

## Round-up of integer bit allocation

Linfeng Guo and Yan Meng

A non-iterative algorithm is presented for the round-up procedure of integer bit allocation. The round-up algorithm will give the optimum result without the disadvantages of the traditional iterative method.

*Introduction:* Bit allocation is an important topic in many source coding and communication applications. Given target average bit rate  $R$  and  $N$  independent scalar linear quantisers, it is well known [1] that the optimal bit allocation result subjects to

$$R_k = R + \frac{1}{2} \log_2 \frac{\sigma_k^2}{\left[ \prod_{j=0}^N \sigma_j^2 \right]^{1/N}}$$

where  $R_k$  and  $\sigma_k^2$  are the allocated bit number and Gaussian-distributed input variance of the  $k$ th quantiser. This also implies identical error variance for each quantiser:

$$\sigma_{qk}^2 = \varepsilon \sigma_i^2 2^{-2R_k} = \varepsilon 2^{-2R} \left[ \prod_{j=1}^N (\sigma_j^2) \right]^{1/N} = \sigma_{opt}^2$$

where  $\sigma_{qk}^2$  is the quantisation error variance for the  $k$ th quantiser, and  $\sigma_{opt}^2$  is the derived error variance for the  $k$ th quantiser under the optimal bit allocation.

Quite often the allocated bit number of each quantiser needs to be an integer. However, the results of classical optimal bit allocation normally contain non-integer numbers. Thus, a final adjustment needs to be implemented. This procedure is often called the round-up procedure. Common ways of doing this are still based upon the iterative try-and-rund methods [2, 3], and no mathematically derived non-iterative solution is proposed. To give an example, Wintz and Kurtenbach [3] proposed an iterative round-up adjustment method to meet the integer bit allocation requirement, such that the values determined are rounded to the nearest integer. Without loss of generality, suppose the non-integer non-negative bit allocation results are in the following descending order:

$$R_1 \geq R_2 \geq \dots \geq R_n$$

Therefore, should the assignment not satisfy the target bit rate, some of the  $R_i$  are arbitrarily adjusted according to the following rules:

- (i) If  $R < \sum_{i=1}^N R_i$ , take that  $R_i$  corresponding to the largest  $i$  such that  $R_i > 1$  and replace it by  $R_i - 1$ .
- (ii) If  $R > \sum_{i=1}^N R_i$ , take that  $R_i$  corresponding to the smallest  $i$  such that  $R_i = R_n$  and replace it by  $R_i + 1$ .

Wintz and Kurtenbach claimed that rules (i) and (ii) constitute a sub-optimum algorithm, which provides an assignment of the  $R$  bits to the  $N$  quantisers.

In this Letter, we present a non-iterative round-up algorithm as well as its mathematical analysis. The analysis of this method is based upon the analysis of marginal returns, such that the minimum error distortion is achieved under the integer constraint.

Suppose the target average bit rate is  $R$ , and the non-negative allocated bit rates of  $N$  quantisers are  $R_i$ , where  $i = 1, 2, \dots, N$ , and the decimal part of  $R_i$  is  $r_i$ . They satisfy

$$R = \frac{1}{N} \sum_{i=1}^N R_i, \text{ or } NR = \sum_{i=1}^N R_i$$

and

$$R_i = \lfloor R_i \rfloor + r_i$$

where  $\lfloor \cdot \rfloor$  is the floor function. The goal of the round-up procedure is to convert each non-integer bit rate value to an integer value. Here, the decimal part of each bit rate can be treated as a random variable ranging in  $[0, 1)$  and having uniform distribution. The problem is equivalent to how to change each  $r_i$  to either 1 or 0 so that the increase of total quantisation error variances is minimal while the target average bit rate remains.

The proposed non-iterative round-up procedure is:

*Step 1:* Set each non-integer bit rate to the largest integer number that is smaller than it.

*Step 2:* Calculate the residue bit budget  $R_s = NR - \sum_{i=1}^N \lfloor R_i \rfloor$ .

*Step 3:* Calculate the decimal part of each bit rate  $r_i = R_i - \lfloor R_i \rfloor$  and rank them in descending sequence, with the first the largest and the last the smallest.

*Step 4:* Each bit rate of the first  $\lfloor R_s \rfloor$  quantisers in the sequence derived from step 3 will be changed to the smallest integer that is larger than it, and each bit rate of the rest is changed to the largest integer that is smaller than it.

The optimality of the above procedure under the integer constraint is proved as below.

Assume after the round-up procedure, allocated bit rate  $R_i$  of the  $i$ th quantiser is adjusted to  $R'_i$ . If  $R_i$  is rounded up to  $\lceil R_i \rceil$ , where  $\lceil \cdot \rceil$  is the ceil function, its bit rate change is  $1 - r_i$ , i.e.  $R'_i = R_i + 1 - r_i$ . Its error variance changes to

$$\hat{\sigma}_{qi}^2 = \varepsilon \sigma_i^2 2^{-2\lceil R_i \rceil} = \varepsilon \sigma_i^2 2^{-2(R_i+1-r_i)} = \varepsilon \sigma_i^2 2^{-2R_i} \times 2^{2r_i-2}$$

Thus we have

$$\hat{\sigma}_{qi}^2 = \sigma_{opt}^2 \times \frac{1}{4} 2^{2r_i}$$

Similarly, if  $R_i$  is rounded up to  $\lfloor R_i \rfloor$ , the change of bit rate is  $-r_i$ , i.e.  $R'_i = R_i - r_i$ . The error variance becomes

$$\hat{\sigma}_{qi}^2 = \sigma_{opt}^2 \times 2^{2r_i}$$

That is to say, after the round-up procedure some of the  $r_i$  would be changed to 1, while others will be changed to zero. Without losing commonality, let us assume the first  $M$   $r_i$  values are rounded up to 1, and the rest are floored to 0.

*Lemma:* No quantiser should receive more than 1 bit in step 4 of the proposed adjustment algorithm.

*Proof:* This can be proved in two steps.

*Step 1:* No quantiser can obtain two bits during the round-up procedure, with the increasing of total quantisation error variances minimum.

*Proof:* Suppose it is not true. There exists at least one quantiser that obtains two bits during the final adjustment. Without loss of generality, let it be the  $i$ th quantiser with decimal part  $r_i$  in its original bit rate. Considering the  $j$ th quantiser whose decimal part of original bit rate equals  $r_j$  and changed to 0 during the roundup. If we decrease the bit rate of the  $i$ th quantiser by 1, and assign it to the  $j$ th quantiser, the change of quantisation error is

$$\begin{aligned} \Delta \sigma_q^2 &= (\sigma_{opt}^2 2^{2(r_i-1)} + \sigma_{opt}^2 2^{2(r_j-1)}) - (\sigma_{opt}^2 2^{2(r_i-2)} + \sigma_{opt}^2 2^{2r_j}) \\ &= \sigma_{opt}^2 \left( \frac{1}{4} 2^{2r_j} + \frac{1}{4} 2^{2r_i} \right) - \sigma_{opt}^2 (2^{2r_j} + 2^{2(r_i-2)}) \\ &= \sigma_{opt}^2 \left( \frac{3}{16} 2^{2r_i} - \frac{3}{4} 2^{2r_j} \right) = \frac{3}{16} \sigma_{opt}^2 (2^{2r_i} - 2^{2(1+r_j)}) \end{aligned}$$

Considering  $r_i \in [0, 1)$ ,  $r_j \in [0, 1)$ , and the monotonically increasing property of  $2^x$ , it is evident that

$$\Delta \sigma_q^2 < 0$$

that is to say, assigning two bits to a single quantiser during the final adjustment is not optimal.

*Step 2:* No quantiser should receive  $n$  bits with  $n > 2$ .

*Proof:* If it is not true. There exists at least one quantiser that receives  $n > 2$  bits during the final adjustment. Similarly, assume it is the  $i$ th quantiser with the decimal part of its bit rate equals to  $r_i$ . Considering the other  $n-1$  quantisers whose decimal parts of bit rates are  $r_1$  to  $r_{n-1}$ , and set to 0 during the round up. If we redistribute the  $n$  bits to these  $n$  quantisers with one bit for each, the difference of total quantisation errors between these two situations is

$$\begin{aligned} \Delta \sigma_q^2 &= \sigma_{opt}^2 \left( \sum_{k=1}^{n-1} \frac{1}{4} 2^{2r_k} + \frac{1}{4} 2^{2r_i} \right) - \sigma_{opt}^2 \left( \sum_{k=1}^{n-1} 2^{2r_k} + 2^{2(r_i-n)} \right) \\ &= \frac{1}{4} \sigma_{opt}^2 \left( \left( 1 - \left( \frac{1}{4} \right)^{n-1} \right) 2^{2r_i} - 3 \sum_{k=1}^{n-1} 2^{2r_k} \right) \end{aligned}$$

Assume  $r_m$  is the smallest value among  $r_1$  to  $r_{n-1}$ , we have

$$\Delta \sigma_q^2 < \frac{1}{4} \sigma_{opt}^2 \left( \left( 1 - \left( \frac{1}{4} \right)^{n-1} \right) 2^{2r_i} - 3(n-1)2^{2r_m} \right)$$

Conversely, considering  $n > 2$  and  $n$  is integer, we have

$$\frac{1 - (1/4)^{n-1} \cdot 2^{2r_i}}{3(n-1)2^{2r_m}} < \frac{2^{2(r_i-r_m)}}{3(n-1)} < \frac{2^2}{3(n-1)} < 1$$

Thus

$$\frac{1}{4} \sigma_{opt}^2 \left( \left( 1 - \left( \frac{1}{4} \right)^{n-1} \right) 2^{2r_i} - 3(n-1)2^{2r_m} \right) < 0$$

and

$$\Delta \sigma_q^2 < 0$$

Therefore, no quantiser should get more than one bit during the roundup procedure.

*Proof:* The roundup procedure can be treated as distributing  $M = \sum_{i=1}^N r_i = NR - \sum_{i=1}^N \lfloor R_i \rfloor$  bits to  $N$  quantisers, whose original bit rates are  $R_i$ . Assume the first  $M$  quantisers are rounded up to  $\lceil R_i \rceil$  and rest to  $\lfloor R_i \rfloor$ . The total error variance after the round up is

$$\begin{aligned} D &= \varepsilon \sum_{i=1}^M 2^{-2\lceil R_i \rceil} \sigma_i^2 + \varepsilon \sum_{i=M+1}^N 2^{-2\lfloor R_i \rfloor} \sigma_i^2 \\ &= \varepsilon \sum_{i=1}^M 2^{-2(R_i+1-r_i)} \sigma_i^2 + \varepsilon \sum_{i=M+1}^N 2^{-2(R_i-r_i)} \sigma_i^2 \\ &= \frac{1}{4} \sum_{i=1}^M (\varepsilon 2^{-2R_i} \sigma_i^2) 2^{2r_i} + \sum_{i=M+1}^N (\varepsilon 2^{-2R_i} \sigma_i^2) 2^{2r_i} \\ &= \sigma_{opt}^2 \left( \frac{1}{4} \sum_{i=1}^M 4^{r_i} + \sum_{i=M+1}^N 4^{r_i} \right) \end{aligned}$$

To keep  $D$  minimum, it is clear that the  $M$  residue bits should be evenly distributed to the  $M$  quantisers with the largest  $M r_i$  values.

*Conclusion:* A non-iterative optimal round-up algorithm is proposed for the final adjustment of optimal bit allocation. It implies the final adjustment should be based on the values of the decimal parts of those allocated bit rates rather than the values of those bit rates themselves.

© IEE 2002

29 December 2001

Electronics Letters Online No: 20020329

DOI: 10.1049/el:20020329

Linfeng Guo and Yan Meng (Department of Electrical Engineering, Florida Atlantic University, 777 Glades Road, Boca Raton, FL 33430, USA)

E-mail: glinfeng@hotmail.com

## References

- JAYANT, N.S., and NOLL, P.: 'Digital coding of waveforms' (Prentice Hall, Englewood Cliffs, NJ, 1984)
- HUANG, Y., and SCHULTHEISS, P.M.: 'Block quantization of correlated gaussian random variables', *IEEE Trans. Commun. Syst.*, 1963, **11**, (3), pp. 289-296
- WINTZ, P.A., and KURTENBACH, A.J.: 'Waveform error control in PCM telemetry', *IEEE Trans. Inf. Theory*, 1968, **IT-14**, pp. 650-661

## Extended Hamming accumulate codes and modified irregular repeat accumulate codes

Sung Ik Park and Kyeongcheol Yang

Two classes of codes, called extended Hamming accumulate codes and modified nonsystematic irregular repeat accumulate (IRA) codes, are introduced. Simulation results show that the performance of modified nonsystematic IRA codes are slightly superior to turbo codes of comparable complexity on an additive white Gaussian noise channel.

*Introduction:* Since Shannon's work in 1948, coding theorists have attempted to design error-correcting codes having performance close to the Shannon limit. Spectacular progress has recently been made on capacity-approaching coding schemes, using codes that are naturally defined on graphs, e.g. turbo codes and low density parity check (LDPC) codes [1, 2].

Remarkably good results also have been achieved with repeat accumulate (RA) codes [3]. Irregular repeat accumulate (IRA) codes are composed of simple irregular repetition codes, a pseudorandom interleaver, and a trivial rate-1 two-state 'accumulate' code [4].

In this Letter we propose two classes of codes, called extended Hamming accumulate (EHA) codes and modified nonsystematic irregular repeat accumulate (IRA) codes, obtained by replacing half-rate repetition codes with [8,4,4] extended Hamming codes in the conventional RA and IRA codes, respectively. Simulation results show that the performance of nonsystematic IRA codes are slightly superior to turbo codes of comparable complexity over an additive white Gaussian noise (AWGN) channel.

*Extended Hamming accumulate codes:* An extended Hamming accumulate (EHA) code is a half-rate serially concatenated code which consists of a [8,4,4] extended Hamming code as an outer code and accumulator as an inner code. This code has a similar structure as in the half-rate RA codes, except that the half-rate repetition codes are replaced by [8,4,4] extended Hamming codes which have the parity check matrix given by

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

and the factor graph shown in Fig. 1 (in this Figure the empty circles, black circles, and check boxes denote information nodes, parity nodes and check nodes, respectively).  $\gamma(0), \gamma(1), \dots, \gamma(7)$  denote the log-likelihood ratio (LLR) values of the messages coming to the information and parity nodes of the extended Hamming code from the inner codes or the channels.

To apply the sum-product algorithm easily to the decoding of the extended Hamming code, it may be necessary to represent its factor graph in Fig. 1 in a tree representation. In this representation the same edge is not used repeatedly if it was used in a previous branch. For example, Fig. 2 shows a tree representation of the first information node of the factor graph in Fig. 1.

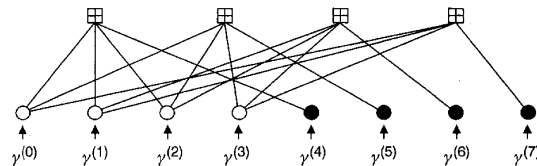


Fig. 1 Factor graph of [8,4,4] extended Hamming code

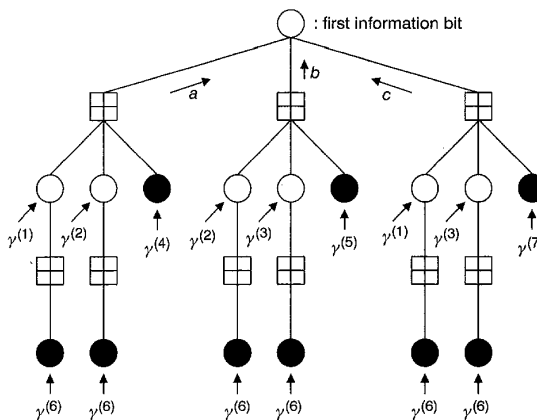


Fig. 2 Tree representation of first information node in [8,4,4] extended Hamming code

Let  $M(i)$  be the message of the  $i$ th information node in the decision step. Using the sum-product algorithm, the updating rule of the first information node is given by

$$M(0) = a + b + c + \gamma(0)$$