Decoding with Early Termination for Raptor Codes

Ali AbdulHussein, Anand Oka and Lutz Lampe
Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada
Email: {alih, anando, lampe}@ece.ubc.ca

Abstract

Rateless codes, and especially Raptor codes, have received considerable attention in the recent past due to their inherent ability to adapt to channel conditions and their capacity-approaching performance. Since decoding of rateless codes typically involves multiple decoding attempts, early termination of such attempts is mandatory for overall efficient decoding. In this letter, we propose a new decoding scheme with early termination that is particularly suited for rateless codes. Simulation results for the example of the binary symmetric channel show complexity reductions (in terms of the total required number of decoding iterations) by 87% compared to conventional message-passing decoding and 54% compared to a recently proposed incremental decoding scheme for Raptor codes.

I. INTRODUCTION

Raptor codes [1] are a class of rateless codes that have been found to achieve a near-capacity performance for a variety of channels. Raptor codes are a concatenation of Luby transform (LT) [2] codes with a high-rate outer code such as a low-density parity-check (LDPC) code. They can be decoded with message-passing algorithms such as belief propagation (BP). Unlike fixed-rate codes like LDPC codes, the decoding graph of Raptor codes grows incrementally with time as new check symbols become available, and a sequence of decoding attempts is made on the corresponding factor graphs until successful decoding. To reduce the complexity of successive decoding attempts, the method of incremental decoding (ID) has recently been proposed in [3]. In ID, a new decoding attempt continues decoding from the results of the previous attempt.
In this letter, we propose an extension of ID, which not only makes use of the decoding “memory”, but also enables early termination of decoding attempts. To this end, we combine ID with a dynamic message-passing schedule and propose a stopping criterion that is sensitive enough to terminate decoding within a fraction of an iteration. The scheduling scheme we adopt is the informed dynamic scheduling (IDS) proposed in [4] for LDPC codes, and we refer to the new decoding method as incremental decoding with informed dynamic scheduling (IDIDS). Simulation results for transmission over the binary symmetric channel (BSC) show that IDIDS with early termination reduces decoding complexity, which is measured as the total number of decoding iterations until successful decoding, by more than 50% compared to ID, without affecting the rate-performance of the code.

The remainder of this letter is organized as follows. Section II briefly reviews transmission with Raptor codes. The proposed IDIDS algorithm together with the stopping criterion are developed in Section III. Simulation results for the different algorithms are presented and discussed in Section IV, and Section V concludes the paper.

II. TRANSMISSION WITH RAPTOR CODES

The transmission model consists of the two-stage Raptor-code encoder, a discrete-time channel, and a BP decoder. The first stage of the encoder, say the LDPC encoder, encodes the k-bit input vector into the k'-bit LDPC codeword. Then, the second stage, the LT encoder, linearly transforms this k'-bit LDPC codeword vector into an infinite stream of check bits, that passes through a binary-input symmetric channel with capacity C. The decoder collects channel output symbols progressively until it believes to have sufficient information to infer the k-bit input vector. For example, if an estimate of C is available at the receiver, \( n_{\text{min}} \approx \frac{k}{C} \) samples are collected for the first decoding attempt. If the decoded information is deemed erroneous, another decoding attempt is made after a new batch of \( n_{\text{inc}} \) samples has been received. If the number of decoding attempts required until successful decoding is denoted by \( \kappa \), then \( n = n_{\text{min}} + (\kappa - 1)n_{\text{inc}} \) is the number of total received samples and the achieved Raptor code rate is \( R = \frac{k}{n} \).

The default scheduling technique for BP decoding is to pass messages in parallel between variable nodes and check nodes in a number of iterations, which is known as “flooding” (FL) schedule. In message-reset decoding (MRD), BP decoding is started from scratch at every arrival of new batch of channel output samples. A more efficient technique is incremental decoding (ID)
proposed in [3], which continues decoding from the results of the previous decoding attempt. That is, the final messages after completing the previous decoding attempt are used for initialization of the current attempt. In the following, we refer to MRD and ID, with FL schedule, as MRDFL and IDFL, respectively.

III. DECODING WITH EARLY TERMINATION

In this section, we first propose the combination of ID with the sequential IDS from [4] to synergize their computational advantages (Section III-A), and then present a stopping criterion suited for termination of decoding attempts for Raptor codes (Section III-B). Together, this accomplishes efficient decoding with early termination and thus reduces complexity compared to the conventional BP decoding approach, i.e., MRDFL, and also IDFL from [3].

A. Incremental Decoding with Informed Dynamic Scheduling (IDIDS)

Unlike the FL schedule, sequential scheduling techniques prescribe a pre-defined sequence of variable-node or check-node updates [5], [6]. Since an update is a much smaller unit of computation than an iteration, sequential scheduling is particularly suited for early termination of a decoding attempt. We adopt one version of sequential scheduling, called informed dynamic scheduling (IDS), which was recently presented in [4] for LDPC codes. IDS repetitively calculates the reliability of each check node based on a metric called a “residual”, orders these residuals for all check nodes in a queue and updates the messages sent by the nodes with the highest residual values only. A full iteration is counted after the number of updates equals the number of check nodes in the factor graph.

The combination of ID with the IDS results in a novel technique for decoding Raptor codes, which we refer to as incremental decoding with informed dynamic scheduling (IDIDS). In IDIDS the decoder follows the ID procedure for propagating information from one decoding attempt to the next, and in each decoding attempt it uses the schedule defined by IDS to update the check nodes. Moreover, IDIDS also retains the ordering metric queue from the previous decoding attempt.

Compared to MRDFL and IDFL, additional operations are required for residual calculations and ordering in IDIDS. To address the former, we employ the “maxlog” approximation to approximate the residuals. This reduces the cost substantially and we observed no degradation
in the error-performance. Ordering can also be handled with modest cost as we only need to sort the full queue once at the start of the decoding process. New metrics can then be inserted into the queue progressively during the update process.

B. Stopping Criterion

In the rateless setting two distinct stopping criteria are required: one to stop the entire decoding process and another to stop a decoding attempt. For the former, we employ a cyclic-redundancy check code with only a few bits of additional redundancy. For the latter, we present a novel criterion which can stop a decoding attempt at a fraction of an iteration, that is before completing a full iteration in IDIDS.

The proposed criterion is composed of two independent detectors, $\mathcal{H}$ and $\mathcal{L}$, each of which functions optimally in different regions of the code rate $R$. Detector $\mathcal{H}$ measures the check-sum satisfaction difference over an interval of an iteration with the metric

$$\mu_H = \left| \frac{N(\ell)}{k' - k + n} - \frac{N(\ell - 1)}{k' - k + n} \right|,$$

where $N(\ell)$ is the number of satisfied check nodes after $\ell$ decoding iterations. Detector $\mathcal{L}$ employs the metric $\mu_L$ which measures the repetitiveness of variable nodes being updated between two consecutive intervals of $w$ updates of check nodes in one iteration. Let $\mathcal{V}_i$ be the set of all variable nodes updated during interval number $i$. Similarly, let $\mathcal{V}_{i+1}$ be the set of variable nodes updated during the interval number $i + 1$. We define $\mu_L$ as

$$\mu_L = \frac{|\mathcal{V}_i \cap \mathcal{V}_{i+1}|}{|\mathcal{V}_{i+1}|},$$

where $|\mathcal{V}_{i+1}|$ is the cardinality of $\mathcal{V}_{i+1}$. Then we formally define the stopping criterion as

$$[\mu_H \leq \gamma_H] \text{ or } [\mu_L \geq \Gamma_L],$$

where $\gamma_H$ and $\Gamma_L$ are user-defined thresholds independent of the instantaneous code rate. Note that $\mu_H$ is sampled at every iteration whereas $\mu_L$ is sampled at every interval of $w$ updates within one iteration. The stopping criterion essentially bridges the following two very different convergence behaviors exhibited by the IDIDS decoder. (i) In the high-rate region, information is scarce and typically several iterations are executed before the messages converge. Hence there is little incentive for stopping within a fraction of an iteration. Convergence in this region is
inferred when the status of check-sum satisfaction is not changing appreciably, which can be
detected by the metric $\mu_H$. (ii) In the low-rate region, however, convergence is rapid, often within
a single iteration, and hence it is worthwhile to try and stop in a fraction of an iteration. This
cannot be accomplished by $\mu_H$ because we need a more fine-grained detector. From simulations
we observed that in the low-rate region, convergence implies repetitive updates confined to a
small set of variable nodes, often in the form of a limit-cycle. This phenomenon can be sensed
by the metric $\mu_L$.

Since the detector $L$ is not efficient in the high-rate region, where the order of message updates
looks practically pseudo random, we need to use a combination of detectors $H$ and $L$ as in (3)
to achieve maximal efficiency. For simplicity, we may set $\gamma_H = 0$ and $\Gamma_L = 1$. This implies
that $\mu_H$ checks for strict invariance in check-sum satisfaction whereas $\mu_L$ checks for an update
limit-cycle.

IV. SIMULATION RESULTS

In this section, we show quantitative results for the complexity reduction achieved by the
proposed IDIDS method with the stopping criterion (3). Our measure of complexity is the total
number of iterations until decoding is successful, i.e., cyclic-redundancy check is positive. In
order to separate the effects of informed scheduling and incremental decoding, we compare
the complexities of the following decoding schemes: MRDFL($L_{\text{max}}$, $T$), IDFL($L_{\text{max}}$, $T$), and
IDIDS($L_{\text{max}}$, $T$), where $T$ is the interval in number of samples between successive decoding
attempts and $L_{\text{max}}$ is the maximal number of iterations per attempt. We note that $L = L_{\text{max}}$
was applied in previous works for Raptor codes (e.g. [3]), where $L$ is the actual number of
iterations in a decoding attempt. For a fair comparison with IDIDS, we apply the check-sum
satisfaction detector $H$ to stop decoding attempts in MRDFL and IDFL. In IDIDS, the interval
length $w = 100$ is chosen. As benchmark scheme, we consider MRDFL(100,100) without any
termination, i.e., $L = L_{\text{max}} = 100$, cf. [3]. As an interesting transmission example, we assume
the BSC with capacity $C = 0.50 \text{ bit/(channel use)}$. The Raptor code consists of a rate-0.95
regular LDPC code and an LT code generated using the degree distribution from [1, Table I].
The input-word length is chosen $k = 9500$ bits, and the first decoding attempt commences when
$k/C$ received samples are available.

First, we make sure that the different BP schedules and early termination do not affect the rate
performance of Raptor codes. To this end, Figure 1 shows a scatter plot for the achieved rate for 50 transmitted words (i.e., the rate $R = k/n$ for which the transmitted word was decoded correctly). That is, for each point the $x$-value is the rate achieved with the benchmark scheme, MRDFL(100,100) without stopping, and the $y$-value is the rate achieved with one of the other decoding schemes for the same transmitted data and received samples. We consider both $T = 100$ and $T = 1$ for the ID schemes, and set $L_{\text{max}} = T$ to fix the ratio $T/L_{\text{max}}$ as in [3]. It can be seen that all measured points lie very close to the 45-degree line, which clearly manifests the equivalent performance achieved with the different decoders. In case of IDFL, this is consistent with the findings in [3], according to which only the ratio $T/L$ determines the rate performance (assuming $L = L_{\text{max}}$ in [3]).

Next, we compare decoding complexities. Figure 2 shows the scatter plot for the required number of iterations for the same 50 transmissions and decoding schemes that were considered in Figure 1. In addition, lines with different slopes are included to emphasize the approximate multiplicative gains in terms of complexity reduction. A number of observations can be made. Firstly, the complexity of MRDFL can be reduced by a factor of two due to check-sum-satisfaction stopping. This is interesting in its own right, since only MRDFL without stopping was used in [3] for a comparison with IDFL. Secondly, we observe that IDFL accomplishes further notable complexity savings. IDFL(1,1) is somewhat advantageous over IDFL(100,100), which again is consistent with [3], where the use of $L = 1$ was advocated for incremental decoding. Thirdly, it can be seen that the proposed IDIDS is by far the most efficient decoding scheme. The total number of iterations is only 13%, 27%, and 46% of those required for MRDFL without stopping, MRDFL with stopping, and IDFL(1,1), respectively. We note that the new stopping criterion (3) and its ability to stop at a fraction of an iteration when employed with IDS are crucial for accomplishing early termination and hence yielding these considerable reductions in complexity. Furthermore, since the number of message updates in IDIDS is adapted to the amount of new information available during each decoding attempt, the choice of the interval $T$ between successive decoding attempts is not critical.

V. CONCLUSIONS

In this letter, we have proposed incremental decoding with informed dynamic scheduling (IDIDS) as an efficient decoding method for (rateless) Raptor codes. One of the essential
Fig. 1. Scatter plot for the realized rate for various methods. Arguments \((L_{\text{max}}, T)\) mean that a decoding attempt is made every \(T\) newly received samples with \(L \leq L_{\text{max}}\) iterations. Check-sum satisfaction stopping for MRDFL/IDFL. IDIDS uses early termination with (3). \(x\)-axis: MRDFL(100,100) with \(L = L_{\text{max}}\).

The ingredients of IDIDS is the early termination of decoding attempts so as to avoid unnecessary message updates. To this end, we have proposed a hybrid stopping criterion, which can stop a decoding attempt at a fraction of an iteration. Simulation results for the example of the BSC with a capacity of 0.5 bit/(channel use) have shown that the proposed IDIDS affords significant savings in decoding complexity without sacrificing performance in terms of realized rate.

REFERENCES


Fig. 2. Scatter plot for the total cost in number of iterations until successful decoding. Arguments \((L_{\text{max}}, T)\) mean that a decoding attempt is made every \(T\) newly received samples with \(L \leq L_{\text{max}}\) iterations. Check-sum satisfaction stopping for MRDFL/IDFL. IDIDS uses early termination with (3). x-axis: MRDFL(100,100) with \(L = L_{\text{max}}\).


