Decoding Times of Repeat–Accumulate Codes

David J. C. MacKay

Department of Physics, University of Cambridge

Cavendish Laboratory, Madingley Road,
Cambridge, CB3 0HE, United Kingdom.

mackay@mrao.cam.ac.uk

October 8, 1998 — Draft 3.1

Abstract

Repeat–Accumulate codes (RA codes) are turbo–like codes with near–Shannon limit performance.

This paper shows histograms of decoding times of RA codes.

Please forgive the cocky language in which it is written.

In the turbo code community the widespread practice when decoding is to run the decoder for a fixed number of iterations. At this point, decoding is halted and the bit error rate is computed.

I have spent the last couple of years advocating a different approach (for scientific purposes at least, if not in practice) — namely to detect convergence of the algorithm and halt when a valid state is reached.

Using a fixed number of decoding iterations is wasteful of computer time because the number of iterations required to reach a valid state is a random variable with considerable variance. To get really good performance, the 'fixed number' crew have to use a large number of iterations; most of their computer time is then actually spent chuntering away in a steady state, in order to eke out a small fraction more decoding successes.

It seems a pity not to detect whether a valid state has been reached because no distinction is then made between the two possible failure modes of a sum–product decoder: detected block errors and undetected errors. A detected error occurs when the decoder fails to reach a valid state before a maximum number of iterations is reached. An undetected error occurs when the decoder finds a valid state that corresponds to a codeword different from the one transmitted.

Distinguishing between undetected errors and detected errors is probably a good idea in many engineering applications if it is possible to do it. (For example, if retransmission is an option.)

I have ridden this hobby horse for some time, but it seems only Brendan Frey has implemented a turbo decoder that detects when it can stop. This week, I implemented RA codes and — it goes without saying — my decoder halts when a valid state is reached. This paper shows some results primarily to emphasize that it *is* possible to decode RA codes in this way. These results also give an indication of the potential computational savings for those who simulate RA codes the old–fashioned way, if they will but see the light.

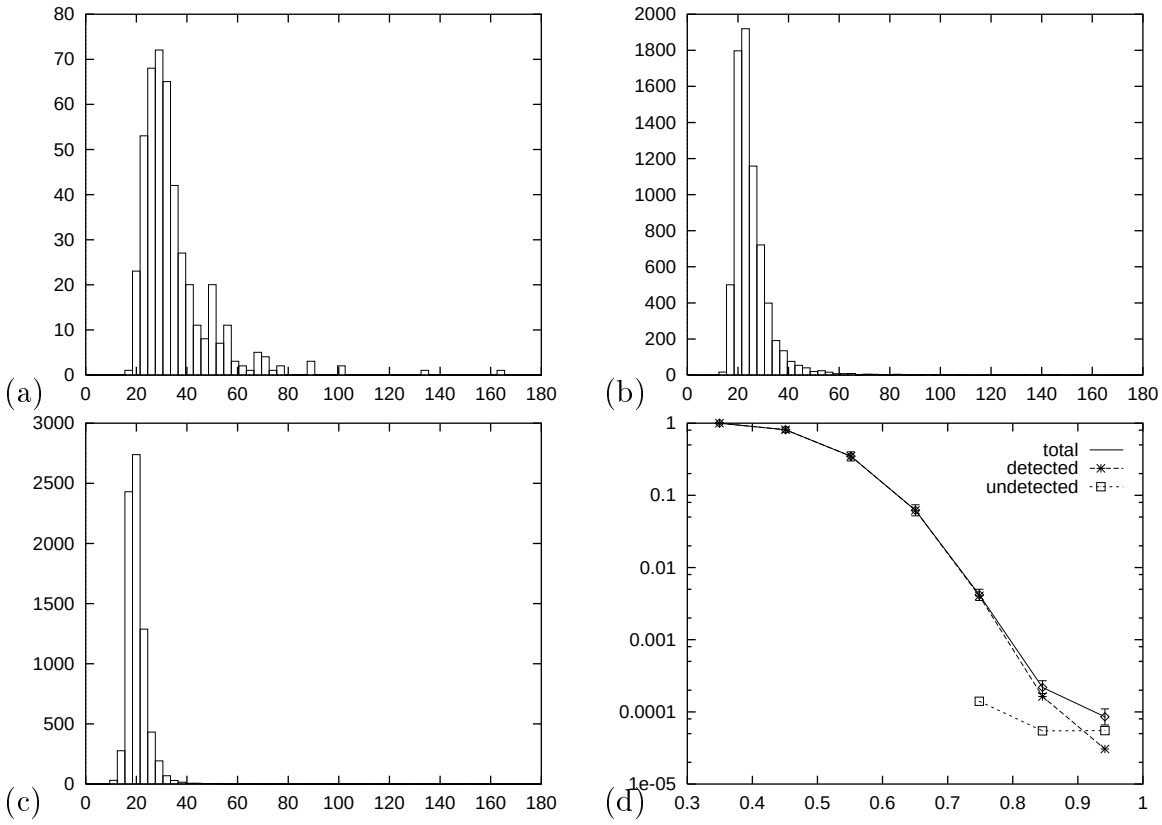

Figure 1: Histograms of number of iterations to find a valid decoding. The RA code has source block length $K = 10000$ and transmitted block length $N = 30000$. (a) Channel signal to noise ratio $x/\sigma = 0.88$, $E_b/N_0 = 0.651$ dB. (b) $x/\sigma = 0.89$, $E_b/N_0 = 0.749$ dB. (c) $x/\sigma = 0.90$, $E_b/N_0 = 0.846$ dB. (d) Block error probability versus signal to noise ratio for the RA code. Most errors in this experiment were detected errors, but not quite all.

Figures 1(a–c) show histograms of the number of iterations for a valid decoding to be found under two different signal–to–noise ratios. The RA code has source block length $K = 10000$ and transmitted block length $N = 30000$. I used a maximum of 200 iterations. If 200 iterations elapsed then a detected error was declared. Detected errors are not included in the histograms.

The histogram of decoding times τ for Gallager codes have been found to follow a power law for large τ [2]. Figures 2(b–c) show power laws fitted by hand to the histograms of figures 1(b–c). Histogram (b) is well fitted by $1/\tau^6$ and (c) is well fitted by $1/\tau^9$.

Figure 1(d) shows the block error probability versus signal to noise ratio for the same RA code of source length $K = 10000$ and transmitted block length $N = 30000$. Most errors in this experiment were detected errors, but at high signal–to–noise ratios some undetected errors occur (one so far, corresponding to a codeword of weight 18, found after 8000 block transmissions).

In my experience RA codes with smaller block lengths make more undetected errors. I'm going to investigate further these undetected errors (which sometimes correspond to codewords and sometimes to *near codewords*).

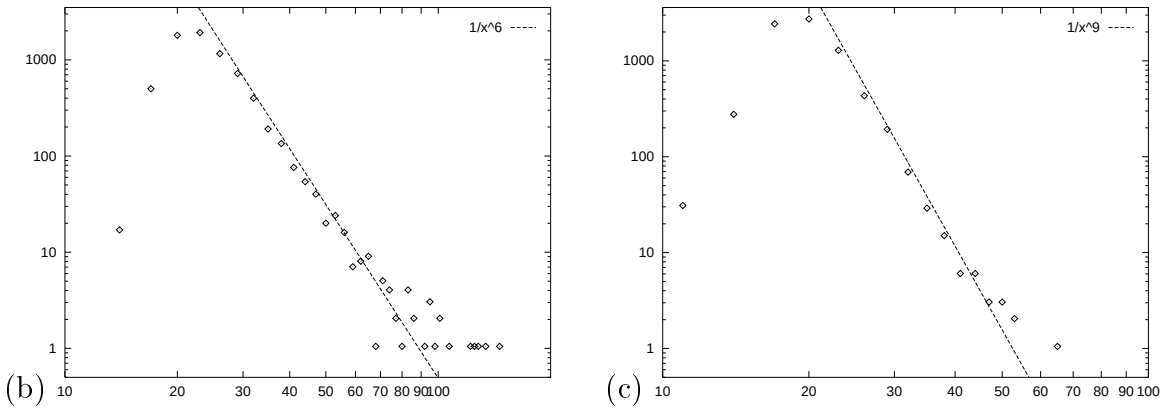

Figure 2: Fits of power laws to the histograms of figure 1. Both the axes are shown on a log scale so that a power law curve appears as a straight line. (b) $x/\sigma = 0.89$, $E_b/N_0 = 0.749$ dB. (c) $x/\sigma = 0.90$, $E_b/N_0 = 0.846$ dB.

Discussion

The Divsalar, Jin and McEliece paper [1] contains graphs for up to 30 iterations. Histogram (a) above shows that roughly 50% of the decodings in my simulation at 0.65dB took more than 30 iterations, with a small but substantial number taking more than 60 iterations. So if you ran the old–fashioned decoder for a whopping twice as long, you could cut the block error probability by a factor of ten or so. However, using the stop–when–it's–done decoder, you can use roughly the same amount of computer time as the original simulation and get the error probability *even lower*.

Nothing is lost, because (if you log the stopping time in a file) you can always recover the old–fashioned graphs if anyone still wants them.

The stop–when–it's–done method

'OK', I hear you say, 'but how do you detect a valid decoder state for an RA code? — Surely any hypothesis about the K source bits is a valid hypothesis?'

The method I used is as follows.

1. While running the sum–product algorithm up and down the accumulator trellis, note the most probable state at each time. (You can get this by multiplying the forward signals by the backward signals.) This state sequence — if you unaccumulate it — defines a sequence of guesses for the source bits, with each source bit being guessed q times. (q is the number of repetitions in the RA code.)

2. When reading out the likelihoods from the trellis, and combining the q likelihoods for each of the source bits, compute the most probable state of each source bit. This is the state which maximizes the product of the likelihoods.

3. If *all* the guesses in (1) agree with the most probable state found in (2) then the decoder has reached a valid state. Halt.

The cost of these extra operations is small compared to the cost of decoding

RA code construction method

In case you want to know how I made the RA code, I simply permuted the source block of length K twice using two randomly chosen $K \times K$ permutations in order to create the scrambled repeated signal. This isn't the same as repeating three times and scrambling the whole block of size N, but I expect it makes little difference when the block length is large.

Results for smaller block lengths

Figure 3 shows results for various block lengths. It also shows results for three codes all having block length $N = 9999$ to assess the variability from code to code in a single ensemble.

References

[1] D. Divsalar, H. Jin, and R. J. McEliece. Coding theorems for 'turbo–like' codes. 1998.

[2] D. J. C. MacKay. Deccoding times of irregular Gallager codes. Unpublished, 1998.

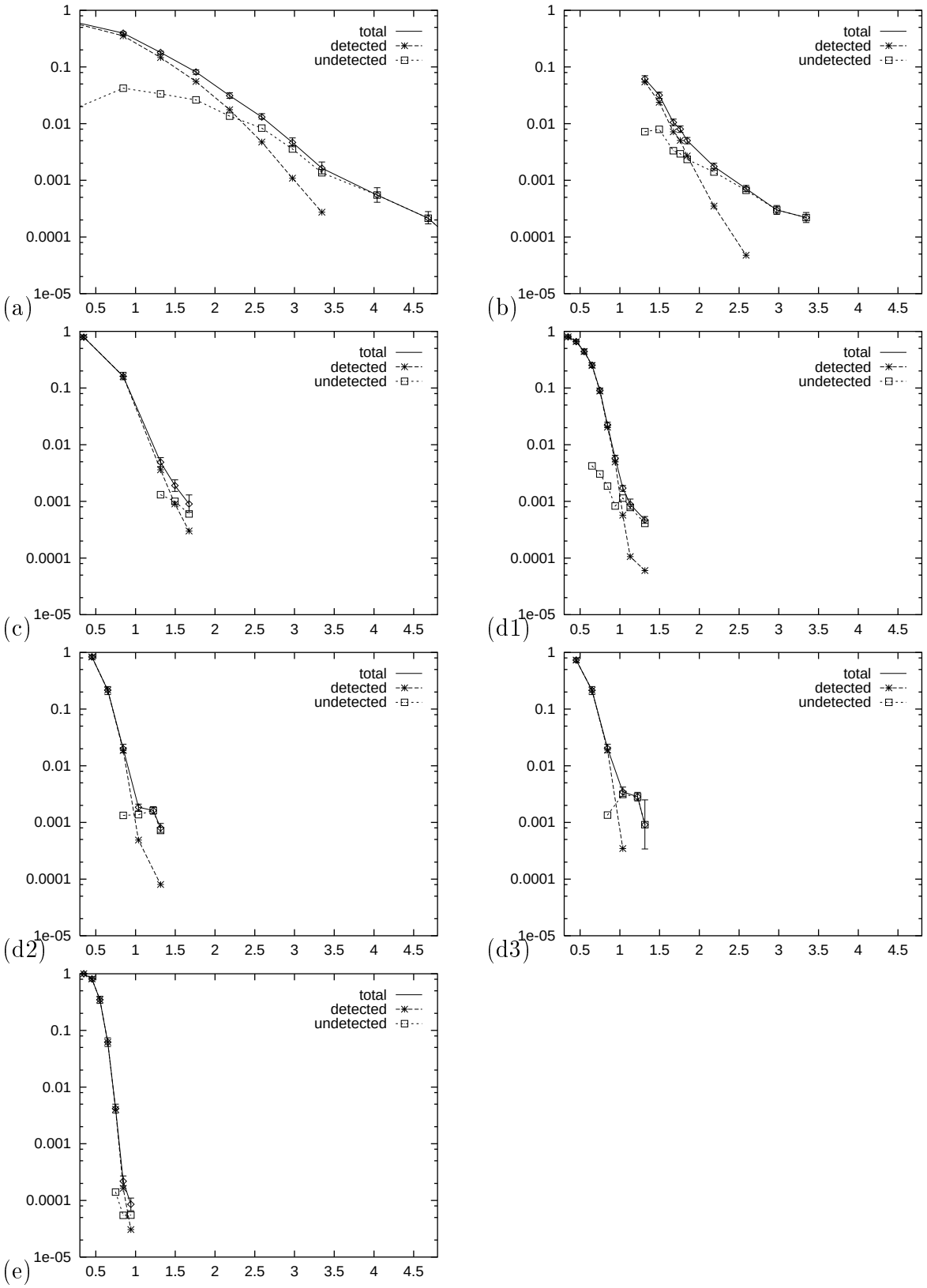

Figure 3: Block error probability versus signal to noise ratio for RA codes with rate 1/3. (a) Source block length $K = 100$, transmitted block length $N = 300$. (b) $K = 333$, transmitted block length $N = 999$. (c) $K = 1000$, transmitted block length $N = 3000$. (d1–3) $K = 3333$, transmitted block length $N = 9999$. (Three codes shown to illustrate the variability from code to code sharing identical parameters.) (e) $K = 10000$, N