Design of LDPC Permutations for Hardware Implementation

Jeremy Thorpe

December 31, 2001

Abstract

A methodology for generating bipartite graphs for LDPC codes which both exhibit good performance under message passing decoding and are particularly ammenable to direct hardware implementation is described. To this end, we define an apparently novel quantitative measure of the "loopiness" of a graph, as well as a quantitative measure of the cost of direct hardware implementation, and use the well-known simulated annealing algorithm to simultaneously minimize both quantities. [Finally, we simulate the decoding of several rather short codes to show that the performance is indeed predicted by our loopiness measure.]

1 Introduction

Recently, researchers have used large ensembles of very long block-length Low-density-parity-check (LDPC) codes [1] to demonstrate very nearly capacity achieving performance [3], [6] with low encoding and decoding complexity. Less explored, however, is the problem of designing good graphs for short block-length codes. Recently, Mao and Banihashemi have proposed selection of graphs based on the "average girth distribution" [5] with apparent success. MacKay [4] has also used another set of heuristics to modify a random graph into a better one.

In this paper, we [will] make a systematic attempt to find criteria which accurately predict code performance, and use powerful optimization algorithms (as opposed to just selection) to find good graphs by such criteria. This should help to answer the question of how much the performance of short block-length LDPC's vary over different graphs. Further, we use such optimization algorithms to design codes which not only should have good performance, but will be ammenable to easy hardware design.

2 Performance and Cost Measures

Regular LDPC's are characterized by a bipartite graph $g = (V, C, E)$, where any edge in E connects a vertex in V with a vertex in C. The vertices in V represent symbols to be transmitted over a channel, while the vertices in c represent parity-check equations of the adjacent edges. In addition, there are λ edges incident with each node in V and ρ edges incident with each node in C.

In this section, we introduce two quantities which can be efficiently calculted and which have to do with the performance and implementation cost, respectively. These quantities are minimized using the Simulated Annealing algorithm, which we discuss in the following section.

2.1 Performance Measure

It is well known that the Message-passing algorithm [2], run for i iterations, calculates exactly the a posteriori probability of each code symbol x_j (corresponding to the node v_j)given the received symbols corresponding to nodes in a certain neighborhood of nodes reachable from v_j in $2i$ steps, but only as long as the that neighborhood is tree-like, meaning that it has no cycles. In order to preserve the exactness of the message-passing algorithm for as many iterations as possible, many have suggested maximizing the girth of a graph, the length of it's smallest cycle.

In this, we define a measure of loopiness which counts *all* of the loops in a graph, weighted by an exponential of its length:

$$L(g, \alpha) = \sum_{i=2,4..} \alpha^i N_i \tag{2.1}$$

where N_i represents the number of loops of length i in the graph. A loop is defined as a sequence $v_1, c_2, v_3, ..c_i$ such that an edge exists between v_{2j} and $c_{2j\pm1}$ for all j, and $v_{2j} \neq v_{2j+2}$ and $c_{2j-1} \neq c_{2j+1}$ for all j. All indexes are taken modulo i. It can be shown that this sum converges, and therefore that the measure is defined, only for $\alpha^2 < (\lambda-1)(\rho-1)$ for regular LDPC's. For values of α smaller than this threshhold, there exist algorithms to efficiently estimate this expression for loopiness of a graph.

2.2 Cost Measure

Since the aim of this research is to generate codes whose quantized message-passing algorithms [2] are easily implementable in direct-form on a microchip, we must consider the cost of such an implementation. direct-form realization refers to the method of instantiating many logic units, each dedicated to making the calculations associated with a particular vertex (in V or C) operating simultaneously to perform the decoding. The messages are passed along wires which are instantiated for each edge in the graph.

It is not difficult to see that if the graph is chosen at random and the vertices of the graph of places at random in a square area, the average length of wire increases like $O(n^{\frac{1}{2}})$, and thus the wiring complexity grows like $O(N^{\frac{3}{2}})$ while the complexity of the logic grows like $O(n)$. It is therefore natural to relate the implementation cost to the total length of wires needed to instantiate all of the edges in the graph. For simplicity, we take the cost measure $W(g)$ to be the sum of the Manhattan distance between the endpoints of all edges in the graph, where the nodes occupy fixed positions.

$$W(g) = \sum_{e \in E} \text{length}(e) \tag{2.2}$$

3 Optimizing with Simulated Annealing

The simulated annealing algorithm is a probabilistic, iterative algorithm designed to approximately solve energy minimization problems, and can be successfully applied to a broad class of such problems. For this problem, we search for a minimum of:

$$E(g) = C_L \cdot L(g, \alpha) + C_W \cdot W(g) \tag{3.1}$$

for some pre-selected α and coefficients C_L and C_W.

In addition to an energy function, a set of transformations $\mathbb{T}(g) = g'$ must also be defined which take a solution into another solution. Such a transformation should have the property that the transformed solution has an energy somewhat similar to the original solution. We take the set of transformation to be those by which two edges, say $v_1 \to c_1$ and $v_2 \to c_2$ are replaced with the edges $v_1 \to c_2$ and $v_2 \to c_1$.

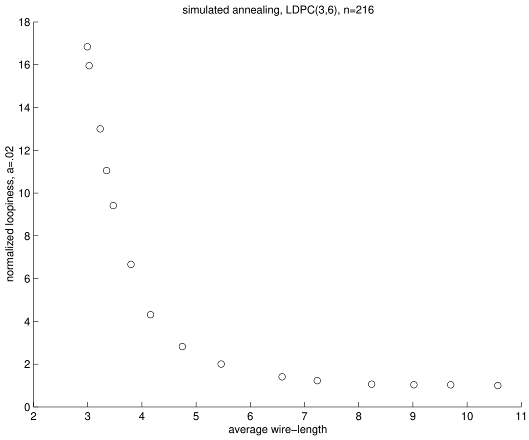
Finally, we define a positive profile $T(i)$ which defines the "temperature" of the system at each iteration i. Conservatively, $T(i)$ can start at a very high temperature decay exponentially with i to end at a very low temperature. The simulated annealing begins with an arbitrary solution g_0. For each i, the simulated annealing algorithm selects a transformation $\mathbb{T}(g_t)$ and selects

$$g_{t+1} = \begin{cases} g_t & w.p \frac{e^{E(g_t)/T(i)}}{A} \\ g_t' & w.p \frac{e^{E(g_t')/T(i)}}{A} \end{cases} \tag{3.2}$$

where A is chosen to make this a valid probability distribution, namely $A = e^{E(g_t)/T(t)} + e^{E(g_t)/T(t)}$.

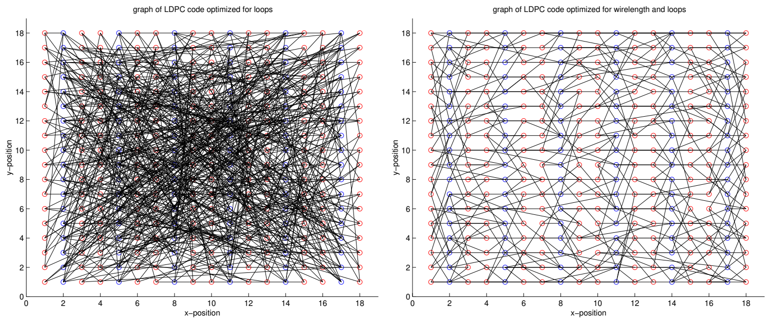
The algorithm terminates at a pre-determined time, and the output of the algorithm is the graph g_{final}. Roughly, the reason for a non-zero temperature that sometimes allows the higher-energy solution to be selected is to avoid local (but not global) minima.

Because there is flexibility in selecting the coefficients C_L and C_W (in fact, only the ratio matters), we can generate a locus of points achievable by the algorithm in terms of $L(g, \alpha)$ and $W(g)$. In the following chart, we plot $L(g, \alpha)/L_0$ vs. $W(g)/n$, where L_0 is the smallest $L(g)$ obtained by the algorithm.



Locus of points obtainable by SA

The chart shows that average wirelength can be reduced substantially with extremely little increase in loopiness. However, at very short wire-lengths, the graph must inevitably admit shorter loops. The following two layouts show the extrema of the previous chart.



graphs optimized primarily for loopiness (left) and wire-length (right)

4 Evaluation of Performance Measure

In this section, we will give comparisons of the performance achieved by various codes in order to verify (or refute) that our loopiness measure predicts the performance of actual codes. The results of Mao [5] give strong preliminary indication that such a prediciton property will hold.

5 Conclusion

We have demonstrated an application of the Simulated Annealing algorithm to the problem of designing graphs for LDPC's which can substantially reduce the loopiness of a graph. We have preliminary evidence that this will improve the performance of a code under message-passing decoding.

Additionally, we have shown that the same algorithm allows a very favorable tradeoff in terms of the wiring complexity of the direct implementation. Further decoding simulations will quantify the extent to which loopiness is related to codec performance.

References

[1] R. Gallager "Low Density Parity Check Codes," Number 21 in Research Monograph Series. MIT Press, Cambridge, Mass., 1963

[2] T. Richardson, R. Urbanke "The Capacity of Low-Density Parity Check Codes under Message-passing Decoding" Submitted to *IEEE Trans. Inform. Theory, 1998.*

[3] T. Richardson, R. Urbanke "The Design of Provably Good Low Density Parity Check Codes" Submitted to IEEE Trans. Inform. Theory, 1999.

[4] D. MacKay "Good Error Correcting Codes based on Very Sparse Matrices" IEEE Trans. Inform. Theory, 1999.

[5] Y. Mao, A. Banihashemi "A Heuristic Search for Good Low-Density Parity-Cehck Codes at Short Block Lengths" Accepted for presentation at IEEE ICC 2001.

[6] S. Chung "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit" To appear in IEEE Communications Letters 2000.