In the Proceedings of the 2001 IEEE Global Communications Conference, November 2001
REDUCED-COMPLEXITY DECODING FOR CONCATENATED CODES BASED ON
RECTANGULAR PARITY-CHECK CODES AND TURBO CODES

John M. Shea and Tan F. Wong
University of Florida
Department of Electrical and Computer Engineering
Gainesville, Florida

Abstract—In this paper, we compare the performance of several different decoding strategies for concatenated codes based on a serial concatenation of a rectangular parity-check code (RPCC) with a turbo code. These concatenated codes are referred to as RPCC+turbo codes. RPCC+turbo codes have been shown to significantly outperform turbo codes in several scenarios [1],[2]. One particularly useful application is to replace a turbo code with an RPCC+turbo code based on constituent codes of smaller memory. This combination can provide comparable or better performance while also achieving a lower decoder complexity. However, the complexity of the iterative MAP decoder for such a code is still relatively high. In this paper, we compare several different decoding strategies for the RPCC+turbo code that offer various trade-offs between performance and complexity. In this paper, we compare the performance of several different decoding strategies for concatenated codes based on a serial concatenation of a rectangular parity-check code (RPCC) with a turbo code. These concatenated codes are referred to as RPCC+turbo codes. RPCC+turbo codes have been shown to significantly outperform turbo codes in several scenarios [1],[2]. One particularly useful application is to replace a turbo code with an RPCC+turbo code based on constituent codes of smaller memory. This combination can provide comparable or better performance while also achieving a lower decoder complexity. However, the complexity of the iterative MAP decoder for such a code is still relatively high. In this paper, we compare several different decoding strategies for the RPCC+turbo code that offer various trade-offs between performance and complexity.

I. INTRODUCTION

The performance of turbo codes [3] is often limited by low-weight error events [4]. This is particularly true of simple turbo codes, such as those based on constituent codes of memory two. These low weight-error events are responsible for the asymptotic performance of turbo codes. This asymptote causes an “error floor” in the performance of the turbo code, beyond which the error probability decreases very slowly as the energy-to-noise density ratio is increased. This error floor tends to occur at high error probabilities when the turbo code is based on very simple constituent codes and when the block length is short (several hundred to several thousand bits). For instance, the turbo codes for the cdma2000 and WCDMA third-generation cellular systems are based on constituent codes of memory three because the error floor for the memory-two code is much higher than that for the memory-three code when the block length is short.

For wireless communication systems, short block lengths and simple constituent codes are usually required. One motivation for these requirements is to keep the decoder simple enough to implement in a cost-effective manner. The complexity of the decoder has currently limited the application of turbo codes to the reverse link, so that the turbo decoder is implemented in hardware at the base station. In addition, several authors have shown that turbo codes can be used in hybrid-ARQ schemes that use code combining [5]–[13], but these methods have not been implemented in any of the standards. One probable reason for this is that the previously proposed code-combining ARQ techniques all require additional iterative MAP decoding of the entire packet when additional code symbols are received. This additional iterative decoding results in significantly higher processing requirements and longer delays.

Thus, it is desirable to consider other code structures that can provide similar or better performance than turbo codes while also reducing the complexity. In this paper, we present results for serial-concatenated codes that use a rectangular parity-check code (RPCC) as the outer code and a turbo code as the inner code. We refer to these codes as RPCC+turbo codes. The RPCC+turbo codes have several features that make them particularly useful in wireless communication systems. The first is that they typically perform better than using only a turbo code, but the additional decoder complexity is small compared to the complexity of the turbo decoder. Also, the rate reduction is small from using the RPCC+turbo code instead of only the turbo code because the rate of the RPCC is very high (typically greater than 0.9). The performance of the RPCC+turbo code is very good even when the turbo code is constructed from very simple constituent codes. These advantages can be combined to replace a turbo code with a code that performs better while also requiring lower decoder complexity. In addition, we propose a code-combining ARQ scheme that takes advantage of the structure of the RPCC+turbo code to improve performance without requiring additional iterative decoding.

For turbo codes to be used between wireless devices or on the forward-link of a cellular communication system, the
complexity and delay of the turbo decoder must be kept low. Thus in this paper, we examine the performance of RPCC+turbo codes when the number of decoding iterations is limited to a small number. We also present results that compare the performance of turbo and RPCC+turbo codes with several decoding schemes that offer different trade-offs between performance and complexity.

II. SYSTEM DESCRIPTIONS

Consider first a packet communications system that uses rate 1/3 turbo coding. Let $B$ denote the block length (the number of input bits that are input to the code to create a codeword). For the purposes of this paper, we assume that $B$ is the square of an integer $b$. A new concatenated code of slightly lower rate than the turbo code is created by serially concatenating a rectangular parity-check code as an outer code with a rate 1/3 turbo code as an inner code, as shown if Figure 1. The rectangular parity-check code used in this paper is the even single parity-check code, where the parity is calculated on the $b$ rows and $b$ columns of a square array of $B$ information bits. The RPCC has rate $b/(b+2)$. The parity-check code and turbo code are separated by an interleaver, which may be pseudo-random or structured in nature. In this paper, we present results using random interleavers. The encoded packet is transmitted over a channel using binary phase-shift-keying (BPSK). The received symbols are corrupted with additive white Gaussian noise (AWGN).

III. DECODING ALGORITHMS

Several decoding algorithms are considered in this paper, along with variations to the number of iterations for some of these algorithms. We separate the decoding algorithms into two classes. Overall-iterative decoders iterate back and forth between the turbo decoder and the RPCC decoder. These decoders are the natural extension of iterative decoders for other parallel- and serial-concatenated codes. When the individual decoders are maximum a posteriori (MAP) decoders, the performance of these codes approaches that of the maximum likelihood decoder $[1,2]$ as the energy-to-noise density ratio is increased.

Overall-noniterative decoders do not iterate between the decoder for the turbo code and the decoder for the RPCC code. Iterative decoding may still take place within the turbo decoder and also within the RPCC decoder. However, in overall-noniterative decoding, there is never an exchange of information from the RPCC decoder to the turbo decoder. Overall-noniterative decoders typically require less complexity and storage than overall-iterative decoders and are particularly appropriate for use with some code-combining ARQ schemes. For instance, if the information bits are sent encoded by only the turbo code sent at first and iterative decoding fails, then the parity-check information can be sent and used to correct errors without further iterations of the iterative turbo decoder. As we show in Section V overall-noniterative decoders sacrifice performance in comparison to overall-iterative decoders, but in comparing the performance to turbo codes, the gains of using RPCC+turbo code with these decoders can still be significant.

A. Overall-Iterative MAP Decoding

The overall-iterative MAP decoder is designed to approximate the performance of the maximum a posteriori (MAP) decoder for the overall code. The decoder iterates between MAP decoders for the recursive constituent codes that make up the turbo code and MAP decoders for the parity-check code in each dimension. This is an extension of the turbo decoding techniques described in $[3]$ and the decoding techniques described in $[14]$ for rectangular parity-check codes. When decoding a particular code, extrinsic information from all of the other codes is used. The RPCC code is treated as a parallel concatenation of parity-check codes, each defined by the parity check bits along one dimension and all the data bits. For each component code, the “soft-in-soft-out” decoding module suggested in $[14]$ is employed.

B. Overall-Noniterative MAP Decoding

In this decoding technique, iterative MAP decoding is employed for the turbo code, and the soft outputs of the turbo code are the inputs to an iterative MAP decoder for the RPCC code. There is no further exchange of information between the turbo decoder and the RPCC decoder. When the parity-check bits are turbo-coded along with the other information bits and transmitted over the channel simultaneously, there is little to be gained in terms of complexity or implementation in using the overall-noniterative MAP decoder instead of the overall-iterative MAP decoder. The main implementation aspect that is improved is the storage requirement for the extrinsic information exchanged between the decoders. However, for these types of applications, the performance of the overall-noniterative MAP decoder provides some guidance as to what kind of performance can be expected from a reduced-complexity overall-noniterative decoding scheme, such as the one described in the next section.

The other real value of the overall-noniterative decoding scheme is when the RPCC+turbo codes are used in code-combining ARQ transmissions. With the exception of $[11]$, most previously proposed code-combining ARQ techniques $[5]–[13]$ are the extension to turbo codes of techniques that have been previously proposed for convolutional codes. These code-combining ARQ techniques require additional iterative
MAP decoding of the turbo code each time that additional parity information is received.

The RPCC+turbo code can be employed for code-combining ARQ in the following way. The information is turbo coded and sent in the usual way. We assume that an error detection code is used, and if the packet fails to decode correctly, then a negative acknowledgment is sent back to the original transmitter. The transmitter then encodes the information bits with a RPCC and “piggybacks” those bits in its next turbo-encoded information packet to the receiver. The receiver can then use either an overall-iterative decoder or an overall-noniterative decoder to decode the packets.

C. Overall-Noniterative MAP/Simple Decoding

As in the previous decoding technique, iterative MAP decoding is employed for the turbo code, and the soft outputs are the inputs to the decoder for the RPCC code. However, the RPCC decoder that is used is known as the simple decoder and is a fast, noniterative pseudo-soft-decision decoder. The first step for the simple decoder is to place the hard-decision values for the information bits from the output of the turbo decoder into a square array. The simple decoder then calculates the horizontal and vertical parity bits based on these hard decisions. These calculated parity vectors are then added modulo two to the hard-decisions for the horizontal and vertical parity vectors from the output of the turbo decoder. We assume that even parity is used. Then the resulting vectors are 1 in any rows that are estimated to contain an odd number of errors and 0 in rows that are estimated to contain no errors or an even number of errors. The simple decoder counts the number of 1s in the horizontal and vertical directions and then tries to determine the positions of symbols that are in error at the output of the turbo code by examining the soft-decision values in the rows and columns in which errors are indicated.

The simple decoder that is used for the results in this paper operates in the following way. If the number of errors indicated by the row and column parities is equal, then for each row, the decoder changes the hard-decision of the least-reliable symbol in any of the columns for which errors are indicated. If the number of errors indicated by the row and column parities is not equal, then the decoder chooses the set (of rows or columns) that indicates the most parity errors and ignores the information from the set that indicates the least number of parity errors. This is best illustrated by considering an example. If the number of errors indicated by the row parities is two and the number of errors indicated by the column parities is zero, then the decoder operates on each of the rows for which errors is indicated. The decoder will find the least-reliable symbol in each of those rows and changes the hard-decision value for that symbol.

IV. PERFORMANCE EVALUATION

When overall-iterative MAP decoding is employed, the performance of these concatenated coding systems can be approximated through the use of their input-output weight enumerating functions (IOWEFs), as described in [15] and [16]. The IOWEF for the recursive convolutional codes can be found using computer searches or transfer function techniques, and the IOWEF for the turbo code can be calculated from these as described in [15]. These bounds indicate the performance of the codes if maximum-likelihood (ML) decoding is employed. When overall-iterative MAP decoding with a sufficiently high number of iterations is employed, the performance approaches that predicted for the ML decoder. However, in this paper, we focus on suboptimal decoding algorithms that do not produce error probabilities close to the bounds. We do present some analytical results for comparison purposes. Some of the details of the analysis are included in [2]. The rest of the results in this paper come from Monte Carlo simulations.

V. PERFORMANCE RESULTS

In this section, we present some initial results on the performance of the various decoding algorithms. The first present results on overall-iterative MAP decoding of turbo and RPCC+turbo codes. The number of iterations for each decoder is set large enough so that little further improvement can be gained from additional iterations. Typically, this is around fifteen to twenty iterations. However, some improvement in performance of the RPCC+turbo codes may be seen up to 100 iterations. The results in Figure 2 illustrate the probability of block (codeword) error for the turbo code and RPCC+turbo code for a block length of $10^4$ bits with full overall-iterative decoding. The turbo code in each case is constructed from identical constituent recursive convolutional codes with feedforward polynomial $1 + D^2$ and feedback polynomial $1 + D + D^2$. We will refer to this convolutional code as the 5/7 code, which is octal notation for its code polynomials expressed as in feedforward/feedback form. The RPCC+turbo code clearly provides superior performance over a broad range of $E_b/N_0$. The rate of the RPCC+turbo code in this case is 0.327, and the rate of the regular turbo code is 1/3. The rate of the RPCC+turbo code can be increased to 1/3 through puncturing with little effect on performance. The analytical results for the RPCC+turbo code are not shown because including any of the meaningful points of the union bound would require too much compression of the axis for the probability of block error. The first meaningful error probability given by the union bound is around $10^{-11}$ at $E_b/N_0 = 1.6 \, \text{dB}$.

As previously mentioned, if turbo-type codes are going to see wide deployment in wireless systems, the decoder complexity must be reduced while maintaining performance. For the results in Figure 2, we consider a block of 900 information bits with the number of decoder iterations limited to five. The results in Figure 3 illustrate the probability of block error, $P_B$, for the turbo code and the RPCC+turbo code. The turbo code in each case is constructed from identical 5/7 recursive convolutional codes. Even under these conditions of smaller block size and limited decoder iterations, the RPCC+turbo code provides significant advantages over using only the turbo code. This is especially true if low block error probabilities are desired. For $P_B = 10^{-3}$, the required bit energy-to-noise density ratio for the RPCC+turbo code is almost 1.3 dB less than for the turbo code. We also observe that puncturing the RPCC+turbo code to rate 1/3 requires approximately 0.1 dB.
higher $E_b/N_0$ to achieve the same $P_B$ as the unpunctured RPCC+turbo code.

For larger packets, the RPCC+turbo code can provide an even more significant advantage over turbo codes if additional iterations of the RPCC decoder are used in place of turbo decoder iterations. In Figure 4, we show the performance of RPCC+turbo codes and turbo codes with three to six decoder iterations. We compare the performance of a turbo code to an RPCC+turbo code that is decoded with one fewer turbo decoder iteration but five additional RPCC decoder iterations. In other words, the RPCC decoder is run one time for each turbo decoder iteration and then is executed five additional times after the last turbo decoder iteration. This comparison is reasonable because the complexity of the turbo decoder is at least an order of magnitude more than that of the RPCC decoder. The results show that the RPCC+turbo code performs significantly better than the turbo code with an equivalent number of decoder iterations over almost the entire useful range of $E_b/N_0$. The turbo code with four decoder iterations, requires more than 1.1 dB higher $E_b/N_0$ to achieve a block error probability of $10^{-3}$ than a RPCC+turbo code with three turbo decoder iterations and five additional RPCC decoder iterations. The required $E_b/N_0$ is more than 1.6 dB lower for the RPCC+turbo code with five turbo decoder iterations than the turbo code with six decoder iterations.

The results in Figure 5 illustrate the probability of block error of the rate 1/3 turbo code employed in both WCDMA and cdma2000. This code is hereafter referred to as the 3GPP turbo code because it is used by both of the third-generation partnership projects. We note, however, that the interleaver we use for the results presented here is a random interleaver...
Instead of one of the algorithmic interleavers specified by the standards. The 3GPP turbo code is constructed from constituent recursive convolutional codes of memory three with feedforward polynomial $1 + D + D^3$ and feedback polynomial $1 + D^2 + D^3$. The results in Figure 5 also illustrate the block error probability for two RPCC+turbo codes that based on identical 5/7 constituent codes of memory two. For all of the results, overall-iterative decoding is employed, and the maximum number of decoder iterations is five.

The results in Figure 5 show that the RPCC+ turbo code that is based on a simpler constituent code can yield performance comparable to the more-complicated turbo code and can even yield lower error probabilities at high $E_b/N_0$. The decoder for the RPCC+ turbo code is much simpler than the decoder for the 3GPP turbo code because the majority of the complexity is in the turbo decoder. The 3GPP constituent codes have memory three and thus have twice the number of states as the memory-two 5/7 codes. Therefore, the RPCC+turbo code can give us a way to replace a more-complicated code with a simpler code while achieving comparable performance. If the block length is increased, the performance of the RPCC+turbo code improves more quickly than turbo code alone, which makes the RPCC+turbo code an even more attractive option.

The decoding algorithms could be further simplified if the overall-iterative decoding scheme were replaced with an overall-noniterative scheme. If the simple decoder described in Section III is used, very little additional processing is required in comparison to the amount of processing required by the turbo decoder. The results in Figure 5 compare the probability of block error for turbo and RPCC+ turbo codes with a block length of 2500 bits. The results show that the overall-iterative decoder can be replaced with an overall-noniterative decoder, but that the performance decreases significantly. At block error probabilities below $10^{-2}$, the performance of the simple decoder is somewhere between that of the overall-noniterative MAP decoder and that of the turbo code. However, the performance gain from the RPCC+ turbo code can still be significant with overall-noniterative decoding. For instance, for a block error probability of $10^{-3}$, the required $E_b/N_0$ for the overall-noniterative MAP decoded RPCC+ turbo code is approximately 1.1 dB less than for the turbo code alone. Similarly, the required $E_b/N_0$ for the simple decoder is approximately 0.75 dB less than is required for the turbo code.

VI. CONCLUSIONS

In this paper we present some results on reduced-complexity decoding strategies for RPCC+turbo codes. The RPCC+turbo codes provide significant performance gains over using only turbo codes, even when the complexity of the decoder for the RPCC+turbo code is reduced. In particular, we have shown that the RPCC+turbo code continues to provide a significant performance gain when the number of decoding iterations is limited. We have also presented some decoding structures that make code-combining hybrid ARQ more practical because they do not require additional iterative decoding of the turbo decoder each time that new parity bits are received. We have shown that the RPCC+ turbo code can be used to replace a turbo code with a code that has a simpler decoder while maintaining comparable performance. Thus, the RPCC+ turbo code can be used to bring turbo code performance to wireless systems that cannot tolerate high decoder complexity and latency.

REFERENCES


