Spatial Coupling and the Threshold Saturation Phenomenon

Shrinivas Kudekar
Qualcomm Research

Ruediger Urbanke
EPFL

July 7th, 2013

https://dl.dropbox.com/u/2826733/ISIT%202013%20Tutorial.m4v

The latest version of these slides (Keynote and PDF) can be found at
https://ipg.epfl.ch/doku.php?id=en:publications:scc_tutorial

Part I: All we know about iterative coding we learnt by looking at the BEC

In part I and II we will talk exclusively about the binary erasure channel (BEC). This will avoid having to deal with technicalities. But we will present things in a way that all statements stay valid for the general case (general binary-input memoryless output-symmetric (BMS) channels).

Uncoupled Codes

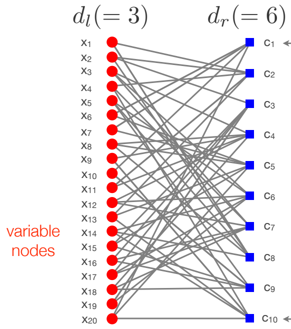
Let us start by looking at the simplest case, namely regular LDPC codes, and how to analyze them.

Introduction - Graphical Codes

Low-density Parity-Check (LDPC) Codes

Low-Density Parity-Check Codes
Robert G. Gallager
1963

$d_l(=3)$ $d_r(=6)$



$c_1 \longleftrightarrow x_6 + x_7 + x_{10} + x_{20} = 0$

$$\text{rate} \geq r_{\text{design}} = \frac{\#\text{variables} - \#\text{checks}}{\#\text{variables}} = \frac{20-10}{20} = \frac{1}{2}$$

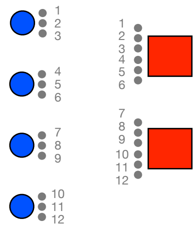
$\text{rate} \sim r_{\text{design}}$

$c_{10} \longleftrightarrow x_4 + x_9 + x_{13} + x_{14} + x_{20} = 0$

It was shown by Miller and Cohen (The rate of regular LDPC Codes, IEEE Trans. IT, 49, 2003, pp. 2989--2992), that with high probability the rate of a randomly chosen regular code is very close to this lower bound. See also page 81 in MCT (Modern Coding Theory). By *regular* code here we mean a code where all variables have degree lets say d_l and all check nodes have degree, lets say d_r.

Ensemble of Codes - Configuration Construction

(3, 6) ensemble



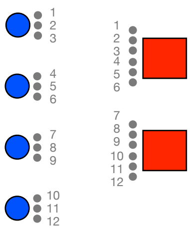
each configuration has uniform probability

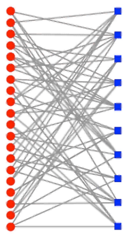
code is sampled u.a.r. from the ensemble and used for transmission

The configuration model is a very convenient way of defining the ensemble since it is trivial to sample from this ensemble. All we need is to create a sample uniformly at random from the set of permutations on E letters, where E is the number of edges. But note that in this definition two distinct permutations can create the "same graph" once we delete the labels of the sockets. In other words, the probability distribution when we delete socket labels is no longer uniform. A further advantage of using the configuration model is that it leads to a simple analysis.

Ensemble of Codes - Configuration Construction

(3, 6) ensemble





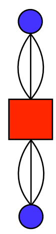
each configuration has uniform probability

code is sampled u.a.r. from the ensemble and used for transmission

Saturday, July 13, 13 6

The configuration model is a very convenient way of defining the ensemble since it is trivial to sample from this ensemble. All we need is to create a sample uniformly at random from the set of permutations on E letters, where E is the number of edges. But note that in this definition two distinct permutations can create the "same graph" once we delete the labels of the sockets. In other words, the probability distribution when we delete socket labels is no longer uniform. A further advantage of using the configuration model is that it leads to a simple analysis.

Ensemble of Codes - Protograph Construction



IPN Progress Report 42-154 August 15, 2003

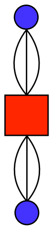
Low-Density Parity-Check (LDPC) Codes Constructed from Protographs

J. Thorpe[1]

We introduce a new class of low-density parity-check (LDPC) codes constructed from a template called a protograph. The protograph serves as a blueprint for constructing LDPC codes of arbitrary size whose performance can be predicted by analyzing the protograph. We apply standard density evolution techniques to predict the performance of large protograph codes. Finally, we use a randomized search algorithm to find good protographs.

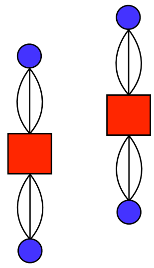
Protographs have two advantages. First, they are a convenient and compact way of specifying graphical codes. Second, the additional structure can be useful if we want to implement codes in practice.

Ensemble of Codes - Protograph Construction



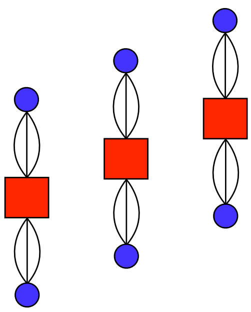
In order to create a “real” graph from a protograph we “lift” it. This means that we make M copies, where M is typically in the order of hundreds or thousands. In the example above we chose M=5. We then connect these copies by permuting the edges in each “edge bundle” by means of a permutation chosen uniformly at random.

Ensemble of Codes - Protograph Construction



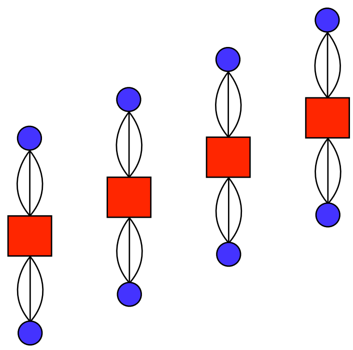
In order to create a “real” graph from a protograph we “lift” it. This means that we make M copies, where M is typically in the order of hundreds or thousands. In the example above we chose M=5. We then connect these copies by permuting the edges in each “edge bundle” by means of a permutation chosen uniformly at random.

Ensemble of Codes - Protograph Construction



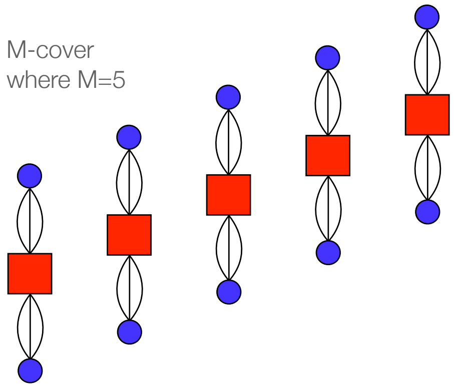
In order to create a “real” graph from a protograph we “lift” it. This means that we make M copies, where M is typically in the order of hundreds or thousands. In the example above we chose M=5. We then connect these copies by permuting the edges in each “edge bundle” by means of a permutation chosen uniformly at random.

Ensemble of Codes - Protograph Construction



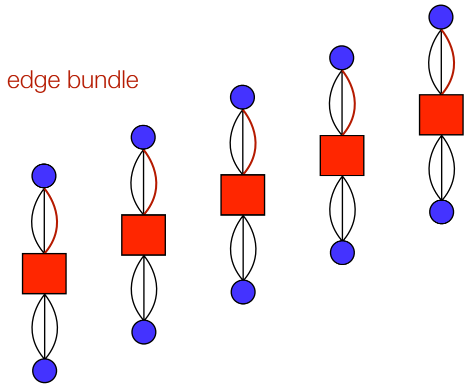
In order to create a “real” graph from a protograph we “lift” it. This means that we make M copies, where M is typically in the order of hundreds or thousands. In the example above we chose M=5. We then connect these copies by permuting the edges in each “edge bundle” by means of a permutation chosen uniformly at random.

Ensemble of Codes - Protograph Construction



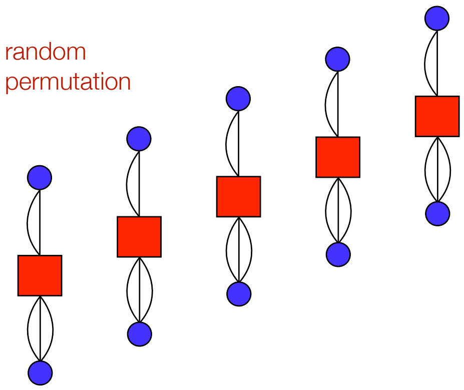
In order to create a “real” graph from a protograph we “lift” it. This means that we make M copies, where M is typically in the order of hundreds or thousands. In the example above we chose M=5. We then connect these copies by permuting the edges in each “edge bundle” by means of a permutation chosen uniformly at random.

Ensemble of Codes - Protograph Construction



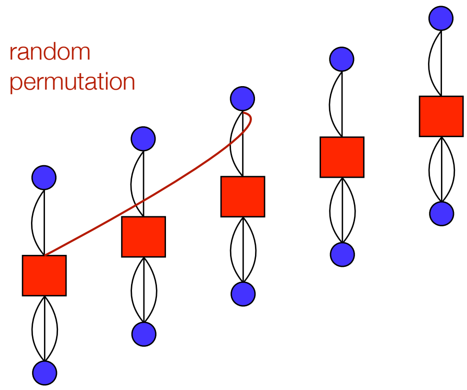
By “edge bundle” we mean here a set of “like” edges. I.e., edges which connect the same variable node and the same check node in each protograph. In the slide above a particular edge bundle is indicated in red.

Ensemble of Codes - Protograph Construction



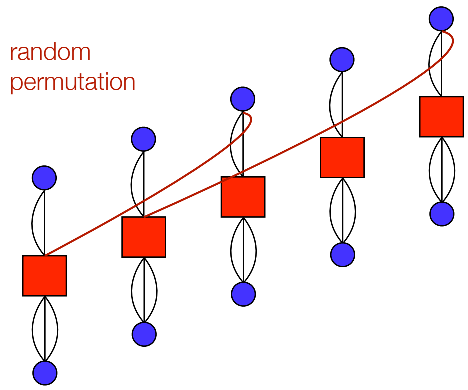
We now permute the edges in this edge bundle. We do the same thing for each edge bundle. Note that strictly speaking the ensemble created in this way is different from the ensemble created by the configuration model. But for the asymptotic analysis (density evolution -- see subsequent slides) the two models are equivalent.

Ensemble of Codes - Protograph Construction



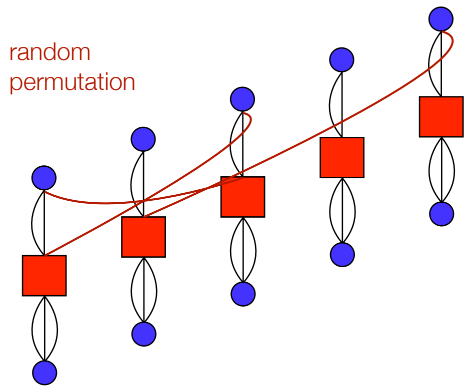
We now permute the edges in this edge bundle. We do the same thing for each edge bundle. Note that strictly speaking the ensemble created in this way is different from the ensemble created by the configuration model. But for the asymptotic analysis (density evolution -- see subsequent slides) the two models are equivalent.

Ensemble of Codes - Protograph Construction



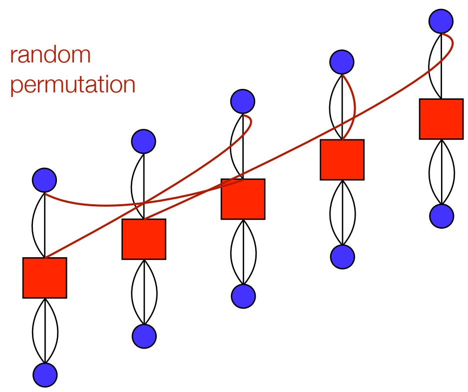
We now permute the edges in this edge bundle. We do the same thing for each edge bundle. Note that strictly speaking the ensemble created in this way is different from the ensemble created by the configuration model. But for the asymptotic analysis (density evolution -- see subsequent slides) the two models are equivalent.

Ensemble of Codes - Protograph Construction



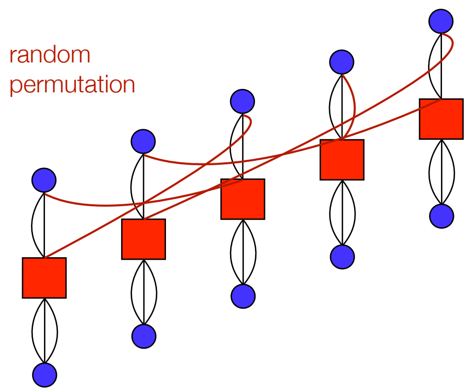
We now permute the edges in this edge bundle. We do the same thing for each edge bundle. Note that strictly speaking the ensemble created in this way is different from the ensemble created by the configuration model. But for the asymptotic analysis (density evolution -- see subsequent slides) the two models are equivalent.

Ensemble of Codes - Protograph Construction



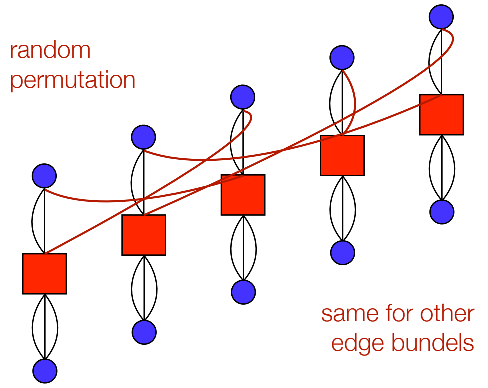
We now permute the edges in this edge bundle. We do the same thing for each edge bundle. Note that strictly speaking the ensemble created in this way is different from the ensemble created by the configuration model. But for the asymptotic analysis (density evolution -- see subsequent slides) the two models are equivalent.

Ensemble of Codes - Protograph Construction



We now permute the edges in this edge bundle. We do the same thing for each edge bundle. Note that strictly speaking the ensemble created in this way is different from the ensemble created by the configuration model. But for the asymptotic analysis (density evolution -- see subsequent slides) the two models are equivalent.

Ensemble of Codes - Protograph Construction



We now permute the edges in this edge bundle. We do the same thing for each edge bundle. Note that strictly speaking the ensemble created in this way is different from the ensemble created by the configuration model. But for the asymptotic analysis (density evolution -- see subsequent slides) the two models are equivalent.

Bit MAP Decoder to Belief Propagation Decoder

$$\begin{aligned}\hat{x}_i^{\mathrm{MAP}} &= \mathrm{argmax}_{x_i \in \{\pm 1\}} p(X_i = x_i \mid y) \\ &= \mathrm{argmax}_{x_i \in \{\pm 1\}} \sum_{x_i} \Big(\prod_j p(y_j \mid x_j) \Big) \mathbf{1}_{\{x \in \mathcal{C}\}}\end{aligned}$$

For the BEC, bit MAP decoding could be done by solving a system of linear equations, i.e., in complexity n^3.But we are interested in an algorithm that is applicable for general BMS channels (where MAP decoding is typically intractable). We therefore only consider a message-passing algorithm which is applicable also in the general case. More precisely, we consider the sum-product (also called belief-propagation (BP)) algorithm. This algorithm performs bit MAP decoding on codes on graphs whose factor graph is a tree.

Bit MAP Decoder to Belief Propagation Decoder

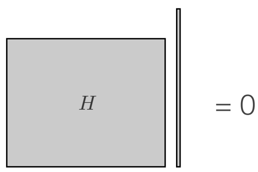
$$\begin{aligned}\hat{x}_i^{\mathrm{MAP}} &= \mathrm{argmax}_{x_i \in \{\pm 1\}} p(X_i = x_i \mid y) \\ &= \mathrm{argmax}_{x_i \in \{\pm 1\}} \sum_{x_i} \Big(\prod_j p(y_j \mid x_j) \Big) \mathbf{1}_{\{x \in \mathcal{C}\}}\end{aligned}$$

Complexity for the BEC O(n^3)

For the BEC, bit MAP decoding could be done by solving a system of linear equations, i.e., in complexity n^3.But we are interested in an algorithm that is applicable for general BMS channels (where MAP decoding is typically intractable). We therefore only consider a message-passing algorithm which is applicable also in the general case. More precisely, we consider the sum-product (also called belief-propagation (BP)) algorithm. This algorithm performs bit MAP decoding on codes on graphs whose factor graph is a tree.

Bit MAP Decoder to Belief Propagation Decoder

$$\begin{aligned}\hat{x}_i^{\mathrm{MAP}} &= \mathrm{argmax}_{x_i \in \{\pm 1\}} p(X_i = x_i \mid y) \\ &= \mathrm{argmax}_{x_i \in \{\pm 1\}} \sum_{x_i} \Big(\prod_j p(y_j \mid x_j) \Big) \mathbf{1}_{\{x \in \mathcal{C}\}}\end{aligned}$$



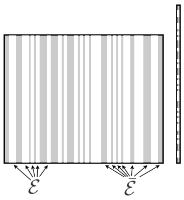
$$Hx^T = 0^T$$

Complexity for the BEC O(n^3)

For the BEC, bit MAP decoding could be done by solving a system of linear equations, i.e., in complexity n^3.But we are interested in an algorithm that is applicable for general BMS channels (where MAP decoding is typically intractable). We therefore only consider a message-passing algorithm which is applicable also in the general case. More precisely, we consider the sum-product (also called belief-propagation (BP)) algorithm. This algorithm performs bit MAP decoding on codes on graphs whose factor graph is a tree.

Bit MAP Decoder to Belief Propagation Decoder

$$\hat{x}_i^{\text{MAP}} = \text{argmax}_{x_i \in \{\pm 1\}} p(X_i = x_i \mid y)$$
$$= \text{argmax}_{x_i \in \{\pm 1\}} \sum_{\sim x_i} \Big(\prod_j p(y_j \mid x_j) \Big) \mathbf{1}_{\{x \in \mathcal{C}\}}$$



= 0

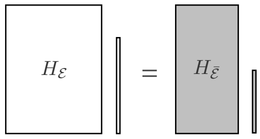
$$Hx^T = 0^T$$

Complexity for the BEC O(n^3)

For the BEC, bit MAP decoding could be done by solving a system of linear equations, i.e., in complexity n^3.But we are interested in an algorithm that is applicable for general BMS channels (where MAP decoding is typically intractable). We therefore only consider a message-passing algorithm which is applicable also in the general case. More precisely, we consider the sum-product (also called belief-propagation (BP)) algorithm. This algorithm performs bit MAP decoding on codes on graphs whose factor graph is a tree.

Bit MAP Decoder to Belief Propagation Decoder

$$\hat{x}_i^{\text{MAP}} = \text{argmax}_{x_i \in \{\pm 1\}} p(X_i = x_i \mid y)$$
$$= \text{argmax}_{x_i \in \{\pm 1\}} \sum_{\sim x_i} \Big(\prod_j p(y_j \mid x_j) \Big) \mathbf{1}_{\{x \in \mathcal{C}\}}$$



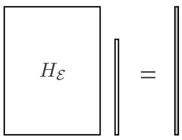
$$Hx^T = 0^T$$
$$H_{\mathcal{E}} x_{\mathcal{E}}^T = H_{\bar{\mathcal{E}}} x_{\bar{\mathcal{E}}}^T$$

Complexity for the BEC O(n^3)

For the BEC, bit MAP decoding could be done by solving a system of linear equations, i.e., in complexity n^3.But we are interested in an algorithm that is applicable for general BMS channels (where MAP decoding is typically intractable). We therefore only consider a message-passing algorithm which is applicable also in the general case. More precisely, we consider the sum-product (also called belief-propagation (BP)) algorithm. This algorithm performs bit MAP decoding on codes on graphs whose factor graph is a tree.

Bit MAP Decoder to Belief Propagation Decoder

$$\hat{x}_i^{\text{MAP}} = \text{argmax}_{x_i \in \{\pm 1\}} p(X_i = x_i \mid y)$$
$$= \text{argmax}_{x_i \in \{\pm 1\}} \sum_{\sim x_i} \Big(\prod_j p(y_j \mid x_j) \Big) \mathbf{1}_{\{x \in \mathcal{C}\}}$$



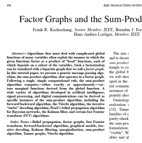
$$Hx^T = 0^T$$
$$H_{\mathcal{E}} x_{\mathcal{E}}^T = H_{\bar{\mathcal{E}}} x_{\bar{\mathcal{E}}}^T$$
$$H_{\mathcal{E}} x_{\mathcal{E}}^T = s^T$$

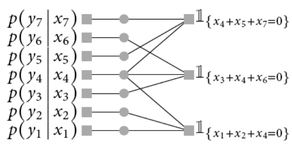
Complexity for the BEC O(n^3)

For the BEC, bit MAP decoding could be done by solving a system of linear equations, i.e., in complexity n^3.But we are interested in an algorithm that is applicable for general BMS channels (where MAP decoding is typically intractable). We therefore only consider a message-passing algorithm which is applicable also in the general case. More precisely, we consider the sum-product (also called belief-propagation (BP)) algorithm. This algorithm performs bit MAP decoding on codes on graphs whose factor graph is a tree.

Bit MAP Decoder to Belief Propagation Decoder

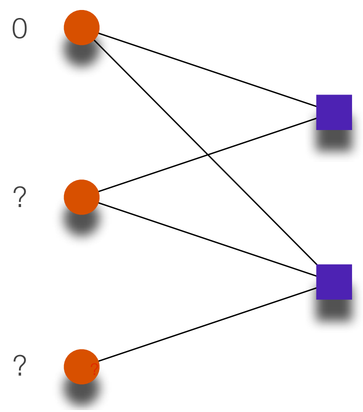
$$\hat{x}_i^{\text{MAP}} = \text{argmax}_{x_i \in \{\pm 1\}} p(X_i = x_i \mid y)$$
$$= \text{argmax}_{x_i \in \{\pm 1\}} \sum_{\sim x_i} \Big(\prod_j p(y_j \mid x_j) \Big) \mathbf{1}_{\{x \in \mathcal{C}\}}$$





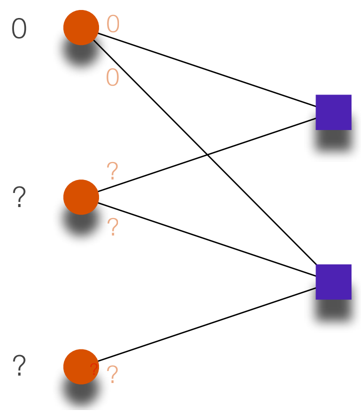
For the BEC, bit MAP decoding could be done by solving a system of linear equations, i.e., in complexity n^3.But we are interested in an algorithm that is applicable for general BMS channels (where MAP decoding is typically intractable). We therefore only consider a message-passing algorithm which is applicable also in the general case. More precisely, we consider the sum-product (also called belief-propagation (BP)) algorithm. This algorithm performs bit MAP decoding on codes on graphs whose factor graph is a tree.

BP Decoder - BEC



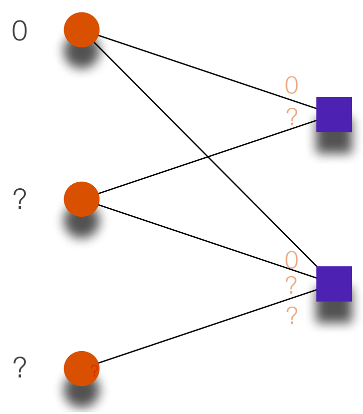
Here we see the BP algorithm in action. For the BEC the BP algorithm is particularly simple and performs a very natural operation. Every time we have a check node so that all but one of its inputs are known, the BP algorithm uses the relationship implied by this check node to determine the unknown input.

BP Decoder - BEC



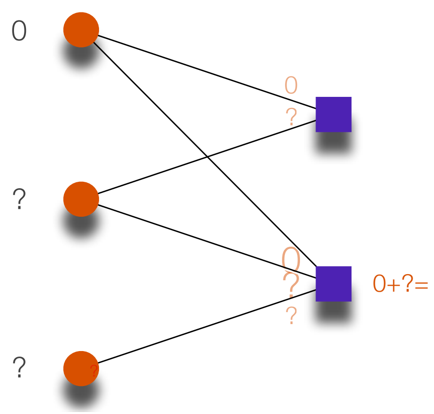
Here we see the BP algorithm in action. For the BEC the BP algorithm is particularly simple and performs a very natural operation. Every time we have a check node so that all but one of its inputs are known, the BP algorithm uses the relationship implied by this check node to determine the unknown input.

BP Decoder - BEC



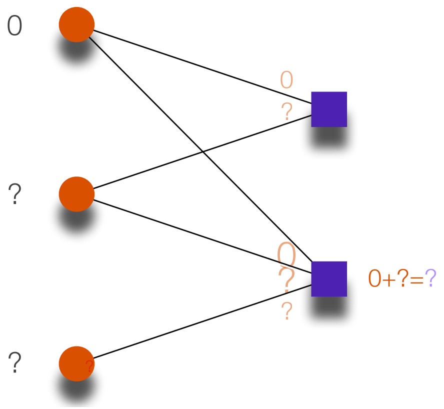
Here we see the BP algorithm in action. For the BEC the BP algorithm is particularly simple and performs a very natural operation. Every time we have a check node so that all but one of its inputs are known, the BP algorithm uses the relationship implied by this check node to determine the unknown input.

BP Decoder - BEC



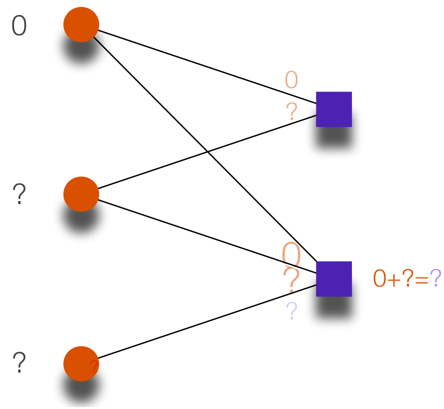
Here we see the BP algorithm in action. For the BEC the BP algorithm is particularly simple and performs a very natural operation. Every time we have a check node so that all but one of its inputs are known, the BP algorithm uses the relationship implied by this check node to determine the unknown input.

BP Decoder - BEC



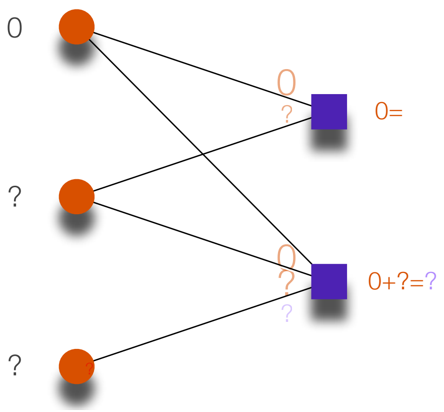
Here we see the BP algorithm in action. For the BEC the BP algorithm is particularly simple and performs a very natural operation. Every time we have a check node so that all but one of its inputs are known, the BP algorithm uses the relationship implied by this check node to determine the unknown input.

BP Decoder - BEC



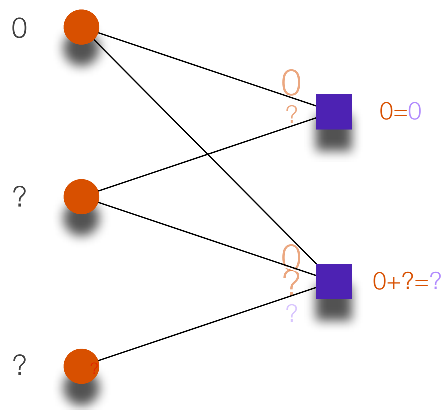
Here we see the BP algorithm in action. For the BEC the BP algorithm is particularly simple and performs a very natural operation. Every time we have a check node so that all but one of its inputs are known, the BP algorithm uses the relationship implied by this check node to determine the unknown input.

BP Decoder - BEC



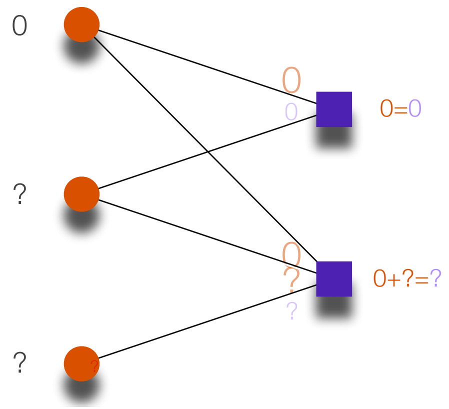
Here we see the BP algorithm in action. For the BEC the BP algorithm is particularly simple and performs a very natural operation. Every time we have a check node so that all but one of its inputs are known, the BP algorithm uses the relationship implied by this check node to determine the unknown input.

BP Decoder - BEC



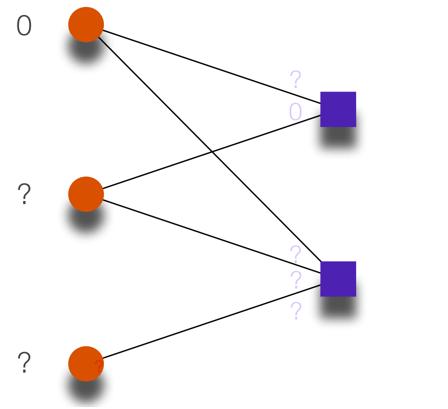
Here we see the BP algorithm in action. For the BEC the BP algorithm is particularly simple and performs a very natural operation. Every time we have a check node so that all but one of its inputs are known, the BP algorithm uses the relationship implied by this check node to determine the unknown input.

BP Decoder - BEC



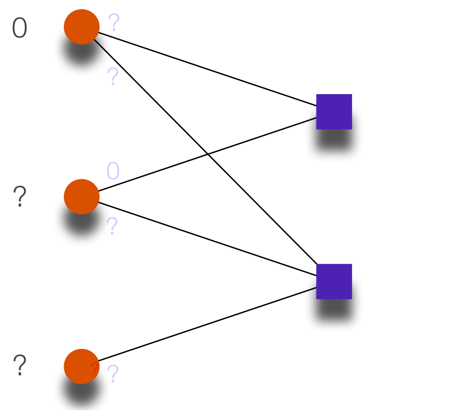
Here we see the BP algorithm in action. For the BEC the BP algorithm is particularly simple and performs a very natural operation. Every time we have a check node so that all but one of its inputs are known, the BP algorithm uses the relationship implied by this check node to determine the unknown input.

BP Decoder - BEC



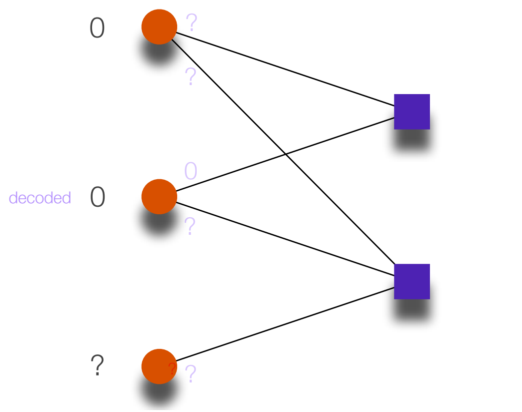
Here we see the BP algorithm in action. For the BEC the BP algorithm is particularly simple and performs a very natural operation. Every time we have a check node so that all but one of its inputs are known, the BP algorithm uses the relationship implied by this check node to determine the unknown input.

BP Decoder - BEC



Here we see the BP algorithm in action. For the BEC the BP algorithm is particularly simple and performs a very natural operation. Every time we have a check node so that all but one of its inputs are known, the BP algorithm uses the relationship implied by this check node to determine the unknown input.

BP Decoder - BEC

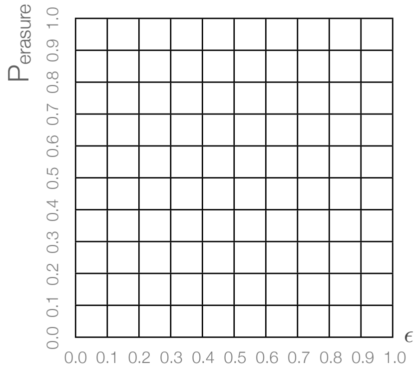


Here we see the BP algorithm in action. For the BEC the BP algorithm is particularly simple and performs a very natural operation. Every time we have a check node so that all but one of its inputs are known, the BP algorithm uses the relationship implied by this check node to determine the unknown input.

How does BP perform on the BEC?

Now where we have defined the class of codes we consider, and the algorithm which we use for decoding, we proceed to see how this combination performs.

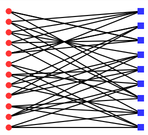
How does BP perform on the BEC?

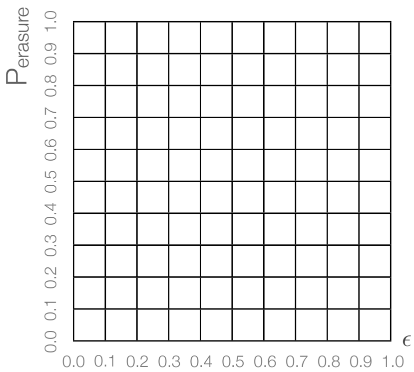


(3, 6) ensemble

Here is the experiment we consider. Fix the ensemble. In the above example it is the (3, 6)-regular ensemble. This will serve as our running example. Now pick very long instances of this ensemble. Pick a random codeword and transmit over a BEC with erasure probability eps. Run the BP decoder until it no longer makes any progress. Record the error probability and average over many instances. Plot the average bit-error probability versus eps. Naturally, as eps decreases the error probability decreases. What is most interesting is that at some specific point we see a jump of the error probability from a non-zero value down to zero. This is the BP “threshold.”In the next slides, we will explain how to locate the BP threshold.

How does BP perform on the BEC?

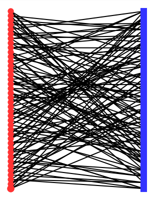


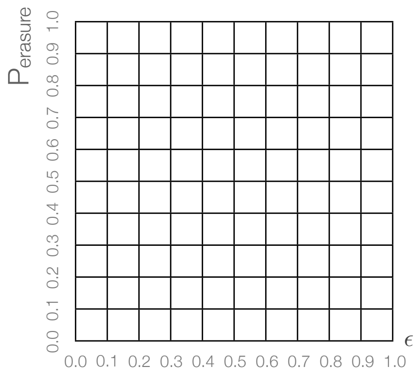


(3, 6) ensemble

Here is the experiment we consider. Fix the ensemble. In the above example it is the (3, 6)-regular ensemble. This will serve as our running example. Now pick very long instances of this ensemble. Pick a random codeword and transmit over a BEC with erasure probability eps. Run the BP decoder until it no longer makes any progress. Record the error probability and average over many instances. Plot the average bit-error probability versus eps. Naturally, as eps decreases the error probability decreases. What is most interesting is that at some specific point we see a jump of the error probability from a non-zero value down to zero. This is the BP “threshold.”In the next slides, we will explain how to locate the BP threshold.

How does BP perform on the BEC?

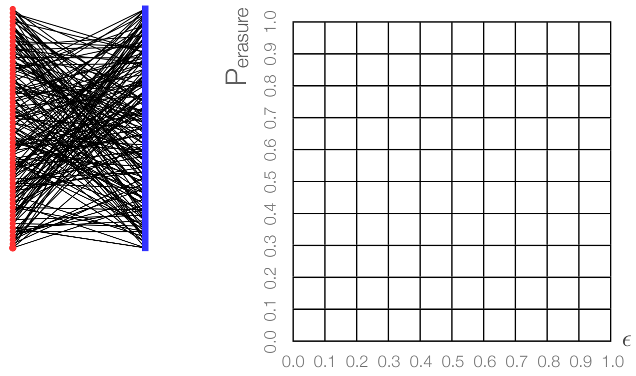




(3, 6) ensemble

Here is the experiment we consider. Fix the ensemble. In the above example it is the (3, 6)-regular ensemble. This will serve as our running example. Now pick very long instances of this ensemble. Pick a random codeword and transmit over a BEC with erasure probability eps. Run the BP decoder until it no longer makes any progress. Record the error probability and average over many instances. Plot the average bit-error probability versus eps. Naturally, as eps decreases the error probability decreases. What is most interesting is that at some specific point we see a jump of the error probability from a non-zero value down to zero. This is the BP “threshold.”In the next slides, we will explain how to locate the BP threshold.

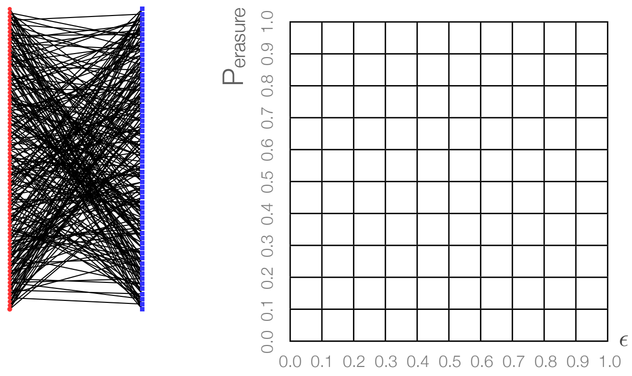
How does BP perform on the BEC?



(3, 6) ensemble

Here is the experiment we consider. Fix the ensemble. In the above example it is the (3, 6)-regular ensemble. This will serve as our running example. Now pick very long instances of this ensemble. Pick a random codeword and transmit over a BEC with erasure probability eps. Run the BP decoder until it no longer makes any progress. Record the error probability and average over many instances. Plot the average bit-error probability versus eps. Naturally, as eps decreases the error probability decreases. What is most interesting is that at some specific point we see a jump of the error probability from a non-zero value down to zero. This is the BP “threshold.”In the next slides, we will explain how to locate the BP threshold.

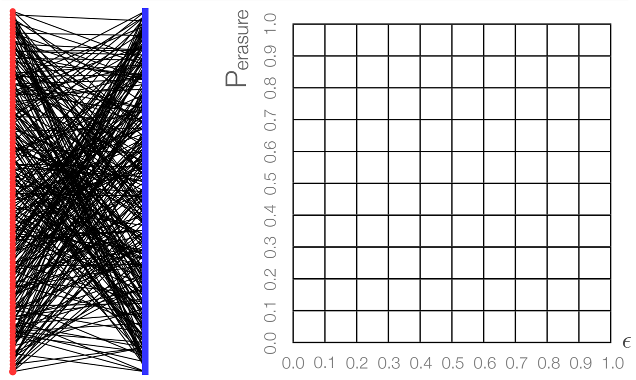
How does BP perform on the BEC?



(3, 6) ensemble

Here is the experiment we consider. Fix the ensemble. In the above example it is the (3, 6)-regular ensemble. This will serve as our running example. Now pick very long instances of this ensemble. Pick a random codeword and transmit over a BEC with erasure probability eps. Run the BP decoder until it no longer makes any progress. Record the error probability and average over many instances. Plot the average bit-error probability versus eps. Naturally, as eps decreases the error probability decreases. What is most interesting is that at some specific point we see a jump of the error probability from a non-zero value down to zero. This is the BP “threshold.”In the next slides, we will explain how to locate the BP threshold.

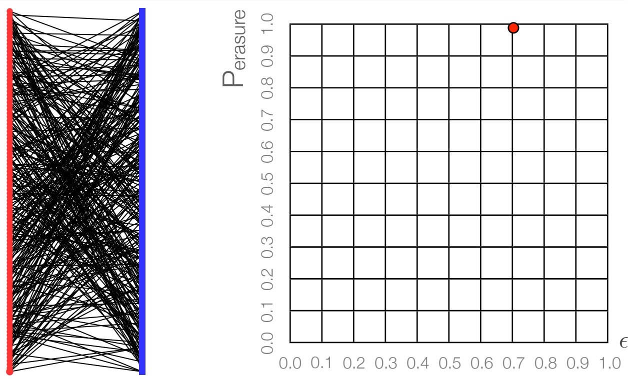
How does BP perform on the BEC?



(3, 6) ensemble

Here is the experiment we consider. Fix the ensemble. In the above example it is the (3, 6)-regular ensemble. This will serve as our running example. Now pick very long instances of this ensemble. Pick a random codeword and transmit over a BEC with erasure probability eps. Run the BP decoder until it no longer makes any progress. Record the error probability and average over many instances. Plot the average bit-error probability versus eps. Naturally, as eps decreases the error probability decreases. What is most interesting is that at some specific point we see a jump of the error probability from a non-zero value down to zero. This is the BP “threshold.”In the next slides, we will explain how to locate the BP threshold.

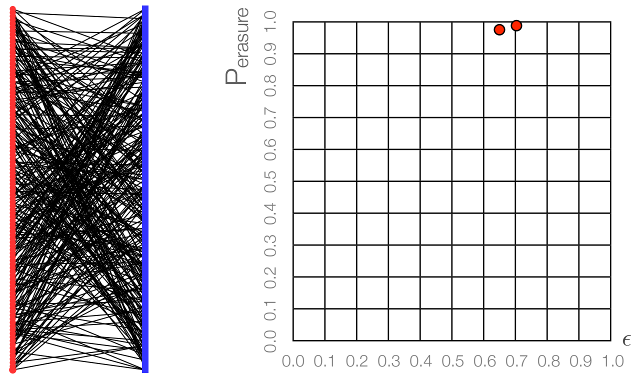
How does BP perform on the BEC?



(3, 6) ensemble

Here is the experiment we consider. Fix the ensemble. In the above example it is the (3, 6)-regular ensemble. This will serve as our running example. Now pick very long instances of this ensemble. Pick a random codeword and transmit over a BEC with erasure probability eps. Run the BP decoder until it no longer makes any progress. Record the error probability and average over many instances. Plot the average bit-error probability versus eps. Naturally, as eps decreases the error probability decreases. What is most interesting is that at some specific point we see a jump of the error probability from a non-zero value down to zero. This is the BP “threshold.”In the next slides, we will explain how to locate the BP threshold.

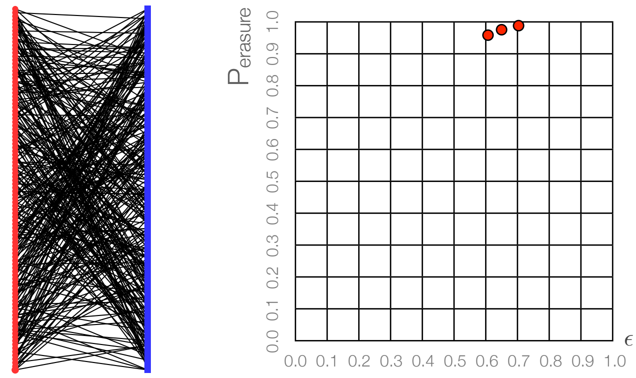
How does BP perform on the BEC?



(3, 6) ensemble

Here is the experiment we consider. Fix the ensemble. In the above example it is the (3, 6)-regular ensemble. This will serve as our running example. Now pick very long instances of this ensemble. Pick a random codeword and transmit over a BEC with erasure probability eps. Run the BP decoder until it no longer makes any progress. Record the error probability and average over many instances. Plot the average bit-error probability versus eps. Naturally, as eps decreases the error probability decreases. What is most interesting is that at some specific point we see a jump of the error probability from a non-zero value down to zero. This is the BP “threshold.”In the next slides, we will explain how to locate the BP threshold.

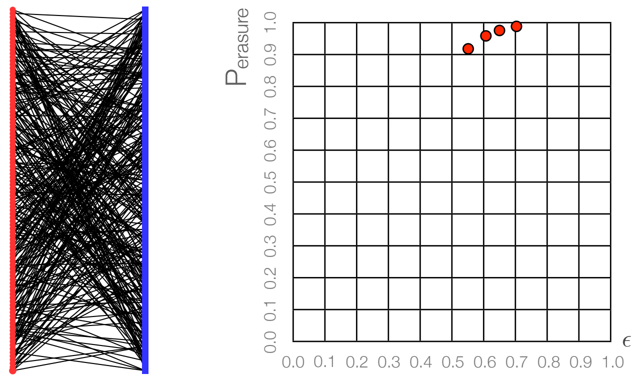
How does BP perform on the BEC?



(3, 6) ensemble

Here is the experiment we consider. Fix the ensemble. In the above example it is the (3, 6)-regular ensemble. This will serve as our running example. Now pick very long instances of this ensemble. Pick a random codeword and transmit over a BEC with erasure probability eps. Run the BP decoder until it no longer makes any progress. Record the error probability and average over many instances. Plot the average bit-error probability versus eps. Naturally, as eps decreases the error probability decreases. What is most interesting is that at some specific point we see a jump of the error probability from a non-zero value down to zero. This is the BP “threshold.”In the next slides, we will explain how to locate the BP threshold.

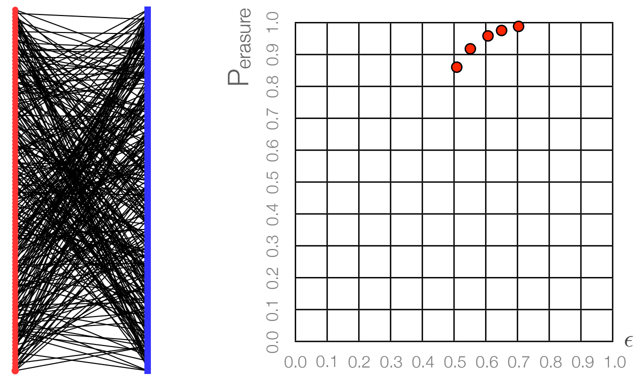
How does BP perform on the BEC?



(3, 6) ensemble

Here is the experiment we consider. Fix the ensemble. In the above example it is the (3, 6)-regular ensemble. This will serve as our running example. Now pick very long instances of this ensemble. Pick a random codeword and transmit over a BEC with erasure probability eps. Run the BP decoder until it no longer makes any progress. Record the error probability and average over many instances. Plot the average bit-error probability versus eps. Naturally, as eps decreases the error probability decreases. What is most interesting is that at some specific point we see a jump of the error probability from a non-zero value down to zero. This is the BP “threshold.”In the next slides, we will explain how to locate the BP threshold.

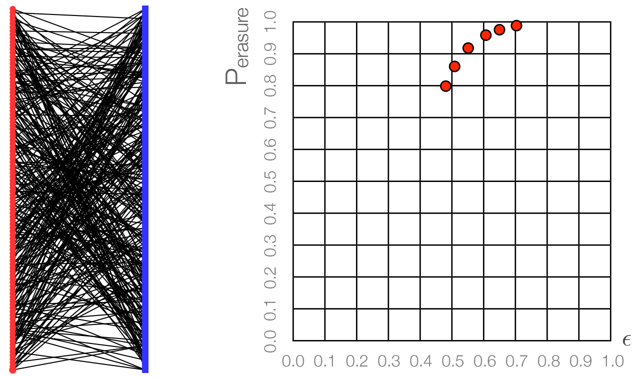
How does BP perform on the BEC?



(3, 6) ensemble

Here is the experiment we consider. Fix the ensemble. In the above example it is the (3, 6)-regular ensemble. This will serve as our running example. Now pick very long instances of this ensemble. Pick a random codeword and transmit over a BEC with erasure probability eps. Run the BP decoder until it no longer makes any progress. Record the error probability and average over many instances. Plot the average bit-error probability versus eps. Naturally, as eps decreases the error probability decreases. What is most interesting is that at some specific point we see a jump of the error probability from a non-zero value down to zero. This is the BP “threshold.”In the next slides, we will explain how to locate the BP threshold.

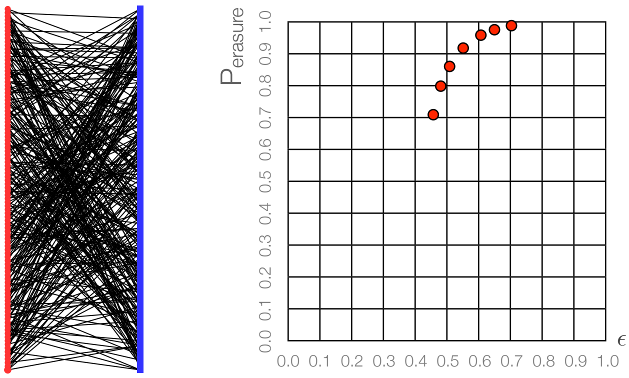
How does BP perform on the BEC?



(3, 6) ensemble

Here is the experiment we consider. Fix the ensemble. In the above example it is the (3, 6)-regular ensemble. This will serve as our running example. Now pick very long instances of this ensemble. Pick a random codeword and transmit over a BEC with erasure probability eps. Run the BP decoder until it no longer makes any progress. Record the error probability and average over many instances. Plot the average bit-error probability versus eps. Naturally, as eps decreases the error probability decreases. What is most interesting is that at some specific point we see a jump of the error probability from a non-zero value down to zero. This is the BP “threshold.”In the next slides, we will explain how to locate the BP threshold.

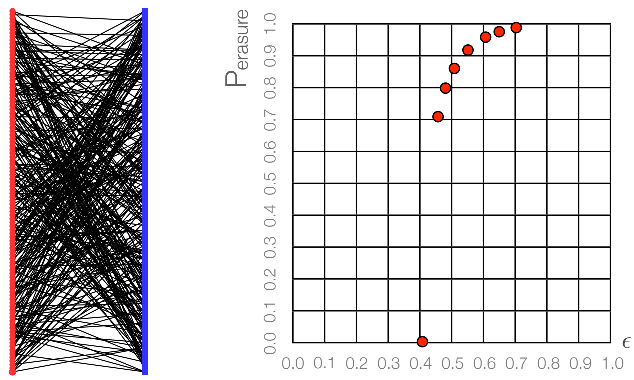
How does BP perform on the BEC?



(3, 6) ensemble

Here is the experiment we consider. Fix the ensemble. In the above example it is the (3, 6)-regular ensemble. This will serve as our running example. Now pick very long instances of this ensemble. Pick a random codeword and transmit over a BEC with erasure probability eps. Run the BP decoder until it no longer makes any progress. Record the error probability and average over many instances. Plot the average bit-error probability versus eps. Naturally, as eps decreases the error probability decreases. What is most interesting is that at some specific point we see a jump of the error probability from a non-zero value down to zero. This is the BP “threshold.”In the next slides, we will explain how to locate the BP threshold.

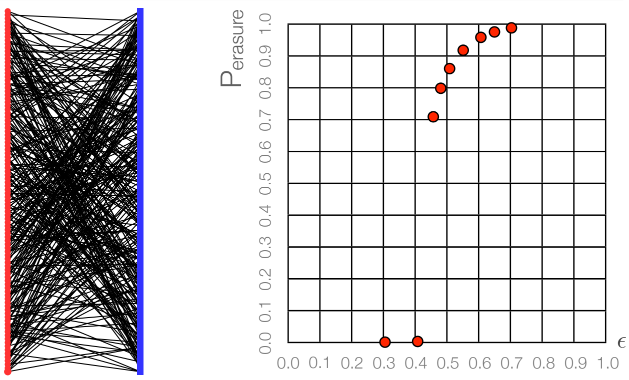
How does BP perform on the BEC?



(3, 6) ensemble

Here is the experiment we consider. Fix the ensemble. In the above example it is the (3, 6)-regular ensemble. This will serve as our running example. Now pick very long instances of this ensemble. Pick a random codeword and transmit over a BEC with erasure probability eps. Run the BP decoder until it no longer makes any progress. Record the error probability and average over many instances. Plot the average bit-error probability versus eps. Naturally, as eps decreases the error probability decreases. What is most interesting is that at some specific point we see a jump of the error probability from a non-zero value down to zero. This is the BP “threshold.”In the next slides, we will explain how to locate the BP threshold.

How does BP perform on the BEC?



(3, 6) ensemble

Here is the experiment we consider. Fix the ensemble. In the above example it is the (3, 6)-regular ensemble. This will serve as our running example. Now pick very long instances of this ensemble. Pick a random codeword and transmit over a BEC with erasure probability eps. Run the BP decoder until it no longer makes any progress. Record the error probability and average over many instances. Plot the average bit-error probability versus eps. Naturally, as eps decreases the error probability decreases. What is most interesting is that at some specific point we see a jump of the error probability from a non-zero value down to zero. This is the BP “threshold.”In the next slides, we will explain how to locate the BP threshold.

Asymptotic Analysis - Density Evolution (DE)

one iteration of BP at variable node

one iteration of BP at check node

Let us analyze how the erasure probability behaves during the decoding process for a large code. We do this by looking how the erasure probability behaves at each of the two types of the nodes. Consider a (d_l, d_r)-regular code, i.e., every variable node has degree d_l and and every check node has degree d_r.

At the variable node, if there is an incoming message which is not an erasure, then the variable node is exactly determined. This is because we are transmitting over the BEC and either we have perfect information or we have absolutely useless information. On the check node side, even if one incoming message is in erasure, the check node output has no way knowing whether it is 0 or 1 and hence the

Asymptotic Analysis - Density Evolution (DE)

one iteration of BP at variable node

one iteration of BP at check node

Let us analyze how the erasure probability behaves during the decoding process for a large code. We do this by looking how the erasure probability behaves at each of the two types of the nodes. Consider a (d_l, d_r)-regular code, i.e., every variable node has degree d_l and and every check node has degree d_r.

At the variable node, if there is an incoming message which is not an erasure, then the variable node is exactly determined. This is because we are transmitting over the BEC and either we have perfect information or we have absolutely useless information. On the check node side, even if one incoming message is in erasure, the check node output has no way knowing whether it is 0 or 1 and hence the

Asymptotic Analysis - Density Evolution (DE)

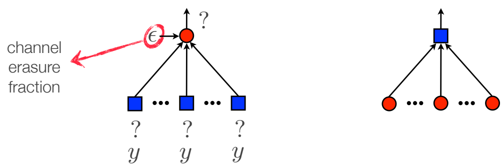
one iteration of BP at variable node

one iteration of BP at check node

Let us analyze how the erasure probability behaves during the decoding process for a large code. We do this by looking how the erasure probability behaves at each of the two types of the nodes. Consider a (d_l, d_r)-regular code, i.e., every variable node has degree d_l and and every check node has degree d_r.

At the variable node, if there is an incoming message which is not an erasure, then the variable node is exactly determined. This is because we are transmitting over the BEC and either we have perfect information or we have absolutely useless information. On the check node side, even if one incoming message is in erasure, the check node output has no way knowing whether it is 0 or 1 and hence the

Asymptotic Analysis - Density Evolution (DE)



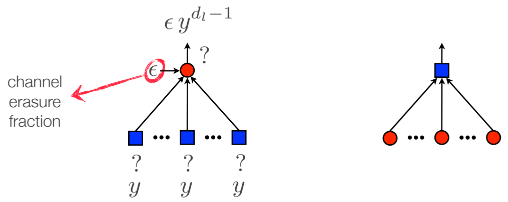
one iteration of BP at variable node

one iteration of BP at check node

Let us analyze how the erasure probability behaves during the decoding process for a large code. We do this by looking how the erasure probability behaves at each of the two types of the nodes. Consider a (d_l, d_r)-regular code, i.e., every variable node has degree d_l and and every check node has degree d_r.

At the variable node, if there is an incoming message which is not an erasure, then the variable node is exactly determined. This is because we are transmitting over the BEC and either we have perfect information or we have absolutely useless information. On the check node side, even if one incoming message is in erasure, the check node output has no way knowing whether it is 0 or 1 and hence the

Asymptotic Analysis - Density Evolution (DE)



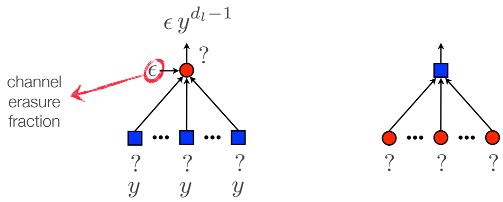
one iteration of BP at variable node

one iteration of BP at check node

Let us analyze how the erasure probability behaves during the decoding process for a large code. We do this by looking how the erasure probability behaves at each of the two types of the nodes. Consider a (d_l, d_r)-regular code, i.e., every variable node has degree d_l and and every check node has degree d_r.

At the variable node, if there is an incoming message which is not an erasure, then the variable node is exactly determined. This is because we are transmitting over the BEC and either we have perfect information or we have absolutely useless information. On the check node side, even if one incoming message is in erasure, the check node output has no way knowing whether it is 0 or 1 and hence the

Asymptotic Analysis - Density Evolution (DE)



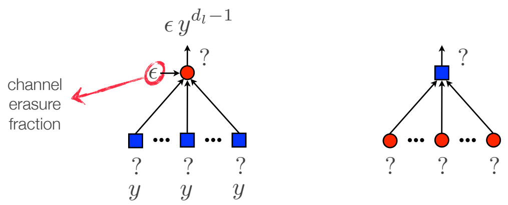
one iteration of BP at variable node

one iteration of BP at check node

Let us analyze how the erasure probability behaves during the decoding process for a large code. We do this by looking how the erasure probability behaves at each of the two types of the nodes. Consider a (d_l, d_r)-regular code, i.e., every variable node has degree d_l and and every check node has degree d_r.

At the variable node, if there is an incoming message which is not an erasure, then the variable node is exactly determined. This is because we are transmitting over the BEC and either we have perfect information or we have absolutely useless information. On the check node side, even if one incoming message is in erasure, the check node output has no way knowing whether it is 0 or 1 and hence the

Asymptotic Analysis - Density Evolution (DE)



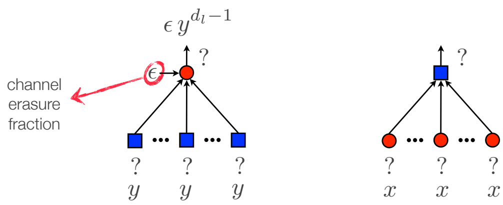
one iteration of BP at variable node

one iteration of BP at check node

Let us analyze how the erasure probability behaves during the decoding process for a large code. We do this by looking how the erasure probability behaves at each of the two types of the nodes. Consider a (d_l, d_r)-regular code, i.e., every variable node has degree d_l and and every check node has degree d_r.

At the variable node, if there is an incoming message which is not an erasure, then the variable node is exactly determined. This is because we are transmitting over the BEC and either we have perfect information or we have absolutely useless information. On the check node side, even if one incoming message is in erasure, the check node output has no way knowing whether it is 0 or 1 and hence the

Asymptotic Analysis - Density Evolution (DE)



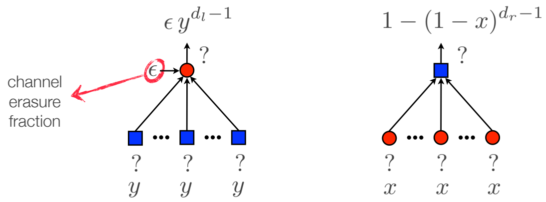
one iteration of BP at variable node

one iteration of BP at check node

Let us analyze how the erasure probability behaves during the decoding process for a large code. We do this by looking how the erasure probability behaves at each of the two types of the nodes. Consider a (d_l, d_r)-regular code, i.e., every variable node has degree d_l and and every check node has degree d_r.

At the variable node, if there is an incoming message which is not an erasure, then the variable node is exactly determined. This is because we are transmitting over the BEC and either we have perfect information or we have absolutely useless information. On the check node side, even if one incoming message is in erasure, the check node output has no way knowing whether it is 0 or 1 and hence the

Asymptotic Analysis - Density Evolution (DE)



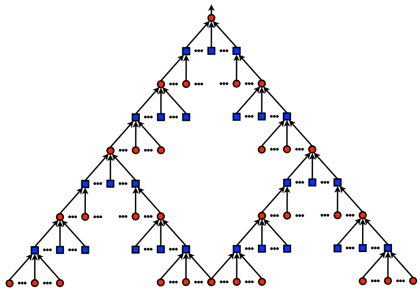
one iteration
of BP at variable
node

one iteration
of BP at check
node

Let us analyze how the erasure probability behaves during the decoding process for a large code. We do this by looking how the erasure probability behaves at each of the two types of the nodes. Consider a (d_l, d_r)-regular code, i.e., every variable node has degree d_l and and every check node has degree d_r.

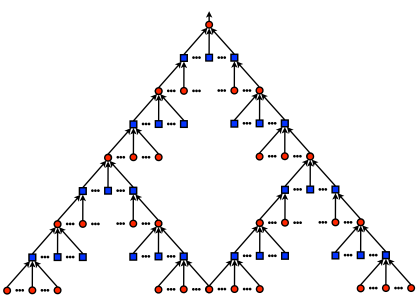
At the variable node, if there is an incoming message which is not an erasure, then the variable node is exactly determined. This is because we are transmitting over the BEC and either we have perfect information or we have absolutely useless information. On the check node side, even if one incoming message is in erasure, the check node output has no way knowing whether it is 0 or 1 and hence the

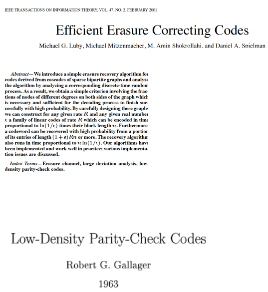
Asymptotic Analysis - Density Evolution (DE)



So if we perform l iterations we get a sequence of erasure probabilities. This is how Gallager analysed LDPC codes. Luby et. al. used a somewhat different procedure. In their analysis they look at the so-called peeling decoder. This decoder is entirely equivalent to the BP decoder (when transmitting over the BEC). In this decoder, as long as there is a degree-one check node, we use this check node to determine one more bit and then remove the used check node as well as the determine variable. We then follow the evolution of the graph. This can be done by writing down a system of differential equations. This method is called the Wormald method (["Differential Equations for Random Processes and Random Graphs", N. Wormald, Ann. Appl. Probability, Vol. 5, Pg. 1217-1235]).

Asymptotic Analysis - Density Evolution (DE)





IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 47, NO. 2, FEBRUARY 2001

Efficient Erasure Correcting Codes

Michael G. Luby, Michael Mitzenmacher, M. Amin Shokrollahi, and Daniel A. Spielman

Abstract—We introduce a simple erasure recovery algorithm fo codes derived from cascades of sparse bipartite graphs and analyz the algorithm by analyzing a corresponding discrete-time randon process. As a result, we obtain a simple criterion involving the frac tions of nodes of different degrees on both sides of the graph whicl is necessary and sufficient for the decoding process to finish suc cessfully with high probability. By carefully designing these graph we can construct for any given rate R and any given real numbe ϵ a family of linear codes of rate R which can be encoded in tim proportional to $\ln(1/\epsilon)$ times their block length n. Furthermore a codeword can be recovered with high probability from a portion of its entries of length $(1+\epsilon)Rn$ or more. The recovery algorithn also runs in time proportional to $n\ln(1/\epsilon)$. Our algorithms hav been implemented and work well in practice; various implementa tion issues are discussed.

Index Terms—Erasure channel, large deviation analysis, low density parity-check codes.

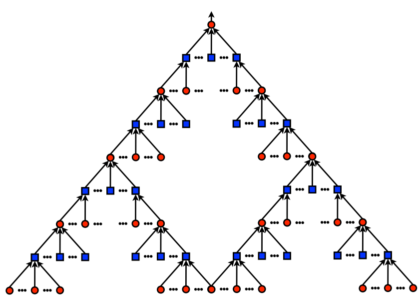
Low-Density Parity-Check Codes

Robert G. Gallager

1963

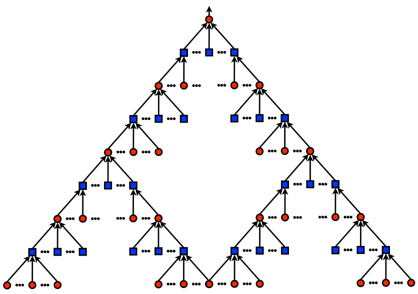
So if we perform l iterations we get a sequence of erasure probabilities. This is how Gallager analysed LDPC codes. Luby et. al. used a somewhat different procedure. In their analysis they look at the so-called peeling decoder. This decoder is entirely equivalent to the BP decoder (when transmitting over the BEC). In this decoder, as long as there is a degree-one check node, we use this check node to determine one more bit and then remove the used check node as well as the determine variable. We then follow the evolution of the graph. This can be done by writing down a system of differential equations. This method is called the Wormald method (["Differential Equations for Random Processes and Random Graphs", N. Wormald, Ann. Appl. Probability, Vol. 5, Pg. 1217-1235]).

Asymptotic Analysis - Density Evolution (DE)



So if we perform l iterations we get a sequence of erasure probabilities. This is how Gallager analysed LDPC codes. Luby et. al. used a somewhat different procedure. In their analysis they look at the so-called peeling decoder. This decoder is entirely equivalent to the BP decoder (when transmitting over the BEC). In this decoder, as long as there is a degree-one check node, we use this check node to determine one more bit and then remove the used check node as well as the determine variable. We then follow the evolution of the graph. This can be done by writing down a system of differential equations. This method is called the Wormald method (["Differential Equations for Random Processes and Random Graphs", N. Wormald, Ann. Appl. Probability, Vol. 5, Pg. 1217-1235]).

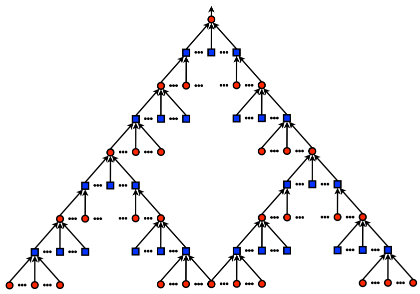
Asymptotic Analysis - Density Evolution (DE)



$$x^{(\ell=0)} = \epsilon$$

So if we perform l iterations we get a sequence of erasure probabilities. This is how Gallager analysed LDPC codes. Luby et. al. used a somewhat different procedure. In their analysis they look at the so-called peeling decoder. This decoder is entirely equivalent to the BP decoder (when transmitting over the BEC). In this decoder, as long as there is a degree-one check node, we use this check node to determine one more bit and then remove the used check node as well as the determine variable. We then follow the evolution of the graph. This can be done by writing down a system of differential equations. This method is called the Wormald method (["Differential Equations for Random Processes and Random Graphs", N. Wormald, Ann. Appl. Probability, Vol. 5, Pg. 1217-1235]).

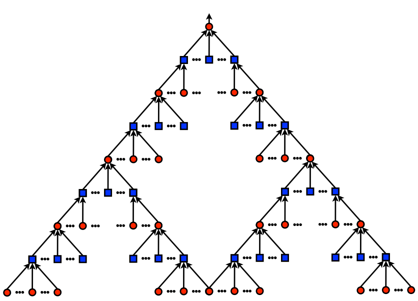
Asymptotic Analysis - Density Evolution (DE)



$$y^{(\ell=1)} = 1 - (1 - x^{(\ell=0)})^{d_r - 1}$$
$$x^{(\ell=0)} = \epsilon$$

So if we perform l iterations we get a sequence of erasure probabilities. This is how Gallager analysed LDPC codes. Luby et. al. used a somewhat different procedure. In their analysis they look at the so-called peeling decoder. This decoder is entirely equivalent to the BP decoder (when transmitting over the BEC). In this decoder, as long as there is a degree-one check node, we use this check node to determine one more bit and then remove the used check node as well as the determine variable. We then follow the evolution of the graph. This can be done by writing down a system of differential equations. This method is called the Wormald method (["Differential Equations for Random Processes and Random Graphs", N. Wormald, Ann. Appl. Probability, Vol. 5, Pg. 1217-1235]).

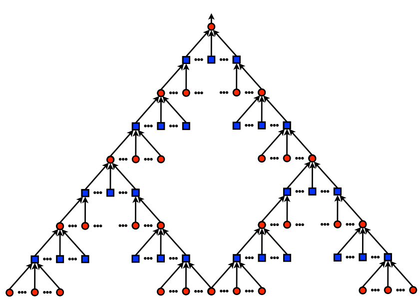
Asymptotic Analysis - Density Evolution (DE)



$$x^{(\ell=1)} = \epsilon (y^{(\ell=1)})^{d_l - 1}$$
$$y^{(\ell=1)} = 1 - (1 - x^{(\ell=0)})^{d_r - 1}$$
$$x^{(\ell=0)} = \epsilon$$

So if we perform l iterations we get a sequence of erasure probabilities. This is how Gallager analysed LDPC codes. Luby et. al. used a somewhat different procedure. In their analysis they look at the so-called peeling decoder. This decoder is entirely equivalent to the BP decoder (when transmitting over the BEC). In this decoder, as long as there is a degree-one check node, we use this check node to determine one more bit and then remove the used check node as well as the determine variable. We then follow the evolution of the graph. This can be done by writing down a system of differential equations. This method is called the Wormald method (["Differential Equations for Random Processes and Random Graphs", N. Wormald, Ann. Appl. Probability, Vol. 5, Pg. 1217-1235]).

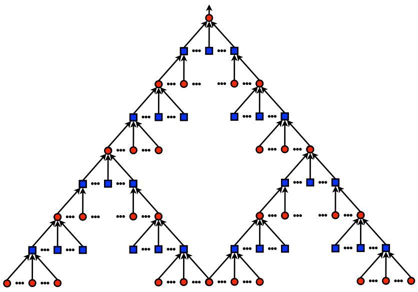
Asymptotic Analysis - Density Evolution (DE)



$$y^{(\ell=2)} = 1 - (1 - x^{(\ell=1)})^{d_r - 1}$$
$$x^{(\ell=1)} = \epsilon (y^{(\ell=1)})^{d_l - 1}$$
$$y^{(\ell=1)} = 1 - (1 - x^{(\ell=0)})^{d_r - 1}$$
$$x^{(\ell=0)} = \epsilon$$

So if we perform l iterations we get a sequence of erasure probabilities. This is how Gallager analysed LDPC codes. Luby et. al. used a somewhat different procedure. In their analysis they look at the so-called peeling decoder. This decoder is entirely equivalent to the BP decoder (when transmitting over the BEC). In this decoder, as long as there is a degree-one check node, we use this check node to determine one more bit and then remove the used check node as well as the determine variable. We then follow the evolution of the graph. This can be done by writing down a system of differential equations. This method is called the Wormald method (["Differential Equations for Random Processes and Random Graphs", N. Wormald, Ann. Appl. Probability, Vol. 5, Pg. 1217-1235]).

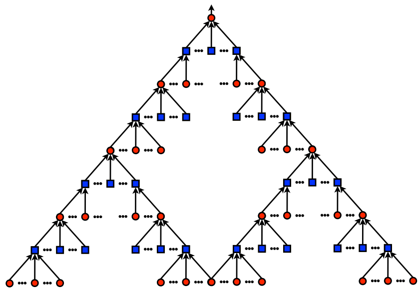
Asymptotic Analysis - Density Evolution (DE)



$$x^{(\ell=2)} = \epsilon(y^{(\ell=2)})^{d_l-1}$$
$$y^{(\ell=2)} = 1-(1-x^{(\ell=1)})^{d_r-1}$$
$$x^{(\ell=1)} = \epsilon(y^{(\ell=1)})^{d_l-1}$$
$$y^{(\ell=1)} = 1-(1-x^{(\ell=0)})^{d_r-1}$$
$$x^{(\ell=0)} = \epsilon$$

So if we perform l iterations we get a sequence of erasure probabilities. This is how Gallager analysed LDPC codes. Luby et. al. used a somewhat different procedure. In their analysis they look at the so-called peeling decoder. This decoder is entirely equivalent to the BP decoder (when transmitting over the BEC). In this decoder, as long as there is a degree-one check node, we use this check node to determine one more bit and then remove the used check node as well as the determine variable. We then follow the evolution of the graph. This can be done by writing down a system of differential equations. This method is called the Wormald method (["Differential Equations for Random Processes and Random Graphs", N. Wormald, Ann. Appl. Probability, Vol. 5, Pg. 1217-1235]).

Asymptotic Analysis - Density Evolution (DE)



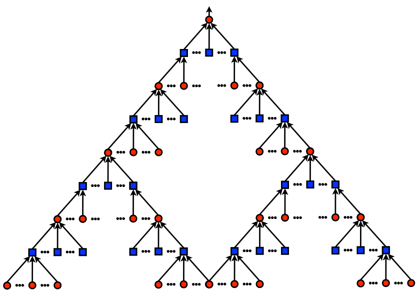
$$y^{(\ell)} = 1-(1-x^{(\ell-1)})^{d_r-1}$$

$$x^{(\ell=2)} = \epsilon(y^{(\ell=2)})^{d_l-1}$$
$$y^{(\ell=2)} = 1-(1-x^{(\ell=1)})^{d_r-1}$$
$$x^{(\ell=1)} = \epsilon(y^{(\ell=1)})^{d_l-1}$$
$$y^{(\ell=1)} = 1-(1-x^{(\ell=0)})^{d_r-1}$$
$$x^{(\ell=0)} = \epsilon$$

So if we perform l iterations we get a sequence of erasure probabilities. This is how Gallager analysed LDPC codes. Luby et. al. used a somewhat different procedure. In their analysis they look at the so-called peeling decoder. This decoder is entirely equivalent to the BP decoder (when transmitting over the BEC). In this decoder, as long as there is a degree-one check node, we use this check node to determine one more bit and then remove the used check node as well as the determine variable. We then follow the evolution of the graph. This can be done by writing down a system of differential equations. This method is called the Wormald method (["Differential Equations for Random Processes and Random Graphs", N. Wormald, Ann. Appl. Probability, Vol. 5, Pg. 1217-1235]).

Asymptotic Analysis - Density Evolution (DE)

erasure fraction
at the root after
ℓ iterations



$$x^{(\ell)} = \epsilon(y^{(\ell)})^{d_l-1}$$
$$y^{(\ell)} = 1-(1-x^{(\ell-1)})^{d_r-1}$$

$$x^{(\ell=2)} = \epsilon(y^{(\ell=2)})^{d_l-1}$$
$$y^{(\ell=2)} = 1-(1-x^{(\ell=1)})^{d_r-1}$$
$$x^{(\ell=1)} = \epsilon(y^{(\ell=1)})^{d_l-1}$$
$$y^{(\ell=1)} = 1-(1-x^{(\ell=0)})^{d_r-1}$$
$$x^{(\ell=0)} = \epsilon$$

So if we perform l iterations we get a sequence of erasure probabilities. This is how Gallager analysed LDPC codes. Luby et. al. used a somewhat different procedure. In their analysis they look at the so-called peeling decoder. This decoder is entirely equivalent to the BP decoder (when transmitting over the BEC). In this decoder, as long as there is a degree-one check node, we use this check node to determine one more bit and then remove the used check node as well as the determine variable. We then follow the evolution of the graph. This can be done by writing down a system of differential equations. This method is called the Wormald method (["Differential Equations for Random Processes and Random Graphs", N. Wormald, Ann. Appl. Probability, Vol. 5, Pg. 1217-1235]).

Asymptotic Analysis - Density Evolution (DE)

Note: DE sequence is decreasing and bounded from below ⇒ converges

Note that in the density evolution approach we assume that we first fix the number of iterations and let the length of the code tend to infinity (so that there are no loops in the graph up to the desired size). We THEN let the number of iterations tend to infinity. In the Wormald approach on the other hand we take exactly the opposite limit. Luckily both approaches give exactly the same threshold. It is then easy to see that in fact we can take the limit in any order, or jointly, and that we always get the same threshold.

Asymptotic Analysis - Density Evolution (DE)

Note: DE sequence is decreasing and bounded from below ⇒ converges

DE corresponds to the limit

$$\lim_{\ell \to \infty} \lim_{n \to \infty}$$

For the BEC it is easy to prove that result is always the same regardless of how the limits are taken

Note that in the density evolution approach we assume that we first fix the number of iterations and let the length of the code tend to infinity (so that there are no loops in the graph up to the desired size). We THEN let the number of iterations tend to infinity. In the Wormald approach on the other hand we take exactly the opposite limit. Luckily both approaches give exactly the same threshold. It is then easy to see that in fact we can take the limit in any order, or jointly, and that we always get the same threshold.

Asymptotic Analysis - Density Evolution (DE)

Note: DE sequence is decreasing and bounded from below ⇒ converges

DE corresponds to the limit

$$\lim_{\ell \to \infty} \lim_{n \to \infty}$$

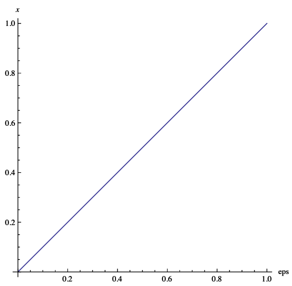
For the BEC it is easy to prove that result is always the same regardless of how the limits are taken

Concentration: It can be shown that the behavior of almost all codes in the ensemble is close to the average predicted by DE

Note that in the density evolution approach we assume that we first fix the number of iterations and let the length of the code tend to infinity (so that there are no loops in the graph up to the desired size). We THEN let the number of iterations tend to infinity. In the Wormald approach on the other hand we take exactly the opposite limit. Luckily both approaches give exactly the same threshold. It is then easy to see that in fact we can take the limit in any order, or jointly, and that we always get the same threshold.

DE for (3, 6) Ensemble

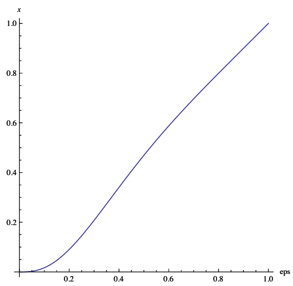
erasure fraction as a function of increasing iterations for a given channel value



Let us now apply DE for our running example. We see that up to the “BP threshold”, which for the running example is around 0.429, the erasure probability tends to zero if we let the number of iterations tend to infinity. For higher values of eps the x-value tends to a non-zero value.

DE for (3, 6) Ensemble

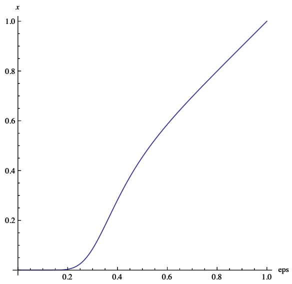
erasure fraction as a function of increasing iterations for a given channel value



Let us now apply DE for our running example. We see that up to the “BP threshold”, which for the running example is around 0.429, the erasure probability tends to zero if we let the number of iterations tend to infinity. For higher values of eps the x-value tends to a non-zero value.

DE for (3, 6) Ensemble

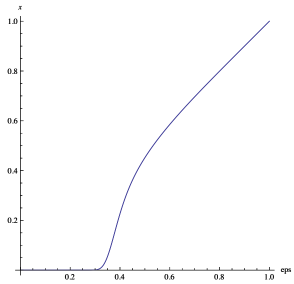
erasure fraction as a function of increasing iterations for a given channel value



Let us now apply DE for our running example. We see that up to the “BP threshold”, which for the running example is around 0.429, the erasure probability tends to zero if we let the number of iterations tend to infinity. For higher values of eps the x-value tends to a non-zero value.

DE for (3, 6) Ensemble

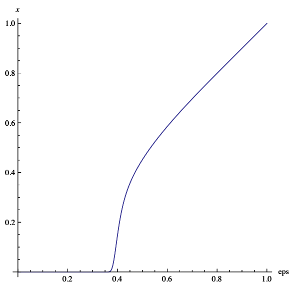
erasure fraction as a function of increasing iterations for a given channel value



Let us now apply DE for our running example. We see that up to the “BP threshold”, which for the running example is around 0.429, the erasure probability tends to zero if we let the number of iterations tend to infinity. For higher values of eps the x-value tends to a non-zero value.

DE for (3, 6) Ensemble

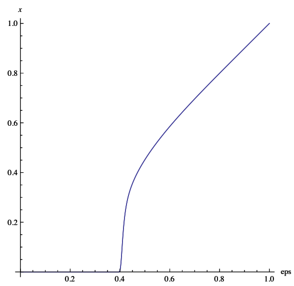
erasure fraction as a function of increasing iterations for a given channel value



Let us now apply DE for our running example. We see that up to the “BP threshold”, which for the running example is around 0.429, the erasure probability tends to zero if we let the number of iterations tend to infinity. For higher values of eps the x-value tends to a non-zero value.

DE for (3, 6) Ensemble

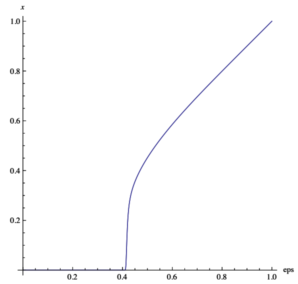
erasure fraction as a function of increasing iterations for a given channel value



Let us now apply DE for our running example. We see that up to the “BP threshold”, which for the running example is around 0.429, the erasure probability tends to zero if we let the number of iterations tend to infinity. For higher values of eps the x-value tends to a non-zero value.

DE for (3, 6) Ensemble

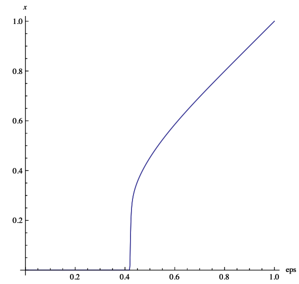
erasure fraction as a function of increasing iterations for a given channel value



Let us now apply DE for our running example. We see that up to the "BP threshold", which for the running example is around 0.429, the erasure probability tends to zero if we let the number of iterations tend to infinity. For higher values of eps the x-value tends to a non-zero value.

DE for (3, 6) Ensemble

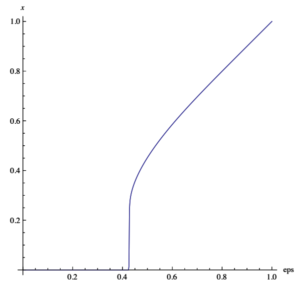
erasure fraction as a function of increasing iterations for a given channel value



Let us now apply DE for our running example. We see that up to the "BP threshold", which for the running example is around 0.429, the erasure probability tends to zero if we let the number of iterations tend to infinity. For higher values of eps the x-value tends to a non-zero value.

DE for (3, 6) Ensemble

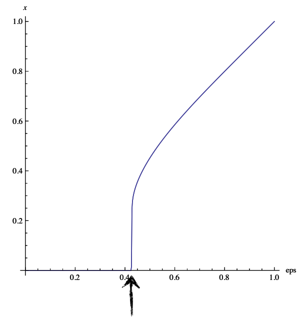
erasure fraction as a function of increasing iterations for a given channel value



Let us now apply DE for our running example. We see that up to the "BP threshold", which for the running example is around 0.429, the erasure probability tends to zero if we let the number of iterations tend to infinity. For higher values of eps the x-value tends to a non-zero value.

DE for (3, 6) Ensemble

erasure fraction as a function of increasing iterations for a given channel value

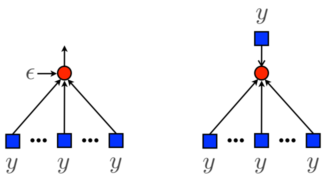


BP successful below $\epsilon \approx 0.429$

Let us now apply DE for our running example. We see that up to the "BP threshold", which for the running example is around 0.429, the erasure probability tends to zero if we let the number of iterations tend to infinity. For higher values of eps the x-value tends to a non-zero value.

x versus EXIT

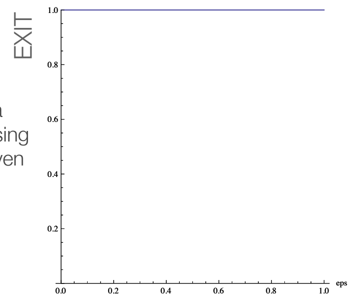
$$x = \epsilon y^{d_l - 1} \qquad \mathrm{EXIT} = y^{d_l}$$



Instead of plotting the “x-value” on the vertical axis it is often more convenient to plot the EXIT value. The EXIT value has a simple interpretation. It is the error probability of the best estimate we can do using all the “internal” messages at a node but without the channel observation at this bit. This is why we have y to the power dl and not dl-1 but we do not have the factor eps corresponding to the channel erasure fraction.

EXIT Curve for (3, 6) Ensemble

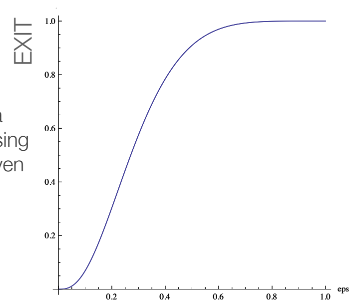
EXIT value as a function of increasing iterations for a given channel value



We now repeat the previous experiment but we plot the EXIT value instead of the x-value on the vertical axis.

EXIT Curve for (3, 6) Ensemble

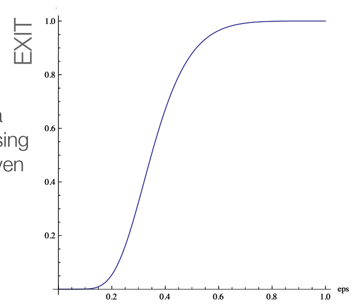
EXIT value as a function of increasing iterations for a given channel value



We now repeat the previous experiment but we plot the EXIT value instead of the x-value on the vertical axis.

EXIT Curve for (3, 6) Ensemble

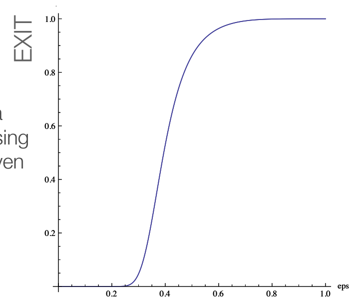
EXIT value as a function of increasing iterations for a given channel value



We now repeat the previous experiment but we plot the EXIT value instead of the x-value on the vertical axis.

EXIT Curve for (3, 6) Ensemble

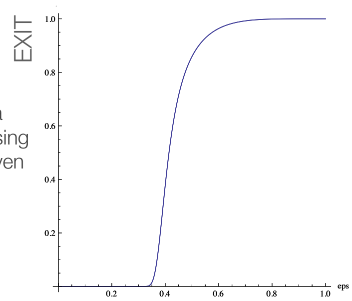
EXIT value as a function of increasing iterations for a given channel value



We now repeat the previous experiment but we plot the EXIT value instead of the x-value on the vertical axis.

EXIT Curve for (3, 6) Ensemble

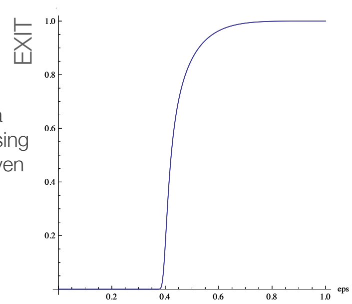
EXIT value as a function of increasing iterations for a given channel value



We now repeat the previous experiment but we plot the EXIT value instead of the x-value on the vertical axis.

EXIT Curve for (3, 6) Ensemble

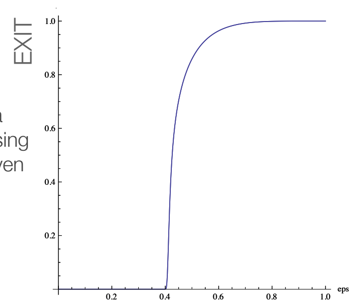
EXIT value as a function of increasing iterations for a given channel value



We now repeat the previous experiment but we plot the EXIT value instead of the x-value on the vertical axis.

EXIT Curve for (3, 6) Ensemble

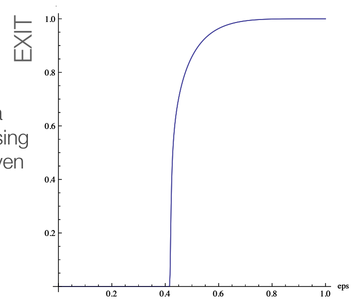
EXIT value as a function of increasing iterations for a given channel value



We now repeat the previous experiment but we plot the EXIT value instead of the x-value on the vertical axis.

EXIT Curve for (3, 6) Ensemble

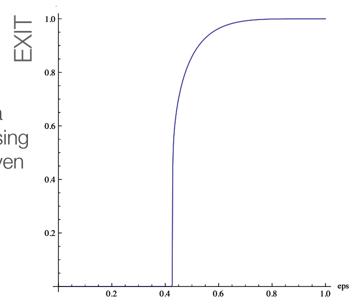
EXIT value as a function of increasing iterations for a given channel value



We now repeat the previous experiment but we plot the EXIT value instead of the x-value on the vertical axis.

EXIT Curve for (3, 6) Ensemble

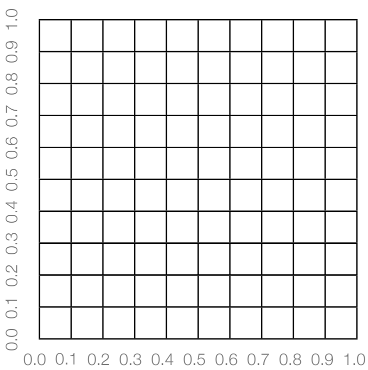
EXIT value as a function of increasing iterations for a given channel value



Saturday, July 13, 13 20

We now repeat the previous experiment but we plot the EXIT value instead of the x-value on the vertical axis.

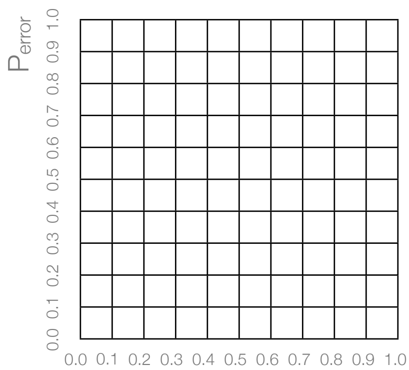
A look back ...



(3, 6) ensemble

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the “right” quantity to plot.

A look back ...

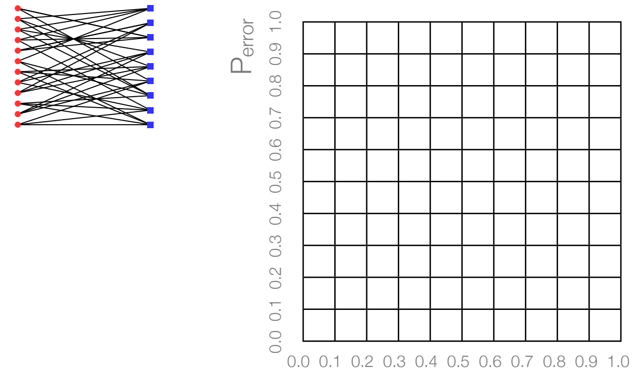


(3, 6) ensemble

Saturday, July 13, 13 21

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the “right” quantity to plot.

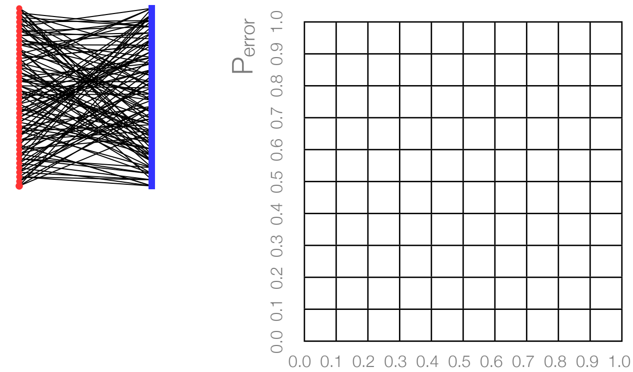
A look back ...



(3, 6) ensemble

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the “right” quantity to plot.

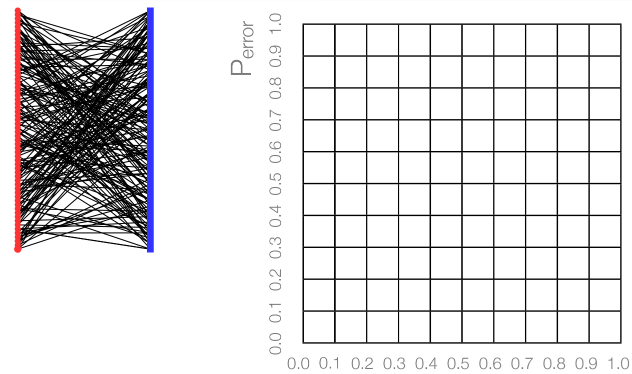
A look back ...



(3, 6) ensemble

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the “right” quantity to plot.

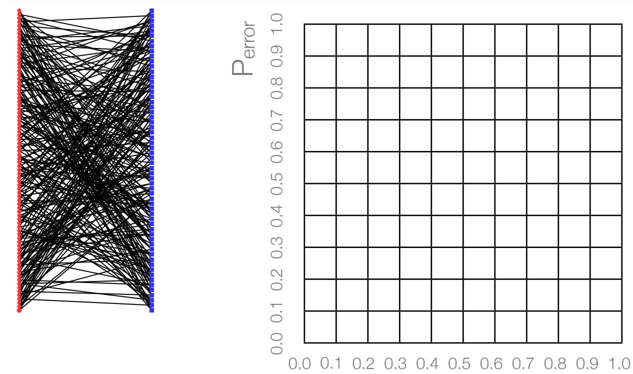
A look back ...



(3, 6) ensemble

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the “right” quantity to plot.

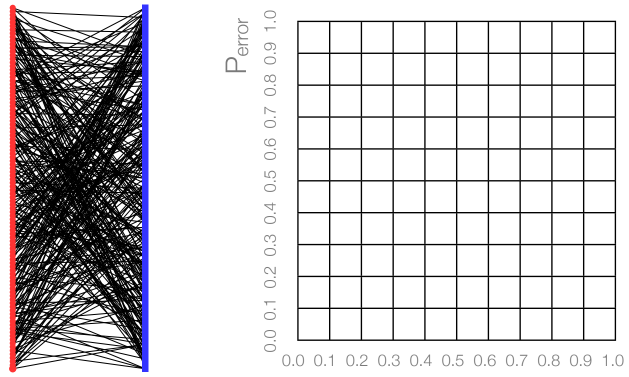
A look back ...



(3, 6) ensemble

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the “right” quantity to plot.

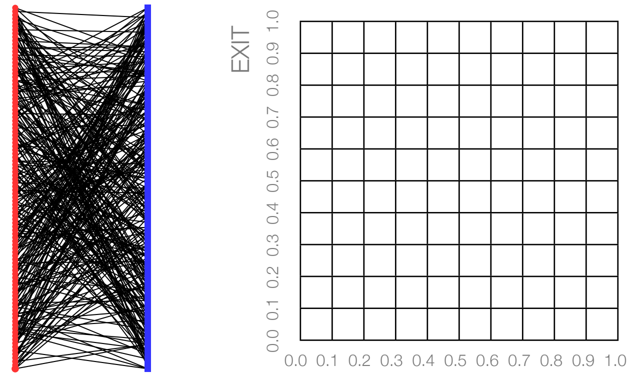
A look back ...



(3, 6) ensemble

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the “right” quantity to plot.

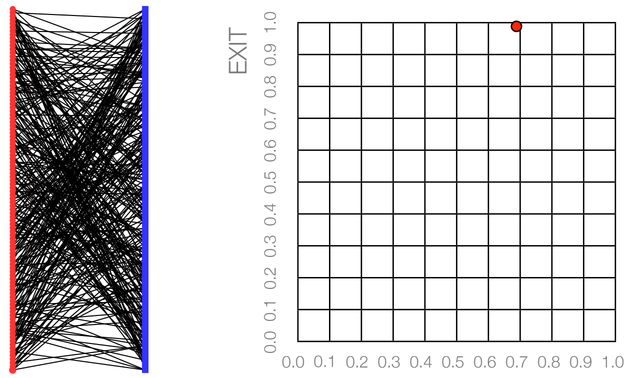
A look back ...



(3, 6) ensemble

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the “right” quantity to plot.

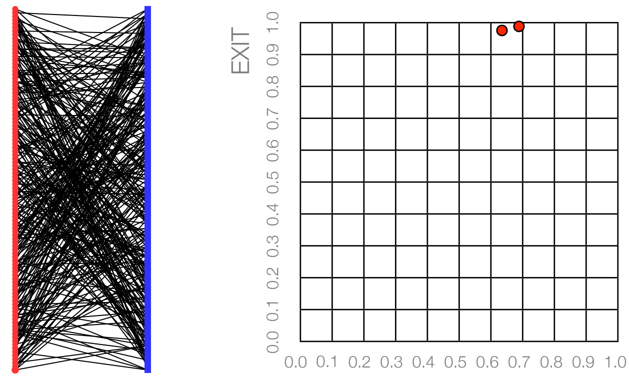
A look back ...



(3, 6) ensemble

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the “right” quantity to plot.

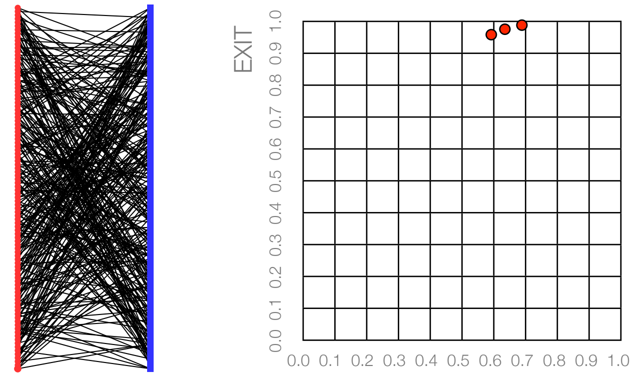
A look back ...



(3, 6) ensemble

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the “right” quantity to plot.

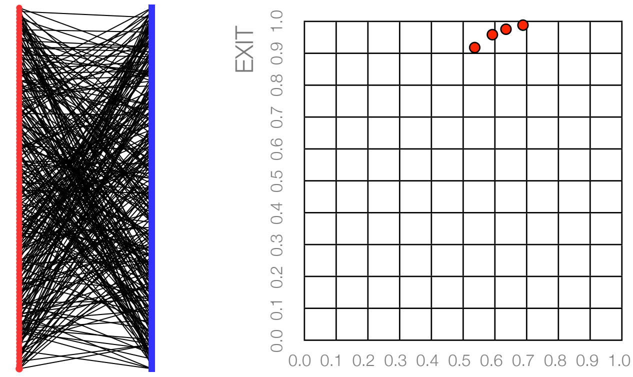
A look back ...



(3, 6) ensemble

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the “right” quantity to plot.

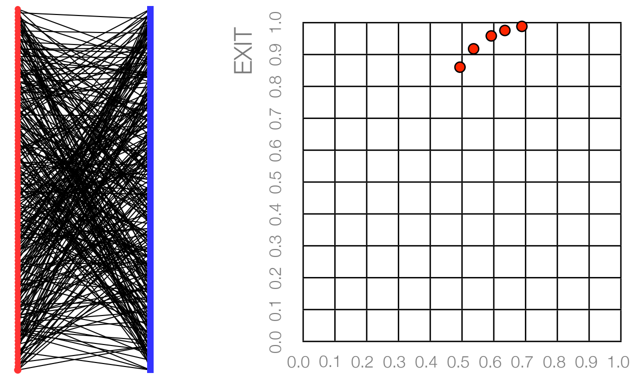
A look back ...



(3, 6) ensemble

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the “right” quantity to plot.

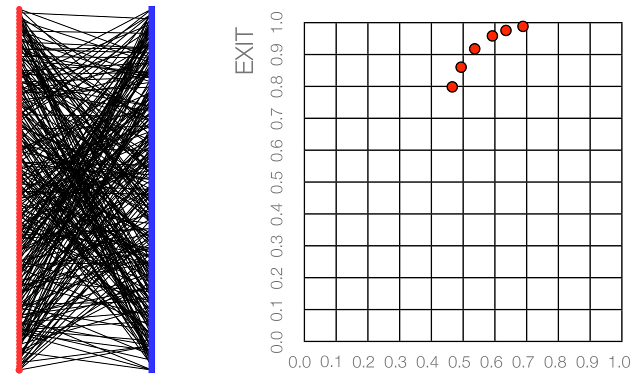
A look back ...



(3, 6) ensemble

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the “right” quantity to plot.

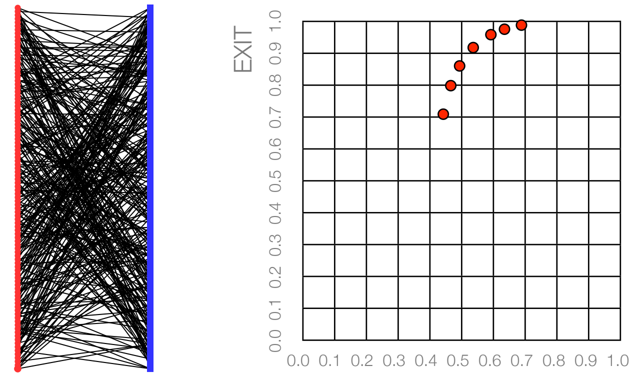
A look back ...



(3, 6) ensemble

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the “right” quantity to plot.

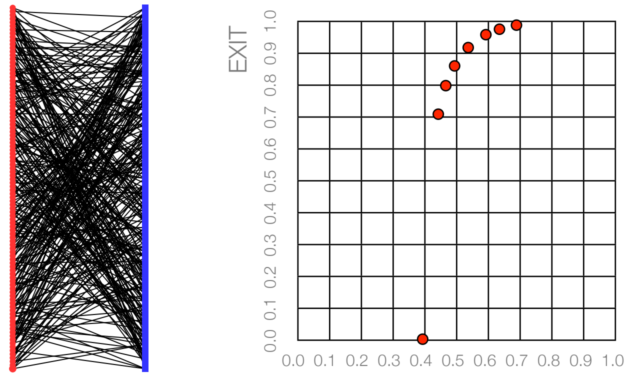
A look back ...



(3, 6) ensemble

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the "right" quantity to plot.

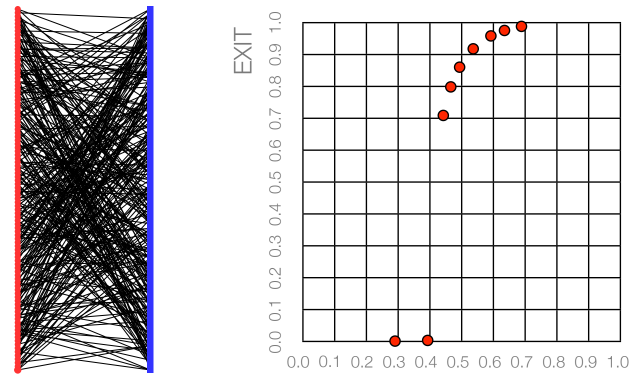
A look back ...



(3, 6) ensemble

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the "right" quantity to plot.

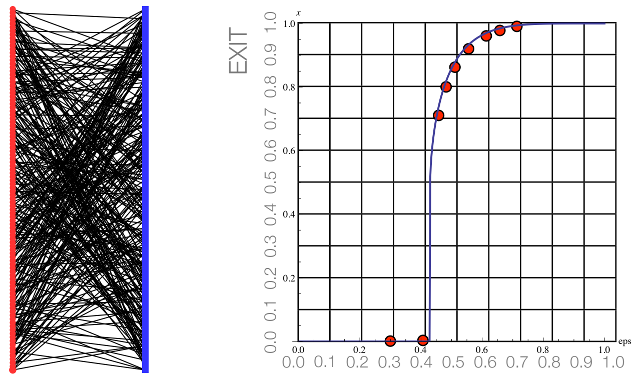
A look back ...



(3, 6) ensemble

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the "right" quantity to plot.

A look back ...



(3, 6) ensemble

Let us now go back to our first experiment. We see now that we can predict where the red dots will lie. In fact, in our original experiment we cheated slightly. We printed the EXIT value and not the bit error probability. These two only differ by a factor eps. We will see soon why the EXIT value is the "right" quantity to plot.

Static Analysis via Fixed points

Forward Fixed points of DE

$$x^{(\ell=0)} = \epsilon$$

$$y^{(\ell)} = 1 - (1 - x^{(\ell-1)})^{d_r - 1}$$

$$x^{(\ell)} = \epsilon (y^{(\ell)})^{d_l - 1}$$

Rather than running the recursion we can right away figure out the value to which the recursion converges. This is because this final value must be a solution to the fixed-point (FP) equation x=f(eps, x), where f() denotes the recursive DE equations. Note that there are in general several values of x which satisfies the FP equation for a given eps, but there is always just a single value of eps for a given x, which is easily seen by solving for eps from the FP equation above. This makes it easy to plot this curve. But note also that in this picture we have “additional” fixed points. These FPs are unstable and we cannot get them by running DE. But as we will see they nevertheless play an important role in the analysis.

Static Analysis via Fixed points

Forward Fixed points of DE

$$x^{(\ell=0)} = \epsilon$$

$$y^{(\ell)} = 1 - (1 - x^{(\ell-1)})^{d_r - 1}$$

$$x^{(\ell)} = \epsilon (y^{(\ell)})^{d_l - 1}$$

$$x = \epsilon (1 - (1 - x))^{d_r - 1})^{d_l - 1}$$

Saturday, July 13, 13 22

Rather than running the recursion we can right away figure out the value to which the recursion converges. This is because this final value must be a solution to the fixed-point (FP) equation x=f(eps, x), where f() denotes the recursive DE equations. Note that there are in general several values of x which satisfies the FP equation for a given eps, but there is always just a single value of eps for a given x, which is easily seen by solving for eps from the FP equation above. This makes it easy to plot this curve. But note also that in this picture we have “additional” fixed points. These FPs are unstable and we cannot get them by running DE. But as we will see they nevertheless play an important role in the analysis.

Static Analysis via Fixed points

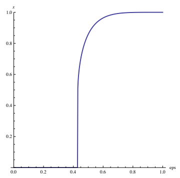
Forward Fixed points of DE

$$x^{(\ell=0)} = \epsilon$$

$$y^{(\ell)} = 1 - (1 - x^{(\ell-1)})^{d_r - 1}$$

$$x^{(\ell)} = \epsilon (y^{(\ell)})^{d_l - 1}$$

$$x = \epsilon (1 - (1 - x))^{d_r - 1})^{d_l - 1}$$



Saturday, July 13, 13 22

Rather than running the recursion we can right away figure out the value to which the recursion converges. This is because this final value must be a solution to the fixed-point (FP) equation x=f(eps, x), where f() denotes the recursive DE equations. Note that there are in general several values of x which satisfies the FP equation for a given eps, but there is always just a single value of eps for a given x, which is easily seen by solving for eps from the FP equation above. This makes it easy to plot this curve. But note also that in this picture we have “additional” fixed points. These FPs are unstable and we cannot get them by running DE. But as we will see they nevertheless play an important role in the analysis.

Static Analysis via Fixed points

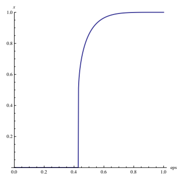
Forward Fixed points of DE

$$x^{(\ell=0)} = \epsilon$$

$$y^{(\ell)} = 1 - (1 - x^{(\ell-1)})^{d_r - 1}$$

$$x^{(\ell)} = \epsilon (y^{(\ell)})^{d_l - 1}$$

$$x = \epsilon (1 - (1 - x))^{d_r - 1})^{d_l - 1}$$



All Fixed points of DE

Saturday, July 13, 13 22

Rather than running the recursion we can right away figure out the value to which the recursion converges. This is because this final value must be a solution to the fixed-point (FP) equation x=f(eps, x), where f() denotes the recursive DE equations. Note that there are in general several values of x which satisfies the FP equation for a given eps, but there is always just a single value of eps for a given x, which is easily seen by solving for eps from the FP equation above. This makes it easy to plot this curve. But note also that in this picture we have “additional” fixed points. These FPs are unstable and we cannot get them by running DE. But as we will see they nevertheless play an important role in the analysis.

Static Analysis via Fixed points

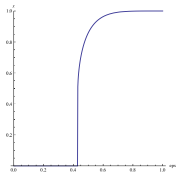
Forward Fixed points of DE

$$x^{(\ell=0)} = \epsilon$$

$$y^{(\ell)} = 1 - (1 - x^{(\ell-1)})^{d_r - 1}$$

$$x^{(\ell)} = \epsilon (y^{(\ell)})^{d_l - 1}$$

$$x = \epsilon (1 - (1 - x))^{d_r - 1})^{d_l - 1}$$



All Fixed points of DE

$$x = \epsilon (1 - (1 - x))^{d_r - 1})^{d_l - 1}$$

$$\epsilon = \frac{x}{(1 - (1 - x))^{d_r - 1})^{d_l - 1}}$$

Rather than running the recursion we can right away figure out the value to which the recursion converges. This is because this final value must be a solution to the fixed-point (FP) equation x=f(eps, x), where f() denotes the recursive DE equations. Note that there are in general several values of x which satisfies the FP equation for a given eps, but there is always just a single value of eps for a given x, which is easily seen by solving for eps from the FP equation above. This makes it easy to plot this curve. But note also that in this picture we have “additional” fixed points. These FPs are unstable and we cannot get them by running DE. But as we will see they nevertheless play an important role in the analysis.

Static Analysis via Fixed points

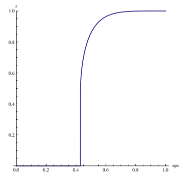
Forward Fixed points of DE

$$x^{(\ell=0)} = \epsilon$$

$$y^{(\ell)} = 1 - (1 - x^{(\ell-1)})^{d_r - 1}$$

$$x^{(\ell)} = \epsilon (y^{(\ell)})^{d_l - 1}$$

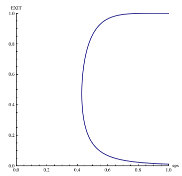
$$x = \epsilon (1 - (1 - x))^{d_r - 1})^{d_l - 1}$$



All Fixed points of DE

$$x = \epsilon (1 - (1 - x))^{d_r - 1})^{d_l - 1}$$

$$\epsilon = \frac{x}{(1 - (1 - x))^{d_r - 1})^{d_l - 1}}$$



Rather than running the recursion we can right away figure out the value to which the recursion converges. This is because this final value must be a solution to the fixed-point (FP) equation x=f(eps, x), where f() denotes the recursive DE equations. Note that there are in general several values of x which satisfies the FP equation for a given eps, but there is always just a single value of eps for a given x, which is easily seen by solving for eps from the FP equation above. This makes it easy to plot this curve. But note also that in this picture we have “additional” fixed points. These FPs are unstable and we cannot get them by running DE. But as we will see they nevertheless play an important role in the analysis.

Static Analysis via Fixed points

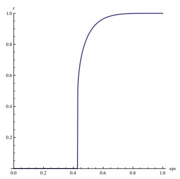
Forward Fixed points of DE

$$x^{(\ell=0)} = \epsilon$$

$$y^{(\ell)} = 1 - (1 - x^{(\ell-1)})^{d_r - 1}$$

$$x^{(\ell)} = \epsilon (y^{(\ell)})^{d_l - 1}$$

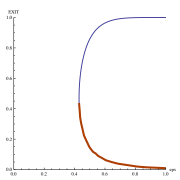
$$x = \epsilon (1 - (1 - x))^{d_r - 1})^{d_l - 1}$$



All Fixed points of DE

$$x = \epsilon (1 - (1 - x))^{d_r - 1})^{d_l - 1}$$

$$\epsilon = \frac{x}{(1 - (1 - x))^{d_r - 1})^{d_l - 1}}$$

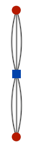


Rather than running the recursion we can right away figure out the value to which the recursion converges. This is because this final value must be a solution to the fixed-point (FP) equation x=f(eps, x), where f() denotes the recursive DE equations. Note that there are in general several values of x which satisfies the FP equation for a given eps, but there is always just a single value of eps for a given x, which is easily seen by solving for eps from the FP equation above. This makes it easy to plot this curve. But note also that in this picture we have “additional” fixed points. These FPs are unstable and we cannot get them by running DE. But as we will see they nevertheless play an important role in the analysis.

Coupled Codes

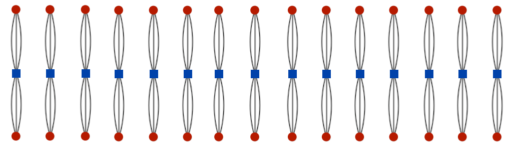
We now get to the topic of this tutorial, namely spatially coupled codes. We start with their construction and then proceed towards their analysis. We will see how these codes are intimately tied to the standard LDPC codes we have introduced till now.

Coupled Ensembles - Protograph Construction



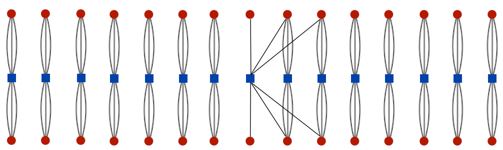
As for uncoupled codes, there are many flavors and variations on the theme. The exact version we use is not so important. They all behave more or less the same. For the purpose of this tutorial we consider two variations. Namely coupled ensembles defined by a protograph as well as a random version. In the protograph version, we connect neighboring copies in a regular fashion as described above. We stress that above we show the construction of the protograph of the coupled code and not the actual code which will be used for transmission. As mentioned before, for the real code, we need to “lift” the graph M times and then randomly permute edges in the same edge bundle. The “spatially coupled” qualifier for the codes comes about naturally since we consider protograph of standard LDPC codes, arrange them

Coupled Ensembles - Protograph Construction



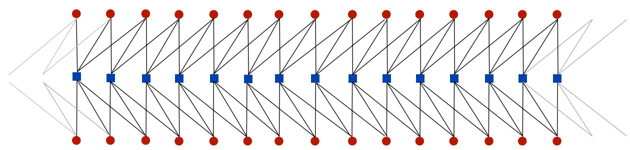
As for uncoupled codes, there are many flavors and variations on the theme. The exact version we use is not so important. They all behave more or less the same. For the purpose of this tutorial we consider two variations. Namely coupled ensembles defined by a protograph as well as a random version. In the protograph version, we connect neighboring copies in a regular fashion as described above. We stress that above we show the construction of the protograph of the coupled code and not the actual code which will be used for transmission. As mentioned before, for the real code, we need to “lift” the graph M times and then randomly permute edges in the same edge bundle. The “spatially coupled” qualifier for the codes comes about naturally since we consider protograph of standard LDPC codes, arrange them

Coupled Ensembles - Protograph Construction



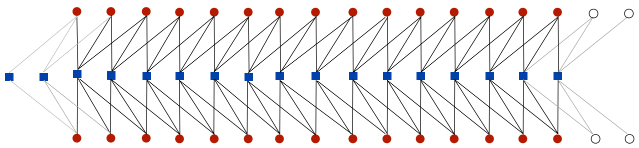
As for uncoupled codes, there are many flavors and variations on the theme. The exact version we use is not so important. They all behave more or less the same. For the purpose of this tutorial we consider two variations. Namely coupled ensembles defined by a protograph as well as a random version. In the protograph version, we connect neighboring copies in a regular fashion as described above. We stress that above we show the construction of the protograph of the coupled code and not the actual code which will be used for transmission. As mentioned before, for the real code, we need to “lift” the graph M times and then randomly permute edges in the same edge bundle. The “spatially coupled” qualifier for the codes comes about naturally since we consider protograph of standard LDPC codes, arrange them

Coupled Ensembles - Protograph Construction



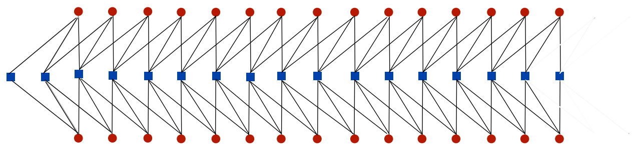
As for uncoupled codes, there are many flavors and variations on the theme. The exact version we use is not so important. They all behave more or less the same. For the purpose of this tutorial we consider two variations. Namely coupled ensembles defined by a protograph as well as a random version. In the protograph version, we connect neighboring copies in a regular fashion as described above. We stress that above we show the construction of the protograph of the coupled code and not the actual code which will be used for transmission. As mentioned before, for the real code, we need to “lift” the graph M times and then randomly permute edges in the same edge bundle. The “spatially coupled” qualifier for the codes comes about naturally since we consider protograph of standard LDPC codes, arrange them

Coupled Ensembles - Protograph Construction



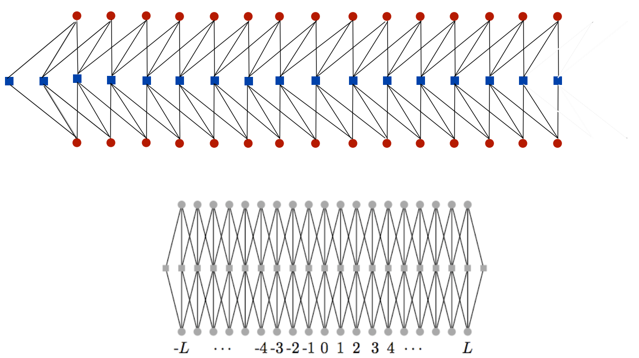
As for uncoupled codes, there are many flavors and variations on the theme. The exact version we use is not so important. They all behave more or less the same. For the purpose of this tutorial we consider two variations. Namely coupled ensembles defined by a protograph as well as a random version. In the protograph version, we connect neighboring copies in a regular fashion as described above. We stress that above we show the construction of the protograph of the coupled code and not the actual code which will be used for transmission. As mentioned before, for the real code, we need to “lift” the graph M times and then randomly permute edges in the same edge bundle. The “spatially coupled” qualifier for the codes comes about naturally since we consider protograph of standard LDPC codes, arrange them

Coupled Ensembles - Protograph Construction



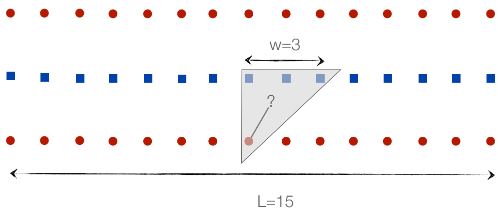
As for uncoupled codes, there are many flavors and variations on the theme. The exact version we use is not so important. They all behave more or less the same. For the purpose of this tutorial we consider two variations. Namely coupled ensembles defined by a protograph as well as a random version. In the protograph version, we connect neighboring copies in a regular fashion as described above. We stress that above we show the construction of the protograph of the coupled code and not the actual code which will be used for transmission. As mentioned before, for the real code, we need to “lift” the graph M times and then randomly permute edges in the same edge bundle. The “spatially coupled” qualifier for the codes comes about naturally since we consider protograph of standard LDPC codes, arrange them

Coupled Ensembles - Protograph Construction



As for uncoupled codes, there are many flavors and variations on the theme. The exact version we use is not so important. They all behave more or less the same. For the purpose of this tutorial we consider two variations. Namely coupled ensembles defined by a protograph as well as a random version. In the protograph version, we connect neighboring copies in a regular fashion as described above. We stress that above we show the construction of the protograph of the coupled code and not the actual code which will be used for transmission. As mentioned before, for the real code, we need to “lift” the graph M times and then randomly permute edges in the same edge bundle. The “spatially coupled” qualifier for the codes comes about naturally since we consider protograph of standard LDPC codes, arrange them

Coupled Ensembles - Random Construction



In the random version, for each edge we pick a position within a window of size w. The only restriction is that the global edge counts have to match up. But for large instance this only imposes a negligible global restriction. It is not hard to see that if we allow the window to span the entire spatial length, we will recover the standard uncoupled LDPC code.

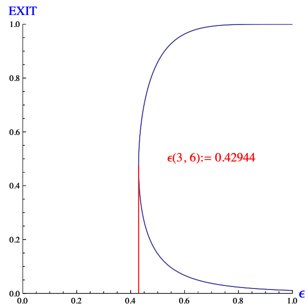
Coupled Ensembles

Protograph construction: good performance
suitable for implementation

Random construction: good for proofs

If we implement such codes in “practice”, protographs are the better choice. They in fact behave better, due to their decreased randomness. Further, the additional structure makes them well-suited for implementations. The random code construction on the other hand is better suited for proofs.

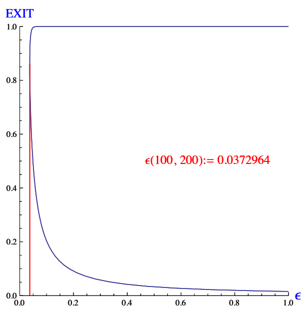
Why coupling *might* help



(d_l, d_r =2d_l)-regular
uncoupled ensemble
d_l increasing; BEC

Before we get to a more serious analysis let us see why spatial coupling might help. Let us first consider what happens if we take uncoupled ensembles and if we keep the rate fixed but increase the degrees. As we see in the sequence of EXIT curves (and as one can easily prove) the threshold decreases to zero as we increase the degrees.

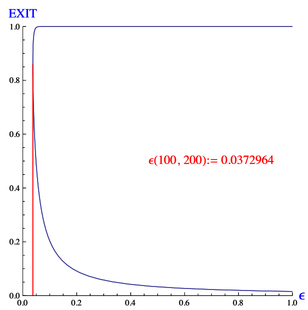
Why coupling *might* help



(d_l, d_r =2d_l)-regular
uncoupled ensemble
d_l increasing; BEC

Before we get to a more serious analysis let us see why spatial coupling might help. Let us first consider what happens if we take uncoupled ensembles and if we keep the rate fixed but increase the degrees. As we see in the sequence of EXIT curves (and as one can easily prove) the threshold decreases to zero as we increase the degrees.

Why coupling *might* help



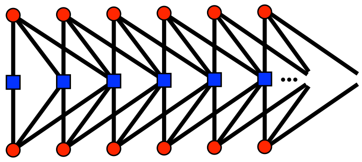
(d_l, d_r =2d_l)-regular
uncoupled ensemble
d_l increasing; BEC

General:

$$\mathtt{h}^{\text{BP}}(d_l, d_r, \{c_h\}) \le \frac{h_2\left(\frac{1}{2\sqrt{d_r-1}}\right)}{1-((1-r)d_r)e^{-2\sqrt{d_r-1}}}.$$

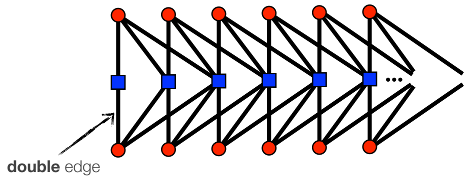
Before we get to a more serious analysis let us see why spatial coupling might help. Let us first consider what happens if we take uncoupled ensembles and if we keep the rate fixed but increase the degrees. As we see in the sequence of EXIT curves (and as one can easily prove) the threshold decreases to zero as we increase the degrees.

Why coupling *might* help



Now let us look what happens if we do the same thing for coupled ensemble. We look here at coupled ensembles constructed via the protograph approach. Further, to make the argument simple, we assume that all the edges are “double” edges, i.e., for every indicated edge above there are in fact two edges.

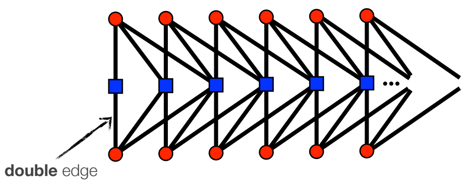
Why coupling *might* help



Now let us look what happens if we do the same thing for coupled ensemble. We look here at coupled ensembles constructed via the protograph approach. Further, to make the argument simple, we assume that all the edges are “double” edges, i.e., for every indicated edge above there are in fact two edges.

Why coupling *might* help

(6, 12)-regular coupled ensemble

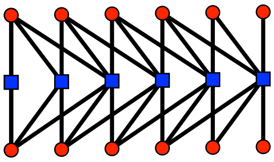


Now let us look what happens if we do the same thing for coupled ensemble. We look here at coupled ensembles constructed via the protograph approach. Further, to make the argument simple, we assume that all the edges are “double” edges, i.e., for every indicated edge above there are in fact two edges.

Why coupling *might* help

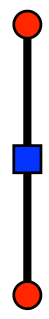
When we decode we can always ignore some additional information. This only makes the decoder perform worse. But as we see in the above case, at the boundary the code contains a (2, 4) cycle code. This code is know to have a BP threshold of 1/4 when transmitting over the BEC. So we can decode just the boundary nodes if we transmit over a BEC with erasure probability at most 1/4 using the BP decoder.

Why coupling *might* help



When we decode we can always ignore some additional information. This only makes the decoder perform worse. But as we see in the above case, at the boundary the code contains a (2, 4) cycle code. This code is know to have a BP threshold of 1/4 when transmitting over the BEC. So we can decode just the boundary nodes if we transmit over a BEC with erasure probability at most 1/4 using the BP decoder.

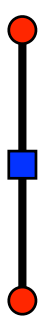
Why coupling *might* help



ignore extra edges when decoding the first section;
this makes the code worse

When we decode we can always ignore some additional information. This only makes the decoder perform worse. But as we see in the above case, at the boundary the code contains a (2, 4) cycle code. This code is know to have a BP threshold of 1/4 when transmitting over the BEC. So we can decode just the boundary nodes if we transmit over a BEC with erasure probability at most 1/4 using the BP decoder.

Why coupling *might* help

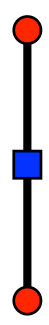


ignore extra edges when decoding the first section;
this makes the code worse

$(2,4)$ cycle code with threshold $\epsilon^{\mathrm{BP}} = \frac{1}{3}$

When we decode we can always ignore some additional information. This only makes the decoder perform worse. But as we see in the above case, at the boundary the code contains a (2, 4) cycle code. This code is know to have a BP threshold of 1/4 when transmitting over the BEC. So we can decode just the boundary nodes if we transmit over a BEC with erasure probability at most 1/4 using the BP decoder.

Why coupling *might* help



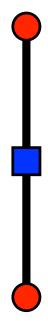
ignore extra edges when decoding the first section;
this makes the code worse

$(2,4)$ cycle code with threshold $\epsilon^{\mathrm{BP}} = \frac{1}{3}$

pick $\epsilon < \frac{1}{3}$

When we decode we can always ignore some additional information. This only makes the decoder perform worse. But as we see in the above case, at the boundary the code contains a (2, 4) cycle code. This code is know to have a BP threshold of 1/4 when transmitting over the BEC. So we can decode just the boundary nodes if we transmit over a BEC with erasure probability at most 1/4 using the BP decoder.

Why coupling *might* help



ignore extra edges when decoding the first section;
this makes the code worse

$(2, 4)$ cycle code with threshold $\epsilon^{\mathrm{BP}} = \frac{1}{3}$

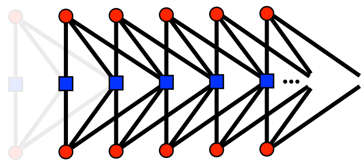
pick $\epsilon < \frac{1}{3}$

then the first section will decode correctly whp on its own

When we decode we can always ignore some additional information. This only makes the decoder perform worse. But as we see in the above case, at the boundary the code contains a (2, 4) cycle code. This code is know to have a BP threshold of 1/4 when transmitting over the BEC. So we can decode just the boundary nodes if we transmit over a BEC with erasure probability at most 1/4 using the BP decoder.

Why coupling *might* help

(6, 12)-regular coupled ensemble

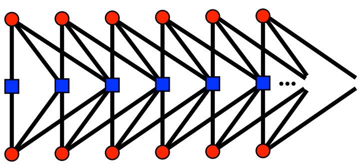


looks just like before; we conclude that the threshold is at at least 1/3

But now where we decoded the boundary nodes we see that the remainder of the code looks just like the original code, except that we “chopped” off the left-most section. The code is “self-similar” in this sense. So we can continue in recursive fashion and now decode the second section and so on. We conclude that this ensemble has a BP threshold of at least 1/3.

Why coupling *might* help

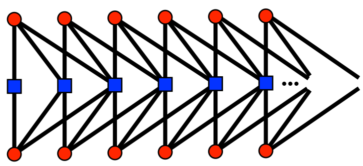
(6, 12)-regular coupled ensemble



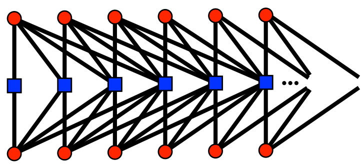
But the same argument holds if we do not start with a (6, 12) ensemble but with any (2 k, 4 k) ensemble, regardless how large the degrees are. So the BP threshold does NOT tend to zero for coupled ensembles if we increase the degrees. Indeed, we will see that they in fact get BETTER.

Why coupling *might* help

(6, 12)-regular coupled ensemble



(8, 16)-regular coupled ensemble

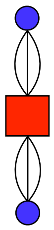


but the same argument holds even if we increase degrees

therefore threshold is lower bounded by 1/3 even if the degrees tend to infinity ....

But the same argument holds if we do not start with a (6, 12) ensemble but with any (2 k, 4 k) ensemble, regardless how large the degrees are. So the BP threshold does NOT tend to zero for coupled ensembles if we increase the degrees. Indeed, we will see that they in fact get BETTER.

Recall DE for the uncoupled case

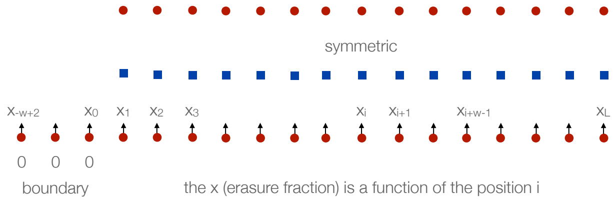


$$x^{(\ell)} = \epsilon(y^{(\ell)})^{d_l - 1}$$

$$y^{(\ell)} = 1 - (1 - x^{(\ell-1)})^{d_r - 1}$$

Let us write down now the DE equations for the coupled case. First recall that for the uncoupled case the "state" used in DE is a single scalar, namely x, which represents the erasure fraction along an outgoing edge from the variable node.

DE for the Coupled Case - Random Construction



For the coupled case the state is no longer a scalar. In the "interior" of the chain the structure of the code is shift invariant (i.e., looks the same for each position) but at the boundary the conditions are no longer uniform. We hence need one scalar x_i to describe the state at each position i. Why is the boundary set to 0. We assume that at the boundary we *know* the values. So hence the erasure probability at the boundary is 0.

DE for the Coupled Case - Random Construction

$$x_i = \epsilon(\frac{1}{w}\sum_{j=0}^{w-1} y_{i+j})^{d_l-1} \qquad y_i = 1 - (1 - \frac{1}{w}\sum_{k=0}^{w-1} x_{i-k})^{d_r-1}$$

$$y_i, \cdots, y_{i+w-1} \qquad x_i, \cdots, x_{i-w+1}$$

Again we look at what happens at the two nodes. The equations are very similar to the uncoupled case but there is one difference. If we look e.g.at the random case then each edge can be connected to positions in a certain range. We therefore need to average over the incoming messages from this range. Once we have done the average, we proceed as in the uncoupled case. Note that with respect to the way we defined the random ensemble, variables are always connected to position "to the right" and check nodes are always connected to variable nodes "on the left."

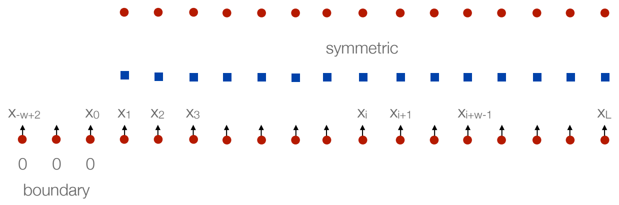
DE for the Coupled Case - Random Construction

$$x_i = \epsilon(\frac{1}{w}\sum_{j=0}^{w-1} y_{i+j})^{d_l-1} \qquad y_i = 1 - (1 - \frac{1}{w}\sum_{k=0}^{w-1} x_{i-k})^{d_r-1}$$

$$y_i, \cdots, y_{i+w-1} \qquad x_i, \cdots, x_{i-w+1}$$

Again we look at what happens at the two nodes. The equations are very similar to the uncoupled case but there is one difference. If we look e.g.at the random case then each edge can be connected to positions in a certain range. We therefore need to average over the incoming messages from this range. Once we have done the average, we proceed as in the uncoupled case. Note that with respect to the way we defined the random ensemble, variables are always connected to position "to the right" and check nodes are always connected to variable nodes "on the left."

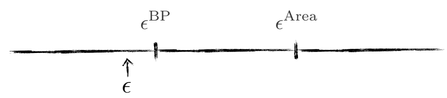
DE for the Coupled Case - Random Construction



$$x_i^{(\ell)} = \epsilon\Big(1 - \frac{1}{w}\sum_{j=0}^{w-1}(1 - \frac{1}{w}\sum_{k=0}^{w-1} x_{i+j-k}^{(\ell-1)})^{d_r-1}\Big)^{d_l-1}$$

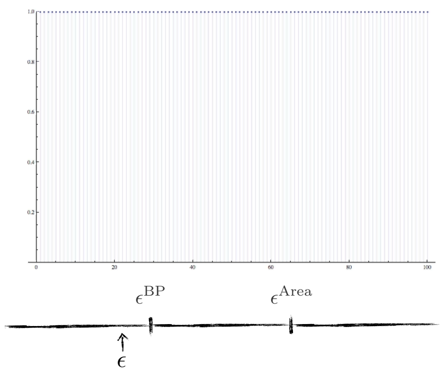
If we put the two parts together we get now the recursive DE equations for the coupled chain. Note that we now have as many x_i values as the length of the chain and that the various equations for the x_i values are "coupled" through the averaging operations.

DE for Coupled Ensemble



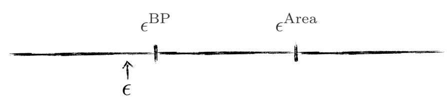
Let us now see how coupled codes perform. Let us first run DE for an eps value below the BP threshold of the uncoupled case. Note that except at the boundary, coupled codes have exactly the same local connectivity as the uncoupled ensemble they are based on. At the boundary they are "better" due to the known variables there. So we expect them to behave no worse than their uncoupled sisters. In the movie above we plot the x_i values as a function of the iteration. For the case above, the chain has length 100. Indeed, as we see from this movie, DE proceeds exactly as for the uncoupled case if we look at the x_i values in the center of the chain. At the boundary we see somewhat better values due to the termination. And as expected, the DE is able to drive the erasure fraction in each section to zero and BP is successful.

DE for Coupled Ensemble



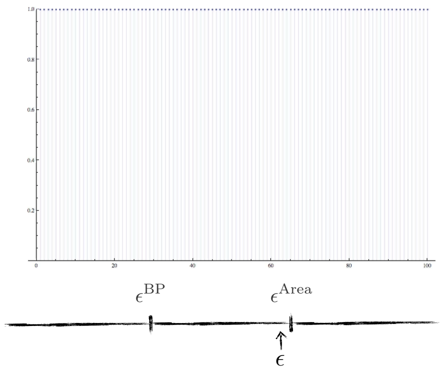
Let us now see how coupled codes perform. Let us first run DE for an eps value below the BP threshold of the uncoupled case. Note that except at the boundary, coupled codes have exactly the same local connectivity as the uncoupled ensemble they are based on. At the boundary they are "better" due to the known variables there. So we expect them to behave no worse than their uncoupled sisters. In the movie above we plot the x_i values as a function of the iteration. For the case above, the chain has length 100. Indeed, as we see from this movie, DE proceeds exactly as for the uncoupled case if we look at the x_i values in the center of the chain. At the boundary we see somewhat better values due to the termination. And as expected, the DE is able to drive the erasure fraction in each section to zero and BP is successful.

DE for Coupled Ensemble



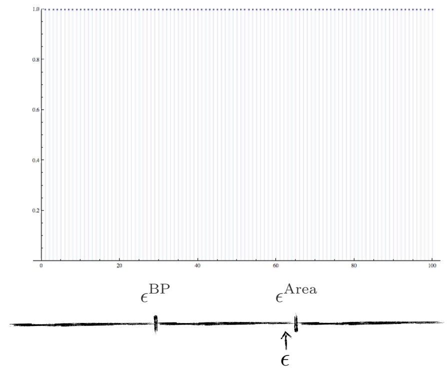
Now let us do the same experiment for a value above the BP threshold of the uncoupled ensemble but not too much larger. As we see we get a very interesting behavior. After a few iterations where all the "obvious" deductions are being made the decoder still proceeds to make progress. On both ends a small "wave front" has formed. These wave fronts move towards the center at a constant speed. Note that the wave connects the two FPs which exist also for the uncoupled case. Namely, the undesired FP in which the uncoupled code gets stuck (forward FP of DE) and the desired FP (namely zero) which an optimal decoder would find. The wave "bridges" these two FPs and thus guides nodes in the interior of the chain towards the desired FP. Also, note the special structure that the erasure fraction at each section forms:

DE for Coupled Ensemble



Now let us do the same experiment for a value above the BP threshold of the uncoupled ensemble but not too much larger. As we see we get a very interesting behavior. After a few iterations where all the "obvious" deductions are being made the decoder still proceeds to make progress. On both ends a small "wave front" has formed. These wave fronts move towards the center at a constant speed. Note that the wave connects the two FPs which exist also for the uncoupled case. Namely, the undesired FP in which the uncoupled code gets stuck (forward FP of DE) and the desired FP (namely zero) which an optimal decoder would find. The wave "bridges" these two FPs and thus guides nodes in the interior of the chain towards the desired FP. Also, note the special structure that the erasure fraction at each section forms:

DE for Coupled Ensemble

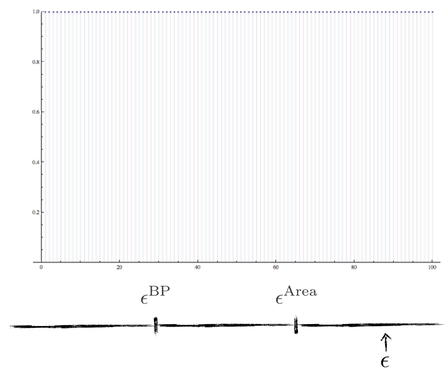


Now let us do the same experiment for a value above the BP threshold of the uncoupled ensemble but not too much larger. As we see we get a very interesting behavior. After a few iterations where all the "obvious" deductions are being made the decoder still proceeds to make progress. On both ends a small "wave front" has formed. These wave fronts move towards the center at a constant speed. Note that the wave connects the two FPs which exist also for the uncoupled case. Namely, the undesired FP in which the uncoupled code gets stuck (forward FP of DE) and the desired FP (namely zero) which an optimal decoder would find. The wave "bridges" these two FPs and thus guides nodes in the interior of the chain towards the desired FP. Also, note the special structure that the erasure fraction at each section forms:

DE for Coupled Ensemble

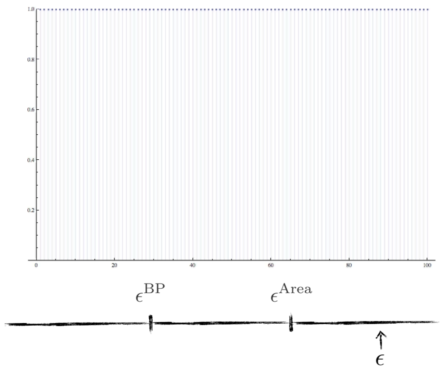
Finally, if we go above a certain value of eps and we run DE then we get a non-trivial FP. Note that in the middle of the chain the xi values are exactly as large as they would be for the same eps value in the uncoupled case. Only at the boundary do we get somewhat better values due to the termination.

DE for Coupled Ensemble



Finally, if we go above a certain value of eps and we run DE then we get a non-trivial FP. Note that in the middle of the chain the xi values are exactly as large as they would be for the same eps value in the uncoupled case. Only at the boundary do we get somewhat better values due to the termination.

DE for Coupled Ensemble



Finally, if we go above a certain value of eps and we run DE then we get a non-trivial FP. Note that in the middle of the chain the xi values are exactly as large as they would be for the same eps value in the uncoupled case. Only at the boundary do we get somewhat better values due to the termination.

Questions

What is ϵ^{Area} ?

Why does this happen?

Spoiler alert: We will see that the area threshold is essentially equal to the MAP threshold of the underlying ensemble! Here, the MAP threshold is the threshold an optimal decoder would achieve. And it turns out to be the BP threshold of the coupled code ensemble!

Given what we have seen, we are faced with a few questions. First, what is this parameter $\varepsilon^{\mathrm{Area}}$ up to which spatially coupled ensembles can decode? Second, why does the system so drastically change its behavior when we couple it? Spoiler Alert: We will see that the answer to the first question is that $\varepsilon^{\mathrm{Area}}$ is essentially equal $\varepsilon^{\mathrm{MAP}}$ of the underlying ensemble! We say here essentially since, as we will explain in more detail later, this is strictly true only when we let certain quantities tend to infinity. But even if we do not let these quantities tend to infinity but choose them reasonably small, the difference is typically very very small.

Historical Outline of Spatially Coupled Codes

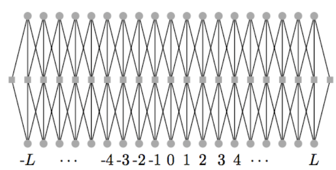
Big bang:

A. J. Felström and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 2181–2190, Sept. 1999.

Historical Outline of Spatially Coupled Codes

Big bang:

A. J. Felström and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 2181–2190, Sept. 1999.



Historical Outline of Spatially Coupled Codes

Variations, code word and pseudo code word analysis:

K. Engdahl and K. S. Zigangirov, “On the theory of low density convolutional codes I,” *Problemy Peredachi Informatsii*, vol. 35, no. 4, pp. 295–310, 1999.

M. Lentmaier, D. V. Truhachev, and K. S. Zigangirov, “To the theory of low-density convolutional codes. II,” *Probl. Inf. Transm.*, vol. 37, no. 4, pp. 288–306, 2001.

R. Smarandache, A. Pusane, P. Vontobel, and D. J. Costello, Jr., “Pseudo-codewords in LDPC convolutional codes,” in *Proc. of the IEEE Int. Symposium on Inform. Theory*, Seattle, WA, USA, July 2006, pp. 1364 – 1368.

——, “Pseudocodeword performance analysis for LDPC convolutional codes,” *IEEE Trans. Inform. Theory*, vol. 55, no. 6, pp. 2577–2598, June 2009.

K. Engdahl, M. Lentmaier, and K. S. Zigangirov, “On the theory of low-density convolutional codes,” in *AAECC-13: Proceedings of the 13th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*. London, UK: Springer-Verlag, 1999, pp. 77–86.

R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, Jr., “LDPC block and convolutional codes based on circulant matrices,” *IEEE Trans. Inform. Theory*, vol. 50, no. 12, pp. 2966 – 2984, Dec. 2004.

D. G. M. Mitchell, A. E. Pusane, K. S. Zigangirov, and D. J. Costello, Jr., “Asymptotically good LDPC convolutional codes based on protographs,” in *Proc. of the IEEE Int. Symposium on Inform. Theory*, Toronto, CA, July 2008, pp. 1030 – 1034.

——, “Pseudocodeword performance analysis for LDPC convolutional codes,” *IEEE Trans. Inform. Theory*, vol. 55, no. 6, pp. 2577–2598, June 2009.

Historical Outline of Spatially Coupled Codes

Termination and density evolution analysis:

A. Sridharan, M. Lentmaier, D. J. Costello, Jr., and K. S. Zigangirov, “Convergence analysis of a class of LDPC convolutional codes for the erasure channel,” in *Proc. of the Allerton Conf. on Commun., Control, and Computing*, Monticello, IL, USA, Oct. 2004.

M. Lentmaier, A. Sridharan, K. S. Zigangirov, and D. J. Costello, Jr., “Terminated LDPC convolutional codes with thresholds close to capacity,” *CoRR*, vol. abs/cs/0508030, 2005.

M. Lentmaier and G. P. Fettweis, “On the thresholds of generalized LDPC convolutional codes based on protographs,” in *Proc. of the IEEE Int. Symposium on Inform. Theory*, Austin, USA, 2010.

M. Lentmaier, A. Sridharan, K. S. Zigangirov, and D. J. Costello, Jr., “Iterative decoding threshold analysis for LDPC convolutional codes,” *IEEE Trans. Info. Theory*, Oct. 2010. Vol. 56, No. 10, pp. 5274

Historical Outline of Spatially Coupled Codes

Termination and density evolution analysis:

A. Sridharan, M. Lentmaier, D. J. Costello, Jr., and K. S. Zigangirov, “Convergence analysis of a class of LDPC convolutional codes for the erasure channel,” in *Proc. of the Allerton Conf. on Commun., Control, and Computing*, Monticello, IL, USA, Oct. 2004.

M. Lentmaier, A. Sridharan, K. S. Zigangirov, and D. J. Costello, Jr., “Terminated LDPC convolutional codes with thresholds close to capacity,” *CoRR*, vol. abs/cs/0508030, 2005.

M. Lentmaier and G. P. Fettweis, “On the thresholds of generalized LDPC convolutional codes based on protographs,” in *Proc. of the IEEE Int. Symposium on Inform. Theory*, Austin, USA, 2010.

M. Lentmaier, A. Sridharan, K. S. Zigangirov, and D. J. Costello, Jr., “Iterative decoding threshold analysis for LDPC convolutional codes,” *IEEE Trans. Info. Theory*, Oct. 2010. Vol. 56, No. 10, pp. 5274

Saturday, July 13, 13 42

Historical Outline of Spatially Coupled Codes

Maxwell Construction, MAP Thresholds, generalized EXIT functions

C. Méasson, A. Montanari, and R. Urbanke, “Maxwell construction: The hidden bridge between maximum-likelihood and iterative decoding,” *IEEE Trans. Inform. Theory*, vol. 54, no. 12, pp. 5277 – 5307, 2008.

C. Méasson, A. Montanari, T. Richardson, and R. Urbanke, “The generalized area theorem and some of its consequences,” *IEEE Trans. Inform. Theory*, vol. 55, no. 11, pp. 4793–4821, Nov. 2009.

Historical Outline of Spatially Coupled Codes

Threshold saturation effect and proof for the BEC:

S. Kudekar, T. Richardson, and R. Urbanke, “Threshold Saturation via Spatial Coupling: Why Convolutional LDPC Ensembles Perform so well over the BEC,” *IEEE Trans. Inform. Theory*, vol. 57, no. 2, pp. 803–834, Feb. 2011.

S. Kudekar, T. Richardson, and R. Urbanke, “Wave-Like Solutions of General One-Dimensional Spatially Coupled Systems.” In preparation, Jan. 2012.

A. Yedla, Y. Jian, P. Nguyen, and H. Pfister, “A Simple Proof of Threshold Saturation for Coupled Scalar Recursions,” 2012, e-print: http://arxiv.org/abs/1204.5703.

Main Message

1. In order to get exactly ϵ^{MAP} we have to let the “connection width” w tend to infinity. But in practice even small widths like w=3 lead to thresholds which are almost indistinguishable ϵ^{MAP}. (e.g., for the (3, 6) case and the BEC the difference is about 10^{-5}.
2. The MAP threshold is an *increasing* function of the degrees and converges to the Shannon threshold exponentially fast. This is contrary to the BP threshold for uncoupled codes which typically *decreases* in the degree.
3. We will see that using spatial coupling we can construct codes which are capacity-achieving *universally* across the whole set of BMS channels.
4. On the downside, due to the termination which is required, we loose in rate. We hence have to take the chain length large enough in order to amortize this rate loss. Therefore, the blocklength has to be reasonably large.

Main Message

Coupled ensembles under BP decoding behave like uncoupled ensembles under MAP decoding.

1. In order to get exactly ϵ^{MAP} we have to let the “connection width” w tend to infinity. But in practice even small widths like w=3 lead to thresholds which are almost indistinguishable ϵ^{MAP}. (e.g., for the (3, 6) case and the BEC the difference is about 10^{-5}.
2. The MAP threshold is an *increasing* function of the degrees and converges to the Shannon threshold exponentially fast. This is contrary to the BP threshold for uncoupled codes which typically *decreases* in the degree.
3. We will see that using spatial coupling we can construct codes which are capacity-achieving *universally* across the whole set of BMS channels.
4. On the downside, due to the termination which is required, we loose in rate. We hence have to take the chain length large enough in order to amortize this rate loss. Therefore, the blocklength has to be reasonably large.

Main Message

Coupled ensembles under BP decoding behave like uncoupled ensembles under MAP decoding.

Since coupled ensemble achieve the highest threshold they can achieve (namely the MAP threshold) under BP we speak of the threshold saturation phenomenon.

1. In order to get exactly ϵ^{MAP} we have to let the “connection width” w tend to infinity. But in practice even small widths like w=3 lead to thresholds which are almost indistinguishable ϵ^{MAP}. (e.g., for the (3, 6) case and the BEC the difference is about 10^{-5}.
2. The MAP threshold is an *increasing* function of the degrees and converges to the Shannon threshold exponentially fast. This is contrary to the BP threshold for uncoupled codes which typically *decreases* in the degree.
3. We will see that using spatial coupling we can construct codes which are capacity-achieving *universally* across the whole set of BMS channels.
4. On the downside, due to the termination which is required, we loose in rate. We hence have to take the chain length large enough in order to amortize this rate loss. Therefore, the blocklength has to be reasonably large.

Main Message

Coupled ensembles under BP decoding behave like uncoupled ensembles under MAP decoding.

Since coupled ensemble achieve the highest threshold they can achieve (namely the MAP threshold) under BP we speak of the threshold saturation phenomenon.

We will see that using spatial coupling we can construct codes which are capacity-achieving *universally* across the whole set of BMS channels.

1. In order to get exactly ϵ^{MAP} we have to let the "connection width" w tend to infinity. But in practice even small widths like w=3 lead to thresholds which are almost indistinguishable ϵ^{MAP}. (e.g., for the (3, 6) case and the BEC the difference is about 10^{-5}.
2. The MAP threshold is an *increasing* function of the degrees and converges to the Shannon threshold exponentially fast. This is contrary to the BP threshold for uncoupled codes which typically *decreases* in the degree.
3. We will see that using spatial coupling we can construct codes which are capacity-achieving *universally* across the whole set of BMS channels.
4. On the downside, due to the termination which is required, we loose in rate. We hence have to take the chain length large enough in order to amortize this rate loss. Therefore, the blocklength has to be reasonably large.

Main Message

Coupled ensembles under BP decoding behave like uncoupled ensembles under MAP decoding.

Since coupled ensemble achieve the highest threshold they can achieve (namely the MAP threshold) under BP we speak of the threshold saturation phenomenon.

We will see that using spatial coupling we can construct codes which are capacity-achieving *universally* across the whole set of BMS channels.

On the downside, due to the termination which is required, we loose in rate. We hence have to take the chain length large enough in order to amortize this rate loss. Therefore, the blocklength has to be reasonably large.

1. In order to get exactly ϵ^{MAP} we have to let the "connection width" w tend to infinity. But in practice even small widths like w=3 lead to thresholds which are almost indistinguishable ϵ^{MAP}. (e.g., for the (3, 6) case and the BEC the difference is about 10^{-5}.
2. The MAP threshold is an *increasing* function of the degrees and converges to the Shannon threshold exponentially fast. This is contrary to the BP threshold for uncoupled codes which typically *decreases* in the degree.
3. We will see that using spatial coupling we can construct codes which are capacity-achieving *universally* across the whole set of BMS channels.
4. On the downside, due to the termination which is required, we loose in rate. We hence have to take the chain length large enough in order to amortize this rate loss. Therefore, the blocklength has to be reasonably large.

(not exhaustive) References

Most papers on this topic can be found on arXiv.org. To get started, look for "spatial" and "coupl" in the title and restrict your search to the time range since 2009 and the Computer Science category. Currently you will find 45 articles on this topic on arXiv.org. But there are many more ... e.g., many important papers which use spatial coupling in the area of compressive sensing do not contain these two keywords in their title and are hence not included in this list.

1. arXiv:1306.3610 [pdf, ps, other]
   Thresholds of Spatially Coupled Systems via Lyapunov's Method
   Christian Schlegel, Marat Burnashev
   Comments: 6 pages
   Subjects: **Information Theory (cs.IT)**

2. arXiv:1305.5625 [pdf, other]
   Memory Efficient Decoders using Spatially Coupled Quasi-Cyclic LDPC Codes
   Vikram Arkalgud Chandrasetty, Sarah J. Johnson, Gottfried Lechner
   Comments: 4 pages, 7 figures, 1 table, submitted to IEEE Communication Letters
   Subjects: **Information Theory (cs.IT)**

3. arXiv:1304.6599 [pdf, other]
   Robust error correction for real-valued signals via message-passing decoding and spatial coupling
   Jean Barbier, Florent Krzakala, Lenka Zdeborová, Pan Zhang
   Comments: 5 pages, 5 figures
   Subjects: **Information Theory (cs.IT)**

4. arXiv:1304.6026 [pdf, ps, other]
   Displacement Convexity, A Useful Framework for the Study of Spatially Coupled Codes
   Rafah El-Khatib, Nicolas Macris, Ruediger Urbanke
   Comments: 9 pages, 1 figure, extended version of paper submitted to ITW 2013
   Subjects: **Information Theory (cs.IT)**

5. arXiv:1303.1858 [pdf, ps, other]
   On the Minimum Distance of Generalized Spatially Coupled LDPC Codes
   David G. M. Mitchell, Michael Lentmaier, Daniel J. Costello Jr
   Comments: Submitted to the IEEE International Symposium on Information Theory 2013
   Subjects: **Information Theory (cs.IT)**

6. arXiv:1303.0296 [pdf, ps, other]
   Performance of Spatially-Coupled LDPC Codes and Threshold Saturation over BICM Channels
   Arvind Yedla, Mostafa El-Khamy, Jungwon Lee, Inyup Kang
   Comments: 7 pages, 6 figures
   Subjects: **Information Theory (cs.IT)**

7. arXiv:1302.1512 [pdf, ps, other]
   Efficient Termination of Spatially-Coupled Codes
   Koji Tazoe, Kenta Kasai, Kohichi Sakaniwa
   Subjects: **Information Theory (cs.IT)**

8. arXiv:1302.1511 [pdf, ps, other]
   Spatially-Coupled Precoded Rateless Codes
   Kosuke Sakata, Kenta Kasai, Kohichi Sakaniwa
   Subjects: **Information Theory (cs.IT)**

9. arXiv:1302.1510 [pdf, ps, other]
   Multi-Dimensional Spatially-Coupled Codes
   Ryunosuke Ohashi, Kenta Kasai, Keigo Takeuchi
   Subjects: **Information Theory (cs.IT)**

10. arXiv:1301.6410 [pdf, ps, other]
    Linear Programming Decoding of Spatially Coupled Codes
    Louay Bazzi, Badih Ghazi, Rudiger Urbanke
    Comments: 37 pages; Added tightness construction, expanded abstract
    Subjects: **Information Theory (cs.IT)**

11. arXiv:1301.6111 [pdf, ps, other]
    A Proof of Threshold Saturation for Spatially-Coupled LDPC Codes on BMS Channels
    Santhosh Kumar, Andrew J. Young, Nicolas Macris, Henry D. Pfister
    Comments: In proceedings of Allerton 2012; Corrected a typo in equation (5)
    Subjects: **Information Theory (cs.IT)**

12. arXiv:1301.5728 [pdf, ps, other]
    A Potential Theory of General Spatially-Coupled Systems via a Continuum Approximation
    Keigo Takeuchi, Toshiyuki Tanaka, Kenta Kasai
    Comments: submitted to IEEE ITW2013
    Subjects: **Information Theory (cs.IT)**

13. arXiv:1301.5676 [pdf, ps, other]
    Spatial Coupling as a Proof Technique
    Andrei Giurgiu, Nicolas Macris, Rüdiger Urbanke
    Comments: 11 pages
    Subjects: **Information Theory (cs.IT)**

14. arXiv:1211.5164 [pdf, other]
    State Evolution for General Approximate Message Passing Algorithms, with Applications to Spatial Coupling
    Adel Javanmard, Andrea Montanari
    Comments: 29 pages, 1 figure, minor updates in citations
    Subjects: **Probability (math.PR)**; Information Theory (cs.IT); Statistics Theory (math.ST)

15. arXiv:1210.6168 [pdf, ps, other]
    Accelerating Iterative Detection for Spatially Coupled Systems by Collaborative Training
    Keigo Takeuchi
    Comments: accepted for publication on IEEE Commun. Lett
    Subjects: **Information Theory (cs.IT)**

16. arXiv:1209.5762 [pdf, ps, other]
    Nonbinary Spatially-Coupled LDPC Codes on the Binary Erasure Channel
    Amina Piemontese, Alexandre Graell i Amat, Giulio Colavolpe
    Comments: Submitted to IEEE International Conference on Communications 2013
    Subjects: **Information Theory (cs.IT)**

17. arXiv:1208.5273 [pdf, ps, other]
    Wave-Like Solutions of General One-Dimensional Spatially Coupled Systems
    Shrinivas Kudekar, Tom Richardson, Ruediger Urbanke
    Comments: 32 pages, 10 figures. Submitted to IEEE Transactions on Information theory
    Subjects: **Information Theory (cs.IT)**

18. arXiv:1208.4294 [pdf, ps, other]
    Guaranteeing Spatial Uniformity in Diffusively-Coupled Systems
    S. Yusef Shafi
    Subjects: **Dynamical Systems (math.DS)**; Systems and Control (cs.SY); Optimization and Control (math.OC)

19. arXiv:1206.5919 [pdf, ps, other]
    Performance Improvement of Iterative Multiuser Detection for Large Sparsely-Spread CDMA Systems by Spatial Coupling
    Keigo Takeuchi, Toshiyuki Tanaka, Tsutomu Kawabata
    Comments: submitted to IEEE Trans. Inf. Theory
    Subjects: **Information Theory (cs.IT)**

20. arXiv:1205.5904 [pdf, other]
    Joint Compute and Forward for the Two Way Relay Channel with Spatially Coupled LDPC Codes
    Brett Hern, Krishna Narayanan
    Comments: This paper was submitted to IEEE Global Communications Conference 2012
    Subjects: **Information Theory (cs.IT)**

21. arXiv:1205.3317 [pdf, ps, other]
    Spatially-Coupled Random Access on Graphs
    Gianluigi Liva, Enrico Paolini, Michael Lentmaier, Marco Chiani
    Comments: To be presented at IEEE ISIT 2012, Boston
    Subjects: **Information Theory (cs.IT)**

22. arXiv:1202.4959 [pdf, ps, other]
    Lossy Source Coding via Spatially Coupled LDGM Ensembles
    Vahid Aref, Nicolas Macris, Rudiger Urbanke, Marc Vuffray
    Comments: Submitted to ISIT 2012
    Subjects: **Information Theory (cs.IT)**; Statistical Mechanics (cond-mat.stat-mech)

23. arXiv:1202.0979 [pdf, ps, other]
    Spatially-Coupled Binary MacKay-Neal Codes for Channels with Non-Binary Inputs and Affine Subspace Outputs
    Kenta Kasai, Takayuki Nozaki, Kohichi Sakaniwa
    Subjects: **Information Theory (cs.IT)**

24. arXiv:1201.2999 [pdf, ps, other]
    Spatially Coupled Ensembles Universally Achieve Capacity under Belief Propagation
    Shrinivas Kudekar, Tom Richardson, Ruediger Urbanke
    Comments: 50 pages, 9 figures
    Subjects: **Information Theory (cs.IT)**

25. arXiv:1112.6320 [pdf, ps, other]
    Threshold Saturation in Spatially Coupled Constraint Satisfaction Problems
    S. Hamed Hassani, Nicolas Macris, Rudiger Urbanke
    Subjects: **Computational Complexity (cs.CC)**; Statistical Mechanics (cond-mat.stat-mech); Information Theory (cs.IT)

26. arXiv:1112.0708 [pdf, other]
    Information-Theoretically Optimal Compressed Sensing via Spatial Coupling and Approximate Message Passing
    David L. Donoho, Adel Javanmard, Andrea Montanari
    Comments: 60 pages, 7 figures, Sections 3,5 and Appendices A,B are added. The stability constant is quantified (cf Theorem 1.7)
    Subjects: **Information Theory (cs.IT)**; Statistical Mechanics (cond-mat.stat-mech); Statistics Theory (math.ST)

27. arXiv:1111.5668 [pdf, other]
    Connecting Spatially Coupled LDPC Code Chains
    Dmitri Truhachev, David G. M. Mitchell, Michael Lentmaier, Daniel J. Costello, Jr
    Comments: Submitted to International Conference on Communications (ICC) 2012
    Subjects: **Information Theory (cs.IT)**; Discrete Mathematics (cs.DM)

28. arXiv:1110.6787 [pdf, other]
    Spatially Coupled Repeat-Accumulate Codes
    Sarah J. Johnson, Gottfried Lechner
    Comments: submitted to IEEE Communications Letters
    Subjects: **Information Theory (cs.IT)**

29. arXiv:1110.0252 [pdf, other]
    Universal Codes for the Gaussian MAC via Spatial Coupling
    Arvind Yedla, Phong S. Nguyen, Henry D. Pfister, Krishna R. Narayanan
    Comments: 8 pages, to appear in proceedings of Allerton 2011
    Subjects: **Information Theory (cs.IT)**

30. arXiv:1107.4900 [pdf, ps, other]
    Threshold Improvement of Low-Density Lattice Codes via Spatial Coupling
    Hironori Uchikawa, Brian M. Kurkoski, Kenta Kasai, Kohichi Sakaniwa
    Comments: 5 pages, 4 figures, submitted to ICNC2012
    Subjects: **Information Theory (cs.IT)**

41. arXiv:1102.0406 [pdf, ps, other]
    Threshold Saturation on Channels with Memory via Spatial Coupling
    Shrinivas Kudekar, Kenta Kasai
    Comments: Submitted to ISIT 2011
    Subjects: **Information Theory (cs.IT)**

42. arXiv:1010.6020 [pdf, other]
    The Effect of Spatial Coupling on Compressive Sensing
    Shrinivas Kudekar, Henry D. Pfister
    Comments: 8 Pages, 7 figures. This is a slightly modified version of the paper which appeared in the 48th Annual Allerton Conference on Communication, Control, and Computing September 29 - October 1, 2010
    Subjects: **Information Theory (cs.IT)**

43. arXiv:1010.1669 [pdf, ps, other]
    Rate-Equivocation Optimal Spatially Coupled LDPC Codes for the BEC Wiretap Channel
    Vishwambhar Rathi, Ruediger Urbanke, Mattias Andersson, Mikael Skoglund
    Comments: Working paper
    Subjects: **Information Theory (cs.IT)**

44. arXiv:1004.3742 [pdf, ps, other]
    Threshold Saturation on BMS Channels via Spatial Coupling
    Shrinivas Kudekar, Cyril Measson, Tom Richardson, Ruediger Urbanke
    Comments: 5 pages, 5 figures
    Subjects: **Information Theory (cs.IT)**

45. arXiv:1001.1826 [pdf, other]
    Threshold Saturation via Spatial Coupling: Why Convolutional LDPC Ensembles Perform so well over the BEC
    Shrinivas Kudekar, Tom Richardson, Ruediger Urbanke
    Comments: 29 pages, 11 figures, To appear in Special Issue of the IEEE Transactions on Information Theory, Facets of Coding Theory: from Algorithms to Networks
    Subjects: **Information Theory (cs.IT)**