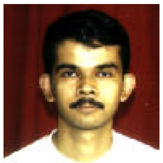
Decoding Codes on Graphs

2. Probabilistic Decoding

A S Madhu and Aditya Nori



A S Madhu

Aditya Nori

A S Madhu and Aditya Nori are graduate students with the Department of Computer Science and Automation, IISc. Their research addresses various aspects of algebraic and combinatorial coding theory.

1.Introduction

Iterative decoding is a widely used technique in modern communications systems. The Low Density Parity Check (LDPC) codes of Gallager were the earliest codes for which iterative decoding techniques were defined. These codes along with a simple iterative decoding algorithm were introduced in the first part of this article. Here we describe a probabilistic iterative decoding scheme, also proposed by Gallager, and show how many modern iterative decoding algorithms are derived from his basic scheme.

2. LDPC Codes

LDPC codes are codes which have sparse parity check matrices. For a linear code where each codeword has n symbols, of which k are message symbols, a parity check matrix is an $(n-k) \times n$ matrix of linearly independent rows. Each row of the matrix represents an independent constraint that the symbols of the codeword must satisfy. For example, if a parity check matrix for a binary code of length 5 with 3 message symbols has a row 11100, it represents the constraint that the first three bits of every codeword should sum to 0 modulo 2. A binary parity check matrix is said to be sparse if the number of ones is small in comparison to the number of zeros in the matrix. One can achieve sparseness by restricting each row and column of the matrix to have a constant number of ones. An (n,c,r) LDPC code is a binary linear code which has a sparse parity check matrix with c ones in each column and r ones in each row. For a (c,r) sparse parity check matrix, each variable partic-

[1] Low Density Parity Check Codes, *Resonance*, Vol.8, No.9, pp.49-59, 2003

Keywords
Low density parity, check codes, maximum a posteriori probability decoding.

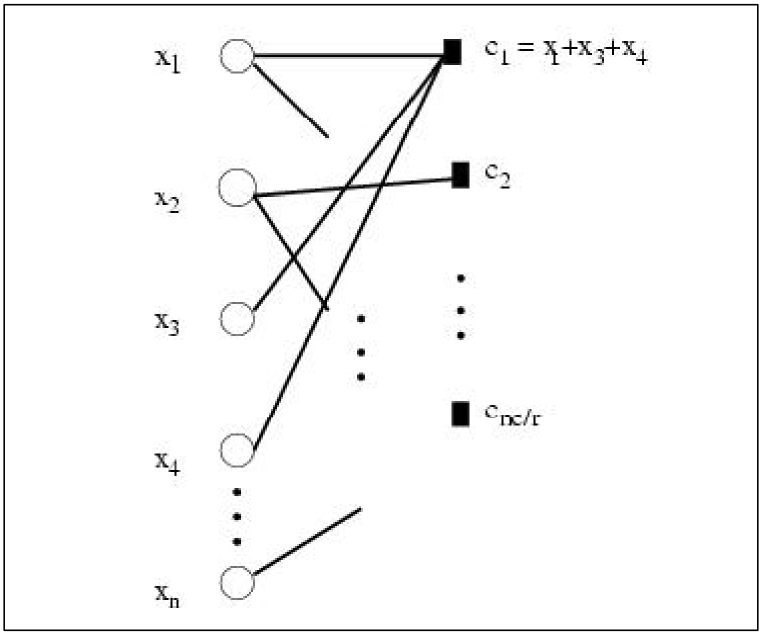


Figure 1. A bipartite graph representing an LDPC code.

ipates in c parity check equations, while each equation involves exactly r variables. LDPC codes can be viewed as bipartite graphs where one set of nodes represents codeword or variable nodes and the other set represents constraint nodes. For binary codes the requirement imposed by each constraint is that the value of neighbours of each constraint node must sum to 0 modulo 2. A typical bipartite graph for an LDPC code is shown in Figure 1.

These simple codes can be decoded very efficiently. We will now describe Gallager's algorithm for probabilistically decoding LDPC codes and carry out a formal analysis.

LDPC codes can be viewed as bipartite graphs where one set of nodes represents codeword or variable nodes and the other set represents constraint nodes.

3. Decoding LDPC Codes Probabilistically

In his doctoral thesis Gallager introduced two low complexity decoding operations for his low-density parity-check codes. He also analysed the performance of his codes for a special class of low-density parity-check codes { those corresponding to graphs with high girth, where the girth of a graph G is the length of a shortest cycle in G. The previous article described his `hard decision' decoding algorithm where for binary codes, the entities involved in the computation are the digits 0 and 1. It is often the case that we have probabilities available for received symbol values. In such cases it is profitable to use `soft-decision' decoding algorithms so that we can exploit the fact that the reliabilities of received symbols are available.

It is often the case that we have probabilities available for received symbol values. In such cases it is profitable to use ‘soft-decision’ decoding algorithms so that we can exploit the fact that the reliabilities of received symbols are available.

We first define the model of the channel used for communication. We assume that the channel is used to transmit binary information and denote by $p(a|b)$, the probability that an $a \in \{0,1\}$ was received at the channel output, given that $b \in \{0,1\}$ was its input. Thus, if $a = b$, $p(a|b)$ represents the probability of error-free transmission. On the other hand if $a \neq b$, $p(a|b)$ would be the probability of a bit being flipped (possibly due to channel noise). These probabilities constitute the channel model. Consider a codeword c consisting of n symbols over the channel input alphabet, that is transmitted over this channel, and let y be the n symbol output of the channel. The objective of probabilistic decoding is to determine a word $x = (x_1, \ldots, x_n)$ such that the following function is maximized,

$$p(x_i|y); \; 1 \leq i \leq n.$$

This kind of decoding strategy is called the maximum a posteriori probability (MAP) decoding strategy as it attempts to estimate each symbol of the codeword that was transmitted, given the received vector. We will use a simple result based on properties of generating functions, for our analysis. We will first give this result before describing the decoding algorithm.

Consider a bit sequence of length m with the i^{th} bit having a probability p_i of being equal to a 1. Let us assume that the bits are independently distributed (that is, for every $j \neq k$, the probability of the j^{th} bit being equal to a $0/1$ has no bearing on the probability of the

k^{th} bit being equal to a 0/1). For such a sequence, the probability that the m bit sequence contains an even number of ones is computed in the following manner. The coefficient of x^i in the expression

$$\prod_{i=1}^{m} ((1 - p_i) + p_i x)$$

is the probability that the length m sequence contains i ones. The expression

$$\prod_{i=1}^{m} ((1 - p_i) - p_i x)$$

differs from the previous one only in the sign of the coefficients of the odd terms. Therefore, the expression

$$\frac{\prod_{i=1}^{m} ((1 - p_i) + p_i x) + \prod_{i=1}^{m} ((1 - p_i) - p_i x)}{2}$$

contains only terms of even degree, and the coefficient of x^i in this expression denotes the probability that the sequence has i ones. Thus the probability that the m bit sequence contains an even number of ones is given by

$$\frac{1 + \prod_{i=1}^{m} (1 - 2p_i)}{2}$$

and the probability that the sequence has an odd number of ones is given by

$$\frac{1 - \prod_{i=1}^{m} (1 - 2p_i)}{2}.$$

Now, suppose we restrict ourselves to sequences of m bits (call this set S) whose individual bits add up to zero modulo 2. Therefore $S = \{(x_1, \ldots, x_m) \in \{0, 1\}^m : \sum_{i=1}^{m} x_i = 0\}$. Given a randomly chosen sequence $x \in S$, is it possible to determine the probability that a particular bit, say the d^{th} bit x_d, is a one? We denote this conditional probability by $p(x_d = 1|M)$, where M denotes the event $\{x \in S\}$. Using Bayes rule, we have

$$p(x_d = 1|M) = \frac{p(M|x_d = 1)p_d}{p(M)}$$

This means that in order to compute $p(x_d = 1|M)$, we first compute the probability that $x \in S$ conditioned on the event that the d^{th} bit x_d, is a one, that is, $p(M|x_d = 1)$. This can be computed by calculating the probability that the remaining $m - 1$ bits contain an odd number of ones. From our previous discussion, we know how to compute this probability and this is given by

$$\frac{1 - \prod_{1 \leq i \leq m;\, i \neq d}(1 - 2p_i)}{2}. \qquad (1)$$

In the context of binary error-correcting codes, specifically linear block codes, a parity check equation consisting of *m* variables is satisfied if and only if these variables add up to a zero modulo 2, or in other words it has an even number of ones.

In the context of binary error-correcting codes, specifically linear block codes, a parity check equation consisting of m variables is satisfied if and only if these variables add up to a zero modulo 2, or in other words it has an even number of ones. Consider an $(n; c; r)$ LDPC code C; we know that each variable representing codeword components participates in exactly c parity checks. Now assume that each of these checks can be independently satisfied. Also assume that we have as channel output, the word r. Then the probability $p(x_d = 1|r)$ can be easily computed. Denote by $p_i(x_d = 1|r)$, the probability that $x_d = 1$ and satisfies the i^{th} parity check equation. By the earlier independence assumption, we have

$$p(x_d = 1|r) = \prod_{i=1}^{c} p_i(x_d = 1|r).$$

Now each $p_i(x_d = 1|r)$; $1 \leq i \leq c$ can be computed as follows. For the sake of clarity, let us focus on the parity check equation $i = 1$. If x_d is to have a value one, then the rest of the $r - 1$ variables participating in parity check equation (1) must contain an odd number of ones. Let us denote these $r - 1$ variables as a sequence $(y_1; \ldots; y_m)$

of length $m = r - 1$. Let p_i be the probability that y_i is equal to a one. This is the probability that the i^{th} transmitted bit is equal to one conditioned on the value of the i^{th} received bit, and can be easily computed as follows.

$$p(x_i|r_i) = \frac{p(r_i|x_i)p(x_i)}{\sum_{x_i \in \{0,1\}} p(r_i|x_i)p(x_i)}$$

where $x_i; r_i$ are the i^{th} components of the transmitted and received words respectively. $p(r_i|x_i)$ is a property of the channel in question, and $p(x_i) = \frac{1}{2}$, since we assume that codeword components are each equiprobably distributed. Therefore, we can now appeal to (1) to compute $p_i(x_d = 1|r)$, which in turn, can be used to compute the desired value of $p(x_d = 1|r)$. The above observations can be formally stated in the form of a theorem due to Gallager.

Theorem 1. Let p_d be the probability that the transmitted bit x_d in position d is a 1 conditional on the received bit, and let p_{il} be the same probability for the l^{th} bit in the i^{th} parity check equation in which x_d participates. Let the bits be independent of each other and let S be the event that the transmitted bits satisfy j parity check constraints on d. Let Y denote the set of received symbols. Then

$$\frac{P[x_d = 0|\{Y; S\}]}{P[x_d = 1|\{Y; S\}]} = \frac{1 - p_d}{p_d} \prod_{l=1}^{j} \frac{1 + \prod_{l=1}^{k-1}(1 - 2p_{il})}{1 - \prod_{l=1}^{k-1}(1 - 2p_{il})}.$$

4. Probabilistic Decoding

Probabilistic decoding is an iterative decoding technique used to decode codes efficiently. Consider an $(c; r)$ LDPC code C as described in Section 2. This code can be represented diagrammatically in the form of a parity check tree rooted at some variable (recall that a variable represents a codeword component), say x_d, as shown in Figure 2. Edges in the tree represent parity check constraints.

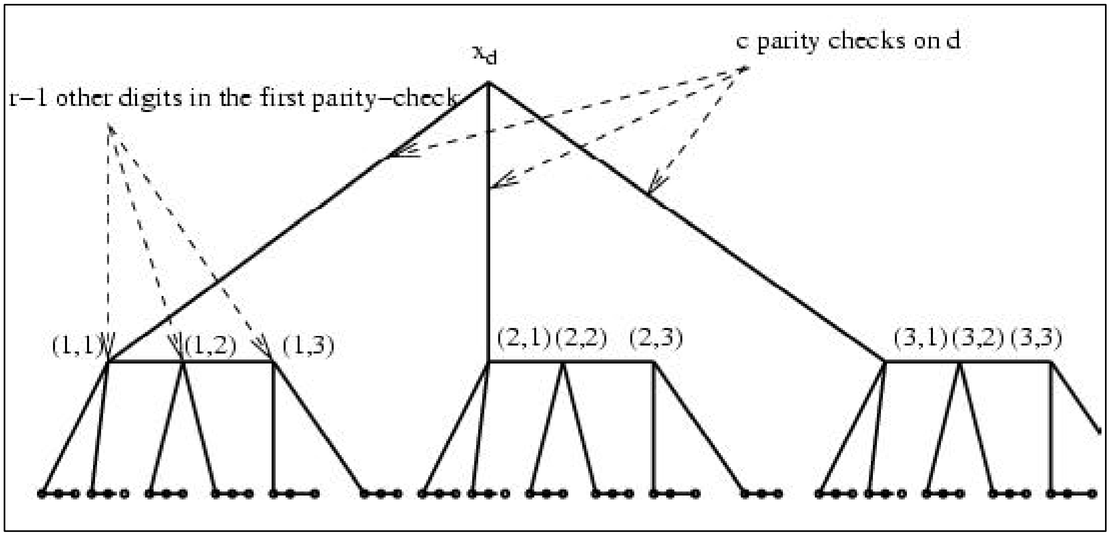


Figure 2. Parity check set tree for an (n,j,k) LDPC code.

For example, there is an edge e from the root of the tree (representing variable x_d) to a node u at depth 1 (assuming that the root is at depth 0) if and only if x_d participates in the parity check equation represented by e. The other variables which take part in the parity check equation represented by e are associated with u, which is essentially a 'supernode'. As an example, in Figure 2, consider the first (leftmost) edge from the root. This edge represents a parity check equation involving variables x_d (the root) and variables $(1;1)$; $(1;2)$; $(1;3)$ (the supernode). This structure is repeated for subnodes (variables) in each supernode, thus resulting in the parity check tree shown in Figure 2.

We will now describe the probabilistic algorithm by illustrating how the variable x_d is decoded. As mentioned earlier, this algorithm runs iteratively. To start with, in the first iteration, only bits in the first level of the parity check set tree are involved in decoding x_d. Recall that these bits correspond to the variables that are engaged in the c parity checks that x_d participates in. Thus we are limiting our attention to the c constraints that x_d is participating in. We are interested in computing the likelihood ratio (LR) of x_d, defined by $\frac{p(x_d=0|y)}{p(x_d=1|y)}$, y being the received word. The computation of the LR for x_d where the probabilities are conditioned on only the received bits at level one of the parity check tree then follows directly from Gallager's Theorem 1. Note that this is an approximation to what is required (as in the exact solution, the probabilities are conditioned on all the digits of the received word), where the degree of approximation depends on the number of iterations of the algorithm. In the second iteration we use the received bits in the first two levels of the tree for decoding x_d. As a consequence, bits that are not directly involved in the parity check equation (at level one) for x_d are also used to decode x_d. Here we first consider the bits in the first level of the tree, and examine the $c - 1$ parity checks connected to each bit, excluding the parity check involving x_d. Using these $c - 1$ constraints, we compute the probability that the bit in the first level of the tree is a one conditioned on the received bits in the second level of the tree. We now use these new probabilities for bits in the first level of the tree and recompute the probability the LR for x_d. The subsequent iterations of the decoding algorithm proceed in this inductive manner until the LR gives a definite indication as to what the bit x_d should be decoded to. In other words, LR $\gg 1$ implies that x_d is most likely a 0, otherwise, $x_d = 1$.

Now that we know how to decode a single bit, we can repeat this procedure for other bits, thus decoding the whole codeword. But this would result in a lot of repeated computations. An efficient procedure to achieve the same, could be formulated as follows. For each bit, say x we compute a set of probabilities, where each probability is conditioned on the received digits involved in any $c - 1$ parity checks (note that x must be a part of these parity checks). It is easy to see that this set of probabilities has cardinality c. To compute the likelihood of an arbitrary variable x, the probability we use is the one obtained by omitting the parity check involving x. The correctness of the decoding procedure depends on the assumption that the parity check constraints can be satisfied independent of each other. This assumption does not hold if the graph representing the code has cycles. In spite of this limitation, this decoding procedure works extremely well in practice. Like the hard-decision decoding algorithm, the soft-decision decoding algorithm just described can be implemented as a message passing decoder.

Belief propagation is an exact algorithm invented by Judea Pearl that solves probabilistic reasoning problems of singly connected networks, that is, those without cycles.

5. Later Work

The algorithm just described is an instance of a belief propagation algorithm. Belief propagation is an exact algorithm invented by Judea Pearl that solves probabilistic reasoning problems of singly connected networks, that is, those without cycles. By an exact algorithm we mean one that finds exact solutions as opposed to approximate solutions. Here what we want to measure at each node is the likelihood of a variable being a zero or a one. Once we have computed the beliefs at all variable nodes we can make a decision on each bit and thus decode the code. Informally, the message passing scheme proceeds as follows. Every node sends a probability vector to each of its neighbours. Suppose Y has two neighbours X and Z. The node X sends a message to Y which, roughly speaking, is the information X knows about the state of Y but Y does not. The node Y combines this information with its own information and sends a message to Z with the information that X and Y know but which Z does not. Similarly Y takes the message from Z and after adding its own information sends a message to X. This procedure is carried out for all nodes in parallel and if there are no cycles the messages reach a steady state. The belief at Y is then obtained by combining the steady state messages from X and Z and the local evidence at Y. The need for having a cycle free graph is so that evidence is not multiply counted after going around a cycle. While using Gallager's algorithm if the number of message passing rounds is small, there will be no multiple copies of the same node in the unwrapped graph and we may consider the bipartite graph as a tree. However this is not the case when there are a large number of decoding rounds. To make the analysis simple, Gallager gave explicit constructions of graphs whose girth logarithmically increased with the number of variable nodes. Since all bipartite graphs for Gallager codes have fixed degree nodes and the expected girth of a random graph of fixed degree increases logarithmically in the number of nodes in the graph, a tree-like neighbourhood assumption is true with high probability with increasing number of nodes, thus ensuring that Gallager's algorithm gives the correct answer with high probability.

Since all bipartite graphs for Gallager codes have fixed degree nodes and the expected girth of a random graph of fixed degree increases logarithmically in the number of nodes in the graph, a tree-like neighbourhood assumption is true with high probability with increasing number of nodes, thus ensuring that Gallager's algorithm gives the correct answer with high probability.

Gallager codes were generalised by Sipser and Spielman in 1996 to expander codes represented by expander graphs. An expander graph is a graph in which every set of vertices has an unusually large number of neighbours. Their construction used the fact that there exist graphs that expand by a constant factor but have only a linear number of edges. They showed that they could obtain asymptotically good, linear codes that had linear time decoding algorithms. Generalizations were also made in 2001, by Luby, Mitzenmacher, Shokrollahi and Spielman, to randomly construct irregular graphs, that is, those where the degrees of variable or constraint nodes were not constant. The error correcting capability for irregular graphs was shown to be strictly greater than that possible using codes with regular bipartite graphs. In 2001 Richardson and Urbanke showed that one could correctly compute to arbitrary precision, the average probability of bit error even if the graph has cycles, by just calculating the probability of error when the graph does not have cycles. Thus an accurate analysis of the performance of such codes is possible even in the presence of cycles. In fact it was shown in 2001 by Richardson, Shokrollahi and Urbanke that one could come very close to the Shannon limit of the channel using irregular low-density parity check codes.

We thus see that low-density parity-check codes, invented by Robert Gallager forty years ago have been shown to have remarkable potential for error correction using relatively simple decoding schemes. The construction and analysis of such codes is an impressive theoretical achievement. On the practical side they have immense potential in applications that require the use of long codes. The Tornado Codes described in the first part of this article are one example of low-density parity-check codes that correct erasures and are used to mitigate the effects of packet loss over digital networks. Undoubtedly other applications will use these codes in the years to come.

Suggested Reading

[1] A S Madhu and Aditya Nori, Decoding Codes on Graphs: Low Density Parity Check Codes, *Resonance*, Vol. 8, No. 9, pp.49-59, 2003.

[2] Robert G Gallager, *Low Density Parity Check Codes*, Ph.D. thesis, MIT, 1963.

[3] M Sipser, D A Spielman, Expander Codes, *IEEE Transactions on Information Theory*, Vol.42, No.6, pp. 1710 -1722, 1996.

[4] Stephen B Wicker, Saejoon Kim, *Fundamentals of Codes, Graphs, and Iterative Decoding*, Kluwer International Series in Engineering and Computer Science, 2003.

[5] T J Richardson, M A Shokrollahi, R L Urbanke, Design of Capacity Approaching Low-Density-Parity-Check Codes, *IEEE Transactions on Information Theory*, Vol. 47, No.2, pp. 619-637, 2001.

[6] T J Richardson, R L Urbanke, The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding, *IEEE Transactions on Information Theory*, Vol.47, No.2, pp. 599-618, 2001.

[7] M G Luby, M Mitzenmacher, M A Shokrollahi and D A Spielman, Improved Low-Density Parity-Check Codes Using Irregular Graphs, *IEEE Transactions on Information Theory*, Vol.47, No.2, pp 585-598, 2001.

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

Address for Correspondence
A S Madhu and Aditya Nori
Department of Computer
Science and Automation
Indian Institute of Science
Bangalore 560012, India.
Email: madhu@csa.iisc.ernet.in
aditya@csa.iisc.ernet.in