An Incremental Redundancy Hybrid ARQ Scheme via Puncturing and Extending of Polar Codes

Hamid Saber, *Student Member, IEEE,* and Ian Marsland, *Member, IEEE*

***Abstract*—We construct polar codes for the specific purpose of incremental redundancy hybrid automatic repeat request (IR-HARQ) schemes. The rate compatibility of our scheme is ensured by both puncturing and extending of the code. A new puncturing algorithm for polar codes is proposed, and we develop an algorithm for finding good extending sequences for polar codes from any arbitrary punctured rate, with the goal of improving the throughput as much as possible. Simulation results for different types of puncturing and extending algorithms are presented. We show how the proposed extending algorithm, when properly operated with a good puncturing algorithm and a well-chosen puncturing rate, yields IR-HARQ coding schemes which can operate within 1 dB of Shannon capacity over a very wide range of signal-to-noise ratios.**

***Index Terms*—Incremental redundancy hybrid ARQ, rate-compatible polar codes, channel polarization.**

I. Introduction

TRANSMISSION schemes for wireless communication should provide flexible transmission rates to compensate for channel state variations. One approach is to have several pairs of encoders and decoders with different fixed rates to adapt to the channel state. However, this requires extra hardware complexity. As an alternative, rate compatible (RC) codes have been proposed that can be implemented with a single encoder/decoder pair, and have become an important tool to address the rate flexibility requirement in communication systems. RC codes are also well-suited for a specific type of hybrid automatic repeat request (ARQ) scheme called incremental redundancy hybrid ARQ (IR-HARQ). In an IR-HARQ scheme, the transmitter keeps sending additional code bits of a mother code to the receiver until a decoding success is announced. There has been a lot of research on code construction for IR-HARQ. In particular RC convolutional and turbo codes were among the first codes proposed for use with IR-HARQ [1]-[3], where puncturing is used to ensure the rate-compatibility. Later, low density parity check (LDPC) codes were studied for IR-HARQ schemes. In particular both asymptotic and finite length puncturing schemes were proposed to produce RC-LDPC codes [4],[5]. However, puncturing by itself is not sufficient to realize rate-compatible codes that are effective over a wide range of rates. In response the authors in [6] proposed that LDPC codes can be extended by extending their parity check matrices, and puncturing and extending became useful tools to design RC-LDPC codes for IR-HARQ [7].

Recently proposed by Arikan [8], polar codes are the first class of structured channel codes which can provably achieve the capacity of binary input memoryless output symmetric (BIMOS) channels under successive cancellation (SC) decoding. Although the capacity achieving property of polar codes is an asymptotic attribute, their performance and code design for finite lengths have been studied as well (e.g. [9], [10]). Motivated by the theoretical fixed-rate capacity-achieving property, it is natural to apply polar codes to IR-HARQ. Puncturing schemes to produce RC polar codes are proposed in [11], [12]. The authors in [13] used these methods and designed a IR-HARQ scheme based on polar codes. Their proposed scheme uses puncturing and a selective repetition of the message bits. The reported results suggest that despite the non-promising fixed rate performance of finite-length polar codes, the throughput performance can remain close to the Shannon capacity, when used with IR-HARQ.

In this paper we focus on designing polar codes for IR-HARQ schemes. Our proposed scheme uses both puncturing and extending. We propose a novel puncturing algorithm and we develop an algorithm to extend the polar codes in such a way that its goal is to maximize the throughput of the system. We extend the polar code by selecting code bits chosen from a finite set of linear combinations of message bits; each combination corresponds to each node in the PC graph (to be defined shortly), while [13] allows only a subset of this set, i.e. the message bits. We also show how the proposed extending can be applied to a polar code which is punctured to an arbitrary rate.

The rest of the paper is organized as follows. In Section II we present preliminaries of polar codes, their encoding and decoding, and code design. We present our heuristic algorithm for puncturing polar codes in Section III. In Section IV we present an algorithm for extending polar codes for IR-HARQ schemes. Section V presents the simulation results for different types of puncturing and extending methods. It also compares the best IR-HARQ code with the previous IR-HARQ schemes based on polar codes. Conclusions and topics for future research are provided in Section VI.

II. Polar codes

A polar code with length $N = 2^n$ is a linear block code whose generator matrix is $\mathbf{G}_N = \mathbf{F}^{\otimes n}$, where $\mathbf{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ and $\otimes$ denotes the Kronecker product.[1] For example, when $N = 8$,

[1] In [8] the generator matrix of the polar code is defined as $\mathbf{G}_N = \mathbf{B}_N\mathbf{F}^{\otimes n}$, where $\mathbf{B}_N$ is a permutation matrix known as the *bit reversal* permutation matrix. However $\mathbf{B}_N$ is just a permutation matrix, and thus, for simplicity we can assume that the message vector is already permutated and the generator matrix can be considered as $\mathbf{G}_N = \mathbf{F}^{\otimes n}$. This representation of the generator matrix is also useful in developing the extending algorithms using the PC graph.

$$\mathbf{G}_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \quad (1)$$

Let $\mathcal{A}$ be a set of cardinality K containing a subset of the row indices of $\mathbf{G}_N$. We refer to this set as the *information* set. The complementary set $\mathcal{A}^c$ is referred to as the *frozen* set. Let $\mathbf{u} = [u_1, \ldots, u_N]$ be a binary vector of size $1 \times N$. The K information bits are placed in those elements of $\mathbf{u}$ corresponding to the set $\mathcal{A}$, and deterministic values (typically zeros) are placed in the other $N - K$ elements. The codeword corresponding to an information vector $u_{\mathcal{A}}$ is then calculated as

$$\mathbf{c} = \mathbf{u}\mathbf{G}_N. \quad (2)$$

We consider a BIMOS channel $\mathcal{W}$ with binary input alphabet, output alphabet $\mathcal{Y}$ and the transition probabilities $f_W(y|0)$ and $f_W(y|1)$ for $y \in \mathcal{Y}$. The codeword $\mathbf{c}$ is transmitted over $\mathcal{W}$ and the channel output vector $\mathbf{y} = [y_1, \ldots, y_N]$ is received. Note that the term "polar code" refers to a specific instance of the set $\mathcal{A}$, chosen according to the *polar rule* which chooses the indices of the bit-channels (suitably defined in [8]) with the largest capacities (for more details see [8]). A polar code is completely described by $(N, K, \mathcal{A})$. After describing the encoding and decoding algorithms for polar codes in Section II.A, a method for determining $\mathcal{A}$ is presented in Section II.B.

A. Encoding and Decoding

Encoding and decoding of polar codes is most usefully described in terms of a graphical representation of the generator matrix, using what we refer to as the polar code (PC) graph. Fig. 1 depicts the PC graph for the polar code of length $N = 8$, corresponding to $\mathbf{G}_8$ given in (1). We represent each node in the PC graph of polar code of length $N = 2^n$ by (i, j), where $i \in \{1, \ldots, N\}$ and $j \in \{0, \ldots, n\}$ are the row and column indices of the node in the graph. The rows are numbered from top to bottom, and the columns are numbered from left to right. Numbering of the columns starts with zero rather than one for simplicity in describing the extending algorithm in Section IV. Starting with labeling the nodes in column 0 by the message bits $u_1, \ldots, u_N$, and the nodes in column n by the code bits $c_1, \ldots, c_N$, each node in the graph corresponds to a linear combination of the message bits. Let $v_{i,j}$ denote the value of this combination at node (i, j), with $v_{i,0} = u_i$ and $v_{i,n} = c_i$.

Encoding is carried out on the PC graph by applying the message bits to the nodes in the leftmost column, so $v_{i,0} = u_i$. These bits are then sent to the next column. Calculation of the bit values are calculated according to the butterfly shapes in the graph. In each butterfly the right-hand top node is equal to the summation of the bits in the two nodes on the left-hand side, and the right-hand bottom node is equal to the left-hand bottom node. The bit values are propagated to the right until the code bits $c_i = v_{i,n}$ are calculated. Note that during the encoding process no values are passed along the edges that slope down to the right on the PC graph (indicated by dashed lines in Fig. 1).

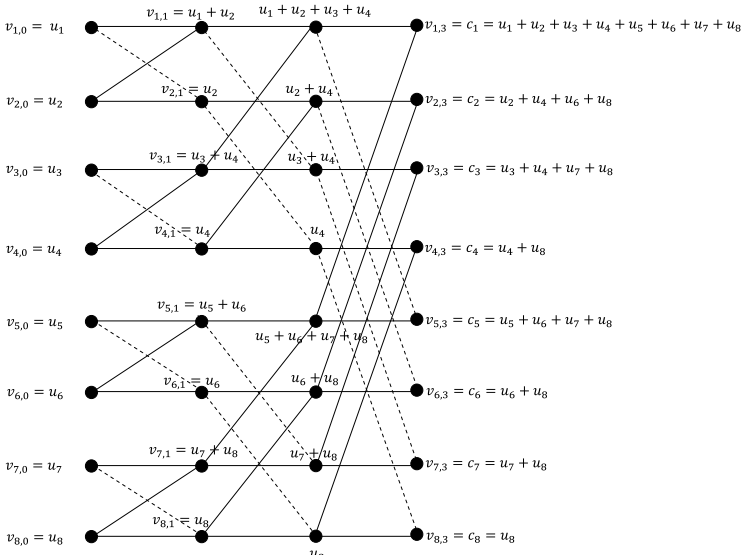

Fig. 1. The PC graph of a polar code with length $N = 8$.

Arikan proposed a SC decoding algorithm for polar codes [8] that can be described by the PC graph. The channel log-likelihood ratios (LLRs), $\lambda(y_i) = \ln\frac{f_W(y_i|0)}{f_W(y_i|1)}$ for code bit $c_i, i = 1, \ldots, N$, are calculated and applied to the rightmost column, so $\lambda_{i,n} = \lambda(y_i)$. These LLRs propagate leftwards through the graph. We say that a node (i, j), $j < n$, is of type I if it is in the upper-left corner of a butterfly, and type II if it is in the lower-left corner. Thus

$$\text{type}(i, j) = \begin{cases} \text{I} & \text{if } \left\lfloor \frac{i-1}{2^j} \right\rfloor \equiv 0 \bmod 2 \\ \text{II} & \text{if } \left\lfloor \frac{i-1}{2^j} \right\rfloor \equiv 1 \bmod 2 \end{cases} \quad (3)$$

where $\lfloor x \rfloor$ is the largest integer not greater than x. The LLR of the type-I node (i, j) is calculated according to

$$\begin{aligned} \lambda_{i,j} &= \ln \frac{e^{\lambda_{i,j+1}} e^{\lambda_{i',j+1}} + 1}{e^{\lambda_{i,j+1}} + e^{\lambda_{i',j+1}}} \\ &= 2\tanh^{-1}\left(\tanh\left(\frac{\lambda_{i,j+1}}{2}\right)\tanh\left(\frac{\lambda_{i',j+1}}{2}\right)\right) \end{aligned} \quad (4)$$

where i' is the row at the lower edge of the butterfly containing (i, j) and $(i, j+1)$ as the upper edge, and the LLR of the type-II node (i', j) is given by

$$\lambda_{i',j} = \lambda_{i',j+1} + (1 - 2\hat{v}_{i,j})\lambda_{i,j+1} \quad (5)$$

where $\hat{v}_{i,j}$ is a hard estimate of $v_{i,j}$. To calculate $\hat{v}_{i,j}$ it is necessary to process nodes in a specific order when decoding. In particular, LLRs are only passed upward to the left at first, until $\lambda_{1,0}$ is calculated, and a hard decision for $\hat{v}_{1,0} = \hat{u}_1$ is made using $\lambda_{1,0}$ ($\hat{v}_{1,0}$ is 0 if $\lambda_{1,0} > 0$ and 1 otherwise). Only then can $\lambda_{2,0}$ be calculated by using (5) along with $\hat{v}_{1,0}$. Then

$\hat{v}_{1,0}$ and $\hat{v}_{2,0}$ can be passed back through the graph so that $\hat{v}_{1,1}$ and $\hat{v}_{2,1}$ can be determined using the same procedure as the encoder. With $\hat{v}_{1,1}$ and $\hat{v}_{2,1}$ known, the decoder can use (5) to calculate $\lambda_{3,1}$ and $\lambda_{4,1}$, allowing $\hat{v}_{3,0}$ and $\hat{v}_{4,0}$ to be eventually calculated. By continuing this procedure the message bits can be successfully decoded in the order of $\hat{u}_1$, $\hat{u}_2, \ldots$, and, under the assumption that correct decisions are made, their effects can be cancelled, enabling reliable estimation of as-of-yet undecoded message bits. Note that since the values of the frozen bits, u_i $i \in \mathcal{A}^c$, are deterministic and known to the receiver, the decoder simply chooses $\hat{v}_{i,0} = \hat{u}_i = u_i$ for $i \in \mathcal{A}^c$.

B. Code Design

To design the $(N, K, \mathcal{A})$ polar code it is necessary to find the set of indices for the information bits, $\mathcal{A}$. This can be done using the algorithm presented in [10], which is based on density evolution, assuming that the all-zero codeword is transmitted. Every node (i, j) in the graph is assigned a probability density denoted by $f_{i,j}(\lambda)$ which is the density of $\lambda_{i,j}$, the LLR calculated at that node. All nodes in the rightmost column are assigned the channel LLR density $f_W^{(0)}(\lambda)$, i.e. $f_{i,n}(\lambda) = f_W^{(0)}(\lambda)$ for $i = 1, \ldots, N$. The density $f_W^{(0)}(\lambda)$ is defined as follows. For a channel with transition probabilities $f_W(y|0)$ and $f_W(y|1)$, define $\lambda(y) = \ln\frac{f_W(y|0)}{f_W(y|1)}$. The channel LLR density, $f_W^{(0)}(\lambda)$, is defined as the probability density function of $\lambda(y)$, when y has the density $f_W(y|0)$. For a binary input AWGN channel with antipodal signaling $\{\pm 1\}$ and noise variance σ^2, we have $\lambda(y) = 2y/\sigma^2$ and

$$f_W^{(0)}(\lambda) = \frac{1}{\sqrt{8\pi/\sigma^2}} \exp\left(-\frac{(\lambda - 2/\sigma^2)^2}{8/\sigma^2}\right). \tag{6}$$

The algorithm calculates the densities at all the other nodes, from right to left, according to

$$f_{i,j}(\lambda) = \begin{cases} f_{i,j+1}(\lambda) \boxplus f_{i',j+1}(\lambda) & \text{if type}(i,j) = \text{I} \\ f_{i,j+1}(\lambda) \circledast f_{i',j+1}(\lambda) & \text{if type}(i,j) = \text{II} \end{cases} \tag{7}$$

where $(i, j+1)$ and $(i', j+1)$ are the two right neighbor nodes of (i, j), $\circledast$ denotes the convolution of the two densities, and $\boxplus$ denotes the operator for calculation of the density at the output of a check node [10]. In particular, if X and Y are two random variables with densities $f_X(x)$ and $f_Y(y)$, and $Z = 2\tanh^{-1}\left(\tanh\frac{X}{2}\tanh\frac{Y}{2}\right)$, then the density of Z is written as $f_Z(z) = f_X(x) \boxplus f_Y(y)$. Each density calculated in the leftmost column, $f_{i,0}(\lambda)$ is the density of the LLR of bit u_i, $\lambda_{i,0}$, under the assumption that the all-zero codeword was transmitted and that $u_1, \ldots, u_{i-1}$ were decoded correctly.

The *first-error* event probability, which is the probability that u_i is the first bit to be incorrectly decoded given that all previously decoded bits are correct, is defined as

$$\begin{aligned} P_e(i) &:= \Pr(\hat{u}_i \neq u_i | \hat{u}_1 = u_1, \ldots, \hat{u}_{i-1} = u_{i-1}) \\ &= \int_{-\infty}^{0^-} f_{i,0}(\lambda) d\lambda + \frac{1}{2}\int_{0^-}^{0^+} f_{i,0}(\lambda) d\lambda \end{aligned} \tag{8}$$

These error probabilities can be sorted in ascending order, and the indices of the K bits with the smallest error probabilities can be used for $\mathcal{A}$.

As an aside, it is worth noting that a rate-compatible polar code can be found by fixing N and varying K, with $\mathcal{A}_K$ the information set for rate K/N, chosen as above. In this case $\mathcal{A}_K$ will contain all elements of $\mathcal{A}_{K-1}$ plus one additional index. The transmitter and receiver only need to know the order of the indices as determined above to produce codes with any rate that is an integer multiple of $1/N$. However, this type of RC polar code is not useful for IR-HARQ schemes, where K is normally fixed and N varied. In this case puncturing and extending can be used to produce RC polar codes.

III. Puncturing algorithms

By transmitting only a subset of the code bits of a rate K/N mother code, puncturing allows for the creation of higher-rate codes. Decoding of punctured polar codes can be accomplished by applying LLR values of zero to the nodes in the rightmost column of the PC graph that correspond to the punctured code bits. The more difficult challenge is determining which bits to puncture to least degrade the performance of the code.

In [11] the authors proposed a stopping tree puncturing algorithm to determine which code bits to puncture to achieve a desired rate. Code bits which depend on the least number of message bits are punctured first. This algorithm is designed for use with a belief propagation (BP) decoder [14], so may not be applicable to SC decoding. The authors in [12] proposed a quasi-uniform puncturing (QUP) algorithm which punctures the code bits in such a way that it tries to keep the distance between any two adjacent punctured code bits as uniform as possible. QUP is shown to possess good properties reflecting the minimum Hamming distance of the punctured code. Neither of these heuristic schemes are necessarily useful for SC decoding in IR-HARQ systems.

In the context of IR-HARQ, puncturing is better framed as determining in which order to transmit the code bits, with the idea that more "important" code bits should be transmitted first, to increase the likelihood of early successful decoding. However, there are no precise metrics for ranking the relative importance of the code bits that are computationally feasible, so instead we propose another heuristic algorithm. We note that since the generator matrices of the polar codes are involutory, the code bits and message bits are calculated from each other with the same type of linear combination. For example c_1 is the summation of all message bits while u_1 is the summation of all code bits, $c_2 = u_2 + u_4 + u_6 + u_8$ while $u_2 = c_2 + c_4 + c_6 + c_8$, and so on. So it seems that there could be a similarity between the reliability of any message bit u_i and the importance of the corresponding code bit with the same index c_i. We therefore conjecture without proof that, just as more reliable message bits are those with indices with smallest first-error probabilities, more important code bits might be those with the same indices. As a result we propose to transmit the code bits in the same order in which the message bit first-error probabilities increase. So if

u_i has smaller first-error probability than u_j then code bit c_i is transmitted before code bit c_j. Note that the problem of designing an optimal puncturing algorithm for finite length polar codes still remains open. The proposed method, as we see in Section V, is nonetheless effective when combined with the extending algorithm, regardless of whether or not the above conjecture could be proved, or is even relevant.

Implementation of this algorithm is straightforward since, as part of the code design process for constructing polar codes using density evolution [10], the first-error event probability of each message bit has already been calculated according to (8). Whereas for code construction we were only interested in selecting the K bits with the lowest probabilities, here we are interested in the order of all N bits, ordered from lowest to highest probability. Let $P_e(i)$ be the first-error probability of the message bit u_i. We make an ordered set $\Omega = \{\omega_1, \ldots, \omega_N\}$ from $\{1, 2, \ldots, N\}$ in such a way that

$$P_e(\omega_i) \leq P_e(\omega_{i+1}). \tag{9}$$

The code bits are then transmitted in the order of $c_{\omega_1}, c_{\omega_2}, c_{\omega_3}, \ldots$.[2] As an example, suppose we have a polar code of length $N = 8$ and we wish to only transmit four code bits. Our algorithm would transmit code bits c_8, c_7, c_6 and c_4, as their corresponding message bits have the smallest first-error probabilities, whereas QUP would transmit c_8, c_4, c_6 and c_2, and stopping tree puncturing would transmit c_1, c_2, c_3 and c_5.

IV. Extending algorithms

Whereas puncturing is a useful tool for constructing higher-rate codes from a lower-rate mother code, extending allows us to generate lower-rate codes from a higher-rate mother code. For linear block codes this is typically achieved by either retransmitting previously transmitted code bits or by adding additional columns to the generator matrix to allow for the creation of new code bits with different parity check equations. Generally speaking, generation of new parity bits gives a more powerful extended code than relying on retransmissions, but comes at the expense of higher decoding complexity. In particular, adding arbitrary columns to the generator matrix would require adding additional nodes and edges to the PC graph. With an increase to the number of edges connected to a node would come an increase of the complexity of the operations that need to be carried out at the nodes over the relative simplicity of (4) and (5). Furthermore, the repetitive symmetry of the butterfly shapes in the PC graph would also be lost, leading to higher decoding and scheduling complexity.

In [13] the authors proposed a simple extending scheme for polar codes that involves selective transmission and retransmission of the code bits along with the transmission of the uncoded message bits. This simple scheme does not require any significant decoding complexity since decoding is performed on the unaltered PC graph of the mother polar code. The LLR for a retransmitted code bit is added to the LLRs from previous transmissions of that code bit prior to decoding, and the LLR for a transmitted message bit is added to the corresponding LLR produced by the decoder immediately prior to making a hard decision on that bit. This simple scheme is quite effective at high code rates, but performance suffers with low rate extended codes.

In this section we propose a new extending algorithm for polar codes that, like the method in [13], allows for decoding on the unaltered PC graph, while also providing good codes over a wide range of rates. Motivating this technique is the observation that every node in the PC graph corresponds to a linear combination of the message bits and could therefore be considered as a code bit for a linear block code. We therefore propose to extend the polar codes by transmitting the bit values ($v_{i,j}$) associated with an appropriately selected subset of all of the nodes in the graph, not just those associated with the message bits or the code bits of the polar code. By sending only these bits (as opposed to arbitrary linear combinations of the message bits), SC decoding is still straightforward. Each node (i, j) in the graph must be modified to accept a channel LLR corresponding to transmission of $v_{i,j}$, and this LLR should be added to the LLR computed according to (4) or (5) before passing it to the neighbors to the left. If $v_{i,j}$ has not been transmitted then a channel LLR of 0 should be used. With this technique it is possible to produce extended polar codes with nearly any rate from K/N down to $K/(K + N\log_2 N)$ without having to resort to retransmitting bits.

The problem of designing a good extended polar code of rate K/N' that can still be decoded with the low-complexity SC decoder, then, becomes one of choosing which N' out of the $N(1 + \log_2 N)$ potential "code bits" corresponding to all of the nodes in the graph, to transmit to achieve the best performance. For an IR-HARQ scheme, the problem is better formulated as deciding which order to transmit these code bits in so that the throughput is maximized. We refer to this order as the *extending sequence*.

A. Proposed Algorithm

Suppose we have a mother code of rate $R_M = K/N_M$, $N_M = 2^n$, that has been punctured to rate $R_I = K/N_I \geq R_M$. Equivalently, suppose we have a mother code of length N_M and we have transmitted a subset of N_I of its code bits. We then wish to transmit additional bits until decoding succeeds. The additional bits may be, but are not necessarily, code bits of the mother code, but must correspond to nodes of the PC graph (from any column). The extending sequence $C = \{(\omega_1, d_1), (\omega_2, d_2), (\omega_3, d_3), \ldots\}$ defines the order in which the additional bits should be transmitted. That is v_{ω_1,d_1} should be transmitted first, followed by v_{ω_2,d_2}, and so on.

As our goal is to produce codes suitable for incremental redundancy schemes, we want to choose the extending sequence that maximizes the throughput. The throughput, T, is defined as $T = \frac{K}{\overline{N}}$, where K is the length of the message word and $\overline{N}$ is the average number of code bit transmissions for successful decoding. If we let $\mathcal{E}_i$ denote the event that the decoding fails

[2]It is important to note that, if we were designing a fixed-rate code, it would be prudent to re-select the information set, $\mathcal{A}$, once the puncturing pattern has been determined, but that is not possible if rate-compatibility is required, so is not done here. The originally selected information set of the mother code remains fixed.

after i code bits (from any column of the PC graph) have been transmitted, then

$$\begin{aligned}\overline{N} &= \sum_{m=K}^{\infty} m \Pr\left(\left[\bigcap_{i=K}^{m-1} \mathcal{E}_i\right] \cap \mathcal{E}_m^c\right) \\ &= \sum_{m=K}^{\infty} m\left[\Pr\left(\bigcap_{i=K}^{m-1} \mathcal{E}_i\right) - \Pr\left(\bigcap_{i=K}^{m} \mathcal{E}_i\right)\right]. \end{aligned} \tag{10}$$

To increase throughput, we need to decrease $\overline{N}$. Defining $\alpha_m = \Pr\left(\bigcap_{i=K}^{m-1} \mathcal{E}_i\right) - \Pr\left(\bigcap_{i=K}^{m} \mathcal{E}_i\right)$, we would like to choose the extending nodes so that $\sum_{m=K}^{\infty} m\alpha_m$ is minimized. However, for an extending sequence of length N_E there are $\left(K + N_M \log_2 N_M\right)^{N_E}$ possible different ways to choose the extending sequence and this grows exponentially with its length, so this optimization problem is hard to solve. Furthermore, even if we could find the optimized extending sequence that maximized the throughput at one signal-to-noise ratio (SNR), it may not be rate compatible with the optimized sequence at another SNR.

We therefore propose a greedy algorithm for extending the code by one bit at a time. Suppose that we have transmitted exactly $m-1$ code bits corresponding to $m-1$ nodes in the PC graph. We want to know which one node to transmit next to help the throughput the most. Given all previous $m-1$ transmitted bits, all α_i $i = K, \ldots, m-1$ are determined. We would like to choose the next extending node in such a way that it minimizes $\overline{N} = \sum_{i=K}^{\infty} i\alpha_i$. Since α_i are fixed for $i < m$, this is equivalent to minimizing $\sum_{i=m}^{\infty} i\alpha_i$ subject to $\sum_{i=m}^{\infty} \alpha_i = 1 - \sum_{i=K}^{m-1} \alpha_i$. Observing that

$$\begin{aligned}\sum_{i=m}^{\infty} i\alpha_i &\geq \sum_{i=m}^{\infty} m\alpha_i \\ &= m \sum_{i=m}^{\infty} \alpha_i = m(1 - \sum_{i=K}^{m-1} \alpha_i), \end{aligned} \tag{11}$$

this lower bound can be reached by choosing $\alpha_i = 0$ $\forall i > m$, and $\alpha_m = 1 - \sum_{i=K}^{m-1} \alpha_i$. Since

$$1 - \sum_{i=K}^{m-1} \alpha_i = \Pr\left(\bigcap_{i=K}^{m-1} \mathcal{E}_i\right) \tag{12}$$

and

$$\alpha_m = \Pr\left(\bigcap_{i=K}^{m-1} \mathcal{E}_i\right) - \Pr\left(\bigcap_{i=K}^{m} \mathcal{E}_i\right), \tag{13}$$

this implies the next node should be chosen in such a way that it makes $\Pr\left(\bigcap_{i=K}^{m} \mathcal{E}_i\right)$ zero. Since we cannot make this probability zero in practice, we try to minimize it by minimizing $\Pr\left(\bigcap_{i=K}^{m} \mathcal{E}_i\right)$.

For a general linear block code there is no relationship between the events $\mathcal{E}_i$ and $\mathcal{E}_{i-1}$. In other words, mathematically it is possible that decoding the code of length $i-1$ succeeds and that of length i does not, and vice versa. However, in an IR-HARQ scheme, where the decoder decodes after it has received each code bit, the decoder does not start decoding the code of length i, unless it has already attempted decoding the code of length $i-1$ and has failed. So for the decoder of an IR-HARQ scheme, we have $\mathcal{E}_i \subset \mathcal{E}_{i-1}$, which implies

$$\Pr\left(\bigcap_{i=K}^{m} \mathcal{E}_i\right) = \Pr\left(\mathcal{E}_m\right). \tag{14}$$

Therefore, minimizing $\Pr\left(\bigcap_{i=K}^{m} \mathcal{E}_i\right)$ can be done by minimizing $\Pr\left(\mathcal{E}_m\right)$, which is simply the block error rate (BLER) after m code bit transmissions.[3] So we propose to find the extending node in such a way that it reduces the BLER of the extended polar code as much as possible. The BLER of a polar code can be estimated by $1 - \prod_i (1 - P_e(i))$, where $P_e(i)$ is the first-error probability of bit u_i, corresponding to the LLR calculated at the node $(i, 0)$, and is given by (8). This implies that we should choose the extending node such that it will minimize $1 - \prod_i (1 - P_e(i))$.

Density evolution is used to evaluate the first-error probabilities, using a technique similar to the approach used for polar code design as described in Section II. Associated with each node (i, j) in the PC graph are two densities: the channel LLR density, $f_{i,j}^{(0)}(\lambda)$, and the calculated LLR density, $f_{i,j}(\lambda)$. The channel LLR densities are

$$f_{i,j}^{(0)}(\lambda) = \begin{cases} f_W^{(0)}(\lambda) & \text{if bit } v_{i,j} \text{ has been transmitted} \\ \delta(\lambda) & \text{otherwise} \end{cases} \tag{15}$$

where $f_W^{(0)}(\lambda)$ is given by (6). The calculated LLR densities in the rightmost column are initialized with $f_{i,n}(\lambda) = f_{i,n}^{(0)}(\lambda)$, and then the other densities are calculated from right to left using (7), except that $f_{i,j}(\lambda)$ is also convolved with $f_{i,j}^{(0)}(\lambda)$. Once $f_{i,0}(\lambda)$ has been calculated, $P_e(i)$ can be calculated according to (8), except for frozen message bits, where $P_e(i) = 0$ $\forall i \in \mathcal{A}^c$.

To find the best extending node we could perform an exhaustive search. Each node in the graph can be considered as candidate for the best node, so we consider each node one at a time. For each candidate node we calculate the LLR densities of the message bits, $f_{i,0}(\lambda)$, the first-error probabilities, and the metric $1 - \prod_i (1 - P_e(i))$ which is used to represent the BLER under the hypothesis that the candidate node is transmitted next. The candidate node with the smallest metric is chosen as the next extending node.

Because the complexity of this extending search is quite high, it is necessary to reduce the complexity. To this end, we define the depth-l left-neighborhood graph of node (i, j), denoted by $\mathcal{N}_{i,j}^{\overleftarrow{l}}$, as the subgraph of the PC graph consisting of all nodes and edges that can be reached from (i, j) by walking at most l edges only to the left. A similar definition holds for the depth-l right-neighborhood graph of a node, denoted by $\mathcal{N}_{i,j}^{\overrightarrow{l}}$. For example $\mathcal{N}_{1,0}^{\overrightarrow{0}} = \{(1, 0)\}$, $\mathcal{N}_{1,0}^{\overrightarrow{2}} = \{(1, 0), (1, 1), (2, 1), (1, 2), (2, 2), (3, 2), (4, 2)\}$, $\mathcal{N}_{3,3}^{\overleftarrow{0}} = \{(3, 3)\}$, and $\mathcal{N}_{3,3}^{\overleftarrow{2}} = \{(3, 3), (3, 2), (7, 2), (1, 1), (3, 1), (5, 1), (7, 1)\}$.

[3] It is noteworthy that if we did not have an IR-HARQ decoding scheme, (14) would not hold mathematically. But we could still make use of it as $\Pr\left(\bigcap_{i=K}^{m} \mathcal{E}_i\right) \leq \Pr\left(\mathcal{E}_m\right)$, and minimizing $\Pr\left(\bigcap_{i=K}^{m} \mathcal{E}_i\right)$ can be turned into minimizing its upper bound, $\Pr\left(\mathcal{E}_m\right)$.

It is possible to reduce the complexity of calculating the metrics. Once the densities of the current extending sequence have been determined, to update the densities under the hypothesis that candidate node (ω, d) is transmitted next only requires updates to the densities in the depth-d left neighborhood graph of (ω, d). Thus instead of calculating all $N(1+\log_2 N)$ densities for each hypothesis, we only need to calculate $2^{d+1}-1$.

Algorithm 1: The proposed extending algorithm.

Input : Current densities: $f_{i,j}(\lambda)$ and $f_{i,j}^{(0)}(\lambda)$ $\forall 1 \le i \le N_M,\ 0 \le j \le n$

Output: The extending node (ω^*, d^*), updated densities $f_{i,j}(\lambda)$, $f_{i,j}^{(0)}(\lambda)$

1. Calculate error probabilities for message nodes

for $i \in \{1, \ldots, N_M\}$ **do**

$P_e(i) \leftarrow \int_{-\infty}^{0^-} f_{i,0}(\lambda)d\lambda + \frac{1}{2}\int_{0^-}^{0^+} f_{i,0}(\lambda)d\lambda$

2. Choose the target node to improve: $\tilde{i} = \text{argmax}_i P_e(i)$

3. Calculate the possible metrics for all nodes in $\mathcal{N}_{\tilde{i},0}^{\overrightarrow{L}}$.

for $(\omega, d) \in \mathcal{N}_{\tilde{i},0}^{\overrightarrow{L}}$ **do**

Calculate the possible metric, $\alpha_{\omega,d}$, if the node (ω, d) is extending:

$$\alpha_{\omega,d} \leftarrow \text{next\_metric}\left(\left\{f_{i,j}(\lambda), f_{i,j}^{(0)}(\lambda) | 1 \le i \le N_M, 0 \le j \le n\right\}, (\omega, d)\right) \tag{16}$$

4. Find the node with the minimum metric: $(\omega^*, d^*) \leftarrow \text{argmin}_{\omega,d}\alpha_{\omega,d}$

5. Update the densities in $\mathcal{N}_{\omega^*,d^*}^{\overleftarrow{d^*}}$:

$f_{\omega^*,d^*}(\lambda) \leftarrow f_{\omega^*,d^*}(\lambda) \circledast f_W^{(0)}(\lambda)$

$f_{\omega^*,d^*}^{(0)}(\lambda) \leftarrow f_{\omega^*,d^*}^{(0)}(\lambda) \circledast f_W^{(0)}(\lambda)$

for $l \in \{1, \ldots, d^*\}$ **do**

for $(i,j) \in \mathcal{N}_{\omega^*,d^*}^{\overleftarrow{l}} \backslash \mathcal{N}_{\omega^*,d^*}^{\overleftarrow{l-1}}$ **do**

Let $(i, j+1)$ and $(i', j+1)$ be the two right neighbors of (i, j)

$$f_{i,j}(\lambda) \leftarrow \begin{cases} f_{i,j+1}(\lambda) \boxplus f_{i',j+1}(\lambda) & \text{if type}(i,j) = \text{I} \\ f_{i,j+1}(\lambda) \circledast f_{i',j+1}(\lambda) & \text{if type}(i,j) = \text{II} \end{cases}$$

$f_{i,j}(\lambda) \leftarrow f_{i,j}(\lambda) \circledast f_{i,j}^{(0)}(\lambda)$

Even with this modification, however, the complexity of the exhaustive search remains prohibitively large because of the size of the search space, particularly for large N. We therefore propose a suboptimal yet effective search algorithm. The algorithm starts by calculating the first-error probability, $P_e(i)$, for each message node $(i, 0)$ based on the current densities, and selects the node with the largest probability. This node, referred to as the target node $(\tilde{i}, 0)$, contributes the most to the BLER of code. To reduce the search space, the algorithm only searches for the best extending node from within the depth-L right neighborhood graph of the target node (i.e. $\mathcal{N}_{\tilde{i},0}^{\overrightarrow{L}}$), where L is a parameter that limits the scope of the search. Since choosing the extending node from outside of $\mathcal{N}_{\tilde{i},0}^{\overrightarrow{L}}$ will not have any significant effect on $P_e(\tilde{i})$, we can reduce the search space to 2^{L+1} nodes instead of $K + N\log_2 N$.

Algorithm 2: The function next_metric.

Input : Current densities: $f_{i,j}(\lambda), f_{i,j}^{(0)}(\lambda), 1 \le i \le N_M, 0 \le j \le n$, the extending node (ω, d)

Output: The resultant metric: $1 - \prod_i (1 - P_e(i))$

1. Update the densities in $\mathcal{N}_{\omega,d}^{\overleftarrow{d}}$

$f_{\omega,d}(\lambda) \leftarrow f_{\omega,d}(\lambda) \circledast f_W^{(0)}(\lambda)$

for $l \in \{1, \ldots, d\}$ **do**

for $(i,j) \in \mathcal{N}_{\omega,d}^{\overleftarrow{l}} \backslash \mathcal{N}_{\omega,d}^{\overleftarrow{l-1}}$ **do**

Let $(i, j+1)$ and $(i', j+1)$ be the two right neighbors of (i, j)

$$f_{i,j}(\lambda) \leftarrow \begin{cases} f_{i,j+1}(\lambda) \boxplus f_{i',j+1}(\lambda) & \text{if type}(i,j) = \text{I} \\ f_{i,j+1}(\lambda) \circledast f_{i',j+1}(\lambda) & \text{if type}(i,j) = \text{II} \end{cases}$$

$f_{i,j}(\lambda) \leftarrow f_{i,j}(\lambda) \circledast f_{i,j}^{(0)}(\lambda)$

2. Calculate first-error probabilities for message nodes

for $i \in \{1, \ldots, N_M\}$ **do**

$P_e(i) \leftarrow \int_{-\infty}^{0^-} f_{i,0}(\lambda)d\lambda + \frac{1}{2}\int_{0^-}^{0^+} f_{i,0}(\lambda)d\lambda$

3. Return $1 - \prod_i (1 - P_e(i))$

In summary, the extending sequence is created one node at a time, by searching for the node that would most reduce the BLER of the code. Given an extending sequence of length $m-1$, and the LLR densities $f_{i,j}^{(0)}(\lambda)$ and $f_{i,j}(\lambda)$ corresponding to that sequence, algorithm 1 finds the best node to transmit next, yielding an extending sequence of length m along with the associated updated LLR densities.

The function next_metric, takes the current densities at the nodes of the graph (corresponding to the as-of-yet constructed extended code) and a node to be considered as the extending node and returns the metric representing the BLER of the extended code for this extending node was used to extend the code. To calculate the resultant metric, it needs to find the updated densities the transmission of the extending node would result in. It updates the densities in the same way as the algorithm does when it chooses the extending node using successive use of (7), taking into accounts all the extending nodes in the left neighborhood graph of the chosen node. When the densities are updated, the function returns $1 - \prod_i (1 - P_e(i))$ as the metric representing the BLER of the extended code.

B. A Less Greedy Extending Algorithm

Algorithm 1 is greedy in the sense that at each step of the algorithm when it makes a decision on the next extending node, it chooses the best one, according to the metric. However, this greediness may not result in the best final extending sequence. It may be interesting to see how a less greedy algorithm performs. We propose to introduce a small perturbation in the algorithm: at each step, when it decides on the extending node, it allows a small compromise so as to

make it possible to choose nodes other than the best one. To make this work, we choose the extending node randomly from all possible extending nodes with a probability reflecting their possible contribution to improve the next metric. In particular, the next possible metrics for all nodes in the depth-L right neighborhood graph of the target message node are calculated. Then a probability is assigned to each of these nodes such that it is proportional to the inverse of its corresponding next metric. That is, after the algorithm calculates the next metric $\alpha_{\omega,d}$ for each node (ω, d) in $\mathcal{N}_{i,0}^{\vec{L}}$, the node is assigned a selecting probability $\rho_{\omega,d} = \frac{\alpha_{\omega,d}^{-1}}{\sum_{j,l} \alpha_{j,l}^{-1}}$. The algorithm chooses the extending node to be (ω, d) with the probability $\rho_{\omega,d}$. Obviously, the node which Algorithm 1 would choose, i.e. that of the best metric, will have the largest chance of being chosen.

C. The proposed IR-HARQ scheme

In this section we describe how an IR-HARQ scheme can be implemented based on the proposed puncturing and extending algorithms. The proposed IR-HARQ scheme transmits the code bits, each corresponding to one node in the PC graph of the mother polar code. The choice of the code bits to transmit is made in two successive phases. In the first phase, which is the puncturing phase, the code bits are transmitted in the transmission order given by the proposed puncturing pattern in Section III. In the second phase, which is the extending phase, the code bits are transmitted according to the extending sequence given by the extending algorithm in this section. The transition time between the puncturing phase and the extending one is determined by the intermediate rate R_I.

For a given mother code of rate $R_M = K/N_M$, a puncturing algorithm is used to determine the transmission order of the N_M code bits. Transmission begins with the transmission of the first K code bits (in the specified order), and decoding is attempted at the receiver. If decoding fails, additional code bits are transmitted, one at a time, with decoding attempted after each bit, until decoding is successful.[4] After N_I of the code bits of the polar code have been transmitted, for some $N_I \leq N_M$, the transmitter begins transmitting bits according to the extending sequence. Let $R_I = K/N_I \geq R_M$ be the intermediate rate after which code bits are selected from the extending sequence instead of according to the puncturing pattern. The transmitter continues to send code bits over the channel until decoding succeeds at the receiver and an acknowledgement is received at the transmitter.[5] The extending algorithm is run to give a long enough extending sequence to consider the case that the SNR is low and a lot of redundancy code bits may be required for successful decoding. As we will show in the next section, the throughput of the system depends on the chosen mother code rate R_M, and the intermediate rate $R_I \geq R_M$, to a significant amount. In the following section we present the simulation results for the proposed IR-HARQ scheme and find good intermediate rate R_I for a given mother code rate R_M.

[4]A more practical method is to send additional code bits in small clusters, with decoding attempted after each cluster. The resulting degradation in throughput arising from this increased granularity will be negligible if the cluster size is sufficiently small relative to K.

[5]Note that unlike LDPC codes, the SC decoder of the polar codes will always produce a valid codeword. In order for it to know if a decoding failure has occurred, one could use an error-detecting code like a CRC code to declare the decoding failure. This would lead to a certain rate loss in the throughput performance, but the rate loss is negligible as the length of the message word increases.

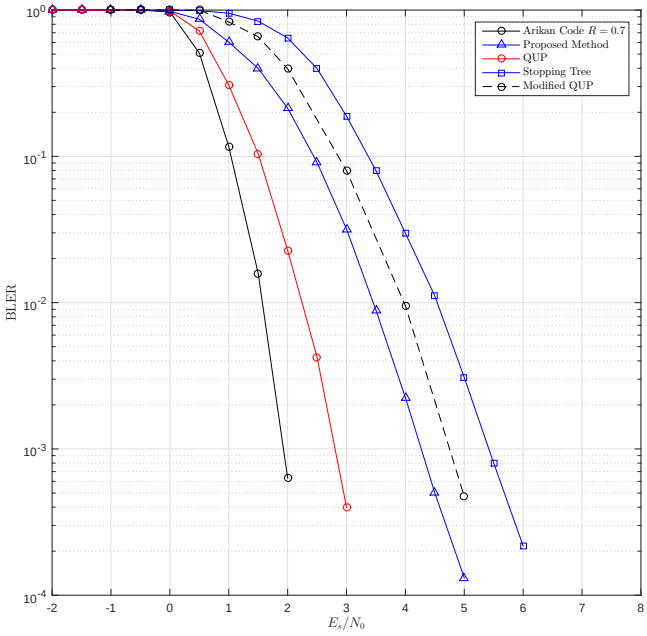

Fig. 2. The BLERs of punctured polar codes with different puncturing algorithms.

V. Simulation Results

In this section we present Monte-Carlo simulation results to evaluate IR-HARQ schemes which use the different combinations of puncturing and extending methods presented in Sections III and IV. We assume a BIAWGN channel with antipodal signaling $\{\pm 1\}$. SNR is defined as $10\log(1/N_0)$, where N_0 is the one-sided power spectral density of the Gaussian noise. Let K be the length of the message word. All mother codes are designed according to [10].

We begin with an investigation into the effects of the puncturing algorithms. Figure 2 shows the BLER performance of a polar code of length $N_M = 2048$ and rate 0.5 that has been punctured to rate 0.7 using each of the three puncturing algorithms described in Section III. While the proposed method outperforms the stopping tree algorithm, the QUP algorithm is even better. In fact, the code produced with QUP is almost as good as a directly generated rate 0.7 Arikan polar code with $N_M = 2048$ (also shown in Figure 2), despite having a shorter codeword length (1462 vs 2048 code bits). Codes produced with QUP also have the advantage over Arikan's codes in that the codeword length is not limited to a power of two. Unfortunately, with QUP the set of frozen bits is determined after the puncturing pattern has been identified. As a result, codes generated with different puncturing lengths are not rate-compatible, and so QUP is not applicable to our system where a mother code with a fixed frozen set is to be punctured and

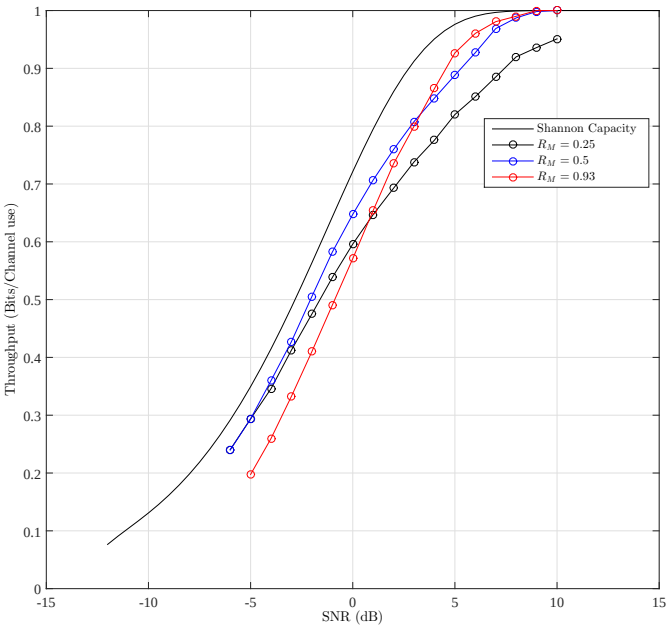

Fig. 3. Throughput of the proposed IR-HARQ scheme with the proposed puncturing algorithm for different R_M, all with $R_I = R_M$.

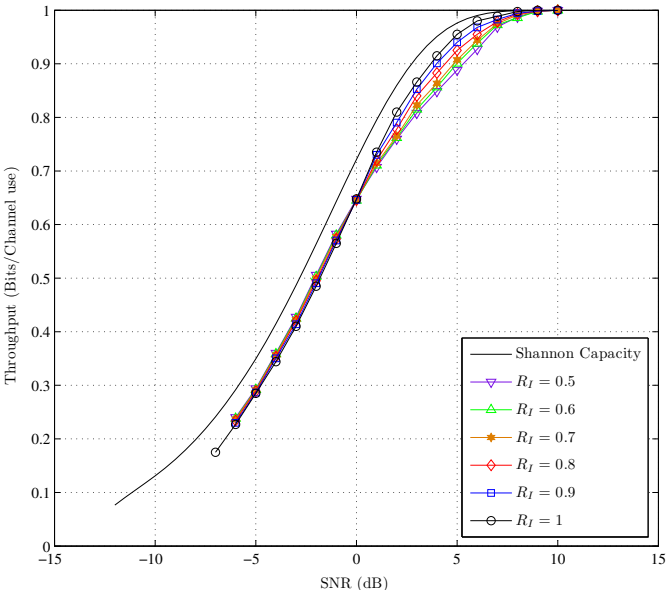

Fig. 4. Throughput of the proposed IR-HARQ scheme with the proposed puncturing and extending algorithm with different R_Is, with $R_M = 0.5$.

extended to different rates without changing the frozen set. In an attempt to find a good puncturing scheme for our system, we considered a modified QUP scheme where the transmission order of the punctured bits is determined according to QUP, but the frozen set is fixed and determined prior to puncturing. Unfortunately, as can be seen in Figure 2, the performance of the modified QUP is not satisfactory, so we employ our proposed puncturing algorithm in our system.

From Fig. 3 we can see the impact of the choice of the mother code rate on the throughput of the IR-HARQ system. Using a high-rate mother code gives superior performance at high SNRs but inferior performance at low SNRs while the converse is true for low-rate codes. To realize reasonable throughput over a wide range of SNRs it therefore seems advisable to choose R_M to be a moderate value, such as $R_M = 0.5$.

It is possible to start extending before all the code bits of the mother code have been transmitted, i.e. $R_I > R_M$. Fig. 4 shows the effect of rate R_I for a mother code rate of $R_M = 0.5$. As illustrated, increasing R_I (i.e. reducing the number of transmitted code bits before commencing extending) leads to a significant gain in throughput at high SNRs, while leading to only a very slight reduction at low SNRs. We therefore recommend using $R_I = 1$, as arguably this gives the best overall performance. In this case the puncturing phase involves transmitting only $N_I = K$ code bits, and those K code bits have the same indices as the K information bits, $\mathcal{A}$. Once these K code bits have been transmitted, the extending algorithm is used to determine the extending sequence. In theory one could start the extending phase even earlier (i.e. $R_I > 1$), but this was found to be ineffective, because the code was not rate-1 decodable (which limits the throughput at very high SNRs), or degrades the performance at low SNRs because of the greedy nature of the algorithm.

Fig. 5 compares the throughput of the proposed IR-HARQ scheme with parameters $(R_M, R_I) = (0.5, 1)$ to the throughput of IR-HARQ schemes based on Raptor and LDPC codes. As can be seen the scheme based on Raptor codes can perform very close to the capacity at low SNRs, while leading to significant throughput drop-off at high SNRs due to non-zero overhead of the mother code. The scheme based on LDPC code, uses both puncturing and extending of an LDPC mother code of rate $8/13$, similar to [7]. The mother code is punctured using the algorithm in [5] to get higher-rate codes, and is extended by adding a number of columns to the right corner of the parity check matrix of the mother code and an equal number of rows to its bottom (for more details see [7]), to get lower-rate codes. It can be seen that this scheme can get close to the capacity at intermediate SNRs. However throughput degradation occurs at both high and low SNRs. In comparison to these schemes, the proposed method can remain close to the capacity for a wider range of SNR.

We also simulated the less greedy version of the extending algorithm. Because this algorithm is random, to measure the performance of IR-HARQ schemes constructed with this algorithm, we took 20 different realizations of the algorithm output. The average of the 20 curves was a bit worse than the proposed greedy algorithm and is not shown here. The performance of the best realization provided by the less greedy algorithm is shown in Fig. 5. It shows slightly lower throughput compared to the greedy algorithm at high SNRs, and a small improvement at low SNRs. Because the greedy algorithm tries to maximize the throughput with each transmitted code bit, it is very effective at high SNRs, but those decisions made early in constructing the extending sequence can lead to weaker low-rate codes. The less greedy algorithm sacrifices the high-rate performance for better low-rate performance. Which algorithm is preferable depends on the application.

Fig. 5 also provides a comparison of the proposed scheme

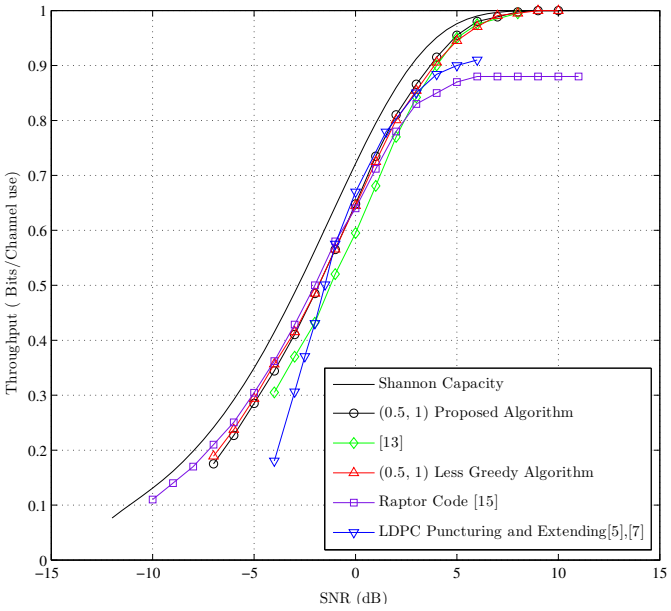

Fig. 5. Throughput comparison of the proposed IR-HARQ scheme with other alternatives.

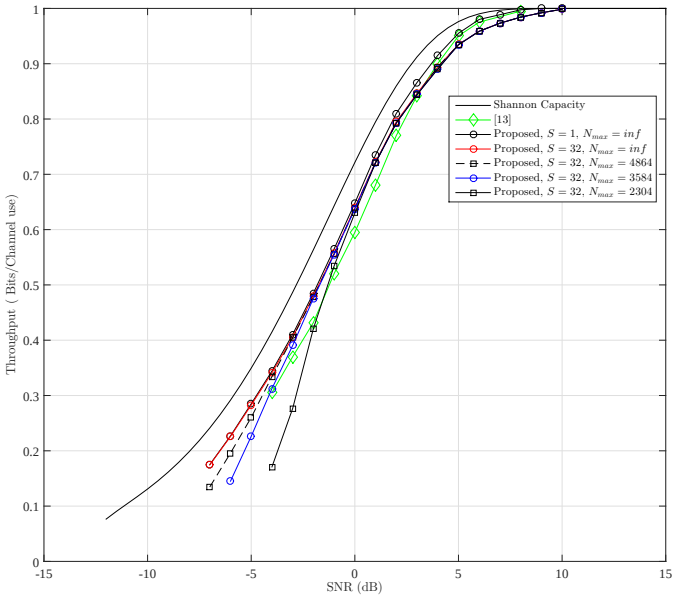

Fig. 6. Throughput of the proposed system for different number of decodings with a cluster size of $S = 32$.

to the scheme proposed in [13]. In the approach proposed therein, the transmitted code bits are either the code bits or the message bits. In other word, they are chosen to be either from column 0 or n, which makes it a special case of our proposed scheme where the code bits can be possibly chosen from all columns of the graph. As can be seen, allowing the extending nodes to be chosen from any nodes of the graph, if chosen appropriately, can result in improvements in the throughput. However, because of the differences in the choice of some system parameters between our scheme and the results in [13], a more detailed comparison is needed.

In [13], transmission is carried out in clusters of S bits at a time, with decoding attempted after each cluster has been received, whereas the results for our scheme use a cluster size of $S = 1$. Furthermore, in [13] the maximum number of transmitted code bits is limited to some value, N_{max}, and if the message is not recovered after receiving N_{max} code bits, a transmission failure is declared and the message is retransmitted from the beginning. In our scheme no such limit is necessary, but can be implemented. The effect of S and N_{max} on the throughput are shown in Fig. 6. As can be seen, increasing S from 1 to 32, while keeping N_{max} unlimited, causes a degradation in the throughput at high SNRs. This degradation is caused by the increased granularity between decoding attempts, whereby a whole cluster is transmitted even through only a few additional bits may have been needed. At low SNRs the cluster size is less important, but limiting N_{max} does have an effect. As shown in Fig. 6 decreasing N_{max} leads to a decrease in throughput. In general, it is better to continue transmitting additional code bits than to drop a packet and restart, although practical system-level considerations generally impose some limit on N_{max} eventually.

It is also worth seeing the effects of increasing the length of the mother code on the performance of the extending

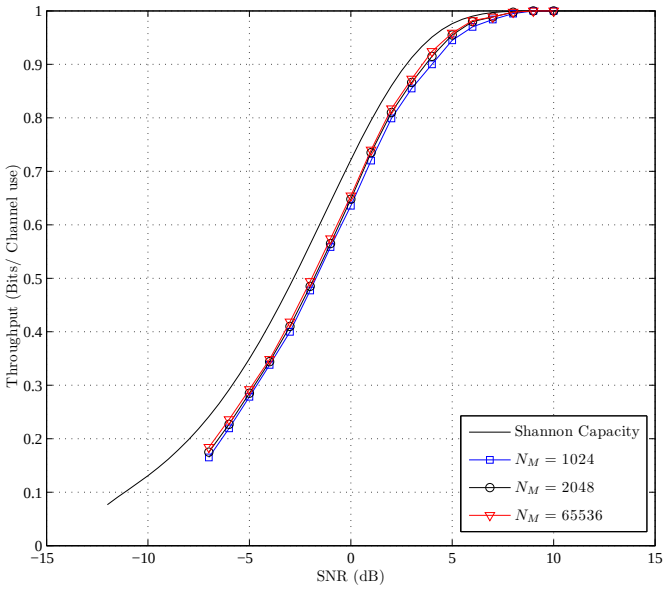

Fig. 7. Throughput of the proposed IR-HARQ scheme for different lengths of the mother polar code, with $R_M = 0.5$ and $R_I = 1$.

algorithms since polar codes are theoretically optimum only in the asymptotic sense. Using the technique in [10], we have designed mother polar codes of three different length: $N_M = 1024$, 2048, and 65536. Fig. 7 depicts the throughput of the three different codes where used with the proposed puncturing and extending algorithms. As can be seen the $N_M = 2048$ code is up to 0.3 dB better than $N_M = 1024$ at intermediate SNRs, while this gain becomes smaller at lower SNRs. This improvement in the throughput brought by the increase in length seems to be limited as a length of 65536 does not yield significant gain over the length $N_M = 2048$.

VI. Conclusion

We have introduced the concept of polar code extending as an effective method to construct polar codes for IR-HARQ schemes. The framework of polar code extending is quite general and includes the schemes which retransmit the code bits or the message bits and perform maximum ratio combining as a special case. It allows a possibly repeated transmission of code bits from a set of code bits of size $N_M \log_2 N_M + K$, for a mother polar code of length N_M with message size K. We proposed a throughput-specific extending algorithm to extend a possibly punctured polar code. The proposed extending method, in conjunction with the proposed puncturing method can result in a universal capacity-approaching IR-HARQ scheme that can be implemented with the same encoding and decoding complexity as the original polar mother code.

For future work, we note that finding the best next N_E extending nodes from the set of code bits of size $N_M \log_2 N_M + K$ can be computationally infeasible for large values of N_M as the size of the search space is $(N_M \log_2 N_M + K)^{N_E}$ which grows exponentially with N_E. The proposed method is one search method to find these extending nodes by selecting them one at a time from within a certain subset of all nodes in the PC graph. One could think of other search methods which can search for the best extending node from within the entire graph, as opposed to from within a subset of the graph, while maintaining an acceptable search complexity. Also as another attempt to reduce the greediness, finding the best D extending nodes at a time can be considered in conjunction with an efficient search algorithm.

References

[1] S. Lin and P. S. Yu, "A hybrid ARQ scheme with parity retransmission for error control of satellite channels," *IEEE Trans. Commun.*, vol. 30, no. 7, pp. 1701-1719, Jul. 1982.

[2] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Trans. Commun.*, vol. 36, no. 4, pp. 389-400, Apr. 1988.

[3] Z. Lin and A. Svensson, "New rate-compatible repetition convolutional codes," *IEEE Trans. Inf. Theory*, vol. 46, no. 7, pp. 2651-2659, Nov. 2000.

[4] J. Ha, J. Kim, and S. W. McLaughlin, "Rate-compatible puncturing of low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, pp. 2824-2836, Nov. 2004.

[5] J. Ha, J. Kim, D. Klinc, S.W. McLaughlin, "Rate-compatible punctured low-density parity-check codes with short block lengths," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 728-738, Feb. 2006.

[6] J. Li and K. Narayanan, "Rate-compatible low density parity check codes for capacity approaching ARQ scheme in packet data communications," in Proc. *Int. Conf. Comm., Internet, and Info. Tech. (CIIT)*, St. Thomas, US Virgin Islands, pp. 376-132, Nov. 2002.

[7] M. R. Yazdani and A.H. Banihashemi, "On construction of rate-compatible low-density parity-check codes," *IEEE Commun. Lett.*, vol. 8, pp. 159-161, Mar. 2004.

[8] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, pp. 3051-3073, Jul. 2009.

[9] E. Arikan, "A performance comparison of polar codes and Reed-Muller codes," *IEEE Commun. Lett.*, vol. 12, pp. 447-449, Jun. 2008.

[10] R. Mori and T. Tanaka, "Performance of polar codes with the construction using density evolution," *IEEE Commun. Lett.*, vol. 13, no. 7, pp. 519-521, Jul. 2009.

[11] A. Eslami and H. Pishro-Nik, "A practical approach to polar codes," *Proc. Int. Symp. Inf. Theory (ISIT)*, St. Petersburg, Russia, Jul. 2011, pp. 16-20.

[12] K. Niu, K. Chen, and J. Lin, " Beyond turbo codes: Rate-compatible punctured polar codes," in Proc. *IEEE Int. Conf. on Commun. (ICC)*, Budapest, Hungary, Jun. 2013, pp. 3423-3427.

[13] K. Chen, K. Niu and J. Lin, "A hybrid ARQ scheme based on polar codes," *IEEE Commun. Lett.*, vol. 17, no. 10, pp. 1996-1999, Oct. 2013.

[14] N. Hussami, S. B. Korada and R. Urbanke, "Performance of polar codes for channel and source coding," *Proc. Int. Symp. Inf. Theory (ISIT)*, Seoul, South Korea, Jun. 2009, pp. 1488-1492.

[15] E. Soljanin, N. Varnica and P. Whiting, "Punctured vs rateless codes for hybrid ARQ," *Proc. Inf. Theory Workshop (ITW)*, Punta del Este, Uruguay, Mar. 2006, pp. 155-159.

[16] T. J. Richardson, A. M. Shokrollahi and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619-637, Feb. 2001.