DÉPARTEMENT DE GÉNIE ÉLECTRIQUE

SECTION COMMUNICATION-INFORMATIQUE

Rapport Technique EP79-R-12

Classification: Library of Congress no ......

PROBABILISTIC DECODING OF CONVOLUTIONAL CODES BY STACK ALGORITHMS*

par

David Haccoun, Ph.D.
Professeur Agrégé

Ecole Polytechnique de Montréal

Campus de l'Université
de Montréal
Case postale 6079
Succursale 'A'
Montréal, Québec
H3C 3A7

PROBABILISTIC DECODING OF CONVOLUTIONAL CODES BY STACK ALGORITHMS*

DAVID HACCOUN Ph.D.

Professeur Agrégé
Département de Génie Electrique
Ecole Polytechnique de Montréal

Don

RAPPORT TECHNIQUE No. EP 79-R-12

ECOLE POLYTECHNIQUE DE MONTREAL
Mars 1979

* This research has been supported in part by a National Research Council of Canada Grant No A-9336 and Quebec Ministry of Education FCAC Grant CRP 377-76.

PROBABILISTIC DECODING OF CONVOLUTIONAL CODES BY STACK ALGORITHMS*

by

DAVID HACCOUN

ABSTRACT

Methods are presented to alleviate some of the difficulties associated with the two principal probabilistic decoding techniques for convolutional codes: Sequential decoding and Viterbi decoding. It is shown that by extending several paths simultaneously and by discarding useless paths from the stack, both sequential and Viterbi decoding appear as special cases of a larger class of decoding algorithms using a stack. It is shown from computer simulations results that by adapting the amount of computation to the prevalent noise conditions on the channel, considerable improvement on the computational variability can be achieved at a cost of a modest increase in average computation and storage. Finally some recent progress on the buffer overflow problem are reviewed.

PROBABILISTIC DECODING OF CONVOLUTIONAL CODES BY STACK ALGORITHMS*

by

DAVID HACCOUN

1. INTRODUCTION

In a digital communication system employing Forward Error Correction, a data source generates binary information symbols at the rate R_s bits/sec. These information symbols are encoded for the purpose of error protection and the encoder output is a binary sequence of rate R_n symbols/sec. The code rate R in bits/symbols is then given by

$$R = R_s/R_n \tag{1}$$

Since $R < 1$, the transmission symbol speed on the channel is larger than the data speed delivered by the source, or equivalently, the introduction of an error control coding requires a bandwidth expansion.

Let the signal power received at the receiver be P and let N_o be the spectral density of the channel noise. Then the signal-to-noise ratio per information bit

$$E_b/N_o = P/(N_o R_s) \tag{2}$$

serves as a figure of merit for different combinations of coding and modulation schemes. It is clear that a coding or modulation scheme which reduces the E_b/N_o required for a given error probability leads to an increase in allowable data rate and/or a decrease in the necessary P/N_o. The problem then is determining the system that will operate at the lowest E_b/N_o with a given quality.

It is well known that whenever the information rate R_s is less than the channel capacity C, there exists some hypothetical coding scheme that gives a decoded error probability as small as desired. Channel capacity for an infinite bandwidth white gaussian noise is

$$C_\infty = \frac{P/N_o}{\ln 2} \quad \text{bits/sec} \qquad (3)$$

Using (2) we then have

$$E_b/N_o = \frac{P/N_o}{R_s} \geq \frac{P/N_o}{C_\infty} = \ln 2 \qquad (4)$$

The lower bound on E_b/N_o is thus about -1.6dB. A noncoded scheme such as coherent PSK requires an E_b/N_o of 9.6dB for a bit error rate of 10^{-5}. Hence a coding gain of 11.2dB is theoretically possible on the required E_b/N_o for this bit error rate.

For discrete memoryless channels (such as the space channel), systems employing convolutional encoding at the input and probabilistic decoding at the output are among the most attractive means of approaching the reliability of communication promised by the theory. Probabilistic decoding refers to techniques where the decoded message is obtained by probabilistic considerations rather than by a fixed set of algebraic operations. Moreover no particular algebraic structure is imposed on the code which may be chosen at random. The two principal probabilistic decoding techniques for convolutional codes are sequential decoding [1] and Viterbi decoding [2].

In general, an optimum detection method (demodulation and decoding) for block coding consists in coherently correlating each received block with all blocks that could possibly have been transmitted. In practice, unless the block is very short such an optimum procedure is just too complex to implement. Hence, most practical detectors consist of a demodulator producing "hard" decisions, followed by a digital decoder operating on these hard decisions. This hard quantization of the demodulator leads to a loss of performance that may reach some 2.0dB in E_b/N_o. By "soft-quantizing" the demodulator output into many more than the two regions of a hard decision device much of the lost performance can be regained.

Until recently most of the practical decoding techniques for block codes have been penalized by a hard decision demodulation. However, recent developments in semiconductor devices have made the application of soft-decision techniques to block coding systems both possible and practical [3]. Results from tests over practical HF channels have demonstrated improvements in the error-rate of 3 to 6 orders of magnitude.

By comparison, soft-decision techniques have been easily incorporated in the probabilistic decoding of convolutional codes. Soft decision Viterbi decoding can provide a coding gain of 5 to 6dB in required E_b/N_o at 10^{-5} bit error rate, whereas soft decision sequential decoding can easily provide a 7 to 8dB gain. The applicability of these decoding techniques to satellite channels has been widely demonstrated [4].

We will present in this paper some recent developments which tend to alleviate the computational difficulties with these two decoding algorithms: variability of the computational effort and buffer overflow problem of sequential decoding, and reduction of the large computational effort of Viterbi decoding. Using the concepts of both Viterbi and sequential decoding, we will present in some detail a class of "generalized" stack decoding algorithms which fills the gap between these two seemingly different algorithms and unifies them. Furthermore these algorithms can easily adapt their decoding effort to the prevalent noise conditions on the channel.

We assume the reader familiar with the tree and trellis structure of convolutional codes. Considering only binary convolutional codes of rate $1/V$ and finite constraint length K, 2 branches each with V coded symbols emerge from each node, and beyond the $(K-1)^{th}$ level there are 2^{K-1} distinct states. A path in the tree is specified by the input information sequence that entered the encoder, and two paths remerge after their input sequences have been identical over the last $(K-1)$ symbols; the tree collapses in a trellis. For an L - bit long input sequence, the trees or trellis are finite with L levels, and the longest paths have L branches.

2. SEQUENTIAL AND VITERBI DECODING

The suboptimum tree search of sequential decoding and the optimal treillis search of Viterbi decoding are two techniques which attempt to find the "best" path $\underline{U} = (u_1, u_2, u_3, \ldots)$ through a graph in which the branches are assigned "branch metrics" $\{\gamma\}$. These branch metrics are essentially the log-likelihood function $\log P(\underline{y}_i \mid \underline{x}_i^{(\underline{U}')})$ between the coded symbols $\underline{x}_1^{(\underline{U}')}$ of the branch corresponding to a possible data input u_i', and the channel output sequence $\underline{y}_i$ corresponding to the actual data input symbol u_i. Assuming a Discrete Memoryless Channel, the branch metrics are cumulative, so that over a graph of length L branches the objective of either decoder is to find the path $\underline{U}$ for which the total metric

$$\Gamma_L^{(\underline{U})} = \sum_{i=1}^{L} \gamma_i \qquad (5)$$

is maximum over all possible transmitted paths. For equally likely input data sequences, the decoded path $\underline{U}$ is the "best" in the sense that the sequence error probability is minimized.

Viterbi and sequential decoding have developed independently and appear to be the opposite for determining the most likely information sequence given the received sequence. The Viterbi algorithm uses the trellis structure of the code and examines all distinct paths at every trellis level, whereas a sequential decoder uses the tree structure of the code and follows only that path in the tree that appears to be the most likely. As a consequence the computational effort is constant but large for Viterbi decoding, whereas it is on the average typically very small but variable for sequential decoding.

SEQUENTIAL DECODING: The central idea of sequential decoding is the decoding of the received message one bit at a time, without searching the entire tree. Starting from the origin of the tree, the path selected to be explored one step further is the path whose metric is the largest among those already examined. The path that first reaches the last level of the tree with the highest metric is accepted as the decoded path. As the decoding proceeds, the decoder occasionally retreats in the tree and extends earlier and possibly incorrect paths. In order to minimize this backing-up and extension of unlikely paths, the metric Γ is biased in such a way that on the average it increases along the correct path and decreases along all incorrect paths.

A node $\underline{U}_j$ lying at level j on correct path $\underline{U}$ is called breakout if

$$\Gamma_j^{(\underline{U})} \leq \Gamma_i^{(\underline{U})} \quad , \quad i \geq j \qquad (6)$$

Breakout nodes play an important role in sequential decoding as they are decoded by a single computation. Furthermore the decoder never backs-up beyond the level of the last decoded breakout node. For non-breakout nodes the decoding effort becomes variable and increases exponentially with the size of the correct path metric dip.

There are two main sequential decoding algorithms: the Fano algorithm [5] and the Zigangirov-Jelinek (Z-J) algorithm [6]. In this paper we will consider only the Z-J algorithm .

In the Z-J or stack algorithm the decoder consists of a list or stack of the already searched paths, ordered in decreasing order of their metric values. The "top" of the stack has the largest accumulated metric and will be searched further, i.e. extended one level further along both branches emerging from the end node. The operations of the stack decoder are thus the finding of the top node, the extension and storage of its successors, and the proper reordering of the stack. As a node is extended, it is removed from the stack.

Sequential decoding involves a random motion in the tree, leading to a variable number of computations to decode a given block of information. The number of computations necessary to decode one information symbol has a Pareto distribution, that is a distribution whose tail decreases only algebraically [7]. This variability of the computational effort constitutes one of the main drawbacks of sequential decoding.

VITERBI DECODING: The Viterbi decoding algorithm is a simple decoding procedure which determines the path having the largest accumulated metric of all possible distinct paths. It uses the trellis structure of the code and retains at each level only the best path that terminates at each of the 2^{K-1} states. At each decoding step the 2^{K-1} remaining or "surviving" paths are extended, and their total metrics compared pair-wise so that for each 2 paths merging at each state only the path with the largest metric is retained. With this procedure none of the discarded paths can ever be the most likely path.

Although the biased metric of sequential decoding could also be used for Viterbi decoding, this bias is inconsequential here since all surviving paths have the same length: the decoder never backs-up in the trellis. The Viterbi operations are identical from level to level and since they must be performed at every state, the complexity of the decoder and number of computations per decoded bit grows exponentially with the constraint length of the code. This exponential growth limits Viterbi decoding to short constraint length codes ($K < 10$).

3. GENERALIZED STACK DECODERS

From the preceding descriptions the Z-J and Viterbi decoding algorithms appear to be quite different decoding techniques. Keeping the notion of a stack to store and order the explored paths, but allowing for the simultaneous extension of several paths from the stack and the elimination of undesirable paths, we now show that these two algorithms are only particular cases of more general stack decoders.

Let a path extension cycle consist of computing the metrics of the successors of the extended paths, entering them in their proper place in the stack, and removing from the stack the nodes just extended. The extension of the M^{th} node, $M \geq 1$, implies the simultaneous extension of all other $(M-1)$ nodes stored higher in the stack. The actual number of extended paths is assumed determined by some decision rule. Following the path extension cycle, a purging cycle eliminates from the stack any unwanted node according to some purging rule and reorders the stack. The decision and purging rules are specified by the users and need not be kept fixed for the entire decoding. They may be varied from cycle to cycle according to any information pertinent to the decoding as it progresses (past stack contents, channel measurements, etc). The set of operations comprising a path extension cycle and a purging cycle is defined as a decoding cycle. An algorithm that uses this decoding cycle at every step is called a generalized stack algorithm (G.S.A.) [8].

Both the Z-J and Viterbi algorithms may be viewed as two particular cases of this G.S.A. In the Z-J algorithm the decoding cycle is degenerate since it contains no purging cycle and the decision rule directs the extension of only the top node of the stack.

Now suppose the Viterbi decoder uses a stack to store and order all explored paths. For a binary code the decision rule is simply to extend the $M_v \triangleq 2^{K-1}$ highest nodes in the stack (the so-called survivors) at every path extension cycle. The purging rule is then recognized as the elimination of all non surviving paths from the stack. All redundancy is thus eliminated from the stack whereas the M_v - path extension cycle garantees the optimality of the decoding and the constancy of the computational effort. Therefore the Viterbi algorithm is also a particular case of the G.S.A.

In conclusion, depending on the particular decision and purging rules, the G.S.A. allows a class of decoding algorithms among which the Z-J and Viterbi are only two members. A class of algorithms can be obtained by modifying the Z-J algorithm in the following way: 1) Use a purging cycle to exploit remergers and eliminate all redundancy from the stack; and 2) Allow the extension of the M, $M \geq 1$ most likely paths. We shall see that these algorithms will yield a smaller computational variability than sequential decoding.

EXPLOITATION OF REMERGERS: When a sequential decoder backs-up in the tree in its search for a better path, it may follow a path that appears new, but because of remergers that path may have already been explored: the redundant information imbedded in the tree is not used by the decoder. By exploiting the reconvergence of the explored paths, some duplicate computations together with their stack storage may thus be avoided.

The Z-J algorithm has therefore been modified as follows: whenever two paths remerge, the remerging is recognized, the two accumulated metrics are compared and only the path yielding the highest metric is stored in the stack. A new branch extension converging onto a node already in the stack is discarded if it does not increase the total metric at that node. If the total metric is increased, the information about the new extension replaces the information of the node previously stored in the stack with the lower metric. With this procedure, the purging cycle is conveniently imbedded in the path extension cycle and all redundancy is removed from the stack. The stack structure of the Z-J algorithm allows a very simple way to determine the occurence of remergers and requires only an additional set of pointers to the stack [9].

This single-path G.S.A. is still a sequential decoder, and hence the random nature of the computation will persist with an asymptotic Pareto distribution. However since each node eliminated by remerging eliminates with it a potential subtree of incorrect paths that may be explored by a sequential decoder, the variability of the computation may be reduced. The improvement will depend on the occurence of remergers and will decrease with increasing K, the constraint length of the code.

Z-J ALGORITHM WITH A MULTIPLE-PATH EXTENSION CYCLE: We now examine the other main difference between the Z-J and Viterbi algorithms, i.e. the multiple path extension cycle. The introduction of an M-path extension cycle fills in some sense the gap between these two algorithms.

Suppose some node $\underline{U}_i$ on the correct path $\underline{U}$ is stored in the stack at the M^{th} position from the top. By the rules of the Z-J algorithm, the minimum number of computations necessary to extend $\underline{U}_i$ is M whereas the maximum is unbounded if the tree is infinite. Suppose now the decision rule allows the extension of the M highest nodes for each path extension cycle. Then node $\underline{U}_i$

will be extended at a cost of exactly M computations. If the correct path metric drops but the correct path were extended at every decoding cycle, then for all purposes this metric dip becomes inconsequential and non-breakout nodes behave as if they were breakout. This M-path stack algorithm would be similar to a Viterbi decoder with M computations per bit. By including the purging cycle described earlier, if $M = M_v = 2^{K-1}$, then clearly this algorithm becomes the Viterbi decoder.

In the usual sequential decoding, even small dips of the correct path may require a relatively large number of computations. This is not necessarily the case with the M-path decoder because the correct path may be extended through the dip while not at the top of the stack. The exploration of the subset of incorrect paths may not be exhaustive as for sequential decoding, and will improve the distribution of the computational effort.

As the channel gets noisier and the correct path metric drops further, M may not be large enough to include the correct path. Back searching for the correct path is thus possible, and a computational behaviour similar to that of sequential decoding is expected for periods of deep searches. Therefore although requiring more than M computations to decode one branch on the correct path is a less likely event with the M-path algorithm than it is for the usual sequential decoder, asymptotically the distributions of computation decrease at the same rate for both algorithms.

The M-path decoder being a suboptimal sequence estimator, its error probability is lower bounded by the error probability of the Viterbi decoder. With a simple intuitive argument [9], we can show that the error probability is upper bounded by that of the usual sequential decoder.

In conclusion, beyond M computations per decoded bit, the distribution for the M-path algorithm should appear as a downward translation of that of the Z-J or the Viterbi decoder, with an average number of computations per decoded bit at least equal to M. This algorithm stands between these two extremes and exchanges a reduction of the variability of the computational effort for an increase of its average value. The implementation of the multiple-path extension presents hardly any added complexity; a flow diagram of an M-path G.S.A. is shown in Figure (1).

COMPUTER SIMULATION RESULTS: In order to compare the performances of the Z-J and M-path G.S.A., both algorithms have been simulated on a computer and run under strictly identical conditions, using the same rate ½ code with antipodal signalling, and the same noise sequence. The output of the matched filter demodulator was quantized into 8 levels that is, a soft, 3 bit-decision was made on each received symbol. For gaussian noise, this soft-quantized channel is within 0.2dB of a continuous channel.

A search consists of all computation done between first reaching some node depth N and first penetrating to depth (N + 1). Empirical distributions of the number of computations per search for $M = 4$ and $E_b/N_o = 3$dB are shown in Figure (2) and Figure (3) for $K = 6$ and 7 respectively. The distributions of the Z-J algorithm are also given for reference.

As expected the distribution of computation for the 4-path G.S.A. shows an improvement over that of the Z-J algorithm, and beyond 5 computations per search, both curves tend to run parallel to one another. As shown in Figure (2), the run with $M = 1$ does not show much improvement over the Z-J algorithm but the slope of the tail appears to be slightly larger. This behaviour was also observed with other runs, which supports the conjecture that the variability of the compuration is reduced when remergers are exploited.

Some of the errors of sequential decoding were corrected as expected, and the average number of computations follows the relation

$$\overline{C}_{M\text{-path}} \leq (M-1) + \overline{C}_{Z\text{-}J} \qquad (7)$$

To summarize, the M-path generalized stack algorithm fills the gap between the single path sequential decoding and the all-path Viterbi decoding.

The simulation results verify the predicted improvements over sequential decoding. The improvements were obtained at a cost of a larger average decoding effort and stack storage. Although the purging cycle tends to reduce these two parameters, its importance has to be weighed against the additional decoder complexity and additional memory requirement. In the next section we examine methods to reduce further the variability of the computational effort without unduly increasing the average decoding effort of the M-path algorithm.

4. ADAPTIVE SEQUENTIAL DECODING

With the fixed decision rule of the M-path G.S.A., we observe that when the cahnnel is quiet, then M may be too large and computations are wasted. On the other hand when the noise gets large, then M may be too small and the decoder goes in a search for the correct path. Two variants of the fixed rule have been examined with the following objectives:

a) Decrease the average decoding effort without affecting the distribution of computations.

b) Reduce further the computational variability without unduly increasing the average decoding effort.

To implement variant (a), instead of extending M paths regardless of their metric values, only up to M paths are extended provided the metric difference between any of these paths and the top node of the stack is within some reasonable bound Γ_Δ. For example, simulation results with $\Gamma_\Delta = 130$ on a run identical to that of Figure (3) reduced the average decoding effort from 4.05 to 3.53 with hardly any difference in the distributions. Furthermore, the error probability was not affected.

For variant (b), clearly more than M-paths must be extended. But in order to prevent the average decoding effort from increasing, a greater number of paths must be extended only when the correct path metric goes through a dip. Since the correct path metric is unknown, this new decision rule is based on the values of successive top nodes of the stack. M-paths are extended as long as the top node metric is the largest over past values, but whenever the top node metric values drops below the maximum ever observed, then a dip is assumed and M'paths, M' > M, are extended. (Clearly monitoring the metric of the top of the stack constitutes a convenient form of channel measurement). We call this variant the (M/M')-path algorithm. Figure (4) shows the improvement obtained by the (4/6)-path and (2/6)-path over the 4-path algorithm.

The variants of the M-path algorithm indicate that using a fixed decision rule induces an unnecessarily large average decoding effort since most of the computations are wasted when the channel is quiet. Hence a clever decoder should use a rule that decides on just enough paths to include the decoded path in the present and future decoding cycles.

AN ADAPTIVE STACK ALGORITHM: We now present an adaptive G.S.A. where the number of paths to be extended depends on the relative metric values of the top nodes of the stack.

Let $\Gamma_n^{(Top)}$, n = 1, 2, ..., be the top node metric value for the current decoding cycle, and let $\Gamma_n^{(max)}$ be the largest metric ever observed,

$$\Gamma_n^{(max)} = \operatorname*{Max}_i \Gamma_i^{(Top)} \quad , n > i. \tag{8}$$

Define the current metric dip D as

$$D = \Gamma_n^{(max)} - \Gamma_n^{(Top)} \tag{9}$$

The decision rule is then as follows:
If $D \leq 0$, a single path is extended and the maximum metric value is updated,

$$\Gamma_{n+1}^{(max)} = \Gamma_n^{(Top)} \tag{10}$$

If D > 0, a number M(D) highest nodes of the stack are extended, where M(D) is a non-decreasing function of D. The maximum metric value remains unchanged,

$$\Gamma_{n+1}^{(max)} = \Gamma_n^{(max)} \tag{11}$$

With this procedure, the decision rule is the function M(D), and the distribution of the computation is expected to be improved over sequential decoding without increasing too severely the average decoding effort. Subjecting M(D) to some maximum value M_{max} computations per cycle, a search mode of operation would correspond to requiring more than M_{max} computations per search. Since occasionally the corect path will not be included in the M(D) extended paths, an asymptotic sequential decoding behaviour is still expected. The average decoding effort as well as the probability of entering a search mode of operation will depend on the particular function M(D) and on M_{max}.

SIMULATION RESULTS: This adaptive algorithm was programmed and run under the same condition as the previous algorithms. Results of a simulation with short constraint length codes are shown in Figure (5) to (7) for the linear functions.

$$M(D)=\begin{cases} a + bD, \ 0 \le a \le 6, \ 1 \le b \le 2 & , D > 0 \\ 1 & , D \le 0 \end{cases} \qquad (12)$$

We observe that for ($C \le M_{max}$), the distribution decreases very steeply whereas for ($C > M_{max}$), the straight line Pareto behaviour of sequential decoding is apparent. The distribution appears to improve with the constant "a" of the function M(D), and further tests have indicated that in general, as soon as a metric dip is detected, some minimum number of paths should be extended regardless of the size of the dip.

It was observed that for a given M(D), the distribution of computation improves up to a certain point with M_{max}, while $\overline{C}$ increases only very slowly. Hence there is an optimum value of M_{max} at which an adaptive G.S.A. should operate. Moreover extending more than a single path when no dip is detected is wasteful; $\overline{C}$ increases without improving the distribution of computation.

A limited comparison between these results indicates that for a given decision rule M(D), M_{max} and energy-to-noise ratio (E_b/N_o), the search mode probability is rather insensitive to K, but the slope of the tail of the distribution tends to increase with decreasing K. This may be explained by the increasing number of remergers as K decreases. Comparing the improvements achieved by the 4-path and adaptive G.S.A., over the Z-J algorithm, the adaptive algorithm appears to be clearly superior to the 4-path. This superiority is obtained at a smaller computational cost and increases with the channel measurement information used in the decision rule.

Several other adaptive procedures have been examined [10]. In particular, instead of determining the number M(D) of nodes to be extended, one could determine a "metric threshold" and extend <u>all nodes</u> in the stack subject to some maximum number which are above this threshold. Thus, the problem is to determine this metric threshold as a function of the observed metric dip D.

Another particularly simple adaptive scheme uses the data structure of the stack. In the Z-J algorithm the stack is not ordered exactly. Instead, the nodes are grouped in equivalence classes or "substacks": a node of metric Γ belongs to substack m if $mH \leq \Gamma < (m+1)H$, where H is some convenient small number (usually one maximum branch metric). Now the extension rule of the adaptive scheme is as follows: extend at all times all the nodes (subject to some maximum number) lying in the top S substacks. This rule assures a self-adaptivity because when the channel is quiet the highest substacks are almost empty, but when the channel becomes noisier more nodes are automatically extended. Figure (8) shows some typical results with this adaptive scheme.

A major theoretical difficulty (with practical consequences) with all these adaptive schemes consists in choosing the proper function M(D), or treshold value, or S..., from the observation of the metric dip. The problem was somewhat alleviated by scouting the behaviour of the metric via a "look-ahead" procedure [10]. However this procedure further complicates the decoding algorithms and remains heuristic. A fundamental description of the dynamic of the population of incorrect paths is needed. Some progress in this respect has been achieved [11-13] by modelling the correct path metric by a Markov Chain and the incorrect paths by a branching process.

5. BUFFER OVERFLOW

The computational variability of sequential decoding requires an input buffer to store the incoming data waiting to be decoded. The problem of this buffer overflow as well as the restarting procedures constitue one the drawbacks of this decoding technique. By reducing the variability of the computational effort all the variants of the G.S.A. alleviate somewhat the overflow problem. Unfortunately an immediate translation of the reduced variability into a corresponding reduction of the buffer overflow probability is not entirely garanteed [10].

A somewhat different approach of the buffer overflow problem was recently proposed [14]. In this technique the idea is to deliver a good estimate of the correct path within some computational limit C_{lim} dictated by the amount of storage available. Such a procedure assures a so-called "erasure free" decoding.

The basic algorithm is a Z-J sequential decoding algorithm, but instead of using a single stack, it uses several stacks; the algorithm is called "Multiple-Stack Algorithm". The operation of the algorithm is as follows: initially the MSA decoder operates exactly like the conventional single-stack Z-J. If a terminal node in the tree is reached before the first stack fills up, decoding is completed, and the decoded sequence is the same as if decoded by the Z-J algorithm. However if extensive searching is needed, then the first stack eventually fills up. In this case, the top T nodes of the first stack are transferred to a second stack and decoding resumes using only those T transferred nodes. If the end of the tree is reached before this stack is full, the terminal node is accepted as a "tentative" decision and stored in a special register. The decoder clears this second stack, then returns to the first stack and attempts to impose this tentative decision by resuming the decoding from the first stack using exactly the same procedure. The new "tentative" decision is compared to the preceding one, and the best one is retained and becomes the final decision.

It is shown [14] that this procedure yields as good a decision as a Z-J algorithm with an arbitrarily large stack.

The process of transferring T nodes to another stack is not limited to only 2 stacks. Additional stacks are formed as needed, and should a stack fills up before the end of decoding,T nodes are transferred to a higher stack until a tentative decision is made. The algorithm terminates decoding if it reaches the end of the tree in the first stack, or if the computational limit C_{lim} is reached while a search is in progress. In this case the best tentative decision is accepted as a final decision.

It is shown [14] that the MSA reaches a decision with an exponentially rather than Pareto distributed computational effort while yielding a practically erasure free performance. Simulation results indicate that the MSA can achieve the error performance of Viterbi decoding while requiring considerably less number of computation. The MSA may therefore be an attractive alternative to Viterbi decoding where low error probabilities are required at high decoding speeds.

6. CONCLUSION

The class of G.S.A. was shown to unify the Viterbi and sequential (Z-J) decoding algorithms. The multiple path extension together with the exploitation of remergers was shown to alleviate the principal draw back of sequential decoding at a cost of a moderate increase of the average decoding effort $\overline{C}$. In the adaptive G.S.A., the decision rule is modified according to some information contained in the stack, resulting in a reduced computational variability without an undue increase of the average decoding effort. As the decision rule M(D) estimates better the number of incorrect paths to be extended, these results will be further improved. However an asymptotic Pareto distribution appears to be inescapable.

The difficult problem of buffer overflow received a partial solution by these adaptive schemes, although the new Multiple Stack Algorithm gives a better solution. In all these algorithms the speed advantage is achieved at the expense of additional stack and buffer storage, but this is becoming tolerable in view of the rapid progress in large scale storage devices. In fact at Ecole Polytechnique de Montreal we are in the process of building stack sequential decoders based on microprocessors and using considerable parallelism in the operations in order to achieve high decoding speeds.

Séquential decoding have tradionnally been contemplated for memoryless channels. However it has recently been successfully used on burst-error channels [15]. The decoding was modified beyond simple interleaving. The idea is to let the decoder utilize as much information about the channel state as it is available at the current state of decoding from the already decoded data. Again, this information is conveniently passed to the decoder via the metrics which control the searching operation.

Finally we have approached the problem of unifying Viterbi and sequential decoding by modifying sequential decoding. Another approach consists in reducing the computational load of Viterbi decoding by reducing the number of states in the trellis. Such a "reduced-state" technique met with success in the intersymbol interference problem, but unfortunately was somewhat disappointing when used in conjunction with error correction [16], [17].

REFERENCES

[1] Wozencraft, J.M., "Sequential Decoding for Reliable Communications", Sc. D. Thesis, MIT 1957.

[2] Viterbi, A.J., "Convolutional Codes and their Performance in Communication Systems", IEEE Trans. on Comm. Tech., Vol. COM-19, Oct. 1971.

[3] Farrell, P.G., "Soft Decision Minimum Distance Decoding",Nato Advanced Study Institute, Darlington, U.K., Aug. 1977.

[4] Heller, J.A., and Jacobs, I.M., "Viterbi Decoding for Satellite and Space Communication", IEEE Trans. on Comm. Tech. Vol. COM-19, Oct. 1971.

[5] Wozencraft, J.M. and Jacobs, I.M., "Principles of Communication Engineering" , Chap. 6, John Wiley, 1965.

[6] Jelinek , F.,"A Fast Sequential Decoding Algorithm using a Stack", IBM Journal of R. and D., Nov. 1969.

[7] Jacobs, I.M.,and Berlekamp, E.R., "A lower bound to the Distribution of Computation for Sequential Decoding", IEEE. Trans. on Inf. Th., Vol. IT-13, April 1967.

[8] Haccoun, D., and Ferguson, M.J., "Generalized Stack Algorithms for Decoding Convolutional Codes", IEEE Trans. on Inf. Th., Vol. IT-21, pp. 638-651, Nov. 1975.

[9] Haccoun, D., "Multiple-path Stack Algorithms for Decoding Convolutional Codes", Ph-D. Dissertation, McGill Univ., June 1974.

[10] Janelle, A., "Decodage Adaptatif des Codes Convolutionnels", M.Sc. App. Thesis, Dept. of E.E., Ecole Polytechnique de Montreal, April 1978.

[11] Haccoun, D., "A Markov Chain Analysis of Sequential Decoding", to appear in IEEE Trans. on Inf. Theory.

[12] Haccoun, D., "Branching Processes and Sequential Decoding", Proc. 1977 IEEE Int. Symp. on Inf. Th., Cornell Univ. Oct. 1977.

[13] Johannesson, R., "On the Computational Problem with Sequential Decoding", Proc. 1976 IEEE Int. Symp. on Inf. Th., Ronneby, Sweden, June 1976.

[14] Chevillat, P.R., and Costello, D.J., "A Multiple Stack Algorithm for Erasurefree Decoding of Convolutional Codes", IEEE Trans. on Comm., Vol-COM-25, Dec. 1977.

[15] Hagenauer, J., "Sequential decoding for burst-error channels", Nato Advanced Study Institute, Darlington, U.K., August 1977.

[16] Conan, J. and Haccoun, D., "Reduced-State Viterbi Decoding of Convolutional Codes", Proc. 14th Allerton Conference on Circuit and System Theory, Oct. 1976.

[17] Chammas, J., "Performance du décodeur de Viterbi réduit pour le décodage des codes convolutionnels", Thèse M.Sc.A., Département de Génie Electrique, Ecole Polytechnique de Montréal, Janvier 1979.

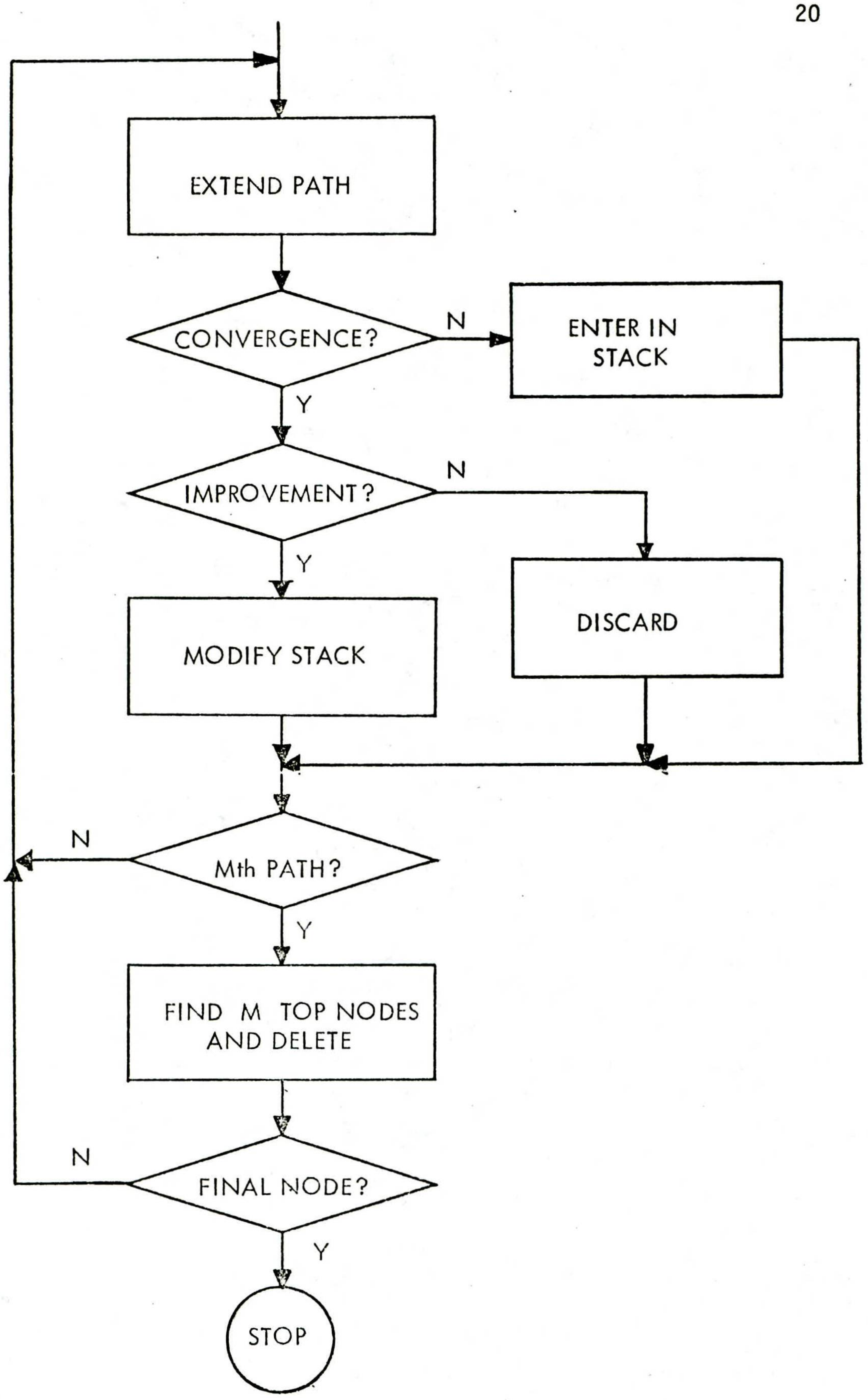

Figure (1) Flow diagram of the M-path algorithm.

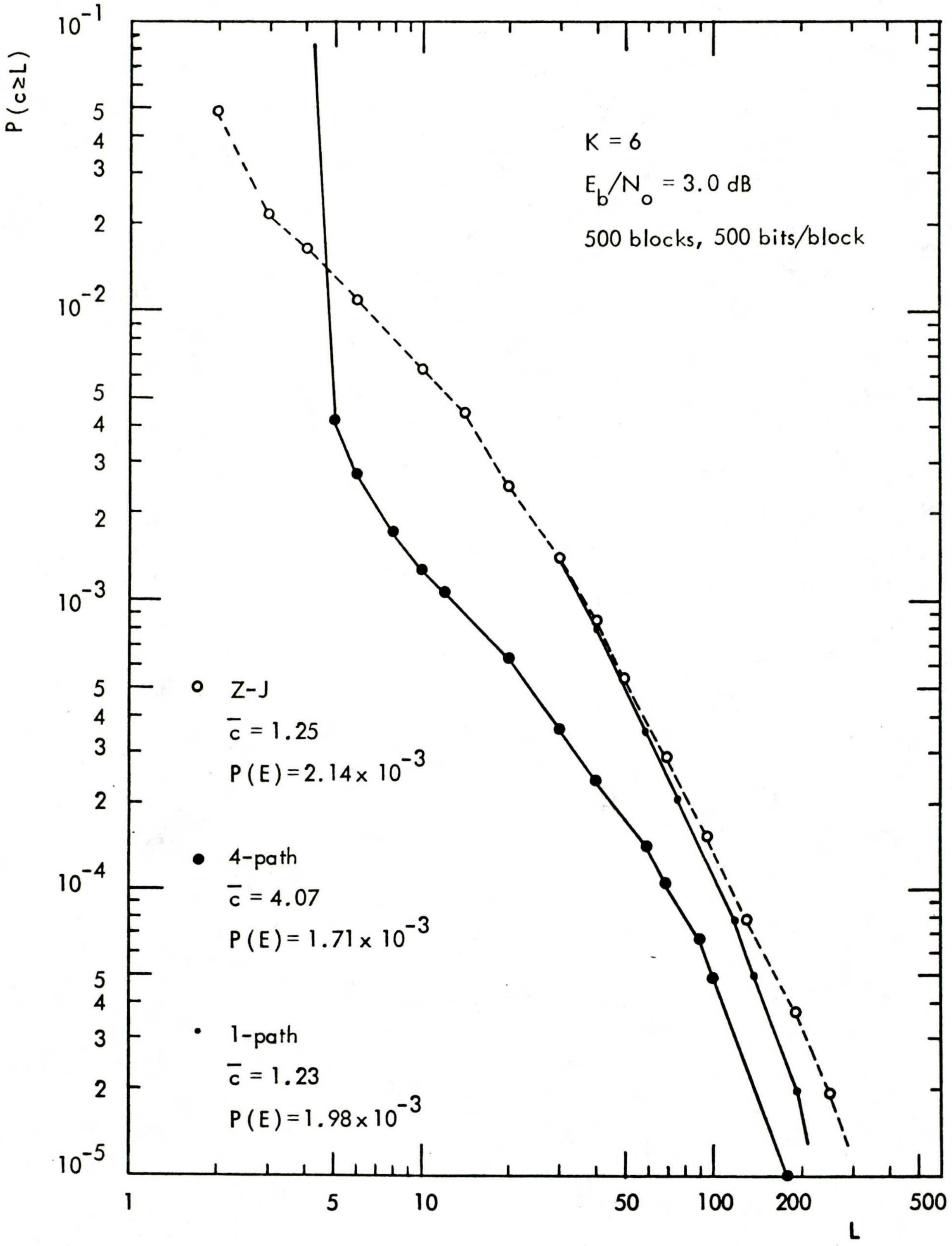

Figure (2) Empirical distribution of computations per search for the M-path Algorithm.

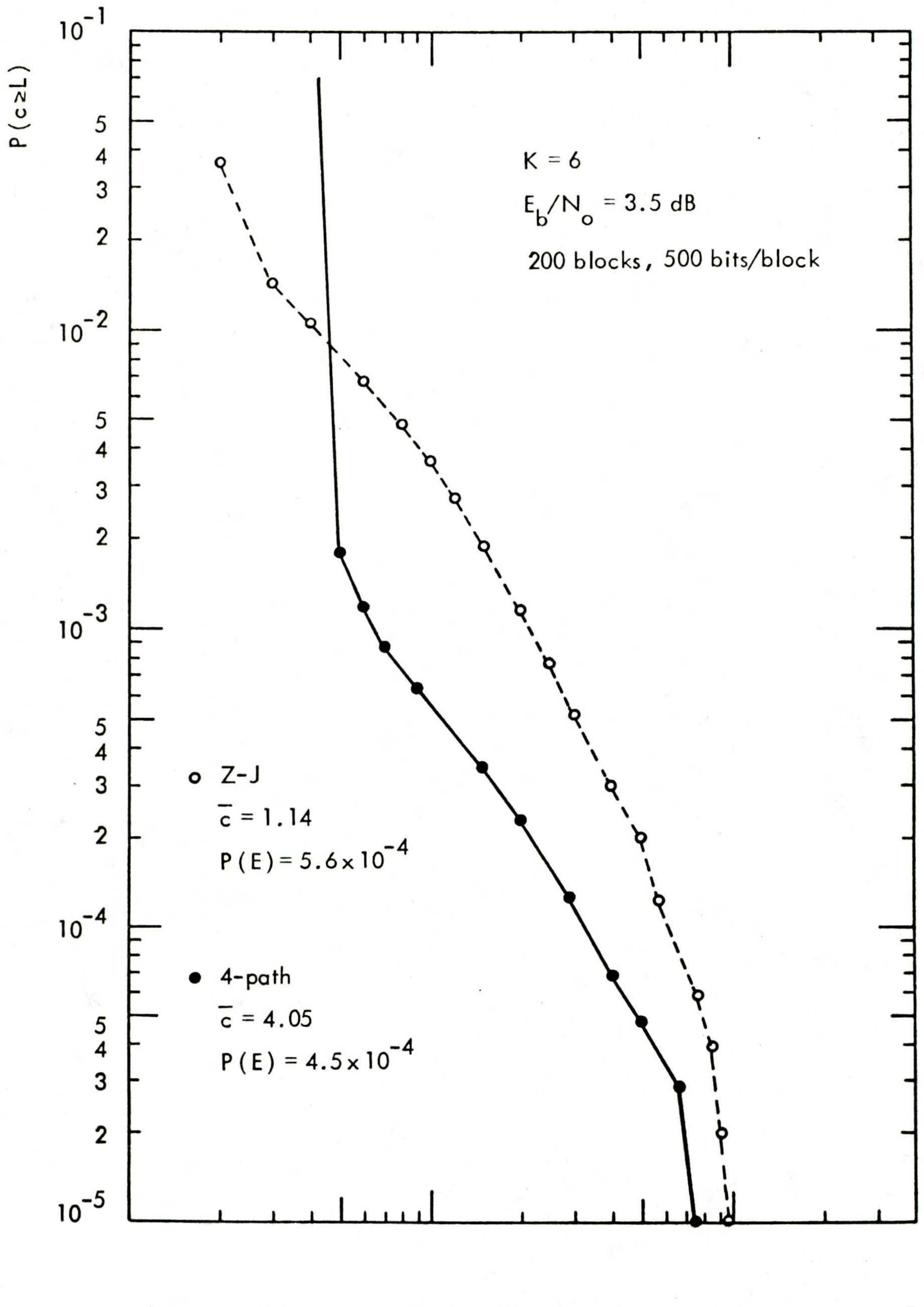

Figure (3) Empirical distribution of computations per search for the M-path Algorithm .

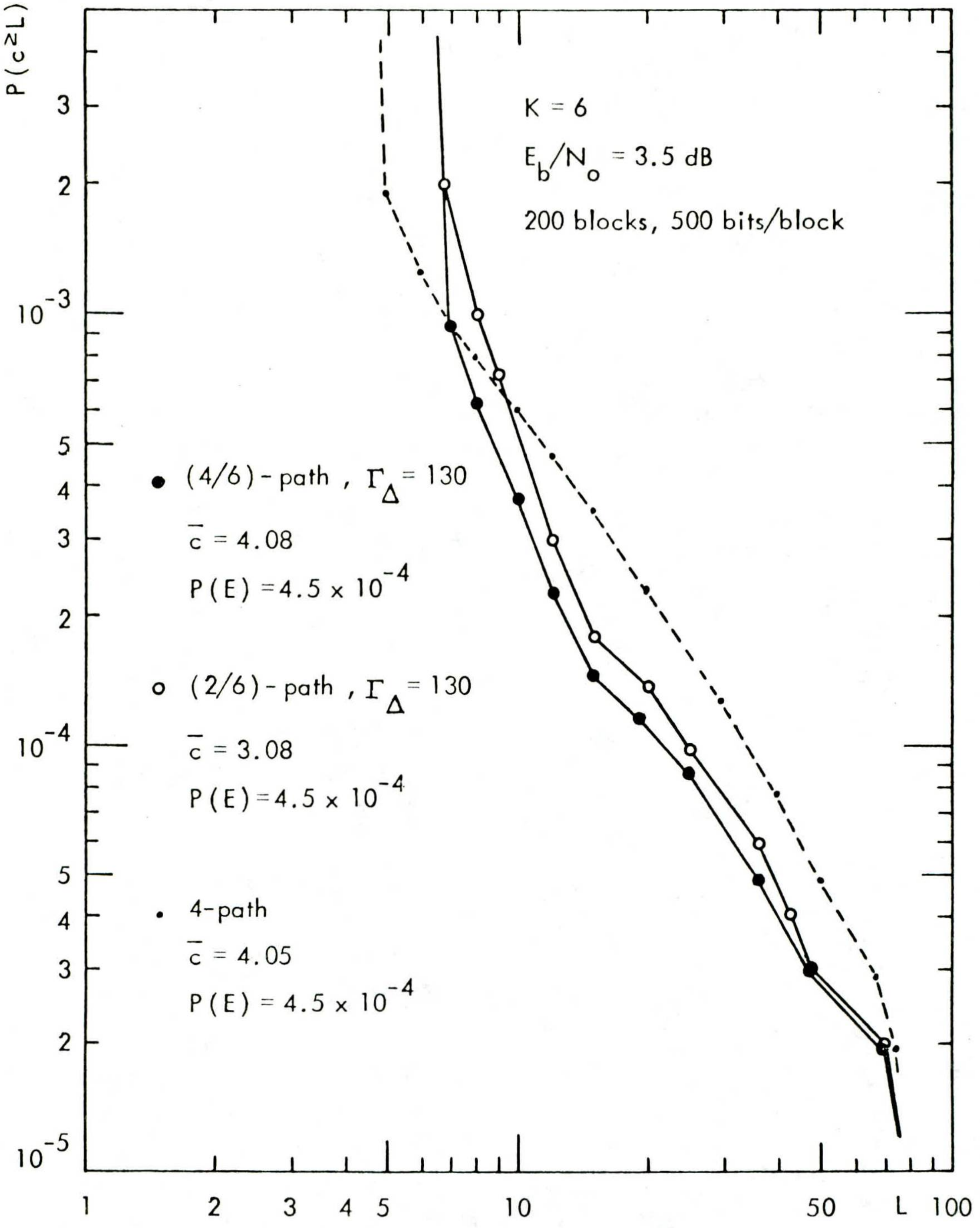

Figure (4) Empirical distribution of computations per search for the variable (M/M') - path algorithm.

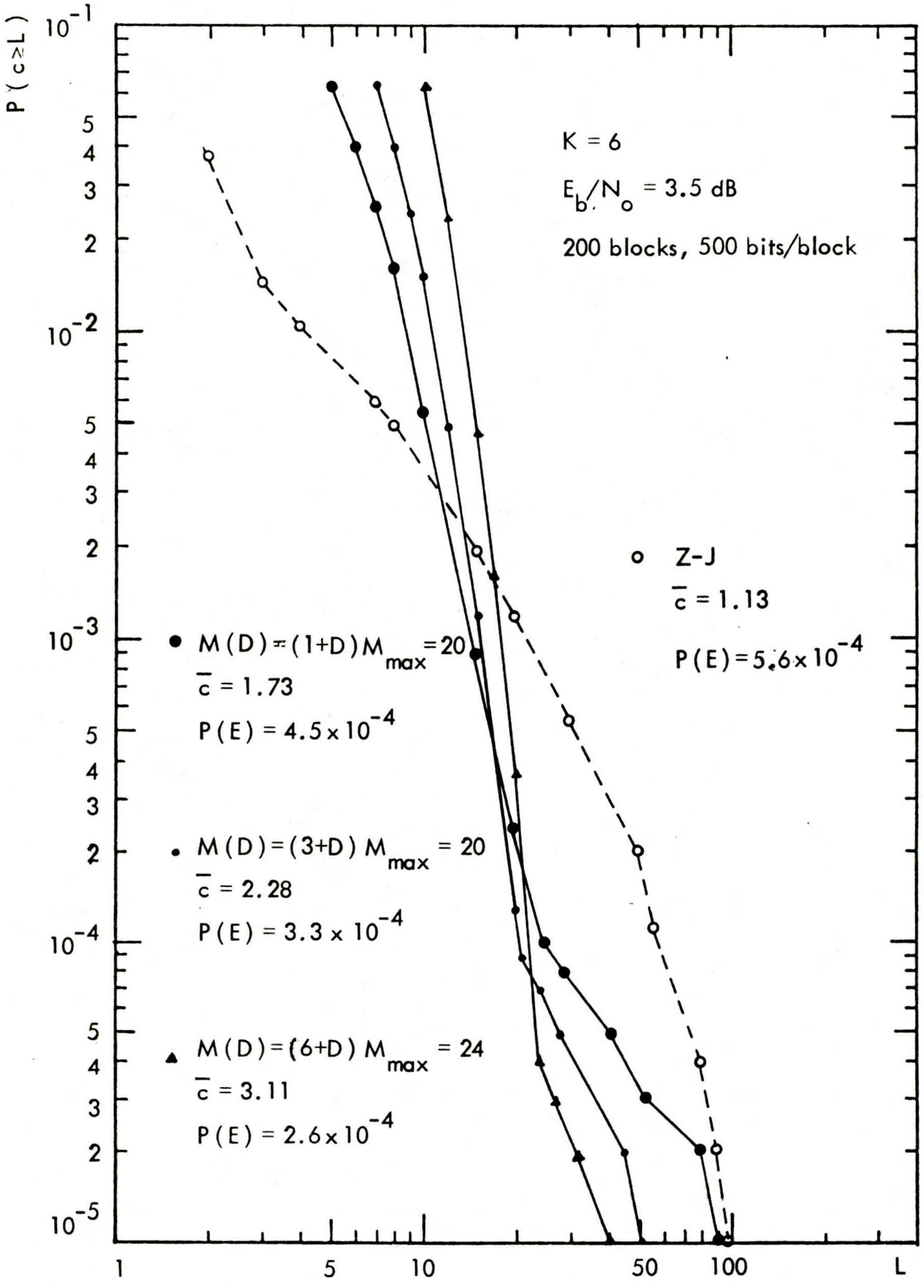

Figure (5) Empirical distribution of computations per search for the Adaptive algorithm.

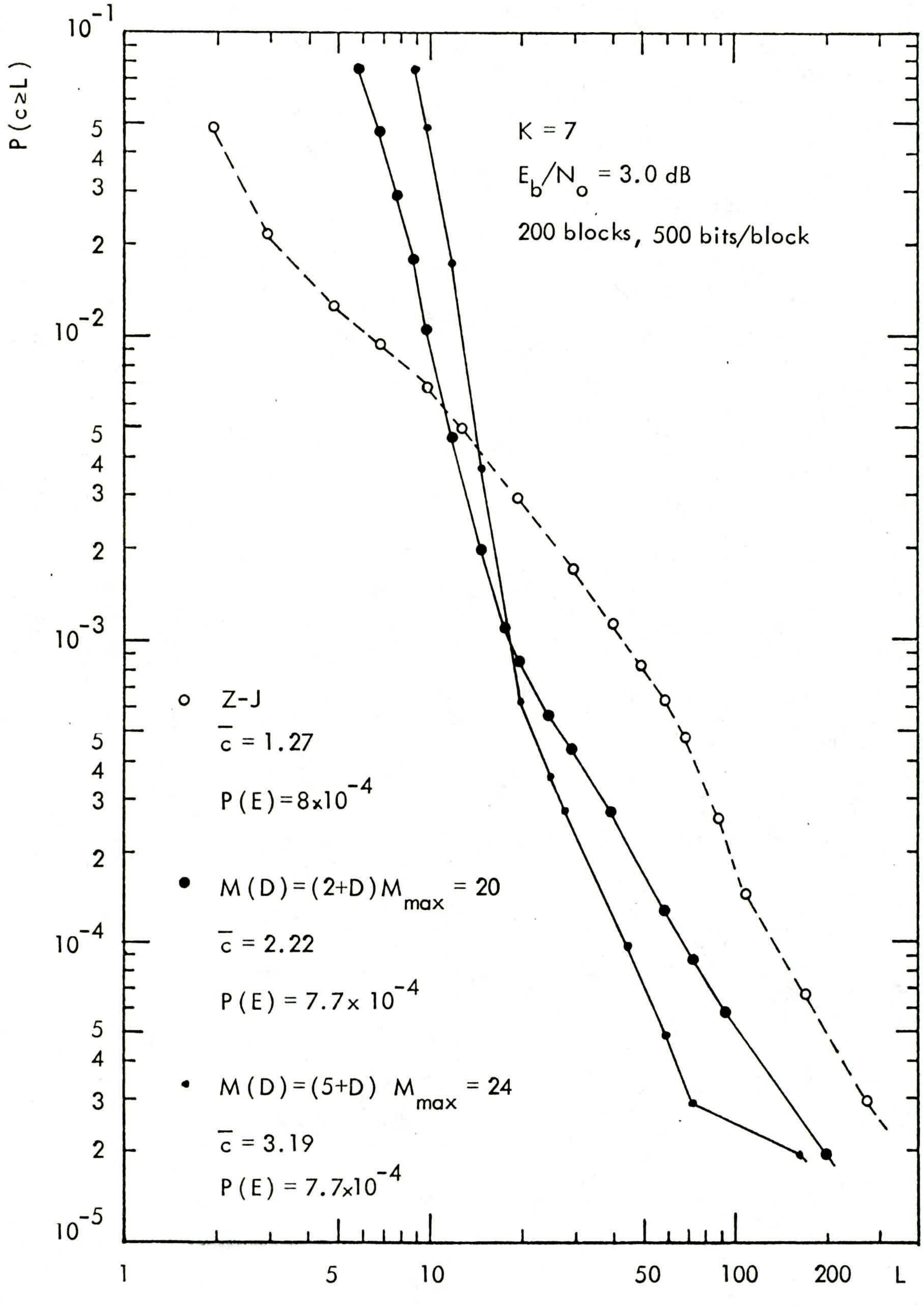

Figure (6) Empirical distribution of computations per search for the Adaptive Algorithm.

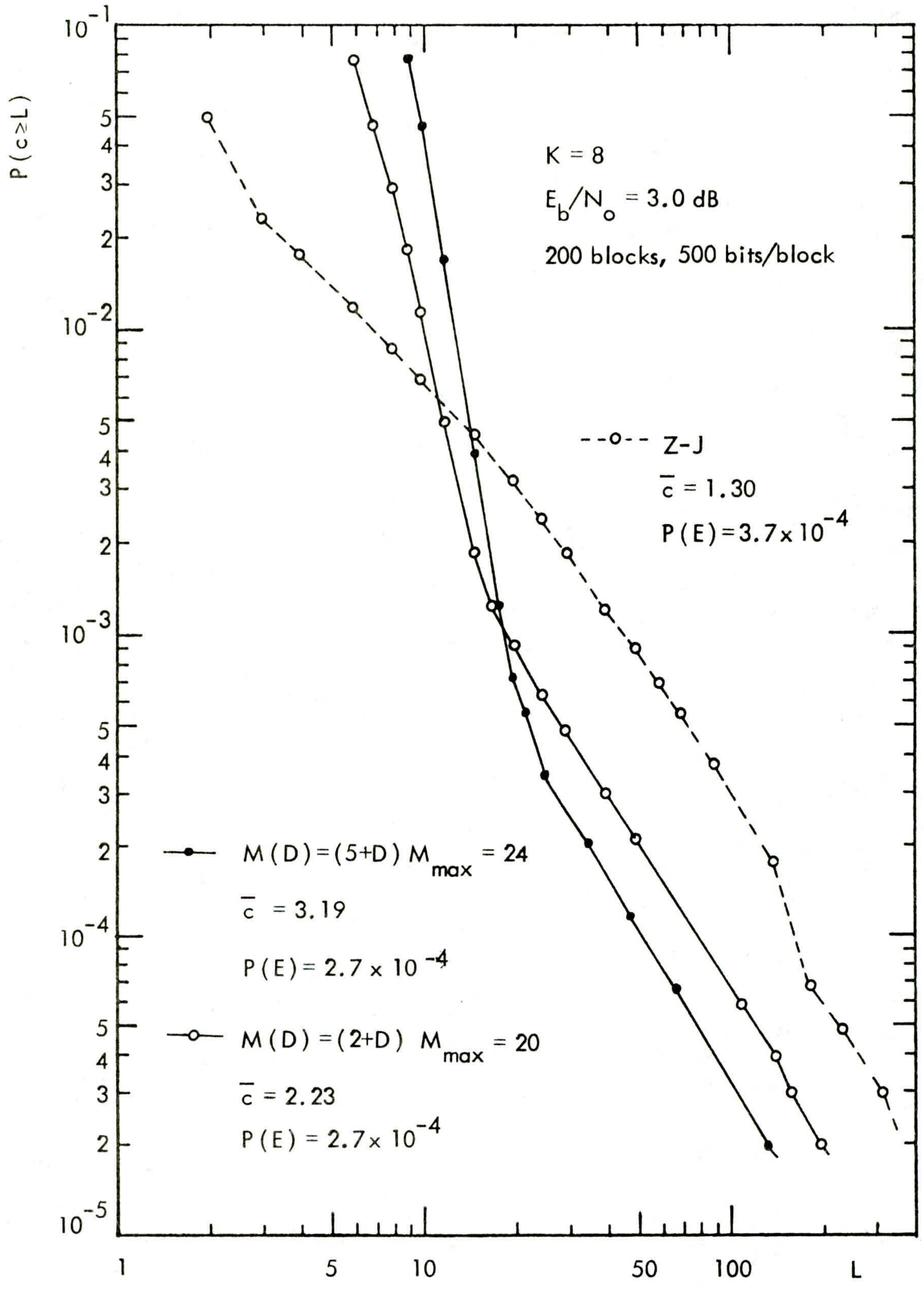

Figure (7) Emperical distribution of computation per search for the adaptive algorithm.

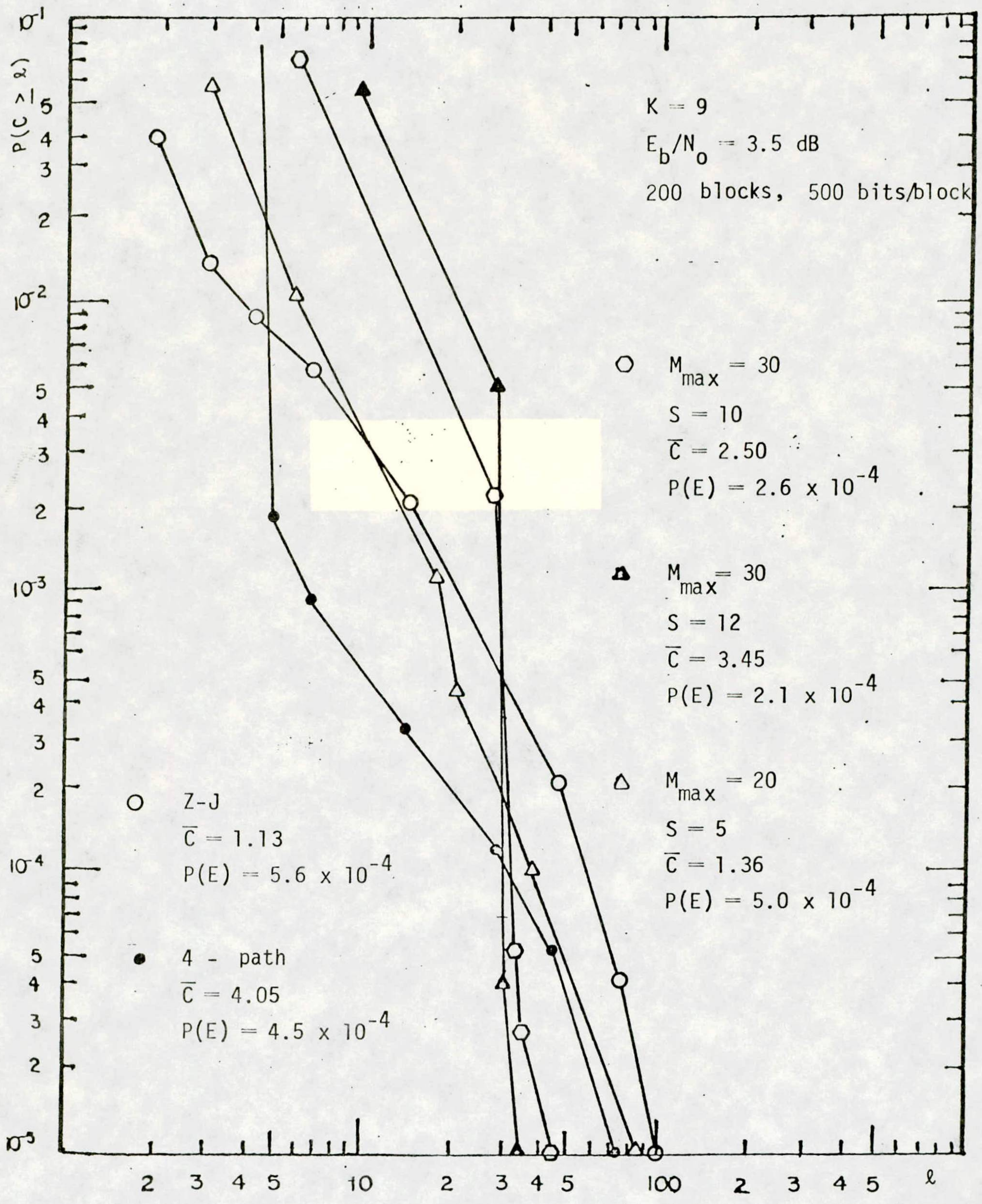

Figure (8) Empirical distribution of the number of computations per search. Substacks algorithm .