Comments on 'An efficient algorithm for computing free distance' by Bahl, L., et al.

Larsen, Knud J.

Published in:
I E E E Transactions on Information Theory

Publication date:
1973

Document Version
Publisher's PDF, also known as Version of record

We have in (17) a nonlinear difference equation with two integer-valued independent variables v and l. The boundary conditions are given by

$$P_v(l) = \begin{cases} 0, & v = 0, \quad l \geq 1 \\ 1, & v \geq 0, \quad -n_0 \leq l \leq 0. \end{cases}$$

The boundary conditions are statements of the facts that a zero-depth tree has zero as the weight of its minimum-weight path, and that $d_{\min}$ is always nonnegative. The difference equation has order one in v and order n_0 in l.

Equation (17) is nonlinear and we are unable to find a closed-form solution $P_v(l)$. However, since the independent variables are integer valued, numerical solutions are easily obtained with a digital computer. From (14a), values of v and l satisfying $l/(vn_0) = D$ for selected rational values of D gives $P_v(l) = \Pr\{\rho_{vn_0}(X \mid B) \geq D\}$.

Examples for the binary alphabet ($q = 2$) and code rates $R = \frac{1}{2}$ and $\frac{1}{3}$ are shown in Figs. 3 and 4. The negative of $\log P_v(l)$ is plotted on a log scale as a function of tree depth v with parameter $D = l/(vn_0)$. $\Pr\{\rho_{vn_0}(X \mid B) \geq D\}$ approaches zero with tree depth v for $D \geq D^*$, where D^* satisfies $R = R(D^*)$, and approaches one for $D < D^*$ for the selected values of D shown. Noting the nearly linear asymptotic behavior of $\Pr\{\rho_{vn_0}(X \mid B) \geq D\}$ on a double log scale for fixed D, we conjecture that tree codes also have the double exponential convergence behavior shown in the preceding for block codes.

The ensemble average distortion $\bar{\rho}_{vn_0}$ for tree codes of depth v can be obtained from the solution for $P_v(l)$

$$\begin{aligned} \bar{\rho}_{vn_0} &= \frac{1}{vn_0} \overline{d_{\min}(v)} \\ &= \frac{1}{vn_0} \sum_{l=0}^{\infty} l[P_v(l) - P_v(l+1)] \\ &= \frac{1}{vn_0} \left\{ \sum_{l=0}^{\infty} lP_v(l) - \sum_{l=0}^{\infty} (l+1)P_v(l+1) + \sum_{l=0}^{\infty} P_v(l+1) \right\} \\ \bar{\rho}_{vn_0} &= \frac{1}{vn_0} \sum_{l=1}^{\infty} P_v(l). \end{aligned} \tag{19}$$

Equation (19) is plotted in Fig. 5 for the binary alphabet ($q = 2$) and code rates $R = \frac{1}{2}$ and $\frac{1}{3}$ bit/symbol as a function of tree depth v.

The difference equation (17) was derived for a symmetric source with restriction to the error-probability distortion measure. We can generalize to balanced distortion measures provided the entries $[d_0, d_1, \cdots, d_{q-1}]$ are rational. In this case we define

$$P_v(l) = \Pr\{d_{\min}(v) \geq lc\}$$

$$P_\Delta(k) = \Pr\{\Delta = kc\}$$

where

$$c = \text{LCD}\,[d_0, d_1, \cdots, d_{q-1}]$$

and Δ/n_0 is the branch distortion random variable. Then (16) with the upper limit of summation appropriately changed is still valid. However $P_\Delta(k)$ must be recomputed depending on $[d_0, d_1, \cdots, d_{q-1}]$.

For an arbitrary balanced distortion measure the variable l in (16) must be allowed to take on a continuum of values, implying an integral difference equation. Since the merits of a pure difference equation from the computational standpoint is lost, we will not consider this case.

Concerning previous work related to achievable distortion with tree codes, Jelinek and Anderson [4] give ensemble average distortion for some classes of tree codes, and distortion for certain fixed good codes. Their results were based on a computer simulation for a suboptimum source encoding algorithm. Jelinek [5] has established the existence of tree codes arbitrarily close to the rate-distortion bound.

References

[1] T. Berger, *Rate Distortion Theory: A Mathematical Basis for Data Compression.* Englewood Cliffs, N.J.: Prentice-Hall, 1971, probl. 2. 12.

[2] R. M. Fano, *Transmission of Information.* New York: Wiley, 1961, ch. 8.

[3] T. J. Goblick, Jr., "Coding for a discrete information source with a distortion measure," Ph.D. dissertation, Dep. Elec. Eng., Mass. Inst. Technol., Cambridge, Mass.

[4] F. Jelinek and J. B. Anderson, "Instrumentable tree encoding of information sources," *IEEE Trans. Inform. Theory*, vol. IT-17, pp. 118–119, Jan. 1971.

[5] F. Jelinek, "Tree encoding of memoryless time discrete sources with a fidelity criterion," *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 584–590, Sept. 1969.

Comments on "An Efficient Algorithm for Computing Free Distance"

KNUD J. LARSEN

***Abstract*—In the above paper,[1] Bahl *et al.* described a bidirectional search algorithm for computing the free distance of convolutional codes. There are some flaws in that algorithm. This correspondence contains a corrected version of the algorithm together with a proof that the corrected version always computes the free distance for noncatastrophic codes.**

Bahl *et al.*[1] have described a very efficient bidirectional search algorithm for computing the free distance of convolutional codes. However, this algorithm had some flaws, which will be described at the end of this paper. A corrected version follows here, together with a proof that ensures that the new algorithm computes the true free distance for noncatastrophic codes.[2]

Like the original algorithm, the new one is based on the state transition diagram. When a state is reached by a path it is stored in an array together with information on the type of the path (forward or backward) and the weight of the path. If there are many paths to a state, then the lowest weight is stored. Thus the information to be stored is as follows.

- S Terminal state of path.
- W Minimum Hamming weight of paths to S known at the moment.
- T Type of state S. The type consists of two parts, $T1$ and $T2$. $T1$ indicates whether the state S is D(ead) or non-D and $T2$ is the type, F(orward) or B(ackward), of the minimum-weight path first found to S. Thus we have four possible types: F, B, DF, and DB.

Let W^* denote the current upper bound on d_{free}. Then the algorithm for a rate-$1/n$ code is the following.

Manuscript received August 31, 1972; revised February 5, 1973.

The author is with the Laboratory for Communication Theory, Technical University of Denmark, Lyngby, Denmark.

[1] L. R. Bahl, C. D. Cullum, W. D. Frazer, and F. Jelinek, *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 437–439, May 1972.

[2] J. L. Massey and M. K. Sain, "Inverses of linear sequential machines," *IEEE Trans. Comput.*, vol. C-17, pp. 330–337, Apr. 1968.

1) Set W^* = an upper bound on d_{free}. Store $S = (100\cdots0)$, $W = \sum_{j=1}^{n} g_0^{(j)}$, $T = F$ and $S = (00\cdots01)$, $W = \sum_{j=1}^{n} g_{\nu-1}^{(j)}$, $T = B$.

2) Search through the storage array for the lowest weight non-D state S_m. It has weight W_m and is of type T_m.

3) If $2W_m \geq W^*$ or all states are D, go to 17.

4) Set $E = 0$. Determine terminal state (S) and weight (W) of the 0-extension of S_m. Go to 6.

5) Set $E = 1$. Determine terminal state (S) and weight (W) of the 1-extension of S_m.

6) If $W > (W^* + n - 1)/2$, go to 15.

7) Check through the array for $S_k = S$. If such an S_k is found, go to 9.

8) Find an unoccupied location in the array and store S, W, T_m. Go to 15.

9) The type and weight of S_k are T_k and W_k. If $T2_k = T2_m$ go to 13.

10) Set $W^* = \min(W^*, W + W_k)$.

11) If $W \geq W_k$, go to 15.

12) Set $W_k = W$ and $T2_k = T2_m$. Go to 15.

13) If $T1_k = D$, go to 15.

14) Set $W_k = \min(W_k, W)$.

15) If $E = 0$, go to 5.

16) Set $T1_m = D$. Go to 2.

17) $d_{free} = W^*$. STOP.

A flow chart of the algorithm is shown in Fig. 1.

We now prove the following theorem.

Theorem: When the algorithm stops, the computed W^* is the free distance of the code if the code is noncatastrophic.

The theorem is proved via a number of lemmas. First an obvious property of the algorithm is presented in the following.

Lemma 1: When a state with weight W_m is extended, all dead states ($T1 = D$) have weights less than or equal to W_m.

Lemma 2: When a state S is made dead, DF (or DB), its weight W_m is the weight of the minimum-weight F-(B-)path to S, and the minimum-weight B-(F-)path to S has weight greater than or equal to W_m.

Proof: This is clearly true for $W_m = 0$. Assume that it is true for all states with weights less than W_m. If S could have lower weight than W_m, its predecessor belonging to this lower weight path would have weight less than W_m. But from Lemma 1 we know that all such states are already extended, and this has not reduced the weight of S. W_m is thus the lowest weight. The lemma now follows by induction.

Lemma 3: States with weight $W > (W^* + n - 1)/2$ can never belong to a path with total weight less than W^*.

Proof: A state S with weight W is the terminal state of an extension from a state S_1 with weight at least $W - n$, and its extension may reach a state S_2 of opposite type ($T2$-part) and weight at least $W - n$. S_2 cannot have weight less than $W - n$, since then it would have been extended earlier (Lemma 1). Hence S would have been reached already with a weight less than W, causing the case considered here not to occur. The path through S_1, S, and S_2 has weight at least $2W - n$, and $W > (W^* + n - 1)/2$ implies $2W - n > W^* - 1$ or $2W - n \geq W^*$.

Proof of the theorem: When the algorithm stops, all states with weights less than or equal to $[(W^* - 1)/2]$[3] are dead. Now two types of paths through the state transition diagram exist: a) paths with only dead states and b) paths with some nondead states.

[3] $[x]$ denotes the integer part of x.

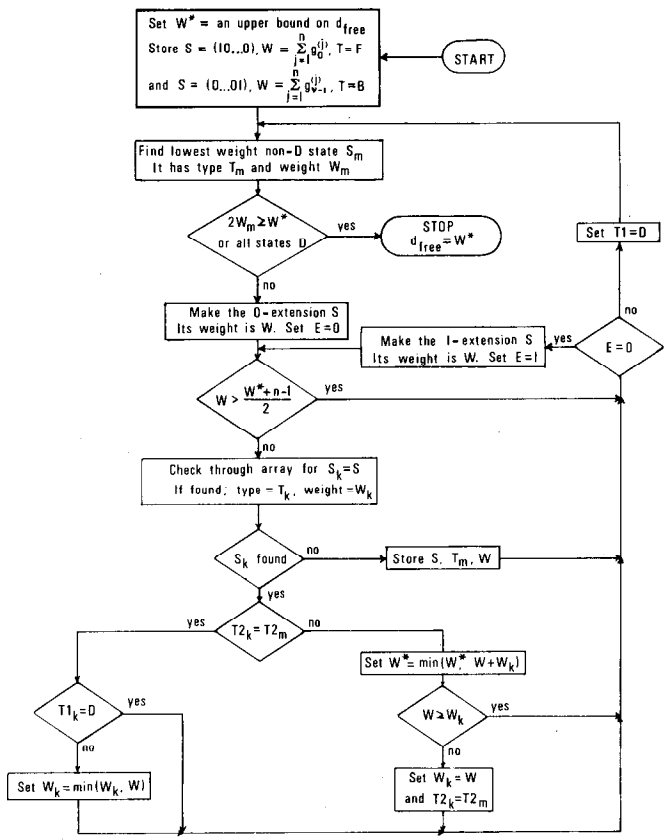

Fig. 1. Flow chart of corrected algorithm.

In the first case, there must be some place in the path where two adjacent states are of different $T2$-types. From Lemma 2 we know that the weights of these two states are absolute minimum weights for paths of any type to the states. The algorithm computes the total weight of this path at least once (algorithm step 10)) and the minimum weight of paths formed in this way is thus greater than or equal to the final W^*. In the second case it suffices to consider the case where only one nondead state occurs in the path. This follows because the weights on the branches are nonnegative. The weight of the minimum-weight F-path to the nondead state is not less than $[(W^* - 1)/2] + 1$, since otherwise the state would have been extended as an F-state. Similarly, the weight of the minimum-weight B-path to this state is not less than $[(W^* - 1)/2] + 1$. The total weights of such paths are thus greater than or equal to $[(W^* - 1)/2] + 1 + [(W^* - 1)/2] + 1 \geq W^*$. Lemma 3 ensures us that we have not left out states that might be part of paths with weight less than the final W^*. Thus this W^* must actually be the weight of the minimum-weight path from state $(00\cdots0)$ back to state $(00\cdots0)$ (apart from the trivial $(00\cdots0) \rightarrow (00\cdots0)$ with weight 0). If the code is noncatastrophic,[2] this is the free distance of the code. (If the code is catastrophic, it is not the real free distance, but it might still be useful if the information sequences were known always to end with $\nu - 1$ zeros.) The theorem is now proved.

The new algorithm differs from the original one in three respects.

1) It differs in that there are two types of dead states. When the $T2$-type of dead state is opposite to the type of the extension reaching the state, the upper bound is checked (algorithm step 10)). This was not done in the original version, but is necessary as is shown in the example of Fig. 2. The indexes of S indicate the

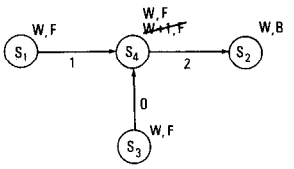

Fig. 2. Example showing that upper bound should be checked when dead state is met. The numbers on branches are weights.

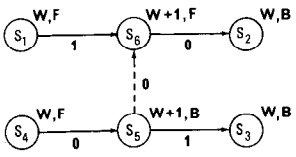

Fig. 3. Example showing forgotten path.

time sequence of extension. When S_4 is extended against S_2 (which is dead), the upper bound W^* should be reduced from $2W + 3$ to $2W + 2$ if not already so due to paths outside this part of the diagram.

2) There is now a change of the $T2$-type if the state receives a path of opposite $T2$-type having weight less than the first path to the state (algorithm steps 12), 13)). Originally this was not done but, as the example in Fig. 3 shows, this is necessary. The path S_4,S_5,S_6,S_2 having weight $2W$ would never be found if the $T2$-types of S_6 and S_5 were not changed when S_2 and S_4, respectively, were extended.

3) The stop-condition (step 3)) includes the possibility of all states being dead before $2W_m \geq W^*$. The necessity of this can be shown by a fairly simple code.

Tests have shown that the changes introduced here affect the efficiency of the algorithm very little.

ACKNOWLEDGMENT

The author is grateful to Prof. J. L. Massey, University of Notre Dame, Notre Dame, Ind., for his encouragement to find a proof that the corrected algorithm finds the true free distance. Dr. E. Paaske of the Laboratory for Communication Theory is acknowledged for many helpful suggestions concerning the final version of the algorithm and especially for noting that the original stop-condition was insufficient.

A Double-Phased-Burst-Error-Correcting Code of Rate 1/2

DAVID M. MANDELBAUM

***Abstract*—A binary double-phased-burst-error-correcting block code of rate 1/2 has been discovered by computer search. This code is quasi-cyclic, has a total burst-length-to-redundancy asymptotic ratio of 2/5, and is quite easily decodable. It can correct two phased-burst errors, each of length two digits or less. To correct larger nonphased bursts, interleaving is required.**

A computer search was made of quasi-cyclic codes [8] of rate 1/2 having subblock length $n_0 = 2$. A (20,10) quasi-cyclic code was found that will correct any two subblocks in error. A codeword will consist of 10 subblocks of 2 digits each

$$x_0p_0x_1p_1x_2p_2\cdots x_9p_9$$

where x_i is an information digit and p_i is a parity check digit. The corresponding parity-check matrix is composed of 10 submatrices

$$H = [H_0 \mid H_1 \mid H_2 \mid \cdots \mid H_8 \mid H_9]$$

$$= \begin{bmatrix} 10 & 01 & 00 & & 00 & 00 \\ 00 & 10 & 01 & & 00 & 00 \\ 00 & 00 & 10 & & 01 & 00 \\ 00 & 00 & 00 & & 00 & 01 \\ 01 & 00 & 00 & \cdots & 01 & 00 \\ 00 & 01 & 00 & & 01 & 01 \\ 01 & 00 & 01 & & 00 & 01 \\ 01 & 01 & 00 & & 01 & 00 \\ 00 & 01 & 01 & & 10 & 01 \\ 01 & 00 & 01 & & 00 & 10 \end{bmatrix}. \tag{1}$$

Each H_i is a cyclic shift downward of H_{i-1}. It was verified by the computer program that this code will correct errors in any two subblocks; that is, it can correct two phased-burst errors of up to two digits each.

The quasi-cyclic property of this code will allow for a relatively simple decoding scheme which is quite similar to Massey's decoding method [1] for the single-burst-correcting Berlekamp–Preparata–Massey convolutional codes [2], [3]. A somewhat similar H matrix for single-burst-error-correcting block codes has been used by Srinivasan [5]. Assume that the zero subblock, x_0p_0, and also another subblock, x_ip_i, $(0 < i \leq 9)$, are in error. Then the resulting syndrome S is in the vector space formed by the union of the column spaces of H_0 and H_i. S is a column vector, $S = He$, where e is the column vector representing the errors. Therefore, detecting whether an error is caused by bursts in subblocks 0 and i is equivalent to determining whether S is in the vector space defined by the columns of the 10×4 matrix H_0H_i.

Associated with the matrix H_0H_i is a 10×6 matrix $G_{0,i}$ such that S is a vector in H_0H_i if and only if

$$G_{0,i}{}^TS = \mathbf{0} \tag{2}$$

where the superscript T indicates the transpose. The column space of $G_{0,i}$ is the null space of H_0H_i and $(H_0H_i)^TG_{0,i} = 0$. See [6] for a procedure to determine $G_{0,i}$. Each of the 6 rows of $G_{0,i}{}^T$ gives a parity check that must be satisfied by every syndrome associated with errors in subblocks 0 and i. The number of inputs to each such check is a maximum of 10.

There are 10 matrices $G_{0,i}$ for $1 \leq i \leq 9$ which will detect burst errors in subblock zero and any other subblock. Now suppose a two-digit error occurs in subblock 0 and a two-digit error occurs in subblock i. Call the resulting syndrome $S(x_0,p_0,x_i,p_i)$. Now consider a two-digit error in subblock one and a two-digit error in subblock $i + 1$. This will yield a syndrome $S(x_1,p_1,x_{i+1},p_{i+1})$ which is equal to $S(x_0,p_0,x_i,p_i)$ shifted cyclically one place downward. If R is an operator representing a cyclic shift of one position downward, then

$$S(x_1,p_1,x_{i+1},p_{i+1}) = RS(x_0,p_0,x_i,p_i).$$

The obvious reason is that

$$H_1 = RH_0 \text{ and } H_{i+1} = RH_i.$$

Then also

$$S(x_2,p_2,x_{i+2},p_{i+2}) = R^2S(x_0,p_0,x_i,p_i).$$

Manuscript received August 25, 1972; revised January 29, 1973.
The author is at P.O. Box 645, Eatontown, N.J. 07724.