e^m + 1} = \tanh\left(\frac{m}{2}\right) \tag{21}$$

Consider a single check node, with incoming messages represented by m_i. This node receives messages from all variable nodes and must send a message to variable node j with a log-likelihood ratio given evidence from other variable nodes. As before, consider $P(b_j = 0) - P(b_j = 1)$.

Consider

$$\prod_{i \in \{\mathscr{N} \setminus j\}} \tanh\left(\frac{m_i}{2}\right) = \prod_{i \in \{\mathscr{N} \setminus j\}} \left(P(b_i = 0) - P(b_i = 1)\right) \tag{22}$$

$$= P\left(\sum_{i \in \{\mathscr{N} \setminus j\}} b_i = 0\right) - P\left(\sum_{i \in \{\mathscr{N} \setminus j\}} b_i = 1\right) \tag{23}$$

$$= P(b_j = 0) - P(b_j = 1) \tag{24}$$

So, we have

$$r_{ij} = 2 \tanh^{-1}\left(\prod_{\hat{j} \in \{C_i \setminus j\}} \tanh\left(\frac{q_{\hat{j}i}}{2}\right)\right) \tag{25}$$

4 Improvements To The Decoding Algorithm

This project experiments with two possible improvements to the decoding algorithm.

Min-sum decoding Most applications employing LDPC decoding require decoding to be done on a small embedded device with limited resources, and where power consumption may be an issue. Such devices may not be able to support the complicated floating point arithmetic required for computation of the hyperbolic tangent or its inverse. As a result, some FPGAs use lookup tables for computing the hyperbolic tangent[19]. However, a simpler approximation, often termed as *min-sum* decoding works just as well. Tanner's original paper[18] implemented a version of min-sum decoding instead of the more complex decoding scheme. Some effort has since been put in finding LDPC codes that perform well under min-sum decoding(eg: [4, 17]).

Min-sum decoding essentially changes the update function in Equation 25 to the following

$$r_{ij} = \min_{\hat{j} \in \{C_i \setminus j\}} \left| q_{\hat{j}i} \right| \prod_{\hat{j} \in \{C_i \setminus j\}} \operatorname{sign}\left(q_{\hat{j}i} \right) \tag{26}$$

Residual belief propagation Elidan et al[7] introduced residual message passing as a method to schedule messages in asynchronous belief propagation so that "a fixed point is reached faster and more often." The intuition behind residual belief propagation is that messages which have changed significantly in the past are more likely to change in the future. The algorithm schedules messages prioritising them by *residuals*, a measure of how much a message has changed in the past.

One of the main issues with residual message passing is that a single lucky update can push a message residual down to zero, which can prevent the message from ever being updated again. To counter this effect, residuals are damped in practice, so that a single update does not prevent an edge from being visited ever again.

Gonzalez et al[10] extended the residual based belief propagation to an approach they termed "Residual Splash", with two major changes

1. Residuals are computed at nodes rather than at edges, because small changes at multiple edges can result in a big change for any outgoing message from a node.
2. Updates are performed not just on the node with the largest residual, but on a neighbourhood around the node, by constructing a breadth-first tree, and performing message passing from leaves to root, and from root to leaves. They provide a mathematical model to choose the optimal depth of the tree.

This approach however requires a priority queue with random deletions, in other words, since an update affects the priority of a number of nodes, the nodes must be resorted to maintain the priority order. The large number of calls to resort the queue adds additional overhead to the algorithm and makes it infeasible to implement.

Residual belief propagation for LDPC decoding has been evaluated by Casado et al[3] and Gong et al[9]. However, these methods perform approximate node-wise scheduling, whereas this project analyses the residual belief propagation algorithm in its original flavour, performing edge-wise scheduling.

5 Experimental Setup

This project models a baseband communication system over an additive white Gaussian noise (AWGN) channel,

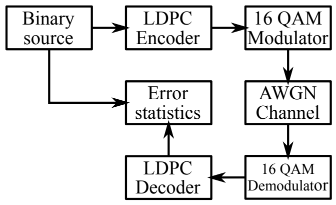

Figure 2: Block diagram of the communication system simulated in this project

simulated using MATLAB®. A block diagram for the system is shown in Figure 2.

The signal to noise ratio (SNR) in the AWGN channel is varied to ± 5 dB around the Shannon SNR, and the performance, in terms of bit error rate (BER), for the LDPC code is recorded. The number of iterations required until convergence is reached is also recorded for each block that is decoded, and cumulative statistics are reported for each SNR level.

LDPC code generation The LDPC code is generated in a fashion unlike the one mentioned in section 3.1. As we are concerned only with systematic codes (and the subset where both the generator matrix and the parity check matrix are in systematic form), a random sparse matrix $\mathbf{P}$ is generated, and concatenated with the identity matrix to generate the parity check matrix $H = \begin{bmatrix} \mathbf{P} & \mathbf{I} \end{bmatrix}$. The matrix is checked to ensure that there is a one in every column, if not, ones are added to the column where they are lacking.

This method is not as robust to generating *good* codes as the method mentioned in section 3.1, however, for the sake of this project, sub-optimal LDPC codes work better at displaying the difference in decoding performance of the different algorithms.

A 4×16 parity check matrix is generated, which corresponds to a $(16, 12)$ binary linear code. The parity check matrix generated for the purpose of this project is shown in Figure 3. From the figure, it is apparent that the last parity check checks just one bit, as a consequence forcing that bit to be zero. This error was not spotted until it was too late to change the parity check matrix and re-run simulations.

Given the code rate ($3/4$) and the 16 QAM modulation scheme used, the Shannon SNR is computed to be ~ 18 dB.

A `C++` class library is written for the LDPC decoder. A MATLAB Executable (MEX) wrapper is written around the `C++` class, which interfaces with the simulation envi-

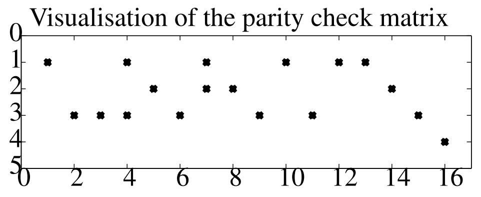

Figure 3: A visualisation of the parity check matrix used in this project

ronment via a MATLAB class. The decoder has two options, one to decide between sum-product and min-sum decoding, and the other to decide between round-robin and residual based decoding. A total of four runs (one for each possible combination of options), each simulating 10,000,000 code blocks is executed at each SNR level. MATLAB's built-in decoder is used as a baseline. While the baseline decoder is set to run for 50 iterations, the test decoders iterate for 50 iterations, or until all parity checks are satisfied, whichever deadline is reached sooner.

6 Results

The four different decoding schemes along with MATLAB®'s built-in LDPC decoder are tested, and bit error rates are reported in Figure 4. We observe that the sum-product round-robin scheme does not perform as well as the baseline, however, the trends are consistent. It is possible that this discrepancy is due to a different order in which nodes are processed by the baseline, or that the baseline performed message passing for a whole 50 iterations without terminating if parity checks were satisfied, whereas the implementations in this report terminated as soon as a parity check was satisfied. As the baseline is closed-source, it is difficult to reach any conclusion.

We observe that for this LDPC code, the min-sum schemes perform slightly better than the sum-product schemes. The result is not surprising, it means that this choice of code is better suited for min-sum decoding than for sum-product decoding. Some effort has been focussed on finding codes that are more amenable to min-sum decoding than sum-product decoding[17].

Another observation is that the residual based algorithms perform better than their round-robin counterparts. While this certainly aligns with the intuition behind the residual algorithm, any confident report will require comparison over a number of codes. Such a comparison will take additional time beyond the time allotted for this project.

A comparison of the average number of iterations required for decoding is shown in Figure 5. The baseline decoder was fixed to 50 iterations per block.

We observe that the min-sum residual based approach takes less iterations than any other decoding scheme for this code. The sum-product based approach is relatively unaffected by the residual implementation. The min-sum approach with the round robin scheme takes less iterations to decode than the sum-product based approach for low SNR, however, crossing the Shannon SNR (18 dB), the min-sum approach with the round-robin ordering ends up requiring more iterations than the sum-product decoders.

7 Discussion And Future Work

This project attempted to compare four decoding schemes for LDPC codes, making use of two different optimisations, proposed by two different communities. The application of residual belief propagation was found to reduce the computational complexity and provide better results satisfying the intuition behind the residual based approach. At the same time, the min-sum decoding scheme offers a low-complexity alternative to the complicated sum-product decoding scheme, and codes can be discovered that perform better under min-sum decoding than sum-product decoding.

Future work will involve implementing these decoding schemes on field programmable gate arrays (FPGAs), and evaluating performance for codes with long block lengths as used in practice, like the codes used in the DVB-S2 standard with block lengths of 64,800[1].

Additional experiments are required to conclusively compare the decoding performance of the residual based approach as contrasted with a fixed-order approach (round-robin).

At the moment, the parameters used in the residual based approach for damping the changes in the residual values are chosen ad-hoc. Further work is needed to choose the best value for the damping constant, as well as for experimenting with different configurations of the residual algorithm (eg: residuals by nodes, two different sets of residuals for the two types of nodes, etc.).

References

[1] Digital video broadcasting (DVB) user guidelines for the second generation system for broadcasting, interactive services, news gathering and other broadband satellite applications (DVB-S2). Technical Report 102 376, ETSI, February 2005.

[2] Unified high-speed wireline-based home networking transceivers — system architecture and physical layer specification. Technical Report G.9960, ITU-T: Telecommunications standardization sector of ITU, December 2011.

[3] A. I. V. Casado, M. Griot, and R. D. Wesel. Improving LDPC Decoders via Informed Dynamic Scheduling. In *2007 IEEE Information Theory Workshop*, pages 208–213. IEEE, Sept. 2007.

[4] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu. Reduced-Complexity Decoding of LDPC Codes. *IEEE Transactions on Communications*, 53(8):1288–1299, Aug. 2005.

[5] J. Chen and M. Fossorier. Density evolution for two improved BP-Based decoding algorithms of LDPC codes. *IEEE Communications Letters*, 6(5):208–210, May 2002.

[6] E. Eleftheriou, D.-M. Arnold, and A. Dholakia. Efficient implementations of the sum-product algorithm for decoding LDPC codes. In *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No.01CH37270)*, volume 2, pages 1036–1036E. IEEE, 2001.

[7] G. Elidan, I. McGraw, and D. Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Proceedings of the Twenty-second Conference on Uncertainty in AI*, 2006.

[8] R. Gallager. Low-density parity-check codes. *IEEE Transactions on Information Theory*, 8(1):21–28, Jan. 1962.

[9] Y. Gong, X. Liu, W. Ye, and G. Han. Effective Informed Dynamic Scheduling for Belief Propagation Decoding of LDPC Codes. *IEEE Transactions on Communications*, 59(10):2683–2691, Oct. 2011.

[10] J. Gonzalez, Y. Low, and C. Guestrin. Residual splash for optimally parallelizing belief propagation. 5, 2009.

[11] IEEE and IEEE-SA standards board. *IEEE standard for information technology — telecommunications and information exchange between systems — local and metropolitan area networks — specific requirements Part 11: wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 5: enhancements for higher throughput*, October 2009.

[12] S. Lin and D. J. Costello, Jr. *Error Control Coding: Fundamentals and Applications*, pages 278–280. Prentice Hall, 1st edition, 1983.

[13] D. MacKay. Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 45:399–431, 1999.

[14] D. MacKay and R. Neal. Near Shannon limit performance of low density parity check codes, 1997.

[15] U. Madhow. *Fundamentals of Digital Communication*, pages 272–280. Cambridge University Press, 1st edition, 2008.

[16] U. Madhow. *Fundamentals of Digital Communication*, pages 342–357. Cambridge University Press, 1st edition, 2008.

[17] B. Smith, F. Kschischang, and W. Yu. Low-Density Parity-Check Codes for Discretized Min-Sum Decoding. In *23rd Biennial Symposium on Communications, 2006*, pages 14–17. IEEE, 2006.

[18] R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, Sept. 1981.

[19] M. Tommiska. Efficient digital implementation of the sigmoid function for reprogrammable logic. *IEE Proceedings - Computers and Digital Techniques*, 150(6):403, 2003.

[20] M. Yang, W. Ryan, and Y. Li. Design of Efficiently Encodable Moderate-Length High-Rate Irregular LDPC Codes. *IEEE Transactions on Communications*, 52(4):564–571, Apr. 2004.

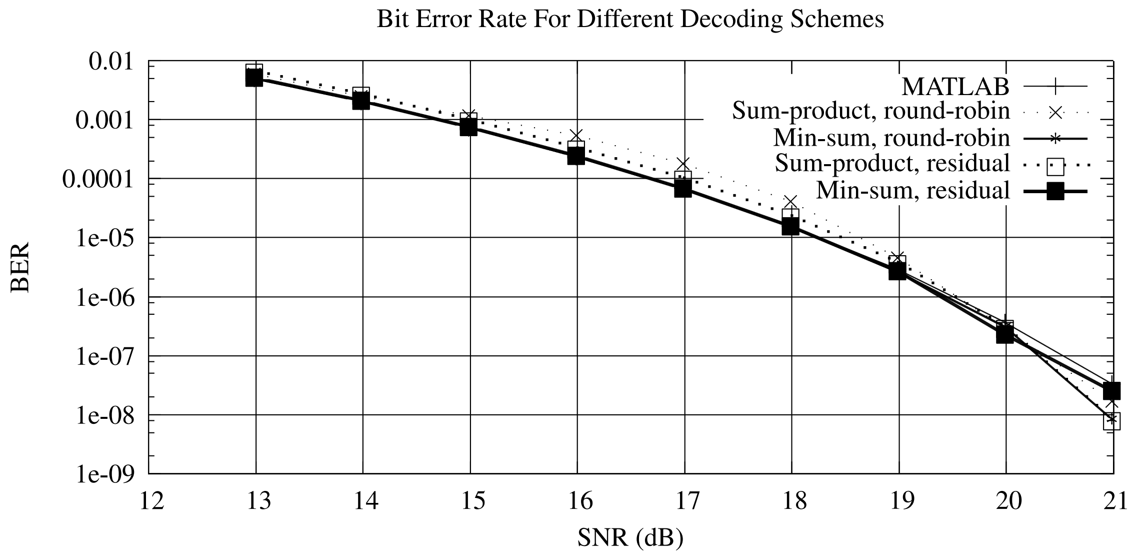

Figure 4: Bit error rates for different decoding schemes

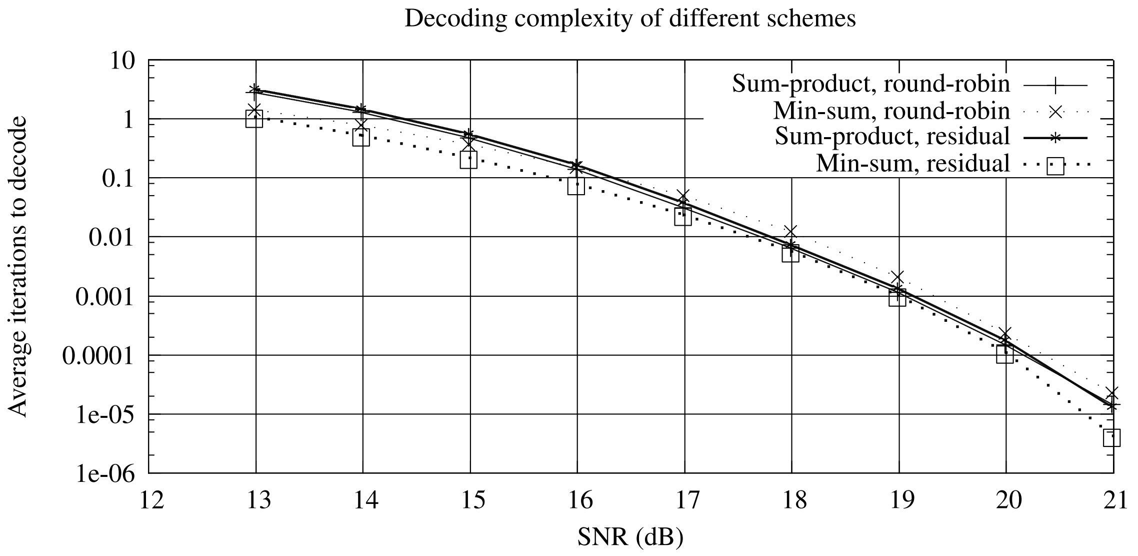

Figure 5: Number of iterations taken by different decoding schemes