Improving the Efficiency of Reliability-Based Hybrid-ARQ with Convolutional Codes

Arun Avudainayagam, John M. Shea and Abhinav Roongta
Wireless Information Networking Group (WING)
Department of Electrical and Computer Engineering
University of Florida
Email: {arun@dsp, jshea@ece, abhinavr@dsp}.ufl.edu

***Abstract*— In this paper, we propose and evaluate the performance of a new adaptive HARQ technique that utilizes the correlation among bit errors at the output of a convolutional decoder. The proposed correlated bit-error hybrid ARQ (CB-HARQ) uses error events encountered in a soft-input soft-output (SISO) decoder to capture the correlation in bit errors. Retransmission is requested for selected coded symbols that have the potential to correct a set of message bits associated with these error events. The performance of the proposed technique is compared to conventional incremental redundancy HARQ that uses rate-compatible punctured convolutional (RCPC) codes and reliability-based HARQ (RB-HARQ) that does not utilize correlation between error events. The fundamental difference between CB-HARQ and RB-HARQ is that CB-HARQ tries to correct error events occurring in a decoder whereas RB-HARQ tries to correct individual bit errors. The results show that the proposed HARQ technique based on correlated bit errors can provide much higher throughput.**

I. Introduction

Conventional hybrid automatic repeat request (HARQ) [1] schemes are not adaptive to the channel in the sense that the response to a packet error is fixed: retransmit the entire packet as in Type-I HARQ or a predetermined subset of the packet as in incremental redundancy HARQ. In [2], [3] a reliability-based HARQ (RB-HARQ) scheme is proposed that adapts to the channel errors by requesting retransmissions for those bits that are deemed unreliable at the output of soft-input soft-output (SISO) decoder. SISO decoders typically accept the received symbols and estimates of the *a priori* probabilities and produce estimates of the *a posteriori* probabilities (APPs) for the message bits. In RB-HARQ, the output of the SISO decoder is used to identify the bits with low reliabilities and retransmission is requested for such bits because the bits with low reliabilities are more likely to be in error. However, as noted in [2], errors at the output of a decoder are typically time-correlated. Since the least reliable bits (LRBs) are more likely to be in error, the LRBs are also correlated i.e., if a bit has a low reliability it is likely that the adjacent bits also have a low reliability. The previous work on RB-HARQ [2], [3] does not make any significant effort to utilize this correlation between the bit errors. Therefore RB-HARQ would request retransmission for a set of consecutive bits with low reliabilities. This a conservative approach because correcting one LRB will have an effect on the neighboring LRBs due to the correlated nature of the output of a decoder. Thus, it is not necessary to request retransmissions for the entire set of consecutive LRBs.

In this paper, we propose and evaluate the performance of a new HARQ technique that utilizes the correlation between the bit errors at the output of the decoder. We call our technique correlated bit-errors based HARQ (CB-HARQ). The work presented in this paper answers the following questions:

- What phenomena in the decoder can succinctly capture the correlated nature of bit errors?
- How much information is required to correct a set of consecutive trellis sections that have decoded with correlated reliabilities?

The max-log-MAP implementation of a SISO decoder computes the reliability of a trellis section by considering the error event that separates the maximum-likelihood (ML) path and a competing path that differs from the ML path in the input for that trellis section. We first demonstrate that correlated bit reliabilities occur because the same error event is considered in the computation of the reliability for adjacent bits. We then design an ARQ scheme that requests for just enough bits for re-transmission to correct a particular error event thereby resolving all the correlated bit errors stemming from that particular error event. RB-HARQ targets individual bits that might require additional information whereas CB-HARQ target error-events encountered in decoding (thereby potentially correcting all bits associated with that error event). We will demonstrate that fewer re-transmission bits are required by the CB-HARQ scheme, thereby improving the throughput over RB-HARQ.

II. The Decoder

In order to understand the understand the CB-HARQ scheme presented in this paper, it is important to have a good grasp of max-log-MAP decoding of convolutional codes. In this section we briefly explain max-$\log$-MAP decoding of convolutional codes. Throughout this paper we assume the use of rate–1/2 convolution codes.

A. Max-log-MAP decoding of convolutional codes

Among all trellis-based decoding algorithms for linear codes, the BCJR maximum *a posteriori* (MAP) decoder [4]

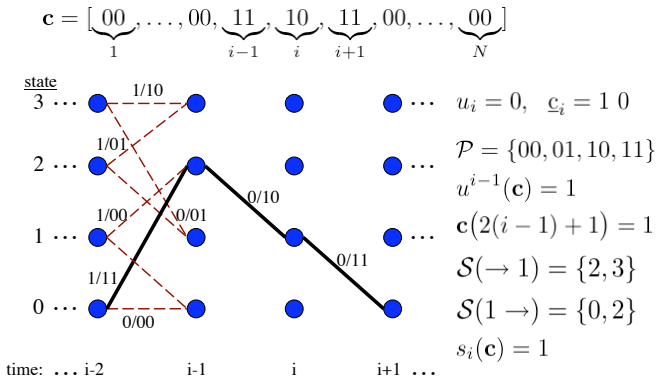

Fig. 1. The code-trellis for the $(5, 7)$ convolutional code with examples of the notation used in this paper.

achieves the optimum bit error probability. The inputs to a BCJR MAP decoder are *a priori* probabilities and likelihoods for the received symbols, and the output consists of *a posteriori* probabilities (APPs). The decoder is usually implemented in the log-domain for fast operation. Let u_i denote the message bit that is the input to the encoder at time i. Let $\underline{c}_i = [c_i^0, c_i^1]$ be the two-dimensional vector consisting of the two parity bits output by the encoder at time i. For each message bit u_i, the Log-MAP decoder computes the log-likelihood ratio (LLR) of the APP as follows,

$$L(u_i|\mathbf{r}) = \ln \frac{\mathrm{P}(u_i = 0|\mathbf{r})}{\mathrm{P}(u_i = 1|\mathbf{r})}, \quad (1)$$

$$= \ln \frac{\sum_{\mathbf{c}\in C_+^i} \mathrm{P}(\mathbf{c}|\mathbf{r})}{\sum_{\mathbf{c}\in C_-^i} \mathrm{P}(\mathbf{c}|\mathbf{r})}, \quad (2)$$

where $\mathbf{c} = [\underline{c}_1, \ldots, \underline{c}_N]$ is the transmitted codeword, $\mathbf{r} = [\underline{r}_1, \ldots, \underline{r}_N]$ is the corresponding received codeword, C_+^i is the set of all codewords with input label 0 at trellis section i, and C_-^i is the set of all codewords with input label 1 at trellis section i. The output LLR is also referred to as the soft-information or soft-output. A suboptimal implementation of the Log-MAP decoder called the Max-Log-MAP decoder is obtained by using the approximation $\ln(\sum x_i) = \max(\ln(x_i))$ to evaluate the log-APP in (2). Using this approximation, and assuming that all the codewords to be equally likely, the soft-output for codewords transmitted on a additive white Gaussian channel (AWGN) with noise variance $\sigma^2 = N_0/2$ can be written as [5]

$$L(u_i|\mathbf{r}) = \min_{\mathbf{c}\in C_+^i} \left(\frac{\|\mathbf{r} - \mathbf{c}\|^2}{2\sigma^2} \right) - \min_{\mathbf{c}\in C_-^i} \left(\frac{\|\mathbf{r} - \mathbf{c}\|^2}{2\sigma^2} \right), \quad (3)$$

Since the union of C_+^i and C_-^i is the set of all valid codewords, one of the terms in (3) corresponds to the metric for the ML codeword ($\mathbf{c}_{\mathrm{ML}}$). Thus, the reliability which is the magnitude of the soft information in (3) can be expressed as

$$\Lambda_i \triangleq |L(u_i|\mathbf{r})| = \frac{1}{2\sigma^2} \left\{ \|\mathbf{r} - \mathbf{c}_{\mathrm{comp}}^i\|^2 - \|\mathbf{r} - \mathbf{c}_{ML}\|^2 \right\}, \quad (4)$$

where $\mathbf{c}_{\mathrm{comp}}^i$ is the codeword that is closest to the received vector among all the codewords that differ from the ML codeword in the input label for trellis section i. $\mathbf{c}_{\mathrm{comp}}^i$ will also be referred to as the competing path or next best path for trellis section i. Since the distance between $\mathbf{r}$ and the ML codeword is smaller than the distance between $\mathbf{r}$ and any other codeword, the difference in (4) is always positive. Thus, the Max-Log-MAP decoder associates with the ith bit, the minimum difference between the metric associated with the ML path and the metric associated with the best path that differs from the ML path in the input label for trellis section i [5]. A high value of reliability implies that the ML path and the next best path with the opposite input label for bit i are far apart, hence there is a lower probability of choosing the other path and making a bit error. Thus, reliability is a measure of the correctness of the bit decision. This has also been shown via simulation results in [2], [6]. A bit with high reliability is more likely to have decoded correctly than a bit with low reliability. The CB-HARQ scheme depends on the knowledge of $\mathbf{c}_{\mathrm{ML}}$ and $\mathbf{c}_{\mathrm{comp}}^i$. We now present an approach to compute $\mathbf{c}_{\mathrm{ML}}$ and $\mathbf{c}_{\mathrm{comp}}^i$ using the computations already performed in the decoder.

B. Obtaining the ML and competing path using the BCJR algorithm

Following the development in [7], the soft information in (3) can be expressed as

$$L(u_i|\mathbf{r}) = \max_{C_+^i} \Big(\alpha_{i-1}(s') + \gamma_i(s', s) + \beta_i(s) \Big) - \max_{C_-^i} \Big(\alpha_{i-1}(s') + \gamma_i(s', s) + \beta_i(s) \Big), \quad (5)$$

where

$$\alpha_k(s) \triangleq \log(P(s_k = s), \mathbf{r}_1^k), \quad \beta_k(s) \triangleq \log(P(\mathbf{r}_{k+1}^N | s_k = s)),$$
$$\gamma_k(s', s) \triangleq \log(P(s_k = s, \underline{r}_k | s_{k-1} = s')).$$

Note $\mathbf{r}_a^b = [\underline{r}_a, \underline{r}_{a+1}, \ldots, \underline{r}_{b-1}, \underline{r}_b]$. Let $\mathcal{S}(\rightarrow s)$ denote the set of states at time $k-1$ that have branches leading into state s at time k. Similarly, let $\mathcal{S}(s \rightarrow)$ be the set of states at time $k+1$ that have branches emerging from state s at time k. Then it can also be shown that (see [7])

$$\alpha_i(s) = \max_{s'\in\mathcal{S}(\rightarrow s)} (\alpha_{i-1}(s') + \gamma_i(s', s)) \quad (6)$$

$$\beta_{i-1}(s) = \max_{s'\in\mathcal{S}(s\rightarrow)} (\beta_i(s') + \gamma_i(s, s')) \quad (7)$$

$$\gamma_i(s', s) \propto -\|\underline{r}_i - \underline{c}_i\|^2. \quad (8)$$

Thus, it is seen from (8) that $\gamma_i(s', s)$ is proportional to the branch metric (cf. [1]), $P(\underline{r}_i/\underline{c}_i)$, used in the Viterbi algorithm (where the constant of proportionality depends only the signal-to-noise ratio). Also $\alpha_k(s)$ ($s \in \{0, 1, 2, 3\}$ since a rate -1/2 code is used) is initialized to [7]

$$\alpha_0(0) = 0, \ \alpha_0(1) = -\infty, \ \alpha_0(2) = -\infty, \ \alpha_0(3) = -\infty. \quad (9)$$

Thus from (9) and (6), it is seen that $\alpha_i(s)$ is proportional to the partial path metric (cf. [1]) of the surviving path (in the

Viterbi algorithm) at state s in trellis section i. Similarly $\beta_i(s)$ can be shown to be proportional to the partial path metric of the surviving path at state s at time i if the Viterbi algorithm is executed in reverse i.e., starting at the end of the trellis[1].

Let the ordered pair of states (s_{i-1}, s_i) that maximizes the first term in (5) be (s_{i-1}^+, s_i^+). Similarly define (s_{i-1}^-, s_i^-). Also let $s_k(\mathbf{c})$ denote the state in the code trellis that codeword $\mathbf{c}$ passes through at time k. By comparing (3) and (5), it is seen that one of the ordered pairs of states (s_{i-1}^+, s_i^+) or (s_{i-1}^-, s_i^-) corresponds to $\mathbf{c}_{\text{ML}}$, while the other ordered pair corresponds to $\mathbf{c}_{\text{comp}}^i$. For example, if

$$\max_{C_+^i}\left(\alpha_{i-1}(s')+\gamma_i(s',s)+\beta_i(s)\right) > \max_{C_-^i}\left(\alpha_{i-1}(s')+\gamma_i(s',s)+\beta_i(s)\right),$$

then $s_{i-1}(\mathbf{c}_{\text{ML}}) = s_{i-1}^+$, $s_i(\mathbf{c}_{\text{ML}}) = s_i^+$, and $s_{i-1}(\mathbf{c}_{\text{comp}}^i) = s_{i-1}^-$, $s_i(\mathbf{c}_{\text{comp}}^i) = s_i^-$. Thus, when computing soft-output for trellis section i, it is possible to identify the branches through the trellis at time i that correspond to ML path and the competing path.

We now present an approach to compute $\mathbf{c}_{\text{ML}}$ given $s_{i-1}(\mathbf{c}_{\text{ML}})$ and $s_i(\mathbf{c}_{\text{ML}})$. There are two states at time $i-2$ with branches ending at state $s_{i-1}(\mathbf{c}_{\text{ML}})$. One of these branches corresponds to the surviving path i.e., one of these branches corresponds to the codeword $\mathbf{c}$ that minimizes the partial path metric until time $i-1$ ($\|\mathbf{r}_1^{i-1} - \mathbf{c}_1^{i-1}\|^2$). As mentioned earlier, $\alpha_{i-2}(s') + \gamma_i(s', s_{i-1}(\mathbf{c}_{\text{ML}}))$ corresponds to the partial path metric until time $i-1$. Thus, the state at time $i-2$ that maximizes $\alpha_{i-2}(s') + \gamma_i(s', s_{i-1}(\mathbf{c}_{\text{ML}}))$ corresponds to the surviving path. Thus

$$s_{i-2}(\mathbf{c}_{\text{ML}}) = \underset{s\in\mathcal{S}(\rightarrow s_{i-1}(\mathbf{c}_{\text{ML}}))}{\operatorname{argmax}} \left\{\alpha_{k-1}(s) + \gamma_k(s, s_{i-1}(\mathbf{c}_{\text{ML}}))\right\}. \tag{10}$$

Now given $s_{i-2}(\mathbf{c}_{\text{ML}})$, the state at time $i-3$ can be found in a similar manner i.e., by following the branch that maximizes the partial path metric until time $i-2$. This procedure can be repeated until the beginning of the trellis is reached to obtain the surviving path, which in this case is $\mathbf{c}_{\text{ML}}$, until time i.

By a similar argument, $s_{i+1}(\mathbf{c}_{\text{ML}})$ can be obtained by following the path that maximizes the sum of $\gamma_{i+1}(s_i(\mathbf{c}_{\text{ML}}), s) + \beta_{i+1}(s)$. Therefore

$$s_{i+1} = \underset{s\in\mathcal{S}(s_i(\mathbf{c}_{\text{ML}})\rightarrow)}{\operatorname{argmax}} \left\{\gamma_{i+1}(s_i(\mathbf{c}_{\text{ML}}), s) + \beta_{i+1}(s)\right\}. \tag{11}$$

It can be shown that this choice of s_{i+1} corresponds to the surviving path until time i if the Viterbi algorithm is executed in reverse starting at the end of the trellis. The latter part of $\mathbf{c}_{\text{ML}}$ can then be computed by repeating this procedure until the end of the trellis is reached. Similarly, the competing path can be obtained given $s_{i-1}(\mathbf{c}_{\text{comp}}^i)$, and $s_i(\mathbf{c}_{\text{comp}}^i)$.

Observe that the previous state (s_{i-1}) (in (10)) conditioned on the current state s_i is chosen such that it corresponds to the branch involved in computing the alpha for the current state (s_i) (in (6)). Similarly using (7) and (11), it is seen that the next state (s_{i+1}) conditioned on s_i is chosen such that it corresponds to the branch involved in computing the beta for the current state(s_i). This observation enables an efficient modification of the BCJR algorithm that enables computing $\mathbf{c}_{\text{ML}}$ and $\mathbf{c}_{\text{comp}}^i$ for any trellis section i. During the computation of the $\alpha_i(s),\ s\in\mathcal{S},\ i\in\{1,\ldots,N\}$, mark the state $s_{i-1} = s'$ that maximizes $\alpha_{i-1}(s') + \gamma_i(s', s)$ as the previous state for s. Similarly, during the computation of the $\beta_i(s),\ s\in\mathcal{S},\ i\in\{1,\ldots,N\}$, mark the state $s_{i+1} = s'$ that maximizes $\beta_{i+1}(s') + \gamma_i(s, s')$ as the next state for s. Then given the branch in the code-trellis corresponding to $\mathbf{c}_{\text{ML}}$ or $\mathbf{c}_{\text{comp}}^i$ for at time i, the entire codeword can be obtained by following the set of previous states (trace-back), and the set of next states (trace-forward) emerging from the given branch.

Note that if $s_{i-k}(\mathbf{c}_{\text{ML}}) = s_{i-k}(\mathbf{c}_{\text{comp}}^i)$ (or $s_{i+k}(\mathbf{c}_{\text{ML}}) = s_{i+k}(\mathbf{c}_{\text{comp}}^i)$), for some k, then the sequence of state-transitions obtained for any time before (or after) k will be the same for $\mathbf{c}_{\text{ML}}$ and $\mathbf{c}_{\text{comp}}^i$. That is, the ML and the competing path coincide for any $j \neq k$, if $s_{i\pm k}(\mathbf{c}_{\text{ML}}) = s_{i\pm k}(\mathbf{c}_{\text{comp}}^i)$. For the objective of this paper, which is to design an ARQ scheme, it is enough to obtain the ML and competing paths only for the sections where they are different. Thus, the trace-back and trace-forward procedures need not be carried out till the beginning or end of the trellis, but only until $s_{i\pm k}(\mathbf{c}_{\text{ML}}) = s_{i\pm k}(\mathbf{c}_{\text{comp}}^i)$.

We now present an ARQ scheme that uses explicit knowledge of the $\mathbf{c}_{\text{ML}}$ and $\mathbf{c}_{\text{comp}}^i$ to exploit correlated reliabilities.

III. Design of CB-HARQ scheme

In this section, we describe the proposed CB-HARQ scheme that exploits the decoders knowledge of the competing path and the ML path. It is well known that the soft-output/reliabilities of adjacent bits in convolution code are correlated [8]. For the special case of max-log-MAP decoders we found through simulations that groups of neighboring bits have the same reliability. Since $\mathbf{r}$ and $\mathbf{c}_{\text{ML}}$ are the same for all trellis sections, (4) implies that bits decoding with the same reliability should have the same competing paths By using the technique described in the previous section, and observing the competing paths for adjacent bits that decoded with the same reliability, it is verified that the competing paths are the same for those bits. Thus, for max-log-MAP decoders, the strong correlation between the reliabilities of adjacent bits is reflected in the the choice of the same competing path in the code-trellis for those bits.

In the reliability-based HARQ scheme proposed in [2], the receiver sorts the bits according to the reliability whenever there is a packet error, and requests for information about a fraction of the least reliable bits (LRBs). Note, that the retransmission set will consist of groups of neighboring bits with the same reliability value (because they all have the same choice of the ML and competing paths). Assuming that the LRBs are in error, it is not necessary to provide retransmissions for all of the neighboring LRBs. For example, assume four neighboring bits with the same reliability are

[1]This is true only for terminated convolutional codes

present in the set of LRBs. These bits will decode incorrectly if the decoder chose the wrong path as the ML path for these bits. If retransmitting the coded symbols for one of those bits corrects it, that will change the ML path and competing path involved in reliability calculation for that bit. Since the ML and competing paths for the adjacent bits are likely to be the same, they are likely to be corrected too. Therefore, it is not required to provide information for all the neighboring bits. It is enough to provide the minimum amount of information that will change the decision about $\mathbf{c}_{\text{ML}}$, thereby potentially correcting all bits that chose the same ML and competing paths. This idea is formalized in the sequel.

A. Estimation of retransmission size

Given the reliability of an LRB, the decoder needs to estimate the amount of information to be requested to correct the bit (the decoder assumes that the bit has decoded incorrectly). We begin by defining an error event.

$\Diamond$ *Definition 1. Error Event* $\mathbf{e}^i$: The event that separates $\mathbf{c}_{\text{ML}}$ and $\mathbf{c}^i_{\text{comp}}$ is called an error event.

$$\mathbf{e}^i = \mathbf{c}_{\text{ML}} \oplus \mathbf{c}^i_{\text{comp}},$$

where $\oplus$ represents the XOR (addition or subtraction in a binary field) operator. For linear convolutional codes considered in this paper, $\mathbf{e}^i$ is a codeword. Since $\mathbf{e}^i$ represents the difference between $\mathbf{c}_{\text{ML}}$ and $\mathbf{c}^i_{\text{comp}}$ it is referred to as an error event.

The reliability in (4) can be further simplified as

$$\Lambda_i = \frac{1}{\sigma^2}\mathbf{r}^{\text{T}} \cdot (\mathbf{c}_{\text{ML}} - \mathbf{c}^i_{\text{comp}}). \tag{12}$$

If bit i is an LRB, then the decoder assumes that it is in error, and tries to estimate the amount of information required to correct this error. There is an error if the codeword ($\mathbf{c}_{\text{ML}}$) that is closest to the received vector ($\mathbf{r}$) is not the true transmitted codeword. If the ML path is not the true codeword, then it is very likely that the competing path is the correct codeword. So the decoder tries to estimate the amount of information required to change the decision from the ML path to the competing path (assuming that this will correct the error).

If $\mathbf{c}_{\text{ML}}[k] = \mathbf{c}^i_{\text{comp}}[k]$[2], it implies that the competing path and the ML path have made the same decision about parity bit k. Thus, requesting additional transmissions of parity bit k will not be helpful because the decoder already has sufficient information to decode parity bit k correctly. The decoder will benefit by receiving additional transmissions for only those parity bits for which the decisions of ML and competing codeword are different. In other words, we want to have retransmissions for only those parity bits that potentially decoded incorrectly.

$\Diamond$ *Definition 2. Candidate set of parity bits* $\mathbf{S}_i$ for trellis section i: The set of parity bits for which the decisions of the ML codeword ($\mathbf{c}_{\text{ML}}$) and competing codeword ($\mathbf{c}^i_{\text{comp}}$) are different.

$$\mathbf{S}_i = \{k : \mathbf{c}_{\text{ML}}[k] \neq \mathbf{c}^i_{\text{comp}}[k]\} = \{k : \mathbf{e}^i[k] = 1\} \tag{13}$$

[2] $\mathbf{x}[l]$ refers to component l of vector $\mathbf{x}$

Once the candidate set of parity bits are obtained, the decoder tries to estimate the number of parity bits from the candidate set $\mathbf{S}_i$ that have to be re-transmitted in order for the decoder to decide in favor of $\mathbf{c}^i_{\text{comp}}$ instead of $\mathbf{c}_{\text{ML}}$.

Let $\mathbf{r}^*$ be the received vector after retransmitting κ coded symbols[3]. The decoder tries to estimate the minimum number of coded symbols retransmitted (k) that can change the decision from $\mathbf{c}_{\text{ML}}$ to $\mathbf{c}^i_{\text{comp}}$. That is, after retransmission we require

$$\|\mathbf{r}^* - \mathbf{c}^i_{\text{comp}}\|^2 < \|\mathbf{r}^* - \mathbf{c}_{ML}\|^2 \tag{14}$$

$$\text{i.e., } \|\mathbf{r}^* - \mathbf{c}^i_{\text{comp}}\|^2 - \|\mathbf{r}^* - \mathbf{c}_{ML}\|^2 < 0 \tag{15}$$

$$\text{i.e., } 2\mathbf{r}^{*\text{T}} \cdot (\mathbf{c}_{\text{ML}} - \mathbf{c}^i_{\text{comp}}) < 0 \tag{16}$$

$$\text{i.e., } 2\mathbf{r}^{\text{T}} \cdot (\mathbf{c}_{\text{ML}} - \mathbf{c}^i_{\text{comp}}) + \sum_{\substack{l \in \eta \\ \eta \subset \mathbf{S}_i, |\eta| = \kappa}} 2\mathbf{r}^{'}[l](\mathbf{c}_{\text{ML}}[l] - \mathbf{c}^i_{\text{comp}}[l]) < 0, \tag{17}$$

where η is the subset of the candidate set that has been re-transmitted, and $\mathbf{r}^{'}$ corresponds to the symbols received due to the re-transmission i.e., $\mathbf{r}^* = \mathbf{r} + \mathbf{r}'$ (assuming BPSK and transmission over an AWGN channel). Using (12), we obtain $2\mathbf{r}^{\text{T}} \cdot (\mathbf{c}_{\text{ML}} - \mathbf{c}^i_{\text{comp}}) = 2\sigma^2\Lambda_i$, where Λ_i is the reliability of trellis section i after the first transmission. Note that in the above equations $\mathbf{c}_{\text{ML}}$ and $\mathbf{c}^i_{\text{comp}}$ refer to the ML path and competing path encountered in computing the soft-output for trellis section i before retransmission of additional coded symbols.

The decoder assumes that the parity bits in the candidate set are in error. Therefore, assuming that the all-zeros[4] CW has been transmitted, then $\mathbf{c}^i_{\text{comp}}(l) = 1$ and $\mathbf{c}_{\text{ML}}(l) = -1$, $\forall l \in \mathbf{S}_i$. Since the all-zeros CW is the true transmitted codeword, $\mathbf{r}'(l) \sim \mathcal{N}(1, \sigma^2)$. Thus,

$$X_i \triangleq \sum_{\substack{l \in \eta \\ \eta \subset \mathbf{S}_i, |\eta| = \kappa}} 2\mathbf{r}^{'}[l](\mathbf{c}_{\text{ML}}[l] - \mathbf{c}^i_{\text{comp}}[l]) \sim \mathcal{N}(-4\kappa, 16\kappa\sigma^2).$$

Thus according to the decoder, after the first-retransmission, correct decoding is made if (using (17))

$$X_i < -2\sigma^2\Lambda_i \tag{18}$$

The decoder estimates the number of coded bits (κ) that have to be re-transmitted as follows

$$\min_{\kappa} P(X_i < -2\sigma^2\Lambda_i) \geq \Theta \tag{19}$$

$$\implies \min_{\kappa} Q\left(\frac{\sigma^2\Lambda_i - 2\kappa}{2\sqrt{\kappa\sigma^2}}\right) \geq \Theta, \tag{20}$$

where κ is the number of parity bits re-transmitted and Θ is a pre-defined threshold. Thus, the decoder estimates the number of bits to be re-transmitted as the minimum number that would cause the decoder to decide in favor of $\mathbf{c}^i_{\text{comp}}$ instead of $\mathbf{c}_{\text{ML}}$ with a probability that is at least Θ. The decoder

[3] $\mathbf{r}^*$ is obtained by combining the original received vector $\mathbf{r}$ and the additional symbols using maximal-ratio combining.

[4] Since convolutional codes are linear codes, this analysis can be carried out for the special case of transmission of the all-zeros CW

assumes that once the decision is reversed, the error will be corrected. $P(X_i < -2\sigma^2\Lambda_i)$ will be referred to as the *correction probability after re-transmission* (P_r). Thus, the receiver requests the minimum number of coded bits such that P_r exceeds Θ. Thus, by requesting for κ parity bits, all the bits that had originally decoded with the same ML and competing paths are corrected with a probability that is greater than Θ.

B. Estimation of the retransmission set

After the decoder estimates the number of parity bits required (κ) from the candidate set, it needs to decide which κ of the parity bits in $\mathbf{S}_i$ should be requested for re-transmission. We use estimates of the instantaneous SNRs of the different trellis sections involved in the error event $\mathbf{e}_i$ that separates $\mathbf{c}_{\text{ML}}$ and $\mathbf{c}^i_{\text{comp}}$ to decide the re-transmission set. The receiver sorts the trellis sections in the error-event according to the instantaneous SNRs, and requests κ parity bits from the trellis sections with low SNRs.

Note that the absolute value of a received symbol provides an estimate of the instantaneous SNR for the corresponding parity bit. We define the instantaneous SNR of a trellis section as follows. If for a particular trellis section i, $\underline{c}_{ML}$ and $\underline{c}^i_{\text{comp}}$ differ in only one parity bit, then the instantaneous SNR of that section is equal to the absolute value of the received symbol corresponding to that parity bit. If for a particular trellis section i, $\underline{c}_{ML}$ and $\underline{c}^i_{\text{comp}}$ differ in both parity bits, then the instantaneous SNR of the trellis section is a function of the instantaneous SNRs of the two parity bits. In this paper, we provide results for three different functions that can be used to define the instantaneous SNR of trellis sections for which the ML path and the competing path differ in both parity bits. The instantaneous SNR for such a trellis section can be defined as the average of the absolute values of the two received parity symbols, or as the minimum of the absolute value of the two parity symbols, or by randomly assigning the absolute value of one of the parity symbols as the instantaneous SNR of the corresponding trellis section. The instantaneous SNR of a particular trellis section for different output labels on $\mathbf{c}_{\text{ML}}$ and $\mathbf{c}^i_{\text{comp}}$ is given in Table I when the SNR of a trellis section is defined as the average of the SNRs of the corresponding coded symbols that belong to the candidate set. Note that all possible output labels can be obtained by interchanging the output labels on the ML and competing paths in each row of Table I. The receiver selects κ parity bits corresponding to trellis sections with the lowest SNRs from the candidate set.

C. Suboptimal schemes

In Section IV it is shown that the throughput of CB-HARQ is better than the throughput of the RB-HARQ proposed in [2] if the size of the feed-back packet is not taken into account. The throughput obtained by ignoring the size of the feedback packet will be referred to as the throughput with zero-cost for feedback. However, in any practical network, the feedback packets constitute actual traffic, and hence will decrease the overall throughput. It is also shown in Section IV that the throughput of CB-HARQ after taking into account the size of the feedback packet is worse than the performance of RB-HARQ at high SNRs. At low SNRs, the throughput is better for the CB-HARQ scheme. The loss in throughput can be attributed to the high overhead required to index each parity bit to be requested. In RB-HARQ, for each bit index requested, two parity bits are re-transmitted. However, in CB-HARQ scheme, for each bit index, there is the potential to request only one of the parity bits. Thus, if we fix the number of parity bits that have to be re-transmitted in each iteration, then there is the potential to feedback more bit indices in the improved scheme when compared to RB-HARQ thereby decreasing the throughput.

TABLE I

INSTANTANEOUS SNR ESTIMATION FOR TRELLIS SECTIONS BASED ON THE AVERAGE OF THE INSTANTANEOUS SNRS OF THE PARITY BITS IN THE CANDIDATE SET

Output label on trellis section i for $\mathbf{c}_{\text{ML}}$	Output label on trellis section i for $\mathbf{c}^i_{\text{comp}}$	Estimate of the instantaneous SNR for trellis section i
$1\ 1$	$-1-1$	$(\lvert r_i^0\rvert + \lvert r_i^1\rvert)/2$
$-1\ 1$	$1-1$	$''$
$-1\ 1$	$1\ 1$	$\lvert r_i^0\rvert$
$-1\ 1$	$-1-1$	$\lvert r_i^1\rvert$

Thus, to improve the throughput, we need to compress the size of the feedback packet. In CB-HARQ scheme, $\log_2 N$ bits are required to index each trellis section, and two additional bits are needed to index the parity bit to be requested ($\underline{c}_i^0$ or $\underline{c}_i^1$ or both). We compress the feedback packet by the following sub-optimal approach. We form the candidate set as in (13), but by choosing parity bits from only the first L trellis sections of the error event. We then index each trellis section differentially with respect to the first trellis section where $\mathbf{c}_{\text{ML}}$ and $\mathbf{c}^i_{\text{comp}}$ diverge. Thus, $log_2 N$ bits are required to index this trellis section. However, the other trellis sections required only $log_2 L$ bits to indicate their distance (in time) from the origin of the error event. Note that two additional bits are still required to indicate which parity bit from each section is to be requested.

IV. RESULTS

In this section, we provide simulation results to evaluate the performance of CB-HARQ scheme proposed in this paper. The simulation setup is identical to the the scenario studied in [2]. The source encodes the a block of 1000 information bits using a rate $1/2$ convolutional code with generator polynomials $1 + D^2$ and $1 + D + D^2$ ($(5, 7)$ in octal notation). The code is then punctured down to rate $4/7$ using a puncturing pattern given in [9]. If the destination fails to decode correctly, it sends a retransmission-request packet containing a list of trellis sections, and the corresponding code symbols in those sections that have to be re-transmitted. In [2], code symbols are requested for 25 LRBs i.e, 50 code symbols are re-transmitted by the source. To keep the comparison fair, the destination in our work also sends a request for 50 code symbols. However, these bits need not correspond to 25 trellis sections since only one of the coded bits is requested for some trellis sections. The

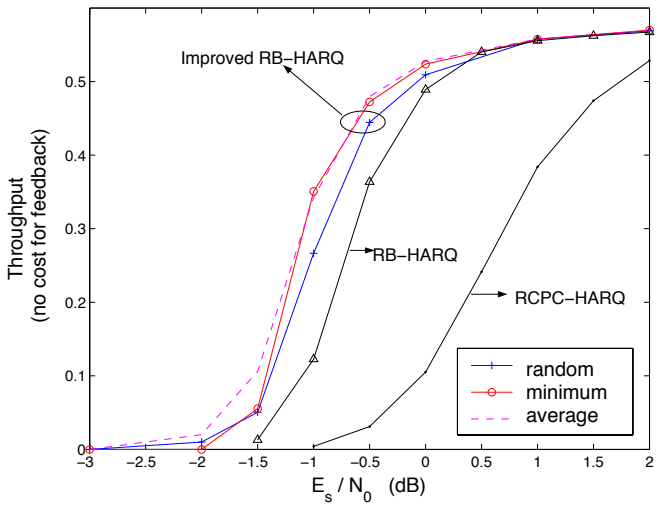

Fig. 2. Throughput for the CB-HARQ scheme using three different approaches to choose the parity bits from the candidate set. For these results, the throughput was calculated without considering the size of the feedback packet required to index the parity bits that have to be re-transmitted.

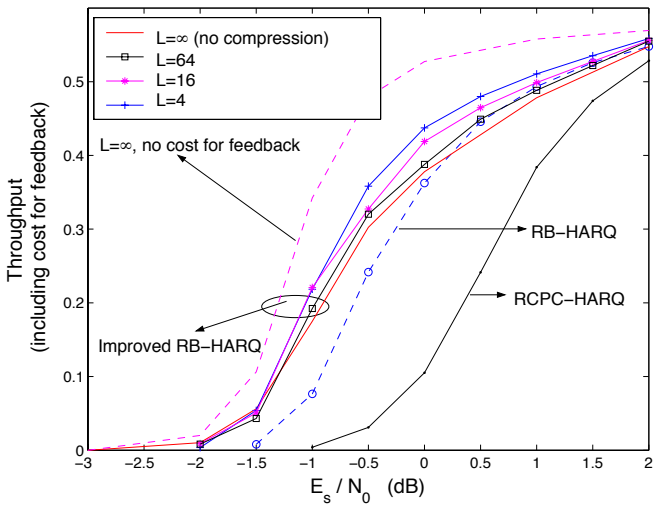

Fig. 3. Throughput for the CB-HARQ scheme using three different approaches to choose the parity bits from the candidate set. The throughput shown in this figure includes the overhead due to the size of the feedback packet.

destinations starts with the LRB, identifies the corresponding error event, and the candidate set. It then requests a number of code symbols depending on its estimate of the amount of information required to make P_r exceed a threshold of $\Theta = 95\%$. The destination then looks at the next LRB. If this bit has the same ML and competing paths as one of the previous LRBs no information is requested. This is because the previously requested information has the ability to correct all the bits that have the same competing path (error event). The decoder proceeds till a total of 50 parity bits are requested for a set of different error events. The code symbols from the re-transmission are combined with all previous copies of the received information, and decoding is performed. If the packet does not decode correctly, another iteration of requests and re-transmission is performed. We allow a total of 5 iterations with 50 parity bits being re-transmitted in each iteration. Thus, in 5 iterations the code rate reduces from $4/7$ to $1/2$.

The throughput of CB-HARQ with zero-cost for feedback is shown in Figure 2 for various functions used to define the instantaneous SNRs of a trellis section. The performance of RB-HARQ and HARQ based on RCPC [9] is also shown. In RCPC based HARQ, all the punctured coded bits are re-transmitted whenever a packet decodes in error. It is seen that all the CB-HARQ schemes perform better than RB-HARQ. It is also seen that randomly choosing the parity bits from the candidate set is not a good strategy. The performance of choosing parity bits based on the average or minimum of the instantaneous SNRs of the individual parity bits in the candidate set that belong to a particular trellis section is almost the same. However, since using the average performs slightly better at low SNRs we use this approach in our sub-optimal schemes. The results in Figure 3 show that in the high SNR region ($E_s/N_0 > 0$ dB), the CB-HARQ scheme without compression performs worse than the RB-HARQ scheme. This is because the throughput shown in Figure 3 takes into account the size of the feedback packet. The size of the feedback packet for CB-HARQ scheme can become bigger than that for the RB-HARQ scheme leading to a loss in throughput. This motivates compressing the feedback packet.

The throughput performance of sub-optimal schemes for three different values of L (see Section III-C) is shown in Figure 3. The packet error rate (PER) for these schemes is shown in Figure 4. It is observed that as L decreases, the PER increases at low SNRs, but remains the same at high SNRs. At low SNRs, longer error events occur more frequently, and restricting the candidate set to the first L sections eliminates

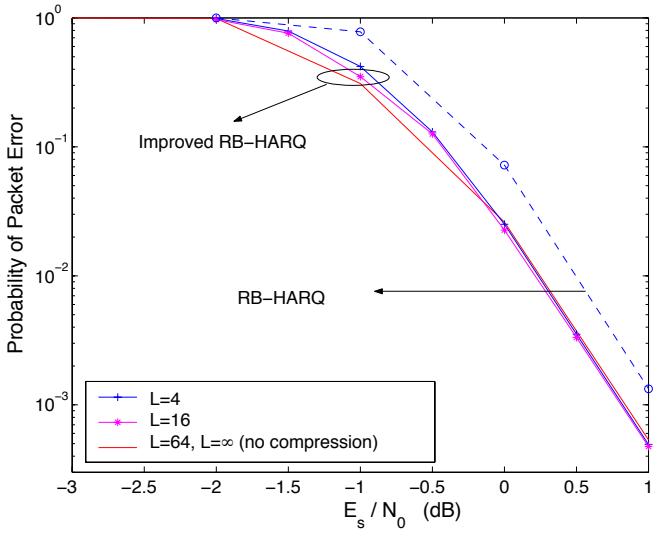

Fig. 4. Probability of packet error for the sub-optimal CB-HARQ scheme.

some sections that can potentially benefit from retransmissions. This degrades performance, but the performance loss is not significant. At high SNRs, the performance is dominated by very short events (the minimum distance event spans three trellis sections for the $(5, 7)$ code), and hence there is no loss even when $L = 4$. The small degradation in PER is well compensated by the increase in throughput caused by the reduction in the size of the feedback packet. With $L = 4$, the throughput of the CB-HARQ scheme is larger compared to RB-HARQ at all SNRs. For instance, the throughput increases by 180% at $E_s/N_0 = -1$ dB.

V. Concluding Remarks

In this paper, a novel HARQ scheme that utilizes the correlated error events in a soft-input soft-output decoder is presented. In the proposed CB-HARQ technique, the decoder first analytically estimates the minimum number of parity bits that have the potential to correct an entire error event with a certain probability. Then the decoder requests for retransmissions of these parity bits. By targeting error events instead of individual bits, this approach has the ability to correct multiple neighboring bits that decode incorrectly. Thus, the error event corresponding to the ML and competing paths in a max-log-MAP decoder provides a convenient approach to exploit the correlated nature of the reliabilities. It shown through simulation results that this approach outperforms RB-HARQ and conventional incremental redundancy HARQ at all SNRs, and has the potential to increase the throughput by 180%.

Acknowledgments

This work was supported by the National Science Foundation under Grant ANI-0220287 and by the DoD Multidisciplinary University Research Initiative administered by the Office of Naval Research under Grant N00014-00-1-0565.

References

[1] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*. Prentice Hall, 1983.

[2] A.Roongta and J. M. Shea, "Reliability based hybrid ARQ using convolutional codes," in *Proc. Int. Conf. Commun. (ICC)*, (Anchorage, AK), pp. 2889 –2893, May 2003.

[3] A. Roongta and J. M. Shea, "Reliability-based hybrid ARQ and rate-compatible punctured convolutional (RCPC) codes," in *Proc. of the 2004 IEEE Wireless Communications and Networking Conference*, (Atlanta, GA), 2004.

[4] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rates," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.

[5] M. P. C. Fossorier, F. Burkert, S. Lin, and J. Hagenauer, "On the equivalence between SOVA and max-log-MAP decodings," *IEEE Commun. Letters*, vol. 2, pp. 137–139, May 1998.

[6] J. M. Shea, "Reliability-based hybrid ARQ," *IEE Electronics Letters*, vol. 38, pp. 644–645, June 2002.

[7] W. E. Ryan, "*Concatenated Codes and Iterative Decoding*" in *Wiley Encyclopedia of Telecommunications* (J. G. Proakis ed.). New York: Wiley and Sons, 2003.

[8] L. Reggiani and G. Tartara, "Probability density functions of soft information," *IEEE Commun. Letters*, vol. 6, pp. 52–54, Feb. 2002.

[9] J. Hagenauer, "Rate-compatible punctired convolutional codes (RCPC codes) and their applications," *IEEE Trans. Commun.*, vol. 36, pp. 389–400, Apr. 1988.