SRC Technical Note
1998 - 009
April 15, 1998

Improved Low-Density Parity-Check Codes Using Irregular Graphs and Belief Propagation

Michael Luby, Michael Mitzenmacher, Amin Shokrollahi, and Dan Spielman

digital

Systems Research Center
130 Lytton Avenue
Palo Alto, California 94301

http://www.research.digital.com/SRC/

Improved Low-Density Parity-Check Codes Using Irregular Graphs and Belief Propagation

Michael G. Luby* Michael Mitzenmacher† M. Amin Shokrollahi‡

Daniel A. Spielman

Abstract

We construct new families of error-correcting codes based on Gallager's low-density parity-check codes, which we call *irregular codes*. When decoded using belief propagation, our codes can correct more errors than previously known low-density parity-check codes. For example, for rate 1/4 codes on 16,000 bits over a binary symmetric channel, previous low-density parity-check codes can correct up to approximately 16% errors, while our codes can correct over 17%. Our improved performance comes from using codes based on irregular random bipartite graphs, based on the work of [7]. Previously studied low-density parity-check codes have been derived from regular bipartite graphs. We report experimental results for our irregular codes on both binary symmetric channels and Gaussian channels. In some cases our results come very close to reported results for turbo codes, suggesting that, with improvements, irregular codes may be able to match turbo code performance.

1 Introduction

Low-density parity-check codes, introduced by Gallager in 1962 [6], and their performance under "belief propagation" decoding, has been the subject of much recent experimentation and analysis [2, 11, 12, 16]. The interest in these codes stems from their near Shannon limit performance, their simple descriptions and implementations, and their amenability to rigorous theoretical analysis [6, 7, 8, 9, 11, 16, 17]. Moreover, there appears to be a connection between these codes and turbo codes, introduced by [1]. In particular, the turbo code decoding algorithm can be understood as a belief propagation based algorithm [10, 5], and hence any understanding of belief propagation on low-density parity-check codes may be applicable to turbo codes as well.

In this paper, we construct new families of low-density parity-check codes, which we call *irregular codes*, that have significantly improved performance over previously known codes of this type. Our codes can correct a significantly higher number of errors, albeit at the expense of a slightly slower running time.

*International Computer Science Institute, Berkeley, CA. Parts of this research were done while still at the Digital Equipment Corporation Systems Research Center, Palo Alto, CA. Research partially supported by NSF operating grant NCR-9416101. E-mail: luby@icsi.berkeley.edu.

†Digital Equipment Corporation, Systems Research Center, Palo Alto, CA. E-mail: michaelm@pa.dec.com.

‡International Computer Science Institute Berkeley, and Institut für Informatik der Universität Bonn, Germany. Research supported by a Habilitationsstipendium of the Deutsche Forschungsgemeinschaft, Grant Sh 57/1–1. E-mail: amin@icsi.berkeley.edu.

Department of Mathematics, M.I.T. E-mail: spielman@math.mit.edu.

Our work significantly lowers the gap between the performance of low-density parity-check codes and the best known turbo codes, opening the question of whether further improvements may yield low-density parity-check codes with equivalent or better performance.

Our codes are most easily visualized in terms of graphs. In the following we refer to the nodes on the left and the right of a bipartite graph as its *message* nodes and *check* nodes respectively. A bipartite graph with n nodes on the left and r nodes on the right gives rise to a linear code of dimension $k \geq n-r$ and block length n in the following way. The bits of a codeword are indexed by the message nodes. A binary vector $\mathbf{x} = (x_1, \ldots, x_n)$ is a codeword if and only if $H\mathbf{x} = 0$, where H is the $r \times n$ incidence matrix of the graph whose rows are indexed by the check nodes and whose columns are indexed by the message nodes. In other words, $(x_1, \ldots, x_n)$ is a codeword if and only for each check node the exclusive-or of its incident message nodes is zero. (We note that our code construction can also be used to construct linear time encodable codes based on cascading series of bipartite graphs, as described for example in [15] or [7], but for convenience we will not address this issue here.)

Most previously studied low-density parity-check codes have been constructed using sparse regular, or nearly regular, random bipartite graphs [2, 6, 11, 12]. We call these codes *regular codes*. Our improved performance comes from using codes based on *irregular* graphs. That is, the degrees of the nodes on each side of the graph can vary widely. In terms of the parity check matrix H, the weight per row and column is not uniform, but instead governed by an appropriately chosen distribution of weights. By carefully choosing the distributions, we achieve improved performance. In fact, the codes we describe use a number of largely disparate weights, suggesting that often the best distributions are far from those that produce regular codes.

As an example of our improved performance, we have found a rate $1/4$ irregular code that, on 16,000 message bits, corrects over 17% random errors with high probability on our experiments. On 64,000 message bits, this code corrects up to 18% random errors on our experiments. In contrast, the best regular code corrects up to approximately 16.0% random errors with 16,000 message bits and approximately 16.2% on 64,000 message bits. (The Shannon bound for rate $1/4$ codes is 21.45%.)

That irregular structure improves performance is not surprising in light of recent work rigorously proving the power of irregular graphs in designing erasure codes [7, 8]. Irregular graphs appear to have been rarely studied in the setting of error-correcting codes because of the difficulty in determining what irregular structures might perform well. The erasure codes and the techniques for finding good erasure codes determined in [7, 8] provide the basis for the codes we define here.

Finally, we note that although we can not provide a formal analysis for the performance of our new codes under belief propagation, we have fully analyzed similar codes that use a simpler, hard decision decoding scheme. This work appears in [9].

The rest of the paper proceeds as follows: we first review the fundamentals of low-density parity-check codes and belief propagation. We then provide some useful intuition for why irregular codes should provide better performance than regular codes. We demonstrate our improved performance with experimental results for both the binary symmetric channel and the Gaussian channel. We conclude with open questions.

2 Low-Density Parity-Check Codes and Belief Propagation

We first review the low-density parity-check codes developed by Gallager and his suggested decoding algorithm, using the framework of MacKay and Neal [6, 11]. The parity check matrix H of the code is obtained by creating a matrix chosen uniformly (or near uniformly) at random such that the weight per column is a fixed constant and the weight per row is as uniform as possible. The decoding algorithm attempts to find the most probable vector $\mathbf{x}$ such that $H\mathbf{x} = 0 \bmod 2$.

In practice, the parity check matrix H can easily be constructed by randomly generating an appropriate bipartite graph. We describe the construction method. Message nodes are located on the left and the check nodes on the right. Each message node has a certain number of edges which connect to check nodes; similarly each check node has a certain number of edges connecting to message nodes. The total number of edges in the graph is e. A random permutation π of $\{1, \ldots, e\}$ is chosen, and then, for all $i \in \{1, \ldots, e\}$, the edge with index i out of the left side is identified with the edge with index π_i out of the right side. Note that this may potentially lead to multi-edges; often in practice multi-edges and small cycles can be removed to improve performance [11].

Gallager's decoding algorithm uses the idea of belief propagation. As explained in [11], the algorithm runs two alternating phases, in which for each non-zero entry in H with row i and column j (or, in other terms, for each edge of the associated bipartite graph) two values q_{ij} and r_{ij} are iteratively updated. The quantity q_{ij}^z approximates the probability that the jth bit of x is z, given the information obtained from all checks of j other than i. Similarly, the quantity r_{ij}^z approximates the probability that the ith check node is satisfied when the jth bit of x is z and all other message bits j' associated with check i have a separable distribution given by the appropriate $q_{ij'}$. That is, we assume that the other message bits j' are independently 1 with probability $q_{ij'}^1$, and use this to calculate r_{ij}^z. If errors occur with probability p, then initially all q_{ij} have the value $1 - p$. In the first phase of a round, all the r_{ij} values are updated in parallel; then in the second phase all the q_{ij} values are updated in parallel. (These parallel updates can also be simulated sequentially in a straightforward manner.) The total amount of work performed each round is linear in the number of edges of the graph. If the bipartite graph defined by H contains no cycles of length up to $2r$, then after r rounds of updates the algorithm produces the exact posterior probabilities that each message bit is in error based on the neighborhood within a diameter of $2r$ of the message node. The presence of cycles in the graph skews the probabilities, but in practice the effect on the algorithm appears to be small. More details can be found in [4, 6, 11, 16].

3 Irregular Graphs: Intuition

Before we demonstrate irregular random graphs that improve the performance of low-density parity-check codes, we offer some intuition as to why irregular graphs should improve performance. Consider trying to build a regular low-density parity-check code that transmits at a fixed rate. It is convenient to think of the process as a game, with the message nodes and the check nodes as the players, and each player trying to choose the right number of edges. A constraint on the game is that the message nodes and the check nodes must agree on the total number of edges. From the point of view of a message node, it is best to have high degree, since the more information it gets from its check nodes the more accurately it can judge what its correct value should be. In contrast, from the point of view of a check node, it is best to have low degree, since the lower the degree of a check node, the more valuable the information it can transmit to its neighbors.

These two competing requirements must be appropriately balanced. Previous work has shown that for regular graphs, low degree graphs yield the best performance [11, 12]. If one allows irregular graphs, however, there is significantly more flexibility in balancing these competing requirements. Message nodes with high degree will tend to their correct value quickly. These nodes then provide good information to the check nodes, which subsequently provide better information to lower degree message nodes. Irregular graph constructions thus lead to a wave effect, where high degree message nodes tend to get corrected first, and then message nodes with slightly smaller degree, and so on down the line.

This intuition (which we observe in our experiments) unfortunately does not provide clues as to how to construct appropriate irregular graphs. Moreover, because belief propagation is not yet well understood mathematically, creating the proper irregular graphs appears a daunting challenge. We meet this challenge by using irregular graphs that have been proven to be effective for erasure codes that function in a similar manner. In the area of erasure codes, the mathematical framework has been established to both design irregular graphs and prove their effectiveness. Intuitively, graphs that work well for erasure codes should

Code Rate 1/2	
Left Degrees	$\lambda_3 = 0.166600$, $\lambda_5 = 0.166600$, $\lambda_9 = 0.166600$, $\lambda_{17} = 0.166600$, $\lambda_{33} = 0.166600$, $\lambda_{65} = 0.166700$
Right Degrees	$\rho_7 = 0.154091$, $\rho_8 = 0.147486$, $\rho_{19} = 0.121212$, $\rho_{20} = 0.228619$, $\rho_{84} = 0.219030$, $\rho_{85} = 0.129561$
Code Rate 1/4	
Left Degrees	$\lambda_3 = 0.166600$, $\lambda_5 = 0.166600$, $\lambda_9 = 0.166600$, $\lambda_{17} = 0.166600$, $\lambda_{33} = 0.166600$, $\lambda_{65} = 0.166700$
Right Degrees	$\rho_4 = 0.160416$, $\rho_{10} = 0.404478$, $\rho_{33} = 0.303338$, $\rho_{34} = 0.131768$

Table 1: Parameters of our codes.

also work well for error-correction codes, since the two are closely related.

4 Irregular Graph Performance: Simulations

We describe a few important details of our experiments and implementations. We performed simulations using two types of channels for several rates and block lengths. The first channel we model is a binary symmetric channel. To more accurately compare code quality, instead of introducing errors with probability p, we introduced the same number of errors at each trial (corresponding to a fraction p of the block length). This procedure allows for easier comparison with other codes and minimizes the variance in the experiments that might arise from the variance in the number of errors. The second channel type we model is a white Gaussian channel with binary input ± 1 and an additive noise of variance σ^2. We report results for the Gaussian channel of rate R and additive noise σ^2 in terms of the signal to noise ratio $E_b/N_0 = 1/2R\sigma^2$ expressed as decibels ($10 \log_{10} E_b/N_0$). Here E_b represents the average energy per bit ($1/R$) and N_0 represents the noise spectral density ($2\sigma^2$).

Rather than encoding a message for each trial, we use an initial message consisting entirely of zeroes. Since the code is linear and the decoding algorithm respects its linearity, no generality is lost. In our experiments, we allowed the belief propagation algorithm to run for up to 200 rounds. If the algorithm failed to converge on a codeword within 200 rounds, a failure was reported. This was in fact the only failure we saw in our experiments; that is, the algorithm never returned a codeword that differed from the initial message.

A different random graph was conducted for each trial. No effort was made to test graphs and weed out potentially bad ones, and hence we expect that our results would be slightly better if several random graphs were tested and the best ones chosen. Also, following the ideas of [11, 14], when necessary we remove double edges from our graphs.

We describe the irregular graphs used in Table 1, using the notation of [7]. We say that an edge has degree i on the left if its adjacent node on the left has degree i, and we similarly define an edge with degree i on the right. Our graphs are described in the following terms: for each graph, there is a corresponding left degree vector λ and a right degree vector ρ. The value λ_i represents the fraction of *edges* with degree i on the left, and similarly the value ρ_j represents the fraction of *edges* with degree j on the right. Note that given a vector λ and ρ one can construct a graph with (approximately) the correct edge fractions for any number of nodes, using the construction method described in Section 2. (Some care must be taken because of rounding and the necessity to have the number of edges on the left equal the number of edges on the right; however, this is easily handled.)

	n	R	f	C	errs	trials
Reg	16000	0.5	0.078	0.605	0	10000
		0.5	0.080	0.598	35	10000
		0.5	0.082	0.591	1033	10000
Irreg	16000	0.5	0.078	0.605	1	10000
		0.5	0.080	0.598	14	10000
		0.5	0.082	0.591	40	10000
		0.5	0.084	0.584	116	10000
Reg	64000	0.5	0.082	0.590	1	1000
		0.5	0.084	0.583	249	1000
Irreg	64000	0.5	0.086	0.577	0	1000
		0.5	0.088	0.570	0	1000
		0.5	0.090	0.563	25	1000
Reg	16000	0.25	0.158	0.370	0	10000
		0.25	0.160	0.366	0	10000
		0.25	0.162	0.361	45	10000
		0.25	0.164	0.356	697	10000
		0.25	0.166	0.352	3767	10000
Irreg	16000	0.25	0.166	0.352	0	10000
		0.25	0.168	0.347	0	10000
		0.25	0.170	0.342	4	10000
		0.25	0.172	0.338	15	10000
		0.25	0.174	0.333	53	10000
Reg	64000	0.25	0.164	0.356	0	1000
		0.25	0.166	0.352	176	1000
Irreg	64000	0.25	0.178	0.324	0	1000
		0.25	0.180	0.320	2	1000
		0.25	0.182	0.316	63	1000

Table 2: Comparing regular and irregular graphs.

4.1 Binary Symmetric Channel

Table 2 compares the performance of regular and irregular codes of rates $1/2$ and $1/4$. Our results for regular codes (based on graphs in which all nodes on the left have degree 3) are slightly better than (but consistent with) previous results reported in [11]. In the table, n represents the block length, R represents the rate, f represents the fraction of errors introduced, and C represents the channel capacity. The results are reported in terms of the number of trials, or blocks decoded, and the number of errors, or the number of blocks for which the decoding algorithm failed to find a solution within 200 rounds.

At rate $1/2$, our irregular codes perform only slightly better than the regular codes at block lengths of 16,000 bits. They do appear notably more robust at higher error rates, however. At 64,000 bits, our code can handle over a half a percent more errors. While it has been previously noted that low-density parity-check codes perform better as the block length increases [11], we believe that this effect is magnified for our irregular codes, because the degrees of the nodes can be quite high. For example, our irregular rate $1/2$ codes have nodes on the left of degree 65 and nodes on the right of degree 85.

At rate $1/4$, we have different irregular codes with lower degrees. This code greatly outperforms the regular codes, even at block lengths of 16,000, where they correct approximately 1% more errors. At block

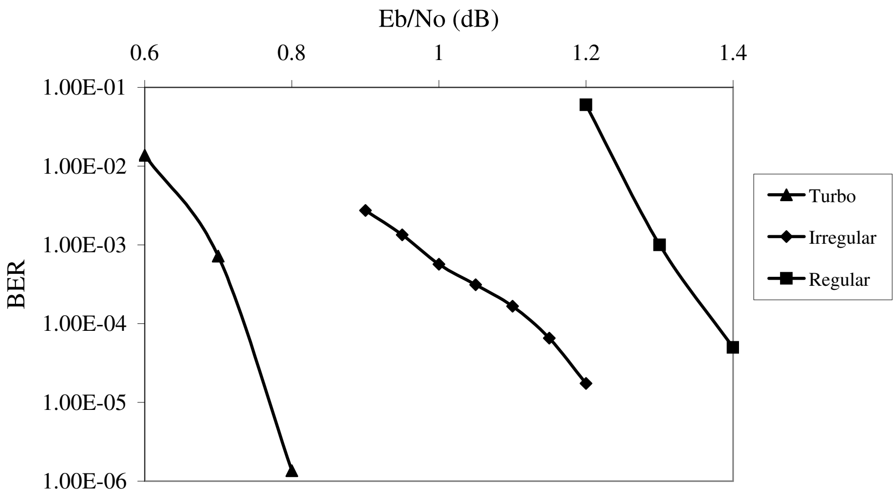

Figure 1: Irregular codes vs. regular codes and turbo codes: rate $1/2$.

lengths of 64,000 bits, the effect is even more dramatic, and the irregular codes appear to correct more than 1% more errors. We note that initial experiments at other rates further validate our contention that irregular codes can outperform regular codes in terms of the number of errors that can be corrected.

For both regular and irregular codes, the number of operations required is proportional to the product of the number of edges in the corresponding graph and the number of rounds until the process terminates. The irregular graphs have approximately 2.5 times as many edges as the regular graphs, and at higher error rates they can take approximately 1.5 times as many iterations to complete. Hence it takes approximately 4 times as many operations to decode at higher rates. In practice, performance can actually be worse than this, however, since the larger graph size for the irregular codes requires more accesses to slower levels of the memory hierarchy. However, we believe the slower running time is not dramatic in light of the improved performance.

4.2 Gaussian Channel

Figures 1 and 2 compare the performance (in terms of the bit error rate (BER)) of irregular codes of rate $1/2$ and $1/4$ with reported results for turbo codes [3] and regular codes [12] at these rates. Again, our experiments were with block lengths of 16,000 bits, and for this block length each data point is the result of 10,000 trials. (We compare with results using comparable block lengths. The results from [3] are available at `http://www331.jpl.nasa.gov/public/TurboPerf.html`. The results we report from [12] are only approximate, as [12] does not provide actual numbers but only a graph.) For our irregular codes, the belief propagation algorithm terminated after 200 rounds if the solution was not found.

For rate $1/2$ codes, our irregular codes perform notably better than regular codes, greatly reducing the gap between the performance of low-density parity-check codes and turbo codes. This gap is further reduced when we move to larger block sizes, as our codes prove to perform better for larger block lengths in this setting as well. At block lengths of 64,000 bits, our code never failed in 1,000 trials at both 0.95 dB and 1.00 dB. Our estimates for the bit error rate from 1,000 trials at 0.9 dB is $3.97 \cdot 10^{-5}$ and at 0.85 dB is $6.03 \cdot 10^{-5}$. Again, this is much better than the performance of regular codes at a comparable block length presented in [12].

Our results for irregular codes at rate $1/4$ (Figure 2) similarly show significant improvement over regular

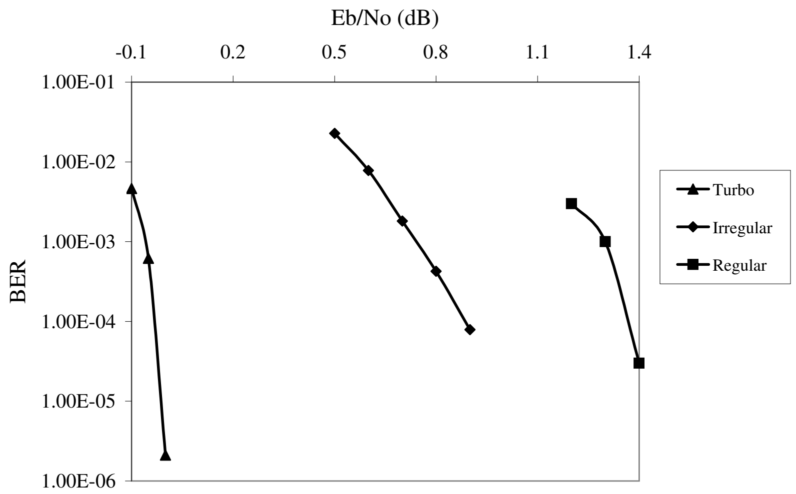

Figure 2: Irregular codes vs. regular codes and turbo codes: rate $1/4$.

codes. At this lower rate and block length, however, turbo codes appear to have a significant edge. We believe this edge can be reduced with further exploration and experimentation.

Our irregular codes at this rate again perform significantly better with larger block lengths. At block lengths of 64,000 bits, our code never failed in 1,000 trials at both 0.70 dB and 0.60 dB. Our estimates for the bit error rate from 1,000 trials at 0.50 dB is $6.18 \cdot 10^{-4}$ and at 0.40 dB is $8.39 \cdot 10^{-3}$.

5 Conclusions

We have demonstrated that low-density parity-check codes based on irregular graphs can substantially outperform similar codes based on regular graphs. Our irregular graph based codes approach the performance of turbo codes at certain rates.

Our graphs were designed using the techniques of [7]. These graphs are known to yield good loss-resilient codes in the model of [7], but there is no clear reason why there cannot be substantially better codes for error-correction based on other irregular graphs. If this is the case, then it may be possible to achieve performance similar to that of turbo codes using irregular codes. Designing and analyzing such codes remains an exciting open problem.

References

[1] C. Berrou, A Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes", *Proceedings of IEEE International Communications Conference, 1993.*

[2] J.-F. Cheng and R. J. McEliece, "Some High-Rate Near Capacity Codecs for the Gaussian Channel", *34th Allerton Conference on Communications, Control and Computing, 1996.*

[3] D. Divsalar and F. Pollara, "On the Design of Turbo Codes", *JPL TDA Progress Report 42-123.*

[4] G. D. Forney, Jr. "The Forward-Backward Algorithm", *Proceedings of the 34th Allerton Conference on Communications, Control and Computing, 1996*, pp. 432-446.

[5] B. J. Frey and F. R. Kschischang, “Probability Propagation and Iterative Decoding”, *Proceedings of the 34th Allerton Conference on Communications, Control and Computing, 1996*.

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

[7] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, “Practical Loss-Resilient Codes”, *Proc.* 29^{th} *Symp. on Theory of Computing*, 1997, pp. 150–159.

[8] M. Luby, M. Mitzenmacher, and M. A. Shokrollahi, “Analysis of Random Processes via And-Or Trees”, *Proc.* 9^{th} *Symp. on Discrete Algorithms*, 1998.

[9] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. Spielman, “Analysis of Low Density Codes and Improved Designs Using Irregular Graphs”, submitted to STOC 1998.

[10] D. J. C. MacKay, R, J. McEliece, and J.-F. Cheng, “Turbo Coding as an Instance of Pearl’s ’Belief Propagation’ Algorithm”, to appear in *IEEE Journal on Selected Areas in Communication.*

[11] D. J. C. MacKay and R. M. Neal, “Good Error Correcting Codes Based on Very Sparse Matrices”, available from **http://wol.ra.phy.cam.ac.uk/mackay**.

[12] D. J. C. MacKay and R. M. Neal, “Near Shannon Limit Performance of Low Density Parity Check Codes”, to appear in *Electronic Letters*.

[13] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, 1988.

[14] M. Sipser, D. A. Spielman, “Expander Codes”, *IEEE Transactions on Information Theory*, 42(6), November 1996, pp. 1710-1722.

[15] D. A. Spielman, “Linear Time Encodable and Decodable Error-Correcting Codes”, *IEEE Transactions on Information Theory*, 42(6), November 1996, pp. 1723-1731.

[16] N. Wiberg, “Codes and decoding on general graphs” Ph.D. dissertation, Dept. Elec. Eng, U. Linköping, Sweeden, April 1996.

[17] N. Wiberg, H.-A. Loeliger and R. Kötter, “Codes and iterative decoding on general graphs”, *European Transactions on Telecommunications*, vol. 6, September 1995, pp. 513-526.