H-ARQ Rate-Compatible Structured LDPC Codes

Mostafa El-Khamy *, Jilei Hou ◇ and Naga Bhushan ◇

* Electrical Engineering Dept. — California Institute of Technology — Pasadena, CA 91125

◇ Qualcomm — 5775 Morehouse Drive — San Diego, CA 92121

* Email: mostafa@systems.caltech.edu

***Abstract*— In this paper, we design families of rate-compatible structured LDPC codes suitable for hybrid ARQ applications with high throughput. We devise a systematic technique of low complexity for the design of structured low-rate LDPC codes from higher rate ones. These codes have a good performance on the AWGN channel and are robust against erasures and puncturing. The codes designed here are protograph-based codes and have fast encoding and decoding structures. These low rate codes are used as the parent codes of rate-compatible families. Then, we propose a number of algorithms for puncturing the codes in a rate compatible manner to construct codes of higher rates. The two most promising ones are the random puncturing search technique and progressive node puncturing. We show that using the techniques in this paper one could construct a high throughput rate compatible family with codes whose rates are in the range from 0.1 to 0.9 and which are within 1 dB from the channel capacity and have good error floors.**

I. Introduction

In this paper, we design rate-compatible (RC) families of LDPC codes for incremental redundancy hybrid automatic repeat request (H-ARQ) applications [1]. Our codes are structured (protograph-based) codes [2], [3]. We first design low rate structured LDPC codes which have low thresholds on the AWGN channel as well as a good performance on the erasure channel. Other approaches to design low rate protograph codes were previously considered [4]. These codes have the property that the punctured (hidden) variable nodes have a very high degree and thus are not expected to perform well under further puncturing. Rate-compatible structured LDPC codes were considered [5], [6]. In these families, the information block size is not constant which makes them not suitable for H-ARQ applications. Puncturing of LDPC codes was investigated by a number of researchers. The puncturing fraction of each set of variable nodes with a certain degree, in the parent code, has been optimized based on asymptotic analysis [7]. This approach is not directly applicable to the codes considered here since it is often the case that all the nodes to be punctured are of the same degree (degree 2). By modeling punctured nodes as erasures, it was recently shown that there exists a cutoff rate, R_c, which depends on the degree distributions and the rate of the parent code such that one could not find a code with a rate $R > R_c$, through puncturing the parent code, that performs well under density evolution [8]. Motivated by this discussion and taking also into account that our parent code already has punctured (hidden) variable nodes, we propose a number of algorithms for puncturing the structured code to get an H-ARQ RC family. Moreover, both puncturing (for higher rates) and extending [9] or information shortening [10] (for lower rates) are essential to get good codes over all the desired rate range. In a related work (not the focus of this paper), we can obtain lower rate (rateless) codes through extending the low rate codes designed here by concatenating them with inner low density generator matrix (LDGM) codes, as suggested in [11], which is similar to Raptor codes [12].

II. Design of Low Rate Codes

Starting from a protograph with a relatively high rate, we devise a systematic technique for constructing protographs of lower rates. These codes should have good performance on both AWGN and erasure channels. A code of the desired length is constructed from the protograph by the progressive edge growth (PEG) algorithm [13] or the ACE-PEG algorithm [14] or other variations such as circulant-PEG [4]. We use a version of the PEG algorithm adapted for protograph codes. We assume that the base protograph, B, has N variable nodes and C check nodes. We assume that E of the variable nodes are punctured (hidden). The number of systematic input nodes is denoted by $K = N - C$. The rate of the protograph is $R_B = \frac{K}{N-E}$. The following algorithm gives a systematic technique for deriving a protograph L of rate $R_L < R_B$ starting from the protograph B.

Algorithm 1: Construction of low rate protographs

1) Copy the base graph, B, α times.
2) Construct another protograph B_α from the α copies of B using the PEG algorithm (or its variants) to maximize the girth of the protograph.
3) Construct the protograph L from B_α by pruning
$$\beta = \alpha K \frac{R_B - R_L}{R_B(1 - R_L)} \tag{1}$$
systematic input nodes and the edges connected to them.
4) The choice of the pruned input nodes is optimized to get good AWGN and erasure thresholds.
5) Further optimization can be done by a small number of edge operations. □

The PEG algorithm, and its variants, often have some randomness in them. In such a case, step 2 can be repeated to a maximum number of iterations and the protograph with the largest girth is chosen. Since L is obtained from B_α by pruning nodes and edges, then L is a subgraph of B_α. This implies that the girth of L is at least as large as the girth of B_α. Then maximizing the girth of B_α is equivalent to a constrained

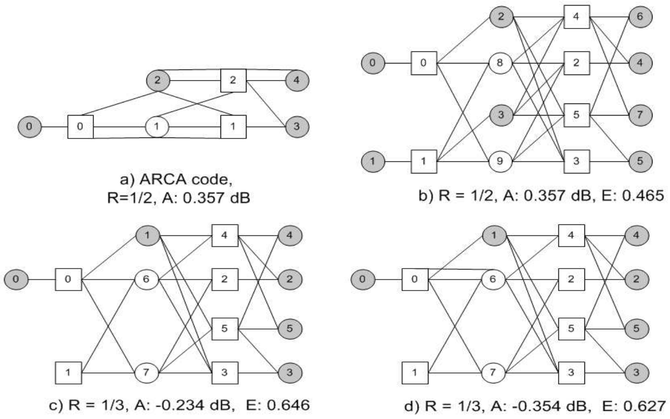

Fig. 1. Construction of rate $1/3$ protograph.

maximization of the girth of L. It is possible to construct the lifted graph with a larger girth if the girth of L is larger. This is a favorable property as a larger girth often corresponds to a low error floor at the high signal to noise ratio (SNR) region. In step 4, optimization can be done by choosing the pruned input nodes at random while using a gradient descent or simulated annealing approach [3] to pick the graph with the best threshold. One can also choose these nodes in a systematic way. For example, if $\beta = \gamma K$, then γ copies of the K input nodes are pruned from B_α. In most cases, the choice of the copies that are pruned does not affect the performance due to the symmetry in the protograph B_α across these copies. If no such symmetry exists, it is feasible to try all possible combinations of the γ copies and pick the combination that will give the lowest threshold. This is due to the small size of the protograph, which makes the search of the threshold for the protograph using density evolution (DE) [15] a fast process. As suggested in step 5, additional optimization of the protograph can be done by adding, removing and swapping edges in L. As we will demonstrate by example, carefully adding a very small number of edges is enough to get protographs with a better threshold. This step can be done by hand. However, it can also be automated in a simulated annealing setting.

Here we consider a popular instance of the algorithm, when the base protograph is a half rate code, $R_B = 1/2$, and the target rate $R_L = 1/T$, where T is an integer greater than 2. In this case the parameters are as follows:

$$\alpha = T-1 \;\; \& \;\; \beta = (T-2)K. \tag{2}$$

It is straightforward to confirm that

$$R_L = \frac{K(T-1) - K(T-2)}{(T-1)(N-E) - K(T-2)} = \frac{1}{T}. \tag{3}$$

A. Construction Examples

The base code of choice is the rate $1/2$ ARCA code, shown in Fig. 1.a, designed by Divsalar *et. al* [5]. In the figures accompanying the following examples, the AWGN and erasure thresholds are labeled by A and E respectively. Check and

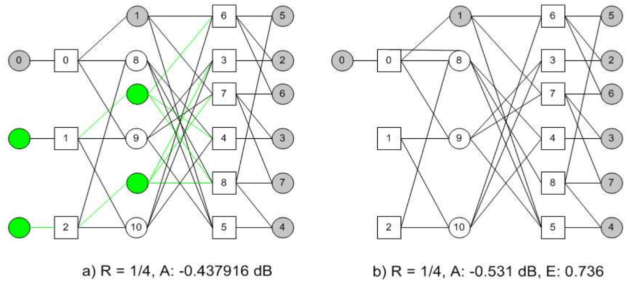

Fig. 2. Construction of rate $1/4$ protograph.

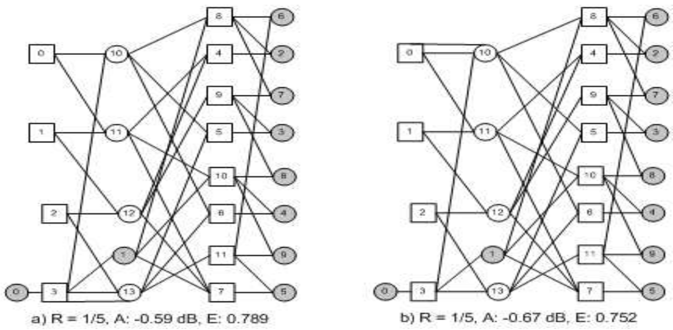

Fig. 3. Construction of rate $1/5$ protograph.

variable nodes will be represented by squares and circles respectively. Transmitted variable nodes are gray in color while punctured (non-transmitted) variable nodes are white. For asymptotic analysis (in the length of the code), we used DE to determine the AWGN and erasure thresholds [15].

The construction steps for a rate $1/3$ protograph are shown in Fig. 1. The protograph in Fig. 1.b is the PEG-lifted graph of two copies of the ARCA code in Fig. 1.a. The rate $1/3$ protograph in Fig. 1.c results by systematically pruning one copy of the input nodes and its AWGN threshold is only 0.26 dB away from the capacity. By simply adding one edge in Fig. 1.d, between C0 and V6 (C and V stand for check and variable respectively) the gap to capacity is reduced to only 0.14 dB. It is to be noted that both codes have a good performance on the erasure channel. However, the better threshold on the AWGN channel (d) comes at a slight degradation of the performance on the erasure channel (0.019 difference in erasure thresholds). Similar observations can be seen in the designed protographs shown in Fig. 2 and Fig. 3 for rates of $1/4$ and $1/5$ respectively. We compared our codes with the RC 3GPP turbo codes for a payload of 4K in Fig. 4, (T and HIT denote the AWGN threshold and the number of half iterations respectively). The codes were constructed using PEG. It is expected that the ACE-PEG algorithm will result in lower error floors. In general, our codes have about 0.3 dB gain over turbo codes at a FER of 10^{-2} (which is our comparison criterion for such applications). We can also show that the speed gains in terms of throughput are about 200%.

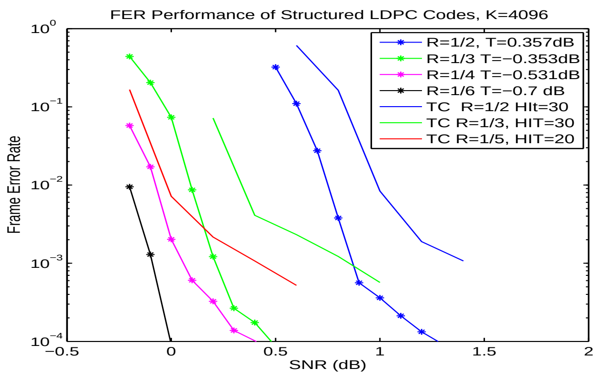

Fig. 4. Comparison with 3GPP turbo codes.

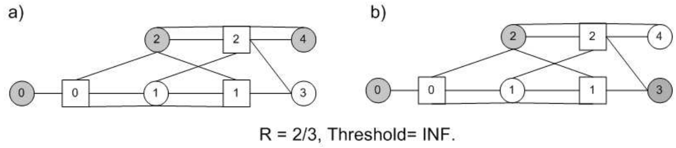

Fig. 5. Puncturing on the protograph level.

Rate	0.5	0.6	0.65	7	0.75	0.8	0.85	0.9
Gap	0.171	0.274	0.279	0.294	0.372	0.41	0.434	0.442

TABLE I
1K PAYLOAD RC FAMILY WITH REGULAR-IRREGULAR PUNCTURING

III. PUNCTURING ALGORITHMS

In this section, we will focus on devising schemes for puncturing the codes designed in the previous section to obtain higher rate codes in a RC family.

Structured (regular) puncturing can be achieved by puncturing on the protograph level. This will have the advantage of low storage requirements. However, this approach has two major drawbacks. The first drawback is that the set of feasible rates obtained by puncturing is very small. Consider the ARCA code in Fig. 1.a. There are two redundancy symbols in the protograph. In this case the feasible rates are only $2/3$ (puncturing any of the two redundancy nodes) and unity (puncturing both redundancy nodes). The second major problem is that these structured protographs already have punctured nodes, which are treated as erasures by the belief propagation (BP) algorithm. As seen in Fig. 5, puncturing any redundancy node will result in at least two edges connected to any check being connected to erased variables. BP cannot start as the set of punctured nodes will form a *stopping set*.

A. Regular Irregular Puncturing

We can see that for such codes irregular puncturing on the lifted graph will yield better results. This also gives the flexibility of choosing any family of required rates. In our proposed scheme, the puncturing pattern will be as regular as possible with respect to the preceding codes in the family. For the higher rates it becomes more irregular and random-like with respect to the parent code. Let $\mathcal{C}(R_m)$ denote the code in the rate-compatible family with rate R_m. We will assume that $\tilde{N}$ and $\tilde{C}$ are respectively the number of variable nodes and check nodes in the Tanner graph of the lifted code. The *regular-irregular puncturing algorithm* is formulated as follows:

Algorithm 2: Regular-Irregular Puncturing

1) Start with a parent code $\mathcal{C}(R_0)$, with rate R_0.
2) For each rate $R_m \in \{R_1, R_2, .., R_p\}$, $R_m > R_{m-1}$
   a) Find the set, Ψ_m of non-punctured redundancy nodes in $\mathcal{C}(R_{m-1})$. The cardinality of Ψ_m is $\tilde{N} - \tilde{E}_{m-1}$, where $\tilde{E}_{m-1}$ is the number of punctured variable nodes in the Tanner graph of $\mathcal{C}(R_{m-1})$.
   b) Calculate the number P_m of nodes to be punctured to go from rate R_{m-1} to rate R_m
   $$P_m = \left\lfloor \tilde{N} - \frac{\tilde{N} - \tilde{C}}{R_m} - \tilde{E}_{m-1} \right\rfloor .$$
   c) Calculate α and β such that puncturing pattern on the non-punctured set is as regular as possible: $\alpha = \lfloor \frac{\tilde{N} - \tilde{E}_{m-1} - P_m}{P_m - 1} \rfloor$ & $\beta = \tilde{N} - \tilde{E}_{m-1} - \alpha(P_m - 1) - P_m$.
   d) The puncturing pattern on Ψ_m is as follows $\{x_1 \underbrace{...}_{\alpha} x_2 \underbrace{...}_{\alpha}, ..., x_{P_m}, \underbrace{...}_{\beta}\}$, where x_i denotes the position of the ith punctured node. □

We have to emphasize, that the set of designed rates has to be carefully chosen to have a good performance. An RC family within 0.5 dB from the capacity and a parent rate $1/2$ was obtained for a payload of 1K as shown in Table I.

B. Random Puncturing

Regular and irregular puncturing patterns may not result in a good family of codes. We construct an RC family by searching for the best random puncturing pattern on the lifted graph in a fast way. The algorithm is stated as follows

Algorithm 3: Random Puncturing Search Algorithm

1) Start with a parent code $\mathcal{C}(R_0)$, with rate R_0.
2) For each rate $R_m \in \{R_1, R_2, .., R_p\}$, $R_m > R_{m-1}$
   a) Initialize the winning SNR, $S_w = \infty$.
   b) Find the set Ψ_m (2.a Alg. 2).
   c) Calculate P_m (2.b Alg. 2).
   d) Obtain a code $\mathcal{C}'$ by randomly puncturing P_m redundancy nodes in Ψ_m.
   e) By density evolution, test if the code has negligible error probability at an SNR S_w;
      - Yes:
        – Search for the new threshold S_n of this code in the range $\{-\infty$ to $S_w\}$.
        – Set $S_w = S_n$.
        – Set the winner code $\mathcal{C}(R_m)$ to be $\mathcal{C}'$.
      - No: Skip
   f) Repeat random search till a maximum number of iterations. (Go to 2.d.) □

Rate	Capacity	Gap R 1k	Gap P 1k	Gap R 4k	Gap P 4k
0.5	0.188	0.171	0.171	0.171	0.171
0.6	0.679	0.188	0.237	0.188	0.236
0.7	1.270	0.262	0.392	0.237	0.376
0.8	2.033	0.305	0.659	0.312	0.578
0.9	3.198	0.426	0.769	0.376	0.685

TABLE II
1K AND 4K PAYLOAD RC FAMILIES WITH RANDOM (R) AND PROGRESSIVE (P) PUNCTURING. PARENT RATE IS 0.5.

In step 2.e, searching for the threshold can be done by iteratively bisecting the range to select a test SNR and see if this test SNR achieves zero error. The process is repeated till a desired accuracy in the SNR. The *random search algorithm* is a greedy algorithm which searches for the best code at each design rate.

C. Progressive Node Puncturing

We devise a systematic algorithm that progressively chooses the puncturing pattern that (i) maximizes the number of checks in the graph which are connected to only one punctured variable while (ii) minimizing the average number of punctured variable nodes connected to each check and (iii) maximizing the connectivity between the checks, connected to only one punctured variable node, and the other punctured nodes.

The reasoning behind condition (i) is that a check (of degree > 1) with only one punctured variable node connected to it will transmit non-zero information to the punctured node and the single punctured node could recover. However, maximizing the number of checks with only one punctured node connected could result in some other checks having a large number of punctured nodes connected to them. A check with more than one punctured node connected will transmit zero information to all the punctured nodes unless all but one of these punctured nodes are recovered by message passing from other checks. Thus, it is crucial to minimize the maximum number of punctured nodes connected to any check, which implies (ii). This also implies that checks with no or only one punctured variables should have a high connectivity to the other checks with more than one punctured node, which is condition (iii).

Algorithm 4: Progressive Node Puncturing-I

For each rate R_m, obtain $\mathcal{C}(R_m)$ from $\mathcal{C}(R_{m-1})$ by progressively puncturing P_m nodes from the set Ψ_m (Alg. 2.2a,b);

1) For each check $c \in \mathcal{N}(v)$ such that $v \in \Psi_m$, calculate [1] $F(c) = |v : v \in \mathcal{N}(c)$ & v is punctured$|$
2) For each $v \in \Psi_m$, calculate
   a) $G(v) = |c : c \in \mathcal{N}(v)$ & $F(c) = 1|$
   b) $H(v) = \sum_{c \in \mathcal{N}(v)} F(c)$
3) While $p < P_m$ (puncture a node v_p)
   a) $\Gamma = \{v_r : v_r \in \Psi_m$ & $G(v_r) = \min_{v \in \Psi_m} G(v)\}$.
   b) If $|\Gamma = 1|$; $v_p = \Gamma$;
   c) Else
      i) $\Omega = \{v_r : v_r \in \Gamma$ & $H(v_r) = \min_{v \in \Gamma} H(v)\}$.
      ii) If $|\Omega| = 1$, $v_p = \Omega$.
      iii) Else, choose v_p at random from Ω.
   d) Puncture v_p and update $\Psi_m = \Psi_m \setminus v_p$.
   e) $\forall\ c \in \mathcal{N}(v_p)$: $F(c) := F(c) + 1$ and $\forall\ v \in \mathcal{N}(c) \bigcap \Psi_m$ update $H(v)$ and $G(v)$.
   f) Increment p, $p := p + 1$. □

[1] $\mathcal{N}(x)$ denotes the set of neighbors of the node x.

The previous algorithm implements conditions (i) and (ii). To implement condition (iii), we define the puncturing score of a check c to be the cardinality of the set of punctured variables connected to the checks reached by a two level expansion of the support tree of check c,

$$\mathbb{S}(c) \triangleq \sum_{c' : c' \in \mathcal{N}(v)\ \&\ v \in \mathcal{N}(c)} F(c'). \tag{4}$$

The puncturing score of the graph G is defined by

$$\mathbb{S}_G \triangleq \sum_{c \in G\ \&\ F(c)=1} \mathbb{S}(c). \tag{5}$$

The puncturing score is an approximate and an efficient way to measure how well the checks, with one punctured node, are connected to the other checks with punctured nodes. The progressive node puncturing (PNP) algorithm is modified as follows to incorporate condition (iii);

Algorithm 5: Progressive Node Puncturing-II

1) Initialize $\mathbb{S}_G = 0$ & $t = 1$.
2) While $t < t_{max}$
   a) Change the random seed
   b) Obtain the code $\mathcal{C}_t(R_m)$ from $\mathcal{C}(R_{m-1})$ by Alg. 4.
   c) Calculate the puncturing score of the Tanner graph of $\mathcal{C}_t(R_m)$, $\mathbb{S}_G(t)$.
   d) If $\mathbb{S}_G(t) > \mathbb{S}_G$, set $\mathcal{C}(R_m)$ to be $\mathcal{C}_t(R_m)$ and $\mathbb{S}_G = \mathbb{S}_G(t)$. (Choose the puncturing pattern with the largest puncturing score.)
   e) $t = t + 1$. □

D. Numerical Results

Table II shows the gap from the AWGN channel capacity compared for our RC families constructed from the ARCA code of rate 0.5 for payloads of 1K and 4K. Random puncturing (R) and PNP (P) result in families within 0.5 dB and 0.8 dB from the channel capacity respectively. Fig. 6 shows the FER of these RC families on an AWGN channel for a payload of 1K. The codes constructed by random puncturing outperform those by PNP at low SNRs due to their lower thresholds. However, PNP results in codes with lower error floors especially for high rates. Using the techniques in Sec. II , we designed two codes of rate $1/6$, $\mathcal{C}_1$ which has AWGN and erasure thresholds of -0.54 dB and 0.82 respectively, and $\mathcal{C}_2$ which has AWGN and erasure thresholds of -0.7 dB and 0.814 respectively. The thresholds of the RC families, designed with $\mathcal{C}_1$, as the parent code are shown in Table III. It is worth noting that although $\mathcal{C}_2$ has a better AWGN threshold,

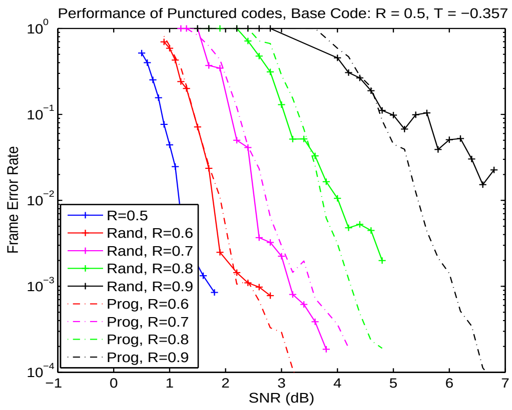

Fig. 6. FER of 1K Payload RC Families with PNP and Random Puncturing.

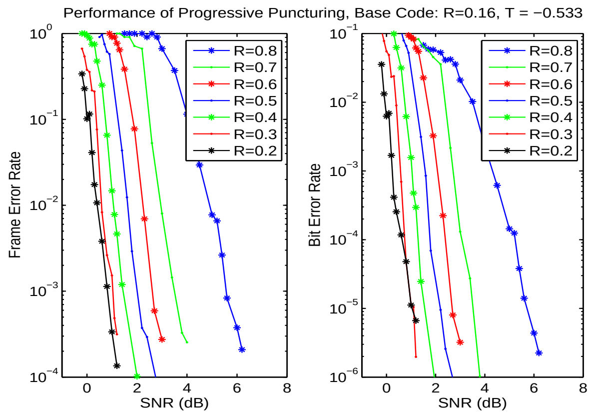

Fig. 7. FER and BER of 1K Payload RC Family with Progressive Puncturing

Rate	Capacity	Gap R 1k	Gap P 1k	Rate	Capacity	Gap R 1k	Gap P 1k
0.167	-1.073	0.539	0.539	0.50	0.185	0.318	0.560
0.20	-0.963	0.580	0.402	0.55	0.424	0.484	0.551
0.25	-0.793	0.375	0.375	0.60	0.679	0.876	0.523
0.30	-0.618	0.319	0.406	0.65	0.960	1.208	0.529
0.35	-0.432	0.311	0.452	0.70	1.270	1.586	0.480
0.40	-0.239	0.312	0.502	0.75	1.622	---	0.478
0.45	-0.032	0.311	0.537	0.80	2.039	2.392	0.861

TABLE III

1K PAYLOAD RC FAMILIES WITH RANDOM (R) AND PROGRESSIVE (P) PUNCTURING. PARENT RATE IS $1/6$.

its family has a poorer performance as it has a worse erasure threshold. As we can see from Fig. 7, for a payload of 1K, the PNP RC family has a good BER and FER performance on the AWGN channel and has a good error floor performance.

IV. CONCLUSIONS

Designing H-ARQ RC LDPC families is three fold:

First: We devised low complexity systematic techniques for designing low rate structured LDPC codes from higher rate ones. The designed codes should have a good performance on both the erasure and AWGN channels. These codes are also suitable for high throughput applications.

Second: Puncturing algorithms were devised to obtain higher rate codes, in the RC family. from such low rate codes. We point out here some general observations on the puncturing algorithms; *Structured puncturing* (puncturing on the protograph level) will not in general work. *Regular-Irregular puncturing* on the lifted graph is of low complexity and will often result in codes with good thresholds. However, one has to cautiously choose the set of designed of rates of the RC family. The *Random Puncturing Search* greedily finds the best (random) puncturing pattern for each code in the family based on the lower rate codes. The low rate codes in the family will have good thresholds. *Progressive Node Puncturing* carefully assigns the puncturing pattern such that further puncturing would result in good higher rate codes. The designed codes will have good thresholds over the whole family range as well as good error floors.

Third: Obtaining lower rate codes from the parent codes by extending or concatenating them with LDGM codes.

ACKNOWLEDGMENT

Thanks to Henry Pfister for providing a program for the progressive edge growth algorithm.

REFERENCES

[1] S. Sesia, G. Caire, and G. Vivier, “Incremental redundancy hybrid ARQ schemes based on low-density parity-check codes,” *IEEE Trans. Commun.*, vol. 52, pp. 1311 – 1321, Aug. 2004.
[2] T. Richardson and V. Novichkov, “Methods and apparatus for decoding LDPC codes,” *United States patent 6,633,856*, Oct 2003.
[3] J. Thorpe, “Low-density parity-check (LDPC) codes constructed from protographs,” *IPN Progress Report 42-154, JPL*, Aug 2003.
[4] D. Divsalar, S. Dolinar, and C. Jones, “Low-rate LDPC codes with simple protograph structure,” in *Proc. ISIT*, 2005, pp. 1627–1631.
[5] D. Divsalar, S. Dolinar, J. Thorpe, and C. Jones, “Constructing LDPC codes from simple loop-free encoding modules,” in *2005 IEEE International Conference on Communications*, pp. 658 – 662.
[6] S. Dolinar, “A rate-compatible family of protograph-based LDPC codes built by expurgation and lengthening,” in *Proc. ISIT*, 2005, pp. 1622–1626.
[7] J. Ha, J. Kim, and S. McLaughlin, “Rate-compatible puncturing of low-density parity-check codes,” *IEEE Trans. Inform. Theory*, vol. 50, no. 11, pp. 2824–2836, 2004.
[8] H. Pishro-Nik and F. Fekri, “Results on punctured LDPC codes,” in *2004 IEEE Information Theory Workshop, San Antonio, Texas*, October 2004.
[9] J. Li and K. R. Narayanan, “Rate-compatible low density parity check (RC-LDPC) codes for capacity-approaching ARQ schemes in packet data communications,” in *Proceeding of International Conference on Communications, Internet and Information Technology (CIIT), US Virgin Islands*, Nov. 2002, pp. 201–206.
[10] T. Tian and C. R. Jones, “Construction of rate-compatible LDPC codes utilizing information shortening and parity puncturing,” *EURASIP JWCN 2005:5 (2005)*, pp. 789–795.
[11] T. J. Richardson and R. Urbanke, “Multi-edge type LDPC codes,” preprint. [Online]. Available: http://lthcwww.ep.ch/papers/multiedge.ps.
[12] A. Shokrollahi, “Raptor codes,” in *Proceedings 2004 IEEE International Symposium on Information Theory (ISIT2004), Chicago, IL, USA*, 2004.
[13] X.-Y. Hu, E. Eleftheriou, and D. Arnold, “Regular and irregular progressive edge-growth Tanner graphs,” *IEEE Trans. Inform. Theory*, pp. 386 – 398, Jan. 2005.
[14] H. Xiao and A. Banihashemi, “Improved progressive-edge-growth (PEG) construction of irregular LDPC codes,” *IEEE Communications Letters*, pp. 715 – 717, Dec. 2004.
[15] S.-Y. Chung, T. Richardson, and R. Urbanke, “Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation,” *IEEE Trans. Inform. Theory*, pp. 657–670, Feb. 2001.