Concentration Theorem for Tripartite LDPC Codes

Peyman Meshkat   Hamid Jafarkhani

Center for Pervasive Communications and Computing

Department of Elec. Eng. & Computer Science

University of California, Irvine

Irvine, CA 92697

`pmeshkat@uci.edu  hamidj@uci.edu`

Abstract

We present the framework of tripartite LDPC codes. In contrast with previous works on LDPC codes, in this framework we consider the case where a channel output is affected by more than a single coded bit. Tripartite LPDC framework is general enough to cover the cases of M-ary modulation and MIMO channels. We show that concentration theorem which was previously proved for the case of binary erasure channel by Luby *et al* and later was generalized to binary-input memory-less channels by Richardson *et. al.* holds in our more general framework as well. Concentration Theorem is essential in designing Irregular LDPC codes and calculating the capacity of message passing decoding. We also present the specific equations for calculating the messages used in decoding algorithm for M-ary modulations and MIMO channels.

Index Terms

LDPC codes, Tripartite LDPC, Concentration theorem, M-ary modulation, Space-time codes, MIMO.

I. Introduction

Irregular LDPC codes are the best known codes (in terms of coding gain and bit error probability) for simple modulation schemes and binary input memory-less channels such as Binary Input Additive White Gaussian Noise (BIAWGN) or discrete Binary Symmetric Channels (BSC). Irregular LDPC codes are in fact modified versions of the original (regular) LDPC codes introduced by Gallager in 1961 [1]. Despite their excellent performance even in their original (regular) version, these codes "were largely neglected" up until the "phenomenal success of turbo codes" [2]. Irregular LDPC codes were introduced in [3] for binary erasure channels and later in [4] for more general binary input channels. The performance of irregular LDPC codes on binary-input memory-less AWGN channel was even better than turbo codes: less than 0.13 dB away from the ultimate theoretical limit of Shannon capacity at bit-error rate of 10^{-6} as reported in [4]. This remarkable performance has turned LDPC codes into a highly active area of research, however, little is known about the design criteria and performance of these codes for band-width efficient (M-ary) modulation schemes and space-time coding scenarios with multiple transmit and multiple receive antennas.

With the advent of turbo-codes in 1993 [5], several researchers began to explore the possibilities of designing codes and their corresponding iterative decoding algorithms based on graphical representation such as Bayesian Networks and Belief Propagation algorithm [6]. Early notable examples of these works include Wiberg [7] , Mac-Kay and McEliece [8] and Kschischang and Frey [9], [10]. This wave of research resulted in recognition of a new class of codes with the following two characteristics. First, they are represented in (sometimes equivalent) graph-based frameworks such as Bayesian networks [6] or factor graphs [10]. Second, they can be decoded by (usually iterative) message passing decoding algorithms such as belief propagation [6] or sum-product algorithm [10]. Turbo codes, serial concatenation of convolutional codes [11], LDPC codes [1], MacKay-Niel (MN) codes [12] and Read-Accumulate (RA) codes [13] belong to this class of codes. This work is another example where a graphical representation is used for defining a coding/modulation and decoding/detection framework.

LDPC codes can be represented by a bipartite graph. The graph consists of two kind of nodes: variable nodes and check nodes. Variable nodes represent bits in a codeword and check nodes represent parity check equations that should be satisfied in a valid codeword. A variable node is connected to a given check node if and only if the bit corresponding to that variable nodes takes part in the parity check equation corresponding to the given check nodes. The set of nodes connected to a given node are called the neighbors of that node.

The code (set of valid codewords) consists of all codewords for which all the parity check equations are satisfied. The iterative decoding is performed by passing "messages" in both directions between check nodes and variable nodes. From the noisy observation at the output of the channel, we can calculate an initial estimate for each coded bit. These initial estimates are the initial set of variable node to check node messages that are sent to the check nodes. We represent variable nodes to check node messages by $\psi_{v\to c}$. Once a check node receives $\psi_{v\to c}$ messages from its neighboring variable node, new updated estimates for each variable nodes can be calculated. This new estimate would be the check node to variable node message represented by $\psi_{v\to c}$. Now that we have updated estimates for each variable node from different neighboring check nodes we can again calculate new estimates for the given variable node. This would be the updated variable to check node message. This procedure can be iteratively repeated until the hard decisions on the variable nodes satisfy all the parity check equations or until a maximum number of iterations has been reached. In the former case, a codeword is declared as the decoding decision and in the latter case a decoding failure is declared.

Implicit in the above decoding algorithm, is the assumption that the different message received by a variable node or check node are functions of independent random variables. To elaborate more, consider a message from a check node to a variable node in iteration l. This message is a function of all messages from all other variable nodes connected to that check nodes. These latter messages are in turn functions of messages from all other check nodes connected to those variable nodes and so on. We can construct a graph in the form of an expanding tree which represents the dependencies between the messages sent during the lth iteration of message passing decoding algorithm. This graph is known as a decoding neighborhood [2]. The nodes in a decoding neighborhood are those nodes whose sent messages are used in calculating the message that is sent on the top link of the decoding neighborhood graph. A decoding neighborhood is tree-like or cycle-free when there is no repeated node in the the neighborhood. As long as the neighborhood is cycle-free the implicit independence assumption about the messages is a correct assumption.

In the original (regular) version of LDPC codes each variable node is connected to a fixed number of check nodes. This fixed number is the degree of variable nodes represented by d_v. Similarly, each check node is connected to a fixed number of variable nodes and this fixed number is the degree of check nodes represented by d_c. Authors of [3] showed that by relaxing this fixed d_v and d_c assumption one can design LDPC codes with superior coding performance. These codes were called *irregular* LDPC codes as opposed to the classic regular case where the degree of variable nodes and check nodes are fixed. Irregular codes were first designed for binary erasure channel in [3] and were generalized to the case of an arbitrary binary input symmetric output channel in [4]. Irregular LDPC codes were designed by finding a probability distribution on the degree of variable nodes and a probability distribution on the degree of check nodes which optimize the average performance according to that distribution. An important question arises at this point. Assume we have a probability distribution that optimizes the average performance of the codes selected from that distribution. What is the guarantee that an individual code which is randomly selected according to that given optimized distribution has a decoding behavior close to the average? The answer is given by an important theorem known as "Concentration Theorem" [2].

Concentration theorem states that for a large enough block size n, the performance of a randomly chosen LDPC code is close to the expected value of the decoding behavior of all codes in the ensemble with a probability that approaches to unity exponentially fast in n. Furthermore, this expected value converges to the expected value of decoding behavior under the assumption that the corresponding decoding neighborhood is tree-like. The importance of this theorem is that it allows us to find the code parameters (degree distribution on check nodes and variable nodes) that optimize the average performance of the codes in ensemble and be “almost sure” that this design criterion is meaningful for a randomly chosen sample from the ensemble of LDPC codes. The validity of the design method of “density evolution” [4] which led to the design of phenomenally successful irregular LDPC codes for binary-input channels is based on the validity of the concentration theorem [2], [4].

An essential assumption in previous works in design of irregular LDPC codes is the assumption of a binary-input channel. This assumption implies that each noisy value received at the output of the channels is connected to only a single variable node. It is of great interest to design irregular LDPC codes for the case of M-ary input and MIMO channels, where the above mentioned assumption does not hold.

In this work, we introduce the framework of tripartite LDPC codes. In this framework a noisy received value from the output of the channel can be connected to multiple variable nodes in the graph. We recognize the output of the channel as a third set of nodes in the graph of the code. We refer to these nodes as *symbol nodes*. The graph representing our framework is a tripartite graph rather than a bipartite graph and hence the name *tripartite* LDPC framework. Our framework is general enough to cover LDPC codes used with M-ary modulations and LDPC codes used over multiple-input and multiple-output (MIMO) channels.

Concentration theorem, which was stated and proved for the case of Binary Erasure Channel (BEC) in [3] and later for the more general case of binary-input symmetric-output channels in [2], is a necessary stepping stone for validity of the design irregular LDPC codes based on finding degree distributions on variable nodes and check nodes. In this paper we state and prove the concentration theorem for the tripartite framework.

The organization of the paper is given in the following. In Section II we introduce the framework of tripartite LDPC code and explain our message passing decoding algorithm for the case of tripartite graph. In Section III we state and prove the concentration theorem for the tripartite framework. Tripartite framework can cover M-ary modulation and MIMO channels as special cases. In Section IV we show that how M-ary modulation and MIMO channel transmission scenario can be considered as special cases in this framework and we derive the message passing decoding algorithms for these special cases. In Section V we provide a summary and mention some possible areas for further research.

II. TRIPARTITE LDPC FRAMEWORK

In this section we introduce the framework of tripartite LDPC codes. This framework is a unifying structure for LDPC codes used in scenarios involving channel models and modulation schemes that are more general than just binary modulation over a memoryless single-input and single-output channel.

As we mentioned before, the standard decoding for an LDPC code consists of passing messages between a set of variable nodes and a set of check nodes. These two sets of nodes constitute a bipartite graph corresponding to a given LDPC code. Strictly speaking, however, there exist a third set of nodes which corresponds to the (noisy) received information from the channel. For the case of a binary-input memory-less channel each received output from the channel corresponds to a single transmitted (coded) bit. This means that in the graph, the nodes corresponding to channel outputs are of degree one (*i.e.* connected to only one neighboring node). In the message passing algorithms such as belief propagation [6] or sum-product algorithm [10], a node with degree one sends messages only once during the initialization. This is due to the fact that in these algorithms, a message that a node A sends to a neighboring node B is a function of the information that node A has received from all its neighboring nodes *excluding* node B. This concept in turbo code literature is known an "extrinsic information" principal. As a result of this, a message transmitted from a node with degree one remains the same and will not be updated in each iteration of the decoding algorithm. Therefore, a node with degree one takes part only in the initialization phase of the iterative decoding algorithm. The "third set of nodes", that was mentioned in the above, therefore, do not send updated messages during consecutive iterations and in effect the message passing is performed over the nodes of a *bipartite* graph.

We consider a case that transmitted symbols depend on more than one of the coded bits (variable nodes). We would like to emphasize that in our framework the LDPC code is still a binary code, however more than one coded bit determine the symbol (or group of symbols) which are transmitted over the channel. This can be extended to the case of non-binary codes as well, however, we do not consider that case in this paper.

Figure 1 shows two graphs. The left graph corresponds to a "classic" LDPC code setup with a binary modulation on a memory-less channel. For the reasons explained in the above discussion, we refer to this case as *b*ipartite LDPC code. The right graph corresponds to our configuration of tripartite LDPC codes. The white circles correspond to the variable nodes and squares correspond to check nodes. The black circles correspond to the output of the channel. In the following discussion we sometimes call a node which corresponds to a variable a *v-node*, a node corresponding to a check node, a *c-node* and finally a node corresponding to a symbol node an *s-node*. For the case of a binary modulation each s-node is connected to a single v-ndoe. This is in contrast with the tripartite case (right graph) where each s-node is connected to several v-nodes. This is a crucial difference which results in a more sophisticated decoding algorithm and a more sophisticated decoding (tree-like) neighborhood.

The iterative decoding in a tripartite setting involves a more diverse set of messages compared to the bipartite counterpart. In the "classic" or bipartite case, the message passing algorithm for decoding consists of sending two types of messages: variable-node to check-node messages and check-node to variable-node message. In the tripartite case we have two more types of messages: symbol-node to variable-node and variable-node to symbol-node messages as well. We use the notation $\psi_{s\to v}$ for s-node to v-node messages, $\psi_{v\to c}$ for v-node to c-node messages, $\psi_{c\to v}$ for c-node to v-node messages and $\psi_{v\to s}$ for v-node to s-node messages.

The iterative message passing algorithm that we propose for tripartite LDPC code starts with an initialization phase. The initialization includes sending messages from symbol nodes to variable nodes and initializing $\psi_{c\to v}$ messages to a equal probabilities for the receiving v-node being a "1" or a "0".

At the beginning of each iteration from the previous $\psi_{s\to v}$ and $\psi_{c\to v}$ (*i.e.* the messages received by v-nodes) we calculate a hard decision on v-nodes. If these hard decisions satisfy all parity check equations the current decisions are declared as the output of the decoder. Otherwise, first we check that maximum number of iterations is not reached. If this is the case the decoding stops and decoding failure is declared. Next, v-node to c-node $\psi_{v\to c}$ messages are calculated and sent to c-nodes and then c-nodes calculate the $\psi_{v\to c}$ messages in return. At this point the tripartite decoding would be different from bipartite decoding. In the bipartite case, the messages from symbol nodes are not updated anymore. In the tripartite case, we update the information that we obtain from symbols by first sending our new estimates of the variable node to the symbol nodes. Next, a given variable node connected to a symbol node receives a new estimate from the symbol node based on the updated estimates for the rest of the variable nodes connected to that symbol node. Note that for the case of bipartite LDPC codes "the rest of the variable nodes connected to that symbol node" is an empty set. Therefore, in the bipartite case the s-node to v-node messages remain constant. The ending of the current iteration is marked by sending variable to check messages.

A more formal description of the tripartite decoding algorithm is given in the following:

Tripartite Decoding Algorithm

```
{0} Initialization:
    Calculate $\psi_{s\to v}$
    Initialize $\psi_{c\to v}$.
    Set the number of iterations to zero.
{1} (Start of Iteration:)
    Make a hard decision on v-nodes.
{2} If the hard decision satisfies parity check equations declare the codeword, STOP.
{3} Increase the number of iterations.
{4} If the maximum number of iteration is reached, declare decoding failure, STOP.
{5} Calculate $\psi_{v\to c}$.
{6} Calculate $\psi_{c\to v}$.
{7} Calculate $\psi_{v\to s}$.
{8} Calculate $\psi_{s\to v}$.
{9} Goto {1}.
```

The above described message passing algorithm results in a decoding neighborhood shown in Figure 2. Each new iteration adds four more tiers to the decoding neighborhood graph. As we explained in the introduction, the nodes in a decoding neighborhood are those nodes whose sent messages are used in calculating the message that is sent on the top link of the decoding neighborhood graph. As we see in the next section, this decoding neighborhood is used in proving the concentration theorem for tripartite framework.

III. Concentration Theorem for Tripartite Framework

In this Section we state and prove the "Concentration Theorem" for the case of tripartite LDPC codes. This theorem was first stated and proved in [3] for the case of a binary erasure channel and was generalized to the case of an arbitrary binary-input channel in [2]. Here we further generalize the theorem to the tripartite LDPC framework.

In order to prove the concentration theorem, we should find an upper bound on the probability of the event that a given decoding neighborhood contains cycles. In the following, d_v represents the degree of variable nodes, d_c represents the degree of check nodes and d_s the degree of symbol nodes. For the decoding neighborhood given in Figure 2 we use C_l for the number of check nodes till iteration l and M_l for the number of variable nodes in the decoding neighborhood till iteration l. We define the random variable Z to be the number of incorrect messages among all nd_v variable-to-check messages sent out in the lth iteration. Note that although we use a notation almost identical to that of [2] in stating and proving the concentration theorem, the variables and parameters should be interpreted in the context of the framework that we have established for the case of a tripartite LDPC code. For example in the case of bipartite LDPC, the random variable Z defined in the above corresponds to the number of incorrect messages on the top most edge of a decoding neighborhood graph with depth equal to $2l + 1$ (see [2]). As discussed in Section II, the corresponding graph for the case of tripartite would be $4l + 2$. We define p to be the expected value of incorrect messages passed on lth iteration of decoding on a variable-to-check edge given that the corresponding decoding neighborhood of that edge is tree-like. Using the above the definition we state the "concentration theorem" for tripartite LDPC codes:

Concentration Theorem for tripartite LDPC codes: Over the probability space of all graphs $C^n(d_v, d_c, d_s)$ and channel realizations let Z be the number of incorrect messages passed at iteration l. Let p be the expected number of incorrect messages passed along an edge whose unique neighboring variable-to-symbol edge has a decoding neighborhood of depth $4\ l$ at iteration l. Then, there exist positive constants $\beta = \beta(d_v, d_c, d_s, l)$ and $\gamma = \gamma(d_v, d_c, d_s, l)$ such that:

- **Concentration Around Expected Value:** For any $\epsilon > 0$ we have:

$$Pr\{|Z - E[Z]| > nd_v\epsilon/2\} \leq 2e^{-\beta\epsilon^2 n}. \tag{1}$$

- **Convergence to Cycle-Free Case:** For any $\epsilon > 0$ and $n > \frac{2\gamma}{\epsilon}$ we have:

$$|E[Z] - nd_v p| < nd_v\epsilon/2. \tag{2}$$

- **Concentration Around Cycle-Free Case:** For any $\epsilon > 0$ and $n > \frac{2\gamma}{\epsilon}$ we have

$$Pr\{|Z - nd_v p| > nd_v\epsilon \leq 2e^{-\beta\epsilon^2 n}\}. \tag{3}$$

Proof: The statement of the theorem above is almost a verbatim copy of the Theorem 2 of [2]. The only distinction is that the code ensemble C^n and the constants γ and subsequently β are functions of d_s as well as d_c and d_v. As mentioned before the parameter d_s represents the degree of the s-nodes. For the case of the binary input channel, we have $d_s = 1$. Therefore, the expressions that we find for γ and β can be considered as generalized versions of those corresponding to the binary input channel.

Similar to the case of binary input channel, (3) follows from (1) and (2). To see this, assume we have shown that (2) holds. Now if the complement of the event in (1) has happened, by triangular inequality, one can deduce that the complement of event in (3) has happened as well. Therefore the event in (3) is a subset of the event in (1). Thus, the probability of the event in (3) is less than the probability of the event in (1) which means that (3) follows from (1) and (2).

We start by proving (2). The first step would be proving that there exists a constant $\gamma(d_v, d_c, d_s, l)$ so that the probability that the decoding neighborhood of a given variable to check edge is not tree-like (i.e. it contains cycles) is upper-bounded by $\frac{\gamma}{n}$. This step of the proof is the part where our proof for the general case of tripartite LDPC is different from the special case of bipartite LDPC. Once this part is established, the rest of the proof follows similar to the case of the bipartite LDPC.

For the case of tripartite LDPC, the decoding neighborhood at iteration l has depth $4l + 2$. Given the enumeration of the tiers of the graph in Figure 2, one can observe that the tiers $4k+1$ with $k = 0, 1, \ldots$ consist of variable nodes, the tiers $4k + 2$ consist of check nodes, the tiers $4k + 3$ again consist of variable nodes and tiers $4k + 4$ consist of symbol nodes. We say two v-nodes are s-neighbors of each other if and only if they are connected to the same s-node. Note that each v-node is connected to one and only one s-node but not the other way around.

Performing each new iteration adds 4 more tiers to the graph. We want to prove that for a fixed l^* the probability that the corresponding decoding neighborhood of depth $4l^*$ is not tree-like is upper-bounded by $\frac{\gamma}{n}$. First we assume that for some $l < l^*$ the tiers corresponding to iteration l have not produced any loops. This means that all the nodes in tiers $1, 2, \ldots, 4l$ are distinct. Furthermore, from the construction of the graph, no two v-nodes in these tires can be s-neighbors; otherwise, their common neighboring s-node would have to be appeared twice in the graph. We ask the question that what would be the probability that the decoding neighborhood that is constructed due to performing one more iteration will not cause any loops. Performing one more iteration corresponds to adding four more tiers to our graph. The last tier from previous iteration (*i.e.* the tier $4l$) is a row of s-nodes. The next tier (*i.e.* tier $4l+1$), would be a tier of v-nodes. This tier would be a deterministic function of the previous tiers. Given that the previous tiers have not caused any loops, this tier would not create a loop either. This is true because all the v-nodes that appeared so far in the previous tiers, have their unique neighboring s-node appeared as well. Therefore a v-node of the newly added tire $4l+1$ cannot be a repeated v-node. Otherwise, its neighboring s-node would also be a repeated node which contradicts the assumption that the nodes till tier $4l$ do not contain any cycles. Now we will consider adding the next tier of c-nodes. Assume that k c-nodes from the tier $4l+2$ have been added without creating any loop. Given that assumption, the probability that a newly added c-node does not create a loop would be:

$$\frac{(m-C_l-k)d_c}{md_c-C_l-k} = 1-\frac{(d_c-1)(C_l+k)}{md_c-C_l-k} \geq 1-\frac{C_{l^*}}{m} \tag{4}$$

In the above expressions, m is the total number of check nodes in the graph of the code, C_l is the total number of c-nodes in the graph corresponding to the decoding neighborhood of l decoding iterations, and d_c is the degree of the c-nodes. Repeating the above argument for each of the $C_l - C_{l-1}$ c-nodes of tier $4l+2$ the probability that the nodes of this tier do not cause any cycles is lower bounded by

$$(1-\frac{C_l^*}{m})^{C_l-C_{l-1}}. \tag{5}$$

Now we add the tiers $4l+3$ and $4l+4$. At each time we add one v-node from tier $4l+3$ and simultaneously we add its unique neighboring s-node from tier $4l+4$. Again we assume k pairs of such v-nodes and s-nodes have been added and no cycles has been created so far. We ask the question that what would be the probability that adding a new pair of v-node and its neighboring s-node does not create a cycle. In order for this pair of node not to make a cycle we need the following two conditions. First, the new variable node that is chosen should not have appeared in the previous rows. Second, the new v-node should not be an s-neighbor of any previously added v-node; otherwise, their common neighboring s-node would appear twice. The number of available possible positions for adding a new edge is $nd_v - M_l - k$ and the number of acceptable positions for not creating a cycle would be $[n-d_s(M_l+k)]d_v$ therefore the probability of not creating a new cycle would be:

$$\frac{nd_v - d_v d_s(M_l + k)}{nd_v - M_l - k} = 1 - \frac{(d_v d_s - 1)(M_l + k)}{nd_v - M_l - k} \tag{6}$$

Since we have:

$$M_l + k < M_{l^*}$$

for sufficiently large n we can upper-bound the above expression by:

$$1 - \frac{d_s M_{l^*}}{n}$$

The number of v-nodes in the tier $4l + 3$ is given by:

$$M_{l+1} - (d_s - 1)M_l - M_l = M_{l+1} - d_s M_l$$

Therefore the probability that adding tiers $4l + 3$ and $4l + 4$ will not create any cycles would be lower-bounded by:

$$(1 - \frac{d_s M_{l^*}}{n})^{M_{l+1} - d_s M_l}. \tag{7}$$

Combining the lower bounds given by (5) and (7) we get the following lower bound for the probability of not causing a cycle by adding all tiers related to iteration l.

$$(1 - \frac{C_l^*}{m})^{C_l - C_{l-1}}(1 - \frac{d_s M_{l^*}}{n})^{M_{l+1} - d_s M_l}$$

Multiplying the above expression for $l = 1, 2, \dots, l^*$ we get the following lower-bound for the probability of creating a cycle after l^* iterations:

$$\begin{aligned} & (1 - \frac{C_{l^*}}{m})^{C_{l^*}}(1 - \frac{d_s M_{l^*}}{n})^{\sum_{i=1}^{i=l^*-1} M_{i+1} - M_i} \\ = \ & (1 - \frac{\frac{d_c}{d_v} C_{l^*}}{n})^{C_{l^*}}(1 - \frac{d_s M_{l^*}}{n})^{\sum_{i=1}^{i=l^*-1} M_{i+1} - M_i} \\ \geq \ & 1 - \frac{\frac{d_c}{d_v} C_{l^*}^2}{n} - \frac{d_s M_{l^*}(\sum_{i=1}^{i=l^*-1} M_{i+1} - M_i)}{n}. \end{aligned}$$

Thus, the probability that the decoding neighborhood is not tree-like is upper bounded by

$$\frac{\frac{d_c}{d_v} C_{l^*}^2 + d_s M_{l^*}(\sum_{i=1}^{i=l^*-1} M_{i+1} - M_i)}{n}.$$

The numerator of the above expression is $\gamma(d_v, d_c, d_s, l)$. For $d_s = 1$ that expressions collapses to the same expression for γ in the case of binary input symmetric output channel which is given in [2]. Now that we have established an upper bound on the probability of having cycles, the proofs of (3) and (1) follow in the same exact way as binary input channel. We will not repeat those proves here as they are given in [2]. □

IV. M-ARY MODULATION AND MIMO CHANNEL

In this section we show that cases of M-ary modulation over a memoryless single-input and single-output channel and also both binary and M-ary modulations over fading MIMO channels are special cases that can be handled in the tripartite framework. In the following two subsections we present specific equations for calculating the messages in the iterative decoding algorithm for the two cases of LDPC codes used with an M-ary modulation and LDPC codes used over a MIMO channel.

A. *M-ary Modulation*

Here the framework of tripartite LDPC code is used with an M-ary modulation scheme. We assume that we have a single-input single-output Additive White Gaussian Noise (AWGN) channel. The M-ary modulation scheme can be PSK or QAM or any other finite constellation.

The encoding part is quite straight forward. First the LDPC encoder maps blocks of k encoded bits to blocks of n coded bits. Assuming we have an M-ary modulation scheme with $M = 2^m$, each sub-block of m coded bits choose a symbol out of $M = 2^m$ possible symbols.

The decoding algorithm starts with sending symbol to variable node messages. A symbol to variable node message is a soft estimate for the value of that variable given the received value from the channel and the previous estimates for the rest of the variable nodes connected to that given symbol node. Assuming that we have an Additive White Gaussian Noise (AWGN) channel, and working in log-likelihood ratio domain the appropriate symbol to variable message would be:

$$\psi_{s\to v} = \log \frac{\sum\limits_{v'_1,v'_2,\ldots,v'_{d_s-1}\in\mathcal{V}_s\backslash\{v\}} p(v'_1)p(v'_2)\ldots p(v'_{d_s-1})e^{\|c(v'_1,v'_2,\ldots,v'_{d_s-1};V=1)-r\|^2/2\sigma^2}}{\sum\limits_{v'_1,v'_2,\ldots,v'_{d_s-1}\in\mathcal{V}_s\backslash\{v\}} p(v'_1)p(v'_2)\ldots p(v'_{d_s-1})e^{\|c(v'_1,v'_2,\ldots,v'_{d_s-1};V=0)-r\|^2/2\sigma^2}} \tag{8}$$

In the above expression, the summation is performed over all possible combinations of the values for the variable nodes $v'_1, v'_2, \ldots, v'_{d_s-1} \in \mathcal{V}_s\backslash\{v\}$ where $\mathcal{V}_s$ represents the set of v-nodes connected to an s-node s. The node v is excluded from the set as one expects from the so-called "extrinsic information" principal. The term $p(v'_i)$ represents our current estimate for the probability of the event that the variable node V'_i has taken a value $V'_i = 0$ or $V'_i = 1$ (Here the random variable V_i takes a value v_i and we have used the common short-hand convention of writing $p(v_i)$ for $p(V_i = v_i)$). The term $c(v'_1, v'_2, \ldots, v'_{d_s-1}; V = 1)$ represents the constellation point corresponding to a given combination of values for variable nodes connected to the corresponding symbol node. Finally r represents the noisy received value from the channel and d_s represents the degree of symbol nodes.

Now we re-write Equation (8) in terms of log-likelihood ratios. If we define $m_i \triangleq \log \frac{p(v'_i=1)}{p(v'_i=0)}$, we have $p(v'_i = 1) = \frac{e^{m_i}}{1+e^{m_i}}$ and $p(v'_i = 0) = \frac{1}{1+e^{m_i}}$. Therefore, we can re-write Equation (8) in terms of log likelihood ratio messages as the following:

$$\psi_{s\to v} = \log \frac{\displaystyle\sum_{v'_1,v'_2,\ldots,v'_{d_s-1}\in\mathcal{V}_s\setminus\{v\}} e^{\sum_{i=1}^{d_s-1} m_i\chi(V'_i=1)+\|c(v'_1,v'_2,\ldots,v'_{d_s-1};V=1)-r\|^2/2\sigma^2}}{\displaystyle\sum_{v'_1,v'_2,\ldots,v'_{d_s-1}\in\mathcal{V}_s\setminus\{v\}} e^{\sum_{i=1}^{d_s-1} m_i\chi(V'_i=1)+\|c(v'_1,v'_2,\ldots,v'_{d_s-1};V=0)-r\|^2/2\sigma^2}} \tag{9}$$

where $\chi(V'_i = 1)$ is a characteristic function equal to 1 if $V'_i = 1$ and equal to zero otherwise.

The variable to check node message is similar to the bipartite case with the only difference that the term which corresponds to the estimate of a variable node from the channel would be updated in each iteration. In the bipartite case that term would remain the same in all iterations. The following equation gives the variable to check message in terms of symbol to variable message and previous check to variable messages from other check nodes.

$$\psi_{v\to c} = \psi_{s\to v} + \sum_{c'\in\mathcal{C}_v\setminus\{c\}} \psi_{c'\to v} \tag{10}$$

In contrast to the bipartite case, the first term on the righthand side of the above equation is updated through consecutive iterations. Equation (10) corresponds to a product of likelihood ratios. Since the messages are written in log-likelihood ratio the product is converted to a summation.

The check node to variable node messages are exactly the same as bipartite case as given in the following equation (see *e.g.* [2] for derivation).

$$\psi_{c\to v} = \log \left(\frac{1 + \prod_{i=1}^{d_c-1} \tanh \frac{1}{2} m_i}{1 - \prod_{i=1}^{d_c-1} \tanh \frac{1}{2} m_i} \right) \tag{11}$$

where m_i's are previous variable to check messages and as before, d_c represents the degree of a check node.

Finally, variable to symbol node messages are the likelihood ratio of each variable node based on information received from all check nodes. Due to independent assumption about the messages received from different check nodes, this likelihood ratio would be the product of all likelihoods which would translate to a summation in the log domain. Thus for variable to symbol message we have:

$$\psi_{v \to s} = \sum_{c \in \mathcal{C}_v} \psi_{c \to v} \tag{12}$$

The updated value of $\psi_{v \to s}$ would be used as new m_i in Equation (9) in the next iterations. Updating Equations (9) through (12) iteratively constitute the iterative decoding algorithm. A hard decision on variable node is made at the beginning of each iteration and iterations are stopped if this hard decision satifsfies all parity check equations.

B. MIMO Channel

Capacity of the MIMO channels increases linearly with the number of transmit antennas as long as the number of receive antennas is greater than or equal to the number of transmit antennas. This surprising theoretical result published in [15] and [14] has made transmission over MIMO channels an active area of research. Celebrated space-time codes [16], [17] are examples of attempts for designing coding techniques for MIMO channels. Given the excellent performance of LDPC codes, it is of great practical and theoretical interest to investigate the design of LDPC codes for a MIMO transmission scenario.

Here we use the tripartite framework to model a transmission scenario which includes a binary LDPC code used over a MIMO channel with an M-ary modulation. In our MIMO model we consider a communication system where N_t signals are transmitted from N_t transmitters simultaneously. For example, in a wireless communication system, at each time slot t, signals $\mathcal{C}_{t,n}, n = 1, 2, \cdots, N_t$ are transmitted simultaneously from N_t transmit antennas. The signals are the inputs of a multiple-input multiple-output (MIMO) channel with N_r outputs. Each transmitted signal, input of the channel, goes through N_r different paths to arrive at the receiver. In a wireless communication system with N_r receive antennas, each output of the channel is a linear superposition of the faded versions of the inputs perturbed by noise. Each pair of transmit and receive antennas provides a signal path from the transmitter to the receiver. The coefficient $\alpha_{i,j}$ is the path gain from transmit antenna j to receive antenna i. Based on this model, the signal

$r_{t,i}$ which is received at time t at antenna i is given by

$$r_{t,i} = \sum_{j=1}^{N_t} \alpha_{i,j} \mathcal{C}_{t,j} + \eta_{t,i}, \tag{13}$$

where $\eta_{t,i}$ is the noise sample of the receive antenna i at time t.

To form a more compact input-output relationship, we collect the signals that are transmitted from N_t transmit antennas during a time slot in a column vector $\mathcal{C}$ of length N_t as follows:

$$\mathcal{C} = \begin{pmatrix} \mathcal{C}_1 \\ \mathcal{C}_2 \\ \vdots \\ \mathcal{C}_{N_t} \end{pmatrix}. \tag{14}$$

This vector is affected by $d_s = m \times N_t$ variable nodes (coded bits) and we may refer to it as $\mathcal{C}(V_1, V_2, \cdots V_{d_s})$.

Similarly, we construct a column vector $\mathbf{r}$ that includes all received signals during a given time slot:

$$\mathbf{r} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_{N_r} \end{pmatrix}. \tag{15}$$

Then gathering the path gains in an $N_r \times N_t$ channel matrix $\mathbf{H}$

$$\mathbf{H} = \begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,N_t} \\ \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,N_t} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{N_r,1} & \alpha_{N_r,2} & \cdots & \alpha_{N_r,N_t} \end{pmatrix}, \tag{16}$$

results in the following vector form of Equation (13):

$$\mathbf{r} = \mathbf{H} \cdot \mathcal{C} + \mathcal{N}, \tag{17}$$

where $\mathcal{N}$ is the $N_r \times 1$ noise vector defined by

$$\mathcal{N} = \begin{pmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_{N_r} \end{pmatrix}. \tag{18}$$

Let us assume a quasi-static slow fading model such that the noise samples η_i are independent samples of a zero-mean complex Gaussian random variable with variance $1/(2\,\mathrm{SNR})$ per complex dimension. This is an additive white Gaussian noise (AWGN) assumption for a complex baseband. If the average energy of the symbols transmitted from each antenna is normalized to be $1/N_t$, the average power of the received signal at each receive antenna is 1 and the received signal to noise ratio is SNR. Note that the path gains are zero mean complex Gaussian random variables with $Var[\Re\{\alpha_{i,j}\}] = Var[\Im\{\alpha_{i,j}\}] = 0.5$. Also, the bandwidth is narrow enough such that the channel is flat over a band of frequency. Such a channel is called non-frequency selective and the channel matrix is constant over the frequency band of interest. In the sequel, we use the quasi-static, non-frequency selective assumptions.

A block of k un-coded information bits will be coded by the LDPC encoder to a block of n coded bits. Assuming an M-ary modulation with $M = 2^m$, blocks of m coded bits from the output of the LDPC encoder select the appropriate symbol from the constellation. Blocks of N_t symbols (constructed from $m \times Nt$ coded bits) are transmitted simultaneously from the N_t transmit antennas. At the receiver side N_r noisy signals are received at the N_r received antennas. Figure 3 shows the corresponding graph for a binary modulation with $N_t = 2$ transmit and $N_r = 2$ receive antennas. The butterfly shaped sections on the left graph show the dependencies between the signals transmitted and received over multiple transmit and multiple receive antennas. Consecutive butterfly shaped sections on the graph correspond to consecutive channel uses.

On the right graph of Figure 3 the nodes corresponding to different receive antennas are lumped into a single *super-node*. This super-node simply is a vector valued random variable $\mathbf{r}$ as described in the above, whose components are the N_r received signals at the received antennas. This allows us to treat the MIMO case in tripartite LDPC framework.

The decoding is similar to the M-ary case. Only the symbol node to variable node messages are computed differently. The difference is that in Equations (8) and (9) the the argument of the exponential term would be the square of the magnitude of a vector whose multiple elements correspond to multiple receive antennas. In other words, the matrix of the channel fade coefficients $\mathbf{H}$ which is assumed to be known (estimated) in the receiver is multiplied by the vector of transmitted symbols for each combination of coded bits and the resulting vector is subtracted from the vector of received values in the multiple receive antennas. The norm of this difference vector is used as the argument of the exponential term in Equations (8) and (9).

We define vector $\mathbf{f}$ the faded version of the transmitted vector $\mathcal{C}$ to be:

$$\mathbf{f}(v_1, v_2, \cdots, v_{d_s}) = \mathbf{H} \cdot \mathcal{C}(v_1, v_2, \cdots, v_{d_s}), \tag{19}$$

where $v_1, v_2, \cdots, v_{d_s}$ are the values $d_s = m \times N_t$ variable nodes that affect the signal vector $\mathcal{C}(v_1, v_2, \cdots, v_{d_s})$.

Similar to the case of M-ary modulation over a single-input single-output channel, we refer to our current estimates for the probability of variables $V_1, V_2, \cdots, V_{d_s}$ as $p(v_1), p(v_2), \cdots, p(v_{d_s})$. The following equation gives the calculation of symbol to node messages for the MIMO case:

$$\psi_{s\to v} = \log \frac{\sum\limits_{v'_1, v'_2, \ldots, v'_{d_s-1} \in \mathcal{V}_s \backslash \{v\}} p(v'_1)p(v'_2)\ldots p(v'_{d_s-1}) e^{\|\mathbf{f}(v'_1, v'_2, \ldots, v'_{d_s-1}; V=1) - \mathbf{r}\|^2/2\sigma^2}}{\sum\limits_{v'_1, v'_2, \ldots, v'_{d_s-1} \in \mathcal{V}_s \backslash \{v\}} p(v'_1)p(v'_2)\ldots p(v'_{d_s-1}) e^{\|\mathbf{f}(v'_1, v'_2, \ldots, v'_{d_s-1}; V=0) - \mathbf{r}\|^2/2\sigma^2}} \tag{20}$$

Similar to the M-ary case we can re-write the above equation in terms of log-likelihood messages $m_i \triangleq \log \frac{p(v'_i=1)}{p(v'_i=0)}$.

$$\psi_{s\to v} = \log \frac{\sum\limits_{v'_1, v'_2, \ldots, v'_{d_s-1} \in \mathcal{V}_s \backslash \{v\}} e^{\sum_{i=1}^{d_s-1} m_i \chi(V'_i=1) + \|\mathbf{f}(v'_1, v'_2, \ldots, v'_{d_s-1}; V=1) - \mathbf{r}\|^2/2\sigma^2}}{\sum\limits_{v'_1, v'_2, \ldots, v'_{d_s-1} \in \mathcal{V}_s \backslash \{v\}} e^{\sum_{i=1}^{d_s-1} m_i \chi(V'_i=1) + \|\mathbf{f}(v'_1, v'_2, \ldots, v'_{d_s-1}; V=0) - \mathbf{r}\|^2/2\sigma^2}} \tag{21}$$

The rest of the messages are calculated similar to the M-ary case. Simulation results with two transmit and two receive antennas are given in [18].

V. Conclusion and Further Direction of Research

We presented the framework of tripartite LDPC codes. In contrast with previous works on LDPC codes, in this framework we consider the case where a channel output is affected by more than a single coded bit. Tripartite LPDC framework is general enough to cover the cases of M-ary modulation and MIMO channels. We prove that concentration theorem which was previously proved for the case of binary erasure channel by Luby *et al* and later was generalized to binary input memory less channels by Richardson *et. al.* holds in our more general framework as well.

Concentration: The decoding performance of a code chosen at random from the ensemble of codes with a given degree characteristic for check nodes and variable nodes is close to the expected value performance over that ensemble with a probability that approaches one exponentially fast in the block-size of the code.

Convergence to Cycle-free case: The percentage of wrong decoding messages in the lth iteration for a particular edge of the graph approaches to the expected value of this percentage assuming that the neighborhood of depth $4l$ of that edge is cycle free.

Concentration around the Cycle-free case: The decoding performance of a code chosen at random from the ensemble of codes with a given degree characteristic for check nodes and variable nodes is close to the expected value performance over that ensemble assuming that the decoding neighborhoods are cycle-free with a probability that approaches one exponentially fast in the block-size of the code.

Similar theorems for the case of binary erasure channel and the case of binary input symmetric output channels were proved in [3] and [2] respectively. Concentration Theorem is essential in designing Irregular LDPC codes and calculating the capacity of message passing decoding. We also presented the specific equations for messages used in decoding algorithm for M-ary modulations and MIMO channels.

It would be desirable to extend the other results that are established for the binary input channel to the case of tripartite LDPC codes as well. These results include "Density evolution and Threshold Determination" [4] and analysis of stopping sets [19].

In this paper we assumed the fade coefficients of the MIMO channel is known or estimated at the receiver. One possible extension of this framework is to include the unknown fade coefficients as a fourth tier to the graph and include the estimation of the coefficients within the belief propagation iterative decoding algorithm. In our approach we are considering a binary LDPC code for a non-binary modulation scheme. Another interesting approach would be trying to design a non-binary LDPC code.

REFERENCES

[1] R. G. Gallager, "*Low-Density Parity-Check Codes.*" Cambridge, MA: MIT Press, 1963.

[2] Richardson, T.J.; Urbanke, R.L. "The capacity of low-density parity-check codes under message-passing decoding".*IEEE Transactions on Information Theory*, vol.47, (no.2), IEEE, Feb. 2001. p.599-618

[3] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, " Improved low-density parity check codes using irregular graphs and belief propagation," presented at the 1998 International Symposium on Information Theory, ISIT98, Cambridge, MA, Aug. 16–21 1998.

[4] Richardson, T.J.; Shokrollahi, M.A., Urbanke, R.L. "Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes".*IEEE Transactions on Information Theory*, vol.47, (no.2), IEEE, p.619–637, Feb. 2001.

[5] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes," *Proc. of ICC '93*, pp. 1064-1070, 1993.

[6] J. Pearl, "Probabilistic Reasoning in Intelligent Systems", San Mateo, CA: Morgan Kaufmann, 1988.

[7] N. Wiberg, "Codes and Decoding on General Graphs", Linkoping Studies in Science and Technology, Dissertation No. 440. Linkoping, Sweden, 1996.

[8] D.J.C. MacKay, R.J. McEliece and J.F. Cheng, "Turbo Decoding as an Instance of Pearl's Belief Propagation Algorithm", IEEE *Journal on Selected Areas in Communication*, vol. 16, pp. 140–152, Feb. 1998. .

[9] F. Kschischang and B. Frey, "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE Journal on Selected Areas in Communication*, vol 16 pp. 219–230, Feb. 1998.

[10] F. R. Kschischang, B. J. Frey and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory,* Vol. 47 No. 2, pp. 498–519, Feb 2001.

[11] S. Benedetto and G. Montorsi, "Serial concatenation of block and convolutional codes," *Electronics Letters,* Vol. 32, pp. 887–888, May 1996.

[12] MacKay, D.J.C. "Good Error-Correcting Codes based on Very Sparse Matrices," *IEEE Transactions on Information Theory,* Vol. 45 No. 2, pp. 399–431, Mar. 1999.

[13] A. Khandekar, H. Jin and R. J. McEliece, "Irregular Repeat-Accumulate Codes," *Proceedings 2nd International Symposium on Turbo codes and Related Topics,* pp. 1–8. Brest, France, Sept. 4, 2000.

[14] G. J. Foschini and M. Gans, "On the limits of wireless communication in a fading environment when using multiple antennas," *Wireless Pers. Commun.*, vol. 6, pp. 311–335, Mar. 1998.

[15] E. Teletar "Capacity of multi-antenna Gaussian channels," *European Transactions on Telecommunications*, vol. 10, pp. 585–595, Nov./Dec. 1999.

[16] V. Tarokh, N. Seshadri and A.R. Calderbank, "Space-time codes for high data rate wireless communication: Performance analysis and code construction," *IEEE Trans. Inform. Theory*, vol. 44, pp. 744–765, Mar. 1998.

[17] V. Tarokh, H. Jafarkhani, A. R. Calderbank, "Space-Time Block Codes From Orthogonal Designs," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1456–1467, July 1999.

[18] P. Meshkat and H. Jafarkhani, "Space-Time Low Density Parity Check Codes," *Asilomar Conference on Signals, Systems, and Computers,* invited paper, Nov. 2002.

[19] C. Di, D. Proietti, E. Telatar, T. Richardson, and R. Urbanke, "Finite length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inform. Theory,* vol. 48, pp. 1570-1579, June 2002.

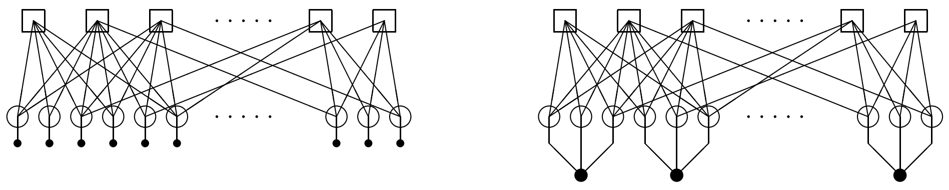

Fig. 1. **Graphs of bipartite and tripartite LDPC codes:** Squares correspond to check nodes, white circles correspond to bit (variable) nodes and black circles correspond to symbol nodes (channel output).

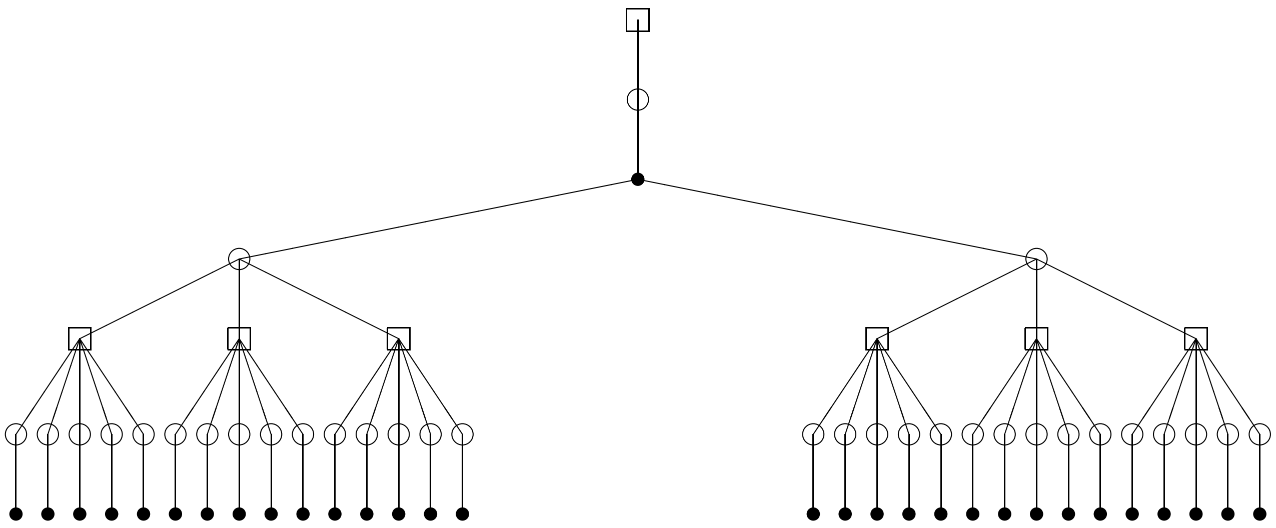

Fig. 2. **Decoding Neighborhood of tripartite LDPC code:** Performing each iteration adds four new tiers to the decoding neighborhood.

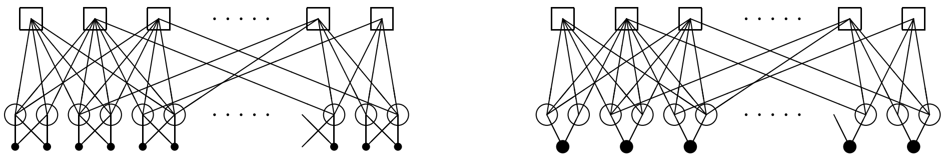

Fig. 3. **LDPC code used over MIMO channel** Squares correspond to check nodes, white circles correspond to bit (variable) nodes and black circles correspond to channel output. The butterfly-shaped sections on the left graph correspond to two transmit and two receive antennas. In the right graph the nodes in the bottom tier of each butterfly-shaped section are combined to a single "super-node". Now the left graph fits in the tripartite framework.