

On the Performance of Hybrid FEC/ARQ Systems Using Rate Compatible Punctured Turbo (RCPT) Codes

Douglas N. Rowitch, *Member, IEEE*, and Laurence B. Milstein, *Fellow, IEEE*

Abstract—This paper introduces a hybrid forward-error correction/automatic repeat-request (ARQ) system that employs rate compatible punctured turbo (RCPT) codes to achieve enhanced throughput performance over a nonstationary Gaussian channel. The proposed RCPT-ARQ system combines the performance of turbo codes with the frugal use of incremental redundancy inherent in the rate compatible punctured convolutional codes of Hagenauer. Moreover, this paper introduces the notion of puncturing the systematic code symbols of a turbo code to maximize throughput at signal-to-noise ratios (SNR's) of interest. The resulting system provides both an efficient family of achievable code rates at middle to high SNR and powerful low-rate error correction capability at low SNR.

Index Terms—Hybrid FEC/ARQ, turbo codes.

I. INTRODUCTION

CONVENTIONAL applications of forward-error correction (FEC) coding usually consist of the selection of fixed rate coding schemes that are well suited to the channel characteristics and the acceptable error rates for the data to be transmitted [1]. In some cases, however, a more flexible scheme may be desirable to accommodate different error protection requirements, or channels with unknown or time-varying parameters [1]. In such cases, automatic repeat-request (ARQ) protocols have been successfully employed to adapt to changes in the transmission medium. In particular, hybrid FEC/ARQ protocols have been established which exploit both the predictable performance of FEC codes and the rate flexibility of ARQ protocols [1]–[4]. Hagenauer [2] introduced rate compatible punctured convolutional (RCPC) codes with such an application in mind.

RCPC codes are constructed from a single rate $1/M$ convolutional code, wherein a family of higher rate codes is formulated by puncturing successively greater numbers of code symbols. These codes have practical utility in that the system requires a single rate $1/M$ convolutional encoder and Viterbi decoder. The

transmitter and receiver need only share a puncturing table to determine which code symbols to transmit at a given time, and the receiver may simply insert erasures for all code symbols that have not yet been received. The rate compatibility requirement of these codes restricts the puncturing rule such that all of the code symbols of a high rate punctured code are used by the lower rate codes; i.e., the high rate codes are embedded in the lower rate codes. In this manner, the transmitter need only transmit *supplemental* code symbols to get to the next lowest rate code. RCPC-ARQ protocols, therefore, fall into the so-called class of *incremental redundancy* codes, in that check digits are incrementally retransmitted to adaptively meet the error performance requirements of the system. These codes are also appealing from an information theoretic standpoint, since no received information is discarded (as in other hybrid FEC/ARQ schemes [1]), especially when used in concert with Chase's code combining [5]. For a comprehensive overview of hybrid ARQ protocols, including RCPC and code combining, see [3] or [4].

Turbo codes, introduced by Berrou *et al.* in 1993 [6], caused a great stir in the coding community and have prompted a great deal of research. One version of a turbo code consists of a parallel concatenation of two or more recursive systematic convolutional (RSC) constituent codes, each separated by a pseudo-random interleaver of block size N , resulting in a composite encoder of overall rate $1/M$. Transmitted are the systematic (i.e., information) code symbols of the first encoder, and the nonsystematic (i.e., parity) code symbols of all constituent encoders. The receiver provides a soft-input, soft-output (SISO) decoder corresponding to each RSC encoder and an iterative structure within which the decoders pass information.

Rate compatible turbo codes have been considered to achieve unequal error protection (UEP) [7], and the use of turbo codes in an ARQ protocol was first proposed in mid 1997 [8], in which additional constituent encoders are added to obtain rates lower than that of a conventional turbo encoder; i.e., rates below $1/2$. Other papers considered hybrid ARQ protocols with turbo codes providing throughput efficiencies below $1/2$ [9].

Rowitch and Milstein [10], Jung *et al.* [11], [12], and Li and Imai [13] independently considered *punctured* rate compatible turbo codes to provide higher throughputs above rate $1/2$. In particular, punctured rates as high as $8/9$ were specified in [10], whereas systems in [12] and [13] both provided for uncoded transmission (rate 1) and punctured code rates as high as $3/4$ and $2/3$, respectively. All of the initial results on rate compatible punctured turbo (RCPT) codes as applied to an ARQ scheme were published in 1997.

Paper approved by R. Koetter, the Editor for Coding Theory and Techniques of the IEEE Communications Society. Manuscript received April 16, 1999; revised October 20, 1999. This work was supported in part by the National Science Foundation under Grant NCR-9213140 and the Focused Research Initiative on Wireless Multimedia Networks under Grant DAAH04-95-1-0248.

D. N. Rowitch was with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0407 USA. He is now with Qualcomm Inc., San Diego, CA 92121 USA (e-mail: drowitch@qualcomm.com).

L. B. Milstein is with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0407 USA (e-mail: milstein@ece.ucsd.edu).

Publisher Item Identifier S 0090-6778(00)05393-9.

The formulation of the RCPT codes follows that of the RCPC codes almost directly; however, there are several notable distinctions. The most obvious distinction is that the rate $1/M$ convolutional encoder is replaced by a rate $1/M$ turbo encoder and, of course, the rate $1/M$ soft decision Viterbi decoder is replaced with a bank of SISO decoders and the associated iterative decoding structure. The more subtle distinction considered in this paper has to do with the process for selecting the puncturing tables corresponding to different transmission rates of the protocol. For RCPC codes, the puncturing tables were selected such that each successive transmission of parity symbols yielded the greatest possible increment in the free distance of the new lower rate code. For RCPT codes, however, there are other, perhaps more important, criteria. In particular, the subsets of code symbols that have been transmitted at a given stage implicitly determine the number of SISO decoders that can be employed in the receiver. Indeed, if nonsystematic parity symbols have not been received for two or more RSC encoders, iterative decoding is not possible. It will be shown that there are situations where it is profitable, at high coding rates, to puncture some of the systematic code symbols so that more nonsystematic symbols may be sent, and thus utilize more or all of the SISO decoders resident in the receiver.

In Section II, we present an overview of the RCPT-ARQ system, and in Section III, we conduct an analysis of system performance. Section IV contains numerical results and, finally, in Section V can be found the conclusions.

II. FEC/ARQ SYSTEM OVERVIEW

A. Channel Assumptions

We assume coherent binary signalling over an additive white Gaussian channel at a given symbol signal-to-noise ratio (SNR) E_s/N_0 , where E_s is the energy-per-symbol and N_0 is the one-sided power spectral density of the noise. We may, therefore, characterize the output of the channel, at any given time, as $y = x + n$, where $x \in \{+1, -1\}$ and n is a zero-mean Gaussian random variable with standard deviation $\sigma = \sqrt{N_0/2E_s}$. We, furthermore, assume an errorless, low capacity feedback channel over which the receiving system can transmit positive (ACK) or negative (NAK) acknowledgments.

B. RCPT Codes

The binary data source is encoded into blocks in accordance with a low redundancy (N, K) block error detection code. This encoded data source is input to the RCPT encoder which: 1) turbo encodes the data sequence and 2) partitions the resulting code symbols for each systematic/parity stream into subblocks of size (N/P) , where P is the so-called puncturing period. In Fig. 1(a), an example RCPT encoder is formed from an underlying rate $1/M$ turbo encoder, consisting of $M-1$ rate $1/2$ constituent RSC encoders. Note that the composite encoder can, in general, be formed by using a fewer number of lower rate RSC encoders, and that the constituent RSC code rates need not be equal. For example, if $M = 4$, then one could use either three rate $1/2$ RSC encoders, or one rate $1/3$ and one rate $1/2$ RSC encoder. It is seen in the figure that the systematic bits of all but the first encoder are discarded and that the resulting single

systematic plus $(M - 1)$ parity streams are each represented as a collection of P subblocks. The subblock A_1 , for example, contains a fraction $1/P$ of the systematic bits consisting of bit 0, bit P , bit $2P$, etc. Since the turbo code is of rate $1/M$, there are MP total subblocks produced for potential transmission. If we collect all such subblocks and store them in matrix form, as shown in the figure, we obtain an $M \times P$ matrix, where each row corresponds to a different systematic/parity stream, and each column refers to a different decimated subsequence of that data stream. The interleavers, $\{\pi_i\}_{i=2}^{M-1}$, are taken to be “ S -random” permutations [14], as they tend to provide superior performance for the short frame sizes typically used in ARQ protocols. S -random permutations satisfy a distance condition ensuring that each symbol to be permuted is a distance S or more from the previous S adjacent symbols. Fig. 1(b) illustrates an example rate $1/2$ constituent RSC encoder of memory $\nu = 4$, and generator matrix $(1, g_b/g_a)$, where the generator polynomials g_a and g_b have octal representations $(37)_{\text{octal}}$ and $(21)_{\text{octal}}$, respectively.

The RCPT puncturing rule amounts to sending collections of one or more subblocks of the turbo encoded data, such that at least P subblocks are sent in the initial transmission and no subblock is sent twice. The code construction allows for a family of codes of rates

$$R_\ell = \frac{P}{P + \ell}, \quad \ell = 0, 1, \dots, (M - 1)P. \quad (1)$$

For each value of ℓ , we define a binary $M \times P$ puncturing matrix $\mathbf{a}(\ell)$. If $a_{ij}(\ell) = 1$, then the j th subblock of the i th systematic/parity stream belongs to the subcode of rate R_ℓ . Therefore, based on the above restrictions, $\mathbf{a}(0)$ must contain P ones, $\mathbf{a}(\ell + 1)$ must have ones in the same positions as in $\mathbf{a}(\ell)$ plus an additional one, and, finally, $\mathbf{a}((M - 1)P)$ is a matrix of all ones. Referring back to Fig. 1(a), if we select a code of rate R_ℓ , we can overlay the puncturing matrix $\mathbf{a}(\ell)$ over the depicted matrix of data; for each 1 in the puncturing matrix, the corresponding subblock of data is selected and considered part of the codeword.

As an example, suppose we construct an RCPT code with $M = 3$ and $P = 4$, formed from two rate $1/2$ constituent RSC encoders. Then, the RCPT code can select any code rate in the set $\{1, (4/5), (2/3), (4/7), (1/2), (4/9), (2/5), (4/11), (1/3)\}$, by selecting, respectively, 4, 5, \dots , 12 subblocks of encoded data. The set of puncturing matrices could, for example, take the form

$$\begin{array}{cccc} \mathbf{a}(0) & \mathbf{a}(1) & \mathbf{a}(2) & \mathbf{a}(8) \\ \left[\begin{array}{c} 1111 \\ 0000 \\ 0000 \end{array} \right] & \left[\begin{array}{c} 1111 \\ 0010 \\ 0000 \end{array} \right] & \left[\begin{array}{c} 1111 \\ 0010 \\ 1000 \end{array} \right] & \dots \left[\begin{array}{c} 1111 \\ 1111 \\ 1111 \end{array} \right]. \end{array}$$

Note that, for this example, all four subblocks of systematic bits (first row) are sent in the first packet $[\mathbf{a}(0)]$, the third subblock of the first encoder’s nonsystematic parity bits is sent in the second packet $[\mathbf{a}(1)]$, and the first subblock of the second encoder’s nonsystematic parity bits is sent in the third packet $[\mathbf{a}(2)]$. Note that after the first packet is received, one can attempt to decode the information bits based on the cyclic redundancy check (CRC) alone, and after the second packet, one can attempt to

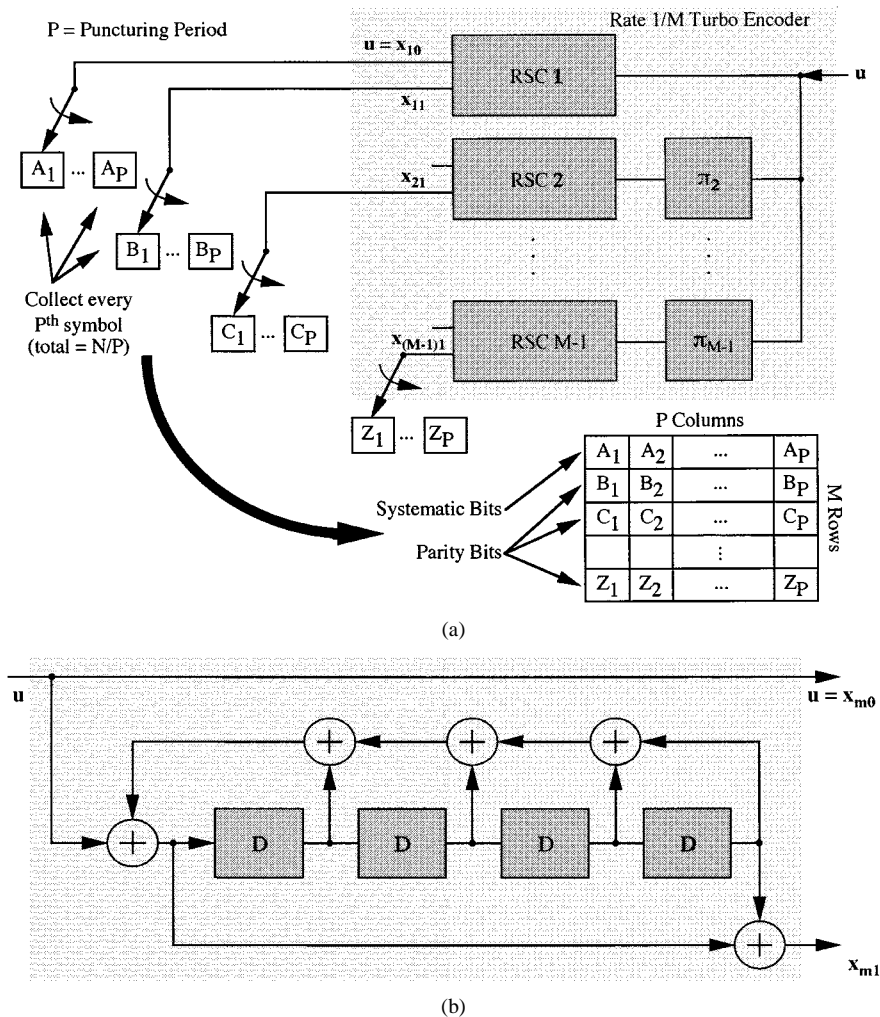


Fig. 1. Example RCPT-ARQ encoder implementation. (a) Example RCPT encoder. (b) Example RSC encoder.

use the SISO decoder for the first constituent encoder to decode the data. It is only after the third packet that the receiver can invoke the iterative turbo decoder. In implementation, one need not utilize all possible code rates. Indeed, for ARQ applications with large channel variations, an exponential increase in ℓ may be desirable (e.g., $\ell = 1, 2, 4, 8, \dots$) [2]. It should be clear at this point that the transmission packet size will, in general, be variable.

C. RCPT-ARQ Protocol

We assume, without loss of generality, a selective-repeat ARQ strategy due to its bandwidth efficiency, noting that the RCPT-ARQ protocol can be trivially adapted to either stop-and-wait or go-back- N schemes. Furthermore, we do not consider the effects of finite memory or unreliable feedback on the performance of the system.

The RCPT-ARQ protocol performs the following steps.

- 1) Encode K information digits in accordance with an (N, K) block error detection code.
- 2) Input the N bit coded information block into the rate $1/M$ RCPT encoder. Store the resulting M systematic and parity streams at the transmitter, possibly in a matrix as depicted in Fig. 1(a), for potential transmission.
- 3) Initialize ℓ (e.g., set $\ell = 1$).
- 4) Transmit the subblocks as indicated by $\mathbf{a}(\ell)$ (that have not yet been transmitted).
- 5) Attempt to decode the rate R_ℓ code using the code symbols received thus far,¹ inserting erasures for those symbols not yet received. Within the iterative decoding loop after each iterative update:
 - a) Hard quantize the cumulative vector of likelihood ratios on the information sequence.
 - b) Calculate the syndrome of the (N, K) error detection codeword.
 - c) If there is an all-zero syndrome, exit the loop, output the K decoded information digits, and send an ACK to the transmitter. Otherwise, continue the decoding loop until a prescribed maximum number of iterations, upon which, if the syndrome is still nonzero, exit the loop, and send a NAK to the transmitter.

¹Note that in the absence of received parity from two or more encoders, one can optionally invoke the constituent SISO decoder on an individual basis, or check the CRC based on the received systematic bits alone. If a zero-syndrome on the embedded CRC can be obtained in this manner, then the overhead and delay of iterative decoding can be avoided.

- 6) At the transmitter, if an ACK is received, reset the protocol and proceed to step 1).
- 7) Otherwise, if a NAK is received, increment ℓ to the next appropriate value and proceed to step 4).

If decoding is not successful after the transmission of all subblocks of code symbols [i.e., $\ell = (M - 1)P$ and $R_\ell = 1/M$], then there are several system design options. The simplest option is to give up and accept a given bit- or frame-error rate upon failure at the lowest code rate. Alternatively, we could allow the protocol to reset, commencing with step 3). The receiver, in turn, may optionally reset its receive buffers, or, ideally, retain the previously received quantities and employ code combining [5] to merge the data. When combining, packets should be weighted according to their relative reliability [5], [15]. Other options might include retransmission of only systematic data, or only nonsystematic data since these two data types play different roles in the decoder; these last options are not, however, treated in this paper. Note that steps 5a)–5c) can be performed directly on the received values corresponding to the systematic code symbols (assuming all N systematic symbols have been received), and potentially avoid the overhead of the iterative decoding process (if the SNR is high enough).

III. ANALYSIS

In this section, we consider the throughput performance of an RCPT-ARQ system with a given encoder and set of puncturing tables. We then briefly discuss the average distance spectrum of a turbo code and how the distance spectrum can be applied to the search for optimal puncturing tables.

A. Throughput Analysis

Let a frame denote the binary vector of K digits which are produced by the decoder. We use the average throughput R_{AV} as the primary figure of merit for the RCPT-ARQ system. Combining the rate of the block error detection code and (1), we have²

$$R_{AV} = \frac{K}{N} \cdot \frac{P}{P + \ell_{AV}} \quad (2)$$

where ℓ_{AV} is the average number of additionally transmitted subblocks. We assume that the number of parity bits $N - K$, used in the error detecting code, is sufficient to guarantee a negligible probability of undetected frame error $P_u(E)$.

For incremental redundancy hybrid-ARQ systems, one can derive a closed-form solution for R_{AV} which, in this case, amounts to determining ℓ_{AV} . For each code rate R_ℓ used in the ARQ protocol, if one can determine the probability of a frame error, then the collection of these probabilities at a given SNR can be used to determine ℓ_{AV} , and thus, R_{AV} [2], [4], [16]. However, determining such probabilities analytically for turbo codes is intractable. Moreover, upper bounds on such probabilities are known to be weak, especially at rates above the cutoff rate [17]. While the authors applied such bounds to the probability of a frame error in an attempt to lower bound the throughput of an RCPT-ARQ system, such bounds were found to be extremely weak [16]. We therefore rely on simulated

²If tail-bits are transmitted for one or more of the constituent encoders, the average throughput will be reduced accordingly.

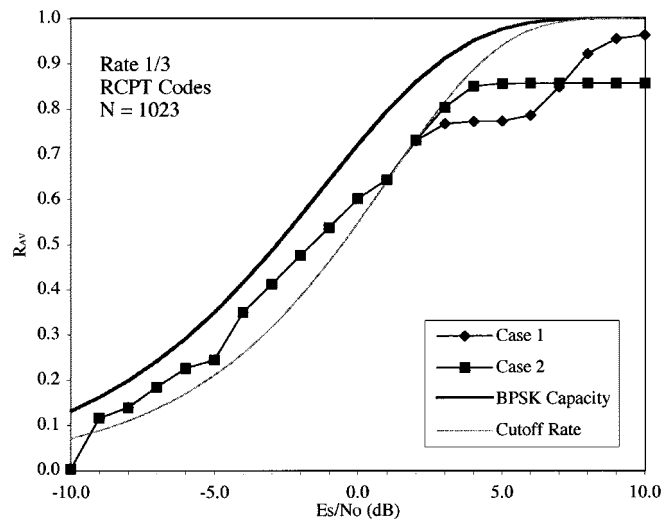


Fig. 2. Throughput of two RCPT-ARQ codes over the Gaussian channel. Codes $P = 8$, $M = 3$, $\nu = 4$, $N = 1023$, $K = 993$, and $S = 20$ (S is the S -random interleaver constraint).

throughput to evaluate the efficacy of RCPT-ARQ systems and refer the reader to the above references for a more thorough analytic treatment of ARQ system throughput.

B. Preliminary Results

Initially, experiments were conducted to determine the sensitivity of throughput performance to different puncturing patterns. As asserted in the Introduction, it is necessary that at least some nonsystematic parity symbols be transmitted to the receiver in order for iterative turbo decoding to be possible. In general, puncturing *systematic* code symbols of a turbo code leads to a greater performance loss than when puncturing nonsystematic code symbols. In fact, there are no examples in the literature, known to the authors, in which puncturing of the systematic symbols of a turbo code is advocated. We, nevertheless, considered this option. Given a turbo encoder and a fixed puncturing period P , we considered the following puncturing strategies.

Case 1: In the initial transmission ($\ell = 0$), transmit all P subblocks of systematic code symbols and rely on the embedded error detection code to identify error-free information blocks. If errors are detected at the receiver, then a transmission of two subblocks of nonsystematic code symbols ($\ell = 2$) would be required based on the above constraint.

Case 2: In the initial transmission ($\ell = 1$), transmit $P - 1$ subblocks of systematic code symbols and two subblocks of nonsystematic code symbols (from distinct constituent encoders). In this case, if errors are detected at the receiver, a transmission of the remaining subblock of systematic code symbols ($\ell = 2$) would transpire.

Puncturing more than one systematic subblock in the initial transmission was also considered and was determined to be inferior [16] to option 2. For Case 1, we may transmit at rates R_ℓ , with $\ell = 0, 2, 3, 4, \dots$, and for Case 2, with $\ell = 1, 2, 3, 4, \dots$. These two options are compared in Fig. 2.

In this figure, we plot simulated throughput, which is defined as the average code rate required for error-free information transmission. We used an RCPT encoder made up of two RSC encoders, as depicted in Fig. 1(b) with $N = 1023$ and $P = 8$, and show throughput for two selections of puncturing tables. For Case 1, we transmit at rates $R_0 = 8/8$, $R_2 = 8/10, \dots$, whereas for Case 2, we transmit at rates $R_1 = 8/9, R_2 = 8/10, \dots$, where it is noted that the effective throughputs are slightly less due to the transmission of error detection parity symbols and encoder tail-bits. The specific puncturing tables used in this experiment can be found in [16]. For these results, we allowed the protocol to repeat three additional times and used code combining to merge repeated data. A (1023,993) binary BCH code was used to perform error detection.

It is immediately seen that by puncturing a small percentage of the systematic code symbols (Case 2), one is able to sustain higher throughput at the lower SNR's of interest. Furthermore, we note that the RCPT-ARQ codes exhibit the exceptional performance of turbo codes at very low SNR's, noting that the simulated throughput exceeds the cutoff rate [18] for SNR below 1 (dB) and provides essentially "error-free" communication at 1–2 (dB) from the binary-input continuous-output additive white Gaussian noise Shannon capacity. It is interesting to note that punctured turbo codes can sustain such performance at rates higher than 1/2.

In subsequent sections, we investigate the search for optimal puncturing tables, given a specific encoder and puncturing period. We use these preliminary results to simplify the search by imposing the following restriction. For the highest rate R_1 , we always puncture one systematic subblock of code symbols and include two subblocks of nonsystematic code symbols from distinct constituent encoders. For the next highest rate R_2 , we always include the previously punctured subblock of systematic code symbols. With these restrictions imposed, the search is otherwise free to consider any puncturing pattern with the goal of maximizing overall throughput.

C. Turbo Average Distance Spectrum

Traditional techniques for bounding the performance of convolutional codes [19] quickly become intractable, when applied to turbo codes, for sufficiently large block sizes, due to the innumerable state mappings introduced by the random interleaver. Weaker transfer function bounds have been developed for turbo codes [17], [20], which average out the effect of the interleaver on the bound. This is accomplished using a so-called *uniform* interleaver, an artificial construct that maps a given input sequence into *all possible* permutations of the sequence with equal probability. This technique yields an *average* upper bound on turbo code error probability; i.e., it is possible to have error rates greater than the upper bound when using particularly "bad" interleavers. Moreover, the bound can be weak when compared to error rates of turbo codes using "good" interleavers.

If one considers an arbitrary turbo decoder error event, the input-weight refers to the number of bit errors and the

output-weight refers to the overall Hamming weight of the error event. One may calculate the so-called *input-output weight enumerating function* (IOWEF) of a turbo code which indicates the average number of error events of a given input and output weight. From the IOWEF, one can derive the *average weight enumerators* (AWE's) of the turbo code. The AWE indicates the average number of error events of a given output weight. Then, the *average distance spectrum* (ADS) is defined as the set of nonzero AWE's. Lastly, we define *average free distance* of the code to be the minimum output weight with a nonzero AWE.

While the ADS is most often used in the evaluation of performance bounds, it was also found to be very useful in the search for optimum puncturing patterns for a given encoder, block size, and puncturing period.

D. Optimal Puncturing

Given a rate $1/M$ turbo encoder of block size N and a selected puncturing period P (we assume N to be an integer multiple of P), we would like to devise a scheme for selecting optimal puncturing tables; that is, puncturing tables which maximize the system throughput.

Attempts to identify the optimal puncturing pattern for a given code rate using computer simulations proved to be both time consuming and potentially flawed because they required the selection of a random interleaver. In simple terms, if one candidate puncturing pattern outperformed another, there was no way to determine if this was due to the relative distance spectra of the two punctured codes, or due to the compatibility of the interleaver with one of the candidates. In some sense, the selection of the interleaver(s) adds an additional dimension to the search for optimal puncturing tables, rendering an already time-consuming task even more so. This provided the motivation to consider the use of the ADS to aid in the search for optimal puncturing tables. Since the ADS, from the previous section, averages out the effect of the interleaver on the performance of the code, the average distance spectra for two candidate codes can be used to designate the better code.

Given two alternative puncturing patterns and their corresponding average distance spectra, we considered several criteria for selecting the "better" candidate. One obvious criterion is selecting the candidate with the maximum average free distance (or the minimum AWE, if there is a tie). Call this the *free-distance* criterion. We also considered the *effective free-distance* [21] criterion which uses the distance spectra due to input-weight 2 error events in the same manner as the previous criterion. Lastly, we considered a criterion which we dub the *ADS slope* criterion. In this case, we fit a regression line to the first 30, or so, terms of the ADS. The slope coefficient of this fitted line represents a measure of the rate of growth of the AWE's as output distance d increases. For this criterion, we select the candidate with the minimum slope. The first two criteria should select a family of punctured codes which perform well at "high"

TABLE I
PUNCTURING TABLES (IN OCTAL) FOR ENCODER A

RCPT CODE: M=3 and P=4

BlockSize	Punctured Code Rate							
	4/5	2/3	4/7	1/2	4/9	2/5	4/11	1/3
N=256	16	17	17	17	17	17	17	17
	02	02	12	12	16	16	17	17
	02	02	02	12	12	16	16	17
N=512	16	17	17	17	17	17	17	17
	02	02	12	12	16	16	17	17
	02	02	02	12	12	16	16	17
N=1024	16	17	17	17	17	17	17	17
	02	02	12	12	16	16	17	17
	02	02	02	12	12	16	16	17
N=4096	16	17	17	17	17	17	17	17
	02	02	12	12	16	16	17	17
	02	02	02	12	12	16	16	17

RCPT CODE: M=3 and P=8

BlockSize	Punctured Code Rate															
	8/9	4/5	8/11	2/3	8/13	4/7	8/15	1/2	8/17	4/9	8/19	2/5	8/21	4/11	8/23	1/3
N=256	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	001	001	003	003	007	007	007	027	037	037	137	137	177	177	377	377
	001	001	001	003	003	007	017	017	017	037	037	137	137	177	177	377
N=512	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	002	002	002	006	016	016	016	056	076	076	076	176	177	177	377	377
	002	002	006	006	006	016	036	036	036	037	137	137	137	177	177	377
N=1024	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	002	002	006	006	016	016	016	056	076	076	076	176	177	177	377	377
	002	002	002	006	006	016	036	036	036	037	137	137	137	177	177	377
N=4096	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	002	002	006	006	016	016	016	056	076	076	276	276	376	376	377	377
	004	004	004	014	014	034	074	074	074	174	174	374	374	376	376	377

SNR, since the minimum weight error events dominate performance at high SNR. We would expect that the *ADS slope* criterion will yield a family of subcodes that perform better at lower SNR where higher distance error events have a nontrivial contribution to error performance. It turns out, as will be shown in subsequent sections, that the *ADS slope* criterion is superior for ARQ applications. The *ADS slope* criterion yields subcodes which have higher (poorer) error floors, but have better performance at lower SNR. Within the ARQ protocol, these subcodes tend to be employed at their respective low SNR values (see [16]). It may be that for other applications, such as UEP, other selection criteria result in a better set of RCPT puncturing tables (it depends whether you plan to be on the steep part of the error rate curve, or the flat error floor portion of the curve). Herein, we use the term “optimal puncturing” to mean optimal subject to the *ADS slope* criterion.

In this investigation, we consider several turbo encoders of varied block size and puncturing periods of 4 and 8. Given these fixed parameters, the search for optimal puncturing tables proceeds as follows, where we begin with the

unpunctured code and work backward, successively puncturing more subblocks.

- 1) Initialize the rate selector $\ell = (M - 1)P$ and puncturing table $\mathbf{a}((M - 1)P)$ to a matrix of all ones (i.e., no punctures).
- 2) Decrement ℓ by 1 (to get to the next highest code rate).
- 3) Form a set of all possible candidate puncturing tables, $\{\tilde{\mathbf{a}}(\ell)\}$, which are obtained by taking the matrix $\mathbf{a}(\ell + 1)$, locating a distinct 1 and setting the 1 to 0. For a given value of ℓ , there will be precisely $\ell - P + 1$ candidate puncturing tables.
- 4) Eliminate any candidates that fail to satisfy the constraints outlined in Section III-B.
- 5) For each surviving candidate puncturing table, calculate the ADS.
- 6) Fit a line to the leading terms of the ADS.
- 7) Select the candidate which has the minimum slope as computed in step 6). If there a tie, select the candidate with the larger average free distance (or minimum AWE, if there is a tie).

TABLE II
PUNCTURING TABLES (IN OCTAL) FOR ENCODER B

RCPT CODE: M=3 and P=4

BlockSize	Punctured Code Rate							
	4/5	2/3	4/7	1/2	4/9	2/5	4/11	1/3
N=256	16	17	17	17	17	17	17	17
	01	01	05	05	15	15	17	17
	01	01	01	05	05	15	15	17
N=512	16	17	17	17	17	17	17	17
	01	01	05	05	15	15	17	17
	01	01	01	05	05	15	15	17
N=1024	16	17	17	17	17	17	17	17
	01	01	05	05	15	15	17	17
	01	01	01	05	05	15	15	17
N=4096	16	17	17	17	17	17	17	17
	01	01	05	05	15	15	17	17
	01	01	01	05	05	15	15	17

RCPT CODE: M=3 and P=8

BlockSize	Punctured Code Rate															
	8/9	4/5	8/11	2/3	8/13	4/7	8/15	1/2	8/17	4/9	8/19	2/5	8/21	4/11	8/23	1/3
N=256	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	004	004	104	104	104	124	125	125	125	135	335	335	375	375	377	377
	020	020	020	220	224	224	224	264	265	265	265	275	275	375	375	377
N=512	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	001	001	001	041	051	051	055	055	055	155	155	175	375	375	377	377
	001	001	041	041	041	045	045	245	265	265	275	275	275	375	375	377
N=1024	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	020	020	024	024	224	224	264	264	265	265	265	275	375	375	377	377
	020	020	020	024	024	224	224	264	264	274	374	374	374	375	375	377
N=4096	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	020	020	024	024	224	224	264	264	274	274	374	374	375	375	377	377
	020	020	020	024	024	224	224	264	264	274	274	374	374	375	375	377

- 8) Designate the puncturing table selected from the previous step as $a(\ell)$.
- 9) If $\ell > 1$, proceed to step 2). If $\ell = 1$, the search is complete.

E. Optimal RCPT Codes

We examined several candidate turbo codes using $\nu = 4$ (i.e., 16 states) constituent encoders. In particular, we considered the following encoders.

- Encoder A:** Rate 1/3; generators: $(1, 21/37)_{\text{octal}} + (21/37)_{\text{octal}}$.
- Encoder B:** Rate 1/3; generators: $(1, 23/35)_{\text{octal}} + (23/35)_{\text{octal}}$.
- Encoder C:** Rate 1/3; generators: $(1, 33/31)_{\text{octal}} + (33/31)_{\text{octal}}$.
- Encoder D:** Rate 1/4; generators: $(1, 33/31)_{\text{octal}} + (33/31)_{\text{octal}} + (33/31)_{\text{octal}}$.

Encoder A was proposed in [6], encoder B was suggested in [22], and encoder C was presented in [23]. Encoder D is a trivial extension of encoder C to obtain a parent code rate of 1/4. The plus sign between code generators indicates the

presence of an interleaver. For these codes, we considered puncturing periods of 4 and 8, and block sizes of 256, 512, 1024, and 4096.

The results of the search are presented in Tables I–IV. The top row of each table indicates the achievable code rates given by (1), for $\ell = 1, \dots, P(M-1)$. The next four sets of M rows are the puncturing tables (in octal). For a given block size and code rate, the puncturing table represents a binary $M \times P$ matrix. While the AWE's for each block size and punctured code rate were generated, due to space limitations, we were unable to provide them in this paper. They are, however, presented in detail in [16].

From these tables, we made four general observations and conclusions. Firstly, for a given encoder and puncturing period, the resulting optimal puncturing tables are not necessarily the same for different block sizes; it turns out that the puncturing tables are all the same (for the four block sizes considered) *only* for encoders A, B, and C with $P = 4$. Secondly, for a given encoder and puncturing period, as N increases, the ADS gets better and better (i.e., better free distances and smaller enumerators at equivalent distances). This improvement is attributable to the so-called interleaver gain of turbo codes

TABLE III
PUNCTURING TABLES (IN OCTAL) FOR ENCODER C

RCPT CODE: M=3 and P=4

BlockSize	Punctured Code Rate							
	4/5	2/3	4/7	1/2	4/9	2/5	4/11	1/3
N=256	16	17	17	17	17	17	17	17
	02	02	12	12	13	13	17	17
	02	02	02	12	12	13	13	17
N=512	16	17	17	17	17	17	17	17
	02	02	12	12	13	13	17	17
	02	02	02	12	12	13	13	17
N=1024	16	17	17	17	17	17	17	17
	02	02	12	12	13	13	17	17
	02	02	02	12	12	13	13	17
N=4096	16	17	17	17	17	17	17	17
	02	02	12	12	13	13	17	17
	02	02	02	12	12	13	13	17

RCPT CODE: M=3 and P=8

BlockSize	Punctured Code Rate															
	8/9	4/5	8/11	2/3	8/13	4/7	8/15	1/2	8/17	4/9	8/19	2/5	8/21	4/11	8/23	1/3
N=256	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	001	001	011	011	013	013	213	213	253	253	253	353	373	373	377	377
	001	001	001	011	011	013	013	113	113	133	173	173	173	177	177	377
N=512	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	001	001	011	011	013	013	213	213	253	253	253	353	373	373	377	377
	001	001	001	011	011	013	013	113	113	133	173	173	173	177	177	377
N=1024	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	002	002	002	042	052	052	252	252	253	253	253	353	373	373	377	377
	001	001	011	011	011	051	051	071	071	073	173	173	173	177	177	377
N=4096	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	002	002	042	042	043	043	043	243	343	343	363	363	373	373	377	377
	002	002	002	042	042	052	252	252	252	253	253	273	273	373	373	377

[21]. Thirdly, for a given encoder and block size, the ADS for the larger puncturing period ($P = 8$) is almost always better than the ADS for the smaller puncturing period ($P = 4$).³

The numerical results section of this paper will serve to compare these alternatives at the highest coding rates. Finally, regardless of the block size and puncturing period, it was concluded, *based on the respective encoder ADS coefficients*, that the codes appear to be presented in ascending order of performance; i.e., encoder A is the worst and encoder D is the best⁴ [16]. Encoders A, B, and C are of identical complexity, whereas encoder D requires the added complexity of an additional constituent encoder and decoder.

In order to determine the utility of using optimal puncturing, we may reverse the logic in step 7) of the search procedure, so as to produce the worst puncturing tables for a given encoder and puncturing period. In [16], such a puncturing table was obtained. The numerical results section of this paper will indicate the results of the comparison.

³We expect to do better with larger values of P since we have more degrees of freedom in how to implement the puncturing.

⁴It will be shown in Section IV that encoder D, in fact, performs worse than the other encoders.

IV. NUMERICAL RESULTS

A. RCPT-ARQ System Performance

Simulations were conducted to estimate the throughput of various RCPT-ARQ systems and to compare RCPT-ARQ performance with that of several reference systems. For each SNR, approximately 10^8 information bits were simulated. The S -random interleavers used in the simulations were selected via an *ad hoc* search procedure [16]. For puncturing period $P = 4$, all possible RCPT rates were used, whereas for puncturing period $P = 8$, rates corresponding to $\ell = 1, 2, 4, 6, 8, \dots$, were used. In these simulations, the ARQ protocol terminated upon reaching the lowest code rate ($1/M$); i.e., no repetitions of the protocol were allowed. The maximum number of iterations allowed at any given decoding attempt was fixed at 12. The log-MAP algorithm was used for the SISO decoding element. For the results presented in this section, we employed triple-error-correcting systematic extended binary primitive BCH codes [e.g., a (256, 231) binary BCH code for $N = 256$].

Figs. 3 and 4 compare all four encoders for several block sizes and puncturing period 8. In general, the throughput of the

TABLE IV
PUNCTURING TABLES (IN OCTAL) FOR ENCODER D

RCPT CODE: M=3 and P=4

BlockSize	Punctured Code Rate											
	4/5	2/3	4/7	1/2	4/9	2/5	4/11	1/3	4/13	2/7	4/15	1/4
N=256	16	17	17	17	17	17	17	17	17	17	17	17
	01	01	01	03	03	03	13	13	13	17	17	17
	01	01	01	01	03	03	03	13	13	13	17	17
	00	00	01	01	01	03	03	03	13	13	13	17
N=512	16	17	17	17	17	17	17	17	17	17	17	17
	01	01	01	03	03	03	13	13	13	17	17	17
	01	01	01	01	03	03	03	13	13	13	17	17
	00	00	01	01	01	03	03	03	13	13	13	17
N=1024	16	17	17	17	17	17	17	17	17	17	17	17
	01	01	01	03	03	03	13	13	13	17	17	17
	01	01	01	01	03	03	03	13	13	13	17	17
	00	00	01	01	01	03	03	03	07	07	07	17
N=4096	16	17	17	17	17	17	17	17	17	17	17	17
	01	01	01	03	03	03	03	13	17	17	17	17
	01	01	01	01	03	03	07	07	07	07	17	17
	00	00	01	01	01	03	03	03	03	13	13	17

RCPT CODE: M=3 and P=8

BlockSize	Punctured Code Rate															
	8/9	4/5	8/11	2/3	8/13	4/7	8/15	1/2	4/9	2/5	4/11	1/3	4/13	2/7	4/15	1/4
N=256	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	001	001	001	011	011	011	011	051	071	071	073	173	173	373	377	377
	001	001	001	001	011	011	011	011	051	071	071	073	173	373	373	377
	000	000	002	002	002	022	023	023	023	033	133	133	173	173	177	377
N=512	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	001	001	001	011	011	011	011	051	071	071	073	173	173	373	377	377
	001	001	001	001	011	011	011	011	051	071	071	073	173	373	373	377
	000	000	001	001	001	021	023	023	023	033	133	133	173	173	177	377
N=1024	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	001	001	001	021	021	021	023	023	063	063	163	173	373	373	377	377
	001	001	001	001	021	021	021	023	023	063	063	163	173	373	377	377
	000	000	001	001	001	021	021	021	023	027	067	067	067	167	167	377
N=4096	376	377	377	377	377	377	377	377	377	377	377	377	377	377	377	377
	001	001	001	001	011	011	011	011	051	071	173	173	373	377	377	377
	000	000	002	042	042	042	043	043	047	047	047	057	057	077	377	377
	001	001	001	001	001	005	005	045	045	047	047	067	167	167	167	377

RCPT-ARQ codes surpass the cutoff rate over a wide range of SNR's. We, initially, note that the throughput is improved for increasing block size. This is due, in part, to the improved performance of turbo codes as the block size is increased, and also due to the fact that for larger block sizes, the error detection parity bits and encoder tail-bits represent a smaller percentage of the total number of bits transmitted. It is also seen that encoders A, B, and C all provide similar throughput efficiency, whereas, surprisingly, the throughput of encoder D is worse (except at very low SNR). This discrepancy was examined in detail in [16] and is summarized as follows. On one hand, to get to the same punctured rate, the constituents of encoder D must be punctured more severely than the constituents of encoder C. However, the addition of a third constituent in encoder D makes for a stronger turbo code. When examining

the frame-error rate (FER) performance of the punctured subcodes of a particular rate (e.g., rate 1/2) between encoders C and D, it turns out that encoder D has a much better error floor, but encoder C has a slight gain in the waterfall portion of the FER curve (the crossover occurs at $FER \approx 10^{-4}$). Within an RCPT-ARQ protocol, the punctured subcode gets successfully employed, on average, over the range of SNR where encoder C is superior [16]. We, therefore, come to the conclusion that using simple two constituent encoders results in the best performance, which is a nice result from a standpoint of minimizing receiver complexity.

It was shown in [16] that the simulated FER is more or less a step function and can be deduced from the throughput curves which reflect a rather steep departure from error-free to error-saturated performance as a function of the SNR.

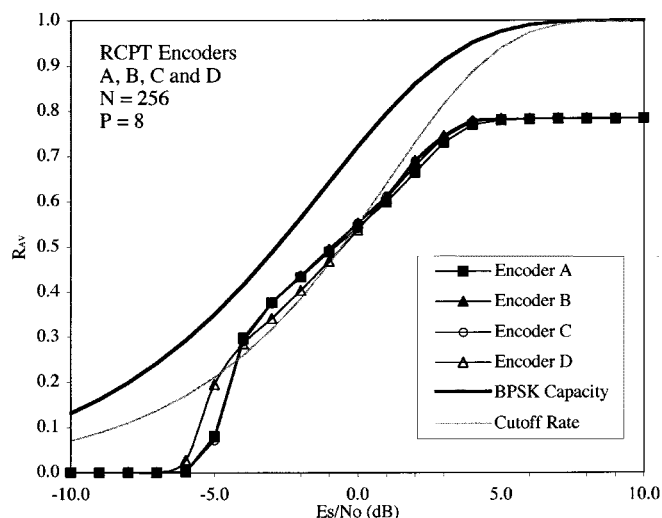


Fig. 3. Simulated throughput for $N = 256$ and puncturing period $P = 8$. Codes A, B, C, and D, $N = 256$, $K = 231$, and $S = 11$.

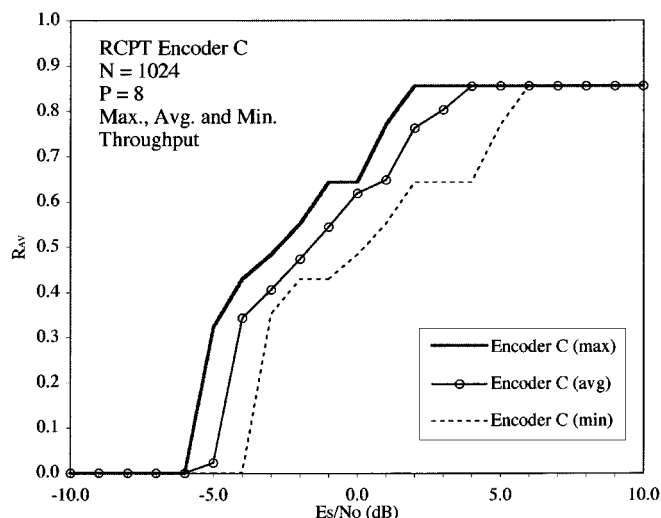


Fig. 5. Simulated minimum, maximum, and average throughput for encoder C, $P = 8$. Codes encoder C, $N = 1024$, $K = 993$, and $S = 19$. Illustrates the robustness and stability of RCPT codes.

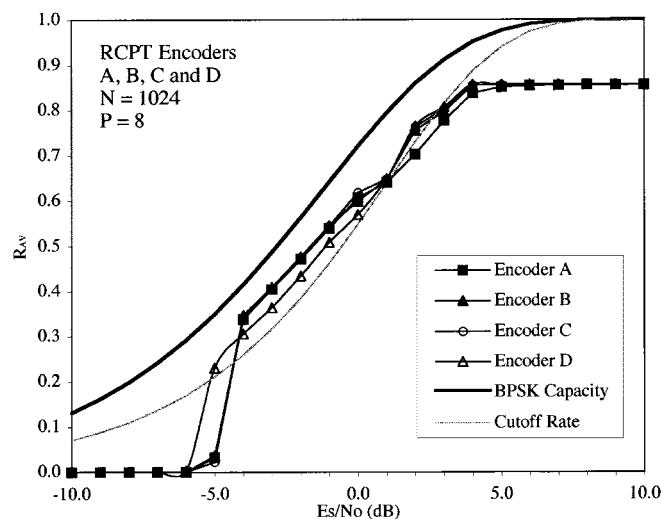


Fig. 4. Simulated throughput for $N = 1024$ and puncturing period $P = 8$. Codes A, B, C, and D, $N = 1024$, $K = 993$, and $S = 19$.

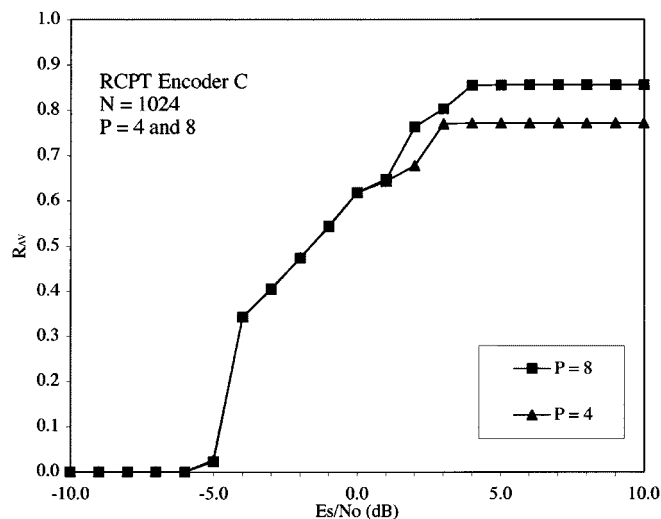


Fig. 6. Simulated average throughput for encoder C, $P = 4$, and $P = 8$. Codes encoder C, $N = 1024$, $K = 993$, and $S = 19$. Demonstrates the throughput gains that can be achieved due to using larger puncturing periods.

To illustrate the stability and robustness of the RCPT codes, we plot, in Fig. 5, the average throughput along with the minimum and maximum throughput achieved on any one simulated transmission. It is apparent that the average throughput is about 1 dB from the maximum and about 0.6–3 dB from the minimum. The compactness of the minimum and maximum about the average suggests that the RCPT-ARQ system is robust.

Next, for encoder C, we compare different puncturing periods for $N = 1024$, as shown in Fig. 6. It is seen that the throughput is improved when using a larger puncturing period. At high SNR, a large improvement is seen due to the fact that for $P = 8$, the system can support a maximum code rate of $8/9$, whereas for $P = 4$, the maximum code rate is $4/5$. In addition, for a larger puncturing period, the puncturing rule is allowed more degrees of freedom, yielding better code distance properties. At lower SNR's, the performance is seen to converge for the two alternatives.

Finally, in Fig. 7, we examine the effects of different schemes for protocol repetition. That is, if we transmit all of the code

symbols for a given block [i.e., $\ell = (M - 1)P$] without a successful decode, we consider various alternatives. In the curves shown thus far, we assumed that the protocol simply terminates and accepts a certain amount of frame and/or bit errors at low SNR [call this case (a)]. Alternatively, we can repeat the protocol one or more times until a successful decode occurs; here we may reset the receive buffers, discarding previously received information [case (b)], or use code combining [5] to merge current and previous data [case (c)]. We omit case (b) from consideration as there is negligible gain over case (a). For case (c), we allow the protocol to repeat up to three additional times. We see that with protocol repetition and code combining, the throughput curve is cleanly extended to accommodate lower SNR's, and the performance remains above the cutoff rate at about a decibel or two from the Shannon limit. Thus, it would appear that protocol repetition and code combining represent a simple low complexity method of obtaining nontrivial

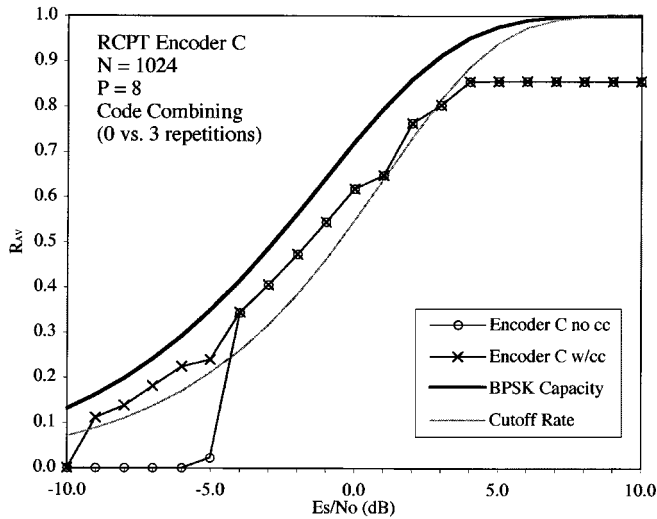


Fig. 7. Simulated average throughput for encoder C, $P = 8$ with and without code combining. Codes encoder C, $N = 1024$, $K = 993$, and $S = 19$. Demonstrates the throughput gains that can be achieved due to protocol repetition with code combining.

throughput at very low SNR. The alternative to code combining would be to use lower rate parent turbo codes, which imply a comparatively significant complexity cost.

In [16], the worst puncturing patterns for encoder A, $P = 8$, and $N = 1024$ were derived. It was shown that the coding gain due to using the best over the worst puncturing patterns was as much as 0.7 dB at an average throughput of approximately 0.5 bits/T [16].

B. Comparison to the Reference System

As a reference system, we consider the performance of the RCPC code of Hagenauer [2], formed from a rate 1/3, 64-state parent convolutional encoder and a block size of $N = 1024$. In addition, we consider the performance of an uncoded ARQ protocol [i.e., (re)transmit the uncoded block with error detection parity bits until a successful decode] and use Chase's code combining at the receiver. In the latter case, we allow up to 64 transmissions of the data block. Thus, the RCPC code and Chase code may transmit at a worst-case rate of 1/3 and 1/64, respectively. These results are depicted in Fig. 8 and are compared to the RCPT-ARQ system using encoder C, $P = 8$, and $N = 1024$. It is clear that the RCPC technique represents a significant improvement over a simplistic uncoded ARQ protocol and that the RCPC system closely tracks the computational cutoff rate over a wide range of SNR's. It is also apparent that the RCPT system outperforms the RCPC system for all SNR's by as much as 1–2 dB. Now, let us make the same comparison at a block size of $N = 256$, shown in Fig. 9. We see that the RCPT system is no longer the outright winner. In this case, there is a small range of SNR where the RCPC system is superior, and at other SNR's, the two performance curves are much closer than the performance curves for the larger block size. We therefore conclude that RCPT systems will outperform RCPC systems for sufficiently large block size.

In implementation, the RCPT system will probably be of greater complexity. In [24], it is claimed that the processing load in the log-MAP decoder is bounded at no more than four

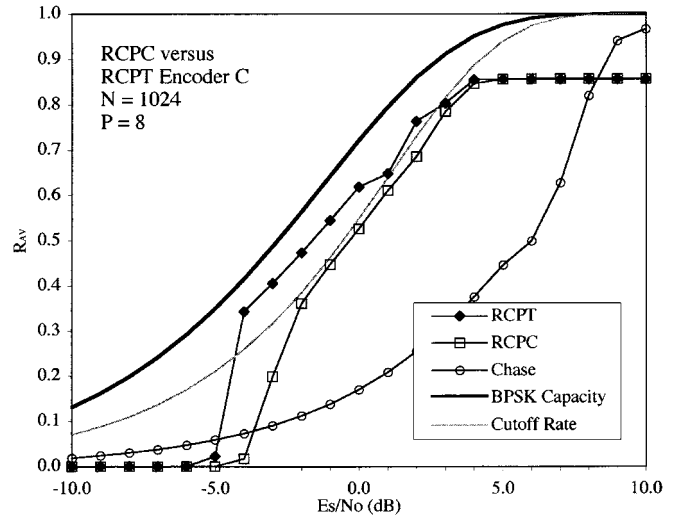


Fig. 8. Throughput comparison between RCPC and RCPT codes for $N = 1024$. Codes RCPT encoder C, $N = 1024$, $K = 993$, and $S = 19$. RCPC rate 1/3, 64-state encoder. Chase rate 1/64 ARQ with code combining. Illustrates the coding gain of RCPT-ARQ systems over established reference systems.

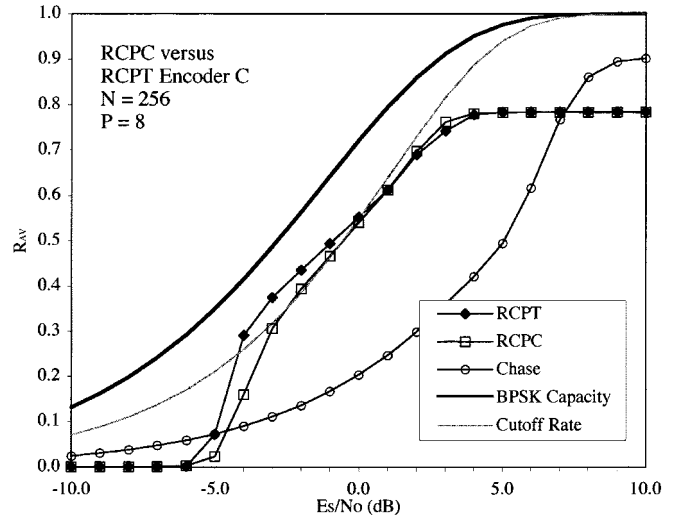


Fig. 9. Throughput comparison between RCPC and RCPT codes for $N = 256$. Codes RCPT encoder C, $N = 256$, $K = 231$, and $S = 11$. RCPC rate 1/3, 64-state encoder. Chase rate 1/64 ARQ with code combining. Illustrates the coding gain of RCPT-ARQ systems over established reference systems. Note that the RCPT gain is smaller than that shown in the previous figure and that RCPT performs worse than RCPC at some SNR's.

times that of a conventional Viterbi decoder, for the same convolutional code. Given that we compared a two-constituent turbo code (each being a 16-state code) to a 64-state convolutional code, we may conclude that the RCPT decoder is at most twice as complex as the RCPC decoder in terms of arithmetic operations. The memory requirements for the RCPT system will likely be higher due to the storage of soft-likelihood and metric quantities within the turbo decoder. In addition, we would expect a greater decoding delay in the RCPT-ARQ system.

V. CONCLUSION

In this paper, we introduced a novel application of turbo codes to a hybrid FEC/ARQ system. RCPT codes provide a versatile

family of achievable code rates, and may be processed with a single encoder and decoder. These codes were shown to perform near the Shannon limit at low SNR, while providing excellent throughput when the SNR is high. This paper also introduced the concept of puncturing the *systematic* code symbols of a turbo code to obtain better performance at high code rates. Optimum puncturing strategies were developed and puncturing tables were specified for several encoders, block sizes, and puncturing periods. The RCPT systems were shown to outperform the RCPC codes of Hagenauer [2] for sufficiently large block size.

Future research may consider the application of serial or hybrid parallel-serial concatenated, interleaved codes within an RCPT-ARQ protocol. In addition, new bounds on turbo codes (e.g., [25]) may prove useful in the search for good RCPT encoders and puncturing patterns based on their remarkable stability above the cutoff rate.

REFERENCES

- [1] S. Lin and D. J. Costello Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
- [2] J. Hagenauer, "Rate compatible punctured convolutional codes (RCPC Codes) and their applications," *IEEE Trans. Commun.*, vol. 36, pp. 389–400, Apr. 1988.
- [3] D. J. Costello Jr., J. Hagenauer, H. Imai, and S. B. Wicker, "Applications of error-control coding," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2531–2560, Oct. 1998.
- [4] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
- [5] D. Chase, "Code combining—A maximum likelihood decoding approach for combining an arbitrary number of noisy packets," *IEEE Trans. Commun.*, vol. COM-33, pp. 385–393, May 1985.
- [6] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. ICC'93*, Geneva, Switzerland, May 1993, pp. 1064–1070.
- [7] A. S. Barbulescu and S. S. Pietrobon, "Rate compatible turbo codes," *Electron. Lett.*, vol. 31, pp. 535–536, Mar. 1995.
- [8] K. R. Narayanan and G. L. Stuber, "A novel ARQ technique using the turbo coding principle," *IEEE Commun. Lett.*, vol. 1, pp. 49–51, Mar. 1997.
- [9] J. Hamorsky, U. Wachsmann, J. B. Huber, and A. Cizmar, "Hybrid automatic repeat request scheme with turbo codes," in *Proc. 1997 Int. Symp. on Turbo Codes*, Brest, France, Sept. 1997, pp. 247–250.
- [10] D. N. Rowitch and L. B. Milstein, "Rate compatible punctured turbo (RCPT) codes in a hybrid FEC/ARQ system," in *Proc. Communications Theory Mini-Conf. of GLOBECOM'97*, Phoenix, AZ, Nov. 1997, pp. 55–59.
- [11] P. Jung, J. Plechinger, M. Doetsch, and F. M. Berens, "A pragmatic approach to rate compatible punctured turbo-codes for mobile radio applications," in *Proc. 6th Int. Conf. on Advances in Commun. and Control*, Corfu, Greece, June 1997.
- [12] P. Jung and J. Plechinger, "Performance of rate compatible punctured turbo-codes for mobile radio applications," *Electron. Lett.*, vol. 33, pp. 2102–2103, Dec. 1997.
- [13] J. Li and H. Imai, "Performance of hybrid-ARQ protocols with rate compatible turbo codes," in *Proc. 1997 Int. Symp. on Turbo Codes*, Brest, France, Sept. 1997, pp. 188–191.
- [14] D. Divsalar and F. Pollara, "Turbo codes for PCS applications," in *Proc. ICC'95*, Seattle, WA, June 1995, pp. 54–59.
- [15] B. A. Harvey and S. B. Wicker, "Packet combining systems based on the Viterbi decoder," *IEEE Trans. Commun.*, vol. 42, pp. 1544–1557, Feb./Mar./Apr. 1994.
- [16] D. N. Rowitch, "Convolutional and turbo coded multicarrier direct sequence CDMA, and applications of turbo codes to hybrid ARQ communication systems," Ph.D. dissertation, Univ. of California at San Diego, La Jolla, CA, June 1998.

- [17] S. Benedetto and G. Montorsi, "Average performance of parallel concatenated block codes," *Electron. Lett.*, vol. 31, pp. 156–158, Feb. 1995.
- [18] S. Benedetto, E. Biglieri, and V. Castellani, *Digital Transmission Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1987.
- [19] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun.*, vol. COM-19, pp. 751–772, Oct. 1971.
- [20] D. Divsalar, S. Dolinar, R. J. McEliece, and F. Pollara, "Transfer function bounds on the performance of turbo codes," Jet Propulsion Lab., Pasadena, CA, JPL TDA Prog. Rep. 42-122, Aug. 1995.
- [21] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 409–428, Mar. 1996.
- [22] L. C. Perez, J. Seghers, and D. J. Costello Jr., "A distance spectrum interpretation of turbo codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1698–1709, Nov. 1996.
- [23] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Trans. Commun.*, vol. 44, pp. 591–600, May 1996.
- [24] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 260–264, Feb. 1998.
- [25] A. M. Viterbi and A. J. Viterbi, "Improved union bound on linear codes for the input-binary AWGN channel, with applications to turbo codes," in *Proc. Winter 1998 Information Theory Workshop*, San Diego, CA, Feb. 1998, p. 72.



Douglas N. Rowitch (S'94–M'98) received the B.A. degree in applied mathematics in 1984, the M.A. degree in applied mathematics in 1984, the M.S.E.E. degree in 1994, and the Ph.D. degree in 1998 from the University of California at San Diego, La Jolla, CA.

He joined Qualcomm Incorporated, San Diego, CA, in 1998, where he is a Senior Staff Engineer/Manager working on third-generation wireless standards and associated ASIC developments. His research interests include coding theory, spread spectrum, and communication theory.



Laurence B. Milstein (S'66–M'68–SM'77–F'85) received the B.E.E. degree from the City College of New York, New York, in 1964, and the M.S. and Ph.D. degrees in electrical engineering from the Polytechnic Institute of Brooklyn, Brooklyn, NY, in 1966 and 1968, respectively.

From 1968 to 1974, he was employed by the Space and Communication Group of Hughes Aircraft Company, and from 1974 to 1976, he was a member of the Department of Electrical and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY. Since 1976, he has been with the Department of Electrical and Computer Engineering, University of California at San Diego (UCSD), La Jolla, where he is a Professor and former Department Chairman, working in the area of digital communication theory with special emphasis on spread-spectrum communication systems. He has also been a consultant to both government and industry in the areas of radar and communications.

Dr. Milstein was an Associate Editor for Communication Theory for the IEEE TRANSACTIONS ON COMMUNICATIONS, an Associate Editor for Book Reviews for the IEEE TRANSACTIONS ON INFORMATION THEORY, an Associate Technical Editor for the IEEE COMMUNICATIONS MAGAZINE, and the Editor-in-Chief of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. He was the Vice President for Technical Affairs in 1990 and 1991 of the IEEE Communications Society, and has been a Member of the Board of Governors of both the IEEE Communications Society and the IEEE Information Theory Society. He is also a member of Eta Kappa Nu and Tau Beta Pi. Dr. Milstein was a recipient of the 1998 Military Communications Conference Long-Term Technical Achievement Award and the 1999 UCSD Distinguished Teaching Award.