Performance and Decoder Complexity Estimates for Families of Low-Density Parity-Check Codes

S. Dolinar[1] and K. Andrews[1]

We present methods to estimate code performance and decoder complexity from the code rate, block size, and word-error rate, for families of related low-density parity-check (LDPC) codes. Performance estimates are generally within a couple tenths of a decibel of results determined by simulation; estimates of complexity (and hence decoder speed) are generally within 10 percent. Experimental data show that there is a trade-off between complexity and code performance determined by the design of the LDPC code, and that each 1 dB (26 percent) of increased complexity is worth about a 0.1-dB reduction in the required signal-to-noise ratio.

I. Introduction

Since the rediscovery of low-density parity-check (LDPC) codes in the mid 1990s, most of the code design research has been directed towards minimizing the signal-to-noise ratio (SNR) required for successful decoding. More recently, some code design work has also been done to reduce the computational burden on the decoder. In this article, we examine the required bit SNR E_b/N_0 and a measure of complexity that counts the total number of messages per decoded bit computed by the iterative decoder. From both metrics, we distill the portion attributable to the code design from the effects of block size, code rate, and word-error rate (WER). Through examples, we quantify the trade-off between performance and complexity that can be achieved through code design.

Several LDPC code families are used in this article, all of which are "protograph" constructions [1,2]. The protograph construction method begins by designing a small bipartite graph with n_p variable nodes, perhaps u_p additional untransmitted or punctured variable nodes, and m_p constraint nodes. Figures 1 through 4 show several protograph families, where filled circles represent transmitted variable nodes, open circles are punctured variable nodes, and squares are check nodes. To construct a full LDPC code, each node of the protograph is replicated T times, and each edge is replaced by a bundle of T edges. Finally, each bundle is severed and reconnected in some permuted way (such as with a cyclic shift), thus interconnecting the copies of the protograph. The result is a code of length $n = Tn_p$, dimension k at least

[1] Communications Architectures and Research Section.

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

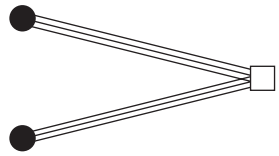

Fig. 1. The G36 protograph.

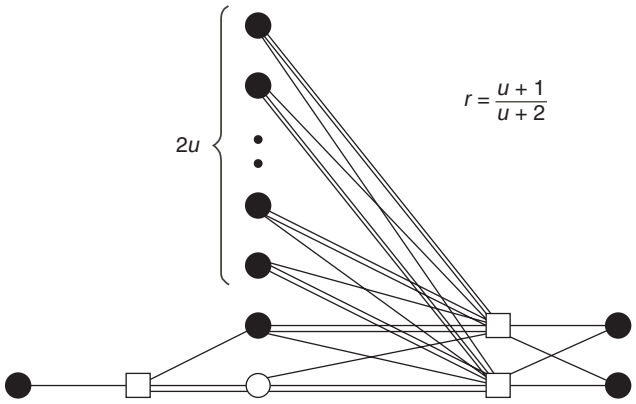

Fig. 2. The AR3A protograph family.

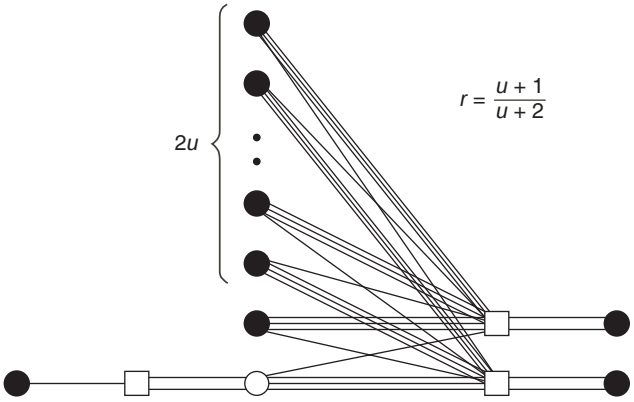

Fig. 3. The AR4A protograph family.

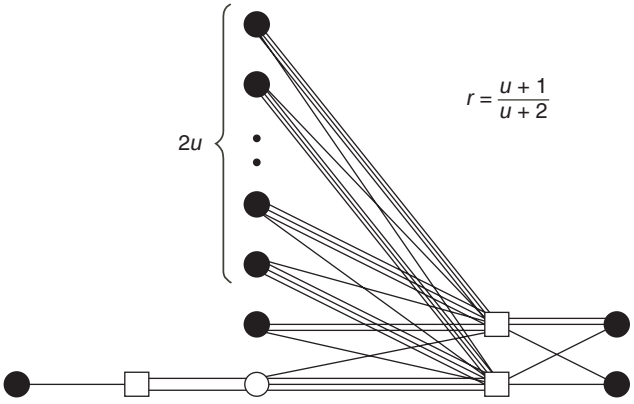

Fig. 4. The AR4JA protograph family.

$T(n_p + u_p - m_p)$, and rate k/n at least $(n_p + u_p - m_p)/n_p$. Note that the nominal rate is characteristic of the protograph and does not depend on the expansion factor. Many researchers have used equivalent code construction methods, e.g., [3–5].

A family of codes is constructed from a collection of related protographs of different rates. Each protograph is expanded with permutations of different lengths to yield related codes of different dimensions. Eight families of codes are considered in this article, as listed in Table 1, all of which are protograph constructions. The G36 family are regular (3,6) LDPC codes of rate 1/2 and different sizes, built from the protograph shown in Fig. 1, a subset of the randomly connected Gallager codes [6]. The AR3A, AR4A, and AR4JA code families (named for an accumulate–repeat–accumulate construction from which these were derived [7]) are built to various sizes and code rates r from the nested protographs shown in Figs. 2 through 4. The four C_i code families have protographs shown in [8]. The initial C_i protographs are constructed by expurgation. The C_i^+ protographs are formed by lengthening them with accumulators, and the C_i^+[light] and C_i^+[med] protographs are variants with different amounts of pre-coding.

Table 1. Eight code families built from protographs.

Code family	Rates (r)	Information block sizes (k)
G36	$r = 1/2$	$k = 1024, 4050, 32400$
AR3A	$r = 1/2, 2/3, 4/5$	$k = 1024, 4096, 16384$
AR4A	$r = 1/2, 2/3, 4/5$	$k = 1024, 4096, 16384$
AR4JA	$r = 1/2, 2/3, 4/5$	$k = 1024, 4096, 16384$
$\{C_i, i = 8, 4, 2, 1\}$	$r = 1/2, 3/4, 7/8, 15/16$	$n = 8176$
$\{C_i^+, i = 16, 8, 4, 2\}$	$r = 1/2, 2/3, 4/5, 8/9$	$k = 8176$
$\{C_i^+[\text{light}], i = 16, 8, 4\}$	$r = 1/2, 2/3, 4/5$	$k = 8176$
$\{C_i^+[\text{med}], i = 8\}$	$r = 2/3$	$k = 8176$

II. Code Performance

Figure 5 shows performance curves on the additive white Gaussian noise (AWGN) channel for the 42 codes in Table 1, varying in code rate from 1/2 to 7/8, and in block size from 1024 to 16384; the legend is shown in Fig. 6. It is difficult to distinguish the best code designs from average ones in this manner, because most of the 4-dB span in required E_b/N_0 is due to variations in block size, code rate, and WER, not to the quality of the code design.

Empirical evidence shows that codes of a given rate and dimension constructed from the same protograph, but differing in the choices of permutations, perform similarly outside their "error floor" regions. Let $S^{\mathcal{F}}(r, k, P_w)$ be the value of E_b/N_0 (measured in decibels) required to achieve WER P_w with a representative LDPC code of rate r and dimension k, selected from code family $\mathcal{F}$.

The impact of a well-designed code family $\mathcal{F}$ can be distinguished from the effects of r, k, and P_w by comparison to Shannon's sphere-packing lower bound [9]. The sphere-packing bound $S^{*CI}(r, k, P_w)$ for the continuous-input AWGN (CI-AWGN) channel gives (a lower bound on) the minimum SNR (in decibels) required to achieve a word-error rate of P_w by any code of rate r and dimension k. The difference between $S^{\mathcal{F}}(r, k, P_w)$ and $S^{*CI}(r, k, P_w)$ is the "imperfectness" of the code [10].

In the limit of infinite block size, and then in the limit of infinitesimal error rate, the sphere-packing bound reduces to the rate-constrained capacity limit $S^{*CI}(r)$ for the CI-AWGN channel,

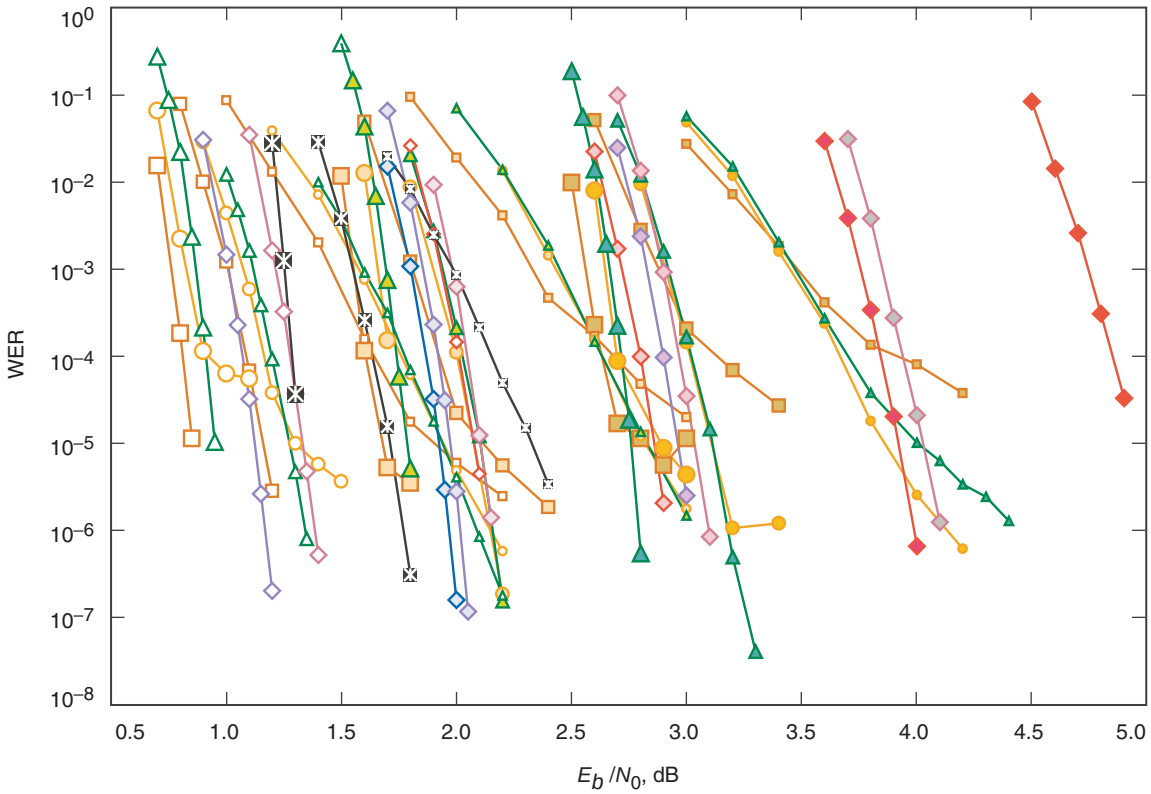

Fig. 5. Decoding performance curves for the 42 LDPC codes in Table 1.

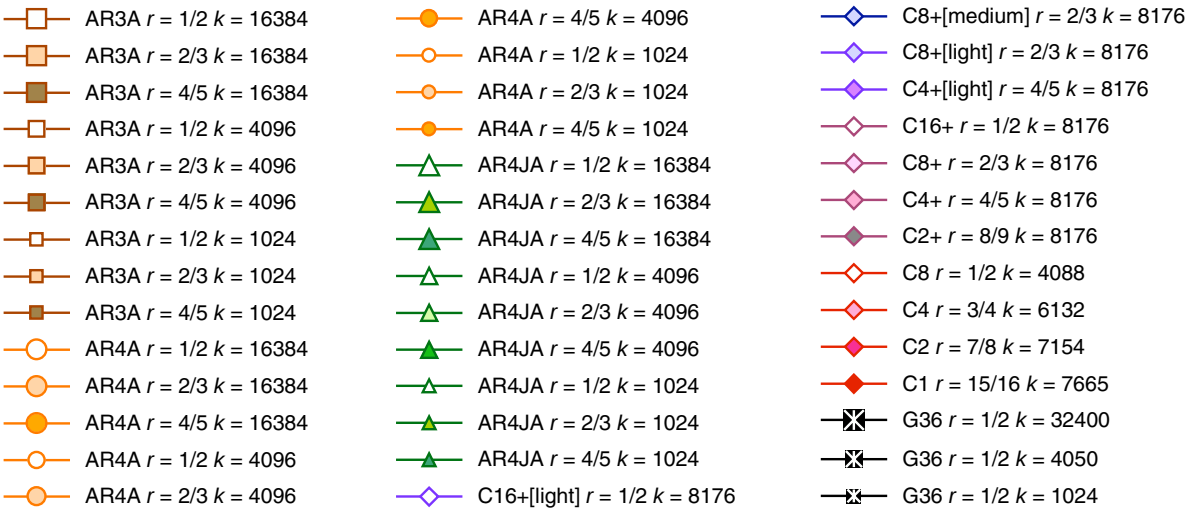

Fig. 6. Legend for Figs. 5, 8, 10, 11, and 12.

$$S^{*CI}(r) \triangleq \lim_{P_w \to 0} \lim_{k \to \infty} S^{*CI}(r, k, P_w) = 10 \log_{10} \left(\frac{2^{2r} - 1}{2r} \right)$$

The difference between these bounds is the *finite-size penalty* computed for the CI-AWGN channel,

$$\Delta S^{*CI}(r, k, P_w) \triangleq S^{*CI}(r, k, P_w) - S^{*CI}(r)$$

This finite-size penalty is plotted in Fig. 7, as a function of k and of $\sqrt{1024/k}$ for code rates $r = 1/2, 4/5,$ and $16/17$, and for word-error rates $P_w = 10^{-2}, 10^{-4}, 10^{-6}$, and 10^{-8}. Curves of different rates virtually coincide, so the finite-size penalty is nearly independent of the code parameter that determines the capacity limit. To a good approximation, the effects of the code's finiteness are separated from those of the constraints imposed on the channel.

The curves in Fig. 7 are nearly linear in $\sqrt{1024/k}$; this asymptotic dependence on $k^{-1/2}$ was observed in [10]. At $P_w = 10^{-8}$, the sphere-packing bound imposes about 1/3-dB penalty for each quarter-unit step of $\sqrt{1024/k}$, or each step in the sequence $k = \infty, 16384, 4096, 1820, 1024$. At $P_w = 10^{-4}$, the penalty is about $1/\sqrt{2}$ as large, and at $P_w = 10^{-2}$, the penalty is about $1/2$ as large. Thus, for the range of k and P_w plotted in Fig. 7, the finite-size penalty $\Delta S^{*CI}(r, k, P_w)$ can be approximated as

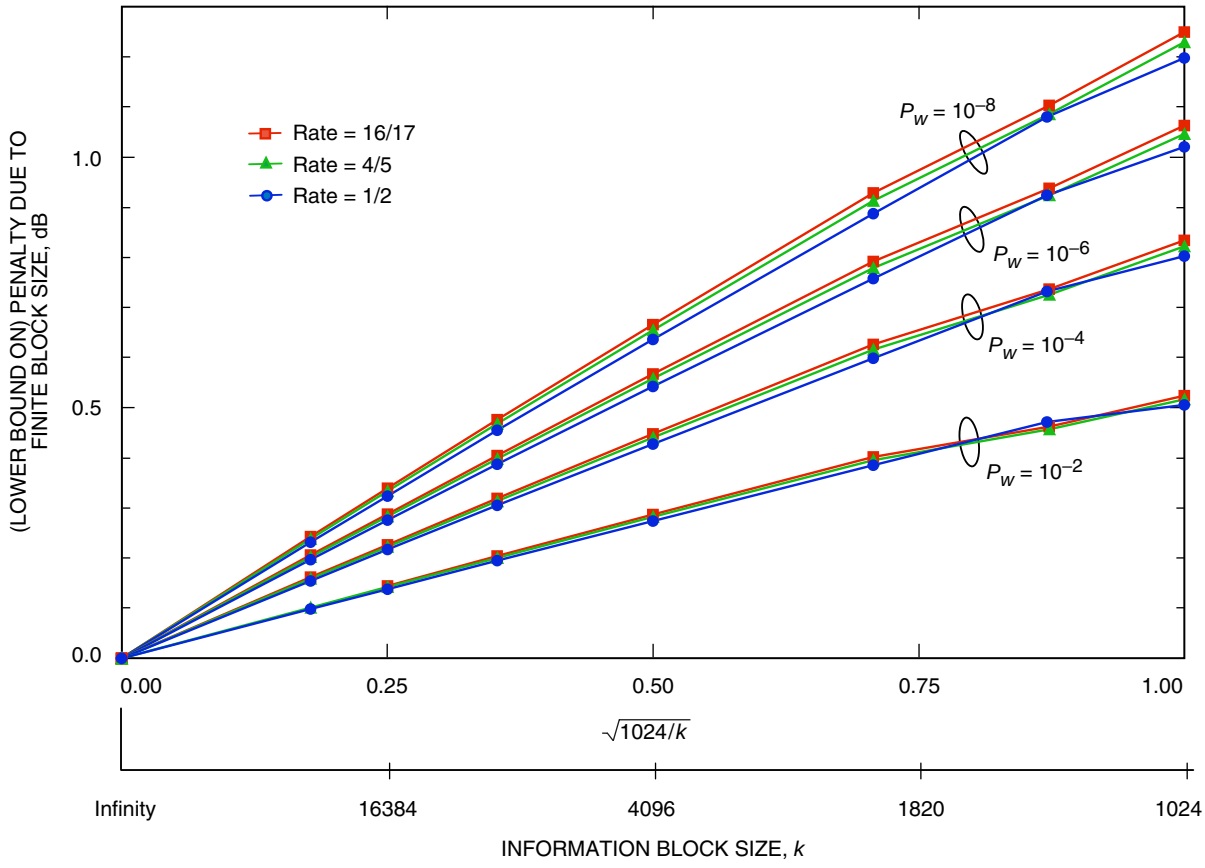

Fig. 7. Finite-size penalty $\Delta S^{*CI}(r,k,P_W)$ for rate r, block size k, and WER P_W, obtained from the CI-AWGN sphere-packing bound relative to the corresponding CI-AWGN capacity limit.

$$\Delta S^{*CI}(r,k,P_w) \approx \frac{4}{3}\sqrt{\frac{1024}{k}}\sqrt{\frac{-\log_{10} P_w}{8}} \quad \text{for} \quad k \gtrapprox 1024,\ 10^{-8} \lessapprox P_w \lessapprox 10^{-2}$$

The capacity threshold $S^{*BI}(r)$ for the binary-input AWGN (BI-AWGN) channel is close to the threshold $S^{*CI}(r)$ of the CI-AWGN channel for rates up to 1/2; for higher rates, a substantial gap appears. Thus, a high-rate binary LDPC code cannot be expected to perform close to limits determined for the CI-AWGN channel. Instead it should be compared to a similar benchmark $S^{*BI}(r,k,P_w)$ which takes into account both the finite-size constraints and the channel constraints. Given the approximate separability of these two types of constraints, we approximate $S^{*BI}(r,k,P_w)$ with the BI-AWGN capacity threshold, corrected by the finite-size penalty computed for the CI-AWGN channel:

$$S^{*BI}(r,k,P_w) \approx S^{*BI}(r) + \Delta S^{*CI}(r,k,P_w)$$

The exact BI-AWGN channel capacity limit $S^{*BI}(r)$ is given implicitly by [11]

$$r = 1 - \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\rho}} e^{-(x-\rho)^2/(2\rho)} \log_2\left(1+e^{-2x}\right) dx$$

where $\rho = 2E_s/N_0 = 2r \times 10^{S^{*BI}(r)/10}$. For $1/2 \lessapprox r \lessapprox 16/17$, a range of rates particularly suitable for LDPC codes, a useful simple approximation for the BI-AWGN capacity threshold is

$$S^{*BI}(r) \approx \log_2 \frac{r}{1-r} \quad \text{for} \quad 1/2 \lessapprox r \lessapprox 16/17 \tag{1}$$

Next we define the *size-constrained non-optimality* $\Delta S_{\mathcal{F}}$ as the difference between the actual code performance $S^{\mathcal{F}}(r,k,P_w)$ and the estimated performance $S^{*BI}(r,k,P_w)$ of an optimal code subjected to the same constraints:

$$\Delta S_{\mathcal{F}} \triangleq S^{\mathcal{F}}(r,k,P_w) - S^{*BI}(r,k,P_w)$$

The size-constrained non-optimality is plotted versus P_w in Fig. 8 for each of the codes in Table 1 with block size $k = 4096$ and higher. Experiments show that the size-constrained non-optimality is nearly independent of k, r, and values of P_w above the code's error floor, so we attribute it to the design of the LDPC code family $\mathcal{F}$.

For protograph LDPC codes, the size-constrained non-optimality $\Delta S_{\mathcal{F}}$ can be divided into two components. First, density evolution can be applied to the protograph to determine its asymptotic iterative decoding threshold $S^{\mathcal{P}}(r)$, which is a capacity-like limit in that it defines a minimum SNR required for achieving arbitrarily small error rates in the limit as the protograph is expanded (sufficiently randomly) to build an arbitrarily large code. The difference between the protograph's iterative decoding threshold and the corresponding capacity limit is the *protograph non-optimality*

$$\Delta S_{\mathcal{P}} \triangleq S^{\mathcal{P}}(r) - S^{*BI}(r)$$

On top of the protograph non-optimality is an additional *expansion non-optimality*

$$\Delta S_{\mathcal{X}} \triangleq \Delta S_{\mathcal{F}} - \Delta S_{\mathcal{P}} \approx S^{\mathcal{F}}(r,k,P_w) - \left[S^{\mathcal{P}}(r) + \Delta S^{*CI}(r,k,P_w)\right]$$

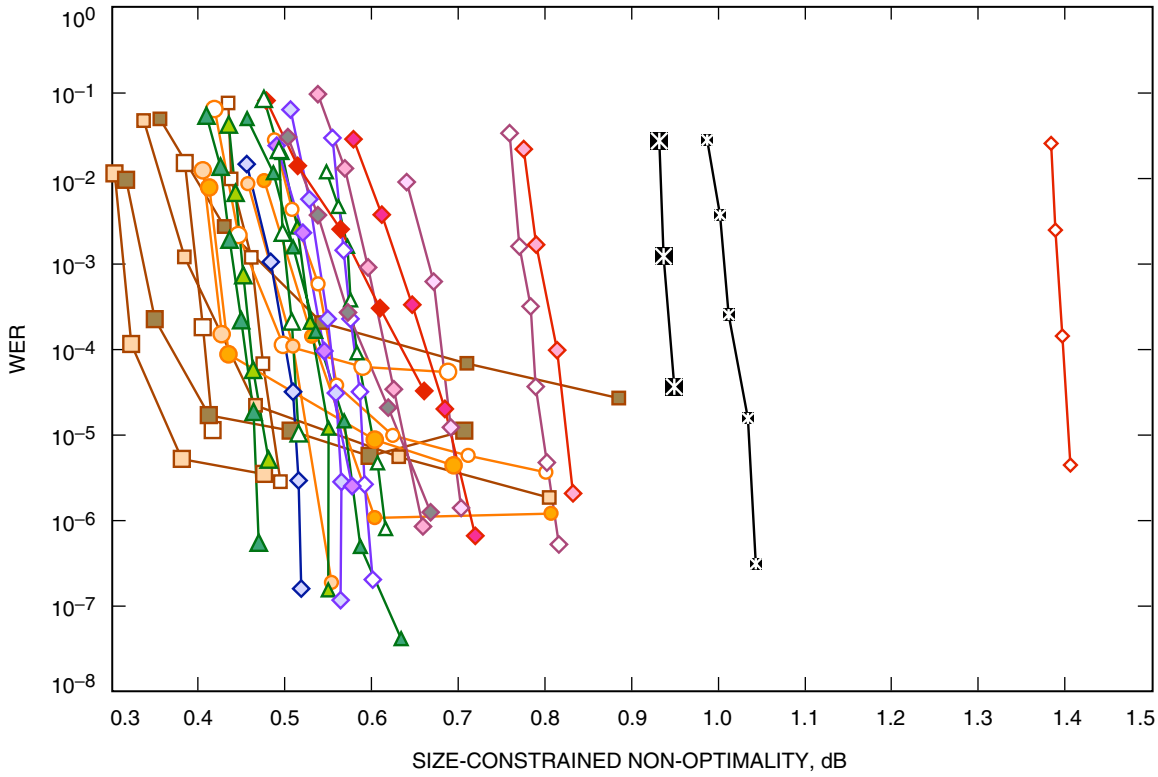

Fig. 8. Normalized performance comparison of the codes in Table 1 with information block size 4096 bits and higher, measured in terms of their size-constrained non-optimalities $\Delta S_{\mathcal{F}}$.

The reference point for computing the expansion non-optimality is approximated by the sum in square brackets: the protograph's iterative decoding threshold, corrected by the finite-size penalty computed for the CI-AWGN channel. Since this approximation is not an exact bound, the expansion non-optimality $\Delta S_{\mathcal{X}}$ computed from it can conceivably be negative, but we have never observed this for any of the codes that we have examined.

The protograph non-optimality is due to constraints imposed by the design of the protograph. The expansion non-optimality is caused by all finite-size effects, including the choice of permutations used to expand the protograph and the possible introduction of an error floor at error rates P_w of interest. The protograph non-optimality, plotted in Fig. 9 for the first seven code families in Table 1, can be held to less than 0.5 dB with careful protograph designs that allow efficient expansions to reasonable-size codes. With such protographs, we have been able to limit the additional expansion non-optimality to only about 0.05 to 0.2 dB if we apply optimized expansion techniques to build the full protograph code and avoid the introduction of an error floor at the error rates of interest. While it is possible to push the protograph non-optimality even closer (perhaps arbitrarily close) to 0 dB, we have found that highly optimized protographs are too complex and produce intolerably large expansion non-optimality when these protographs are expanded to codes of a few thousands or tens of thousands of bits. Thus, we generally look for protographs optimized under the constraint that they are also simple and small.

III. Decoder Complexity

An LDPC belief propagation decoder is usually designed to update each edge message according to a variable node rule, and then again by a check node rule, and to iterate this process until either a codeword is found or some maximum number of iterations is reached. This decoder computes an average

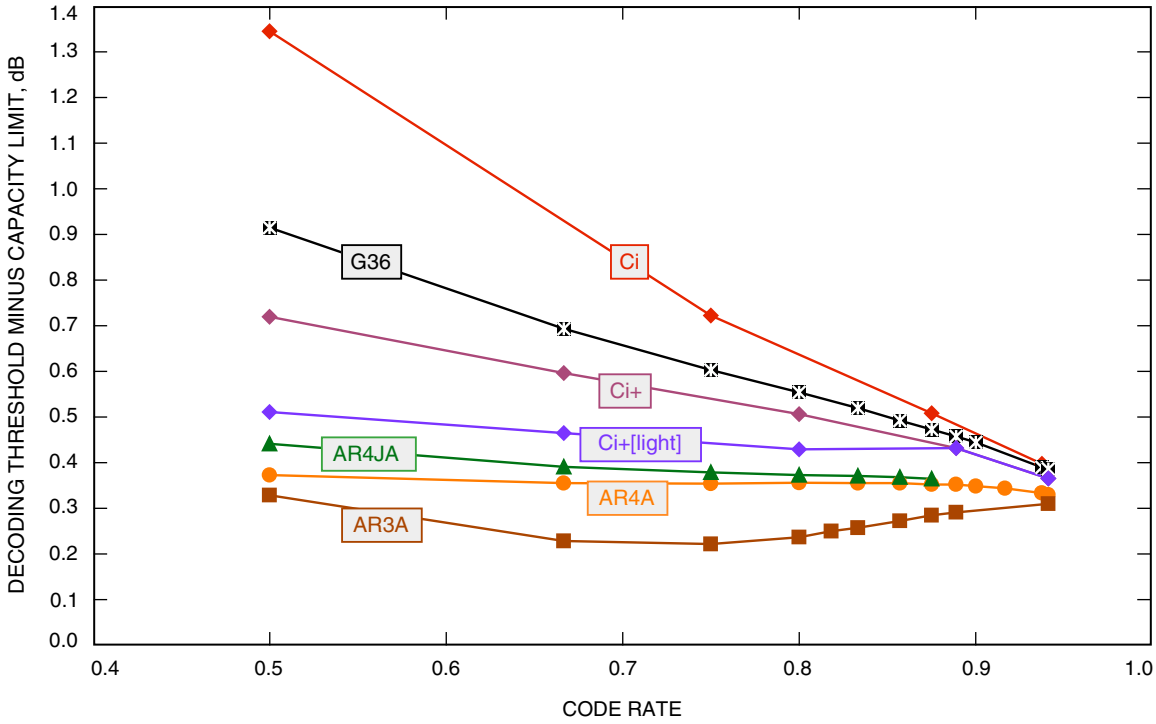

Fig. 9. Protograph non-optimalities $\Delta S_{\mathcal{P}}$ for seven of the protograph families in Table 1.

of $2IE/rn$ edge messages per decoded bit, where E is the number of edges in the LDPC graph and I is the average number of iterations performed. While different decoder implementations operate in very different ways, we have found that this is an appropriate complexity metric for both software and hardware decoders.

Figure 10 shows decoder complexity for the 42 codes in Table 1. The horizontal axis is logarithmic, showing $M = 10\log_{10}(2IE/rn)$, the complexity measured in decibels. For a given LDPC code, the complexity is lower when the decoder is operating at lower error rates, because this implies a higher SNR and fewer average iterations required. Small high-rate codes are easier to decode than large low-rate codes, and decoder complexity spans more than a factor of 10 among the cases shown.

Lacking theoretical bounds on complexity, we nonetheless attempt to normalize the raw complexity numbers with respect to a rate and size dependence of complexity determined by empirically fitting a model to experimental data. The data show that to a good approximation the decoder complexity depends on the code rate and on E_b/N_0 only in the combination $rE_b/N_0 = E_s/N_0$, the symbol SNR, or, equivalently, on $S_s \triangleq 10\log_{10} E_s/N_0$, the symbol SNR measured in decibels. Thus, we write the complexity (in decibels) in Fig. 10 as $M = M^{\mathcal{F}}(S_s, k, P_w) = 10\log_{10}(2IE/rn)$, a function of symbol SNR, code dimension, word-error probability, and the design of the family of LDPC codes.

When complexity is measured in decibels, its dependence on S_s is roughly linear, and its dependence on k is roughly logarithmic. Thus, we decompose $M^{\mathcal{F}}(S_s, k, P_w)$ into two components,

$$M^{\mathcal{F}}(S_s, k, P_w) \approx M^*(S_s, k) + \Delta M^{\mathcal{F}}(P_w)$$

where $M^*(S_s, k)$ is an estimate of the lowest complexity attained by any of the code families in Table 1 at low error rates, and $\Delta M^{\mathcal{F}}(P_w)$ measures the remaining component of complexity dependent

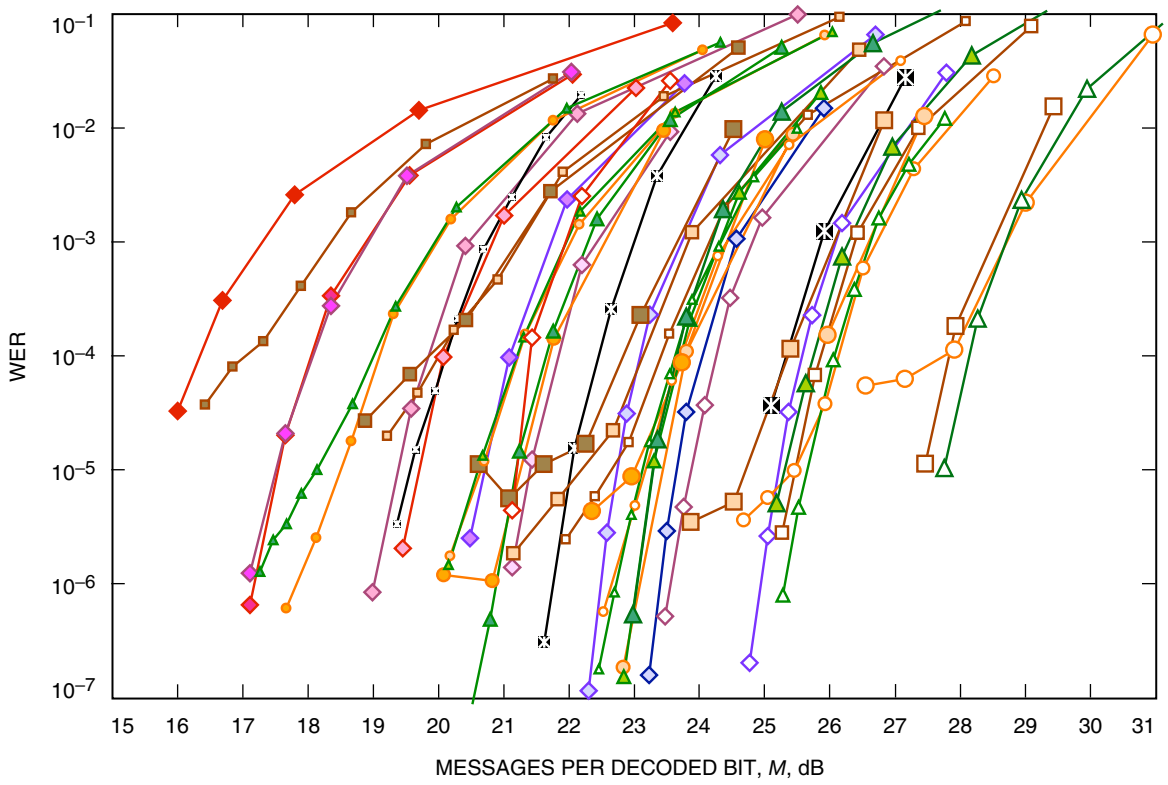

Fig. 10. Decoding complexity for the 42 LDPC codes in Table 1.

on error rate and code family. Curve-fitting the data in Fig. 10 leads to an approximation for $M^*(S_s, k)$:

$$M^*(S_s, k) \approx 10 - 1.2S_s + 0.84 \log_2(k)$$

The residual complexity component, $\Delta M^{\mathcal{F}}(P_w)$, is plotted in Fig. 11. Here it is noted that codes in the same family, but of different rates and block sizes, now cluster together. Each of these clusters has a similar shape, with different horizontal displacements. The shape shows how the decoder's complexity depends on P_w; this can be approximated by $\Delta M^*(P_w) \approx 5.6 P_w^{1/4}$. The remaining horizontal displacement, $\Delta M_{\mathcal{F}}$, is determined by the design of the LDPC code family, and varies between 0 and 4 dB for the cases shown. Collecting these results, the decoder complexity, measured in decibels, is approximated by

$$M^{\mathcal{F}}(S_s, k, P_w) \approx \left[M^*(S_s, k) + \Delta M^*(P_w)\right] + \Delta M_{\mathcal{F}}$$

The reference complexity formula in brackets, $M^*(S_s, k) + \Delta M^*(P_w)$, is not a theoretical lower bound, and it is easy for the family-dependent component of complexity, $\Delta M_{\mathcal{F}}$ (in decibels), to be negative for poorly performing codes. However, the empirical evidence suggests that it provides a useful benchmark on the complexity of capacity-approaching LDPC codes that achieve small size-constrained non-optimality $\Delta S_{\mathcal{F}}$.

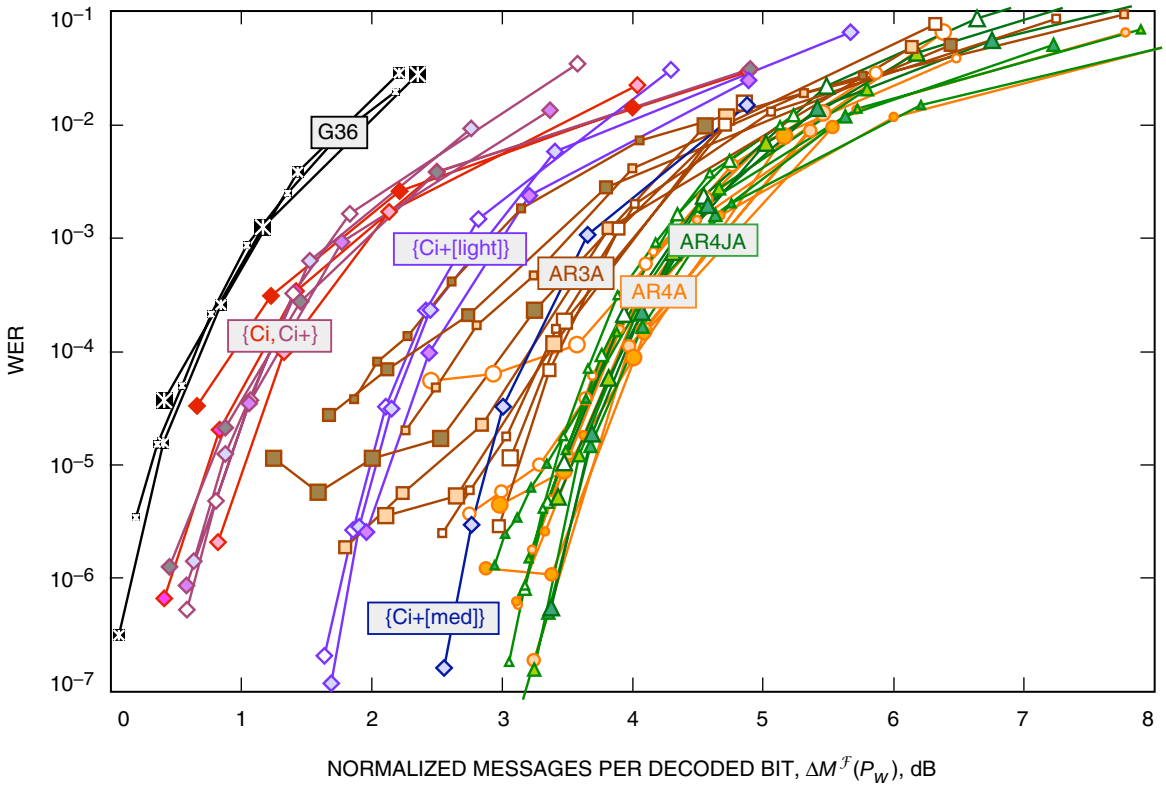

Fig. 11. Normalized decoding complexity $\Delta M^{\mathcal{F}}(P_W)$ for the 42 codes in Table 1, after adjusting for an empirically observed dependence $M^*(S_S, k)$ on symbol SNR and block size.

IV. Performance–Complexity Trade-off

In Sections II and III, code performance and decoder complexity were normalized by code rate, block size, and other factors, leaving the size-constrained non-optimality $\Delta S_{\mathcal{F}}$ and the family-dependent complexity component $\Delta M_{\mathcal{F}}$. These are plotted against each other in Fig. 12 for a WER of 10^{-4}.

On this plot LDPC codes within the same family cluster together, showing that their family resemblances run deeper than the simple appearances of their protographs. Furthermore, there are substantial performance–complexity trade-offs among the families. As is well-known, the iterative decoding threshold of the regular (3,6) Gallager construction is 0.9 dB from the corresponding capacity limit, but interestingly its decoding complexity is particularly low. Code families with irregular degree distributions and carefully designed protographs, such as AR3A, AR4A and AR4JA, can get within about 0.4 dB of $S^{*BI}(r, k, P_w)$, but at a cost in decoder complexity. This substantial cost, 4 dB or a factor of 2.5 times, is due to several factors, including a larger number of average iterations, an inefficiency due to the presence of punctured variable nodes, and a larger number of graph edges per code symbol.

Figure 12 suggests that there is a trade-off between required SNR and decoder complexity that is determined by the design of an LDPC code or its protograph family. At the boundary of the achievable region of these two quantities, they trade at a rate of about 1 dB of increased complexity for every 0.1 dB of reduced SNR.

It is well-known that code design (in combination with details of the decoder design and implementation) determines the location of an error floor. In the error-floor region, both performance and complexity deviate markedly from the estimates derived here. In this case, the size-constrained non-optimality can

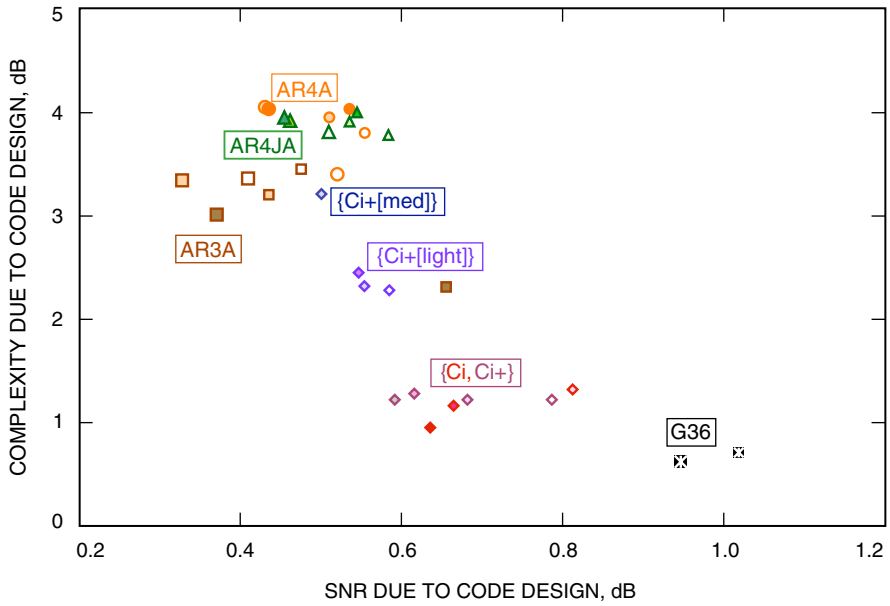

Fig. 12. Performance and complexity trade-offs, for the portions of each attributed to code design, measured for a WER of 10^{-4}.

easily go off the horizontal scale of Fig. 12 to the right, and conversely the complexity will be abnormally low because the increase in required SNR allows the decoder to converge more quickly. Codes such as those in the AR3A family, which appear from Fig. 12 to give a very favorable performance–complexity trade-off at a WER of 10^{-4}, will suffer in comparison to regular (3,6) and AR4JA codes at lower WER values, where the AR3A family's error floors arise (as seen in Figs. 5 and 8).

V. Summary

A code's required bit SNR S (in decibels) on the BI-AWGN channel can be well-approximated by

$$S \approx S^{*BI}(r) + \Delta S^{*CI}(r, k, P_w) + \Delta S_{\mathcal{F}}$$

where the three terms are as follows:

- $S^{*BI}(r)$ is the capacity-constrained threshold. For $1/2 \lessapprox r \lessapprox 16/17$, this is well-approximated by $S^{*BI}(r) = \log_2(r/(1-r))$. That is, each step in the rate sequence 1/2, 2/3, 4/5, 8/9, 16/17 costs about 1 dB of E_b/N_0.
- $\Delta S^{*CI}(r, k, P_w)$ is the finite-size penalty. For $k \gtrapprox 1024$ and $10^{-8} \lessapprox P_w \lessapprox 10^{-2}$, this is well-approximated by $\Delta S^{*CI}(r, k, P_w) = (4/3)\sqrt{1024/k}\sqrt{(-\log_{10} P_w)/8}$. Each shorter block size in the sequence $\infty, 16384, 4096, 1820, 1024$ costs about 1/3 dB of E_b/N_0 at WER $= 10^{-8}$, or $1/(3\sqrt{2})$ dB at WER $= 10^{-4}$, or 1/6 dB at WER $= 10^{-2}$.
- $\Delta S_{\mathcal{F}}$ is the size-constrained non-optimality of the code. Approximately 0.3 to 0.8 dB of required SNR for the codes in Table 1 is attributed to the particular LDPC code design.

For protograph LDPC codes, the size-constrained non-optimality is further subdivided into two components. The protograph non-optimality is computed from the protograph's iterative decoding threshold, determined by density evolution. The expansion non-optimality is due to the selected permutations and

other factors, and is determined by simulation of the finite-size code expanded from the protograph. Our best code designs typically include a protograph non-optimality of around 0.4 dB and an expansion non-optimality of around 0.1 dB. Our attempts to reduce the protograph non-optimality further have typically produced codes with greater expansion non-optimality and/or prominent error floors.

For codes that achieve small size-constrained non-optimality $\Delta S_{\mathcal{F}}$, the average number of edge messages per decoded bit, M (measured in decibels), that must be computed by the decoder is well approximated by

$$M \approx M^*(S_s, k) + \Delta M^*(P_w) + \Delta M_{\mathcal{F}}$$

where S_s is the symbol SNR in decibels and the three terms are as follows:

- $M^*(S_s, k) \approx 10 - 1.2S_s + 0.84 \log_2(k)$ is an empirical measure of complexity dependence on rate, block size, and SNR. Decoder complexity decreases by about 1.2 dB for each 1 dB increase in symbol SNR, and increases by about 0.84 dB for each doubling of block size.
- $\Delta M^*(P_w) \approx 5.6 P_w^{1/4}$ is an empirical measure of complexity dependence on WER, taking a similar shape for all codes studied. Decoder complexity increases roughly in proportion to the fourth root of the word-error rate.
- $\Delta M_{\mathcal{F}}$ is determined by the design of the code family, and it is fairly independent of rate, block size, SNR, and error rate. It ranges between 0 dB and 4 dB for the code families studied.

Code performance and decoder complexity depend upon the design of the LDPC code mainly through $\Delta S_{\mathcal{F}}$ and $\Delta M_{\mathcal{F}}$. We observed in Fig. 12 a trade-off between these two parameters at the boundary of an achievable region, at the approximate rate of about 1 dB of increased complexity for every 0.1 dB of reduced SNR.

References

[1] J. Thorpe "Low-Density Parity-Check (LDPC) Codes Constructed from Protographs," *The Interplanetary Network Progress Report 42-154, April–June 2003*, Jet Propulsion Laboratory, Pasadena, California, pp. 1–7, August 15, 2003. http://ipnpr/progress_report/42-154/154C.pdf

[2] T. Richardson, "Multi-Edge Type LDPC Codes," presented at the Workshop Honoring Prof. McEliece on his 60th birthday (not in the proceedings), California Institute of Technology, Pasadena, California, May 2002.

[3] Y. Kou, H. Tang, S. Lin, and K. Abdel-Ghaffar, "On Circulant Low Density Parity Check Codes," *IEEE International Symposium on Information Theory*, Lausanne, Switzerland, p. 200, June 2002.

[4] A. Sridharan, D. Costello, and R. M. Tanner, "A Construction for Low Density Parity Check Convolutional Codes Based on Quasi-Cyclic Block Codes," *IEEE International Symposium on Information Theory*, Lausanne, Switzerland, p. 481, June 2002.

[5] J. Thorpe, K. Andrews, and S. Dolinar, "Methodologies for Designing LDPC Codes Using Protographs and Circulants," *IEEE International Symposium on Information Theory*, Chicago, Illinois, p. 238, June 2004.

[6] R. G. Gallager, *Low Density Parity-Check Codes*, Cambridge, Massachusetts: MIT Press, 1963.

[7] A. Abbasfar, D. Divsalar, and K. Yao, "Accumulate Repeat Accumulate Codes," *IEEE International Symposium on Information Theory*, Chicago, Illinois, p. 505, June 2004.

[8] S. Dolinar, "A Rate-Compatible Family of Protograph-Based LDPC Codes Built by Expurgation and Lengthening," *IEEE International Symposium on Information Theory*, Adelaide, Australia, pp. 1627–1631, September 2005.

[9] C. E. Shannon, "Probability of Error for Optimal Codes in a Gaussian Channel," *Bell System Technical Journal*, vol. 38, pp. 611–656, 1959.

[10] S. Dolinar, D. Divsalar, and F. Pollara, "Code Performance as a Function of Block Size," *The Telecommunications and Mission Operations Progress Report 42-133, January–March 1998*, Jet Propulsion Laboratory, Pasadena, California, pp. 1–23, May 15, 1998. http://ipnpr/progress_report/42-133/133K.pdf

[11] S. J. Dolinar and F. Pollara, "The Theoretical Limits of Source and Channel Coding," *The Telecommunications and Data Acquisition Progress Report 42-102, April–June 1990*, Jet Propulsion Laboratory, Pasadena, California, pp. 62–72, August 15, 1990. http://ipnpr/progress_report/42-102/102G.PDF