PERFORMANCE OF THE ADAPTIVE VITERBI ALGORITHM: ERROR PROBABILITY AND COMPUTATIONAL EFFORT

gratuit

François Chan and David Haccoun

École Polytechnique de Montréal

Juin 1996

Dépôt légal, Juin 1996
Bibliothèque nationale du Québec
Bibliothèque nationale du Canada

Performance of the adaptive viterbi algorithm:
error probability and computational effort

François Chan et David Haccoun
Département de Génie Électrique et
de Génie Informatique

Pour se procurer une copie de ce document, s'adresser au:

Service des Éditions
École Polytechnique de Montréal
Case postale 6079, Succursale Centre-Ville
Montréal (Québec) H3C 3A7
Téléphone: (514) 340-4473
Télécopie: (514) 340-3734

Compter 0,10$ par page et ajouter 3,00$ pour la couverture, les frais de poste et la manutention. Régler en dollars canadiens par chèque ou mandat-poste au nom de l'École Polytechnique de Montréal.

Nous n'honorerons que les commandes accompagnées d'un paiement, sauf s'il y a eu entente préalable dans le cas d'établissements d'enseignement, de sociétés ou d'organismes canadiens.

PERFORMANCE OF THE ADAPTIVE VITERBI ALGORITHM: ERROR PROBABILITY AND COMPUTATIONAL EFFORT*

François Chan and David Haccoun

Département de génie électrique et de génie informatique

Ecole Polytechnique de Montréal

C.P. 6079, succ. Centre-Ville, Montréal, H3C 3A7

***Abstract*—In this paper an adaptive decoding algorithm for convolutional codes, which is a modification of the Viterbi algorithm (VA) is presented. For a given code, the proposed algorithm yields nearly the same error performance as the VA while requiring a substantially smaller average number of computations. Unlike most of the other suboptimum algorithms, this algorithm is self-synchronizing: if the transmitted path is discarded, the AVA can recover the state corresponding to the transmitted path after a few trellis depths. Using computer simulations over hard and soft 3-bit quantized Additive White Gaussian Noise channels, it is shown that codes with a constraint length K up to 11 can be used to improve the bit error performance over the VA with K=7 while maintaining a similar average number of computations. Although a small variability of the computational effort is present with our algorithm, this variability is exponentially distributed, leading to a modest size of the input buffer and hence, a small probability of overflow.**

* This research has been supported in part by the Natural Sciences and Engineering Research Council of Canada, the Fonds pour la formation des Chercheurs et l'Aide à la Recherche (FCAR) du Québec and by the Canadian Institute for Telecommunications Research under the NCE program of the Government of Canada.

I. INTRODUCTION

The use of convolutional codes with probabilistic decoding can significantly improve the error performance of a communications system [1], [2]. The Viterbi algorithm (VA) [1]–[3], which is one of the most widely used decoding algorithms is optimal but its complexity in both number of computations and memory requirement increases exponentially with the constraint length K of the code, limiting it in practice to codes with $K \leq 9$. Hence, when codes with a longer constraint length are required in order to achieve a lower error probability, decoding algorithms whose complexity does not depend on K become especially attractive.

Several multiple-path, breadth-first, decoding algorithms, such as the M-algorithm and Simmons's algorithm have been proposed as alternatives to the VA [4]–[8]. Unfortunately with these algorithms, should the correct path be lost, then its recovery is rather difficult, leading to very long error-events. The error propagation is usually contained by organizing the data in frames or blocks with a known starting state or by using some special recovery scheme [9]. Although these algorithms may achieve a low error-event probability, the bit error probability is rather poor in the absence of a good recovery scheme. In the algorithm proposed here, which is a modification of the VA, this shortcoming may be easily circumvented.

In the proposed algorithm, called Adaptive Viterbi Algorithm (AVA), instead of keeping all the 2^{K-1} states at each trellis level, only some of the best or most likely states are kept. For any trellis level, should the state corresponding to the correct path be discarded, then unlike most of the other suboptimum algorithms, the AVA can automatically recover the correct state after a few trellis levels. Hence, catastrophic error propagation is avoided without requiring the data to be separated in blocks with a tail of known bits. As a consequence, not only can the AVA achieve a good error-event probability, but also a good bit error probability. In addition, since for a given code the number of survivors is smaller than with the VA, then a longer and more powerful code can be used while requiring a smaller average number of survivors.

The paper is organized as follows. In Section II, the Adaptive Viterbi Algorithm is described and compared to other decoding algorithms. The error performance of this algorithm is then investigated in Section III. Simulations results on the performance of the proposed algorithm over

Additive White Gaussian Noise channels are provided in Section IV and compared with those of the Viterbi algorithm. For a given code the AVA can achieve an error performance similar to that of the VA while requiring a much smaller average number of survivors. In Section V decoding dynamics and the distribution of computation are investigated. Simulations results indicate that the probability of input buffer overflow can be made as small as desired.

II. DESCRIPTION OF THE PROPOSED ALGORITHM

The Viterbi algorithm (VA) examines all possible distinct paths in the trellis and determines the most likely one [1]–[3]. In order to reduce the complexity of the VA, defined as the number of survivors, and hence be able to use more powerful codes, the number of survivors at each trellis level must be reduced by discarding some of the least likely paths. In the proposed algorithm, instead of keeping all the states at each trellis level, only a number of the most likely states are kept and all the others are discarded. The selection is based on the likelihood or metric value of the paths, which for the VA over a binary symmetric channel is the Hamming distance. The number of survivors is not constant but varies according to the needs imposed by channel noise. We call this algorithm *Adaptive Viterbi Algorithm* (AVA).

In the AVA, every surviving path at trellis level (l–1) is extended and its successors at level l are kept if their Hamming distances are smaller or equal to $(d_m + T)$, where T is a discarding threshold selected by the user and where d_m is the minimum Hamming distance of all survivors at level (l-1), that is, d_m is the Hamming distance of the most likely path at level (l-1). Furthermore, for reasons of practicality the total number of survivors is not allowed to exceed some number N_{max}, selected by the user. Therefore, a successor is kept only if the number of survivors does not exceed N_{max}. Using the minimum distance at the previous level (l–1) instead of the minimum distance at the current level l allows to decide immediately whether a path should be kept or not without waiting for all path distances at level l to be computed, i.e., without knowledge of the minimum distance at the current level. As in the VA, path remergers are considered. However, to simplify the implementation, when a path merges to a given state with a smaller distance than a previous path, that previous path is not eliminated immediately.

Instead, a large number is added to its distance so that at next trellis level, the distance will not satisfy the threshold and hence, the path will be discarded. Clearly, the value of the discarding threshold T directly influences the number of survivors and the probability of error. Determining appropriate values for T is discussed in Section III.

Since at any trellis depth the number of survivors is limited to N_{max}, then clearly, only the N_{max} most likely among all possible survivors satisfying the threshold must be extended. To that effect a simple sorting operation is used. The states satisfying the threshold rule are sorted using a number of (T+1) bins: state i with Hamming distance d_i is stored in bin j if $d_i \leq d_m + j$, $0 \leq j \leq T$. At the next trellis level, paths stored in bin 0 are extended first, followed by those in bin 1 and so on until no more path remains or until the number of survivors exceeds N_{max}. This very simple sorting method ensures that when the number of paths satisfying the discarding threshold exceeds N_{max}, the paths extended are more likely than the ones not extended. The effect of N_{max} on the performance is analyzed below.

In order to illustrate how the AVA operates, an example using a rate R=1/2, constraint length K=3 code is given in Fig. 1 where the received sequence **Y**=(01,10,00,01,00,11), including a tail of 2 zeros, is decoded with the AVA. The threshold T is set equal to 1, i.e., a path is rejected if its distance from the received sequence **Y** is larger than $(d_m + 1)$. As expected, it can be seen from Fig. 1 that the number of survivors with the AVA is smaller than with the VA which has 2^{K-1} or 4 survivors at each trellis level. The decoded path, shown in heavy line, would be the same if the VA had been used in this example.

When there is no channel error, the Hamming distance spread of the extended paths is large, leading to a small number of survivors. When the channel is very noisy, the distance of the transmitted path may not satisfy the threshold anymore. Hence, the transmitted path may be discarded, leading to erroneous decoding. In this case, the distance spread is small because all the extended paths tend to have approximately the same Hamming distance on the average and thus, the number of survivors tends to increase. If the maximum number of survivors N_{max} is chosen to be equal to 2^{K-1} and if the threshold value is large enough, after a few trellis depths the number of survivors increases to 2^{K-1} which corresponds to the number of survivors of the

VA. Since the total number of distinct states in the trellis is equal to 2^{K-1}, then clearly, the correct state must be one of the survivors. It should be noted that the correct path has been discarded for a few branches but the correct state can be recovered and therefore, the decoded path can remerge with the transmitted one. Erroneous decoding does not last until the end of the information sequence, but is limited to the time period during which the correct path has been lost. By recovering the correct state, catastrophic error propagation is avoided, thus preventing a high bit error probability. For example, it has been observed from our simulations that with a threshold T=4, the correct state is recovered within one constraint length on the average with a K=7 code and within two constraint lengths with a K=10 code.

We now determine the probability P_{rec} that the correct state is recovered. It has been mentioned above that when the channel is noisy and the correct path is lost, the distances of extended paths tend to be close to each other. Let the number of paths satisfying the threshold at a given trellis level be larger than N_{max}. The recovery probability P_{rec} or the probability of selecting at that trellis level the correct state among the N_{max} survivors is the ratio of the favorable to the total number of outcomes and may be written as

$$P_{rec} = \frac{\binom{2^{K-1}-1}{N_{max}-1}}{\binom{2^{K-1}}{N_{max}}} \tag{1}$$

This problem is equivalent to that of obtaining a specific card when N_{max} cards are drawn from a deck of 2^{K-1} cards. The numerator is the number of favorable outcomes, that is, the number of groups containing the correct state: the correct state is selected and then the $N_{max}-1$ other states are chosen among the $(2^{K-1}-1)$ remaining states. Thus, it is equal to $\binom{2^{K-1}-1}{N_{max}-1}$. The denominator represents the number of different groups of N_{max} states that can be chosen from a set of 2^{K-1} states. After some manipulations, (1) becomes

$$P_{rec} = \frac{N_{max}}{2^{K-1}} \tag{2}$$

Hence, the probability of recovering the correct state P_{rec} is equal to 1 if and only if $N_{max} = 2^{K-1}$ and the threshold, which is analyzed in Section III, is large enough so that N_{max} can be reached. In all other cases, $P_{rec} < 1$ and resynchronization or recovery of the correct state once it is lost is not guaranteed. As a consequence, N_{max} is selected to be equal to 2^{K-1} in all our simulations.

In summary, this algorithm is similar to the VA except for the selection and the number of survivors which is not constant. The AVA decoder adapts its computational effort to the current quality of the channel. We now describe what distinguishes the AVA from other well-known decoding algorithms.

Comparison of the AVA with other decoding algorithms

Let us first consider the VA. Being optimal, for a given code with a constraint length K, the VA provides the best possible error performance. However, it requires a very large number of survivors, 2^{K-1} per trellis level. The advantage of the AVA over the VA is that on average the number of survivors is reduced at the expense of a small computational variability. Just like the VA, the AVA can resynchronize automatically without requiring a tail of known bits at the end of the frames. Hence, the AVA can be used in the same applications as the VA, provided a modest input buffer is available. It will be shown that the error-event probability and the bit error probability of the two algorithms for a given code can both be very close with the choice of appropriate parameters values for the AVA. Compared to the VA, for a given error performance, the average number of survivors of the AVA is smaller, or for a given average decoding effort, the error performance of the AVA is superior.

Next, sequential decoding is considered. The major shortcoming of sequential decoding [1] is the large, Pareto-type variability of the computational load [10], [11]. Compared to sequential decoding, the average computational load of the AVA is larger but far less variable. Therefore, for a given buffer size, the typical input buffer overflow probability is much smaller for the AVA than it is for sequential decoding. To summarize, the decoding effort of sequential decoding is very small but very variable, leading to a high buffer overflow probability, while with the AVA, the number of survivors is larger on average but far less variable, resulting in a very small overflow probability.

Turning to the M-algorithm [6], [9], the main difference between the AVA and the M-algorithm is resynchronization, one of the M-algorithm's biggest disadvantage. Since the M-algorithm cannot easily recover the lost transmitted path, frames with a tail of known bits are required, which reduces slightly the effective data throughput. Furthermore, when the correct path is lost, the bit error probability of the M-algorithm is quite large due to error propagation. On the other hand, since the AVA can recover the correct state as mentioned previously, it can achieve both a small error-event probability and a small bit error probability. For a given error performance, the AVA requires a smaller average number of survivors at the expense of a small variability and modest input buffer. Using an input buffer, the AVA can take advantage of the smaller average computational load by operating at a higher data rate without increasing the clock frequency. Finally, it should also be noted that the selection of M among 2M paths is much more complex than a simple comparison to a threshold.

The AVA presents some similarities with Simmons's algorithm (SA) [8]. They both use a threshold to select surviving paths, like many other adaptive algorithms [4], [12]. There is, however, a major difference between the AVA and the SA: the AVA can resynchronize automatically whereas the SA cannot. Hence, the bit error probability of the AVA can be very close to that of the VA, whereas with the SA, only the error-event probability has been considered [8].

Several other differences between these two algorithms can be observed. First, the AVA considers the minimum distance at the previous trellis level. The SA uses the minimum distance of the current level to select the survivors and hence, requires two steps to process the extended paths: the minimum distance must first be determined and then, the paths not satisfying the threshold are rejected. Another difference between the AVA and the SA is sorting. The SA does not sort the selected paths, while the sorting in the AVA is very simple and avoids the "iterative reprocessing with tightened thresholds" proposed by Simmons when the number of survivors exceeds the allowed maximum number. In addition, the SA does not consider path remergers.

In summary, when comparing the five steps of the SA as described by Simmons [8], only the first one, which consists in extending the surviving paths and computing their new metrics, is common to both algorithms. Since the SA, just like the M-algorithm, cannot recover the lost transmitted path, a tail is still required at the end of each frame and hence, the bit error probability may be quite high due to error propagation.

III. ELEMENTS OF ERROR PERFORMANCE ANALYSIS

In this section, we show that the error performance of the AVA can be very close to that of the VA if the algorithm's parameters (especially the threshold and the maximum number of survivors) are properly set. Error events of the AVA can be classified into two categories:

1) Errors caused by a burst of noise exceeding the error-correcting capability[1] t of the code. These errors are due to the limitations of the code and occur even with an optimum algorithm, such as the VA.

2) Errors resulting from the discarding of the correct path even though the error-correcting capability t of the code is not exceeded. These errors, due to the limitations of the decoding algorithm, could have been avoided if the VA had been used.

The two categories above are mutually exclusive because of the condition in 2) specifying that the number of channel errors does not exceed t. An error-event occurring with both the AVA and the VA belongs to category 1 because category 2 comprises only error-events that can be avoided with the VA. Hence, a given error-event belongs either to category 1 or to category 2.

Likewise, bursts of noise can be classified into two categories: category 1 corresponds to bursts causing a Viterbi-type behavior, category 2 to bursts causing a suboptimum-type behavior with loss of the correct path and erroneous decoding, but no error with the VA. A given burst of noise belongs either to category 1 or category 2. These two categories are mutually exclusive and represent all possible cases. Fig. 2 (a) illustrates the classification of the bursts of noise in the two categories. A burst of noise can cause no error-event, which corresponds to subset A of category 1, or it can cause an error-event with the AVA but not with the VA (subset B of category 2). It can also cause an error-event with any algorithm, including the VA (subset C of category 1). The number of channel transitions or symbol errors that a burst of noise must cause in order

[1] For a convolutional code, with a free distance d_{free}, the error-correcting capability is defined as $t = \left\lfloor \frac{d_{free}-1}{2} \right\rfloor$, where $\lfloor x \rfloor$ is the integer part of x, when we consider an information sequence with a length greater than the length required to obtain d_{free}.

to belong to subsets A, B or C cannot be specified in advance since it depends on the position of these transitions, but may be determined by computer simulations as explained later. As shown in Fig. 2 (b), a burst of noise may have more than t transitions but still belong to subset B if these transitions can be decoded without error by the VA. Similarly, (T+1) non-successive transitions do not necessarily lead to a correct path loss and hence, the burst may belong to subset A.

An error-event can occur with a category 2 burst (subset B in Fig. 2 (a)) or with a category 1 burst (subset C). From the theorem on total probability, the probability of error-event is thus given by

$$P(E) = P(E/V)P_V + P(E/L)P_L \tag{3}$$

where

P_V=Probability that the burst of noise belongs to category 1

$P(E/V)$=Probability of an error-event given the burst of noise belongs to category 1

P_L=Probability that the burst of noise belongs to category 2 (correct path lost and no error with VA)

$P(E/L)$=Probability of an error-event given the burst of noise belongs to category 2.

$P(E/V)$ is the error probability of an optimum decoding algorithm, such as the VA. Let $P(E/V) = P(E)_{opt}$. Bounds on $P(E)_{opt}$ are well known [1]-[3].

Since the two categories are disjoint we have

$$P_V + P_L = 1 \tag{4}$$

Using (4), (3) becomes

$$\begin{aligned} P(E) &= P(E)_{opt}(1 - P_L) + P(E/L)P_L \\ &= P(E)_{opt} + P_L\Big(P(E/L) - P(E)_{opt}\Big) \end{aligned} \tag{5}$$

But

$$P(E/L) = 1 \tag{6}$$

since an error-event will occur with certainty if the correct path is lost. Furthermore, for Eb/No large enough we can write

$$P(E)_{opt} \ll 1 \tag{7}$$

Hence, (5) can be written as

$$P(E) \approx P(E)_{opt} + P_L \tag{8}$$

where $P(E)_{opt}$ is due to the code and P_L is due to the decoding algorithm.

From (8), P_L represents the degradation of the performance of the AVA with respect to the VA. Determining analytically the exact performance degradation due to the suboptimality of a decoding algorithm is a complex problem, which remains open. The approach proposed here is not a rigorous and complete analysis but an approximation examining for which values of T, P_L may be neglected. We recall that P_L is the probability of discarding the correct path without exceeding the error-correcting capability of the code, that is, P_L is the probability of losing the correct path and not making any error if the VA had been used. Let P_{loss} be the probability of correct path loss with the AVA and $P_{correct/loss}$ the probability that the VA is correct given the discarding of the correct path by the AVA. Then, P_L is the product of these two probabilities

$$P_L = P_{correct/loss} P_{loss} \tag{9}$$

Assuming the maximum number of survivors N_{max} is equal to 2^{K-1} we now determine P_{loss} for a Binary Symmetric Channel (BSC). The correct path is discarded or lost if the BSC is affected by more than (T+1) consecutive transitions where T is the discarding threshold value. However, if the n received symbols are affected by i non-consecutive transitions, T+1≤i≤n, the number $\alpha_{i,n}$ of combinations of i transitions provoking the discarding of the correct path must first be determined by computer search. This search is performed as follows. First, n is set to its initial value, i; the $\binom{n}{i}$ possible combinations of i transitions in n symbols are generated and the number $\alpha_{i,n}$ of combinations causing the loss of the correct state is determined, $\alpha_{i,n} \leq \binom{n}{i}$. The value of n is incremented and the process is repeated until $\alpha_{i,n} = 0$, i.e., the i transitions are so distant that they do not cause a loss of the correct path. Let n_i be the last value of n for which $\alpha_{i,n}$ is nonzero. It is conjectured that if n is increased beyond n_i, we still have $\alpha_{i,n} = 0$ and thus, it is unnecessary to increment n further. Hence, over a BSC with channel transition probability p, the probability of path loss due to i transitions is given by

$$P_i = \sum_{j=i}^{n_i} \alpha_{i,j} p^i (1-p)^{j-i} \tag{10}$$

In a good channel with a small transition probability p, $(1-p)^{j-i}$ can be approximated by 1. Therefore, (10) becomes

$$P_i \approx \sum_{j=i}^{n_i} \alpha_{i,j} p^i \approx \alpha_i p^i \tag{11}$$

where $\alpha_i = \sum_{j=i}^{n_i} \alpha_{i,j}$.

$P_{correct/loss}$ is the probability that the VA can correct the channel transitions having caused the loss of the correct path with the AVA. The α_i previous distinct sequences of channel transitions are generated again and the number of sequences among them that can be decoded without error by the VA is denoted β_i. For i transitions

$$P_{correct/loss_i} = \frac{\beta_i}{\alpha_i} \tag{12}$$

Using (9) the degradation caused by i transitions is

$$P_{L_i} = P_i P_{correct/loss_i} \approx \left(\alpha_i p^i\right) \frac{\beta_i}{\alpha_i} \approx \beta_i p^i \tag{13}$$

Since the number of transitions we have to consider may vary from (T+1) to ∞, the total degradation is given by

$$P_L = \sum_{i=T+1}^{\infty} P_{L_i} \approx \sum_{i=T+1}^{\infty} \beta_i p^i \tag{14}$$

However, the number β_i of sequences affected by i transitions that the VA can correct tends towards 0 as i increases because if t is the error-correcting capability of the code, the probability that the VA can correct (t+k) transitions, k=1, 2, ... , decreases rapidly with k. Hence, since β_i tends towards 0 when i>t+1, we can write

$$P_L \approx \sum_{i=T+1}^{t+1} \beta_i p^i \tag{15}$$

Therefore, the degradation P_L is practically zero with T>t. If the threshold value T is equal to t, we obtain from (15)

$$P_L \approx \beta_{t+1} p^{t+1} \tag{16}$$

If p is small, we can set $\beta_{t+1} = 1$ and thus,

$$P_L \approx p^{t+1} \tag{17}$$

Substituting in (8) gives

$$P(E) \approx P(E)_{opt} + p^{t+1} \tag{18}$$

In conclusion, both the AVA and the VA offer approximately the same error performance if the threshold T is set to a value equal to the error-correcting capability t of the code: the degradation P_L, in the order of p^{t+1}, where p is the transition probability of the BSC, is negligible. As an example, in a BSC with a signal-to-noise ratio Eb/No=5 dB, the transition probability p is equal to 3.768×10^{-2}. If the threshold value T is equal to 4, the degradation $P_L \approx p^5$ is approximately equal to 7.6×10^{-8} indicating that the AVA is quasi-optimum with a threshold value T equal to t. Computer simulations presented in Section IV will confirm this result.

IV. SIMULATION RESULTS

The performance of the AVA over Additive White Gaussian Noise channels has been simulated on Sun workstations. Bit error probability and complexity, defined as the average number of survivors per decoded bit are first considered over a Binary Symmetric Channel (BSC) and then, over a 3-bit quantized channel. The optimal free distance, rate 1/2 convolutional codes [1], [2] with a constraint length K, $7\leq K\leq 12$, are used. Randomly generated information bits are not divided into blocks. The length of the information sequence depends on the expected decoded error probability and can reach 250 million bits for the K=11 code at a signal-to-noise ratio Eb/No=4.0 dB over a 3-bit quantized channel. The decoding depth or path memory L [1]-[3] in all our simulations is selected to be six times the constraint length K of the code, as in the VA [13] and almost no improvement is gained by increasing further the value of L. The maximum number of survivors N_{max} is set equal to 2^{K-1} for all codes.

Binary Symmetric Channel

Fig. 3 shows the bit error probability $P(B)$ as a function of Eb/No for the VA and AVA over a BSC as the constraint length K varies from 7 to 10. Fig. 3 shows that with T=4, the AVA yields nearly the same bit error probability as the VA with the K=7 and K=8 codes which have an error-correcting capability t=4. This confirms that the performance of the AVA is similar to that of the VA if T=t. For codes having constraint lengths K=9 and K=10, keeping T=4, the performance of the AVA is slightly inferior to that of the VA because the threshold is lower than the error-correcting capability, which is equal to 5 for these codes. However, as indicated in Fig. 3, these codes provide a better error performance with the AVA than the K=7 code with the VA while requiring a lower average computational load.

In Fig. 4, the bit error probability at a signal-to-noise ratio Eb/No=5.5 dB is plotted as a function of the average number of survivors. For example, we can see from Fig. 4 that the average number of survivors is approximately 30 with T=4 and that this number varies only slightly with K. For K=8, the AVA requires on the average about one quarter of the number of survivors (26.0 instead of 128) while achieving the same probability of error as the VA. The improvement is even more important when the AVA is used with K=10: its error performance is slightly superior to that of the VA with K=9; the average number of survivors is, however, only 29.5 instead of 256 for the VA. This figure illustrates the advantage of the AVA: for the same bit error probability the AVA requires a smaller average number of survivors than the VA. Furthermore, because of the adaptive nature of the AVA, this algorithm is even more efficient when the channel is good. As the signal-to-noise ratio increases, the AVA's average number of survivors decreases whereas for the VA, the number remains constant. As a consequence, the higher the signal-to-noise ratio, the more advantageous it is to use the AVA instead of the VA.

Since the computational effort is affected mostly by the threshold value and only slightly by the constraint length of the code, Fig. 5 indicates which code should be used for a given threshold or a given computational effort. With T=4, the K=10 code gives the best probability of error. Using the more powerful K=12 code with the same threshold results in a worse bit error performance since the threshold is now too low for this code. Hence, for a given threshold value, there is an optimum code choice and a code with a longer constraint length does not improve the bit error probability: if the threshold is too low, the resynchronization time is longer and an error-event may contain more bit errors.

Fig. 5 also allows the determination of the threshold value required in order to achieve a given bit error probability at a given Eb/No value. Suppose a bit error probability $P(B) \approx 2 \times 10^{-5}$ is to be achieved at Eb/No=5.5 dB. In addition to the VA with K=9, Fig. 5 indicates two possible choices: the AVA with K=10 and T=4 or the AVA with K=9 and T=5. In the first case, the average number of survivors per decoded bit is 29.5, whereas in the second one, it is 62.8. Clearly then, the K=10 code with T=4 is the code to choose.

Soft-quantized channels

3–bit (8–level) soft quantization of the channel is performed to provide approximately a 2 dB gain in Eb/No over hard quantization [13]. Integer code symbol metrics are used instead of the log-likelihood metrics [13], [3] and the smallest metric is considered the maximum likelihood metric. For the sake of simplicity, the number of bins is not equal to (T+1) and the sorting is not exact. Simulations showed that the error performance is not affected by the number of bins when the maximum number of survivors is not limited. However, with the path remerger procedure described in Section II, a single bin increases slightly the average number of survivors. Hence, the number of bins has been arbitrarily selected to be equal to 6. State i with metric d_i is stored in bin j if $\lfloor \frac{6}{T+1} \times (d_i - d_m) \rfloor = j$, $0 \leq j \leq 5$, where $\lfloor x \rfloor$ is the integer part of x, d_m is the maximum likelihood metric at level l–1 and T is the value of the discarding threshold.

The appropriate value of the discarding threshold for a 3–bit quantized channel is determined by computer simulations. Fig. 6 shows the bit error probability of two different codes under two different Eb/No values as a function of the threshold value T. As for hard decision, there is one value of the threshold beyond which the error performance is not improved anymore even if the threshold were increased further as shown in Fig. 6; the optimum performance achieved by the

AVA is then very close to that of the VA as illustrated in Fig. 6 where the performance of the VA is represented by two points, corresponding to K=11 and K=7. As expected from our previous results with hard quantization, the threshold values should be larger for a longer constraint length code. For the K=11 code the performance of the AVA with $24 \leq T \leq 26$ is already very close to that of the VA and therefore, a threshold value between 24 and 26 may constitute an adequate choice depending on the desired error performance and decoding complexity. Incrementing T by one yields approximately a 20% increase in the average number of survivors.

Fig. 7 compares the bit error rates of the AVA and the VA as a function of Eb/No for codes varying from K=7 to K=11. It is seen that for the K=7 code, the AVA with a threshold value T=24 gives almost the same error performance as the VA, demonstrating that, as expected, the AVA can approach the optimum error performance while requiring a much smaller number of survivors. As an example, at 3.5 dB, the AVA requires an average number of 29.5 survivors for the K=7 code while the VA requires 64. For the K=11 code and a threshold T=26 the AVA, with a lower average number of survivors (41.6 at 4 dB), can achieve a gain of 1dB approximately over the VA with a K=7 code at $P(B) = 10^{-6}$; increasing the value of T by one from 24 to 26 for the K=11 code has resulted in an improvement in Eb/No of 0.1 dB approximately.

Considering the probability of error-event, the gain is slightly larger and can reach about 1.1 dB at $P(E)=10^{-6}$, as shown in Fig. 8. The improvement of the gain with the error-event probability over that with the bit error probability can be explained as follows. When using a code with a longer constraint length for the same threshold and Eb/No values, the number of error-events is reduced, resulting in a lower error-event probability. However, the number of erroneous bits in an error-event increases with K in general. Furthermore, with the AVA an error-event caused by a correct path loss may contain many erroneous bits due to the recovery time, which for a given threshold is longer as K increases, leading to a larger bit error probability. As an example, Table 1 shows some simulations results for specific values of K, Eb/No and T. It can be seen that at Eb/No=3.5 dB and 4.0 dB the K=12 code achieves a lower error-event probability but a higher bit error probability than the K=11 code. We also notice the decrease in the average number of survivors as Eb/No improves. Through extensive simulations, it has been observed that for a given threshold, error-event probability improves with the constraint length K of the code, while the bit error probability improves with K up to a maximum value of K and then degrades as K is increased further. In applications such as hybrid ARQ systems, where the error-event probability is more important than the bit error performance, a lower error-event probability can be achieved by increasing K.

Comparing the performance of the AVA with soft and hard quantizations, we notice that soft quantization provides more flexibility and more alternatives for the discarding threshold, allowing a somewhat "finer tuning". As an example, with hard quantization and a K=10 code, increasing the threshold value from 4 to 5 involves more than a doubling of the average number of survivors. On the other hand, with 3-bit quantization and a K=11 code, increasing the threshold value from 24 to 25 improves the bit error probability, while increasing only slightly the average number of survivors. Consequently, a larger gain over the VA can be achieved using soft instead of hard quantization. It is shown in Section V that the required buffer is modest, confirming that the threshold value T=25 can be used with an AVA decoder performing 64 computations per decoded bit.

In conclusion, we have shown in this section that the AVA can offer practically the same error performance as the VA while requiring on the average a much smaller number of survivors. Hence, for a given average number of survivors, a longer code can be used with the AVA. The average number of survivors gives a good idea about the overall decoding effort. However, since this number varies during the decoding, input buffer requirements must now be examined.

V. COMPUTATIONAL VARIABILITY

Since the number of survivors required to decode one bit with the AVA may vary, a buffer is required at the input of the decoder. Let μ be the decoder speed factor, that is, the number of computations which the decoder can perform in the interarrival time of the information bits. If the current number of survivors that must be extended is larger than μ, then a newly received branch must be stored in an input buffer until the decoder has the time to process it. However, unlike sequential decoding, the computational variability of the AVA does not come from backing up in the trellis but from extending a varying number of paths, all of the same length. It is therefore far less severe than for sequential decoding.

Before considering the required size of the input buffer, first the distribution of computation has to be determined. It is known that sequential decoding has a Pareto distribution of computation due to the combined result of two opposing effects [10], [11]: the probability of a burst of noise of length n in a memoryless channel decreases exponentially with n, whereas the number of computations required by the decoder due to such a burst increases exponentially with n.

By a similar argument, we show that over a memoryless channel the distribution of the computational effort of the AVA is exponential. For such a channel, the probability of a channel burst of length n decreases exponentially with n. However, the number of computations caused by that burst does not increase exponentially since backtracking is not allowed by the algorithm and the number of survivors is limited to a finite value N_{max}, $N_{max} \leq 2^{K-1}$. The combined effect of these two behaviors results in an exponential distribution as confirmed by computer simulations.

Fig. 9 shows simulations results of the computational distribution P(C≥N) for K=8 over a BSC, where C is the required number of computations to decode one bit and N, a given number. The curve is plotted on semi-log axes and we see that P(C≥N) can be approximated by a straight line, corresponding to an exponential distribution.

The distribution of computation being exponential, we now relate it to the distribution of the size of the queue in the input buffer $P(q_l \geq s_l)$, where q_l is the queue size in the buffer at trellis level l, that is, the number of received branches stored in the buffer when the decoder is at trellis level l and s_l, a given value. Code symbols are assumed to arrive periodically from the channel at a constant rate and processed according to a First-Come First-Served (FCFS) rule. The size of the queue in the buffer is measured periodically at the instant of arrival of each new branch from the channel. The queue size at trellis level l depends on the number of computations C_0 required by the first branch in the buffer. If the buffer is empty at level l, C_0 is, of course, the number of computations required by the current branch. If $C_0 > \mu$, where μ is the decoder speed factor, then it is clear that the queue size increases by one received branch. Hence, we have

$$P(q_l \geq s_l) = P(C_0 \geq N) \tag{19}$$

where N is a number which depends on μ and s_l. Since the distribution of computation is exponential, the distribution of the size of the queue in the input buffer is also exponential.

The probability of buffer overflow is now considered. To simplify the analysis, it is assumed that the buffer is empty at trellis level l. Let B be the buffer size, C_0 the number of computations required to decode the received branch at level l, which is the first branch in the buffer and μ the decoder speed factor. Then, the probability of buffer overflow caused by branch l is given by

$$\begin{aligned} P_{overflow} &= P(C_0 > B\mu + \mu) \\ &= P(C_0 > \mu(B+1)) \end{aligned} \tag{20}$$

Thus, if $C_0 > \mu(B+1)$, overflow occurs at level $(l+B)$. From (20), the probability of overflow decreases exponentially with μ and B since C_0 is exponentially distributed; hence, unlike a Pareto distribution [10], [11], it can be made as small as desired by increasing μ or B. Furthermore, since C_0 decreases rapidly with the noise, the probability of buffer overflow also decreases in the same fashion with the noise. Therefore, unlike sequential decoding, overflows should not present a problem with the AVA, as verified by computer simulations discussed below.

In our simulations, in addition to the probability of error and the number of survivors, the size of the queue in the input buffer is examined at each trellis level. When an information bit is generated, the queue size is incremented by one branch at the time of arrival of the new branch. We assume that μ=64, i.e., the decoder can process 64 survivors at each trellis level like a K=7 Viterbi decoder. If the first branch in the buffer requires C computations, C>μ, then at the next trellis level the queue size is incremented but the received branch is not processed until the current branch is decoded. If, on the other hand, C<μ, then the queue size is decremented by one branch and the decoder processes the next branch in the buffer. If this branch requires fewer than (μ-C) computations, the queue size is again decremented, and so on. Computer simulations results over the BSC shown in Table 2 indicate that for K=10, when decoding 100 millions bits at $E_b/N_o = 5.5$ dB, the maximum queue size of the buffer is only 456 branches. For the

K=11 code the queue reached a maximum size of 2049 branches and hence, a larger buffer is required. However, Fig. 5 showed that this code does not provide a better error probability than the K=10 code. Therefore, this code should not be used with the AVA and T=4, not only because of the required input buffer but also because of the error performance. For a given threshold value, which determines the decoding complexity, the code selection is governed by error performance and not by buffer size.

From the queue size at each trellis level, the queue size distribution P(q>s) has been determined by simulations and is illustrated in Fig. 10 for the BSC and in Fig. 11 for a soft quantized channel. First, we notice that the distribution is well approximated by a straight line on a semi-log graph, indicating an exponential distribution. The tail of the distribution, which diverges from the straight line, is unreliable and may be ignored since it corresponds to a queue size for which insufficient data have been obtained. In Figs. 10 and 11 different codes and parameters are used. It is seen that the slopes of the curves grow as Eb/No is increased. We also notice from Fig. 11 the effect of the threshold value T on the queue size distribution: as T increases, a larger buffer is required. It should be noted that for K=11 and T=26, which provides a gain of 1 dB over the VA with K=7 at P(B)=10^{-6}, the buffer requirements are really small: the probability that the size of the queue exceeds 650 branches is about 10^{-5}. By extending this straight line further, the probability would be 10^{-10} for 1500 branches, implying that the probability of buffer overflow can be made as small as desired, as expected. From these results, it can be concluded that an AVA decoder performing μ=64 computations per trellis level like a K=7 Viterbi decoder can use a K=11 code and achieve at P(B)=10^{-6} a 1 dB improvement in Eb/No over the Viterbi decoder: the number of survivors is variable but, since on the average it is smaller than 64 and the variability is small, only a modest input buffer is required. Under noisy channel conditions, the number of survivors increases and incoming branches may have to be stored temporarily in the input buffer. However, except for periods of intense channel noise, the required decoding effort is lower than the computational capability of the decoder and thus, the decoder is able to process the branches accumulated in the buffer, quickly eliminating any backlog.

VI. CONCLUSION

A reduced-complexity adaptive decoding algorithm for convolutional codes which can inherently recover the lost correct state has been presented. With properly selected parameters the bit error probability can match that of the VA but at a substantially smaller average number of survivors. These advantages are obtained at the expense of a modest decoding delay and the requirement of a modest input buffer. Since the quality of a memoryless communications channel is not constant, a decoding algorithm performing a variable number of computations and adapting its effort to the quality of the channel may be more efficient than an algorithm with a constant number of computations. The decoding effort spent by the latter algorithm will be either too high when the channel is quiet or too low for a span of very heavy noise. For a given complexity, the AVA offers a better performance than the VA both in terms of error rate and decoding time if the channel is not too noisy as verified by our simulations under the same conditions. The computational variability is small and hence does not create an input buffer overflow problem. Furthermore, simulations over Rayleigh fading channels have shown that the AVA also provides advantages over the VA in this environment. At a bit error probability of 10^{-5} and with a normalized fading rate $F_DT=0.1$, a gain of up to 2 dB can be achieved with the AVA and a K=12 code without increasing the average number of survivors compared to the VA with a K=7 code [14]. Finally, a VLSI implementation of an AVA decoder with parallel processors has been proposed [14].

REFERENCES

[1] V. Bhargava, D. Haccoun, R. Matyas, and P. Nuspl, *Digital Communications by Satellite*. New York: John Wiley, 1981.

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

[3] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 751–772, Oct. 1971.

[4] D. Haccoun and M. Ferguson, "Generalized stack algorithms for the decoding of convolutional codes," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 638–651, Nov. 1975.

[5] T. Aulin, *A New Trellis Decoding Algorithm — Analysis and Applications*. Tech. report Nr. 2, Chalmers Univ. of Technology, Göteborg, Sweden, Dec. 1985.

[6] C. F. Lin and J. B. Anderson, "M-algorithm decoding of channel convolutional codes," *Proc. Princeton Conference on Information Science and Systems*, pp. 362–366, Mar. 1986.

[7] J. B. Anderson, "Limited search trellis decoding of convolutional codes," *IEEE Trans. Inform. Theory*, vol. IT-35, pp. 944–955, Sep. 1989.

[8] S. J. Simmons, "Breadth-first trellis decoding with adaptive effort," *IEEE Trans. Commun.*, vol. 38, No. 1, pp. 3-12, Jan. 1990.

[9] C. F. Lin, *A Truncated Viterbi Algorithm Approach to Trellis Codes*. PhD thesis, Rensselaer Polytechnic Institute, Troy, NY, Sep. 1986.

[10]J. E. Savage, "The distribution of the sequential decoding computation time," *IEEE Trans. Inform. Theory*, vol. IT-12, pp. 143–147, April 1966.

[11]I. M. Jacobs and E. R. Berlekamp, "A lower bound to the distribution of computation for sequential decoding," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 167–174, April 1967.

[12]D. Haccoun, "Décodage séquentiel des codes convolutionnels de taux de codage élevés," *Traitement du Signal*, vol. 4, n^o 6, pp. 471–478, 1987.

[13]J. A. Heller and I. M. Jacobs, "Viterbi decoding for satellite and space communication," *IEEE Trans. on Commun. Technology*, vol. COM-19, pp. 835–848, Oct. 1971.

[14]F. Chan, *Décodage simplifié: algorithme adaptatif pour codes convolutionnels et technique de perforation pour modulation codée*. PhD thesis, Ecole Polytechnique, Montréal, Nov. 1994.

K	Eb/No	T	P(E) (Number of error-events)	P(B) (Number of bit errors)	Avg. Number of survivors
7	3.5 dB	24	3.75×10^{-5} (75)	1.86×10^{-4} (373)	29.5
8	2.6 dB	24	1.82×10^{-4} (182)	1.13×10^{-3} (1127)	57.0
11	2.6 dB	24	4.40×10^{-5} (88)	3.85×10^{-4} (771)	93.2
11	3.5 dB	24	1.48×10^{-6} (148)	1.54×10^{-5} (1545)	40.4
		25	1.25×10^{-6} (125)	1.10×10^{-5} (1098)	50.3
		26	1.16×10^{-6} (116)	8.36×10^{-6} (836)	62.3
11	4.0 dB	24	2.80×10^{-7} (70)	3.20×10^{-6} (801)	27.6
		25	2.20×10^{-7} (55)	2.01×10^{-6} (503)	33.9
		26	1.76×10^{-7} (44)	1.29×10^{-6} (323)	41.6
		27	1.52×10^{-7} (38)	1.06×10^{-6} (264)	51.0
12	3.5 dB	25	8.70×10^{-7} (87)	1.32×10^{-5} (1321)	58.7
12	4.0 dB	25	2.12×10^{-7} (53)	4.45×10^{-6} (1112)	39.0

Table 1 Simulation results of the AVA at some different signal-to-noise ratios over a 3-bit quantized channel

K	Number of bits decoded	Size of queue in buffer	
		Avg. size	Max. size
8	4×10^6	0.47	46
9	15×10^6	1.07	143
10	100×10^6	3.08	456
11	50×10^6	4.07	2049

Table 2 Average size and maximum size of the queue in the input buffer with T=4, μ=64 and Eb/No=5.5 dB over a BSC — the queue size is given as the number of received branches

K	Eb/No	T	Size of queue in buffer	
			Avg. size	Max. size
8	2.6 dB	24	32.7	373
11	3.5 dB	24	16.7	1576
		25	48.2	1217
		26	641.8	6191
11	4.0 dB	24	2.6	1015
		25	5.5	1245
		26	12.7	864
		27	36.3	1239

Table 3 Average size and maximum size of the queue in the input buffer with μ=64 over a 3-bit quantized channel — the queue size is given as the number of received branches

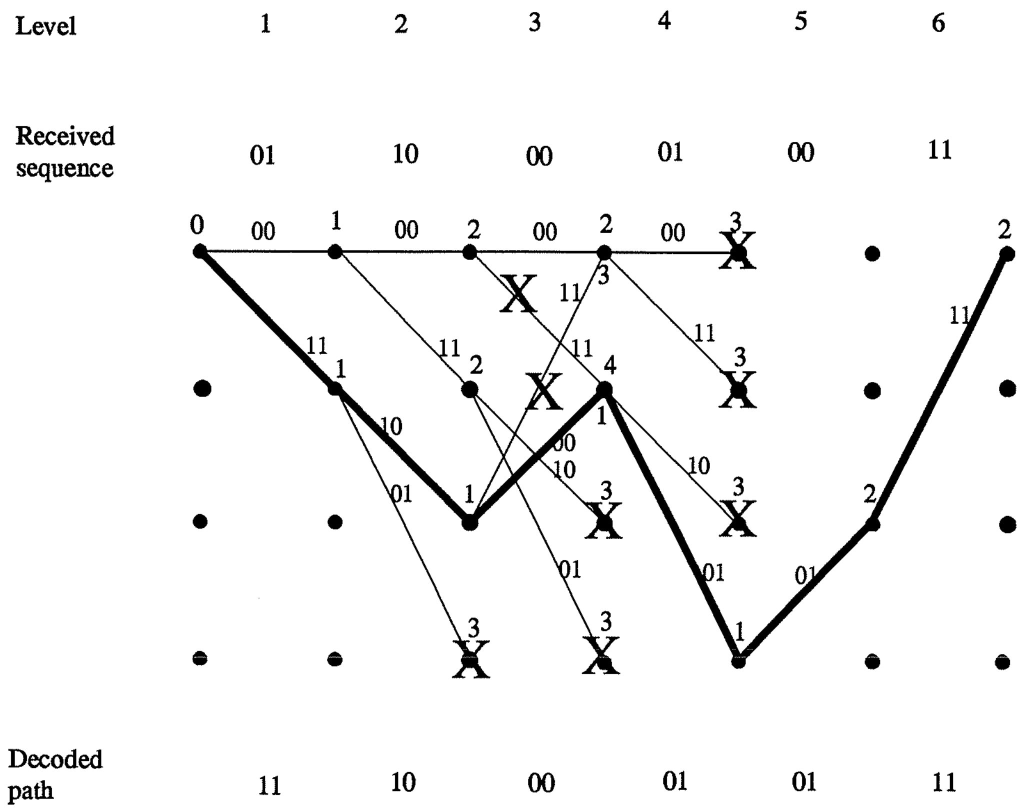

Fig. 1. Decoding a sequence with the Adaptive Viterbi Algorithm (AVA) and a discarding threshold T = 1. X denotes a discarded path. The last two branches correspond to the tail of the sequence.

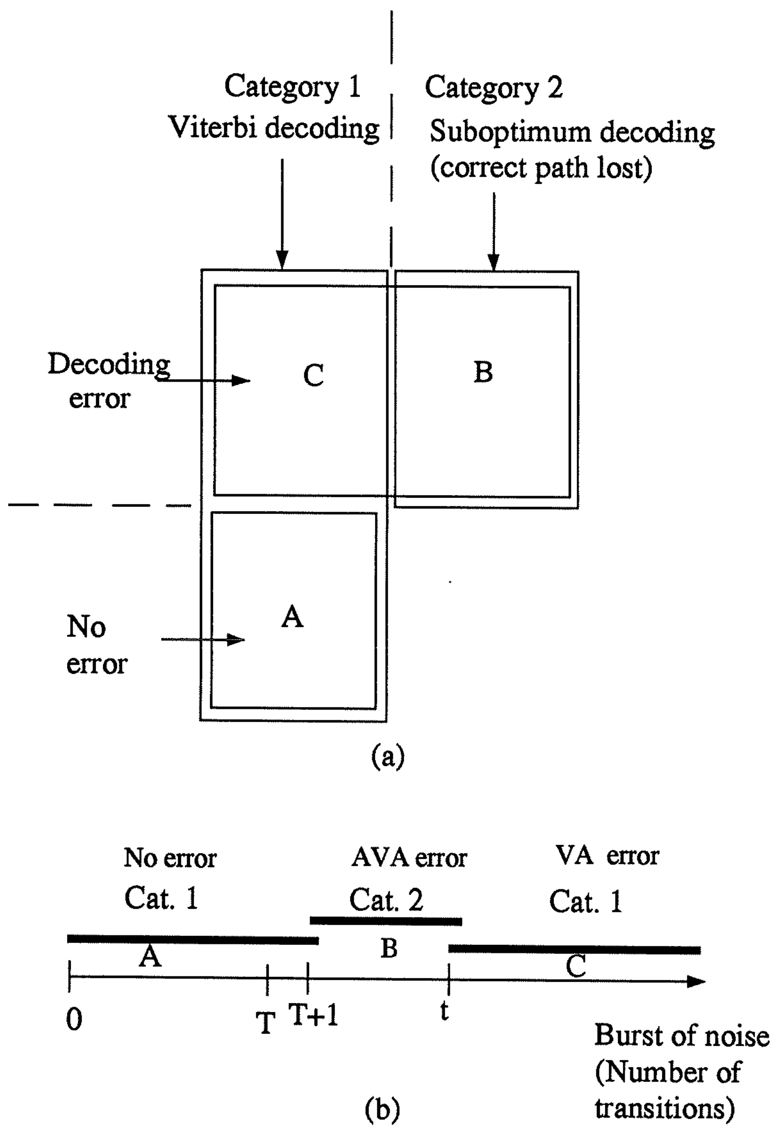

Fig. 2. (a) Classification of bursts of noise; (b) Example with T<t

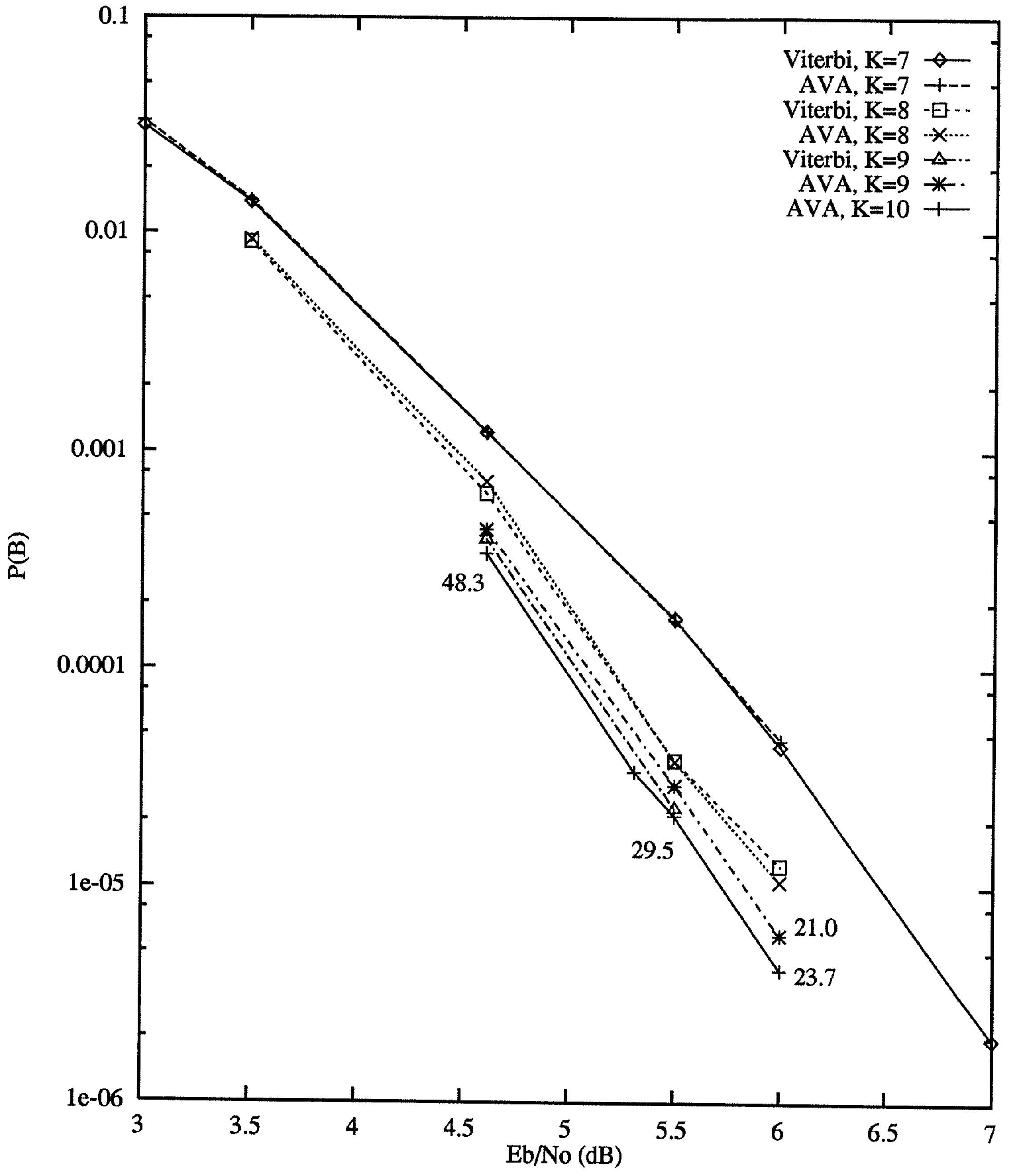

Fig. 3. Comparison of the bit error probability over a BSC of the Viterbi algorithm and the Adaptive Viterbi Algorithm (AVA) with maximum number of survivors $N_{max} = 2^{K-1}$ and threshold T = 4; average numbers of survivors are indicated next to a few points

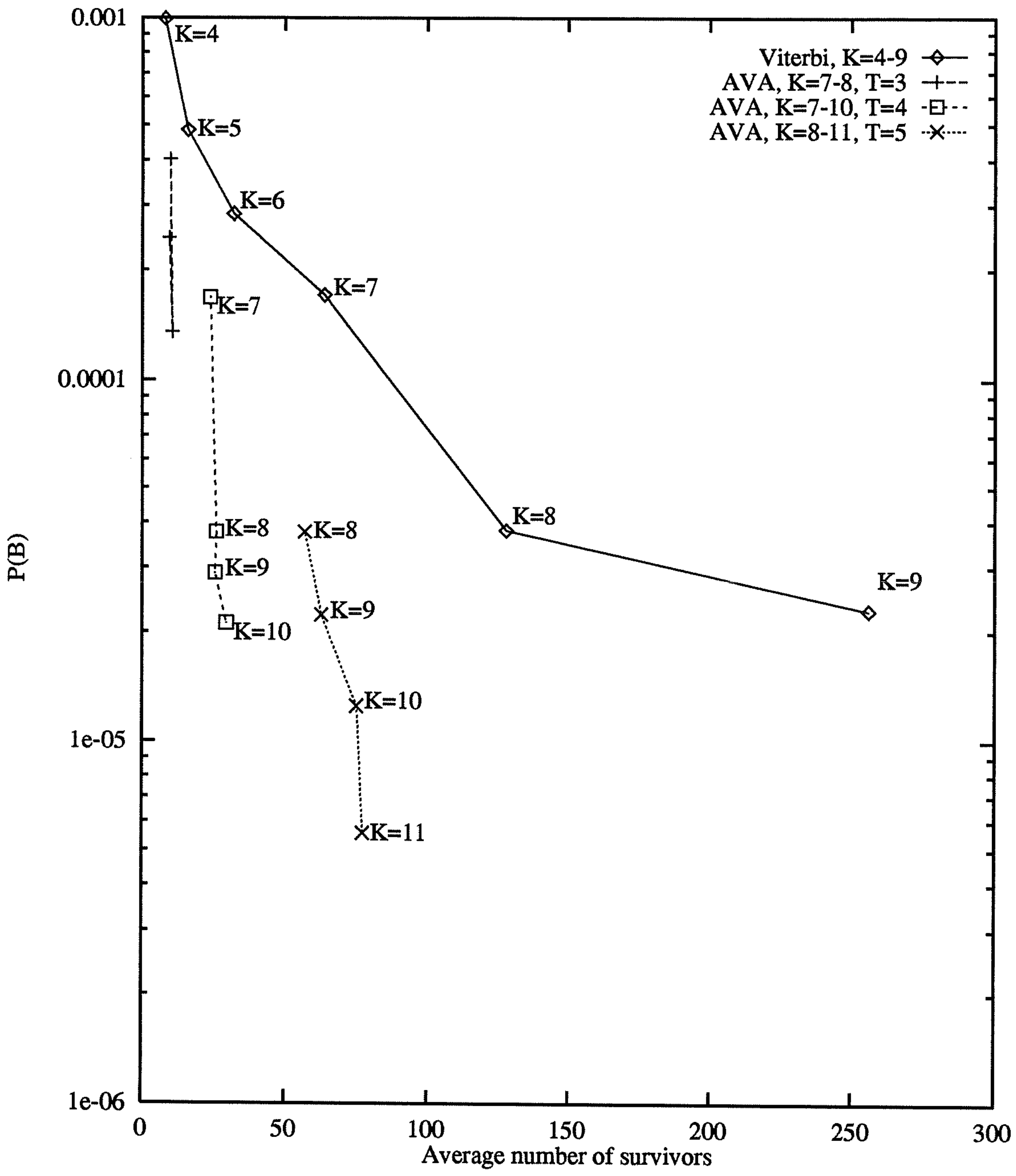

Fig. 4. Bit error performance versus complexity, defined as average number of survivors — Eb/No = 5.5 dB, BSC

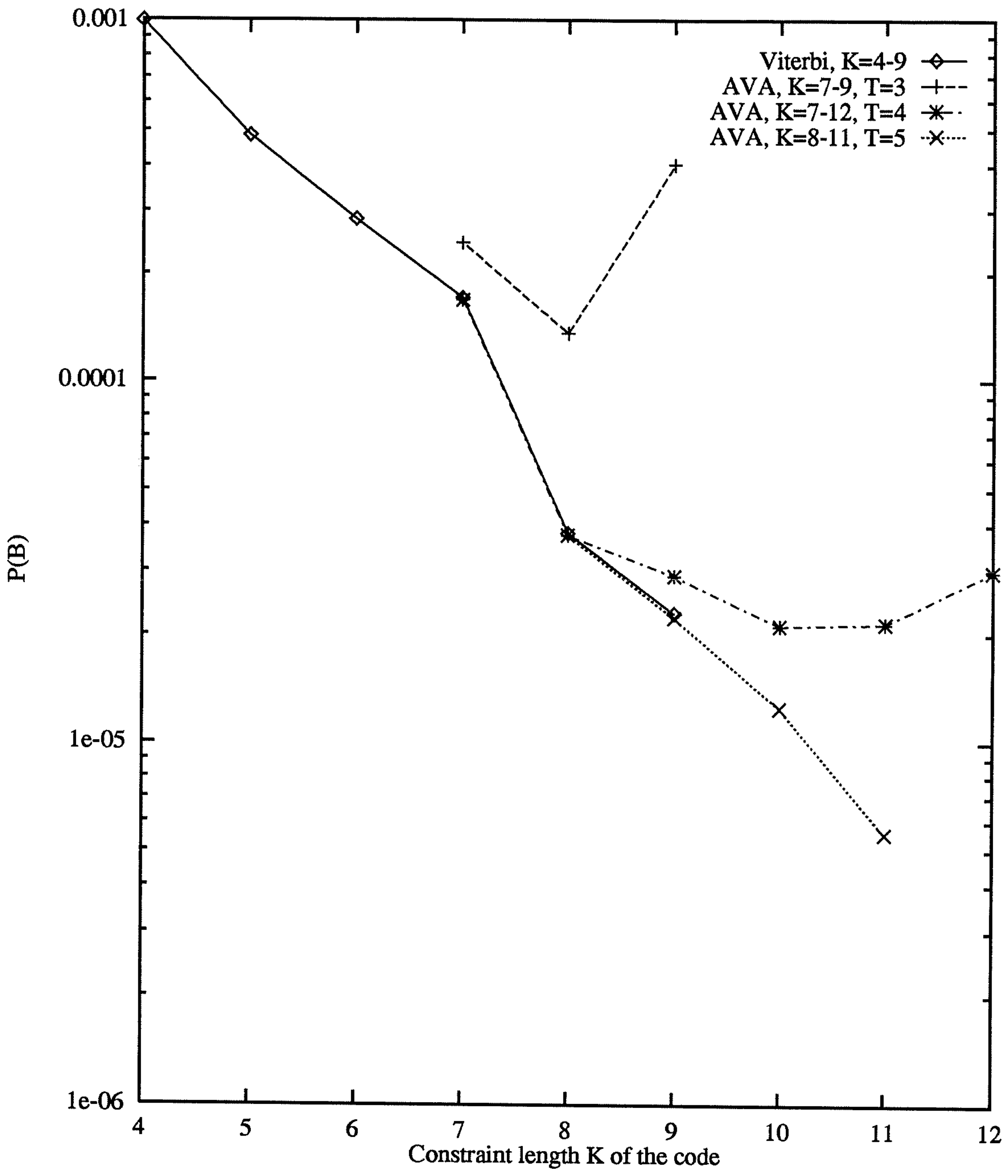

Fig. 5. Bit error performance versus constraint length of the code for the VA and the AVA over the BSC at Eb/No = 5.5 dB

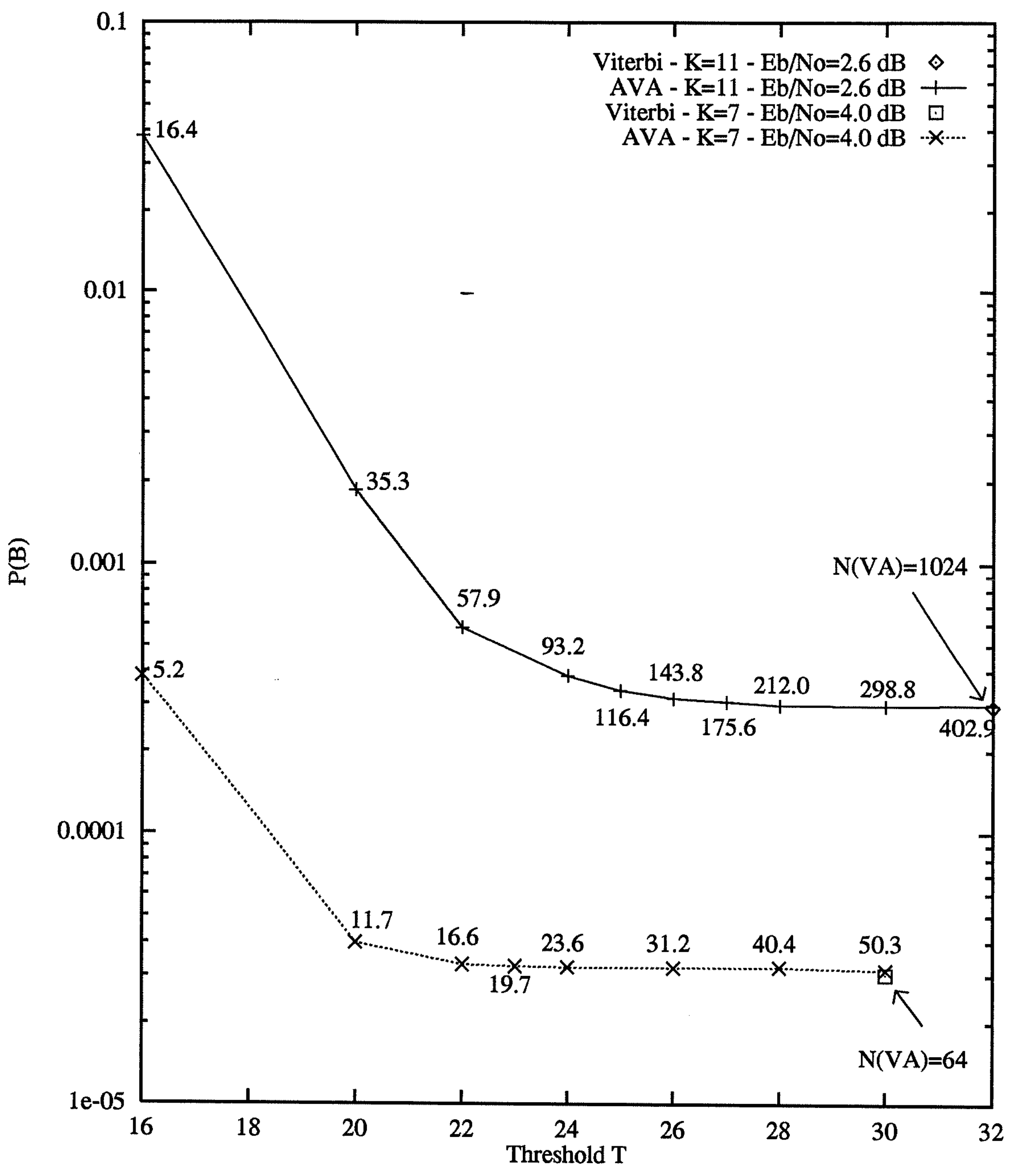

Fig. 6. Influence of T on the bit error probability of the AVA for two different codes over a 3–bit quantized channel — the average number of survivors of the AVA is given next to each point and the number of survivors of the VA is indicated by N(VA)

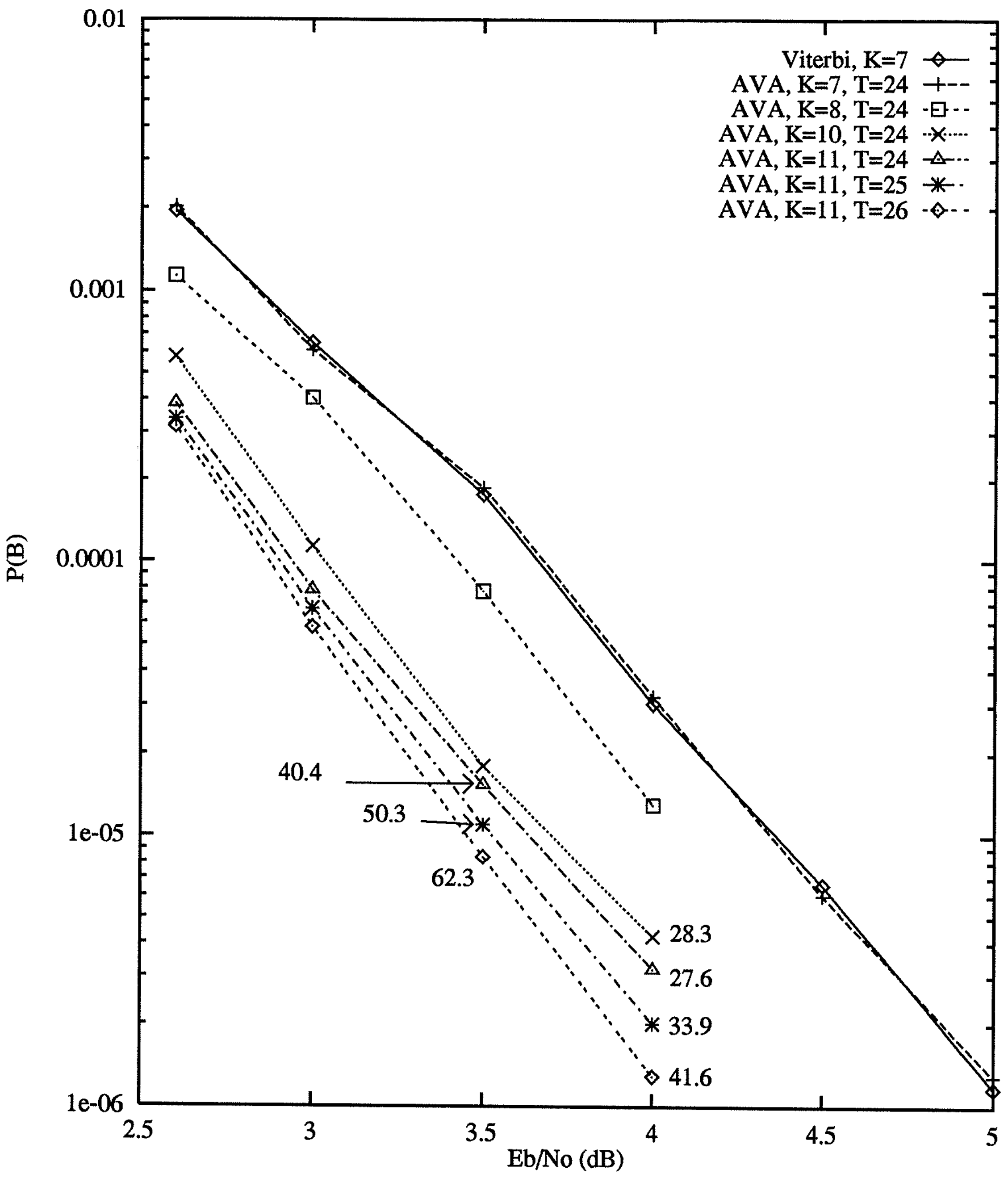

Fig. 7. Bit error probability of the AVA over a 3-bit quantized channel; average numbers of survivors are indicated next to a few points

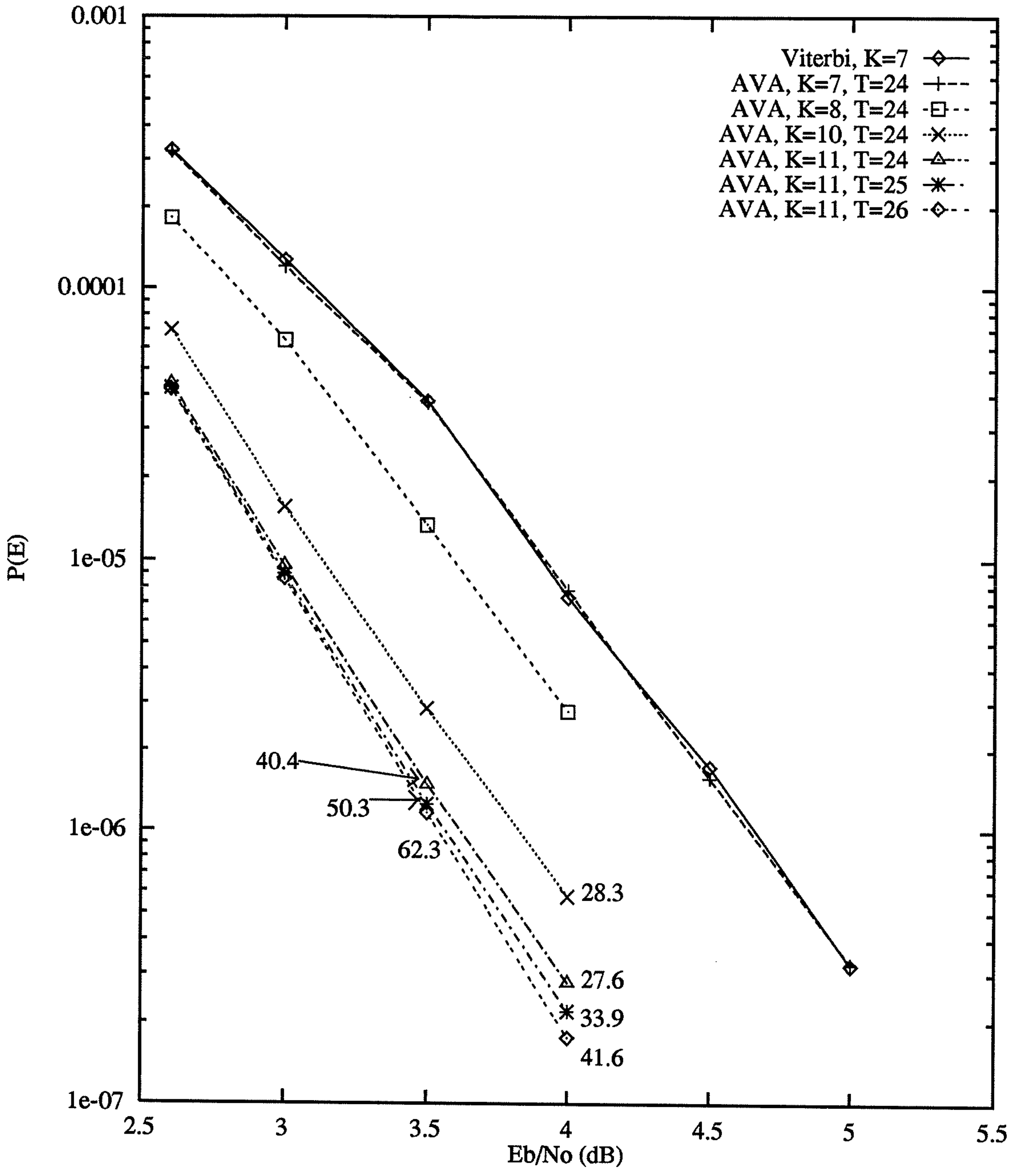

Fig. 8. Error-event probability of the AVA over a 3-bit quantized channel; average numbers of survivors are indicated next to a few points

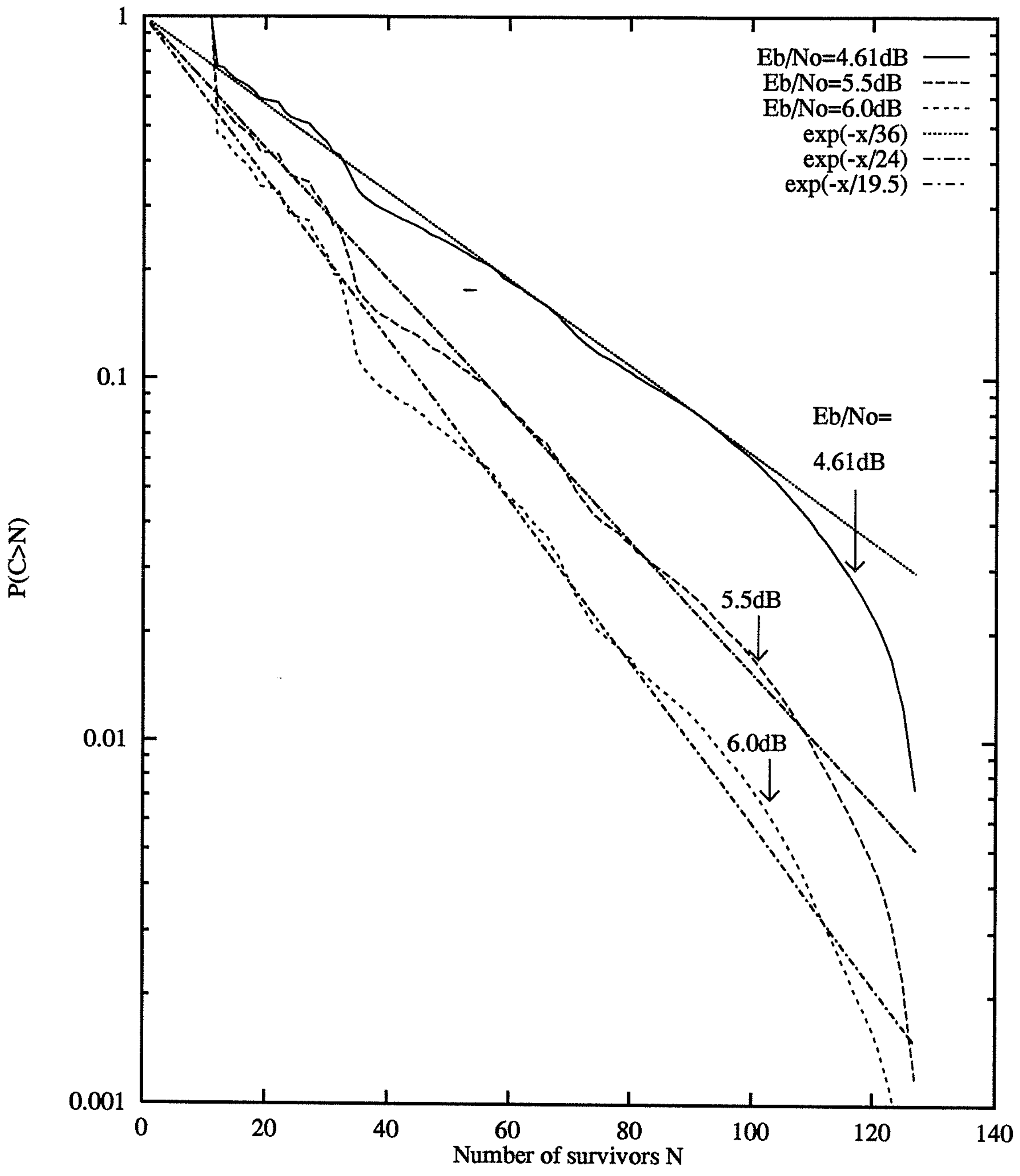

Fig. 9. Distribution of computation for the AVA using K=8 and T=4 over a BSC for Eb/No=4.61 dB, 5.5 dB and 6.0 dB

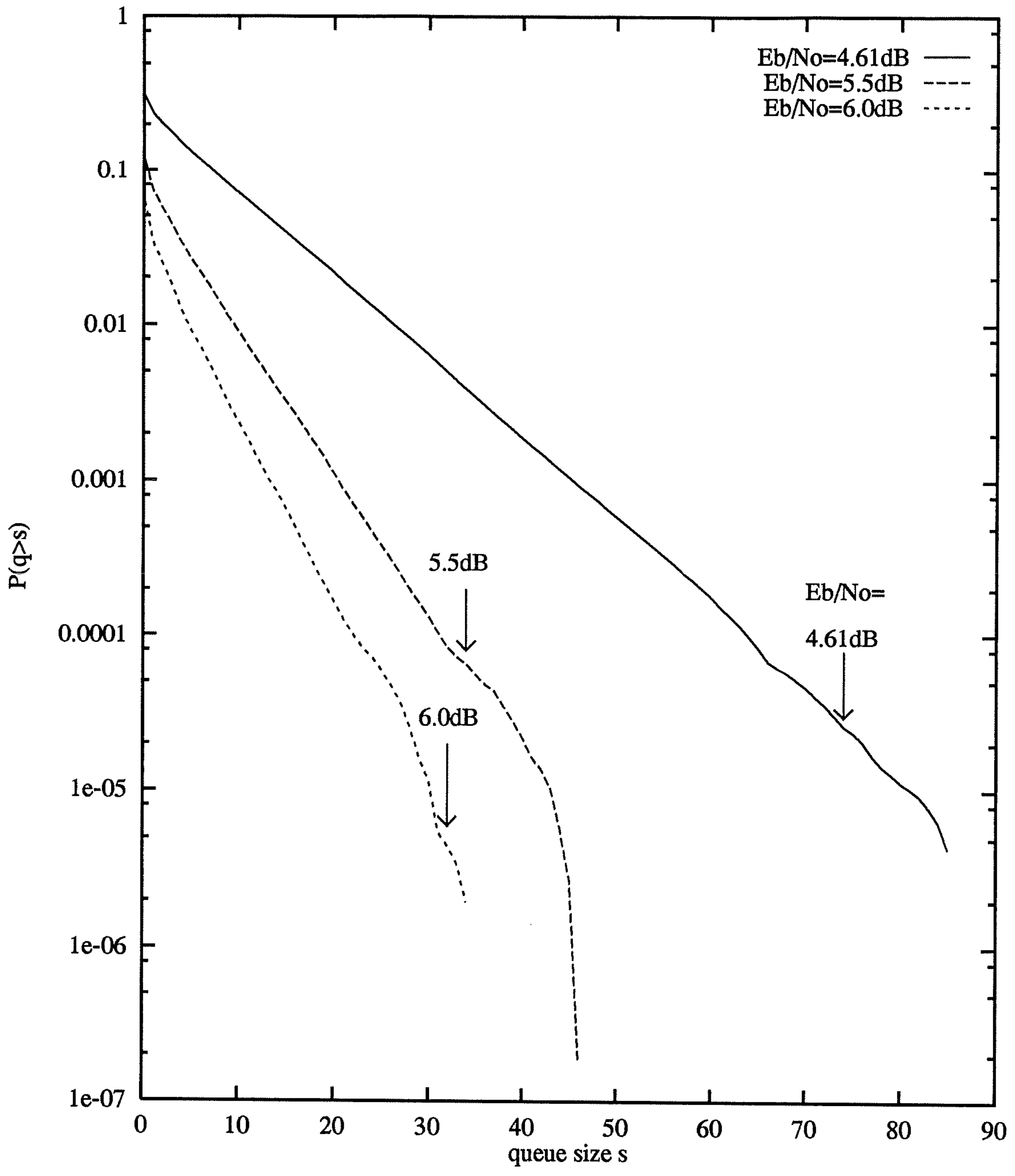

Fig. 10. Distribution of queue size of the input buffer for the AVA using K=8, T=4 and μ=64 over a BSC for Eb/No=4.61 dB, 5.5 dB and 6.0 dB

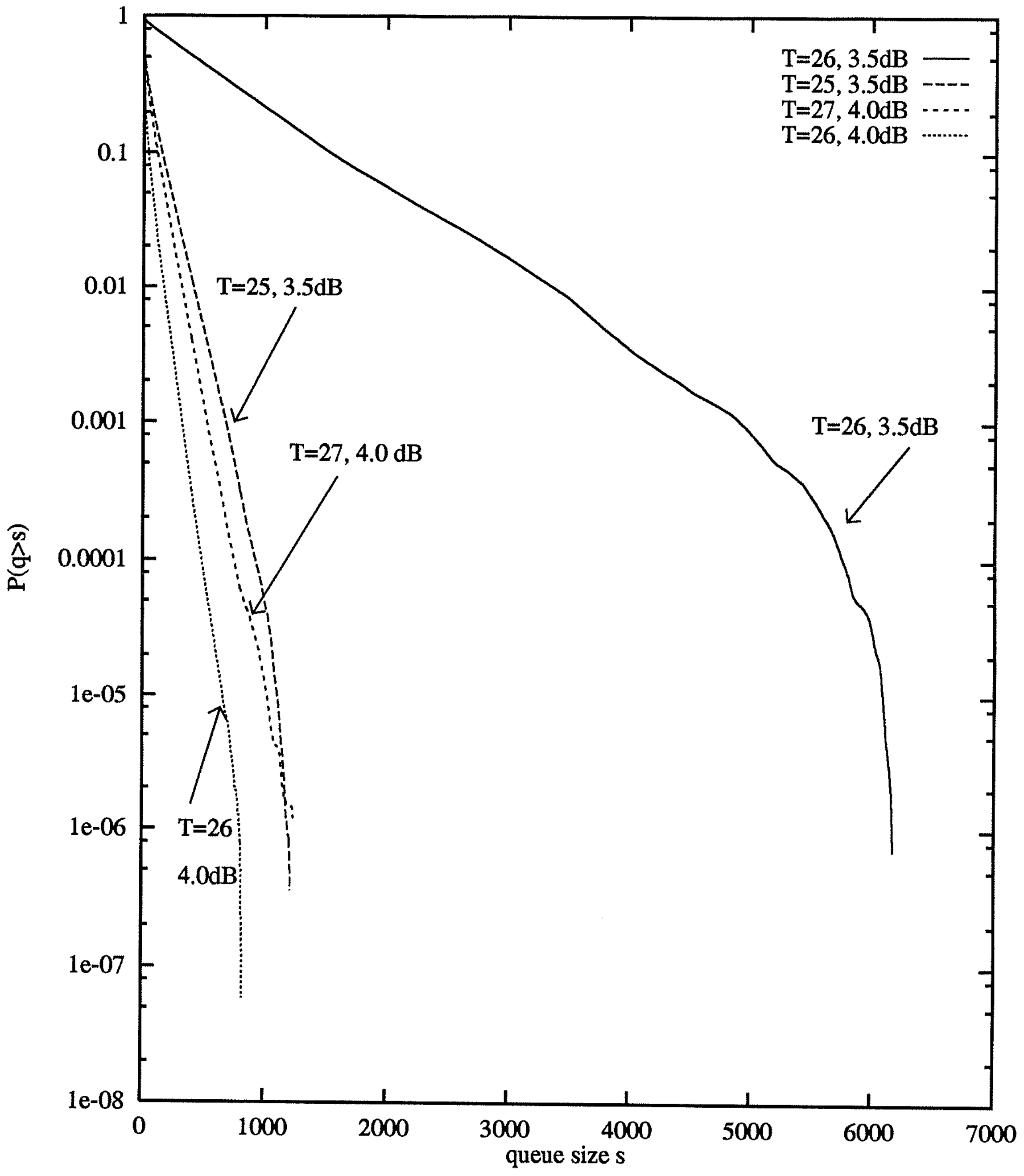

Fig. 11. Distribution of queue size of the input buffer for the AVA using K=11 and μ=64 over a 3-bit quantized channel

École Polytechnique de Montréal
C.P. 6079, Succ. Centre-ville
Montréal (Québec)
H3C 3A7