Space-Time Turn Prohibitions for Low Density Parity-Check Codes

Ari Trachtenberg Mark Karpovsky
Reliable Computing Laboratory
Boston University

1 Introduction

In 1948 Shannon [1] proved that error-free communication is possible through a communication channel at rates up to its capacity. Unfortunately, Shannon's proof was non-constructive and practically decodable codes achieving such capacity remain elusive to this day. Low Density Parity-Check codes, originally introduced by Gallager [2], have recently regained popularity because they have experimentally come very close to the Shannon limit (within 0.045dB recently [3]), promising that an efficient implementation might finally provide optimal error-correction capabilities for various channel models.

Low Density Parity-Check (LDPC) codes are decoded with an iterative decoding algorithm, known as the sum-product algorithm, that operates on a graph that represents the code. The algorithm is actually fairly general, encompassing as special instances both the forward-backward algorithm on a trellis, due to Bahl, Cocke, Jelinek, and Raviv [4], and the decoding algorithm for the well-known Turbo codes [5, 6]. In a modified form as the min-sum algorithm it is also the classical Viterbi algorithm for a trellis [7]. This decoding algorithm is naturally parallelizable into a message-passing network of very simple processors, allowing for efficient implementation on a chip in many practical scenarios.

In the case where the underlying graph used to represent an error-correcting code is cycle-free, the sum-product algorithm is known to converge to an optimal codeword. However, little is known about the convergence of the algorithm for graphs with cycles [8, 9]; in some cases the algorithm does not converge. Recent work [10] shows that cycle-free graphs only admit error-correcting codes with a very low minimum distance, and consequently poor asymptotic performance.

Thus, graph cycles are a difficulty inherent in this decoding scheme. The fact that maximum likelihood decoding of a linear code is an NP-hard problem [11] confirms the intuition that decoding on a graph with cycles is intrinsically difficult. For this reason, we feel it is important to look at techniques for breaking cycles in such graphs.

The concept of cycle-breaking is not new [8]. The traditional techniques for doing this involve removing edges until the graph is cycle free; however, removal of an edge alters the underlying error-correcting code and results in a weak code. An alternative technique for

breaking cycles involves repeatedly splitting nodes in a cycle into two duplicates forming an infinite, cycle-free graph that can only be decoded probabilistically.

We propose applying a novel method of cycle-breaking using turn prohibitions to mitigate the effects of cycles in the graph of an error-correcting code. The idea is to prohibit (or slow) information from traveling along certain turns in the graph. For example, three vertices v_1, v_2, and v_3 could be designated as a prohibited turn so that information from v_1 to v_3 would not be permitted to flow through v_2.

Such prohibitions can be applied naturally to the sum-product algorithm. We may enforce these prohibitions for specific decoding iterations, giving time-varying prohibitions, or with specific dampening weights, giving space-varying prohibitions. The union of these models, which we call space-time prohibitions, shows promise for both improved decoding performance and provable convergence. Moreover, since these prohibitions merely improve the decoding process, without changing the underlying code, they can be applied to the best LDPC codes as they are discovered.

2 Background

The sum-product decoding algorithm operates on the *Tanner graph* [12] of a corresponding error-correcting code. Tanner graphs are specific instances of a more general graph introduced by Wiberg, Loeliger, and Kötter [7] for factoring computations into minimal constituents. Formally, the Tanner graph of a code is simply the incidence graph of its parity-check matrix. This graph is bipartite; the nodes of one part, known as the *symbol nodes*, represent bits of the received vector and the nodes of the other part, known as *check nodes*, impose check constraints on the symbol as dictated by the underlying error-correcting code. Figure 1 shows the Tanner graph for the code given by the following parity-check matrix:

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

Note that the Tanner graph of a code is not sensitive to coordinate permutation, but is sensitive to the choice of parity-check matrix. Thus, the Tanner graph for one parity-check matrix might have cycles whereas the Tanner graph for the same code under a different parity-check matrix might not.

2.1 The sum-product algorithm

Given a Tanner graph G for a linear code C, decoding is typically performed using the *sum-product algorithm*, which is a generalization of approximate belief propagation [13].

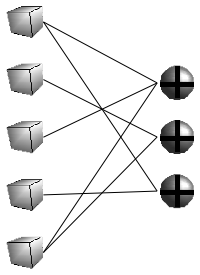


Figure 1: The Tanner graph for a code determined by the parity check matrix (1).

Thus, in each iteration, each symbol node sends to its neighboring check nodes a probability associated with its belief that the corresponding transmitted bit was a 0, and likewise each check node sends its own belief (based on its own neighboring symbols) back to the symbol. Ideally, this iterative scheme converges to reveal the transmitted vector.

More formally, consider sending the codeword $\mathbf{c}$ over a noisy channel, resulting in the received vector $\mathbf{z} = \mathbf{c} + \mathbf{e}$. Adopting the notation in [9, 14], we associate quantities $q^a_{m,n}$ with the probability that symbol n of $\mathbf{z}$ is a, given information from checks other than m; likewise we associate $r^a_{m,n}$ with the probability that check m is satisfied if symbol n of $\mathbf{z}$ is set to a. The characteristics of the communication channel are instantiated through the variable f^a_n, which is set to the likelihood that $z_n = a$.

The algorithm is initialized by setting $q^a_{m,n} = f^a_n$ for all m, n. Thereafter, q and r quantities are updated in some order according to the rules

$$q^a_{m,n} = \alpha_{m,n} \; f^a_n \prod_{\substack{\text{all checks } j \neq m \\ \text{neighboring symbol } n}} r^a_{j,n} \tag{2}$$

(3)

$$r^a_{m,n} = f^a_n \sum_{\text{all vectors } x \text{ satisfying check } m} \; \prod_{\text{all sites } j \neq n \text{ neighboring } m} q^{x_j}_{m,j} \tag{4}$$

where $\alpha_{m,n}$ is a normalization coefficient chosen to make $q^0_{m,n} + q^1_{m,n} = 1$. The iteration continues until one of three conditions is met:

- the values $\mathbf{q}$ and $\mathbf{r}$ converge
- a maximum number of iterations (typically between 200 and 1000) is reached

- the decoding $\mathbf{d}$ given by:

$$d_n = \text{argmax}_a \quad f_n^a \prod_{\substack{\text{all checks } j \text{ neighbor-} \\ \text{ing symbol } n}} r_{j,n}^a$$

  satisfies all the parity-checks of the code

2.2 Low Density Parity-Check Codes

The running time of sum-product algorithm is clearly exponential in the maximum degree of a check node in the underlying graph. Thus, for practical decoding this maximum degree has to be kept fairly low. One of the surprising qualities of Gallager's LDPC codes [2, 15] is that these codes both afford practical decoding and also support very powerful error correction.

Formally, Gallager LDPC codes are defined by a very sparse, random parity-check matrix. Given a transmitted block length N and a source block length K, this $(N-K) \times N$ matrix is constructed to have t ones in each column and as close to $tN/(N-K)$ ones in each row as possible. MacKay and Neal [9, 16] applied several heuristics to avoid particularly bad random choices for these matrices. They showed that such LDPC codes, including their own variants called MN-codes, are clearly competitive with the latest Turbo code realizations and come very close to Shannon's capacity.

2.3 Turn prohibitions

Let $G = (E, V)$ be an undirected graph with vertices V and edges E. A cycle C of length L in this graph is a path whose initial and final vertices are the same. A turn (a, b, c) is a triple of vertices in V along some path; it is said to break the cycle C if a, b, and c are consecutive vertices in C.

We denote by $Z(G)$ the minimal set of turns breaking all cycles in G (including those cycles where the same node or edge appears several times) while preserving connectivity. Thus, for every cycle in G there is a corresponding turn in $Z(G)$; moreover, any two nodes v and w are on some path that does not contain any turns in $Z(G)$. We shall say that $Z(G)$ is an irreducible turn prohibition set for G if no proper subset of $Z(G)$ breaks all cycles in G.

The problem of constructing minimal turn prohibition sets is important for developing deadlock-free routing protocols in computer communication networks [17]. One approach for solving this problem is known as the "up and down" approach [18]. In this case, we first construct a rooted spanning tree, which imposes a partial order on the graph (i.e. $a < b$ if a is a parent of b in the tree). Then, turn (a, b, c) is prohibited if $a < b < c$. With this approach, the fraction of prohibited turns for a given graph depends on the chosen spanning

tree and may be close to 1. It was shown in [17] that, for any graph G it is possible to construct an irreducible turn prohibition set $Z(G)$ with the property that

$$|E| - |V| + 1 \leq |Z(G)| \leq \frac{1}{6} \sum_{i=1} |V| d_i (d-1), \tag{5}$$

where d_i is the number of neighbors of vertex $v_i \in V$. The amount of time and memory needed for constructing $Z(G)$ meeting (5) is at most $O(|V|^2)$.

In the case of the Tanner graph of a length N LDPC code with check degree t and site degree $\frac{tN}{N-K}$, equation (5) becomes

$$(t-1)N + 1 \leq |Z(G)| \leq \frac{1}{2} N \min \left\{ \frac{1}{3}(t(t-1) + \frac{t}{R}(\frac{t}{R} - 1)), t(t-1) \right\},$$

where $R = 1 - \frac{K}{N}$. For example, for $t = 3$ and $R = 0.5$ we have that

$$2N + 1 \leq |Z(G)| \leq 3N.$$

Thus, by prohibiting a number of turns that is linear in the length of the code, one can break all cycles in an arbitrary LDPC code and force belief propagation to converge quickly.

3 Space-time turn prohibitions

As mentioned in the introduction, little is known about the convergence of the sum-product algorithm over a graph with cycles. Figure 2 shows one example of a Tanner graph for a simple repetition code for which the sum-product algorithm does not converge. Traditional means of breaking cycles in a graph (*i.e.* forming a minimum spanning tree) either reduce the practicality of decoding or else result in a trivial code.

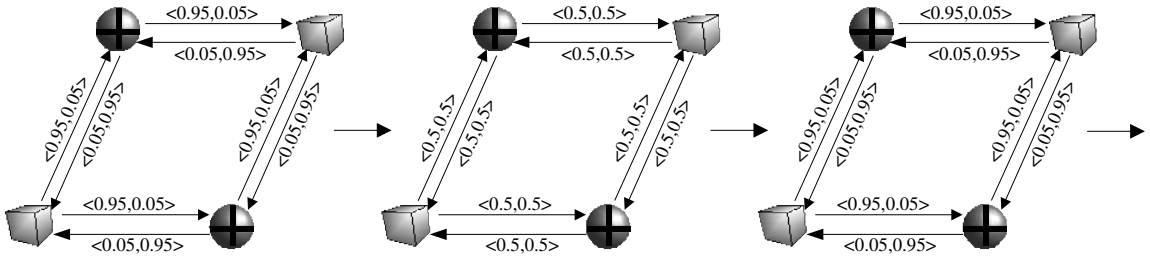


Figure 2: Messages transmitted in the sum-product decoding of the Tanner graph of a repetition code. In this case, a vector [0,1] has been received over a Binary Symmetric Channel with crossover-probability 0.05. The algorithm does not converge, unable to decide which vector was actually transmitted.

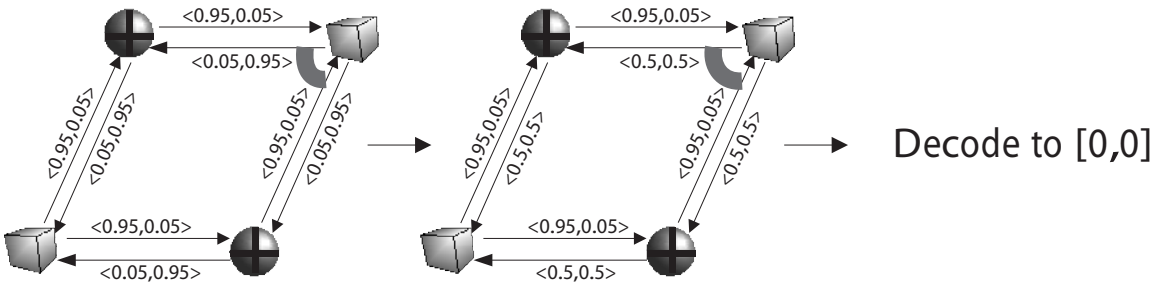


Figure 3: Sum-product decoding of the Tanner graph of a repetition code, where one turn has been prohibited. In this case, a vector [0,1] has been received over a Binary Symmetric Channel with crossover-probability 0.05. The prohibition of a turn forces the algorithm to converge to some decoding.

Turn prohibition provides a convenient method of breaking cycles without changing the underlying error-correcting code and, at the same time, improving the convergence of the decoding algorithm. Fast convergence is particularly important for LDPC codes because the best codes have lengths up to tens of millions [3] of symbols, meaning that unnecessary iterations significantly affect the decoding complexity.

Using the turn prohibition model, we prohibit information from flowing along proscribed turns by restricting the update equations (2) and (4) as follows:

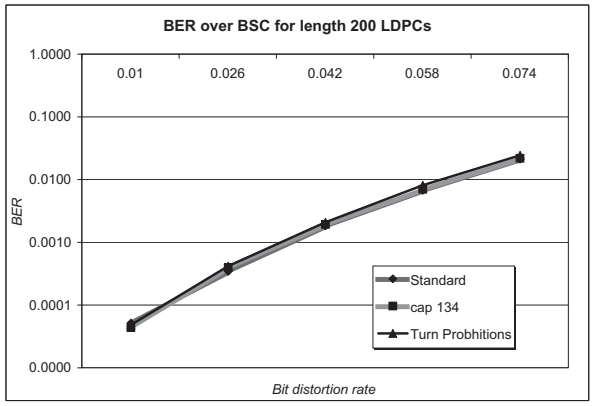
$$q^a_{m,n} = \alpha_{m,n} f^a_n \prod_{\substack{\text{all checks } j \neq m \text{ neighboring symbol } n;\ (m,n,j) \text{ is not a prohibited turn}}} r^a_{j,n}$$

$$r^a_{m,n} = f^a_n \sum_{\text{all vectors } x \text{ satisfying check } m} \prod_{\substack{\text{all sites } j \neq n \text{ neighboring } m;\ (m,n,j) \text{ is not a prohibited turn}}} q^{x_j}_{m,j}$$

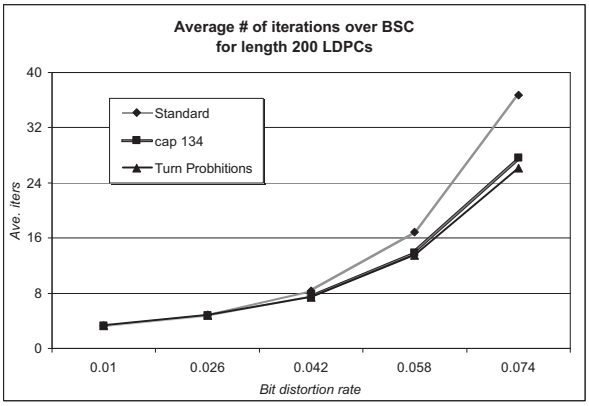
Figure 3 shows that prohibiting one turn from the Tanner graph in 2 can cause the sum-product algorithm to converge where, otherwise, it would not.

It appears that the proof in [12, 8] that the sum-product algorithm converges on a cycle-free graphs transfers straightforwardly to turn-prohibited graphs. However, unlike cycle-free graphs, turn-prohibited graphs do not necessarily yield optimal decodings. In fact, cycles do play an important role in decoding performance. For this reason, it is desirable to enforce turn prohibitions only for certain iterations, in a time-varying manner. We call such enforcement *time turn prohibitions*.

Figure 4 shows experimental results for a very short Gallager LDPC of length 200. Note that turn prohibition dramatically improves the average number of iterations in the sum-product



(a)



(b)

Figure 4: Performance of length=20, rate=3/4 Gallager LDPC's on a binary symmetric channel using three different variations of the sum-product algorithm: [Standard] - the standard algorithm cutoff after 201 iterations; [Turn prohibition] - the standard algorithm with turn prohibitions added after 20 iterations; [Cutoff] - the standard algorithm cutoff after 32 iterations (i.e. the maximum number of iterations used by the algorithm with turn prohibitions). All algorithms were run on the *exactly* the same input data.

algorithm without significantly affecting the probability of error; in fact, the sum-product algorithm with turn prohibitions never takes more than 134 iterations to decode an error. Using turn prohibition even results in a smaller average number of iterations than stopping decoding after 134 iterations, without compromising the bit error rate.

3.1 Space prohibitions

It is clear that turn prohibitions serve as a damping mechanism preventing information from feeding back on itself in a graph cycle. At the same time, cycles are intrinsic to good decoding performance. For this reason, we consider *space turn prohibitions*, where instead of forbidding a turn outright we simply add a dampening coefficient to specific turns that forces convergence of information that is cycling.

Thus, for example, every turn (m, n, j) around a site m now contributes $\left(r^a_{j,n}\right)^{c_{m,n,j}}$ to any overall product, rather than just $r^a_{j,n}$, where $c_{m,n,j}$ is the coefficient corresponding to the turn (m, n, j). The value $c_{m,n,j} = 1$ corresponds to a completely permitted turn and the value $c_{m,n,j} = 0$ corresponds to a completely prohibited turn; values in between 0 and 1 correspond

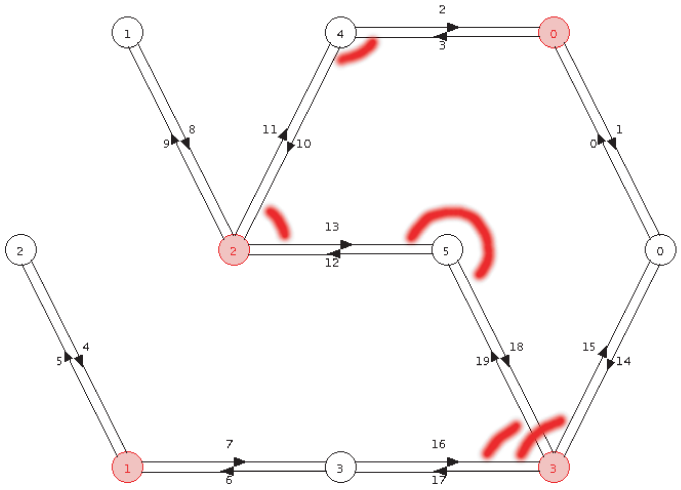


Figure 5: An example of performance improvement using a simple block code with 6 symbols (i.e. the circles) and 4 parity-checks (i.e. the shaded circles) over a Binary Symmetric Channel with crossover probability 0.1. Shaded turns correspond to turn prohibitions with coefficient 0.5 whereas unshaded turns have coefficient 1.0.

to progressive damping of the turn. The update equations (2) and (4) thus become:

$$q^a_{m,n} = \alpha_{m,n} f^a_n \prod_{\substack{\text{all checks } j \neq m \\ \text{neighboring symbol } n}} \left(r^a_{j,n}\right)^{c_{m,n,j}} \tag{6}$$

(7)

$$r^a_{m,n} = f^a_n \sum_{\substack{\text{all vectors } x \text{ satisfying check } m}} \prod_{\substack{\text{all sites } j \neq n \text{ neighboring } m}} \left(q^{x_j}_{m,j}\right)^{c_{m,n,j}} \tag{8}$$

Thus, we can first determine a minimal set of turn prohibitions for cycles in a graph and then adjust the weights of these prohibitions for optimal performance over a specific channel. One reasonable weight model involves assigning to each vertex a weight related to the length of the shortest cycle including that vertex. Figure 5 shows improvements gleaned using space turn prohibitions with coefficients 0.5 and 1 alone. Specifically for this code, standard sum-product decoding requires an average of 2.755 iterations (ranging from 1 for the best trial to 62 for the worst trial); decoding with generalized turn prohibitions, on the other hand, requires an average of 1.642 iterations (ranging from 1 to 11). Moreover, in this small example, decoding with generalized turn prohibitions results in 56 fewer bit errors over $100,000$ simulated transmissions.

In effect, we can improve the performance of some codes with appropriate use of damping.

This would suggest that it is possible to tune performance of a general LDPC code to a specific channel with the appropriate space-time turn prohibitions.

4 Conclusion

We have considered a novel method of breaking cycles in the Tanner graph of Low Density Parity-Check codes. By prohibiting specific turns in such a graph, one can force the convergence of the belief-propagation algorithm, typically making it much faster than without turn prohibition. These turn prohibitions can be scheduled either in a time-varying fashion, in which turns are prohibited only for certain iterations of the decoding algorithm, or in a space-varying fashion, in which changes along specific turns are steadily dampened, or both fashions. Experimental evidence suggests that that error-correcting performance and the convergence rate of the decoding algorithm can be improved through the careful selection of space-time turn prohibitions.

References

[1] C.E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 623–656, 1948.

[2] R.G. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, vol. 8, pp. 21–28, jan 1962.

[3] S.-Y. Chung, Jr. G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 db from the shannon limit," *IEEE Commun. Letters*, vol. 5, pp. 58–60, February 2001.

[4] L.R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, pp. 284–287, 1974.

[5] A. Glavieux C. Berrou and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: turbo codes," *Proc. IEEE Int. Conf. on Communications*, pp. 1064–1070, 1993.

[6] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Trans. Inform. Theory*, vol. 20, pp. 1261–1271, 1996.

[7] H.-A. Loeliger N. Wiberg and R. Kötter, "Codes and iterative decoding on general graphs," *Euro. Trans. Telecommun.*, vol. 6, pp. 513–526, 1995.

[8] Niclas Wiberg, *Codes and Decoding on General Graphs*, Ph.D. thesis, Linkoping University, 1996.

[9] D.J.C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, March 1999.

[10] Ari Trachtenberg Tuvi Etzion and Alexander Vardy, "Which codes have cycle-free tanner graphs?," Tech. Rep. 0925, Technion–Israel Institute of Technology, November 1997, Submitted to IEEE Trans. on Inf. Theory.

[11] E.R. Berlekamp, R.J. McEliece, and H.C.A. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Transactions on Information Theory*, vol. 24, pp. 384–386, 1978.

[12] R.M. Tanner, "A recursive approach to low-complexity codes," *IEEE, Trans. Inform. Theory*, vol. 27, pp. 533–547, 1981.

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

[14] M.C. Davey and D.J.C. MacKay, "Low density parity check codes over gf(q)," *Proceedings of the 1998 IEEE Information Theory Workshop*, June 1998.

[15] R.G. Gallager, *Low-density parity-check codes*, MIT Press, Cambridge, MA, 1963.

[16] D.J.C. MacKay and R.M. Neal, "Near shannon limit performance of low-density parity-check codes," *Electronics Letters*, vol. 32, pp. 1645–1646, 1996.

[17] L.A. Zakrevski, *Fault-tolerant wormhole message routing in computer communication networks*, Ph.D. thesis, Boston University, 2001.

[18] M. Schroeder, "Autonet: a high-speed, self-configuring, local area network using point-to-point links," 1991.