

Effects of floating point arithmetic in enumerative coding

Kees A. Schouhamer Immink and Augustus J.E.M. Janssen *

Summary - *The storage requirements of conventional enumerative schemes can be reduced by using floating point arithmetical operations instead of the conventional fixed point operations. The enumeration scheme will incur coding loss. The relationship between the storage requirements and the coding loss is derived.*

1 Introduction

In channel coding schemes we are usually faced with the problem of translating a given source word into another codeword and *vice versa* that satisfies some prescribed constraints. In the absence of an algorithmic rule defining the relationship between the source word and codeword, the translation operation will be simple look-up tables. As hardware grows with the number of codewords used, i.e. exponentially with the codeword length, there is a technological limit to the length of the words that can be translated using such a simple look-up table.

A preferable alternative technique, called *enumerative coding*, makes it possible to perform the translation by invoking an algorithmic procedure [1]. Essentially, enumerative decoding is accomplished by forming the weighted sum of the codeword received. The integer-valued weights used in forming the sum are a function of the channel constraints in force. Encoding is done by a method which is similar to decimal-to-binary conversion where, instead of the usual powers of two, the weights are used. The weights are usually pre-computed and stored in a look-up table. Translation using enumeration has the virtue that the complexity (weight storage) grows polynomially with the codeword length. This relationship between complexity and word length can be approximated by, say, n^t , $n \gg 1$, where n denotes the word length and t is an arbitrary positive integer. Note that the fixed-point binary representation of each weight requires Rn , $0 < R < 1$, bits.

In an alternative scheme where a floating point instead of a fixed point representation of the weights is employed, the binary representation of each weight requires e_p bits, where e_p is a fixed integer chosen by the designer. As a result, the complexity grows with n^{t-1} , which could be beneficial for large n . As a penalty, we will incur some coding loss because the full set of constrained codewords cannot be used (see later for more details).

The article addresses the problem of quantitatively assessing the coding loss in the case of (dk) constrained sequences. We start with a description of the state of the art

*K.A. Schouhamer Immink and A.J.E.M. Janssen are with Philips Research Laboratories, Prof. Holstlaan 6, 5656 AA Eindhoven, The Netherlands.

followed by an exposition of the floating point enumeration scheme. Next, we develop a relationship between the coding loss and the weight storage hardware.

2 Coding and decoding using enumeration

The description of the enumeration method given below is due to Cover [1].

Let $\{0, 1\}^n$ denote the set of binary sequences of length n and let S be a given (constrained) subset of $\{0, 1\}^n$. The set S can be ordered lexicographically as follows: if $\mathbf{x} = (x_1, \dots, x_n) \in S$ and $\mathbf{y} = (y_1, \dots, y_n) \in S$, then \mathbf{y} is called less than \mathbf{x} , in short, $\mathbf{y} < \mathbf{x}$, if there exists an i , $1 \leq i \leq n$, such that $y_i < x_i$ and $x_j = y_j$, $1 \leq j < i$. The position of \mathbf{x} in the lexicographical ordering of S is defined to be the *rank* of \mathbf{x} , denoted by $i_S(\mathbf{x})$.

Let $n_s(x_1, x_2, \dots, x_u)$ be the number of elements in S for which the first u coordinates are (x_1, x_2, \dots, x_u) . The inverse rank of $\mathbf{x} \in S$, i.e. the number of elements of S that have a lexicographic index *higher* than \mathbf{x} , is given by

$$i_S^c(\mathbf{x}) = \sum_{j=1}^n \bar{x}_j n_s(x_1, x_2, \dots, x_{j-1}, 1), \quad (1)$$

where $\bar{x}_j = 1 - x_j$, the complement of x_j . Next, we will apply the above theory to the enumeration of (dkr) sequences.

2.1 Encoding and decoding of (dkr) sequences

A dk -limited sequence, (dk) sequence in short, has at least d and at most k 'zeros' between consecutive 'ones' [2]. A (dkr) sequence is a (dk) sequence with at most r trailing 'zeros'.

Given a dkr -codeword \mathbf{x} , let $\mathbf{x}_j^1 = (x_1, x_2, \dots, x_{j-1}, 1)$. Denote by $N^0(i)$ the number of (dkr) -constrained sequences of length i starting with a 1. We define the quantity $a_j(\mathbf{x})$ as the length of the trailing zero-run of the sub-vector (x_1, \dots, x_{j-1}) if it is not the all-zero sequence. Or,

$$a_j(\mathbf{x}) = \begin{cases} \min\{(j - i - 1) : 1 \leq i < j, x_i = 1\} & \text{if } j > 1 \text{ and} \\ & (x_1, \dots, x_{j-1}) \neq (0, \dots, 0); \\ d & \text{otherwise.} \end{cases}$$

By invoking (1) to (dkr) sequences, we observe that for a given j , $1 \leq j \leq n$, $n_s(\mathbf{x}_j^1)$ can take one of two different values, depending on \mathbf{x} . If $a_j(\mathbf{x}) < d$ then $n_s(\mathbf{x}_j^1) = 0$, because in this case \mathbf{x}_j^1 violates the prescribed d constraint. Alternatively, if $a_j(\mathbf{x}) \geq d$ then $n_s(\mathbf{x}_j^1) = N^0(n - j + 1)$. The next relationship for the enumeration of (dkr) sequences is now immediate [3] [4] [5].

$$i_S^c(\mathbf{x}) = \sum_{j=1}^n \delta_j(\mathbf{x}) N^0(n - j + 1),$$

where

$$\delta_j(\mathbf{x}) = \begin{cases} 1 & \text{if } x_j = 0 \text{ and } a_j(\mathbf{x}) \geq d; \\ 0 & \text{otherwise.} \end{cases}$$

In words, the inverse rank of \mathbf{x} can be obtained by summing the appropriate weight of each 'zero' of \mathbf{x} that is either contained in the leading zero-run or is preceded by at least d 'zeros'.

The computation of the weights $N^0(n)$ is an exercise in combinatorics. To that end, let $U(n)$ be the number of (dk) sequences that start and end with a '1'. Then the following recursions are immediate:

$$U(n) = \begin{cases} 0, & \text{if } n \leq 0, \\ 1, & \text{if } n = 1, \\ \sum_{i=d}^k U(n-i-1), & \text{if } n \geq 2. \end{cases} \quad (2)$$

The number of (dkr) -constrained sequences of length n , $N^0(n)$, is simply

$$N^0(n) = \sum_{i=0}^r U(n-i). \quad (3)$$

For $n > d+k$ we may verify the following recursion:

$$N^0(n) = \sum_{i=d+1}^{k+1} N^0(n-i). \quad (4)$$

2.2 Enumeration using floating-point arithmetic

The hardware for implementing the enumeration algorithm comprises a (binary) adder, a subtracter, a comparator, and a look-up table of the pre-computed set of weights $\{N^0(i)\}$, $1 \leq i \leq n$. The binary fixed-point representation of a single weight requires Rn bits per weight, where R , $0 < R < 1$, is a constant. For the overall scheme, we need therefore storage proportional to Rn^2 . In the sequel we will develop an enumeration method where the weights are specified in floating-point notation.

Let I be one of the weights $N^0(i)$. It is well known that the non-negative integer I , $I < 2^v$, can be uniquely represented by a binary v -tuple $\mathbf{x} = (x_{v-1}, \dots, x_0)$, where

$$I = \sum_{i=0}^{v-1} x_i 2^i.$$

The binary v -tuple \mathbf{x} is called the binary fixed-point representation of I . Let

$$u = \lfloor \log_2 I \rfloor$$

be the position of the leading 'one' element of \mathbf{x} . Then the p -bit truncation of I , denoted by $\lfloor I \rfloor_p$,

$$\lfloor I \rfloor_p = \lfloor 2^{-(u+1-p)} I \rfloor 2^{u+1-p}, \quad (5)$$

can be represented in binary floating-point representation whose mantissa requires at most p non-zero bits. If the above finite-precision arithmetic is used in the enumeration algorithms we must modify the set of weights developed in the previous section. To that end, let $\hat{N}^0(i)$ denote the number of (dkr) sequences of length i starting with a '1' that can be encoded with p -bit mantissa representation, then

$$\hat{N}^0(i) = \begin{cases} N^0(i), & i \leq k+d, \\ \lfloor \sum_{j=d+1}^{k+1} \hat{N}^0(i-j) \rfloor_p, & i > k+d. \end{cases} \quad (6)$$

For clerical convenience, it is assumed that the weights $N^0(i) < 2^p$, $1 < i \leq k + d$, i.e. they can be represented by a mantissa of p bits; otherwise more bookkeeping is required. The encoding and decoding algorithms developed in the previous section can be employed directly by using the 'truncated' coefficients $\hat{N}^0(i)$ *in lieu* of $N^0(i)$. The enumeration algorithm itself remains unchanged. The effect on the set of codewords will be that recursively the $N^0(i) - \lfloor N^0(i) \rfloor_p$ highest ranking (dkr) words of length i are discarded from the set of all lexicographically ordered dkr sequences starting with a '1'.

2.3 Effects of finite-precision arithmetic

Using finite precision of the weights representation (truncation) will result in coding loss.

The number of (dkr) sequences of a length n that start with a '1', $N^0(n)$, $n > k + d$, satisfies (see (4))

$$N^0(n) = \sum_{i=d+1}^{k+1} N^0(n-i).$$

The number of (dkr) sequences grows exponentially with n , the growth factor being $\lambda_0 = 2^{C(d,k)}$, where $\lambda = \lambda_0$ is the largest root of the characteristic equation [2]

$$\lambda^{k+2} - \lambda^{k+1} - \lambda^{k-d+1} + 1 = 0 \quad (7)$$

Recall that for sufficiently large n , the number of (dkr) sequences of length n , $\hat{N}^0(n)$, that can be encoded using a p -bit mantissa representation can be expressed as

$$\hat{N}^0(n) = \lfloor \sum_{i=d+1}^{k+1} \hat{N}^0(n-i) \rfloor_p. \quad (8)$$

Conceptually, the computation of the growth factor $\hat{\lambda}$ of $\hat{N}^0(n)$ and the capacity $\hat{C}(d, k) = \log_2 \hat{\lambda}$ becomes very simple if we notice that the sequence behavior of the p -bit mantissa of $\hat{N}^0(n)$ versus n can be described in terms of an autonomous finite-state machine. The dynamic behavior of the finite-state machine is implied by the recursive equation (8).

Proposition 1 *The capacity $\hat{C}(d, k)$ is rational.*

Proof: From the theory of feedback registers [6] we know that the sequence of the mantissa of $\hat{N}^0(n)$ will ultimately become (and remain) *periodic*. That is, there are integers h and f such that

$$\hat{N}^0(n)2^h = \hat{N}^0(n+f). \quad (9)$$

In other words, per cycle period of length f the number of sequences increases with a fixed factor, which is equal to a power of two, 2^h . From the above it is immediate that

$$\hat{C}(d, k) = \frac{h}{f}. \quad (10)$$

Results of a computer search are listed in Table 1.

Table 1: Capacity $\hat{C}(d, k)$ for selected values of d , k , and p .

d	k	$C(d, k)$	p	$\hat{C}(d, k)$
1	7	0.67929	8	40/59
1	12	0.69299	9	9/13
2	12	0.54711	7	6/11
2	15	0.55011	11	11/20

The above method for computing the capacity $\hat{C}(d, k)$ does not provide a useful relationship between p and the capacity loss $C(d, k) - \hat{C}(d, k)$. Below we try to obtain a relatively simple relationship without the need of a computer search.

The capacity $C(d, k)$ can be expressed as $C(d, k) = \log_2 \lambda_0$, where λ_0 is the largest root of the characteristic equation (7) corresponding to the linear recursion (4), i.e. $\lambda = \lambda_0$ satisfies

$$f(\lambda) = \frac{\lambda^{k+2} - \lambda^{k+1}}{\lambda^{k-d+1} - 1} = 1. \quad (11)$$

Due to the $\lfloor \cdot \rfloor_p$ operation, the recursion (8) is non-linear, whence it is not straightforward how $\hat{C}(d, k)$ can be expressed as $\log_2 \hat{\lambda}$ with $\hat{\lambda}$ the largest root of a characteristic equation. We propose here to linearize the recursion (8) by replacing it by

$$\hat{N}^0(n) = (1 - \sigma_p) \left(\sum_{i=d+1}^{k+1} \hat{N}^0(n-i) \right), \quad (12)$$

where σ_p is the limiting average relative truncation error,

$$\sigma_p = \lim_{N \rightarrow \infty} E \left[\frac{\underline{x}(N) - \lfloor \underline{x}(N) \rfloor_p}{\underline{x}(N)} \right]$$

with $\underline{x}(N)$ an integer random variable, uniformly distributed on $\{2^{N-1}, \dots, 2^N - 1\}$. Accordingly, we approximate $\hat{C}(d, k)$ by $\log_2 \hat{\lambda}(\sigma_p)$, where $\hat{\lambda}(\sigma_p)$ is the largest root of the characteristic equation corresponding to (12) so that $\lambda = \hat{\lambda}(\sigma_p)$ satisfies

$$f(\lambda) = \frac{\lambda^{k+2} - \lambda^{k+1}}{\lambda^{k-d+1} - 1} = 1 - \sigma_p.$$

Note that in particular $\hat{\lambda}(0) = \lambda_0$ (no truncation, see (11)). Then we get after linearizing f around λ_0

$$\hat{\lambda}(\sigma_p) \approx \lambda_0 - \frac{\sigma_p}{f'(\lambda_0)} = (1 - \sigma_p \mu) \lambda_0,$$

where

$$\mu^{-1} = d + \frac{\lambda^{k+2} - (k+1-d)}{\lambda^{k+1-d} - 1}.$$

By elementary means it can be shown that

$$2^{-p} \ln 2 - 2^{-(2p+2)} \leq \sigma_p \leq 2^{-p} \ln 2.$$

The capacity $\hat{C}(d, k)$ can therefore be approximated by

$$\hat{C}(d, k) = \log_2 \hat{\lambda} \approx \log_2 \lambda_0 - \frac{\sigma_p \mu}{\ln 2} \approx C(d, k) - \mu 2^{-p},$$

where we have replaced σ_p by $2^{-p} \ln 2$.

Practically, we have $\mu \approx 0.25$ in the working range $d = 1, \dots, 3, k \gg d$, which means that the loss in capacity resulting from the truncation of the weights can be simply approximated by

$$C(d, k) - \hat{C}(d, k) \approx 2^{-(p+2)}. \quad (13)$$

A comparison of the above result with the outcome of the method of computing the cycle time revealed that (13) can serve as a reliable rule of thumb.

3 Conclusions

We have introduced a scheme of enumerative coding of dk sequences using floating-point arithmetic. This scheme offers, for long codewords, a significant advantage in storage requirements. A simple relationship between coding efficiency versus encoding or decoding hardware (storage) has been derived.

References

- [1] T.M. Cover, 'Enumerative Source Coding', *IEEE Trans. Inform. Theory*, vol. IT-19, no. 1, pp. 73-77, Jan. 1973.
- [2] K.A.S. Immink, *Coding Techniques for Digital Recorders*, Prentice-Hall International (UK) Ltd., Englewood Cliffs, New Jersey, 1991.
- [3] L. Patrovics and K.A.S. Immink, 'Encoding of $dklr$ -sequences using one weight set', *IEEE Trans. Inform. Theory*, vol. IT-42, no. 5, pp. 1553-1554, Sept. 1996.
- [4] Tj. Tjalkens, 'Runlength limited sequences', *IEEE Trans. Inform. Theory*, vol. IT-40, no. 3, pp. 934-940, 1994.
- [5] H. Hollmann, *Modulation Codes*, Thesis Eindhoven University of Technology, Dec. 1996.
- [6] S.W. Golomb, *Shift Register Sequences*, Holden-Day, Inc., San Francisco, 1967.