Conference Paper

Iteration-constrained design of IRA codes

Author(s):
Grant, Alex; Land, Ingmar; Wang, Guangsong

Publication Date:
2012

Permanent Link:
https://doi.org/10.3929/ethz-a-007050352

Rights / License:
In Copyright - Non-Commercial Use Permitted
Iteration-constrained design of IRA codes

Alex Grant, Ingmar Land and Guangsong Wang
Institute for Telecommunications Research
University of South Australia
{alex.grant,ingmar.land}@unisa.edu.au, wangy012@mymail.unisa.edu.au

Abstract—This paper addresses extrinsic information chart based design of non-systematic irregular repeat-accumulate codes for a given number of decoding iterations. This criterion is of practical importance in many applications where complexity or latency is limited. Our main contribution is a novel formulation of the optimization problem in a particular way that makes its actual evaluation possible. This is achieved by introducing additional optimization variables and constraints which have the effect of “unrolling” the iteration. Our approach enforces the finite iteration condition, while avoiding the need to know arbitrary repeated functional compositions of the component EXIT functions. We restrict attention to the binary erasure channel, for which the EXIT chart approach is exact. Extension to other sparse-graph codes and other communication channels is straightforward.

I. INTRODUCTION

Sparse-graph codes with iterative message-passing decoding represent the state-of-the-art of modern channel coding. The two major classes are low-density parity-check (LDPC) codes, and irregular repeat-accumulate (IRA) codes. For both classes, encoding and decoding complexity are linear in the code length, and there are efficient code design methods [1–3].

The typical design goal in the literature is capacity-approaching codes. These codes that have maximum rate while the probability of error diminishes with increasing code length and increasing number of iterations. The predominant approaches are density evolution and extrinsic information transfer (EXIT) charts [1, 4, 5]. While density evolution provides exact results for the decoding threshold, the EXIT chart method employs a Gaussian approximation. The advantage of the latter, however, is that code design can often be formulated in terms of a convex optimization problem, allowing convenient numerical evaluation. Tradeoffs between the two methods for the design of IRA codes were discussed in [6].

For erasure channels, the two methods are equivalent.

Two important parameters characterizing implementation complexity are the parity-check matrix density and the number of decoding iterations. The relationship between parity-check density and performance in terms of the gap to capacity have been analyzed in [7, 8], the required number of iterations for successful decoding and corresponding bounds have been addressed in [9]. As opposed to LDPC codes, non-systematic IRA codes can achieve capacity on the binary erasure channel with bounded complexity per information bit [7].

Whereas the above approaches assume infinite iteration, the number of iterations required to achieve a target performance is of practical interest. In [10] a method for the design of irregular LDPC codes was proposed, which minimizes the number of iterations for a given target error probability. The key concept is to approximate the number of iterations by a continuous function, and use this as the objective function in the optimization. Starting from a capacity-achieving code, the variable-node degree distribution is iteratively adapted to improve this objective function.

We take a different approach, seeking a code that is optimized for a fixed number of iterations. We restrict ourselves to non-systematic IRA codes over the binary erasure channel (BEC). Extension to other spares-graph codes and to other communication channels (given the usual assumptions and approximations with EXIT charts) is possible.

II. IRA CODES AND EXIT CHARTS

We assume familiarity with low-density parity-check codes, their graphical representation, iterative decoding, and EXIT chart based design [1–3, 5]. We focus on non-systematic irregular repeat accumulate codes, where information bits are repeated, interleaved, passed through parity-checks, and finally scrambled by an accumulator [6, 11]. Only the code bits produced by the accumulator are transmitted over the channel. We restrict our analysis to the binary erasure channel, for which the EXIT chart approach is exact [12].

The encoder maps \( k \) information bits onto \( n \) code bits, the code rate is \( R = k/n \). The code is defined by the edge perspective degree polynomials, \( \lambda_i(z) = \sum_{i} \lambda_i z^i \) for the variable-nodes and \( \rho(z) = \sum_{i} \rho_i z^{i-1} \) for the check-nodes, where \( \lambda_i, \rho_i, i = 1, 2, \ldots \) denotes proportion of edges connected to variable nodes of degree \( i \), and \( \rho_i, i = 1, 2, \ldots \) denotes the proportion of edges connected to check nodes of degree \( i \).

According to common practice in the literature, \( \rho_i \) does not connect the connection to the accumulator. The design rate is

\[
R_d = \frac{\sum_i \lambda_i / \rho_i}{\sum_i \lambda_i / \rho_i} \leq R
\]

The code bits are transmitted over a binary erasure channel with erasure probability \( \delta \) and capacity \( C = 1 - \delta \).

The iterative decoder consists of three components: the variable-node decoder, the check-node decoder, and the accumulator decoder. Only the accumulator obtains observations from the channel. In every iteration the component decoders are activated in the order variable-check-accumulator-check.

In the EXIT chart method, the mutual information transfer between the component decoders is modeled and used to analyze convergence behavior [5]. For the following analysis,
The check-accumulator EXIT function is [12]

\[ I = \text{extrinsic information} \]

As shown in Figure 1, we orient the extrinsic information transfer chart with the \( x \)-axis representing the variable-node extrinsic information \( I_{E_v} \) and the check-accumulator a-priori information \( I_{A_v} \). The \( y \)-axis carries the variable-node a-priori information \( I_{A_v} \) and check-accumulator extrinsic information, \( I_{E_{ac}} \). In order to avoid proliferation of subscripts, where it will not cause confusion, we simply use \( x \) and \( y \) to represent these quantities. The variable-node EXIT function \( f(y) \) maps from \( y = I_{A_v} \) onto \( x = I_{E_v} \). Conversely, the check-accumulator function \( g(x) \) maps from \( x = I_{A_{ac}} \) onto \( y = I_{E_{ac}} \).

The variable node EXIT function is given by [12]

\[ f(y) = \sum_i \lambda_i f_i(y) \]

\[ f_i(y) = 1 - (1 - y)^{i-1}, \quad i = 1, 2, \ldots \]

where \( f_i \) is the transfer function for a degree-\( i \) variable node. The check-accumulator EXIT function is [12]

\[ g(x) = \sum_i \rho_i g_i(x), \]

\[ g_i(x) = x^{i-1} \left( \frac{C}{1 - (1 - C)x^i} \right)^2, \quad i = 1, 2, \ldots \]

where \( g_i \) is the transfer function for a degree-\( i \) check node combined with the accumulator. We can show that \( f(y) \) and \( g(x) \) are linear in \( \lambda \) and \( \rho \) and convex in \( y \) and \( x \), respectively.

Denote \( x^{(\ell)} = I_{E_v}^{(\ell)} \) the extrinsic information at the output of the variable node decoder after iteration \( \ell \), and let \( y^{(\ell)} = I_{A_v}^{(\ell)} \) be the corresponding a-priori information. With reference to Figure 1, the corresponding values of the trajectory for \( L \) iterations are given recursively for \( \ell = 1, 2, \ldots \) by

\[ x^{(\ell)} = f(y^{(\ell-1)}), \quad y^{(\ell)} = g(x^{(\ell)}). \]

The initial values are \( x^{(0)} = 0 \) and \( y^{(0)} = g(0) \), resulting from the fact that the variable nodes are not directly connected to the communication channel.

We can eliminate the \( y^{(\ell)} \) from (6) to obtain \( x^{(\ell)} = f(g(x^{(\ell-1)}) \). Going a step further, we may also substitute these equalities into each other to express the variable node extrinsic information after \( L \) iterations \( I_{E_v}^{(L)} \) as

\[ x^{(L)} = h_L(0) \]

where \( h(x) = f(g(x)) \) and \( h_L \) denotes \( L \)-fold functional composition.

The areas under the variable node curve (with the orientation indicated in Figure 1) and under the check-accumulator curve are given by

\[ 1 - \int_0^1 f(y) \, dy = \sum_i \lambda_i = \int_0^1 \lambda(z) \, dz, \]

\[ \int_0^1 g(x) \, dx = C \sum_j \rho_j = C \int_0^1 \rho(z) \, dz, \]

respectively. Substituting from (1) we have the following theorem, first proved for low-density parity-check codes in [12].

**Theorem 1 (Area Theorem):**

\[ C - R_d = \frac{A}{\int_0^1 \rho(z) \, dz}, \]

\[ A = \int_0^1 g(x) \, dx - \left( 1 - \int_0^1 f(y) \, dy \right). \]

where \( A \) is the area between the check-accumulator curve \( g \) and the variable node curve \( f \).

Thus, finding the code with the smallest rate gap from capacity corresponds to minimizing the area between \( f \) and \( g \). Note a minor difference from the original LDPC result, where \( f \rho \) sets a “scaling factor”. We do not require the inner check-accumulator code to have rate 1.

For code analysis and design, it is useful to express the (information) bit error rate of the decoder as a function of the mutual information exchanged during the iterative process.

**Theorem 2 (Bit Error Rate):** The bit error rate \( P_b \) of a non-systematic IRA code transmitted over a BEC, with variable node degree distribution \( \lambda \) and a-priori information \( I_{A_v} \) (to the variable nodes) is given by

\[ P_b = \frac{1 - I_v}{2}, \]

\[ I_v = \left( \sum_i \lambda_i i \right)^{-1} \sum_i \lambda_i \left( (1 - I_{A_v})^i \right). \]

The value \( I_v \) denotes the overall variable node information.
Note that $P_b$ is monotonically increasing in $I_{Av}$. For fixed design rate $R_d$, \[
I_v = \left( R_d \sum_j \frac{\rho_j}{j} \right)^{-1} \sum_i \frac{\lambda_i}{i} \left( 1 - (1 - I_{Av})^i \right)^{1 - i} \tag{14} \]
which is linear in the $\lambda_i$, convex in the $\rho_j$ and concave in $I_{Av}$.

With sufficiently many iterations, the decoder will converge to the fixed point $x = f(g(x) = h(x)$, which for a well-designed code will occur at a value of $x$ very close to 1, corresponding to a low bit error rate. To prevent an undesirable error floor, we may wish to enforce a condition that the iteration does not get stuck at a fixed point prior to some target value $x = 1 - \xi$ for some small $\xi$. To this end, we require $x < h(x)$ for all $x \in (1 - \xi, 1)$ . For this range of $x$, we use a Taylor series expansion around $x = 1$, and obtain the condition $x < h(1) + h'(1) \cdot (x - 1)$, which is equivalent to \[
h'(1) = \lambda_2 \cdot \left( \frac{2 - C}{C} \right) \sum_{j \geq 1} j \rho_j - 1 < 1, \tag{15}\]
referred to as the stability condition. This condition is always met if $\lambda_2 = 0$ (this choice is usually bad for capacity-achieving codes). Otherwise it is equivalent to \[
\sum_{j \geq 1} j \rho_j < 1 + \frac{\lambda_2}{\lambda_2} \cdot \frac{C}{2 - C}. \tag{16}\]

III. CODE DESIGN

The prevailing approach to code design finds the degree distribution which results in the minimal rate loss from capacity. According to Theorem 1, this corresponds to a curve fitting problem. Equivalently, for fixed $\rho_i$, the maximum design rate is achieved by maximization of $\sum_i \lambda_i/i$. This results in the following linear program:

\textbf{Problem 1 (Capacity Approaching Code):} Given the channel capacity $C$ and check-node degree distribution $\rho_i$, find the variable-node degree distribution $\lambda^*$ as the solution to the following linear program:

\[
\max_{\lambda} \sum_i \frac{\lambda_i}{i} \tag{17}
\text{s.t. } \sum_i \lambda_i \cdot f_i(y) > g^{-1}(y), \quad y \in [0, 1] \tag{18}
(1 + \lambda_2) \frac{C}{2 - C} - \lambda_2 \sum_j j \rho_j > 0 \tag{19}
\lambda_i \geq 0, i = 1, 2, \ldots \tag{20}
\sum_i \lambda_i = 1. \tag{21}
\]

The constraints (18) ensure that there are no sub-optimal fixed points and (19) is the stability condition.

In many situations of practical interest, the implementation complexity of the decoder is limited, and it is therefore of interest to consider the design of codes that reduce decoder complexity, even if this is at the expense of reduced code rate. Complexity constraints may be introduced into the optimization problem (17) to a certain extent via (a) fixing the maximal variable node degree (which needs to be done anyhow in order to have a finite number of variables in Problem 1), or (b) introducing other linear constraints on the $\lambda_i$, e.g. corresponding to the cost of high degree nodes.

For a code optimized according to (17), operating at rates close to capacity, it may take many iterations to achieve the target bit error rate. It is therefore of great practical interest to consider the design of codes that achieve the best possible performance after a fixed number of iterations, denoted by $L$. One obvious approach would be to replace the objective function in (17) with the bit error rate at iteration $L$. From (12) and (13), we see that this is equivalent to maximizing the variable node information at iteration $L$,

\[
I_v^{(L)} = \left( \sum_j \frac{\lambda_j}{j} \right)^{-1} \sum_i \frac{\lambda_i}{i} \left( 1 - (1 - g(h^{L-1}(0)))^i \right) \tag{22}\]

$h^{L-1}(x) = f(g(f(g(\ldots f(g(x)))\ldots))$.

Note that $I_v^{(L)}$ is a multinomial of degree $L$ times the square of the maximal variable node degree. This could potentially present numerical difficulties for optimization. Furthermore, this formulation requires us to compute $h^L$ for each value of $L$ of interest. In the case of IRA codes and the BEC, we have analytical expressions for $f_i(y)$ and $g_i(x)$. In more general settings, we may only have numerical samples of these functions.

Motivated by (6), the key novel step in this paper is to introduce dummy variables $x_\ell, y_\ell, \ell = 0, 1, \ldots, L$. With reference to Figure 1, these are the values of the variable node extrinsic and a-priori information at iteration $\ell$. For given target rate $R$, we formulate our design problem as follows (where we have used (14)).

\textbf{Problem 2 (Minimum bit error rate after $L$ iterations):} Given $R < C$ determine the variable node degree distribution $\lambda^*_L$ and check node degree distribution $\rho^*_L$ that minimize the bit error rate after $L$ iterations as the solution to:

\[
\max_{\lambda, \rho, x, y} \left( R_d \sum_i \frac{\rho_i}{i} \right)^{-1} \sum_i \frac{\lambda_i}{i} \left( 1 - (1 - y_L)^i \right) \tag{22}\]

\text{s.t. } (1), (19), (20), (21) and \[
\rho_i \geq 0, \quad i = 1, 2, \ldots \tag{23}
\sum_i \rho_i = 1, \tag{24}
x_0 = 0, y_0 = g(x_0) \tag{25}
x_\ell = \sum_j \lambda_j f_i(x_{\ell-1}), \quad \ell = 1, 2, \ldots, L \tag{26}
y_\ell = \sum_j \rho_j g_i(x_\ell), \quad \ell = 1, 2, \ldots, L \tag{27}
\]

In this setup, the target rate $R$ would be chosen to be less than capacity, sacrificing some code rate for potential performance improvements after $L$ iterations.
The new constraints (25),(26),(27) ensure consistency of the decoding trajectory according to (6). The trajectory constraints are convex, however as discussed earlier, the objective function is neither convex nor concave.

We have empirically observed that there is very little bit error rate penalty incurred by replacing the bit error rate objective function (22) with $y_L$ or $x_L$ (i.e. maximize the information passed between the decoders). In fact, this may be of interest if the code is an inner component of some other outer iteration, e.g. a multi-user decoder. This results in the following optimization problem:

**Problem 3 (Maximal information after $L$ iterations):**

Given $R < C$ determine the variable node degree distribution $\lambda^{*L}$ and check node degree distribution $\rho^{*L}$ that maximizes the variable node extrinsic information after $L$ iterations as the solution to the following convex optimization problem:

$$\max_{\lambda,\rho} x_L \quad \text{s.t.} \quad (1), (19), (20), (21), (23), (24), (25), (26), (27)$$

Note that by “unrolling” the iteration, we obtain constraints that are independent of the maximum number of iterations $L$. Furthermore, we do not need to know the $L$-fold functional composition $f(g(f(\ldots)))$. Instead, we only need to know $f(y)$ and $g(x)$ individually. This could be an additional advantage if these characteristics are only available numerically (e.g. for other kinds of codes).

**IV. NUMERICAL EXAMPLE**

We seek for a code of rate $R = 0.5$ for transmission over a BEC with erasure rate $\delta = 0.3$, that achieves a low bit error probability with $L = 10$ decoding iterations. We consider variable node degrees from the set $\{2, 3, 4, 5, 6, 13, 20, 30\}$, and (for simplicity) the fixed check-node distribution $\rho_1 = 0.2, \rho_3 = 0.8$. (Note that degree-1 check nodes are required for non-systematic IRA codes.) Solving Problem 3 for maximal information, we obtain the variable node distribution $\lambda^{(1)}$, see Table I, and the EXIT function $f^{(1)}$, depicted in Fig. 2.

As an alternative approach, we designed a capacity-approaching code, according to Problem 1, for a BEC with a higher erasure rate, namely $\delta = 0.47$, to obtain a code rate of $R = 0.5$. We then use this code with $L = 10$ decoding iterations for the given BEC with $\delta = 0.3$. The resulting variable node distribution $\lambda^{(2)}$ is given in Table I, and the EXIT function is depicted in Fig. 2.

For the maximal-information code, the tunnel between the EXIT functions is clearly more open towards the end, and for a given number of iterations a better performance can be achieved. Resulting from that, the bit error rate after $L = 10$ iterations is $10^{-11}$ for the maximal-information code and only $10^{-3}$ for the 'capacity-achieving' code.

**ACKNOWLEDGMENT**

This work has been supported in parts by the Australian Research Council under the ARC Discovery Grant DP0986089.

**REFERENCES**


