On the distribution of the number of computations in any finite number of subtrees for the stack algorithm

Johannesson, Rolf; Zigangirov, Kamil

Published in:
IEEE Transactions on Information Theory

1985

Link to publication

Citation for published version (APA):

Total number of authors:
2

General rights
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
On the Distribution of the Number of Computations in Any Finite Number of Subtrees for the Stack Algorithm

ROLF JOHANNESSON AND KAMIL SH. ZIGANGIROV

Abstract—Multitype branching processes have been employed to determine the stack algorithm computational distribution for one subtree. These results are extended here to the distribution of the number of computations in any finite number of subtrees. Starting from the computational distribution for \( K - 1 \) subsequent subtrees, a recurrent equation for the distribution for \( K \) subsequent subtrees is determined.

I. INTRODUCTION

It is well known [1]–[3] that the curse of sequential decoding is that its behavior is limited by a computational distribution (conditioned on correct decoding), which is asymptotically Pareto. Until now little attention has been paid to the problem of determining the computational distribution for a small number of computations. In our previously published papers [4], [5] we employed multitype branching processes to determine the stack algorithm computational distribution for the first incorrect subtree. The results in [5] were obtained for random tree codes, while in [4] we considered the class of binary, rate 1/2, complementary + random tree codes, a fictitious entity that is a reasonable model for constant convolutional codes that have column distance \( d_c = 2 \).

In a complementary + random tree code the channel input symbols on the transmitted path are all zeros, and the channel input symbols on the incorrect branches stemming from nodes on the correct path are all ones. For all other branches, each channel symbol is chosen independently and according to a specified probability distribution. In this work we extend the results in [4] and determine the probability distribution for the number of computations in any finite number of subsequent subtrees.

II. PRELIMINARIES

Let \( C_n, n = 1, 2, 3, \ldots \), denote the number of computations made by the sequential decoder in order to decode the \( n \)th correct node. In Fig. 1 we show a partially explored code tree with \( C_1 = 4 \). For a tree of unbounded length the random variables \( C_1, C_2, C_3, \ldots \) have the same distribution, but they are certainly not independent.

Let \( M_n \) be the cumulative metric for the first \( n \) branches of the correct path; i.e.,

\[
M_n = \sum_{k=1}^{n} z_k,
\]

where \( z_k \) is the branch metric for the \( k \)th branch. Let \( D_n \), \( n = 0, 1, 2, \ldots \), be the difference between the cumulative metric and the smallest succeeding value; i.e.,

\[
D_n = M_n - M_{\text{min},n},
\]

where

\[
M_{\text{min},n} = \min \{ M_k, M_{k+1}, \ldots \}.
\]

The nonnegative random variables \( D_n \) all have the same well-known distribution [4], [6] since the metrics for the correct path stemming from the \( n \)th node have the same statistical character for all \( n \).

Finally, let \( C_N^n \) be the total number of computations in the \( n \)th, \((n+1)\)th, \( \ldots \), \( N \)th subtree

\[
C_N^n = \sum_{k=n}^{N} C_k.
\]

III. RECURRENT EQUATION FOR DETERMINING THE DISTRIBUTION OF \( C_N^n \)

Let \( F_N^n(r, i, k) \) be the conditional distribution for the random variable \( C_N^n \) on the condition that \( D_n = i \) and \( z_n = k \); i.e.,

\[
F_N^n(r, i, k) = P\{ C_N^n = r | D_n = i, z_n = k \}.
\]

By dividing the number of computations in the \( n \), \( n + 1 \), \( \ldots \), \( N \)th subtrees into two parts, \( C_n \) and \( C_{n+1} \), we have

\[
F_N^n(r, i, k) = \sum_{s=1}^{r-1} P\{ C_n = s | D_n = i, z_n = k \} \cdot P\{ C_{n+1} = r - s | D_n = i, z_n = k \}.
\]

The conditional probability distribution for \( C_{n+1} \) can be expanded as

\[
P\{ C_{n+1} = r - s | D_n = i, z_n = k \} = \sum_{j=0}^{\infty} \sum_{l=0}^{j} \cdot P\{ D_{n+1} = j, z_{n+1} = l | D_n = i, z_n = k \}.
\]

Fig. 1. Part of a complementary + random code tree partially explored by a sequential decoder.
where the variable $i$ is summed over all branch metric values.

The random variable $C_N$ conditioned on $D_n+1$ and $z_{n+1}$ is independent of $D_n$ and $z_n$. Furthermore, the random variables $D_{n+1}$ and $z_{n+1}$ conditioned on $D_n$ are independent of $z_n$. Hence we obtain the recurrent equation for $F_N(r, i, k)$

$$F_N(r, i, k) = \sum_{s, j} f(s, i, k) \cdot F_N(r - s, j, l) g(j, l, i),$$

where $n < N$,

$$f(s, i, k) = F'(s, i, k)$$

and

$$g(j, l, i) = P[D_{n+1} = j, z_{n+1} = l | D_n = i].$$

The conditional probability distribution $g(j, l, i)$ is easily calculated as follows:

$$g(j, l, i) = P[D_{n+1} = j, z_{n+1} = l | D_n = i] = P[D_n = i] P[z_{n+1} = l | D_n = i],$$

where we have used the fact that the random variables $D_{n+1}$ and $z_{n+1}$ are statistically independent. Let us assume that we have the branch metric set $\{+1, -4, -9\}$, cf. [4]; then

$$P[D_n = i | D_{n+1} = j, z_{n+1} = l] = \begin{cases} 1, & i = j = 0, l = 1 \\ 0, & i = j = 1, l = 0 \\ \frac{1}{2}, & i = j = 0, l = 0. \end{cases}$$

Starting from the conditional probability distribution $f(s, i, k)$, which can be determined by the methods given in [4], we can use (8) and recursively obtain the conditional probability distribution $F_N(r, i, k)$ for any $n, 0 < n < N$. Finally, we calculate the probability distribution for the number of computations in $K \geq 1$ subtrees as

$$P[C^k = r] = \sum_{i, k} F_k(r, i, k) h(i, k),$$

where

$$h(i, k) = P[D_n = i, z_n = k] = P[D_n = i] P[z_n = k].$$

### IV. DISCUSSION OF NUMERICAL RESULTS AND SIMULATIONS

The problem of resolving ties among the cumulative metrics is discussed in detail in [4]. When there was no obvious way to resolve ties the pessimistic policy "in case of ties extend an incorrect node first" was used to obtain an upper bound on the distribution function for $C_l$, and the optimistic policy "in case of ties extend the correct node first" was used to obtain a lower bound. Because of the close agreement between the simulated $C_l$-curves and the corresponding lower bounds in [4], we will use these lower bounds as our conditional distribution $f(s, i, k)$ in (8).

By combining the branching processes methods described in [4] and the recurrent equation (8), we evaluate theoretical lower bounds on the distribution function of the random variable $C_l$, $P[C_l \geq r]$ for the case where transmission takes place over a binary symmetric channel (BSC) with crossover probability $p = 0.045$. Since we are only considering rate $R = 1/2$ codes, this crossover probability corresponds to transmission at the computational cutoff rate $R_c$. The bounds are evaluated for the Fano
A New Class of Check-Digit Methods for Arbitrary Number Systems

H. PETER GUMM

Abstract—For arbitrary number systems we present a new check-digit method that detects all single-digit errors and all transpositions of adjacent digits using a single check digit from the given number system. In previous methods at least one type of transpositional error had to remain undetected. The key to this method lies in using the dihedral groups together with appropriate transformations in the important cases, where the numbers are represented in base $2^r$ with $r$ odd.

I. INTRODUCTION

Empirical studies have shown [9], [10] that the most common typing errors that occur when data are entered on a keyboard are

- single-digit errors (one digit wrong)
- format errors (one digit inserted or left out)
- transpositions (interchanging of two adjacent digits).

To detect such errors, the original string of data is supplied with one check digit, where digit now means a "digit" in the chosen number system. Each method is able to detect all "single-error mistakes," but they fail to detect all transposition errors. At least one (and often not more than one) erroneous transposition is undetectable with those methods, see [1]-[3].

The format errors principally cannot be detected by only one check digit, and every method that detects single-digit errors will automatically detect about 90 percent of all format errors. The use of two check digits as proposed in [3] and [11] is not advised since the previously mentioned studies have also shown that the absolute number of errors that occur roughly doubles when the number of digits increases by two. Thus generally specifying and checking for a fixed format seems appropriate to detect format errors. We therefore concentrate on methods to detect single-digit errors and transposition errors.

Clearly, for numbers to the base $2$ such a check digit method is impossible since the numbers $00, 01, 10$ would have to be impossible since the numbers $00, 01, 10$ would have to be supplied.

II. DESCRIPTION OF THE CHECK-DIGIT METHOD

A new class of check-digit methods for arbitrary number systems has been developed that has the following basic properties:

1. It is possible to represent any number in base $2^r$ with a different check digit.
2. The check digit is computed using a different formula for each number system.
3. The check digit method is based on the dihedral groups and can be applied to any number system.
4. It is possible to detect all single-digit errors and all transposition errors.
5. The method is more general and simpler than previous methods.

REFERENCES


Manuscript received August 26, 1983; revised January 16, 1984.
The author is with the Gesellschaft für Strahlen- und Umweltforschung Munich-Neuherberg, Institut für Medizinische Informatik und Systemforschung, D-8042 Oberschleißheim, Federal Republic of Germany.

1After this paper was written, I was informed that T. Vorbeck in his thesis "Error correcting decimal codes," (Mathematical Centre Tracts, no. 29, Amsterdam: Mathematisch Centrum, 1969) has presented a different check-digit method. Our method, nevertheless, appears to be more general and simpler from a conceptual as well as an implementational standpoint.