Unary-prefixed encoding of lengths of consecutive zeros in bit vector

S. Xue and B. Oelmann

The unary-prefixed encoding (UPE) algorithm in coding the lengths of zeros in a bit vector is proposed. While the lengths of consecutive zeros in a bit vector can be mapped to integer sources with geometrical distribution (when the bits in the bit vector are independent from each other), the actual case is more often that the distributions are exponential, in a more real-world situation, with high peaks and heavier tails (when the bits in a bit vector are correlated). For the geometric distribution, the UPE code set can be proven to be optimal. For the integer sources with high peaks and heavier tails, the UPE almost always provides better compression compared to the existing suboptimal codes.

Introduction: Golomb [1] observed that the lengths of consecutive zeros of an independent and identically distributed (i.i.d.) source is geometrically distributed and can therefore be described using the integer source with probability density function (pdf):

$$p_k^\theta = (1-\theta)\theta^k, \quad 0 < \theta < 1 \tag{1}$$

Such an integer source is of an infinite alphabet and cannot be coded using the Huffman coding algorithm. Therefore optimal codes are difficult to construct. For the pdf in (1), Golomb studied the case when θ is a power of $1/2$ and introduced a class of optimal codes that is called Golomb Rice (GR) code. Gallager and Van Voorhis [2] generalised Golomb's result by allowing to vary in the range $0 < \theta < 1$ and proved that the optimal code for the pdf in (1) can be obtained as follows.

Let l be the integer satisfying:

$$\theta^l + \theta^{(l+1)} \leq 1 \leq \theta^{(l-1)} + \theta^l \tag{2}$$

and represent each non-negative integer k as $k = lj + r$ where $j = \lfloor k/j \rfloor$, and $r = [k] \bmod l$. Gallager and Van Voorhis encoded j by a unary code, and encoded r by a Huffman of length $\lfloor \log_2 l \rfloor$, for $r < 2^{\lfloor \log_2 l \rfloor + 1} - l$, and length $\lfloor \log_2 l \rfloor + 1$ otherwise. The resulting code is a concatenation of the unary prefix for j and the Huffman suffix.

In practice, the bits in a bit vector are usually not generated from i.i.d.'s. Therefore the geometric integer source is empirically unsatisfactory. Exponential integer sources with heavier tails are more often found to be suitable. Teuhola [3] introduced a class of codes under the name 'exp-Golomb' (EG) codes. The EG codes are widely used in practice, which, although suboptimal, have been found to be efficient for any particular exponential distribution and have found applications in subband image codings. With the parameter s, every codeword group with 2^{s+l-1} ($l = 1, 2, 3, \ldots$) codewords are encoded by assigning a common unary prefix for l and fixed-length $(s + l - 1)$-bit binary suffixes for each code within the group.

Kiely and Klimesh [4] designed a class of pdf's that are well matched to the EG codes and they also showed that these pdf's are good probability models for empirically observed integer sources. These integer sources can be expressed using the pdf:

$$p_k^\alpha = \frac{1}{\psi'(\alpha)}(\alpha + k)^{-2} \tag{3}$$

where $\alpha > 0$, ψ' is the first derivative of the digamma function $\psi(y) = \Gamma'(y)/(\Gamma(y))$, and Γ is the Euler gamma function.

The UPE we propose in this Letter focuses on the coding of the integer sources with the distributions described in (3) since it provides a good practical model. Actually it can be proven that, for geometric distributions in (1), the codes constructed by the UPE are equivalent to those described in [2] and are therefore optimal. For the probability distribution in (3), the code sets resulting from the UPE are shown to have better compression than the existing EG codes.

UPE algorithm: The basic idea of the UPE is to segment an infinite integer source with probability distribution $\{p_k\}_{k=0}^{\infty}$ into subsets $\{p_1\}_{l=1}^{\infty}$, with $P_1 = \{p_{s_{l-1}}, p_{s_{l-1}}+1, p_{s_{l-1}}+2, \ldots, p_{s_l}-1\}$ and $S_l = \Sigma_{i=s_{l-1}}^{s_l-1} p_i$, where the summations of the subsets $\{S_l\}_{l=1}^{\infty}$ are made to be as close to $\{1/2^l\}_{l=1}^{\infty}$ as possible. For the $N_l = s_l - s_{l-1}$ probability values within each subset P_l, we assume them to be equal and then perform Huffman coding to these N_l equal probability values. Binary codes of length $\lfloor \log_2 N_l \rfloor$ or $\lfloor \log_2 N_l \rfloor + 1$ will then be assigned to the probability values in the subset P_l. For each codeword within P_l, the UPE code is then expressed as a concatenation of a unary prefix for l and the binary suffix of length $\lfloor \log_2 N_l \rfloor$ or $\lfloor \log_2 N_l \rfloor + 1$.

The UPE algorithm can be fully described by the following steps:

1 Let $s_0 = 0$.

2 For $l = 0$ to ∞, let:

$$S_{-l} = \sum_{i=s_l}^{\infty} p_i \tag{4}$$

Performing normalisation to the probability set $\{p_{s_l}, p_{s_l}+1, p_{s_l}+2, \ldots, p_{s_{l+j}}, \ldots\}$, we have:

$$\bar{P}_l = \left\{\frac{p_{s_l}}{S_{-l}}, \frac{p_{s_l+1}}{S_{-l}}, \frac{p_{s_l+2}}{S_{-l}}, \ldots, \frac{p_{s_l+j}}{S_{-l}}, \ldots\right\} \tag{5}$$

3 Find s_{l+1} such that:

$$\left|\frac{1}{2} - \sum_{i=s_1}^{s_{l+1}-1} \frac{p_i}{S_{-1}}\right| \tag{6}$$

is minimised.

4 Let:

$$P_{l+1} = \{p_{s_l}, p_{s_{l+1}}, \ldots, p_{s_{l+1}-1}\} \tag{6}$$

$$S_{l+1} = p_{s_1} + p_{s_{1+1}} + \cdots + p_{s_{1+1}-1} \tag{7}$$

For probability set P_{l+1}, there are $s_{l+1} - s_l$ probability values. We assume these $N_{l+1} = s_{l+1} - s_l$ probabilities to be equal to each other and then perform Huffman coding. The resulting codes will be binary codes either of length $\lfloor \log_2 N_{l+1} \rfloor$ or $\lfloor \log_2 N_{l+1} \rfloor + 1$. We assign a common unary prefix 111...10 (with l ones in a row) or equivalently 000...01 (with l zeros) to each of these binary codes and thus we get the UPE codes.

5 Let $l = l + 1$ and repeat from step 2.

Let us look at a simple example. Suppose we have an infinite probability distribution: $\{p_k = 1/3 \cdot 2^{\lfloor k/3 \rfloor + 1}\}_{k=0}^{\infty}$, which looks like: {1/6, 1/6, 1/6, 1/12, 1/12, 1/12, 1/24, 1/24, 1/24,... }. By performing the UPE algorithm, we will get $\{P_l = \{1/3 \cdot 2^l, 1/3 \cdot 2^l, 1/3 \cdot 2^l\}\}_{l=1}^{\infty}$, $N_l = 3$ and $\{S_l = 1/2^l\}_{l=1}^{\infty}$. The three probability values in each subset P_l are already equal to each other, so the Huffman codes would be {1,00,01} or {0,11,10}. The UPE is then a concatenation of the common unary prefix 111...10 (with l ones in a row) or equivalently 000...01 (with l zeros) and one of the Huffman codes accordingly.

Performance of UPE codes: For the UPE codes, the codes within each probability subset P_l are assigned a common l-bit unary prefix. The EG codes also have a common l-bit unary prefixes for every 2^{s+l-1} codewords, with 2^{s+l-1} probability values associated with them. Within the subset of codewords sharing the same prefix, it can be shown that, in the EG codes, as well as in the UPE codes, Huffman coding is applied to generate the suffixes by assuming the 2^{s+l-1} probability values (for EG) or the probability values in P_l (for UPE) to be equal. It can be proven that the UPE algorithm is able to segment the probability sequence into subsets whose summations are optimally coded using unary codes; therefore the prefixes of the UPE codes are optimal. As the coding strategy of the EG and the UPE are the same for the suffixes, the UPE codes in general perform better than the EG codes in terms of compression.

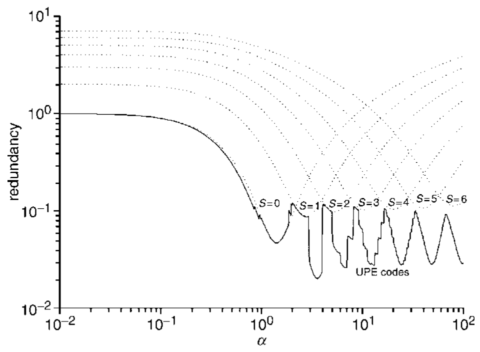

Fig. 1 *Comparison of redundancies of EG and UPE codes*

Fig. 1 shows a comparison of the redundancies of the UPE codes and the EG codes with different s in coding the pdf's in (3) under a wide range of different α values. The group of pdf's in (3), as mentioned earlier, has been shown to be good models for the distributions of lengths of zeros in many practical cases such as in coding the quantised subband of wavelet-transformed images [4]. From the Figure, it is obvious that the UPE codes are better in compression compared to the EG codes. Moreover, since the UPE algorithm works adaptively according to different pdf's with different parameters, we do not need to make selections of s to get a better performance, which is the case for the EG codes.

Conclusions: A UPE algorithm in coding the lengths of zeros in a bit vector is proposed. Compared to the existing codes, the UPE algorithm works adaptively according to the source, and provides good matches to the source pdf's. The UPE codes achieve optimality for geometric distributions and for more empirical sources, they are able to outperform the EG codes, which are widely used in practice, in terms of coding redundancy.

Acknowledgment: The authors would like to thank N. Gu, at the department of Mathematics, Purdue University, for discussions and proof reading.

© IEE 2005 *22 October 2004*
Electronics Letters online no: 20057325
doi: 10.1049/el:20057325

S. Xue and B. Oelmann (*Department of Information Technology and Media, Mid Sweden University, Sundsvall SE-851 70, Sweden*)

E-mail: xue.shang@mh.se

References

1 Golomb, S.W.: 'Run-length encodings', *IEEE Trans. Inf. Theory*, 1966, **7**, (12), pp. 399–401
2 Gallager, R.G., and Van Voorhis, D.C.: 'Optimal source codes for geometrically distributed integer alphabets', *IEEE Trans. Inf. Theory*, 1975, **3**, (21), pp. 228–230
3 Teuhola, J.: 'A compression method for clustered bit-vectors', *Inf. Process. Lett.*, 1978, **10**, (7), pp. 308–311
4 Kiely, A., and Klimesh, M.: 'Generalized Golomb codes and adaptive coding of wavelet-transformed image sub-bands', (IPN PR 42-154) April–June 2003, pp. 1–14