POLYPUBLIE

Sequential decoding of punctured convolutional codes by the Fano and stack algorithms : extensive simulation results

Guy Bégin, David Haccoun

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

Date	1991
Type	Rapport / Report Rapport technique/Technical Report
Version	Version officielle de l'éditeur / Published version
URL de PolyPublie PolyPublie URL	https://publications.polymtl.ca/10049/
Évaluation par les pairs Peer reviewed	
Conditions d'utilisation Terms of use	© Tous droits réservés / All rights reserved
Référence Citation	Bégin, G., & Haccoun, D. (1991). Sequential decoding of punctured convolutional codes by the Fano and stack algorithms : extensive simulation results. (Rapport technique n° EPM-RT-92-03). https://publications.polymtl.ca/10049/
Publication officielle Official publication	
Institution Institution	École Polytechnique de Montréal
Numéro de rapport Report number	EPM-RT-92-03
URL officielle Official URL	
Mention légale Legal notice	

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal
https://publications.polymtl.ca

16 MARS 1992

RAPPORT TECHNIQUE
EPM/RT-92/3

SEQUENTIAL DECODING OF PUNCTURED CONVOLUTIONAL CODES BY THE FANO AND STACK ALGORITHMS: EXTENSIVE SIMULATION RESULTS

gratuit

Guy Bégin, Ing., Ph.D.

David Haccoun, Ing., Ph. D.
Professeur titulaire

École Polytechnique de Montréal

Mars 1992

Tous droits réservés. On ne peut reproduire ni diffuser aucune partie du présent ouvrage, sous quelque forme que ce soit, sans avoir obtenu au préalable l'autorisation écrite des auteurs.

Dépôt légal, 1e trimestre 1992
Bibliothèque nationale du Québec
Bibliothèque nationale du Canada

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

Éditions de l'École Polytechnique de Montréal
École Polytechnique de Montréal
Case Postale 6079, Succursale A
Montréal (Québec) H3C 3A7
(514) 340-4000

Compter 0,10$ par page (arrondir au dollar le plus près) et ajouter 3,00$ (Canada) 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.

SEQUENTIAL DECODING OF PUNCTURED CONVOLUTIONAL CODES BY THE FANO AND STACK ALGORITHMS: EXTENSIVE SIMULATION RESULTS *

Guy Bégin † David Haccoun ‡

Abstract

The decoding of long memory high-rate punctured convolutional codes by sequential decoding algorithms is investigated. Both the stack and the Fano algorithms have been thoroughly tested through computer simulation. Both algorithms have been used with success for decoding long memory punctured codes with coding rates ranging from $R = 2/3$ to $R = 7/8$. With respect to decoding behavior and performance, the two algorithms have much in common. Error and overflow probabilities and variability of decoding effort are similar for both algorithms. With hard quantization, plateaus appear in the cumulatives of decoding effort for both algorithms. These plateaus correspond to the frequent occurence of full explorations of subtrees over a length of one period of the punctured code. Differences between the two algorithms show up as larger numbers of computations for the Fano decoder which are due to its forward and backward movement. Finally, comparing the punctured approach of decoding to the more traditional technique for high-rate codes, it is found that punctured decoders perform a larger number of simpler computations. For all but the lower rate codes, the overall decoding effort is on the average more important for the usual decoder than it is for its punctured counterpart. Several aspects of computational performance exhibit an exponential dependence with b for a usual decoder versus a linear dependence for a punctured decoder. Computational variability, error and overflow probabilities are no worse for punctured decoders than they are for normal decoders.

I INTRODUCTION

The class of high-rate punctured convolutional codes was introduced because it provides attractive solutions to the difficulties of decoding high-rate convolutional codes by the Viterbi algorithm [1]. Later, it was also recognized that a sequential decoding algorithm like the stack algorithm of Zigangirov and Jelinek [2], [3] could be used for decoding punctured codes advantageously [4]. Another strong motivation for using punctured codes with either decoding technique is the fact that they lend themselves to an easy implementation of variable-rate coding-decoding [5] [8], as well as rate-compatible coding-decoding [6].

*This work was supported in part by the Natural Sciences and Engineering Research Council of Canada, the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche du Québec and the Fonds de l'UQAM.

†Department of Mathematics and Computer Science, Université du Québec à Montréal, Montréal (Québec) Canada H3C 3P8

‡Department of Electrical and Computer Engineering, Ecole Polytechnique de Montréal, Montréal (Québec) Canada H3C 3A7

In short, a punctured convolutional code is a high-rate convolutional code derived from a low-rate original encoder by periodic deletion of some output symbols. By a careful selection of the deleting rule, called the *perforation pattern*, the specification of almost any coding rate is possible. For a punctured rate-b/v code derived from a rate-$1/v_0$ original encoder, the perforation pattern has period b bits and is conveniently described by a v_0 rows and b columns binary matrix wherein a 0 entry indicates a deleted or punctured symbol and a 1 entry indicates a transmitted symbol.

Viterbi decoding of punctured codes has been used with success for a number of years. Today, commercial convolutional codecs using punctured codes are readily available and widely used [7]. By contrast, less attention was devoted to sequential decoding of punctured codes. Recently, extensive lists of good long memory punctured codes suitable for sequential decoding have been proposed by the authors [8]–[10], opening the door for easy applications of sequential decoding for punctured convolutional codes.

The purpose of this paper is to investigate and compare the decoding of long memory punctured codes by two well known sequential decoding algorithms: the stack algorithm and the Fano algorithm. Both algorithms have been thoroughly tested through computer simulation of decoding over memoryless additive white Gaussian channels. By analysing the results, valuable information about the effects of the special structure of punctured codes on the decoding processes is obtained.

The paper is organized as follows. The main aspects of the sequential decoding algorithms are reviewed briefly in section II. After identifying the simulation hypotheses, simulation results obtained with Fano and stack decoders are presented in section III, where the main points of interest of the simulation results are discussed. In section IV, a comparison is made between the normal and the punctured approaches to decoding for both decoding algorithms. Finally, concluding remarks are provided in section V.

II SEQUENTIAL DECODING ALGORITHMS

In sequential decoding, only a portion of the entire coding tree is explored as the decoder follows the path that is locally most likely. Compared to the full exploration of Viterbi decoding, such a partial exploration reduces substantially the average work load for a given tree complexity. However, because of noise affecting the received symbols, occasionally the correct path may be abandoned in favor of a locally more likely path. The decoder then follows a number of incorrect paths (which may be quite substantial) before backtracking in the tree, searching and eventually finding the correct path again. This leads to the well known problem of computational variablility of sequential decoders [11]–[13].

Adapting sequential decoding algorithms to the decoding of punctured codes is not a complex procedure. In order to exploit the special structure of punctured codes, the operations of the algorithms must be performed on the tree of the low-rate original code. Decoding is carried out one bit at a time, regardless of the actual punctured high coding rate. The punctured symbols are taken care of by inhibiting metric evaluations where symbols have been deleted.

A Metric for punctured codes

For memoryless channels, the measure of likelihood used with sequential decoding is the Fano metric [14], which depends on the coding rate R and also on the characteristics of the channel via its transition probabilities $P(y \mid x)$. For a received channel symbol y_i corresponding to a

transmitted symbol x_i, the symbol Fano metric γ_i is

$$\gamma_i = \log_2 \frac{P(y_i \mid x_i)}{f(y_i)} - R \tag{1}$$

where the $f(\,)$ are output probabilities for the channel. This metric is additive so that the likelihood of a path is evaluated by summing the metrics of individual symbols composing it, i.e. $\Gamma = \sum \gamma_i$.

While the trellis for a classical convolutional code is regular, the trellis for a punctured code is not because the number of code symbols remaining on a branch varies from branch to branch according to the perforation pattern. Although the overall coding rate is fixed, the instantaneous coding rate varies from one branch to the other. Hence for best operation, the Fano metric values used by the decoders should reflect these variations. However, preliminary simulations conducted with several punctured codes have shown that using a single set of metric values based on the average coding rate does not entail any significant degradation in the computational variability of the decoders. Because of its inherent simplicity, this solution was retained for all the simulations conducted with both the stack and the Fano algorithms.

B Zigangirov-Jelinek algorithm

As its name implies, the stack algorithm of Zigangirov [2] and Jelinek [3] is based on a memory structure called a stack that keeps visited nodes sorted according to their cumulative metric values. At the 'top of the stack' is the current node of best accumulated metric amongst the nodes already visited. Since the basic decoder extends only the most likely path, then the only requirement for correct operation is that the best node may be retrieved from the stack; a complete ordering of the stack is therefore usually not required, and in most cases it is not practically feasible anyway. Instead, a partial ordering may be performed using either a systolic priority queue [15], [16] or a quantized stack partially sorted by a scheme of buckets [3]. Because it is widely used and easily implemented in software, this last solution was retained for the simulations of stack sequential decoding performed here.

C Fano algorithm

Contrarily to the Zigangirov-Jelinek algorithm, the Fano algorithm [14] does not have a memory structure to keep track of visited paths, but relies instead on sets of thresholds to guide its exploration of the most likely paths. The decoder proceeds forward as long as the cumulative metric of the path followed does not violate the current threshold value. When a noise span is encountered, the cumulative metric of the current best path tends to fall, leading to a threshold violation. The decoder then starts moving backward from the current node, trying to find an alternate path that may be extended forward. In noisy spans, the threshold value may need to be lowered so that a path that do not cause a violation may be found. Conversely in quiet spans, the threshold is regularly increased, keeping it as tight as possible. The decoder thus proceeds forward and backward, one node at a time, possibly extending the same nodes over and over again, until an acceptable path extending from the beginning to the end of the tree is found. For a more complete discussion of the algorithm's operation, the reader is refered to [14].

D *Computations*

Regardless of the algorithm, a computation is defined as all the operations performed by a decoder as it moves from one node to the next, either in the forward or in the backward direction. Clearly a computation of the stack decoder is different from a computation of the Fano decoder, and in the latter case, a forward computation is slightly different from a backward computation. As a further inconvenience, the operations of a punctured decoder and those of a normal (i.e. non-punctured) decoder for high-rate codes are not directly comparable using this definition. Using a punctured high-rate b/v code derived from a rate-$1/v_0$ original encoder, a computation of the punctured decoder involves extending a node into two successors while a computation of the normal decoder involves extending a node into its 2^b successors. Nevertheless, this definition is useful for evaluating the computational behavior of decoders independently of specific implementation factors. Since it is widely used, we have chosen to use this definition despite of its weaknesses, resorting to indirect methods of comparison for more precise comparison of the decoding effort of punctured and normal decoders.

III SIMULATION RESULTS

A *Simulation parameters*

Computer programs for simulating both sequential decoding algorithms have been implemented on workstation-type computers. Exploiting the linearity of convolutional codes, the all-zero message was assumed to be transmitted on a channel corrupted with additive white Gaussian noise. Using a pseudorandom number generator to generate the noise, received symbols corresponding to hard (two levels) or soft (eight levels) quantization were produced in frames of several hundred branches terminated by a tail. Sequences comprising several thousand frames were decoded so that statistics for the analysis and comparison of the behavior of the decoders were compiled over millions of bits. Simulations were performed for three values of signal-to-noise ratio, starting from values yielding a computational cut-off rate very close to the computational limit $R_{\text{comp}} = R$ and extending to higher signal-to-noise ratios. The actual values of bit energy to noise ratio E_b/N_0, symbol energy to noise ratio E_N/N_0 and corresponding values of R/R_{comp} used for the simulations appear in Table 1.

B *Fano algorithm: Threshold determination*

A first series of simulations have been conducted with the Fano decoder in order to determine good values of threshold spacings Δ to use with the punctured codes listed in [8]–[10]. The role of the threshold is to enable the detection of metric drops of the (locally) most likely path.

For small values of threshold increment Δ the threshold is tightened or lowered in small steps, providing good detection of any metric drop. But since after each lowering of the threshold the decoder must revisit all the nodes with metrics higher than the current threshold value, small lowerings of the threshold may force the decoder to make a lot of useless computations.

On the other hand, large values for Δ yield fast lowerings of the threshold. This will tend to keep the number of computations small, even though the lack of sensitivity to small metric drops may occasionnally give rise to large numbers of computations. But large differences between threshold values have an adverse effect on the error performance of the decoder. Among the incorrect paths

that are examined before the correct path, there may be a path that does not violate the threshold even though its cumulative metric falls below that of the correct path. As illustrated on Fig. 1, this incorrect path will be followed by the decoder, leading to a decoding error that would not have occured with a small value of Δ.

Just like for the Fano decoding of low-rate codes, a trade-off exists between the computational load and the error performance of the punctured high-rate decoder, and in setting a value for the threshold spacing Δ a compromise must be made between these two opposite effects. Several series of simulations with varying values of Δ were performed with the punctured codes for which errors could be observed within a reasonable simulation time. From the results, graphs of error probability versus Δ and of average number of computations for decoding one bit versus Δ have been plotted. Two typical such graphs shown on Figs. 2 and 3 clearly illustrate the trends identified above. In view of the results, a range of valid values for Δ has been identified according to the following criteria: on the large side, values of Δ leading to error probabilities more than 15 % larger than the best probability achievable are rejected; on the small side, values of Δ leading to more than 30 % added computations over the minimum possible are rejected.

Although the results indicate that the ranges are sensitive to several factors including coding rate, memory length, signal-to-noise ratio and the actual code used, values of Δ common to the majority of the codes could be identified: $\Delta = 19, 20, 21$ and $\Delta = 29, 30$ for hard and soft quantization respectively. Using one value of Δ from these sets for all simulations is not an optimal choice but a good compromise suitable for comparison and analysis purposes.

C Stack and Fano algorithms: simulation results

In this section we present results of simulations conducted with high-rate punctured codes listed in [8] and [9] using the stack and Fano algorithms of sequential decoding over hard quantized additive white Gaussian channels.

C.1 Error probability

A fundamental measure of performance of a decoder is the residual probability of error. An optimal decoder (e.g. Viterbi decoder) provides an error probability that is entirely dependent on the error correcting power of the specific code used. Of course, the use of a suboptimal decoder entails an error performance degradation. Unfortunately, for convolutional codes of interest, determining the exact theoretical error probability is a formidable task. We must rely instead on upper bounds of the union type [17] derived from the weight spectra of the codes [8], [9].

Since we are dealing with powerful long memory convolutional codes, error events are rather unlikely and simulations comprising very large numbers of bits must be performed if significant error probabilities are to be observed. With reasonably large numbers of bits, errors were observed only with the shorter memory codes.

Table 2 compares the bit error probabilities observed with the stack and Fano decoders for the memory $M = 10$ codes with the theoretical upper union bound. The results are representative of similar results obtained for other punctured codes listed in [8] and [9]. It can be seen that in both cases the observed error probabilities are smaller than the theoretical upper bound. Aside from the fact that the bound is an upper bound, this is due in part to the fact that the blocks which resulted in overflows have not been considered in computing the error probabilities. Understandably, these blocks could have resulted in additionnal errors if full decoding had been allowed.

A comparison between the error probabilities obtained with the stack decoder and those obtained with the Fano decoder reveals a slight advantage for the former. This may be explained in part by the degradation of error correcting power resulting from the compromise made on the threshold spacings of the Fano decoder.

C.2 Decoding effort

Another important aspect of the performance of a sequential decoder is its computational behavior, as measured by the average decoding effort and the distribution of the number of computations for decoding one bit.

Table 3 presents the values of average number of computations for decoding one bit for the Fano and the stack decoders, denoted by C_{Fano} and C_{ZJ} respectively, for the memory $M = 10$ and $M = 18$ punctured codes.

Since in the punctured approach the decoding is carried out one bit at a time, at least one computation per bit is required. As expected, for both decoders the average number of computations decreases with increasing signal-to-noise ratio. It should be noted that because overflows are allowed, the decoders make fewer computations on the average since frames that would require more computations than the limit are simply declared as overflows. It can also be observed that, as with usual low-rate codes, the average decoding effort is relatively insensitive to the memory length M of the codes.

Comparing the results for the stack and the Fano decoders, it can be seen that the Fano decoder performs consistently more computations than the stack decoder. The ratio $\frac{C_{\text{Fano}}}{C_{\text{ZJ}}}$, also included in Table 3, decreases with increasing signal-to-noise ratio. For the simulations presented here, the ratio ranges between 4.86 and 1.41. Clearly, the Fano decoder performs more computations than the stack decoder because of repeated computations in its forward and backward motions in noisy spans. The computations of the Fano decoder are simpler though, because they do not require accessing and sorting a stack.

The computational variability of the decoders has also been evaluated. Cumulative distributions of the number of computations for decoding one branch have been plotted for the same memory $M = 10$ and $M = 18$ punctured codes. The results obtained with the stack decoder are presented on Figs. 4 to 9 for the $M = 10$, $R = 2/3, 3/4, 4/5, 5/6, 6/7, 7/8$ punctured codes respectively, and on Figs. 10 to 15 for the $M = 18$, $R = 2/3, 3/4, 4/5, 5/6, 6/7, 7/8$ punctured codes respectively. Corresponding results obtained with the Fano decoder appear on Figs. 16 to 9 for the $M = 10$ punctured codes, and on Figs. 22 to 27 for the $M = 18$ punctured codes.

From these results, it can be seen that simulations with larger signal-to-noise ratios are more easily performed: the variability of decoding effort is smaller for large signal-to-noise ratios. The curves also exhibit the familiar Pareto asymptote that is typical of sequential decoders [11], [18].

A more surprising feature not usually found on cumulative distributions of sequential decoding for low-rate codes may be observed on the distributions of computational effort obtained for both types of decoders with punctured codes: all curves exhibit a plateau before assuming the asymptotic Pareto behavior. The plateaus start at about 2 or 3 computations and extend progressively further as the coding rates increase: up to about 10 computations for rate-2/3 codes, but reaching about 100 computations for rate-7/8 codes.

The presence of plateaus in the cumulatives may be explained by the following reasoning. A plateau in a cumulative extending from n_1 to n_2 computations indicates that the decoder (almost)

never performs n computations for $n_1 < n < n_2$. Now consider a sub-tree of length l branches. The number of nodes in this sub-tree is 2^l. Hence, in a worst case situation, the decoder must perform a number of computations that is exponential in the length l to carry out a full exploration of such a sub-tree. For a stack decoder, the number of computations required is simply $2^l - 1$. For a Fano decoder, backward and lateral moves have to be considered also. It is known that the number of backward and lateral moves at depth l is no more than 2^b times the number of forward moves at depth $l-1$ [20, p. 273], where 2^b is the number of immediate descendants for one node in the tree (for punctured codes, $b = 1$). Since terminal nodes are visited only, we find that the total number of computations for a full exploration over length l is $(2^{l+1} - 3)$.

Consider Table 4 which presents the numbers of computations for full explorations of subtrees of length b branches for both types of decoders. It can be seen that for a given b, the numbers in Table 4 correspond roughly to the upper limits n_2 of the plateaus on the cumulative distributions for rate-b/v punctured codes. Hence, the presence of plateaus in the cumulatives indicate that, apart from branches that are decoded with minimal effort requiring one or a few computations, a large proportion of branches require a full exploration over about b branches, that is, over a length corresponding to a period of the punctured code, in order to be decoded.

The frequent occurrences of full explorations is attributable to a peculiarity of most rate-$(v-1)/v$ punctured codes that are, as those of [8] and [9], derived from complementary original codes, that is, from original codes for which the two branches leaving a node in the tree bear complementary symbols. With the perforation patterns specified in [8] and [9], only one branch position per period has more than one symbol remaining after perforation. The $b-1$ other branch positions bear only one symbol. Because the original encoder is complementary, within the sequence of $b-1$ branch positions, one of the two branches leaving each node bears the symbol '1' and the other bears the symbol '0'. When it comes to decoding then, any received sequence of $b-1$ symbols has a matching sequence of $b-1$ branches, which means that a true decision of the decoder is suspended until a branch that bears more than one symbol is encountered. Therefore true decoding decisions take place only once per period, which explains why full explorations over a length of one period occur frequently.

As a final remark on these plateaus, it should be noted that with the larger number of possible metric values observed with soft quantization, true decisions can take place at any time, and hence plateaus are not found on the cumulative distributions. This is illustrated by Fig. 28 which shows cumulative distributions obtained with the Fano decoder using both hard and soft quantization. A plateau is clearly visible on the curve for hard quantization but not on the curve for soft quantization.

C.3 Overflow probability

Because of its computational variability, a sequential decoder, regardless of the algorithm used, is subject to very long decoding delays when confronted to frames severely corrupted by noise. These delays give rise to many types of decoder overflows, which may result in losses of information and/or synchronisation.

In the context of simulations, overflows are usually declared on the basis of the number of computations alone: an overflow is said to occur if the number of computations for decoding a frame exceeds a predetermined limit. For stack decoders, this limit has been set to 5000 computations per frame, that is, about 10 computations per bit since the frames are about 500 bits long. For

Fano decoders, the limit has been set to 48 computations per bit to allow for the larger number of computations due to forward and backward motion.

As representative results of the overflow probabilities observed with stack and Fano decoders while decoding the punctured codes of [8] and [9], Table 5 presents the overflow probabilities for the memory $M = 10$ and $M = 18$ codes with coding rates varying from $R = 2/3$ to $R = 7/8$ using both algorithms. It can be seen from Table 5 — as is true for the other codes also — that the probabilities range between 10^{-2} to 10^{-4}, with a majority at about 10^{-3}. As expected, for a given code the overflow probabilities tend to be lower for larger values of signal-to-noise ratio. Overflow probabilities tend to be lower also as the coding rate increases, reflecting an increase in signal-to-noise ratio for maintaining the ratio R/R_{comp} as R increases.

IV COMPARISONS BETWEEN NORMAL AND PUNCTURED DECODING

Since high-rate punctured codes are proposed as an alternative to usual high-rate convolutional codes, a comparison between both approaches to high-rate convolutional coding is in order. In their 1979 paper, Clark *et al.* addressed the questions of error correcting power and computational effort for short memory punctured codes. Later, the error correcting power of long memory punctured codes was considered by the authors [8], [9]. Here, emphasis will be put on the computational behavior of the sequential decoders that are used for decoding long memory punctured codes: computational complexity, overflow probability, average decoding effort, etc.

By conducting simulation runs with identical received noisy sequences and using punctured codes that are equivalent to the best known usual high-rate convolutional codes [10], a comparison between usual and punctured decoding that excludes factors that do not pertain to differences between the two approaches is possible. The simulations have been performed with punctured codes that are equivalent to good systematic long memory high-rate convolutional codes of Hagenauer [19].

A Error probability

Since the metric is additive, the most likely path determined by the decoder should be the same whether the decoding of a high-rate code is performed one bit at a time like a punctured code or b bits at a time like a usual high-rate code [9]. Hence the error performance should be essentially the same for both decoding approaches. This was confirmed by the simulation results for both the stack and the Fano algorithms: when decoding errors were observed, the probabilities were essentially equal for both approaches and the cumulative metrics of the decoded paths were of similar orders of magnitude. Obviously, without errors the cumulative metrics were equal.

B Complexity of a computation

As mentioned before, comparing the computations of punctured and normal decoders is somewhat problematic. Whatever the decoding algorithm (i.e. stack or Fano), a computation of the punctured decoder involves the extension of a node into two successors while a computation of the normal decoder involves the extension of a node into 2^b successors. Hence there is an inherent complexity ratio of 2^{b-1} between normal and punctured computations. The only exception is for backward computations of the Fano algorithm which have the same complexity for both approaches.

A normal computation is more complex than a punctured computation, but of course more work is accomplished. Let us consider a coding rate $R = b/v$. On one side of the spectrum, assume a noiseless sequence of length L/b bits is to be decoded. This sequence will be decoded in L computations by a punctured decoder and in L/b computations by a normal decoder. When only forward motion is involved then, the normal decoder needs to perform b times fewer computations to do the same work as the punctured decoder.

On the other side of the spectrum, assume that a very noisy sequence of the same length is to be decoded. The decoding of this sequence will require in the worst case a full exploration of the tree. For the punctured stack decoder, the number of computations required is $2^L - 1$, while the number of computations for the normal decoder is $(2^L - 1)/(2^b - 1)$. Hence the punctured stack decoder must perform $2^b - 1$ times as many computations to do the same work as the normal stack decoder. For Fano decoders, the situation is slightly more complex because of forward and backward computations, but it leads to the same general conclusion: the punctured decoders must perform approximately 2^b times as many computations to do the same amount of work as the normal decoders.

This could lead us to believe that punctured decoders are not as efficient as usual decoders, but we still have to account for the relative complexity of the computations of each type of decoder. Doing that, we find that on the noiseless side, the ratio between the overall complexities of punctured and normal computations is $b/2^{b-1}$, while on the very noisy side, the ratio is 2. In normal operation, the decoder will operate between these two extremes. With moderate to high signal-to-noise ratio, the decoder will spend most of its time closer to the noiseless case, and hence we could expect that the overall complexity ratio should be close to $b/2^{b-1}$, a ratio increasingly favorable to the punctured decoder as b is increased.

C Decoding effort

The values of average number of computations for decoding one bit observed for the stack and Fano decoders confirm our observations about the number of computations that each approach (punctured or normal) requires for performing an equivalent amount of work. The values of average number of computations observed with normal and punctured stack decoders are compared in Table 6. It can be seen that for moderate to high signal-to-noise ratios, the punctured decoder performs slightly more than b times as many computations as the normal decoder. Even for values of R/R_{comp} close to 1, the ratio between the average numbers of computations does not differ by more than 15 % from b. The results obtained with the Fano algorithm indicate a similar relationship between the normal and punctured approaches.

In order to take into consideration the relative complexity of the computations, a comparison based on CPU times for decoding is instructive. Table 7 presents the comparison between average times for decoding one bit observed with normal and punctured stack decoders. According to the observations made in the preceding section, the ratio between the average CPU times of normal and punctured decoders should follow the variation of $2^{b-1}/b$ with b. It can be seen from Table 7 that the ratio follows quite closely the variation of $2^b/b$ with b, suggesting a further advantage for the punctured approach. One should keep in mind though that a comparison based on CPU times has an inherent qualitative flavor and should not be used for a precise rating of the different approaches.

The results obtained with the Fano decoders follow similar variations of the ratios with b. Hence,

for both decoding algorithms, as the coding rate $R = b/v$ increases, the computational load for decoding increases linearly for the punctured approach but exponentially for the usual approach. For lower rate codes (i.e. close to $R = 1/2$), the usual approach has a slight advantage but for higher rate codes, the advantage goes rapidly to the punctured approach.

D Computational variability

Comparing the computational variability of punctured and usual decoders is even more difficult because not only do computations have different complexities but the number of information bits that each branch carries also differs with each approach. A direct comparison of the cumulative distributions is nevertheless instructive, especially if it is interpreted with the results of the preceding section in mind. Cumulative distributions of the number of computations for decoding one branch obtained for both the punctured and the usual approach have been plotted on the same graph. Figs. 29 to 31 present typical results obtained with the $R = 2/3, 4/5, 5/6$ long memory punctured codes of [10] respectively with stack decoders. Results obtained with Fano decoders were similar, as can be seen on Fig. 32 which presents the cumulative distributions for the $R = 5/6$ case.

The cumulative distributions seem to indicate that punctured decoders are subject to more variability than normal decoders, and that the difference is more pronounced for higher coding rates. This conclusion would be valid if branches and computations were the same for both approaches. But a fair comparison should allow for the facts that a computation of the normal decoder is 2^{b-1} times as complex as a computation of the punctured decoder[1] and that for the same depth (in information bits), b branches of the punctured code must be counted for every branch of the normal code. In order to compensate for these effects, a corrective interpretation of the cumulative distributions is needed. Let $P(C_n > N)$ and $P(C_p > N)$ be the frequency of occurrence of branches whose decoding requires more than N computations for the normal and punctured approaches respectively. Let us take branches of b bits and punctured computations as standards of comparison. We may estimate the probability of performing more than N_e standard computations for decoding one standard branch for the punctured approach as

$$P(C_p > N_e) = P(C_p > N) \text{ for } N_e = bN \tag{2}$$

since on the average, the punctured decoder performs about b computations for decoding one branch of b bits. On the other hand, the usual decoder performs only one computation, but this computation is as complex as 2^{b-1} standard computations, so we have

$$P(C_n > N_e) = P(C_n > N) \text{ for } N_e = 2^{b-1}N \tag{3}$$

for the usual approach. Comparing the decoding approaches on this basis, a correction is in fact applied to the cumulative distributions before comparing them. As b increases, the correction turns out clearly to the advantage of the punctured approach, compensating for the differences observed between the cumulative distributions plotted on Figs. 29 to 32. With this interpretation, it appears that the variability is not worse for punctured decoders than it is for normal decoders. A more refined analysis would be required to include the effects of the tail or of backward computations in the case of Fano decoders, but it is believed that the general conclusion would not be much different.

[1] In first approximation, excluding the effects of backward computations for Fano decoders and of the tail for both algorithms.

E *Overflow probability*

In order to compare the overflow probabilities of normal and punctured decoders, the limits on the number of computations fixed for the decoders must reflect the differences between the computations of the decoders. To be fair, the punctured decoder should be allowed at least b times as many computations as the normal decoder. In order to account for the impact of noisy spans where explorations are required, we decided to allow a margin of 20 % more computations to the punctured decoder. The simulation results confirmed the validity of this choice: with limits on the number of allowed computations set accordingly, the overflow probabilities of normal and punctured decoders were of the same order of magnitude.

F *Stack size*

An aspect of computational performance that pertains solely to the stack decoder is the amount of stack memory that is actually needed for completing the decoding of a frame. Although memory is generally cheap, a more efficient usage of the memory alloted for the stack translates into a smaller probability of stack overflow. As can be seen from Table 8 which compares the average numbers of stack entries for both approaches, this aspect is clearly to the advantage of the punctured approach. In a noiseless situation, the average number of stack entries per bit is two for the punctured stack decoder while it is $2^b/b$ for the normal decoder. Table 8 confirms once again that the actual results are very close to what may be estimated by such an optimistic analysis.

This exponential dependence has significant practical consequences. The amount of stack memory required by a usual stack decoder becomes rapidly prohibitive as the coding rate gets close to 1. For instance, for decoding $R = 15/16$ codes, the stack would have to accommodate over 2000 (about $2^{15}/15$) stack entries per bit for a usual decoder! Decoding such very high-rate codes with a punctured stack decoder presents no special difficulty.

V CONCLUDING REMARKS

A number of concluding remarks regarding the decoding of high-rate punctured convolutional codes by sequential decoding algorithms may be drawn from the simulation results presented. First, it has been clearly demonstrated that sequential decoding is applicable to the decoding of high-rate long memory punctured codes. Both the stack algorithm and the Fano algorithm have been used with success for decoding long memory punctured codes with coding rates ranging from $R = 2/3$ to $R = 7/8$. Various aspects of decoding performance show that the decoding behavior generally conforms to expectations based on the special structure of punctured codes.

Second, the simulations have made clear that with respect to decoding behavior and performance, the two algorithms have much in common. The results for error and overflow probabilities and variability of decoding effort were quite similar for both algorithms. In particular, plateaus were observed in the cumulatives of decoding effort obtained with both algorithms when hard quantization of received symbols was assumed. These plateaus have been shown to correspond to full explorations of subtrees over a length of one branch of the resulting high-rate code.

There are nonetheless differences between the two algorithms. Due to its forward and backward motion, the Fano decoder performs on the average a larger number of computations than the stack decoder, especially for relatively low signal-to-noise ratio, but its computations are simpler. Hence,

for high signal-to-noise ratio, the Fano algorithm is a good choice but for low signal-to-noise ratio, the stack algorithm is to be preferred.

Finally, comparing the punctured approach of decoding to the more traditional technique for high-rate codes, it is found that punctured decoders perform a larger number of simpler computations. The net effect is that for all but the lower rate codes, the overall decoding effort is on the average more important for the usual decoder than it is for its punctured counterpart. Several aspects of computational performance, such as the average time to decode one bit, exhibit an exponential dependence with b for a usual decoder but only a linear dependence for a punctured decoder. This exponential dependence has significant consequences, like the fact that the decoding of very high-rate codes with a usual stack decoder is impractical, while it presents no special difficulty for a punctured stack decoder.

As for the computational variability, it is not worse for punctured decoders than it is for normal decoders. Error and overflow probabilities are the same for both approaches. Given the undeniable flexibility and the practical advantages of the punctured approach, it is clear that the combination of long memory punctured codes and sequential decoding represent an attractive — and in some cases, the only viable — alternative for powerful error correction and detection.

References

[1] J.B. Cain, G.C. Clark, J. Geist, "Punctured Convolutional Codes of Rate $(n-1)/n$ and Simplified Maximum Likelihood Decoding," *IEEE Trans. Inform. Theory*, Vol. IT-25, pp. 97–100, January 1979.

[2] K. Zigangirov, "Some Sequential Decoding Procedures," *Problemii Peredachi Informatsii*, Vol. 2, pp. 13–15, 1966.

[3] F. Jelinek, "A Fast Sequential Decoding Algorithm Using a Stack," *IBM Jour. Res. and Develop.*, Vol. 13, pp. 675–685, Nov. 1980.

[4] G. Bégin, D. Haccoun, "Decoding of Punctured Convolutional Codes by the Stack Algorithm," *Abstracts of Papers, 1986 IEEE International Symposium on Information Theory*, Ann Arbor, Michigan, Oct. 1986, p. 159.

[5] Y. Yasuda, K. Kashiki, Y. Hirata, "High-Rate Punctured Convolutional Codes for Soft Decision Viterbi Decoding," *IEEE Trans. Commun.*, Vol. COM-32, pp. 315–319, March 1984.

[6] J. Hagenauer, "Rate Compatible Punctured Convolutional Codes and their Applications," *IEEE Trans. Commun.*, Vol. COM-36, pp. 389–400, April 1988.

[7] W.W. Wu, D. Haccoun, R. Peile, Y. Hirata, "Coding for Satellite Communication," *IEEE J. Seelect. Areas Commun.*, Vol. SAC-5, pp. 724–748, May 1987.

[8] D. Haccoun, G. Bégin, "High-Rate Punctured Convolutional Codes for Viterbi and Sequential Decoding," *IEEE Trans. Commun.*, Vol. COM-37, pp. 1113–1125, November 1989.

[9] G. Bégin, D. Haccoun, C. Paquin, "Further Results on High-Rate Punctured Convolutional Codes for Viterbi and Sequential Decoding," *IEEE Trans. Commun.*, Vol. COM-38, pp. 1922–1937, November 1990.

[10] G. Bégin, D. Haccoun, "High-Rate Punctured Convolutional Codes: Structure Properties and Construction Technique," *IEEE Transactions on Communications*, Vol. COM-37, No. 12, pp. 1381–1385, December 1989.

[11] D. Haccoun, M.J. Ferguson, "Generalized Stack Algorithms for Decoding Convolutional Codes," *IEEE Trans. Inform. Theory*, Vol. IT-21, pp. 638–651, November 1975.

[12] P.R. Chevillat, D.J. Costello Jr, "A Multiple Stack Algorithm for Erasure Free Decoding of Convolutional Codes," *IEEE Trans. Commun.*, Vol. COM-25, pp. 1460–1470, December 1967.

[13] I.M. Jacobs, 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.

[14] R.M. Fano, "A Heuristic Discussion of Probabilistic Decoding," *IEEE Trans. Inform. Theory*, Vol. IT-9, pp. 64–73, April 1963.

[15] C.Y. Chang, K. Yao, "Systolic Array Architecture for the Sequential Stack Decoding Algorithm", Proc. SPIE vol. 696, Advanced Alg. and Arch. for Signal Proc.", 1986, pp. 196–203.

[16] P. Lavoie, D. Haccoun, Y. Savaria, "A Systolic Architecture for Fast Stack Sequential Decoders, " submitted for publication in *IEEE Trans. Commun.*, March 1991.

[17] A.J. Viterbi, "Convolutional Codes and Their Performance in Communications Systems," *IEEE Trans. Commun. Technol.*, Vol. COM-19, pp. 751–772, October 1971.

[18] S. Lin, D.J. Costello Jr., *Error Control Coding*, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[19] J. Hagenauer, "High Rate Convolutional Codes with Good Profiles," *IEEE Trans. Inform. Theory*, Vol. IT-23, pp. 615–618, September 1977.

[20] R.G. Gallager, *Information Theory and Reliable Communication*, Wiley, New York, 1968.

List of Tables

1 Values of E_b/N_0, E_N/N_0 and R/R_{comp} used for the simulations. 18
2 Error probabilities and upper bound for stack and Fano decoders, $M = 10$. 19
3 Average number of computations for decoding one bit for stack and Fano decoders. . 20
4 Number of computations for a full exploration over b branches. 20
5 Overflow probabilities for stack and Fano decoders. 21
6 Comparison between average numbers of computations for punctured and usual stack decoders. 21
7 Comparison between average CPU times for decoding one bit observed with normal and punctured stack decoders (in milliseconds). 22
8 Average number of stack entries per bit. Punctured approach: $(\frac{ent.}{bit})_p$, usual approach: $(\frac{ent.}{bit})_n$. 22

List of Figures

1 Decoding error caused by a large threshold spacing Δ. 23
2 Error probability versus threshold increment Δ. 23
3 Number of computations for decoding one bit versus threshold increment Δ. 23
4 Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 2/3$, hard quantization. 24
5 Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 3/4$, hard quantization. 24
6 Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 4/5$, hard quantization. 24
7 Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 5/6$, hard quantization. 25
8 Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 6/7$, hard quantization. 25
9 Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 7/8$, hard quantization. 25
10 Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 2/3$, hard quantization. 26
11 Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 3/4$, hard quantization. 26
12 Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 4/5$, hard quantization. 26
13 Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 5/6$, hard quantization. 27
14 Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 6/7$, hard quantization. 27
15 Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 7/8$, hard quantization. 27
16 Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 2/3$, hard quantization. 28

17 Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 3/4$, hard quantization. 28
18 Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 4/5$, hard quantization. 28
19 Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 5/6$, hard quantization. 29
20 Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 6/7$, hard quantization. 29
21 Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 7/8$, hard quantization. 29
22 Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 2/3$, hard quantization. 30
23 Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 3/4$, hard quantization. 30
24 Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 4/5$, hard quantization. 30
25 Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 5/6$, hard quantization. 31
26 Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 6/7$, hard quantization. 31
27 Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 7/8$, hard quantization. 31
28 Cumulatives of the number of computations for decoding one branch for the Fano decoder, soft and hard quantization. $M = 10$, $R = 3/4$ 32
29 Comparison between cumulative distributions of the number of computations for decoding one branch for punctured and usual stack decoders. $R = 2/3$, hard quantization. 32
30 Comparison between cumulative distributions of the number of computations for decoding one branch for punctured and usual stack decoders. $R = 4/5$, hard quantization. 32
31 Comparison between cumulative distributions of the number of computations for decoding one branch for punctured and usual stack decoders. $R = 5/6$, hard quantization. 33

32 Comparison between cumulative distributions of the number of computations for decoding one branch for punctured and usual Fano decoders. $R = 5/6$, hard quantization. 33

Table 1: Values of E_b/N_0, E_N/N_0 and R/R_{comp} used for the simulations.

R	E_b/N_0 (dB)	E_N/N_0 (dB)	R/R_{comp}
2/3	5.261	3.50	0.9998
	5.761	4.00	0.9383
	6.261	4.50	0.8853
3/4	5.750	4.50	0.9960
	6.250	5.00	0.9451
	6.750	5.50	0.9022
4/5	6.069	5.10	0.9983
	6.569	5.60	0.9541
	7.069	6.10	0.9174
5/6	6.392	5.60	0.9939
	6.892	6.10	0.9556
	7.392	6.60	0.9244
6/7	6.569	5.90	0.9978
	7.069	6.40	0.9628
	7.569	6.90	0.9346
7/8	6.780	6.20	0.9963
	7.280	6.70	0.9648
	7.780	7.20	0.9397

Table 2: Error probabilities and upper bound for stack and Fano decoders, $M = 10$.

R		$R/R_{\text{comp}} = 0.9998$	$R/R_{\text{comp}} = 0.9383$	$R/R_{\text{comp}} = 0.8853$
2/3	stack	5.19×10^{-5}	5.63×10^{-5}	1.09×10^{-5}
	Fano	1.16×10^{-4}	1.58×10^{-4}	1.90×10^{-5}
	bound	(3.03×10^{-4})	(3.96×10^{-5})	(5.00×10^{-6})
R		$R/R_{\text{comp}} = 0.9960$	$R/R_{\text{comp}} = 0.9451$	$R/R_{\text{comp}} = 0.9022$
3/4	stack	4.09×10^{-5}	9.86×10^{-6}	1.50×10^{-6}
	Fano	9.30×10^{-5}	2.60×10^{-5}	3.10×10^{-6}
	bound	(2.23×10^{-4})	(2.85×10^{-5})	(4.02×10^{-6})
R		$R/R_{\text{comp}} = 0.9983$	$R/R_{\text{comp}} = 0.9541$	$R/R_{\text{comp}} = 0.9174$
4/5	stack	3.28×10^{-4}	6.27×10^{-5}	3.96×10^{-6}
	Fano	8.32×10^{-4}	1.79×10^{-4}	2.30×10^{-6}
	bound	(4.55×10^{-4})	(5.29×10^{-5})	(7.17×10^{-6})
R		$R/R_{\text{comp}} = 0.9939$	$R/R_{\text{comp}} = 0.9556$	$R/R_{\text{comp}} = 0.9244$
5/6	stack	1.06×10^{-4}	1.46×10^{-5}	3.20×10^{-6}
	Fano	2.57×10^{-4}	6.40×10^{-5}	1.40×10^{-5}
	bound	(9.99×10^{-4})	(9.83×10^{-5})	(1.15×10^{-5})
R		$R/R_{\text{comp}} = 0.9978$	$R/R_{\text{comp}} = 0.9628$	$R/R_{\text{comp}} = 0.9346$
6/7	stack	1.80×10^{-5}	9.52×10^{-6}	6.67×10^{-6}
	Fano	7.20×10^{-5}	1.50×10^{-5}	2.00×10^{-6}
	bound	(3.91×10^{-4})	(3.18×10^{-5})	(3.22×10^{-6})
R		$R/R_{\text{comp}} = 0.9963$	$R/R_{\text{comp}} = 0.9648$	$R/R_{\text{comp}} = 0.9397$
7/8	stack	1.35×10^{-5}	4.84×10^{-6}	
	Fano	1.20×10^{-4}	2.10×10^{-4}	3.00×10^{-6}
	bound	(3.96×10^{-4})	(3.62×10^{-5})	(4.26×10^{-6})

Table 3: Average number of computations for decoding one bit for stack and Fano decoders.

		C_{ZJ}	C_{Fano}	$\frac{C_{\mathrm{ZJ}}}{C_{\mathrm{Fano}}}$	C_{ZJ}	C_{Fano}	$\frac{C_{\mathrm{ZJ}}}{C_{\mathrm{Fano}}}$	C_{ZJ}	C_{Fano}	$\frac{C_{\mathrm{ZJ}}}{C_{\mathrm{Fano}}}$
R	M	$R/R_{\mathrm{comp}} = 0.9998$			$R/R_{\mathrm{comp}} = 0.9383$			$R/R_{\mathrm{comp}} = 0.8853$		
2/3	10	1.86	7.69	4.13	1.37	4.30	3.14	1.15	2.44	2.12
	18	1.82	7.03	3.86	1.29	3.55	2.75	1.14	2.19	1.92
R	M	$R/R_{\mathrm{comp}} = 0.9960$			$R/R_{\mathrm{comp}} = 0.9451$			$R/R_{\mathrm{comp}} = 0.9022$		
3/4	10	1.51	5.93	3.93	1.22	3.27	2.68	1.11	2.15	1.94
	18	1.66	7.01	4.22	1.30	3.59	2.76	1.15	2.29	1.99
R	M	$R/R_{\mathrm{comp}} = 0.9983$			$R/R_{\mathrm{comp}} = 0.9541$			$R/R_0 = 0.9174$		
4/5	10	1.56	6.33	4.06	1.23	3.13	2.54	1.11	1.95	1.76
	18	1.60	7.77	4.86	1.26	3.29	2.61	1.12	1.91	1.71
R	M	$R/R_{\mathrm{comp}} = 0.9939$			$R/R_{\mathrm{comp}} = 0.9556$			$R/R_{\mathrm{comp}} = 0.9244$		
5/6	10	1.63	6.26	3.84	1.29	3.24	2.51	1.14	2.05	1.80
	18	1.52	5.33	3.51	1.23	2.74	2.23	1.12	1.84	1.64
R	M	$R/R_{\mathrm{comp}} = 0.9978$			$R/R_{\mathrm{comp}} = 0.9628$			$R/R_{\mathrm{comp}} = 0.9346$		
6/7	10	1.40	4.59	3.28	1.18	2.43	2.06	1.09	1.67	1.53
	18	1.43	4.93	3.45	1.20	2.50	2.08	1.11	1.68	1.51
R	M	$R/R_{\mathrm{comp}} = 0.9963$			$R/R_{\mathrm{comp}} = 0.9648$			$R/R_{\mathrm{comp}} = 0.9397$		
7/8	10	1.40	4.40	3.14	1.18	2.40	2.03	1.09	1.65	1.51
	18	1.35	4.23	3.13	1.16	2.31	1.99	1.09	1.54	1.41

Table 4: Number of computations for a full exploration over b branches.

b	Stack decoder	Fano decoder
2	3	5
3	7	13
4	15	29
5	31	61
6	63	125
7	127	253

Table 5: Overflow probabilities for stack and Fano decoders.

	stack	Fano	stack	Fano	stack	Fano
			$R = 2/3$			
M	$R/R_{\text{comp}} = 0.9998$		$R/R_{\text{comp}} = 0.9383$		$R/R_{\text{comp}} = 0.8853$	
10	2.34×10^{-2}	2.24×10^{-2}	6.43×10^{-3}	8.30×10^{-3}	1.20×10^{-3}	1.00×10^{-3}
18	2.29×10^{-2}	2.25×10^{-2}	5.30×10^{-3}	6.60×10^{-3}	1.00×10^{-3}	9.00×10^{-4}
			$R = 3/4$			
M	$R/R_{\text{comp}} = 0.9960$		$R/R_{\text{comp}} = 0.9451$		$R/R_{\text{comp}} = 0.9022$	
10	1.49×10^{-2}	1.40×10^{-2}	2.30×10^{-3}	2.50×10^{-3}	6.00×10^{-4}	2.10×10^{-4}
18	1.97×10^{-2}	2.25×10^{-2}	3.70×10^{-3}	3.60×10^{-3}	5.00×10^{-4}	6.00×10^{-4}
			$R = 4/5$			
M	$R/R_{\text{comp}} = 0.9983$		$R/R_{\text{comp}} = 0.9541$		$R/R_{\text{comp}} = 0.9174$	
10	1.23×10^{-2}	1.17×10^{-2}	2.80×10^{-3}	1.79×10^{-3}	1.00×10^{-3}	1.10×10^{-3}
18	1.78×10^{-2}	5.00×10^{-2}	4.30×10^{-3}	5.30×10^{-3}	6.00×10^{-4}	4.00×10^{-4}
			$R = 5/6$			
M	$R/R_{\text{comp}} = 0.9939$		$R/R_{\text{comp}} = 0.9556$		$R/R_{\text{comp}} = 0.9244$	
10	1.48×10^{-2}	1.35×10^{-2}	2.80×10^{-3}	1.50×10^{-3}	1.20×10^{-3}	3.00×10^{-4}
18	1.04×10^{-2}	1.52×10^{-2}	1.70×10^{-3}	1.70×10^{-3}	4.00×10^{-4}	4.00×10^{-4}
			$R = 6/7$			
M	$R/R_{\text{comp}} = 0.9978$		$R/R_{\text{comp}} = 0.9628$		$R/R_{\text{comp}} = 0.9346$	
10	1.09×10^{-2}	8.80×10^{-3}	2.30×10^{-3}	1.30×10^{-3}	5.00×10^{-4}	1.00×10^{-4}
18	1.06×10^{-2}	1.21×10^{-2}	2.70×10^{-3}	1.20×10^{-3}	8.00×10^{-4}	0
			$R = 7/8$			
M	$R/R_{\text{comp}} = 0.9963$		$R/R_{\text{comp}} = 0.9648$		$R/R_{\text{comp}} = 0.9397$	
10	1.04×10^{-2}	6.30×10^{-3}	3.00×10^{-3}	8.10×10^{-4}	5.00×10^{-4}	0
18	7.90×10^{-3}	1.09×10^{-2}	2.30×10^{-3}	2.10×10^{-3}	6.00×10^{-4}	0

Table 6: Comparison between average numbers of computations for decoding one bit for punctured and usual stack decoders; C_p: punctured decoder, C_n: usual decoder.

$R = b/v$	R/R_{comp}	C_p	C_n	$\frac{C_p}{C_n}$	$\frac{C_p/C_n}{b}$
2/3	0.9998	2.060	0.940	2.1915	1.0957
2/3	0.9383	1.344	0.640	2.1000	1.0500
2/3	0.8853	1.161	0.564	2.0585	1.0293
3/4	0.9960	1.637	0.491	3.3340	1.1113
3/4	0.9451	1.233	0.385	3.2026	1.0675
3/4	0.9022	1.118	0.358	3.1229	1.0410
4/5	0.9983	1.632	0.370	4.4108	1.1027
4/5	0.9541	1.257	0.292	4.3048	1.0762
4/5	0.9174	1.140	0.273	4.1758	1.0440
5/6	0.9701	1.352	0.238	5.6807	1.1361
5/6	0.9361	1.161	0.216	5.3750	1.0750
5/6	0.9086	1.086	0.209	5.1962	1.0392
7/8	0.9648	1.202	0.159	7.5597	1.0800
7/8	0.9397	1.120	0.154	7.2727	1.0390

Table 7: Comparison between average CPU times for decoding one bit observed with normal and punctured stack decoders (in milliseconds); T_p: punctured decoder, T_n: usual decoder.

R	R/R_{comp}	T_p	T_n	T_n/T_p	$2^b/2b$	$2^b/b$
2/3	0.9998	0.350	0.370	1.057		
	0.9383	0.186	0.217	1.167	1.00	2.00
	0.8853	0.152	0.184	1.208		
3/4	0.9960	0.238	0.356	1.495		
	0.9451	0.159	0.269	1.688	1.33	2.67
	0.9022	0.143	0.248	1.735		
4/5	0.9983	0.260	0.721	2.769		
	0.9541	0.185	0.540	2.916	2.00	4.00
	0.9174	0.165	0.506	3.063		
5/6	0.9701	0.201	1.178	5.859		
	0.9361	0.168	1.038	6.158	3.20	6.40
	0.9086	0.156	1.008	6.444		
7/8	0.9648	0.187	4.610	24.616	9.14	18.29
	0.9397	0.174	4.412	25.326		

Table 8: Average number of stack entries per bit. Punctured approach: $(\frac{ent.}{bit})_p$, usual approach: $(\frac{ent.}{bit})_n$.

R	R/R_0	$(\frac{ent.}{bit})_p$	$(\frac{ent.}{bit})_n$	$2^b/b$
	0.9998	4.02	3.66	
2/3	0.9383	2.59	2.46	2.0
	0.8853	2.22	2.16	
	0.9960	3.19	3.82	
3/4	0.9451	2.38	2.96	2.7
	0.9022	2.15	2.76	
	0.9983	3.18	5.74	
4/5	0.9541	2.43	4.51	4.0
	0.9174	2.20	4.20	
	0.9701	2.64	7.40	
5/6	0.9361	2.26	6.72	6.4
	0.9086	2.11	6.50	
7/8	0.9648	2.27	19.28	18.3
	0.9397	2.11	18.60	

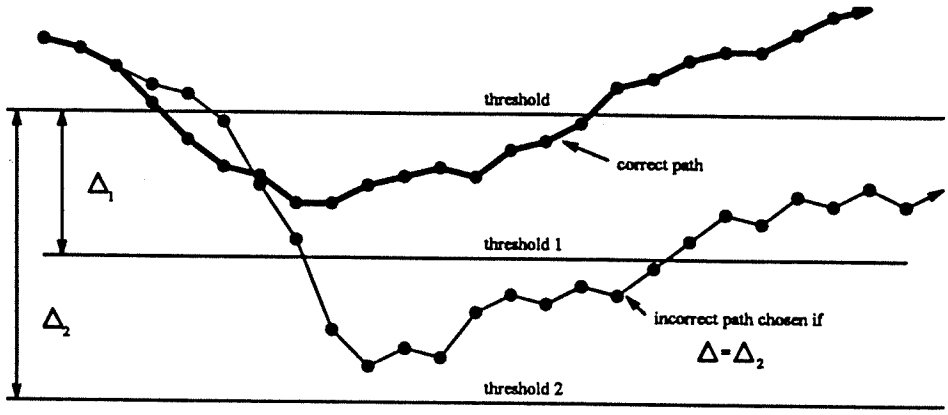


Figure 1: Decoding error caused by a large threshold spacing Δ.

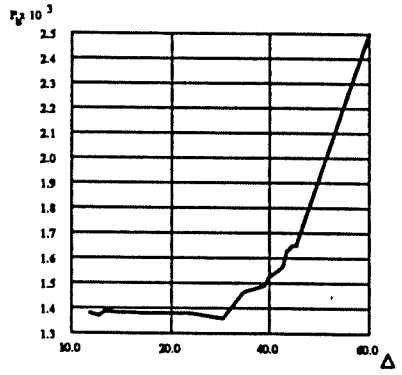


Figure 2: Error probability versus threshold increment Δ.

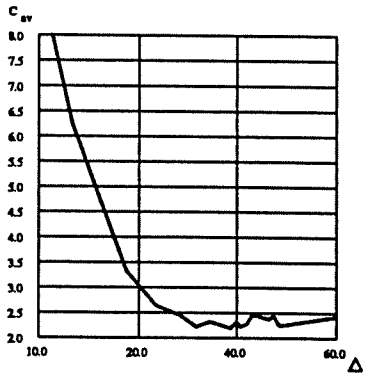
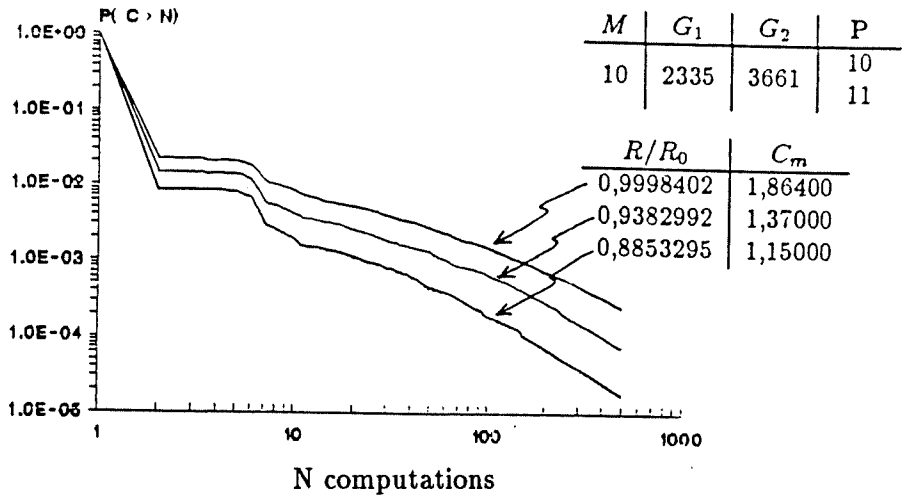


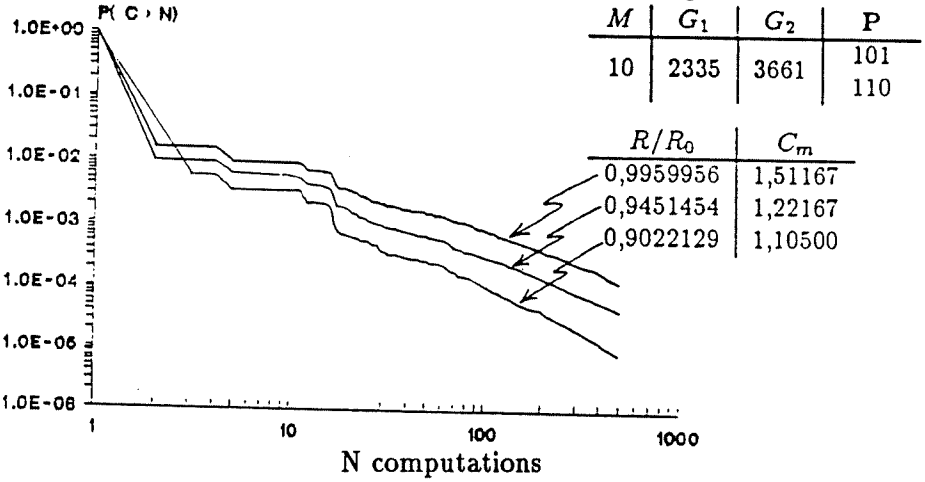
Figure 3: Number of computations for decoding one bit versus threshold increment Δ.



M	G_1	G_2	P
10	2335	3661	10
			11

R/R_0	C_m
0,9998402	1,86400
0,9382992	1,37000
0,8853295	1,15000

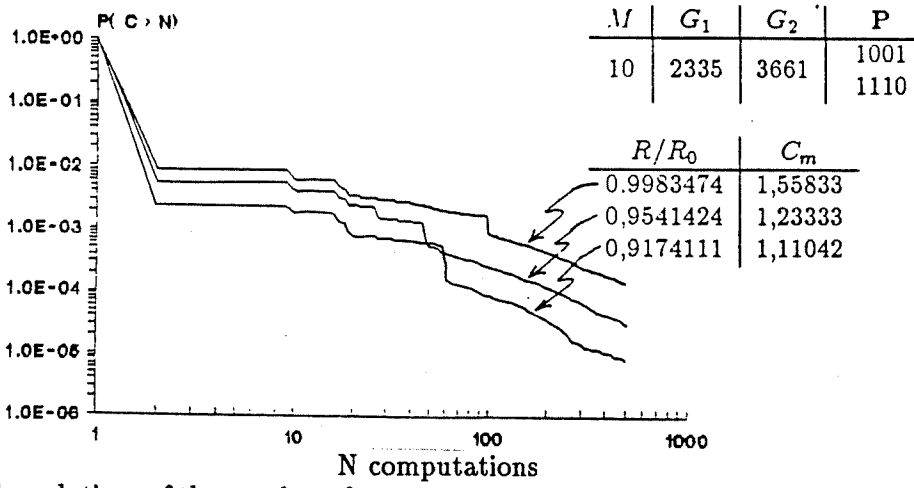
Figure 4: Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 2/3$, hard quantization.



M	G_1	G_2	P
10	2335	3661	101
			110

R/R_0	C_m
0,9959956	1,51167
0,9451454	1,22167
0,9022129	1,10500

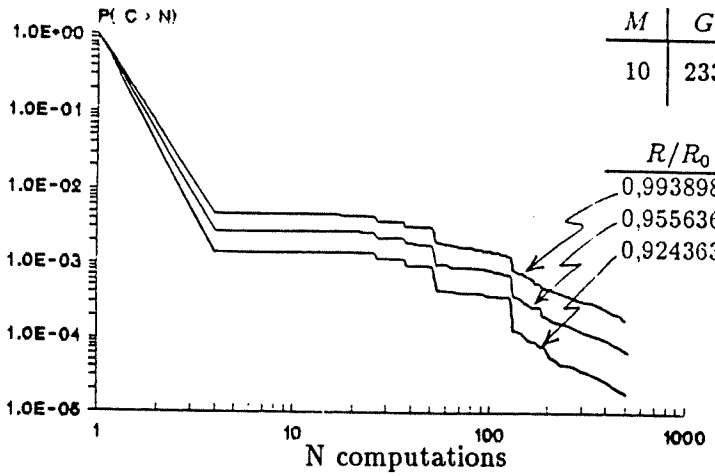
Figure 5: Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 3/4$, hard quantization.



M	G_1	G_2	P
10	2335	3661	1001
			1110

R/R_0	C_m
0.9983474	1,55833
0,9541424	1,23333
0,9174111	1,11042

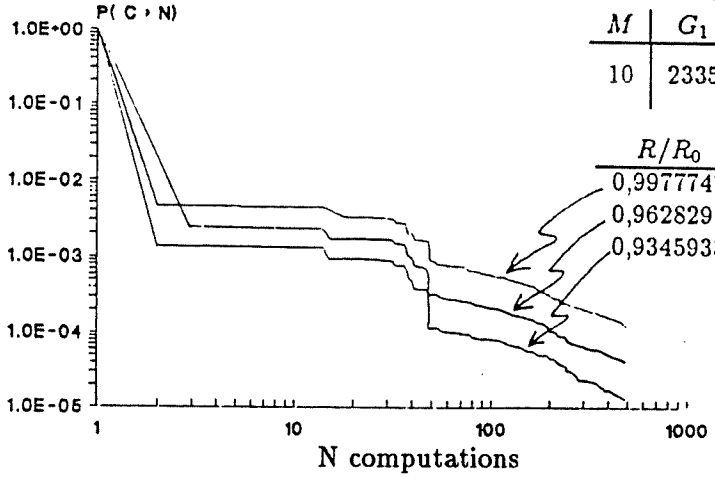
Figure 6: Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 4/5$, hard quantization.



M	G_1	G_2	P
10	2335	3661	11000
			10111

R/R_0	C_m
0,9938983	1,62600
0,9556365	1,29000
0,9243635	1,14000

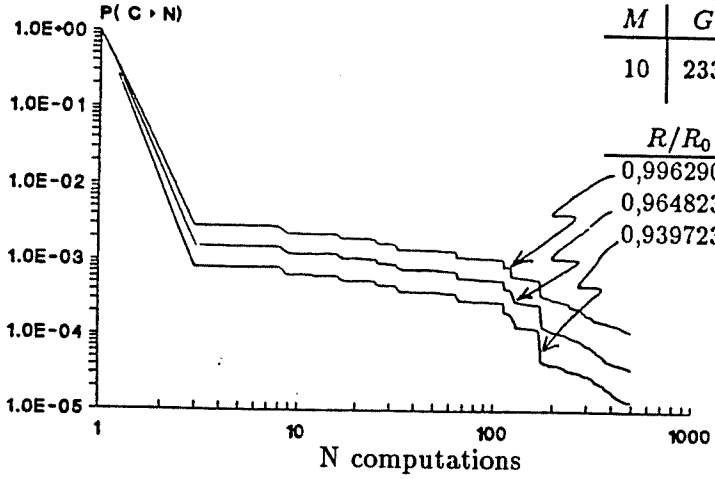
Figure 7: Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 5/6$, hard quantization.



M	G_1	G_2	P
10	2335	3661	111100
			100011

R/R_0	C_m
0,9977747	1,39667
0,9628291	1,17667
0,9345933	1,08667

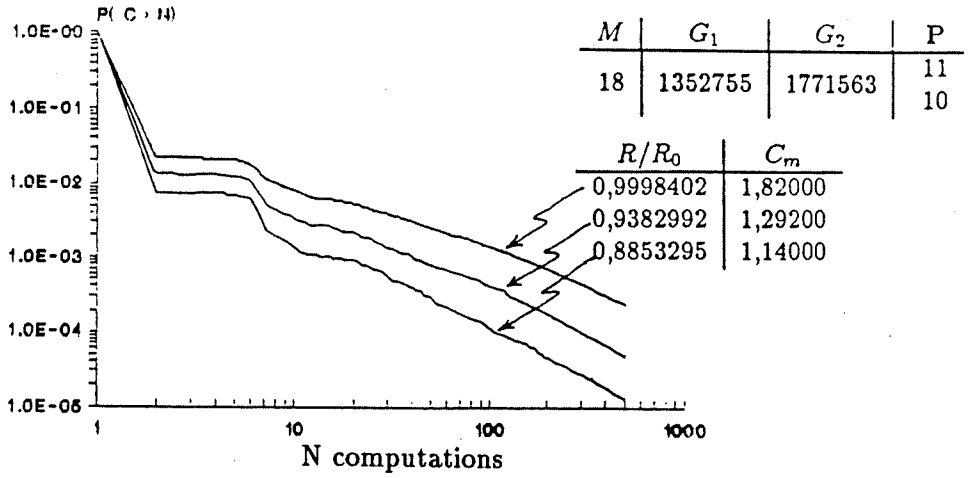
Figure 8: Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 6/7$, hard quantization.



M	G_1	G_2	P
10	2335	3661	1010111
			1101000

R/R_0	C_m
0,9962902	1,39643
0,9648232	1,18036
0,9397232	1,08929

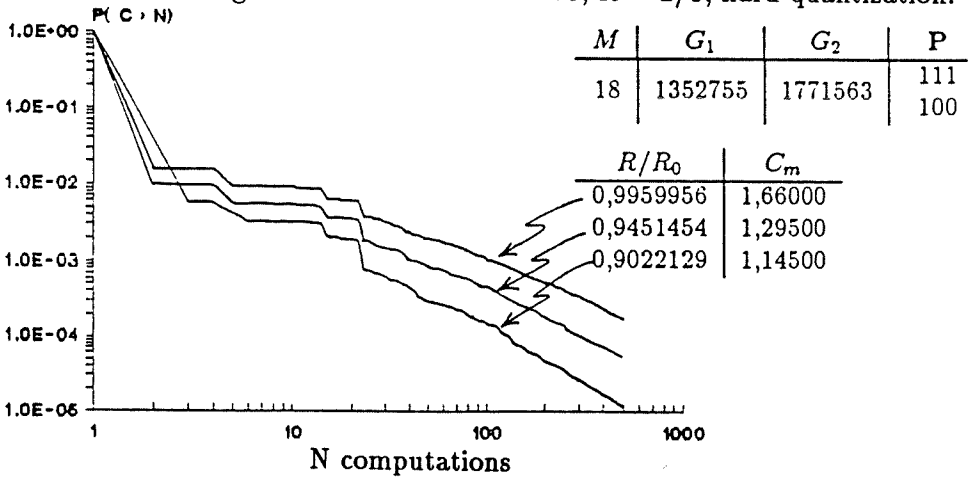
Figure 9: Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 7/8$, hard quantization.



M	G_1	G_2	P
18	1352755	1771563	11 10

R/R_0	C_m
0,9998402	1,82000
0,9382992	1,29200
0,8853295	1,14000

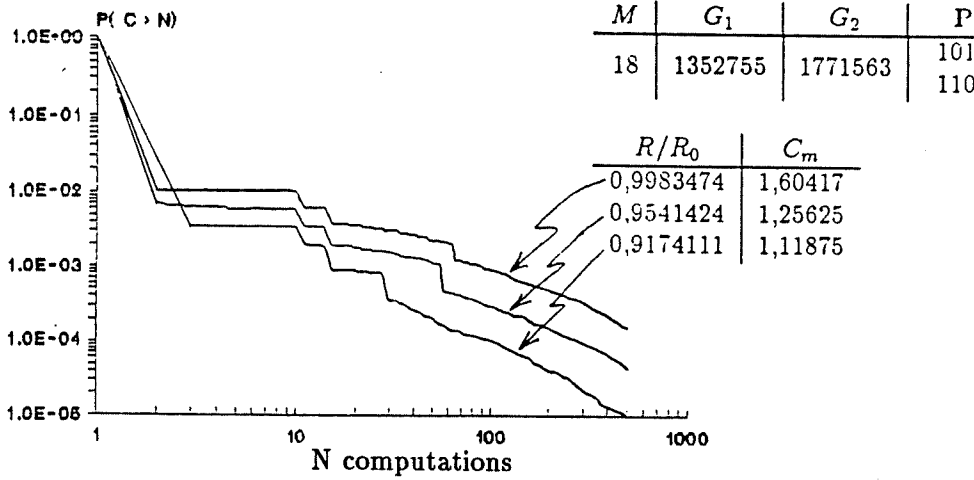
Figure 10: Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 2/3$, hard quantization.



M	G_1	G_2	P
18	1352755	1771563	111 100

R/R_0	C_m
0,9959956	1,66000
0,9451454	1,29500
0,9022129	1,14500

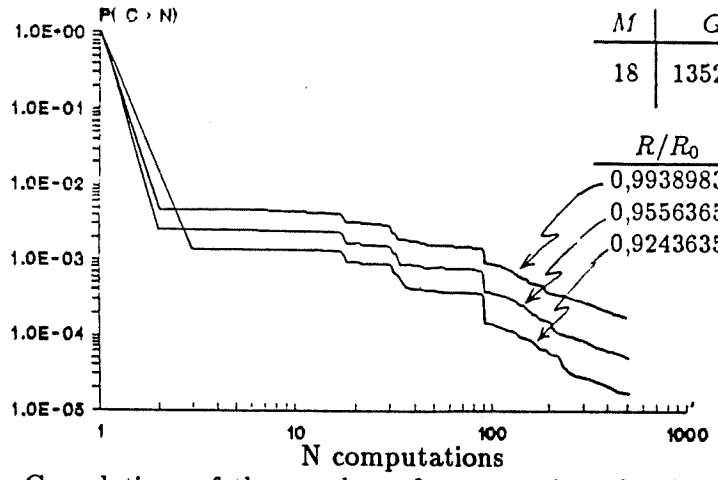
Figure 11: Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 3/4$, hard quantization.



M	G_1	G_2	P
18	1352755	1771563	1011 1100

R/R_0	C_m
0,9983474	1,60417
0,9541424	1,25625
0,9174111	1,11875

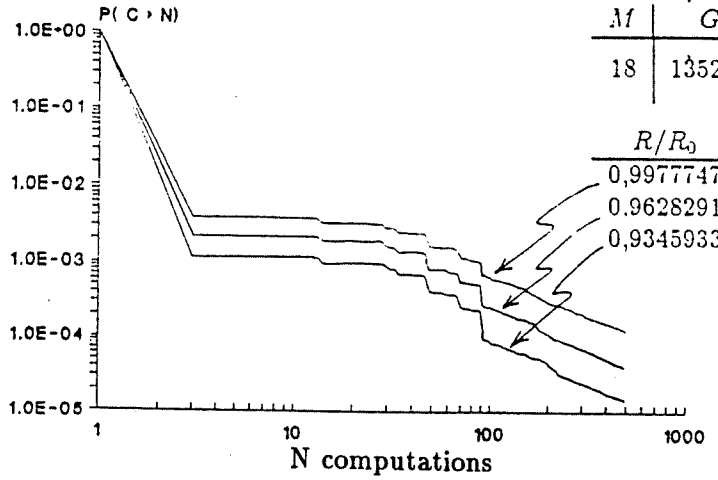
Figure 12: Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 4/5$, hard quantization.



M	G_1	G_2	P
18	1352755	1771563	11111 10000

R/R_0	C_m
0,9938983	1,51800
0,9556365	1,23400
0,9243635	1,12200

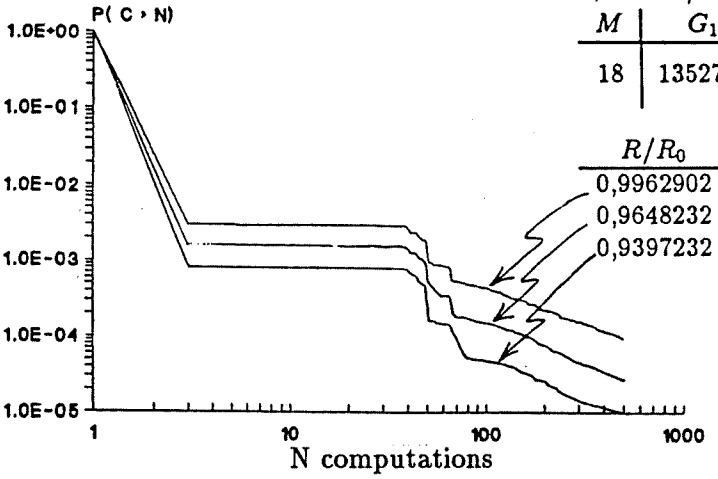
Figure 13: Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 5/6$, hard quantization.



M	G_1	G_2	P
18	1352755	1771563	111111 100000

R/R_0	C_m
0,9977747	1,42500
0,9628291	1,20000
0,9345933	1,10833

Figure 14: Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 6/7$, hard quantization.



M	G_1	G_2	P
18	1352755	1771563	1101101 1010010

R/R_0	C_m
0,9962902	1,34643
0,9648232	1,16071
0,9397232	1,08571

Figure 15: Cumulatives of the number of computations for decoding one branch for the stack decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 7/8$, hard quantization.

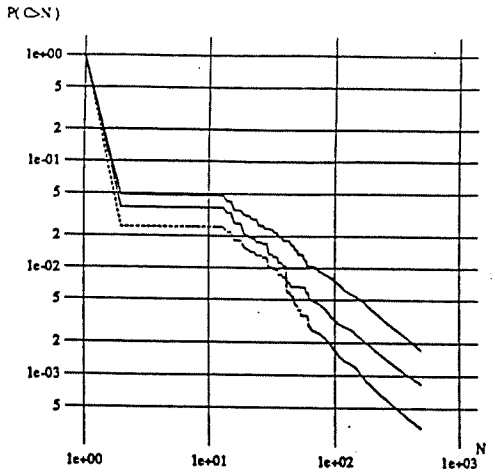


Figure 16: Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 2/3$, hard quantization.

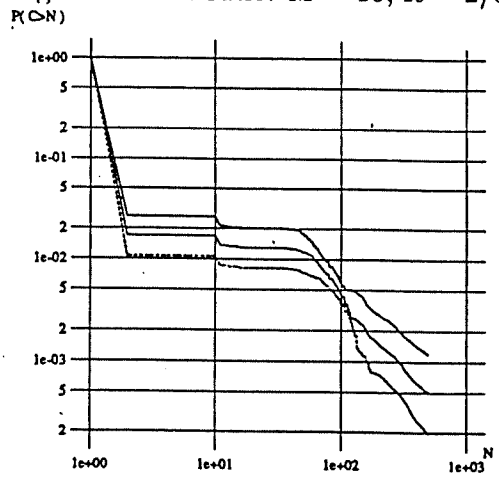


Figure 17: Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 3/4$, hard quantization.

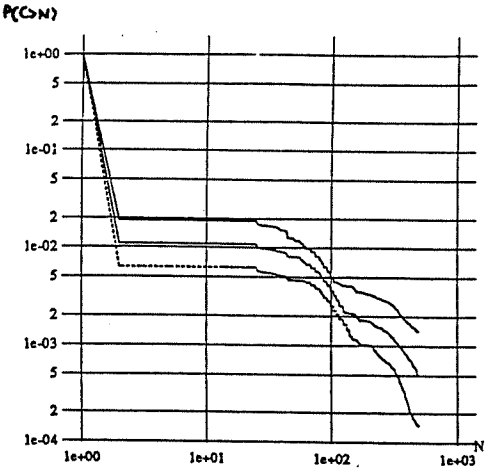


Figure 18: Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 4/5$, hard quantization.

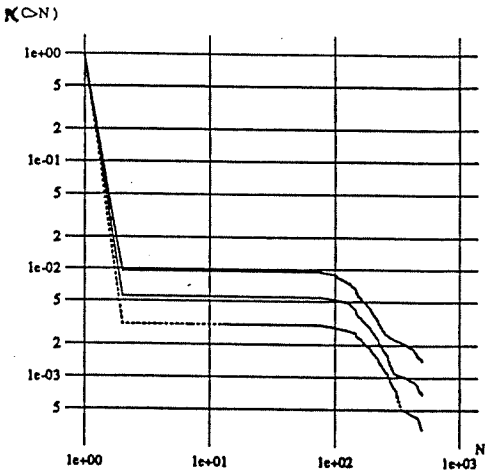


Figure 19: Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 5/6$, hard quantization.

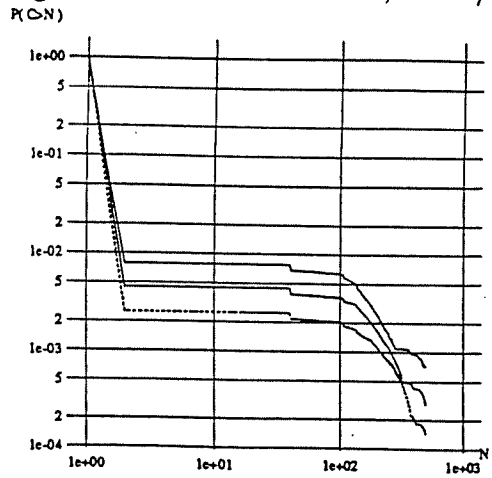


Figure 20: Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 6/7$, hard quantization.

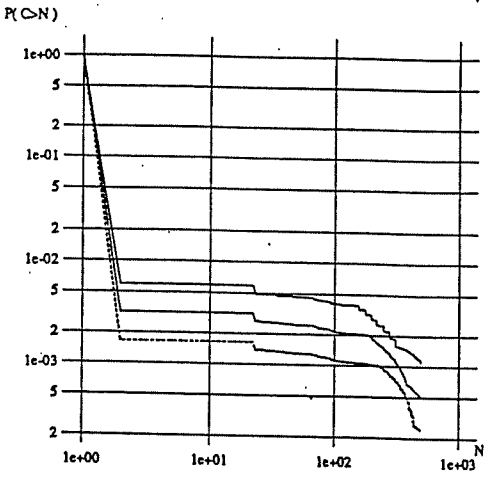


Figure 21: Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 10$, $R = 7/8$, hard quantization.

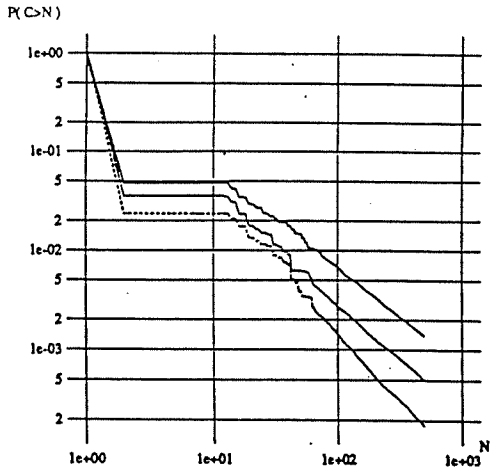


Figure 22: Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 2/3$, hard quantization.

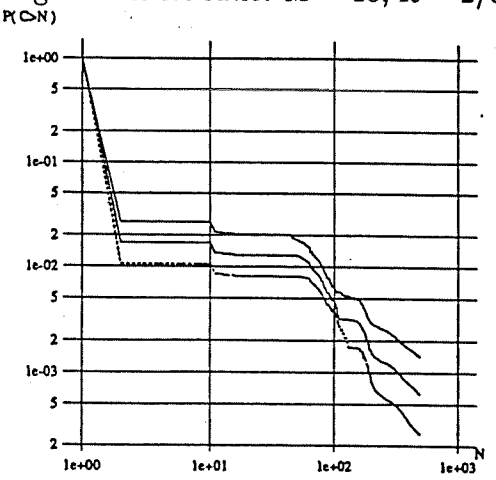


Figure 23: Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 3/4$, hard quantization.

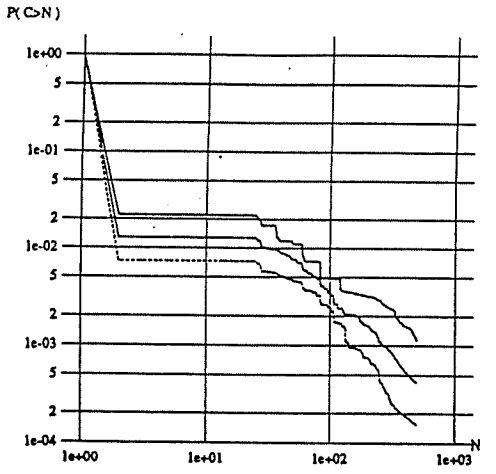


Figure 24: Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 4/5$, hard quantization.

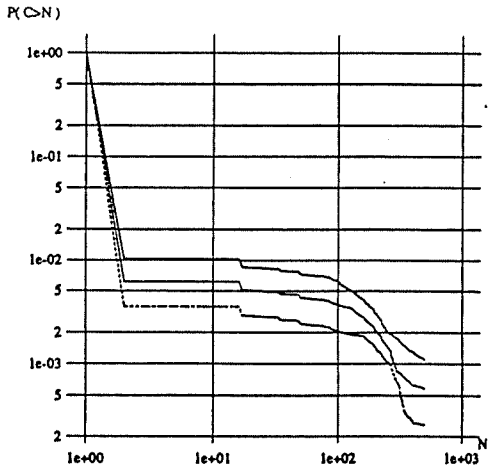


Figure 25: Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 5/6$, hard quantization.

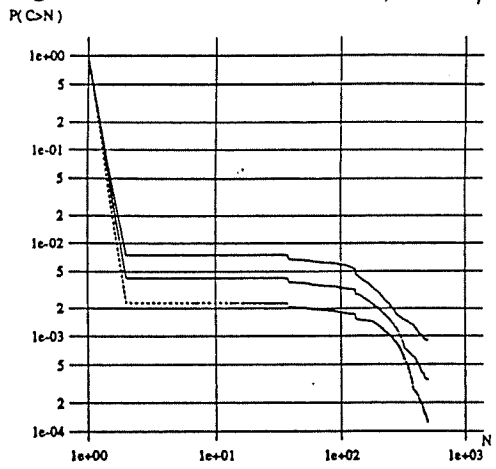


Figure 26: Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 6/7$, hard quantization.

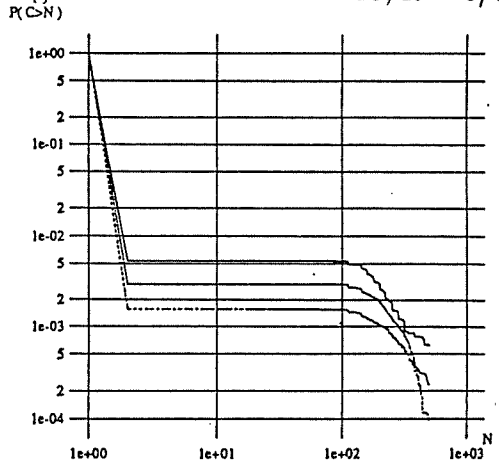


Figure 27: Cumulatives of the number of computations for decoding one branch for the Fano decoder, for three values of signal-to-noise ratio. $M = 18$, $R = 7/8$, hard quantization.

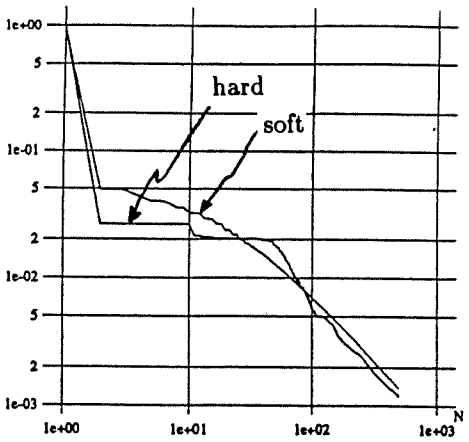
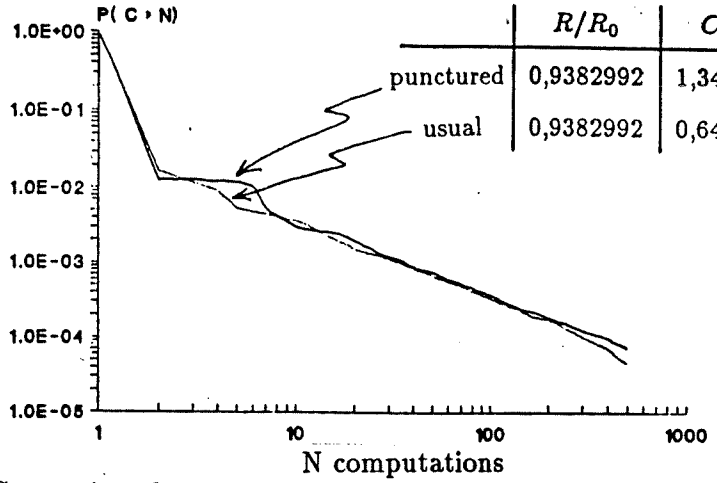
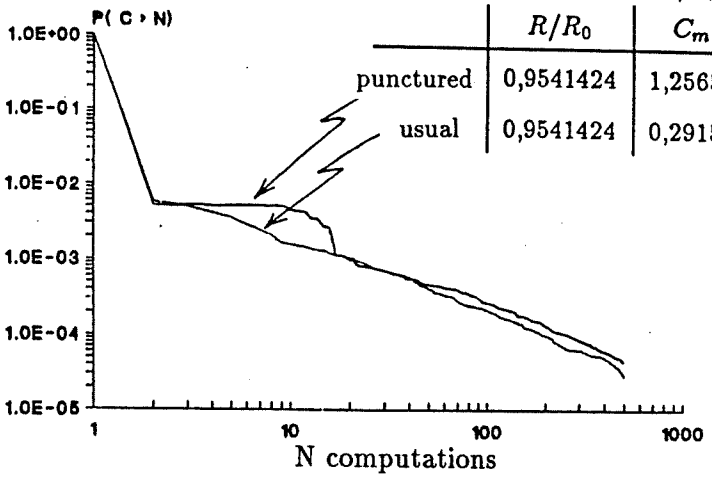


Figure 28: Cumulatives of the number of computations for decoding one branch for the Fano decoder, soft and hard quantization. $M = 10$, $R = 3/4$



	R/R_0	C_m
punctured	0,9382992	1,34400
usual	0,9382992	0,64000

Figure 29: Comparison between cumulative distributions of the number of computations for decoding one branch for punctured and usual stack decoders. $R = 2/3$, hard quantization.



	R/R_0	C_m
punctured	0,9541424	1,25650
usual	0,9541424	0,29150

Figure 30: Comparison between cumulative distributions of the number of computations for decoding one branch for punctured and usual stack decoders. $R = 4/5$, hard quantization.

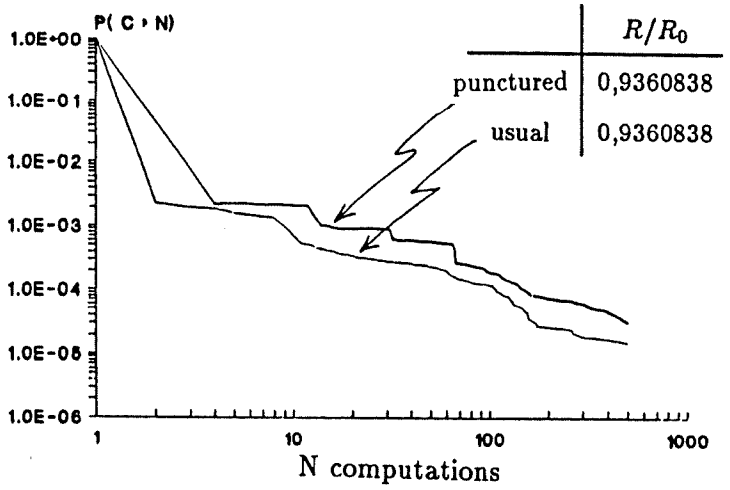


Figure 31: Comparison between cumulative distributions of the number of computations for decoding one branch for punctured and usual stack decoders. $R = 5/6$, hard quantization.

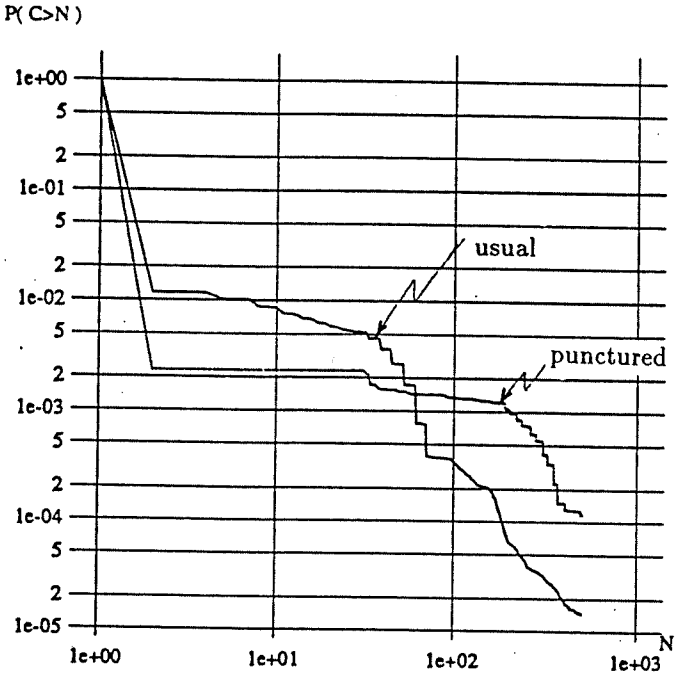


Figure 32: Comparison between cumulative distributions of the number of computations for decoding one branch for punctured and usual Fano decoders. $R = 5/6$, hard quantization.

ÉCOLE POLYTECHNIQUE DE MONTRÉAL
3 9334 00289712 0