DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
SECTION COMMUNICATIONS

SIMULATION OF THE STACK ALGORITHM OVER CHANNELS WITH MEMORY : A COMPARISON WITH THE THEORY

Marie-José Montpetit
David HACCOUN, ing. Ph.D.

gratuit

EPM/RT-88/33

RAPPORT TECHNIQUE

EPM/RT-88/33

SIMULATION OF THE STACK ALGORITHM OVER CHANNELS WITH MEMORY : A COMPARISON WITH THE THEORY

Marie-José Montpetit
David Haccoun, Eng. Ph.D.
Department of Electrical Engineering
École Polytechnique de Montréal
Montréal (Québec) Canada

October 1988

Dépôt légal, 3e trimestre 1988
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.

SIMULATION OF THE STACK ALGORITHM OVER CHANNELS WITH MEMORY: A COMPARISON WITH THE THEORY*

by

Marie–José Montpetit and David Haccoun

Department of Electrical Engineering
Ecole Polytechnique de Montréal
P.O. Box 6079, Station "A"
Montréal, Québec, Canada
H3C 3A7

* This research has been supported in part by the Natural Science and Engineering Research Council of Canada.

SIMULATION OF THE STACK ALGORITHM OVER CHANNELS WITH MEMORY: A COMPARISON WITH THE THEORY

by

Marie–José Montpetit and David Haccoun

ABSTRACT

A bound on the average number of computations of the stack algorithm in a slowly varying channel has been found in by applying the theory of branching processes in random environments (BPRE). This bound must now be confronted with simulation results to assess its validity. This paper presents the results obtained when comparing simulation results of the average number of computations with the theoretical bounds when the conditions for the theory to remain valid throughout the simulation are verified.

SIMULATION DE L'ALGORITHME A PILE DANS UN CANAL A MEMOIRE: COMPARAISON AVEC LA THEORIE

par

Marie-José Montpetit et David Haccoun

RESUME

Une borne sur le nombre moyen de calculs de l'algorithme à pile dans un canal à mémoire peut être dérivée en utilisant la théorie des processus de ramification en environnement aléatoire (BPRE). Cette borne doit cependant être comparée aux résultats de simulation de l'algorithme à pile dans le même canal afin d'en démontrer la validité. Ce document présente la comparaison des résultats de simulation et des résultats théoriques quand tous les paramètres importants afin que la théorie reste valide au long de la simulation sont vérifiés.

TABLE OF CONTENTS

1 INTRODUCTION

One of the major drawbacks of sequential decoding is the variability of its decoding effort. Bursts of channel noise can cause the decoder to behave erratically and the decoding effort to grow exponentially. It is known that the number of computations necessary to decode one bit is a random variable with an asymptotic Pareto distribution. In a binary symmetrical channel (BSC), a simple branching process model has given closed–form bounds on the average number of computation C_{av} [2]. This approach provides better results than the traditional asymptotic analy–sis using Chernoff bounding over the ensemble of codes [3].

Because in a channel with memory the simple BSC model could not be applied, a new branching process model has been developed [1] in order to understand the behaviour of the stack algorithm of sequential decoding in a channel with memory. The channel used in this paper is a variation on the Gilbert–Elliot channel [4], [5] and it forms the random environment for which the theory of branching processes in random environments (BPRE) [6] has been used. The BPRE analysis gives new closed–form expressions for the bounds on C_{av} in channels with memory [1]. The average number of computations is formulated in terms of the branch metrics and their probability assignments, as used by the actual decoder. These theoretical results must now be compared to simulation results of the stack algorithm in a similar channel to assess their validity. But in comparing theoretical results to actual performances, the conditions for the theory to be applicable must be satisfied and the characteristics of the actual channel must correspond to the model.

This paper presents the results obtained when simulating the stack algorithm of sequential decoding in a slowly varying channel with memory under the same conditions as those used for the theoretical evaluation. In section 2, the channel model is introduced and the sequential decoding operations that are modeled by the branching processes in random environment (BPRE) are presented together with a review of the theoretical model. In section 3, the simula–tion of the stack algorithm with side–information on the channel state is presented. It is shown that under the same conditions, the theoretical results give a good bound on the actual performances of the algorithm.

2 INTRODUCTION TO BRANCHING PROCESSES AND SEQUENTIAL DECODING

In this section we define sequential decoding in a varying channel and its relationship to branching processes in random environment.

2.1 Sequential decoding in a varying channel

Sequential decoding by the stack algorithm is a metric–first, sub–optimal procedure for the decoding of convolutional codes [7]. The exploration of the code tree is performed one branch at a time using the integer–valued Fano branch metric. Over a path of length L the cumulative metric μL is:

$$\mu L = \sum_{i=1}^{L} \delta_i = \sum_{i=1}^{L} \sum_{j} \log_2 P(y_{ij}|x_{ij})/P(y_{ij}) - R \qquad (1)$$

Consider decoding over a channel with memory. This channel is a slowly varying channel [4],[5] having two distinct states or environments τ_0 and τ_1, binary symmetrical channels (BSC) with crossover probability ε_0 and $\varepsilon_1 >> \varepsilon_0$ respectively. Assuming a coding rate equal to 1/2, the only possible branch metric increments δ_i are $+a_0$, $-b_0$, and $-c_0$ in state τ_0 and $+a_1$, $-b_1$, and $-c_1$ in state τ_1. As shown in Figure 1, the system may be regarded as a two–state, τ_0 and τ_1, Markov chain with transition probability P and Q, P<Q. The markovian relationship between channel states satisfies the conditions for the theory of branching processes in random environments to be applicable [6].

In sequential decoding the decoder always extends the path with the highest metric among all the examined paths, only a fraction of the code tree is explored and the path reaching first the end of the tree is accepted as the decoded path. When the channel behaves normally, that is, in our model, stays in state τ_0, on the average, the total metric is increasing on the correct path. However, occasionally a burst of channel noise causes the metric to fall rapidly. The metric dip D_K is the difference between the metric μ_K of a node on the correct path at depth k and its smallest succeeding value (see Figure 2). When $D_K=0$, node k is called a "breakout node" and will be decoded by a single computation. When $D_K > 0$, node k is non–breakout and becomes the root of a tree of incorrect nodes, which will be explored by the decoder. As the size of the dip D_K increases the amount of computation required to decode one branch grows exponentially [8]. Due to the markovian relationship between the metric dips D_K this behaviour can be modeled by the BPRE model [1] [2] (see Figure 3) resulting in obtaining closed–form expressions for the average number of computation performed by the stack algorithm [1].

2.2 Theoretical results

To compute theoretical of C_{av} values a rate 1/2 convolutional code is used. Probability ε_0 of the BSC in channel state τ_0 is identical to the one in [2], $\varepsilon_0= 0.044$, and probability ε_1 is fixed successively to 0.65, 0.75 or 0.85. These probabilities give fairly dense bursts of noise [1]. The set of Fano metrics corresponding to ε_1 is the same as if ε_1 were equal to $(1-\varepsilon_1)$. In the equivalent BSC channel, given here for comparison purposes, the transition probability is equivalent to the total number of errors over all received bits, slightly above R_{comp} for all probabilities. All the metric sets are shown in Table 1. In order to limit the size of the branching process absorbing barriers are set at metric dip value, or height, 0 and H (see Figure 3). If H is high enough these absorbing barriers will not distort the results.

Once the channel probabilities and the absorbing barrier are known, the stationary probabilities π_i of the metric dip states can be computed. All computations were performed on a 386–based microcomputer with mathematical co–processor for more accuracy. An iterative algorithm, where solutions are found within an interval of confidence, is used. In our channel the determination of these stationary probabilities involves using some results form the theory of BPRE [1]. The stationary probability for state i, $\pi_i = P[D_K=i]$, $i \geq 0$, depends on the transition probabilities between states, the Fano metric sets and their probability assignments in each state [1] as seen in Table 1. With P=0.001 and Q=0.05, overall the amount of time spent in state τ_1 corresponds to less than 2% of the time. Using the algorithm developed for the computation of the stationary probabilities, over our time–varying channel the stationary probabilities of the metric dips should be the same as over a memoryless channel. Simulations with other values of P and Q show, as seen in Table 2 for ε_1=0.75, that if the channel is overall most of the time in state τ_i, i= 0 or 1, on the average, the stationary probabilities of the channel tend towards the stationary probabilities of state τ_i. As for the case when the channel is about 50% of the time in each channel, simulations have shown that the results are very unstable and lie between the two solutions for τ_0 and τ_1. As can be seen in Table 2, when the code rate is much larger than the R_{comp}. of the equivalent BSC channel, the distance between breakout nodes, the inverse of stationary probability π_0 and a measure of the efficiency of the decoding, becomes very large. In the actual decoder, this indicates that the average number of computations becomes unbounded, a typical situation when decoding in a very noisy channel: the number of incorrect nodes to be explored grows exponentially.

This last situation must be avoided to yield a bounded decoding effort. In terms of the branching process theory, the branching process initiated in the incorrect trees must become extinct with certainty. As we see below, we then get closed–form expressions for C_{av}.

Let us define matrix $B_k = N\,P_k$ of dimensions HxH. For a rate 1/2 random tree code where the branch symbols are chosen at random, N the number of extensions per node, is equal to 2. Let P_k be the matrix of the unconditional transition probabilities between metric states:

$$P_k = P_{\tau 0}\,p_k(\tau_0)+P_{\tau 1}\,p_k(\tau_1) \qquad (2)$$

where $P_{\tau 0}$ and $P_{\tau 1}$ are the transition probability matrices between metric states in channel state τ_0 and τ_1 respectively; they are also of dimensions HxH. Probability $p_k(\tau_i)$ is the probability of being in channel state τ_i at generation k. Here, with only two possible channel states τ_0 and τ_1 we can write [1]:

$$[p_k(\tau_0),p_k(\tau_1)]' = \underline{G}_0\, G^k \qquad (3)$$

where $\underline{G}_0$ is the 2x1 vector of initial channel states and G is the 2x2 transition matrix between channel states. The symbol " ' " denotes matrix transpose.

Let $\underline{C}_k(L)$ be the vector of the expected number of computations performed by the stack algorithm in extending up to length L all incorrect paths issued from a root node at depth k in environment e_k that is τ_0 or τ_1. $\underline{C}_k(1) = \underline{1}$ is the all–one vector.

Using a recursive relationship, the expected number of computations over a length L for the process initiated at depth k is shown to be [1]. [2]:

$$\underline{C}_k(L) < \underline{1} + B_k \underline{C}_{k+1}(L-1) \tag{4}$$

$L \geq 2$

Recursively we obtain:

$$\underline{C}_k(L) < \underline{1} + B_k \underline{1} + B_k B_{k+1} \underline{1} + \ldots + B_k \ldots B_{k+L-1} \underline{1} \tag{5}$$

where:

$$\underline{C}_k(L) = (C_{k1}(L), \ldots, C_{kH}(L))' \tag{6}$$

$L \geq 2$

and B_k was defined as $2P_k$.

$$\underline{C}_k(L) < (\mathfrak{I} + B_k + B_k B_{k+1} + \ldots + B_k \ldots B_{k+L-1}) \underline{1} \tag{7}$$

where $\mathfrak{I}$ is the identity matrix.

If on the average, all matrices B_k have eigenvalues smaller than 1, then the branching processes initiated by the incorrect trees eventually will be extinguished with certainty [1], [6]. Under this condition, the average number of computation by state converges to C_∞ as the length of the path goes to infinity. However, as all matrices B_k are different, since they they depend on the varying channel, we must use an approximation to the convergence matrix, matrix M, of dimensions HxH, with eigenvalues smaller than 1 but such as its elements are larger than the elements of B_k. With matrix M, C_∞ may be upper–bounded by [1]:

$$\underline{C}_\infty < (\mathfrak{I}-M)^{-1} \underline{1} \tag{8}$$

where $\mathfrak{I}$ is the HxH identity matrix and where $\underline{1}$ is the all–one vector of length H. Once the initial conditions and the channel transition probabilities matrices are known the derivation of M and $\underline{C}_\infty$ is straightforward.

Using $\underline{\pi}$, the vector of the stationary probabilities of the metric dips, then the average number of computations C_{av} is [1],[2]:

$C_{av} < \pi_0 + ((N-1)/N)\ \underline{\pi}\ .\ \underline{C}_\infty$ (9)

where $\underline{\pi} = (\pi_1,\pi_2...\pi_{H-1})'$ (10)

and " . " denotes a scalar product.

The bound on C_{av} given by (9) can clearly be improved by the addition of specific information on the code actually used [1]. We now extend the bound to semi–random 1, 2 and 3 codes. With a semi–random t code, the N extensions of the tth generation at depth k are deterministic and all the following generations are random. We expect the bound to become tighter and tighter as t increases.

For semi–random 1 codes, following the method described in [1] and [2], from generation k to generation k+1 we have:

$\underline{C}_k^{(i)}(1) < \underline{1} + P(1,\tau_i)\ \underline{C}_{k+1}(L-1),\quad L \geq 2,\ i= 0 \text{ or } 1$ (11)

the vector equation for the average number of computations over a length L for the process initiated at depth k, assuming a first extension in channel state τ_i. Matrix $P(1,\tau_i)$ is the transition probability matrix between metric states for the first branch extension in state τ_i, i=0,1 [1],[2]. It depends on the channel state. In the simulation, when $\varepsilon_1 \geq 0.5$ bits are inverted, so that $P(1,\tau_1)$ will be computed with $(1-\varepsilon_1)$. Beyond generation k+1, random coding is used, with $\underline{C}_{k+1}(L-1)$ expressed by (7).

We force convergence to obtain a closed–form expression of $\underline{C}_\infty^{(i)}(1)$ the average number of computations for a first extension in state i as L goes to infinity:

$\underline{C}_\infty^{(i)}(1) < \underline{1} + P(1,\tau_i)\ \underline{C}_\infty,\ i= 0 \text{ or } 1$ (12)

where (1) denotes a semi-random 1 code and $\underline{C}_\infty$ is the random coding expression given by (10).

Finally the average number of computations assuming a first extension in state i is [1],[2]:

$$\mathbf{C}_{av}^{(i)}(1) < \pi_0 + \underline{\pi}' \, \underline{\mathbf{C}}_\infty^{(i)}(1) \qquad (13)$$

For the bounds on semi-random 2 and 3 codes, additional assumptions are made to simplify the expressions which can get very complicated with two channel states. First, since the channel is slowly varying, we assume there is no channel transition on the first 3 extensions. This hypothesis is used for both semi-random 2 and semi-random 3 codes. Second, all incorrect paths of length 3 branches lie at the minimum Hamming distance from the correct path. This further simplifies the expressions for semi-random 3 codes.

The expressions for semi-random 2 codes are, following [1] and [2]:

$$\underline{\mathbf{C}}_\infty^{(i)}(2) < \underline{1} + P(1,\tau_i)\underline{1} + 2\, P(1,\tau_i)\, P(2,\tau_i)\, \underline{C}_\infty \qquad (14)$$

where i= 0 or 1 and where $\underline{C}_\infty$ is the random coding expression; all matrices depend on the channel state. Again to take into account the bit inversion, $P(2,\tau_1)$, the transition probability matrix between metric states for the second branch extension, will be computed with $(1-\varepsilon_1)$.

The average number of computations assuming a first extension in state i is [1]:

$$\mathbf{C}_{av}^{(i)}(2) < \pi_0 + \underline{\pi}' \, \underline{\mathbf{C}}_\infty^{(i)}(2) \qquad (15)$$

For semi-random 3, again following [1],[2]:

$$\underline{\mathbf{C}}_\infty^{(i)}(3) < \underline{1} + P(1,\tau_i)\underline{1} + 2\, P(1,\tau_i)\, P(2,\tau_i)\, \underline{1}$$
$$+ 4\, P(1,\tau_i)\, P(2,\tau_i)\, P(3,\tau_i)\, \underline{C}_\infty \qquad (16)$$

Matrices $P(1,\tau_i)$ and $P(2,\tau_i)$ have already been defined and matrix $P(3,\tau_i)$ is the transition probability matrix between metric states for the second and third branch extensions in state τ_i, i=0,1 [1], [2]. They all depend on the channel state. Finally, in each channel state we have [1]:

$$\mathbf{C}_{av}^{(i)}(3) < \pi_0 + \underline{\pi}' \, \underline{\mathbf{C}}_\infty^{(i)}(3) \qquad (17)$$

$\mathbf{C}_{av}(i)$ is a combination of $\mathbf{C}_{av}^{(0)}(i)$ and $\mathbf{C}_{av}^{(1)}(i)$, i= 1, 2, 3 for each semi–random codes, depending on the overall contribution of each channel state to the first incorrect extension. Considering that a metric dip is caused by channel errors, the overall probabilities p(0) of being in channel state τ_0, and p(1) to be in channel state τ_1 (p(1)= 1–p(0)) at the first branch extension, can be calculated using the stationary probabilities of the two channel states [1].

Consequently:

$$\mathbf{C}_{av}(i) < \mathbf{C}_{av}^{(0)}(i)p(0) + \mathbf{C}_{av}^{(1)}(i)p(1) \tag{18}$$

for i= 1, 2, 3.

With these results and the values obtained for $\underline{\pi}$, the bounds on C_{av}, the average number of computations, can be computed for ε_0= 0.04 and ε_1= 0.65, 0.75 and 0.85. They are listed below in Table 3 for random and semi–random coding 1, 2 and 3. To estimate the validity of these theoretical results they must be compared with actual performances. Before this discussion, the simulation of sequential decoding under the same channel and coding conditions is now considered.

3 SIMULATION RESULTS

We use a rate 1/2 convolutional code of constraint length of 24 [11]. The simulations are organized in blocks of 500 information bits and the number of blocks per simulation is 100. The transmission of the all–zero sequence is assumed. The basis of our simulation program is the stack algorithm [12] with stack sizes ranging from 5000 to 25000. All simulation parameters (initial metric, metric bin spacing etc.) have been taken from [12]. The simulations were performed on an IBM mainframe computer with statistics computed on the same 386–based micro–computer used for determining the stationary probabilities and the bounds on C_{av}.

The varying channel model as defined for the branching process analysis is also integrated throughout the simulations. The transitions in the channel model, P=0.001 and Q=0.05 are the same. Other channels, as found in [3], have also been investigated but will not be presented here. Our modified algorithm, which is described below, was verified for a single channel to assess if any modifications was hindering the decoding process. For $\varepsilon = 0.044$, the results were identical to those found in [2].

3.1 Side–information of the channel state

In the analysis, knowledge of the channel states and of matrices B_K, allows us to compute the bounds appropriately. In order to match these conditions in the simulation, some information about the channel state must extracted from the incoming symbols to be decoded. The information about the channel states sets the choice of branch metrics to be used.

The branch metrics corresponding to the incoming symbols are computed in a sub–program prior to the decoding itself. In this sub–program, errors in the incoming data are looked for and the channel state inferred accordingly. In the state τ_1, with $\varepsilon_1 \geq 0.5$, the bits are inverted. A computation is also made on the incoming sequence to decide if it is corresponding to an un–bounded branching process, also called supercritical [1]. These operations will now be examined.

3.2 Transfer between states

To infer the current channel state, the information present in the incoming symbols must be interpreted. The number T of error–free symbols necessary for transitions from state τ_1 to state τ_0 is determined as to minimize the probability of error on the state [13]. This means that the probability of receiving T consecutive error–free symbols and staying in the τ_1 state must be smaller than the probability of transferring to the τ_0 state, staying in that state $(T-1)$ error–free steps and transferring back to the τ_1 state. T is the minimum number satisfying the inequality:

$$(1-Q)^T(1-\varepsilon_1)^T < PQ\,(1-P)^{T-1}(1-\varepsilon_0)^{T-1} \tag{19}$$

Depending on the value of ε_1, 0.65, 0.75 and 0.85, T was found to be 9, 7 and 5 respectively.

For the transition from states τ_0 to τ_1, a similar computation was performed yielding:

$$(1-P)^T \varepsilon_0{}^T < PQ\,(1-Q)^{T-1} \varepsilon_1{}^{T-1} \qquad (20)$$

In our case, T was found to be 3 for all three probabilities.

The validity of T was verified by computer simulations. The criteria used were the agreement with channel probabilities as computed in the noise generator and the number of blocks decoded. This is shown in Table 4 for the transition τ_0 to τ_1 for $\varepsilon_1 = 0.75$. The agreement was best for the computed T values using (20). It is important to stress out that the value T=1, as used in the Gilbert channel [3], [13] was rejected as it is more realistic to assume that even in state τ_0 errors can occur.

3.3 Influence of the supercritical blocks

The conditions for matrix M of (8) to have eigenvalues smaller than 1 must be satisfied for the branching process initiated in the incorrect tree to become extinct. This means that if too many incoming bits were to come from state τ_1, then the process would become unbounded, or supercritical, and decoding would be impossible since the number of nodes in the incorrect trees becomes unbounded. The ratio of symbols from τ_0 to symbols from τ_1 that is acceptable is computed with the eigenvalues corresponding to the matrices $NP_{\tau 0}$ and $NP_{\tau 1}$, the HxH matrices of the average number of descendants per metric dip states in channel states τ_0 and τ_1. These matrices were introduced in (2) and with a rate 1/2 random code N=2. On the average the overall eigenvalue must be smaller than one [1],[6]. The computation was performed for all ε_1 probabilities and the ratio of symbols coming from state τ_0 to symbols coming from state τ_1 was found to be 10:1, 9:1 and 8:1 for 0.65, 0.75 and 0.85 respectively.

In the simulation, a large number of overflowing blocks even at a stack size of 25000 entries is associated to unbounded branching processes. Under these conditions, blocks would overflow even with an infinite stack. By taking these blocks out, the average number of computations drops and should agree more closely to the theoretical bounds. Table 5 shows the difference in the average number of computations when the supercritical blocks (SCB) are left in the decoding and when they are removed.

Although the elimination of these SCB can be seen as a bias in the simulation results, in the theoretical computations the supercritical cases are not considered since they give a useless infinite bound on the number of computations. By comparing the theoretical results of Table 3 to values in Table 5 we see that some of the bounds are high enough to accept even the supercritical cases. In the following, both simulation values will be considered.

3.4 Influence of the stack size on experimental results

In the algorithm, we must give a finite dimension to the stack, whereas in the analysis the stack is considered infinite. Stack overflow, the effects of a limited stack size, must be prevented in order to keep the events corresponding to the τ_1 state. Table 6 shows the effects of the stack size on the channel parameters P, Q, ε_0 and ε_1.

As the stack size increases, the average number of computations also increases since the algorithm stores more data in the stack, consequently allowing itself to do more searching in the code tree. The result is the higher number of computations per node. The channel parameters also vary with the stack size as more events corresponding to the real channel behaviour are kept. The best agreement between theoretical and experimental parameters is found when more blocks are decoded.

3.5 Discussion

By comparing simulation results of Table 5 and the theoretical bounds of Table 3, we can see that some values of the bound lie below the simulation results. This is because the true value of the upper bound is reached asymptotically for large values of the absorbing barrier H.

Although theoretical bounds have been computed for H larger than 100, it was found that the value at H=60 gave a good comparison value.

For semi–random 3 codes, the assumptions made on the behaviour of the channel introduce too much bias and the bounds do not agree well with the simulations. It is true that since T, the number of erroneous symbols necessary for the transfer from τ_0 to τ_1, is equal to 3, it is not realistic to hypothesize that no state transition takes place on the third branch. For this reason the theoretical bounds obtained for the semi–random 3 code cannot be validated by the simulation.

For semi–random 1 codes, the agreement between simulation results and the theoretical bounds is not the best. It is known [2] that the first branch extension is very significant on the computational behaviour. In our two–state channel, the fact of considering randomness at the second branch introduces a bias. If we consider a first extension in the τ_0 state, the random code at branch two is inflating results. However, if the first extension is in the τ_1 state, the random code is a lower estimate. For these reasons the results for semi–random 1 codes are not as good as those for semi–random 2 codes.

The semi–random 2 codes give the best agreement between theoretical bounds and simulation results. The simulation results confirm our hypothesis that over two branches there are few channel state transfers and the two first branch extensions are significant enough for the decoding not to be influenced by the randomness at branch extension 3. Considering the variations in the channel, this bound is very tight and confirms the usefulness of the analysis.

Finally, one can see that the overall random code is an upper bound on all results as predicted by the theory. It is worthwhile to note that while the simulation results were obtained from lengthy simulation runs on an IBM mainframe, the bounds were calculated in just a few minutes by mathematical packages available on the 386 micro–computer.

4 CONCLUSION

We have presented the simulation results of the stack algorithm over a slowly varying channel with memory. Our purpose was to compare our theoretical bound on the average number of computations of the stack algorithm in a slowly varying channel with simulation results and assess its validity. All simulation parameters were set to agree with the analysis and insure that the conditions of the theory remained valid throughout the simulation. Under these conditions, most of the simulation results agree with the theoretical bounds and the discrepancies can be explained within our model. The BPRE model therefore gives a new method for determining an upper bound on the average number of computations performed by the stack algorithm over a slowly time–varying channel without the need for lengthy simulations.

REFERENCES

[1] M.J. Montpetit, D. Haccoun and G. Deslauriers, "A Branching Process Analysis of the Stack Algorithm for Variable Channel Conditions." Submitted for Publication in IEEE Transactions on Information Theory.

[2] D. Haccoun, "A Branching Process Analysis of the Average Number of Computations of the Stack Algorithm", **IEEE Transactions on Information Theory**, Vol. IT–30, no. 3, pp. 497–508, May 1984.

[3] R.G. Gallager, **Information Theory and Reliable Communications.** Wiley, 1968.

[4] E.N. Gilbert, "Capacity of a Burst–noise Channel", **Bell System Technical Journal**, Vol. 39, pp. 1253–1266, Sept. 1960.

[5] E.O. Elliot, "Estimates of Error–rates for Codes on Burst–noise Channels", **Bell System Technical Journal**, Vol. 42, pp. 1977–1998, Sept. 1963.

[6] K.B. Athreya and S. Karlin, "On Branching Processes with Random Environments: I Extinction Probabilities", **The Annals of Mathematical Statistics**, Vol. 42, no. 5, pp. 1499–1520, 1971.

[7] V.K. Bhargava, D. Haccoun, R. Matyas and P. Nuspl, **Digital Communications by Satellite.** Wiley, 1981.

[8] J. Massey, J.M. Geist and M.K. Sain, "Certain Infinite Markov Chains and Sequential Decoding", **Discrete Mathematics**, Vol. 3, pp. 163–175, 1972.

[9] D. Haccoun, "A Markov Chain Analysis of the Sequential Decoding Metric", **IEEE Transactions on Information Theory**, Vol. IT–26, no. 1, pp. 109–113, Jan. 1980.

[10] R. Johannesson, "On the Distribution of Computation for Sequential Decoding using the Stack Algorithm", **IEEE Transactions on Information Theory**, Vol. IT–25, pp. 323–331, May 1979.

[11] R. Johannesson, "Robustly Optimal Rate One–Half Binary Convolutional Codes." **IEEE Transactions on Information Theory**, Vol. IT–21, pp. 464–468, July 1975.

[12] P.Y. Pau, "Logiciel de communications numériques." Master degree thesis, Ecole Polytechnique, 1985.

[13] K. Berglund, R. Johannesson and K.Sh. Zigangirov, " A New Sequential Decoding Algorithm for Fading Channels." **Proceedings of the 2nd Joint Swedish-Soviet International Workshop on Information Theory.** April 14–19, 1985, Sweden, pp. 212–216.

List of tables

ε_0	ε_1	ε_*	Fano metrics ε_0	Fano metrics ε_1	Fano metrics ε_*
0.044	0.65	0.052	[1 –4 –9]	[0 –1 –2]	[1 –3 –7]
0.044	0.75	0.054	[1 –4 –9]	[0 –1 –3]	[1 –3 –7]
0.044	0.85	0.056	[1 –4 –9]	[1 –2 –5]	[1 –3 –7]

ε_* is the error on the equivalent BSC.

Table 1: Metric sets.

P	Q	% time in state τ_1	$R>R_{comp^*}$	d_0
0	1	0	no	1.53
1	0	100	yes	unbounded
.001	.05	2	slightly	1.53
.4	.6	60	yes	11.19
.5	.5	50	yes	2290

R_{comp^*} is computed using the equivalent BSC error probability.

Table 2: Average distance between breakout nodes with $\varepsilon_0 = 0.04$ and $\varepsilon_1 = 0.75$.

	H	random	semi–random 1	2	3
$\varepsilon_1 = 0.65$	10	1.7312	1.3971	1.5926	1.9769
	20	3.6678	2.2003	2.6015	3.6848
	30	6.0519	3.1803	3.8309	5.7783
	40	8.7370	4.2728	5.1992	8.1019
	50	11.7969	5.5373	6.7875	10.8117
	60	15.2095	6.9534	8.5673	13.8498
$\varepsilon_1 = 0.75$	10	1.7184	1.3436	1.4752	1.8248
	20	3.5846	2.0472	2.3156	3.3261
	30	5.8021	2.8747	3.3037	5.1025
	40	8.2111	3.7630	4.3623	7.0018
	50	10.8624	4.7588	5.5533	9.1460
	60	13.7176	5.8367	6.8429	11.4676
$\varepsilon_1 = 0.85$	10	1.6899	1.1859	1.1862	1.3777
	20	3.4597	1.6395	1.6173	2.3932
	30	5.4941	2.1552	2.1082	3.5625
	40	7.6257	2.6868	2.6132	4.7677
	50	9.8860	3.2646	3.1634	6.0798
	60	12.2299	3.8681	3.7384	7.4481

Table 3: Theoretical results.

T, # errors for state transition	P	Q	ε_0	ε_1	% of decoded blocks
2	.002	.10	.04	.72	70
3	.001	.09	.04	.81	72
4	.000	.07	.04	.88	65

The stack size is 7500, the theoretical value of ε_1 is 0.75 and the computed value of T is 3.

Table 4: Channel probabilities vs T.

ε_1	stack size	average number of computations with SCB	without SCB
0.65	5000	3.27	2.89
0.65	10000	4.87	4.09
0.65	15000	6.29	5.15
0.65	20000	7.77	6.23
0.75	5000	2.87	2.49
0.75	10000	4.05	3.30
0.75	15000	5.06	3.92
0.75	20000	6.07	4.52
0.85	5000	2.00	1.72
0.85	10000	2.48	1.85
0.85	15000	2.88	1.95
0.85	20000	3.23	2.05

Table 5: Influence of the supercritical blocks.

ε_1 th.	Average number of computations w/o SCB	with SCB	Stack Size	P	Q	ε_0	ε_1	decoded blocks (%)
.65	2.89	3.27	5000	.0006	.101	.043	.727	69
	3.49	4.07	7500	.0006	.094	.043	.727	71
	4.09	4.87	10000	.0006	.094	.043	.727	71
	4.61	5.57	12500	.0006	.089	.043	.715	73
	5.15	6.29	15000	.0006	.089	.043	.715	73
	6.23	7.77	20000	.0006	.089	.043	.715	73
	7.30	9.00	25000	.0006	.089	.043	.715	73
.75	2.49	2.87	5000	.0007	.087	.043	.800	75
	2.94	3.47	7500	.0007	.087	.043	.800	75
	3.30	4.05	10000	.0008	.080	.043	.800	78
	3.62	4.57	12500	.0008	.080	.044	.802	80
	3.92	5.06	15000	.0008	.080	.044	.800	81
	4.52	6.07	20000	.0008	.079	.044	.800	81
	5.10	7.04	25000	.0008	.079	.044	.791	82
.85	1.72	2.00	5000	.0009	.067	.044	.870	90
	1.79	2.25	7500	.0009	.065	.044	.870	91
	1.85	2.48	10000	.0009	.065	.044	.870	91
	1.90	2.69	12500	.0009	.065	.044	.870	91
	1.95	2.88	15000	.0009	.065	.044	.870	91
	2.05	3.23	20000	.0009	.065	.044	.870	91
	2.16	3.57	25000	.0009	.065	.044	.870	91

Table 6: Influence of the stack size on the average number of computations.

List of figures

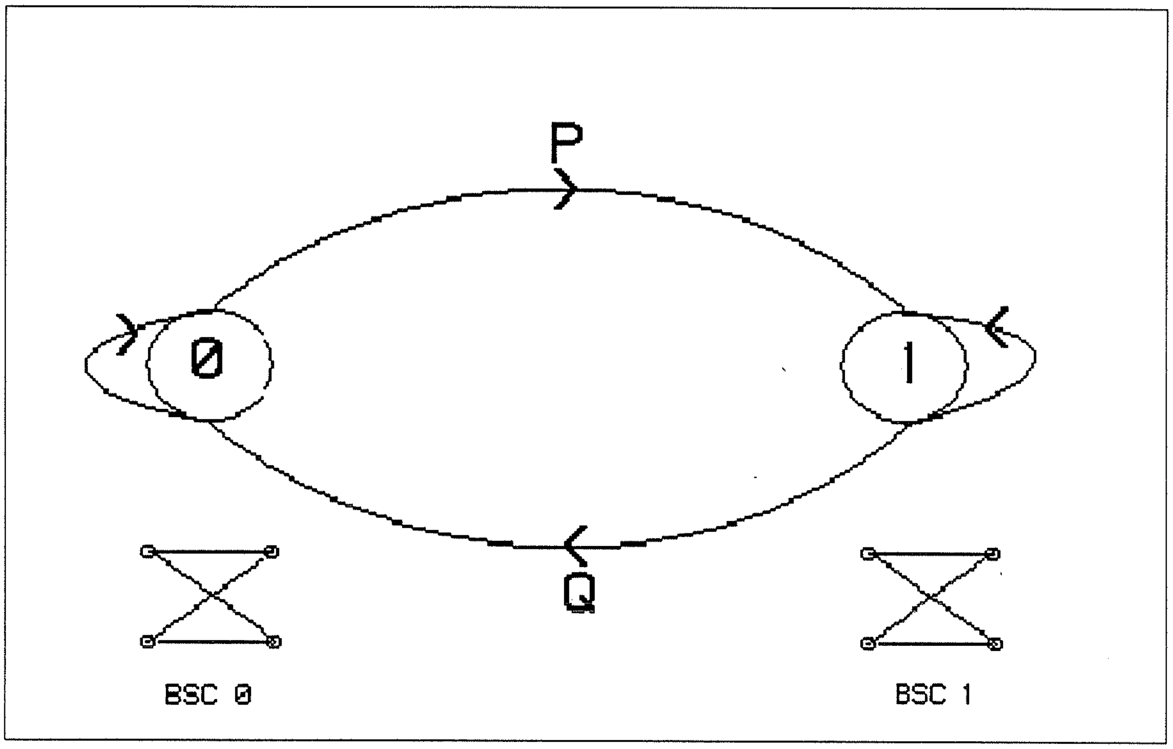

Figure 1: Model of the channel with memory.

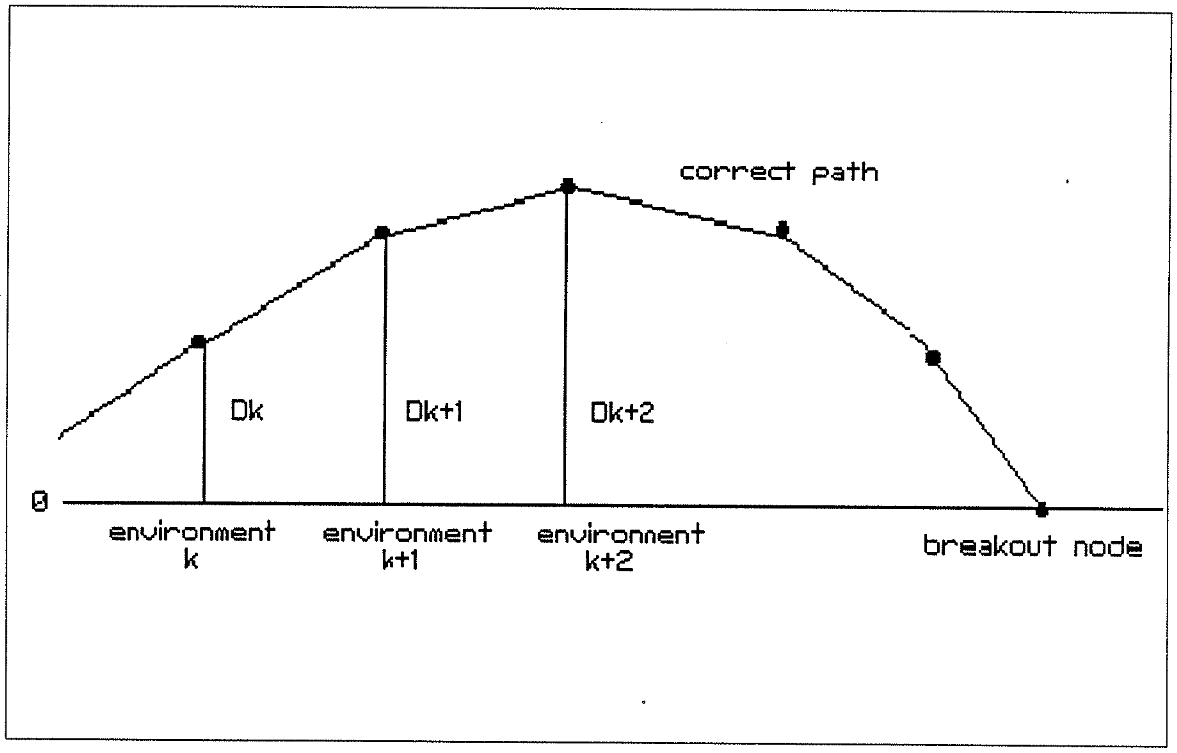

Figure 2: Metric dips on the correct path.

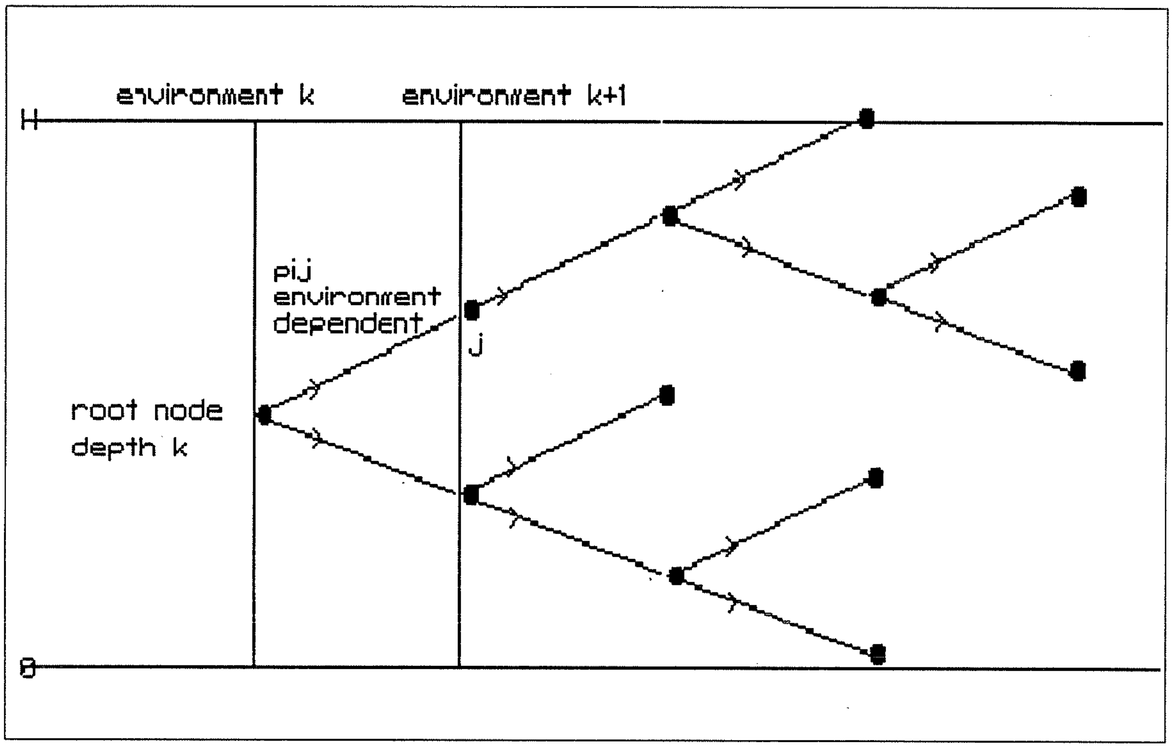

Figure 3: Branching processes in random environments.