Designing Rate-Compatible Irregular Repeat Accumulate Codes through Splitting

Min Xiao, Lin Wang, *Senior Member, IEEE*, and Jing Li, *Senior Member, IEEE*

***Abstract*—A new scheme of rate-compatible (RC) irregular repeat accumulate (IRA) codes is proposed which can be encoded and decoded using a single encoder-decoder pair and which can provide good performance across a wide range of code rates. Conventional strategies generally start from a low-rate mother code and puncture the rate up. Considering that the initial transmissions (and hence the higher-rate codes) are most important, we propose to start from a good high-rate code and "split" the rate down. Rather than do it gradually in a greedy manner, we propose to design a good one-shot-split to control the worst-case (lowest-rate) performance, and subsequently work out all the intermediate rates. We show the proposed strategy is both simple and powerful resulting in RC-IRA codes that outperform the existing ones.**

***Index Terms*—Rate-compatible, IRA codes, splitting.**

I. Introduction

TECHNIQUES for implementing rate-compatible (RC) channel codes have primarily pursued a *puncturing* approach, which starts from a good low-rate code and achieves higher rates by incrementally puncturing parity bits. In the context of low-density parity-check (LDPC) codes, although puncturing allows a single encoder-decoder pair to serve all the different codes in the RC family, these codes typically exhibit increasingly widening gaps to the capacity as the number of punctured parity bits and hence the code rate increase. This can be disappointing in practice, since in a hybrid automatic-repeat-request (HARQ) system with incremental retransmission, one expects the initial transmission(s) of the high-rate code(s) to be powerful, so as to avoid cumbersome re-transmission as much as possible.

This shortcoming can be addressed with an opposite philosophy: starting from a good high-rate code and work the rate down. The techniques of *extending* and *splitting* have been suggested for this purpose. Since extending and splitting may result in different code graphs, caution should be exercised in order to ensure the resultant codes are indeed rate-compatible and can share exactly the same LDPC encoder/decoder.

This paper considers the design of RC irregular repeat accumulate (IRA) codes using one single encoder-decoder pair through splitting. The reasons for choosing IRA codes are two-fold: First, IRA codes are known for both good performance *and* low encoding and decoding complexity; second, there exists an interesting reciprocal relationship between puncturing and splitting for IRA codes [1] [2], which inspires new ways to design RC-IRA codes with good performance at the highest rate. The proposed algorithm here exercises a philosophy different from the conventional "low-rate-up straight" or "high-rate-down straight" approaches, and achieves rate-compatibility in three steps: design a good highest-rate code, split it all the way to the lowest rate, and progressively reverse the splitting operation to construct the entire family of rate-compatible codes by puncturing. The proposed algorithm benefits from the single-encoder-decoder-pair advantage, and at the same time, allows the highest-rate code to be optimally designed/selected. The key step is to design the one-shot-split pattern and the reverse puncturing pattern in a controlled way to ensure good performance along the way and especially in the lowest-rate case. We discuss the general guideline, propose three criteria to guide practical splitting, and demonstrate their effectiveness through a greedy approach.

Manuscript received December 24, 2010. The associate editor coordinating the review of this letter and approving it for publication was M. Ardakani.

This work was supported by the US NSF under Grants No. CCF-0635199, CCF-0829888, and CMMI-0928092, the China NSF under Grant No.60972053, and the NSF of Fujian Province in China under Grant No.2008J0035.

M. Xiao and L. Wang are with the Department of Communication Engineering, Xiamen University, Xiamen, Fujian, China, 361005 (e-mail: mix209@gmail.com, wanglin@xmu.eud.cn).

J. Li is with the Department of Electrical and Computer Engineering, Lehigh University, Bethlehem, PA, USA, 18015 (e-mail: jingli@ece.lehigh.edu).

Digital Object Identifier 10.1109/LCOMM.2011.080811.102549

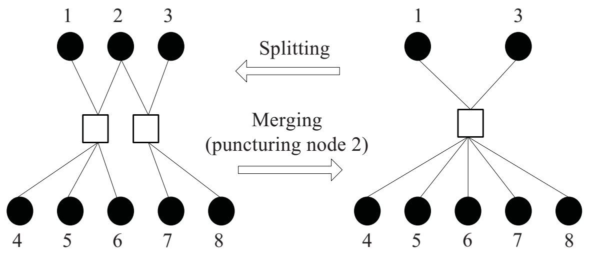

Fig. 1. Check node splitting and merging. Upper and lower circles denote parity and information nodes respectively, and squares denote check nodes.

II. Relation between Puncturing and Splitting

It is known that the degree-2 variable nodes (parity nodes) of an IRA code are linked together in a zig-zag pattern (the dual diagonal part of the parity check matrix). Specifically, if one such degree-2 parity node is punctured, then the two check nodes connecting to it can be merged into one and still maintain the zig-zag structure [1]; see Fig. 1. Following the convention, we call the original Tanner graph with unmerged checks the *oTanner graph*, and the equivalent/effective Tanner graph after the merge the *eTanner graph*. It can be proven that the oTanner graph and the eTanner graph exhibit the same finite-length performance given enough iterations.

On the other hand, a degree-k check node of an IRA code (or a general LDPC code) can also be split into two check nodes of degrees i and $j = 2 + k - i$ respectively, by inserting a new degree-2 parity node; see Fig. 1. If all the newly-added degree-2 parity nodes preserve the dual-diagonal parity structure, then not only does the resultant code remain an IRA code and hence continue to enjoy linear-time encodability, but the girth will likely also be increased to provide improved performance [2]. This suggests a possibility for constructing good low-rate IRA code(s) from a high-rate one by splitting. However, the rate-compatible splitting scheme proposed by [2] still needs different encoder-decoder pairs for different code rates and requires the parity-check matrices for all the code rates to be saved. Below we provide guidelines for effective splitting which will result in RC-IRA codes that perform well and fit in a single encoder/decoder pair.

III. Constructing RC-IRA Codes by Splitting

The reciprocal relationship between puncturing (check node merging) and splitting for IRA codes inspires us to design RC-IRA codes through splitting. The proposed procedure involves three steps:

Step 1: *Optimize an IRA code at the desired highest rate, there after referred to as the mother code*, to ensure quality initial transmissions. The optimization of the code can be accomplished using the existing efficient algorithms proposed in literature, e.g. [3] and [4].

Step 2: *Starting from the mother code, split the check nodes and correspondingly inject new parity nodes to construct the lowest-rate code of interest, thereafter referred to as the proxy code.* To ensure a consistent graph structure (and hence encoder/decoder-compatibility), we restrict the added parity nodes to having degree 2 only and preserving the double diagonal structure. The key is to design a good splitting procedure (will be discussed later), as it directly determines the performance of the proxy code.

Step 3: *Starting from the proxy code, progressively puncture the previously-injected parity nodes and merge back the check nodes to obtain codes of increasing rates.* We recommend uniform puncturing, such as the algorithm in [5], because it has produced the best-performing puncturing patterns for IRA codes in literature.

It is obvious from the above three steps that the resultant RC-IRA scheme is a single-encoder-decoder-pair system which uses the same encoder and decoder corresponding to the proxy code for all the different code rates.

Splitting Procedure: The splitting procedure in Step 2, which is a one-shot splitting that gets the highest rate directly to the lowest rate, involves the design of two important aspects: the *splitting pattern*, which determines how many new check nodes each original check code should split into, and the *connection pattern*, which determines what degrees should the new check nodes take and how they should be connected to the original and the injected variable nodes.

Splitting Pattern: It has been shown that for IRA codes, the optimal puncturing pattern is uniform puncturing among the parity bits [5]. Our splitting pattern is designed based on this fact and the reciprocity between puncturing and splitting. Consider splitting the M original check nodes in the mother code into a total of m (new) check nodes in the proxy code (m is also the total number of parity nodes in the proxy code). If it is a good splitting, then reversing the splitting operation should lead to a uniform puncturing pattern for the highest rate. Specifically, given the values of M and m (i.e. highest and lowest code rates in the RC family), one can easily determine the uniform puncturing pattern, and then work out the corresponding splitting pattern. Let binary vector $\mathbf{P}=\{p_1,p_2,...,p_m\}$ denote a given puncturing pattern, where $p_i = 0$ indicates ith parity node is punctured, and $p_i = 1$ otherwise. Without loss of generality, suppose the first parity bit is always preserved, i.e. $p_1=1$. Let $\mathbf{K} = \{k_1,k_2,...,k_M\}$ denote the numbers of new check nodes which result from splitting the 1st, 2nd, $\cdots$ and Mth original check nodes, respectively, where $\sum_{i=1}^{M} k_i = m$. The splitting pattern $\mathbf{K}$ can be quickly derived from $\mathbf{P}$, by noting that (k_i-1) is the number of zeros between ith "1" and $(i+1)$th "1" in $\mathbf{P}$.

Example: Suppose we are to design for $M = 5$ and $m = 18$. We can first get a desirable uniform puncturing pattern such as $\mathbf{P} = \{100010010001001000\}$ ($\mathbf{P}$ is not unique), and subsequently determine $\mathbf{K} = \{4,3,4,3,4\}$.

Connection Pattern: The splitting pattern provides the number of new check nodes that each original check node should be split into. To assign proper degrees to the new check nodes and to arrange proper connection between the new check nodes and variable nodes is also critical to the code performance. Here we propose three criteria to guide the realization of the oTanner graph of the proxy code:

A. concentrate the degrees of new check nodes,
B. uniformly connect variable nodes of different degrees to each new check node,
C. maximize the girth of the underlying graph.

The first criterion is based on plenty of investigations in literature that a good LDPC code should in general have check nodes with similar degrees. The purpose of Criterion B makes each check node connect to both high-degree and low-degree variable nodes. This will avoid the case where some check nodes connect only to low-degree variable nodes that are less protected. We will show the necessity of this criterion in the next section. The rational for Criterion C is that a large girth can help enlarge the size of the smallest stopping/trapping sets and will in general make the message-passing decoding algorithm more effective. In our algorithm, Criterion A takes precedence over Criterion B which in turn takes precedence over Criterion C.

It should be noted that our splitting procedure differs from a previous one proposed by Shi *et al* in [6]. Shi *et al* adopted a progressive (greedy) procedure that uses a set of "mini-step" uniform splitting patterns, each getting the current code to the immediate next code-rate level, without worrying about the cumulative effect on the last one. In comparison, ours is a one-shot splitting that gets the highest rate directly down to the lowest rate. After the lowest-rate code (long-term and worst case) is under control, we then re-combine some of the splits we did, to claim all the intermediate code-rate levels. To illustrate, consider a simple example of splitting 1 check of degree 20 to 2 checks and then to 3 checks. Progressive splitting will result in $20 \Longrightarrow 11|11 \Longrightarrow 7|6|11$, whereas one-shot splitting will directly lead to $8|8|8$ for the last level. Our study shows that the one-shot splitting in general leads to better performance over all. Additionally, three criteria are proposed to guide the connections between the variable nodes and the newly-split checks in this work, while [6] determined the connections with density evolution.

TABLE I
PARAMETERS OF DIFFERENT RATE-COMPATIBLE IRA CODES

Code name	Rate-compatible algorithm	Mother code			Proxy code	
		Rate	Information block length	Construction method	Rate	Connection criteria
Proposed-1	Proposed one-shot splitting algorithm	0.9	512	Algorithm in [4]	0.5	A,C
Proposed-2		0.9	512		0.5	A,B,C
Proposed-3		0.9	1024		0.5	A,B,C
Conventional	Puncturing, in [5]	0.5	512	Algorithm in [4]		
Code-Yue	Puncturing, in [5]	0.5	1024	Algorithm in [3]		
E^2RC	Progressive splitting, in [6]	0.5	1024	Algorithm in [6]		

IV. EXAMPLES AND SIMULATIONS

We now present and compare the simulated performance of our codes with a variety of RC-IRA codes in literature (see Table I) over binary-input additive white Gaussian noise (AWGN) channels. The rate-0.5 mother IRA code in the conventional puncturing algorithm uses an optimized degree profile $\lambda(x) = .0001x^0 + .3040x + .2768x^2 + .4191x^6$ and $\rho(x) = .3886x^5 + .6114x^6$. The proposed algorithms use a rate-0.9 optimized IRA code with a degree profile $\lambda(x) = .0002x^0 + .0460x + .3794x^2 + .5744x^6$ and $\rho(x) = .9103x^{42} + .0897x^{43}$. A maximum of 100 iterations of the sum-product decoding algorithm is used for all the codes. When applicable, the puncturing patterns are obtained by the algorithm in [5].

Fig. 2 compares the frame error rate (FER) of our codes with those designed by conventional puncturing at information block size of 512. Our RC-IRA codes, and especially "proposed-2", significantly outperforms "conventional" for a wide rage of rates and pushes the error floors lower, particularly at rate 0.9. Comparing the two codes we designed, it can be seen that "proposed-2" generally outwins "proposed-1", which clearly speaks for the necessity of the connection Criterion B, especially when designing for a wide rate range.

We also compare our codes with the best-reported RC-IRA codes in literature in Fig. 3. All the codes have information block length of 1024. It is encouraging indeed to see that our codes ("proposed-3") also outperform Code-Yue [5] and E^{2}RC [6] for a wide range of rates both at the waterfall region and the error floor region!

V. CONCLUSION

Aiming at quality initial transmissions (high-rate codes) in a HARQ system and exploiting the reciprocal relation between puncturing (check node merging) and splitting for IRA codes, we have developed a new splitting-based procedure to construct finite-length rate-compatible IRA codes that can provide a wide range of rates using a single encoder/decoder pair. Exercising a philosophy different from the conventional "low-rate-up" or "high-rate-down", the algorithm starts from designing a powerful high-rate code, splits it all the way down to the lowest rate of interest, and subsequently re-gains the intermediate levels by merging back some of the split check nodes. Simulation shows that the resultant codes not only guarantee superb performance at very high rate (0.9), but also promise very good performance across the board. That they outperform the best RC-IRA codes reported so far speaks volume for the effectiveness of the proposed strategy.

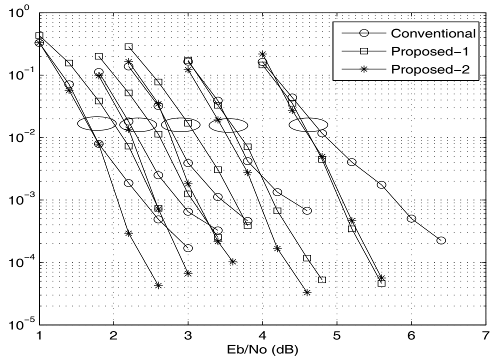

Fig. 2. The FER performance comparison of the proposed RC-IRA codes and the conventionally punctured RC-IRA codes. Information block length is 512. Rates are 0.5, 0.6, 0.7, 0.8 and 0.9 from left to right.

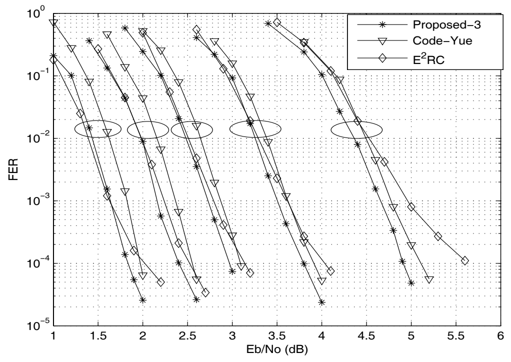

Fig. 3. The FER performance comparison of the proposed RC-IRA codes, the best-reported RC-IRA codes in literature and the E^{2}RC codes. Information block length is 1024. Rates are 0.5, 0.6, 0.7, 0.8 and 0.9 from left to right.

REFERENCES

[1] S. Hong and J. Cho, "Rate-compatible puncturing for finite-length low-density parity-check codes with zigzag parity structure," *IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications,* Sep. 2006.

[2] H. Joo, S. Hong, and D. Shin, "Design of rate-compatible RA-type low-density parity-check codes using splitting," *IEEE Trans. Commun.,* vol. 57, no. 12, pp. 3524-3528, Dec. 2009.

[3] G. Yue, B. Lu, X. Wang, and M. Madihian, "Analysis and design of finite-length LDPC codes," *IEEE Trans. Veh. Technol.,* vol. 56, no. 3, pp. 1321-1332, May 2007.

[4] A. Serener, B. Natarajan, and D. M. Gruenbacher, "Optimized LDPC codes for OFDM and spread OFDM in correlated channels," in *Proc. IEEE VTC2007-Spring,* Apr. 2007, pp. 2276-2280.

[5] G. Yue, B. Lu, X. Wang, and M. Madihian, "Design of rate-compatible irregular repeat accumulate codes," *IEEE Trans. Commun.,* vol. 55, no. 6, pp. 1153-1163, June 2007.

[6] C. Shi and A. Ramamoorthy, "Design and analysis of E^2RC Codes," *IEEE J Sel. Areas Commun.,* vol. 27, no. 6, pp. 889-898, Aug. 2009.