Some Soft-Decision Decoding Algorithms for Reed-Solomon Codes

Stephan Wesemeyer*, Peter Sweeney, and David R.B. Burgess

Centre for Comm. Systems Research, University of Surrey, Guildford GU2 5XH, U.K.

Abstract. In this paper we introduce three soft-decision decoding algorithms for Reed-Solomon (RS) codes. We compare them in terms of performance over both the AWGN and Rayleigh Fading Channels and in terms of complexity with a special emphasis on RS codes over $\mathbb{F}_{16}$. The algorithms discussed are variants of well known algorithms for binary codes adapted to the multilevel nature of RS codes. All involve a re-ordering of the received symbols according to some reliability measure. The choice of reliability measure for our simulations is based on a comparison of three in terms of how they affect the codes' performances.

1 Introduction

It is well known that one way of facilitating soft-decision decoding for linear block codes is to represent them by a trellis and apply the Viterbi algorithm (VA) to decode them. However, the complexity of the VA makes its use infeasible for all but a small number of linear codes. Because of the widespread use of RS codes, it would be highly desirable to find efficient soft-decision algorithms for them. Various approaches have been proposed (see [1] for a recent example). This paper introduces a further three. Our simulations were based around an AWGN and a Rayleigh fading channel with BPSK (binary-phase-shift-keyed) modulation and 8-level uniform quantisation. Except in a very few cases with extremely long simulation runs, we based the results on 100 error events (word errors, not bit errors). Throughout the paper we denote by $\mathbb{F}_q$, a finite field of $q = 2^l$ elements and assume an $[n, k]$ linear code over $\mathbb{F}_q$ which can correct t errors.

2 The Algorithms

The Dorsch algorithm was proposed in [2] for binary codes and has more recently been applied by Fossorier and Lin [3]. Given a code of length n and dimension k the idea is to find k most reliable symbols whose positions are such that they can be used as an information set of the code. Various error patterns are added to this information set and each result is re-encoded. In each case, the distance of the obtained codeword from the received word is computed. Decoding stops as soon as we have a maximum-likelihood solution or the number of permitted decoding

* The research was supported by an EPSRC grant.

M. Walker (Ed.): IMA - Crypto & Coding'99, LNCS 1746, pp. 290–299, 1999.

tries has been exhausted (in which case the best solution up to that point is output). Our first two algorithms ($A1$ and $A2$) are based on this technique.

The codeword closest to the received word in terms of the following metric is the maximum-likelihood solution we want to find.

Definition 1. *Let $s'_i = (s'_{i1}, \ldots, s'_{il}) \in \mathbb{F}_q$ be the symbol obtained by using hard decision ($s'_{ij} \in \{0, 7\}$) on $r_i = (r_{i1}, \ldots, r_{il})$, the ith received symbol after quantisation. The distance between a received word $r = (r_1, r_2, \ldots, r_n)$ and a word $c = (c_1, \ldots, c_n) \in \mathbb{F}_q^n$, with $c_i = (c_{i1}, \ldots, c_{il}) \in \mathbb{F}_q$, is defined as*

$$\mathrm{dist}(r, c) = \sum_{i=1}^{n} (\mathrm{dist}_{sym}(r_i, c_i)) \text{ where } \mathrm{dist}_{sym}(r_i, c_i) = \sum_{j=1}^{l} |r_{ij} - c_{ij}| - \sum_{j=1}^{l} |r_{ij} - s'_{ij}|.$$

Furthermore, all algorithms produce a continuous stream of possible solutions which are subjected to a stopping criterion that, if satisfied, is sufficient (though not necessary) to guarantee a maximum-likelihood solution [4], in which case the decoding stops. Since we are concerned here with RS codes, any k symbols may be used as an information set. Hence we simply sort the symbols according to reliability (see Chapter 3) and, in algorithm $A1$, we use the k most reliable as the information set. $A2$ repeats $A1$ using the k least reliable symbols unless a maximum likelihood solution has already been found by $A1$.

Fossorier and Lin's implementation of the Dorsch algorithm checks error patterns corresponding to i errors in the information set. This has been termed order-i reprocessing [3]. In our version, we take a slightly different approach which is closer to the original Dorsch algorithm. Our order for testing the error patterns to be added to the chosen information set is the proximity of the resulting sequence to the corresponding part of the received word. The index used is the generalisation of 'dist' to different length sequences which takes the sum of all the 'dist_{sym}'s over the symbols of the sequence. This is achieved by using a stack-type algorithm, whereby stacks of sequences of different lengths are kept in storage, ordered according to the index. A sequence from the stack of lowest index is extended in q different ways by appending a symbol, the indices of the resulting sequences are calculated and they are each put in the appropriate stack. The memory requirement of this implementation is determined by the maximum number of decoding tries. Let MDT be this maximum and DT be the number of decoding tries so far. Then we only need to keep $MDT - DT$ information sets of smallest index in our array as none of the others will be used.

Our third algorithm ($A3$) simply applies $A1$ and, if that algorithm does not produce a maximum-likelihood solution, then a Chase-style algorithm is applied to the sorted word, i.e. we apply a fixed number of error patterns of least distance to the least reliable symbols and then use an algebraic decoder to decode. This approach has already been applied successfully to binary codes by Fossorier and Lin [5].

3 Sorting

All the algorithms depend on sorting the received symbols according to some reliability measure. In the case of binary codes, Fossorier and Lin showed that on an AWGN and on a Rayleigh fading channel with BPSK modulation, the absolute value of the received symbol is the most appropriate choice [3]. The higher that value is, the more reliable hard decision on the received symbol will be. With RS or indeed any code whose symbols come from a non-binary finite field we need to find a slightly different approach. In such a case, each 'received symbol' will, in fact, be a string of symbols which, between them, indicate the binary representation of the 'received symbol'.

Definition 2. *Let* $r = (r_1, \ldots, r_l)$ *with* $0 \leq r_i \leq 7$ *be a received symbol after quantisation and define*

$$\mathrm{Rel}_1(r) = \sum_{i=1}^{l} |3.5 - r_i|, \;\; \mathrm{Rel}_3(r) = \min\{|3.5 - r_i| \mid 1 \leq i \leq l\}$$

$$\textit{and } \mathrm{Rel}_2(r) = \prod_{i=1}^{l} P(hd(r_i)|r_i), \;\; \text{where } hd(j) = \begin{cases} 0 & : \;\; 0 \leq j \leq 3 \\ 1 & : \;\; 4 \leq j \leq 7 \end{cases}$$

The natural generalisation of the reliability measure of the binary case is to add the absolute values of the symbols in the string, thus obtaining an overall reliability of the 'received symbol'. As we use 8-level uniform quantisation this translates into Rel_1 above. Another approach is to use Bayes' rule. One can easily determine the probability $P(j|0)$ (resp. $P(j|1)$) of a received bit being quantised to level j given that a 0 (resp. a 1) was transmitted. From that we can work out the probability $P(0|j)$ (resp. $P(1|j)$) that a 0 (resp. 1) was transmitted given that we are in level j. Hence we arrive at Rel_2. Lastly, the most basic approach simply takes the least reliable bit in a symbol and uses its value as the overall reliability of the symbol (Rel_3). These three reliability measures were felt to be the most natural ones. It can easily be seen that the higher the computed reliability of a symbol is, the more likely it is to be correct.

Figure 1 (respectively Figure 2) contains the results for a $[16, 8, 9]$ ($[16, 12, 5]$) extended RS code over the AWGN channel (see Section 6.2 for the Rayleigh channel results), decoded using algorithm $A1$ with a maximum number of decoding tries corresponding to the number of order-2 (order-2 and order-1) reprocessing attempts with and without sorting. Note that, to compute the probabilities accurately for Rel_2, we need to know at which signal-to-noise ratio (SNR) the bits were transmitted. As this information is not always available in practice, we computed the probabilities for a SNR of 1dB adjusted by the code rate, R say, i.e. SNR$= R \cdot 10^{0.1}$ and used these values throughout. (The simulations showed that - if anything - this approach proved slightly better than using the exact values for the different SNRs.)

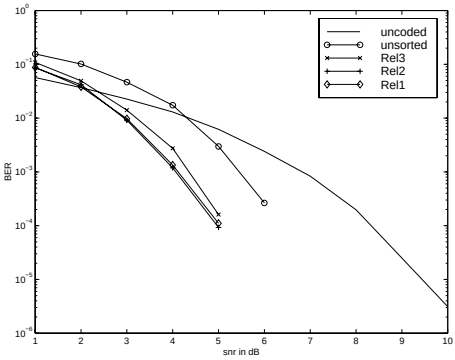

Fig. 1. Sorted vs unsorted $[16, 8, 9]$ extended RS code - 529 decoding tries

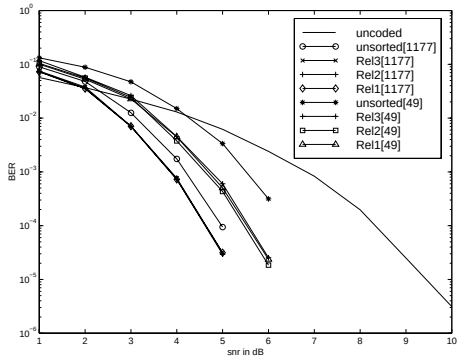

Fig. 2. Sorted vs unsorted $[16, 12, 5]$ extended RS code - 1177 and 49 decoding tries

As can be seen all reliability measures result in a marked improvement over the unsorted case. The reason why Rel_3 is slightly worse than the other two (except in the case of 1177 decoding tries for the higher rate code) can be explained by the observation that the number of different reliability levels attached to each symbol in that method is rather low (4 compared to 26 for Rel_1 and 35 for Rel_2). At 1177 decoding tries, the algorithm performs close to maximum-likelihood decoding in any case - it is not important whether or not the least distorted symbols are used as an information set.

Because there was no significant difference between sorting the symbols of the received words according to Rel_1 or Rel_2 we used sorting by Rel_1 in all the remaining simulations.

4 Number of Decoding Tries

The most crucial feature of the proposed algorithms is the number of decoding tries they entail. The more decoding tries the more likely it is that we find the maximum-likelihood solution. However, as Fossorier and Lin [3] demonstrated the actual gain obtained from further decoding tries has to be measured against the extra computation involved.

Figure 3 is an example of how the maximum number of decoding tries (using algorithm $A1$) after sorting (with respect to Rel_1) can affect the performance of a code and how this performance compares to the unsorted case. Note that 41449, 5489, and 529 correspond to the number of decoding tries given by order-4, order-3, and order-2 reprocessing respectively. There is a marked improvement of about 1dB going from 529 decoding tries to 5489 but only a slight improvement of roughly 0.25dB when 41449 attempts are used instead of 5489 which does not justify the almost 8-fold increase in number of decoding tries. However, even then the complexity of the proposed algorithm is several orders of magnitude lower

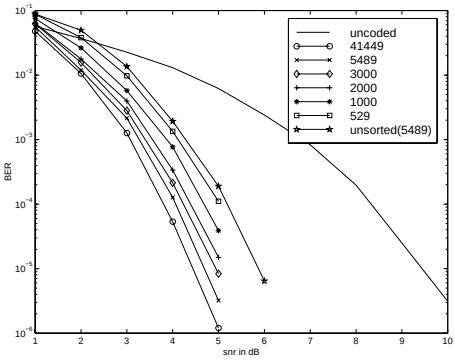

Fig. 3. $[16, 8, 9]$ extended RS code - different numbers of permitted decoding tries

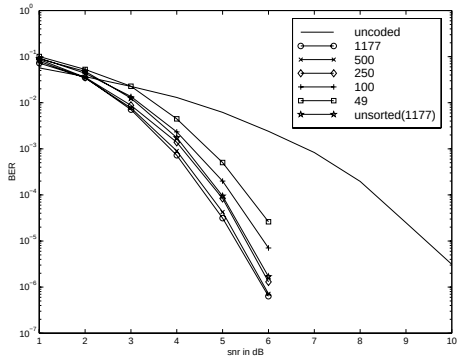

Fig. 4. $[16, 12, 5]$ extended RS code - different numbers of permitted decoding tries

than that of the Viterbi algorithm which would have to deal with $16^8 \approx 4.3 \cdot 10^9$ states for this code.

Figure 4 shows the effect of different numbers of permitted decoding tries (using algorithm $A1$) for a $[16, 12, 5]$ extended RS code. This time we restricted the number of decoding tries to lie in between 49 and 1177(= number of decoding tries for order-1 and order-2 reprocessing respectively). Note that decoding after sorting with a maximum of 250 decoding tries slightly outperforms unsorted decoding with maximum 1177 decoding tries and there is only a very slight improvement going from 500 to 1177 decoding tries.

5 Measures of Complexity

The complexity of each algorithm is expressed in terms of additions, multiplications and comparisons which, for simplicity, are considered equivalent operations. All the estimates we give are based on our implementation; the idea is to give a rough idea of how much computational effort has to be expended on decoding. To enable us to compare the results with other algorithms and to eliminate the code rate as a factor, for each code considered we measure the complexity in operations per information bit. We compare our results throughout with the Viterbi algorithm applied to a convolutional code of rate $R = 0.5$ and memory $k = 7$ even though the rate of the RS codes vary. Higher rate convolutional codes are usually obtained by puncturing which does not greatly affect the number of operations which can be estimated at 128 comparisons (= number of states) plus 256 additions (= number of branches).

Our implementation of the $A1$ and $A2$ algorithms require, before the re-encoding starts, computing the metric and some values for the stopping criterion ($n(q-1)^2$ comparisons and nlq additions), sorting the symbols according to reliability (approximately $n \log_2(n)$ comparisons) and reducing a (k, n) matrix to

reduced echelon form (REF) (nk^2 multiplications and $nk(k-1)$ additions). This latter is performed twice in $A2$ (two directions of decoding), so the preliminary operations for algorithm Ar (r=1,2) total:

$$PopAr = n(\log_2(n) + lq + (q-1)^2 + r(k^2 + k(k-1))). \tag{1}$$

For each decoding try (both algorithms), there are the following approximations: re-encoding ($(n-k)k$ multiplications and $(n-k)(k-1)$ additions), determining the distance from the received word ($(n-1)$ additions), determining whether the stopping criterion is satisfied ($(n+2+n\log_2(n)$ comparisons and $n-k+1$ additions), determining the best solution (1 comparison per decoding try after the first). Thus altogether the algorithm Ar (r=1,2) requires the following total operations (where DT is the number of decoding tries).

$$TopAr = PopAr + DT(2(n-k)k + 2n + n\log_2(n) + 2) + (DT-1) \tag{2}$$

The estimates for our implementation of the Chase part of $A3$ are based on a very general algorithm presented in Stichtenoth [6] and due to A.N.Skorobogatov and S.G.Vladut. The following are the operations per decoding try: Computing the syndrome ($(n-k)n$ multiplications and $(n-k)(n-1)$ additions), checking whether the syndrome is 0 ($n-k$ comparisons), reducing the $(t, t+1)$ syndrome matrix to REF ($(t+1)t^2$ multiplications and $(t+1)t(t-1)$ additions), finding the error locator polynomial ($t(t+1)/2$ multiplications and the same number of additions), determining the roots of that polynomial (a maximum of qt multiplications, qt additions and q comparisons), finding the error values ($((t+1)(n-k)^2$ multiplications and $(t+1)(n-k)(n-k+1)$ additions), obtaining the codeword (t additions) and computing the distance from the received word and applying the stopping criterion ($(n-1)+(n-k+1)$ additions and $(n+2+n\log_2(n))$ comparisons). Thus, denoting by DTC the number of decoding tries involved in the error-only decoder, the total number of operations required for the $A3$ algorithm is given by

$$\begin{aligned} TopA3 = TopA1 + DTC\cdot(\ & 2(n-k)n + (t+1)(2t^2 + 2q + (n-k)(2(n-k)+1)) \\ & + t + 3n - k + 2 + q + n\log_2(n)) \end{aligned} \tag{3}$$

As all our algorithms apply a stopping criterion it is easy to see that the higher the SNR, the fewer the decoding attempts needed on average. In our simulation we computed the average number of decoding tries per received word which is then used to compute the total number of operations as given by the above formulae. It is worth noting that the complexity of all three algorithms is dominated by the number of decoding tries. Only for high SNRs, when the average number of decoding tries becomes very small, do the preliminary operations contribute significantly to the average number of operations per information bit.

6 Comparing the Three Decoding Algorithms in Terms of Performance and Complexity

6.1 AWGN Channel

Figures 5, 7 and 9 show the performance of the algorithms when applied to respectively a $[16, 8, 9]$, a $[16, 10, 7]$ and a $[16, 12, 5]$ extended RS code. The numbers next to the various algorithms indicate the number of permitted decoding tries, e.g. for the $[16, 8, 9]$ code, the $A2$ algorithm was run with maximum 1500 decoding tries for each side, and the $A3$ algorithm was run with 529 (first number) Dorsch-style decoding tries permitted and the same maximum number of Chase-style decoding tries. We have included the performance of Forney's GMD [7] and an error-only decoder to enable the reader to compare the new algorithms with two standard ones. Tables 1, 2 and 3 show how many decoding tries were needed for each algorithm at various SNRs. Figures 6, 8 and 10 show the complexity of the algorithms based on the figures in the tables.

Table 1. Ave. num. of decoding tries ($[16, 8, 9]$ extended RS code)

Algorithm	1dB	2dB	3dB	4dB	5dB
$A1[41449]$	40018	36407	27781	14732	4353
$A1[3000]$	2912	2636	1985	1076	316
$A2[1500, 1500]$	2902	2666	2006	1063	310
$A3[529, 529]$	$[519, 519]$	$[462, 462]$	$[356, 355]$	$[189, 187]$	$[57, 55]$

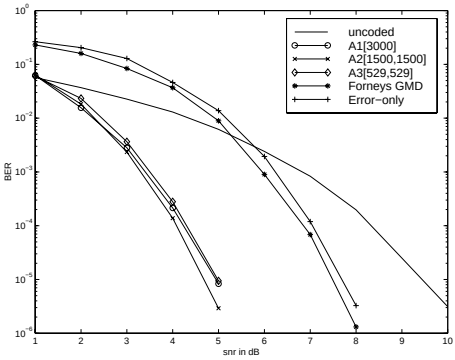

Fig. 5. $[16, 8, 9]$ extended RS code decoded using $A1$, $A2$, and $A3$

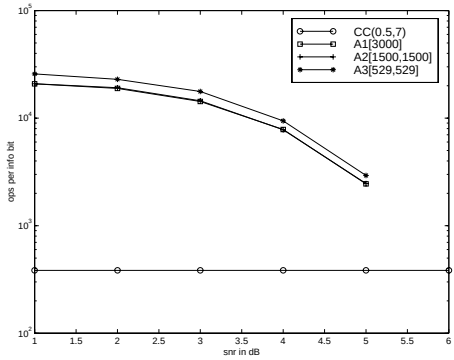

Fig. 6. Complexity of the algorithms ($[16, 8, 9]$ extended RS code)

Comparing Figure 5 with Figure 7, in terms of the BER at various SNR there is hardly any difference between the $[16, 8, 9]$ and the $[16, 10, 7]$ codes, probably

Table 2. Ave. num. of decoding tries ($[16, 10, 7]$ extended RS code)

Algorithm	1dB	2dB	3dB	4dB	5dB
$A1[3000]$	2865	2513	1805	882	229
$A2[1500, 1500]$	2908	2460	1811	883	229
$A3[821, 301]$	$[795, 291]$	$[693, 254]$	$[497, 181]$	$[241, 87]$	$[64, 22]$

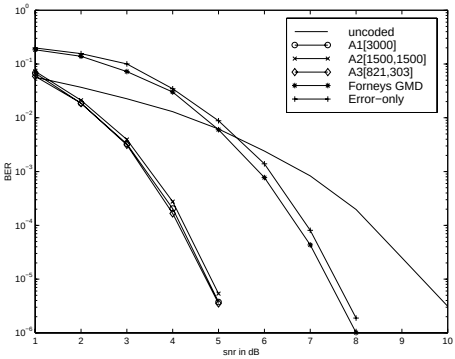

Fig. 7. $[16, 10, 7]$ extended RS code decoded using $A1$, $A2$, and $A3$

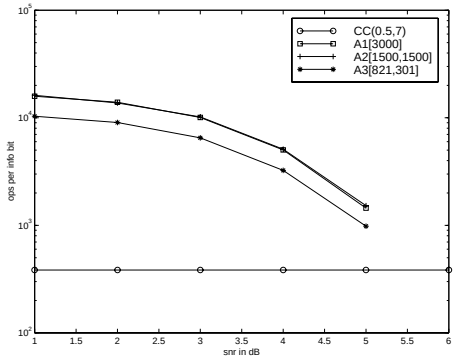

Fig. 8. Complexity of the algorithms ($[16, 10, 7]$ extended RS code)

Table 3. Ave. num. of decoding tries ($[16, 12, 5]$ extended RS code)

Algorithm	1dB	2dB	3dB	4dB	5dB	6dB
$A1[500]$	482	401	305	145	44	7
$A2[250, 250]$	477	403	304	145	44	7
$A3[49, 17]$	$[46, 15]$	$[41, 14]$	$[29, 10]$	$[16, 5]$	$[6, 2]$	$[2, 1]$

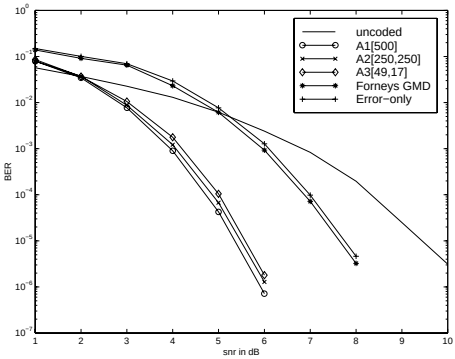

Fig. 9. $[16, 12, 5]$ extended RS code decoded using $A1$, $A2$ and $A3$

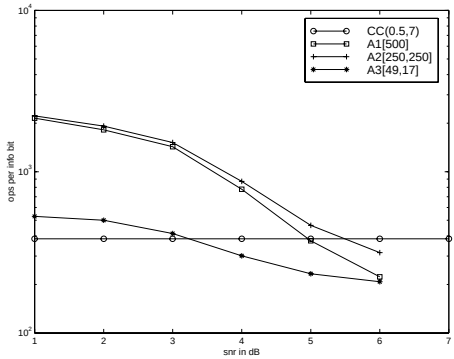

Fig. 10. Complexity of the algorithms ($[16, 12, 5]$ extended RS code)

due to the fact that these decoding algorithms are suboptimal and hence do not achieve the full potential of the lower rate code. In addition, in both Figure 6 and Figure 8, the pair of curves "$A1[3000]$" and "$A2[1500, 1500]$" overlap. However, whereas the $A3$ algorithm appears to be the best choice for the $[16, 10, 7]$ code in terms of both performance and complexity, for the rate $1/2$ code, $A2$ slightly outperforms the other two algorithms and (like $A3$) allows a straightforward parallel implementation, so it is the preferable choice for this code.

In the case of the $[16, 12, 5]$ extended RS code the $A1$ algorithm performs slightly better than the other two. However, it is worth noting that the $A3$ algorithm achieves good results with a very low maximum number of decoding tries and that by slightly increasing the number of decoding tries for the $A3$ algorithm, from $[49, 17]$ to $[100, 50]$, say, one gets a similar performance to the $A1$ algorithm while still having a lower complexity and the advantage of being able to implement it in parallel. This time even for low SNRs the complexity of $A3$ is only slightly worse than that of the Viterbi algorithm. At higher SNRs all algorithms achieve good results with few decoding tries resulting in very few operations per information bit.

6.2 Rayleigh Fading Channel ($[16, 8, 9]$ Extended RS Code Only)

In our simulations we have assumed a perfectly interleaved Rayleigh fading channel, i.e. the fading amplitudes for each bit were completely independent and no channel side information was used in the decoding. We have used the same metric as for the AWGN channel.

Table 4. Ave. num. of decoding tries ($[16, 8, 9]$ extended RS code (Rayleigh channel))

Algorithm	2dB	3dB	4dB	5dB	6dB	7dB	8dB
$A1[3000]$	3000	2929	2865	2670	2383	1941	1397
$A2[1500, 1500]$	3000	2964	2879	2726	2359	1944	1400
$A3[529, 529]$	$[529, 529]$	$[518, 518]$	$[506, 506]$	$[475, 475]$	$[426, 425]$	$[346, 345]$	$[250, 248]$

Figure 11 shows that the results for the Rayleigh fading channel do not differ very much from the ones we obtained for the AWGN channel. We see, as for AWGN in Section 3, that sorting with respect to the reliability measure Rel_1 or Rel_2 (the two curves overlap) is better than Rel_3. Furthermore, as in the AWGN channel, sorting with 529 decoding tries yields a better performance than unsorted decoding with 5489 decoding tries. For the Rayleigh fading channel, sorting yields a coding gain of about 2dB when compared to the unsorted case with the same number of decoding tries. This time it seems that the algorithms $A1$ and $A2$ perform identically. However, looking at the computed BER values, there is an indication that A2 might outperform A1 slightly for SNRs higher than these. In terms of complexity - see Figure 12 and Table 4 - the only difference

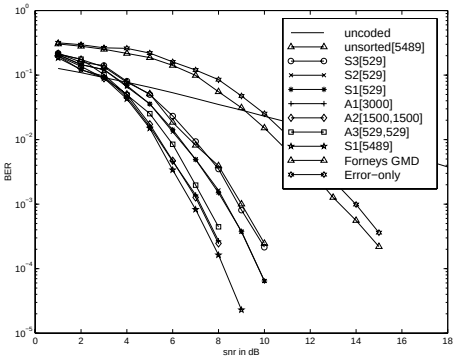

Fig. 11. [16, 8, 9] extended RS code decoded using $A1$,$A2$,$A3$ (Rayleigh)

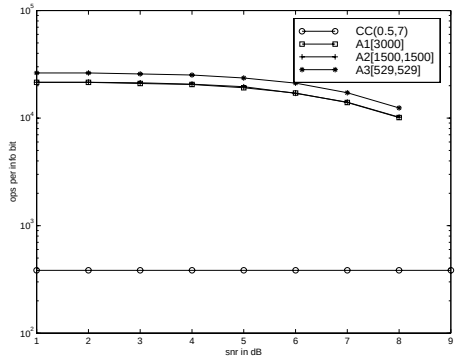

Fig. 12. Complexity of algorithms ([16, 8, 9] extended RS code (Rayleigh))

from the AWGN channel is that the average number of decoding tries decreases more slowly which is obviously due to the nature of the Rayleigh fading channel. Note that, again, the curves for "A1[3000]" and "A2[1500,1500]" overlap.

7 Conclusion

In this paper we have introduced three suboptimal decoding algorithms for RS codes all of which achieve a reduction in complexity of several orders of magnitude over the Viterbi algorithm for these codes whilst keeping the loss in coding gain very small. These algorithms are not restricted to RS codes and could be applied to any linear block code. They achieve their full potential with high rate codes where a small number of decoding tries yields almost maximum-likelihood decoding performance with low decoding complexity.

References

1. Wesemeyer S. and Sweeney P.: Suboptimal soft-decision decoding for some RS-codes. IEE Electronics Letters **34(10)** (1998) 983–984
2. Dorsch B.G.: A decoding algorithm for binary block codes and J-ary output channels. IEEE Trans. Inform. Theory **IT-20(3)** (1974) 391–394
3. Fossorier M.P.C. and Lin S.: Soft-decision decoding of linear block codes based on ordered statistics. IEEE Trans. Inform. Theory **41(5)** (1995) 1379–1396
4. Taipale D.J. and Pursley M.B.: An improvement to generalized-minimum-distance decoding. IEEE Trans. Inform. Theory **37(1)** (1991) 167–172
5. Fossorier M.P.C. and Lin S.: Complementary reliability-based decodings of binary linear block codes. IEEE Trans. Inform. Theory **43(5)** (1997) 1667–1672
6. Stichtenoth, H.: Algebraic function fields and codes. Springer-Verlag, 1993
7. Forney Jr, G.D.: Generalized minimum distance decoding. IEEE Trans. Inform. Theory **IT-12** (1966) 125–131