Efficient Run-Length Encoding of Binary Sources with Unknown Statistics

Max H. M. Costa[*] and Henrique S. Malvar

December 19, 2003

Technical Report
MSR-TR-2003-95

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

[*] Max H. M. Costa is with the Departamento de Telecomunicações (DECOM), Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Caixa Postal 6101, 13081-970 Campinas, SP, Brazil. This work was developed while M. H. M. Costa was a visiting researcher at Microsoft Research.

Efficient Run-Length Encoding of Binary Sources with Unknown Statistics

Max. H. M. Costa and Henrique S. Malvar

Abstract

We present a new binary entropy coder of the Golomb family, with an adaptation strategy that is nearly optimum in a maximum-likelihood sense. This new encoder can be implemented efficiently in practice, since uses only integer arithmetic and no divisions. That way, the proposed encoder has a complexity nearly identical to that of popular adaptive Rice coders. However, whereas Golomb-Rice coders have an excess rate with respect to the source entropy of up to 4.2% for binary sources with unknown statistics, the proposed encoder has an excess rate of less than 2%.

1. Introduction

Binary entropy coders are used in many modern compression standards, such as JPEG-LS [1], JPEG2000 [2], and H.264 [3]. They are especially useful in scalable encoders, in which quantized data is encoded in bit planes, so that lower-fidelity representations can be obtained by just truncating the bitstream [2]. Within each bit plane the bits are usually classified according to different contexts, and each has significantly different statistics. Therefore, the binary encoder needs to maintain its efficiency under a wide variation of source statistics, not only across encoding blocks, but usually within such blocks.

Let us consider a binary memoryless source whose most probable symbol is zero, which occurs with probability $\theta \geq ½$ (we can always add a simple predictor that flips the source bits if $\theta < ½$, so there is no loss of generality). It is well known [4] that a stationary source with known θ can be efficiently encoded with a run-length Golomb code [5]. Golomb codes are controlled by a parameter *M*, and map input strings to variable-length codewords in the following way:

- Input string is a sequence of *M* zeros (a run of length *M*): emit the codeword "0".
- Input string is "000..01", a complete run of $x < M$ zeros followed by 1: emit a codeword starting with a "1", of length $\lfloor \log_2 M \rfloor$ if $x < 2^{\lfloor \log_2 M + 1 \rfloor} - M$, and length $\lfloor \log_2 M \rfloor + 1$, otherwise.

The Golomb code was later independently rediscovered as truncated run-length (TRL) code [6] and MELCODE [7]. In spite of its simplicity, the Golomb code is quite efficient, since its rate is at most 4.1% above the source entropy, and on average only 2% above the source entropy, if the parameter *M* is appropriately chosen (see the Appendix). That near optimality is a consequence of the Golomb code being the Huffman code for the Tunstall extension of the source with dictionary length *M* + 1 [8]. The Golomb code is relatively simple, but its implementation needs to use divisions or large tables, if a wide range of *M* is supported. For encoding high bit planes of quantized transform coefficients of pictures, for example, values of M above 500 may be needed, since the source entropy may be as low as $H = 0.01$ bits/bit. The encoding rule is much simpler, and requires no tables, if $M = 2^k$, a case pointed by Golomb [5] but later rediscovered by Teuhola [9], Rice [10], and others; in that case we usually refer to the run-length coder as a Rice coder [11].

In this paper we propose the use of Golomb codes with two subsets of parameters, as well as a special additional code, as in TRL codes [6]. In Section 2 we describe the code, and introduce new conditions for optimality. In Section 3 we derive a maximum-likelihood estimator for M using previously encoded-symbols, which allows the encoder to track changes in θ. In Section 4 we review the performance for stationary and non-stationary sources. Conclusions are presented in Section 5.

2. Efficient encodings of the Golomb kind

We propose the code defined in Table 1. It uses two subsets of Golomb codes: $M = 2^k$ (for $h = 0$) or $M = 3 \cdot 2^{k-1}$ (for $h = 1$; a “half-mode”). The code for the special case $k = 0$ and $h = 1$ is not a pure run-length code, but a code for the vector formed by one symbol plus a run of up to two zeros. The corresponding excess rate is shown in Fig. 1 (the excess rate is defined as the ratio $(R–H)/H$, where R is the average bit rate of the code per input binary symbol, and H is the source entropy in bits/symbol). The code in Table 1 is a Rice code for $h = 0$, so for that case the curves in Fig. 1 match. Note that the curves in Fig. 1 assume that the optimal parameters $\{k, h\}$ are used, for any θ, a problem that we address in the Appendix. We see that the Rice code has a peak excess rate of 4.17%,

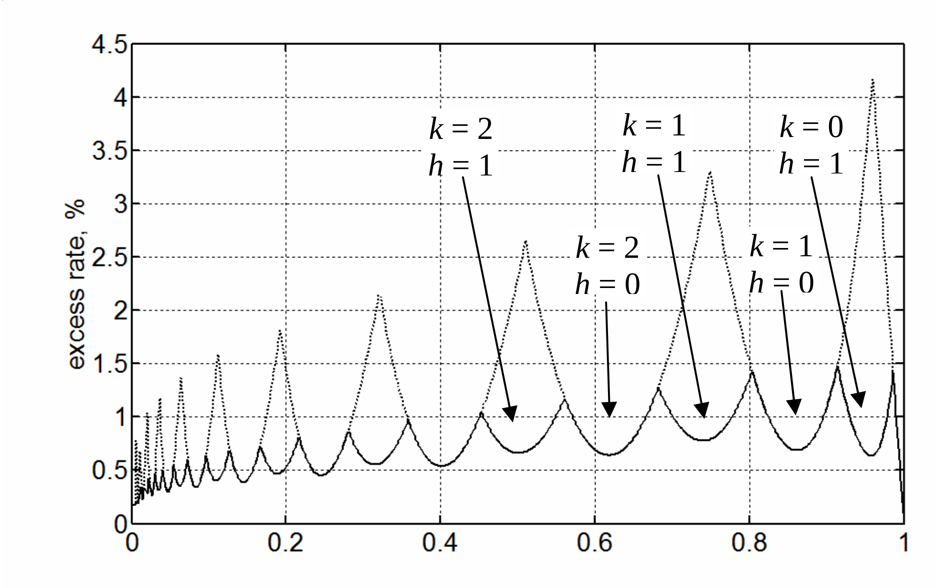

Figure 1. Excess rate for the code in Table 1 using optimal $\{k, h\}$. The dotted line shows the excess rate for the Golomb-Rice code.

Parameter value	Input string	Run length	Output Codeword	Δ
$k = 0$ $(M = 1)$ $h = 0$	0 1	1 0	0 1	$+2$ $+2$
$k = 0$ $h = 1$ (not a true Golomb code)	0.00 0.01 0.1	n/d	00 100 01	$+2$ $+1$ 0
	1.00 1.01 1.1		101 110 111	-1 -1 -4
$k = 1, 2, \ldots$ $(M = 2^k)$ $h = 0$	00…0	M	0	$+3$
	00…01	x	1 *bb…b* (*bbb…b* = binary representation of *x*)	-4
$k = 1, 2, \ldots$ $(M = 3 \cdot 2^{k-1})$ $h = 1$	00…0	M	0	$+3$
	00…01	x	10 *bb…b* if $0 \le x < 2^{k-1}$ (*bbb…b* = binary representation of *x*) 11 *cc…c* if $2^{k-1} \le x < 3 \cdot 2^{k-1}$ (*ccc…c* = binary representation of $x - 2^{k-1}$)	-4

Table 1. Proposed code with parameters *k*, and *h*: *k* controls the choice of the Golomb parameter *M*, and *h* = 1 is a "half-mode". The special code for {*k* = 0, *h* = 1} is not a run-length code, but it is a Tunstall-Huffman combination for the range of probabilities shown in Fig. 1 and discussed in the Appendix.

whereas the proposed code has a peak excess rate of 1.48%, for a source with known θ. It is clear from Fig. 1 that the excess rate may quickly grow for an incorrect choice of the {*k*, *h*} parameters. Therefore, in practice it is important to monitor θ and automatically adjust those parameters.

3. Adaptation strategies

The usual approach for the design of adaptive binary encoders is to estimate θ and then adjust the encoding parameters accordingly. The encoder can either look ahead for a certain number of source symbols and then determine the parameters (forward adaptation, in which the parameter values precede each encoded block of symbols), or the parameter estimation can be computed based on previously-encoded strings (backward adaptation, in which the decoder can duplicate the estimation procedure used at the encoder). In this paper we estimate the {*k*, *h*} parameters for the encoder in Table 1 via a maximum-likelihood (ML) approach, based on the last *N* previously-encoded strings. We chose a backward approach because of its ability to quickly track varying θ, as well as for the lack of overhead (no bits needed to specify parameter values).

For the binary memoryless source, the expected run length $\overline{x} \equiv E[x]$ is given by

$$\overline{x} = \frac{\theta}{1-\theta} \quad \Leftrightarrow \quad \theta = \frac{\overline{x}}{1+\overline{x}} \tag{1}$$

The ML estimate for θ, given an observation of a string of symbols with n_0 zeros and n_1 ones, is the classical relative frequency estimator:

$$\hat{\theta} = \frac{n_0}{n_0 + n_1} \quad \Leftrightarrow \quad \hat{\overline{x}} = \frac{n_0}{n_1} \tag{2}$$

where the caret (^) denotes an estimated value. The ML estimate of the expected run-length $\overline{x}$ is given[†] by the ratio n_0/n_1.

Suppose we have seen *N* previous runs, from which we built the current expected run-length estimate $\overline{x}$ by

$$\overline{x} = \frac{x_0 + x_{-1} + \cdots + x_{-(N-1)}}{N} \tag{3}$$

where x_0 is the last observed run, x_{-1} the one before the last, and so on. Combining (3) and (2), we can update the cumulative run length estimate by

$$N\overline{x} \leftarrow \frac{N}{N+n_1}\left(N\,\overline{x} + n_0\right) \tag{4}$$

after each codeword is generated, with n_0 and n_1 as the number of zeros and ones, respectively, in the input string (cf. Table 1). Although apparently simple, the estimate in (4) has the drawback of requiring a division by $N + n_1$. We can avoid it by using the approximation $N/(N+n_1) \simeq (N-n_1)/N$; the error in this approximation is of the order of $(n_1/N)^2$. Thus, it works well because usually $n_1 \ll N$ (from Table 1 we see that $n_1 \le 2$ and we usually set *N* around 32 or so). The result is

$$N\overline{x} \leftarrow \frac{N-n_1}{N}\left(N\,\overline{x} + n_0\right) \tag{5}$$

In practice, we can keep the value of $N\overline{x}$ in an integer variable, updating it according to (5) for every new encoded string. If we set *N* equal to a power of 2, then the division becomes just a shift. The multiplication by $(N - n_1)$ may be replaced by at most two shifts and two if statements, if necessary, since n_1 can only assume the values 0, 1, or 2.

[†] The estimate for $\overline{x}$ is also ML because of the remarkable property that ML estimators commute with nonlinear monotonic functions [12]. It can also be shown that $\hat{\theta}$ is unbiased but not efficient, i.e., it does not meet the Cramer-Rao bound. As it is an ML estimator, it becomes asymptotically efficient for larger strings.

Given the updated $\bar{x}$, we can look into a short table of transition points to determine the $\{k, h\}$ parameters to be used for the next symbol [see Appendix]. Thus, the adaptation rule in (5) is nearly ML and can be efficiently implemented in practice.

We also consider a simple sub-optimal adaptation strategy, where the parameters are fractionally incremented by the following steps:

- Encode current string s according to Table 1, for the current values of $\{k, h\}$.
- Adjust a scaled parameter k' by

$$k' \leftarrow k' + \Delta(s, k, h) \tag{6}$$

  where the increment parameter Δ in general depends on the string s and the code parameters $\{k, h\}$. In Table 1 we present a typical set of increment parameters, but they can easily be adjusted with finer granularity, if desired.

- Set $k'' = \lfloor 2k'/L \rfloor$, and then set $k = \lfloor k''/2 \rfloor$ and $h = 1$ if k'' is odd, $h = 0$ otherwise. We choose $L = 2^l$, so that the division by L becomes a right shift by l units.

The parameters N for the ML rule and L for the simplified adaptation rule control the fundamental tradeoff between speed of adaptation and excess rate for stationary sources. Larger values favor lower excess rates, and smaller values favor speed.

4. Performance

In Fig. 2 we compare the performance of our proposed code, using both the ML and the simplified adaptation rules, for a stationary source. The curves show average measured rates after convergence (averaging over 10^6 samples). We see that the simple adaptation rule performs quite well when compared to the optimal ML rule. The performance of our new code is particularly improved at high entropies, when compared to [11], because of the support for a Golomb mode with $M = 3$ and the addition of the "symbol+run" mode for $k = 0$ and $h = 1$. Fig. 3 shows the adaptation speed of the various methods, to test the performance with a stationary source with unknown θ. We initialize the parameters assuming $\theta = 0.7$, but encode a source with $\theta = 0.99$. The curve represents the average of 1,000 runs. We see that all adaptive Rice-like coders have a similar learning rate.

To test the performance of the coders under non-stationary conditions, we simulated a non-stationary input with a binary source whose θ is controlled by a binary state variable γ, which stays at $\gamma = 0$ with probability 0.995, and at $\gamma = 1$ with probability 0.95. For $\gamma = 0$, the source has $\theta = 0.95$, and for $\gamma = 1$, the source has $\theta = 0.7$. For the sample sizes considered, this simple Markov process behaves as a non-stationary source. In that case the observed excess rates for four selected adaptive coders were: [11] – 2.3%, [13] – 1.1%, proposed with ML rule – 1.8%, proposed with simple rule – 1.5%. Such results are typical under non-stationary situations, that is, the best performance is usually obtained with the coder in [13], a recently reported adaptive Golomb coder that was designed for non-stationary sources. Also, our simple adaptation rule usually outperforms the ML rule for nonstationary sources.

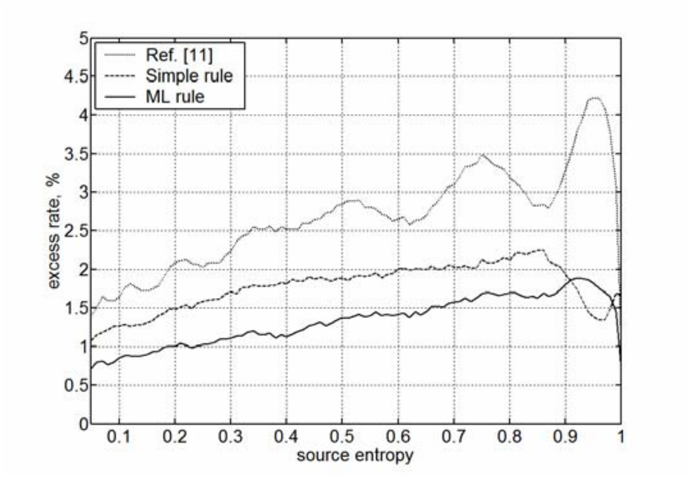

Figure 2. Excess rate for the proposed Golomb-like coder, with parameters L = 32 and N = 16, compared to an adaptive Rice coder [11], for stationary sources.

The performance of our ML encoder with non-stationary sources is expected to improve if different values of N are used for the ML estimates that respond to complete zero runs and to incomplete zero runs. The estimation of $\overline{x}$ after complete zero runs is important to decrease the value of M when it is set too high. Similarly, the adaptation that follows incomplete runs (runs of length M) is important to swiftly increase the value of M when it is set too low. Using different values of N for the ML estimates in these circumstances may allow for a better match of the encoder with the varying source statistics. For example, we could use N = 32 for complete zero run adaptation and N = 16 for incomplete zero run (and $\{k,h\}$ = $\{0,1\}$) adaptation. Naturally, N should be kept as a power of two for low computational complexity.

5. Conclusion

We have presented a new binary entropy coder that can encode binary stationary sources at rates that are at most 1.5% above the source entropy, whereas for Golomb-Rice coders the excess rate can be up to 4.2%. For efficient encoding of sources with unknown statistics, we showed that an ML estimate can be closely approximated by an update rule that is quite simple to compute. Thus, the proposed coder can be a good alternative to popular Golomb-Rice coders.

Appendix

We now derive the transition probabilities θ that change the optimal $\{k, h\}$ parameter values. For a Golomb coder the optimal parameter M is the unique integer for which [4]

$$\theta^M + \theta^{M+1} \le 1 < \theta^M + \theta^{M-1} \tag{A.1}$$

For a Rice coder with $M = 2^k$ the optimal M satisfies [6],[8]

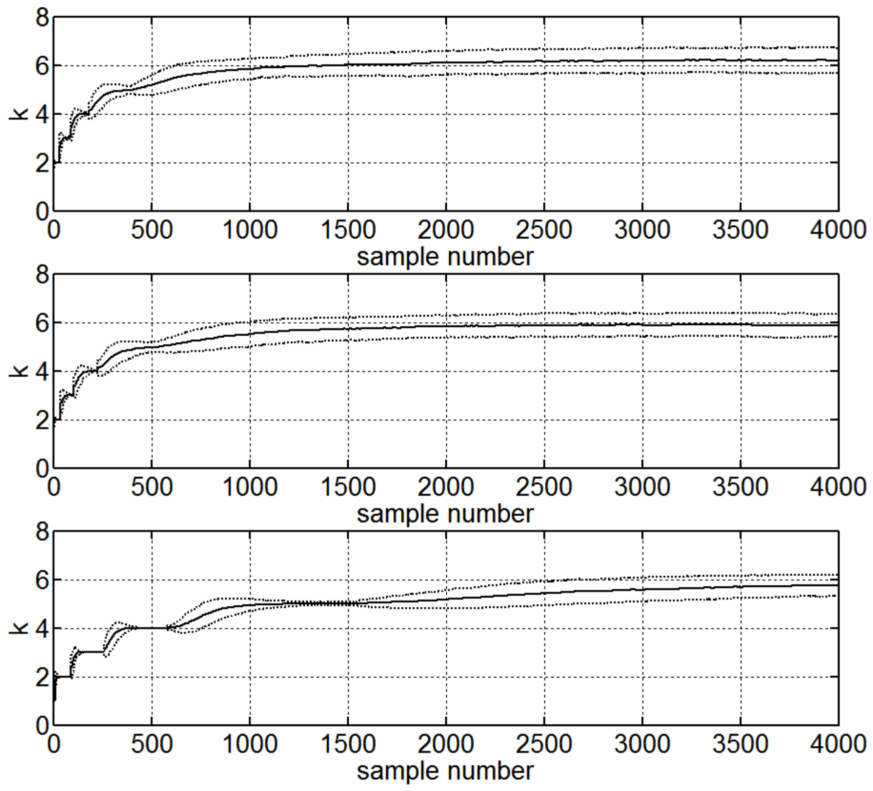

Figure 3. Learning curves; solid line is average, dotted lines are one standard deviation bounds. Top: Rice coder in [11] ; middle: proposed coder, simple adaptation rule, L = 32; bottom: proposed coder, ML rule, N = 16. For the proposed coder, we plot k as $k + h/2$.

$$\theta^M \leq \phi < \theta^{M/2} \tag{A.2}$$

where $\phi \triangleq \left(\sqrt{5}-1\right)\big/2$ is the golden ratio, the only positive root of $x^2+x-1=0$.

It is costly to evaluate the transition points from (A.1) and (A.2). A better approach is to start with expressions for the coding rate of Golomb codes. The coding rate in bits per input symbol is the ratio of the expected code length to the expected input string length, that is

$$R_G\left(\theta,M\right)=\frac{E\left[L_G\left(\theta,M\right)\right]}{E\left[L_C\left(\theta,M\right)\right]} \tag{A.3}$$

where $E[L_G\left(\theta,M\right)]$ is the expected length of the Golomb code for a source with probability θ and parameter M, and $E[L_C\left(\theta,M\right)]$ is the expected length of the corresponding comma code (that is, the runs being transmitted unencoded). It is easy to see that

$$L_C(\theta, M) = M\theta^M + \sum_{n=0}^{M-1}(n+1)(1-\theta)\theta^n = \frac{1-\theta^M}{1-\theta} \quad \text{(A.4)}$$

since the comma code encodes zero runs of length M as one intermediate symbol, and all the complete runs of length zero to $M-1$ as the remaining intermediate symbols.

For a Rice code, i.e. a Golomb code with $M = 2^k$ (i.e. the code of Table 1 for $h = 0$) a run of M zeros maps to a one-bit codeword, and codewords of $k + 1$ bits represent complete zero runs of length $0 \leq x < M - 1$. Thus, it follows

$$L_G\left(\theta, M = 2^k\right) = \theta^M + \sum_{n=0}^{M-1}(k+1)(1-\theta)\theta^n = \theta^M + (k+1)\left(1-\theta^M\right) \quad \text{(A.5)}$$

and the coding rate is

$$R_G\left(\theta, M = 2^k\right) = (1-\theta)\left(k + \frac{1}{1-\theta^M}\right) \quad \text{(A.6)}$$

This expression can be used to evaluate the crossover probabilities between consecutive Golomb/Rice codes. This is found by setting

$$R_G\left(\theta, M = 2^k\right) = R_G\left(\theta, M = 2^{k+1}\right) \quad \text{(A.7)}$$

By substituting $x = \theta^{2^k}$, the above equation is equivalent to $x^2 + x = 1$, whose solution is the golden ratio $\phi \triangleq \left(\sqrt{5}-1\right)/2$, as discussed before. So, the collection of crossover probabilities for the Rice case can be expressed as [6]

$$\theta_n = \phi^{2^{-n}}, \ n = 0, 1, 2, \ldots \quad \text{(A.8)}$$

Now, let's consider the cases where $M = 3 \cdot 2^{(k-1)}$. For the comma code, it follows that

$$L_C(\theta, M) = \frac{1-\theta^M}{1-\theta} = \frac{1-\theta^{3 \cdot 2^{(k-1)}}}{1-\theta} \quad \text{(A.9)}$$

Thus we can write

$$L_G\left(\theta, M = 3 \cdot 2^{k-1}\right) = \theta^M + (k+1)A + (k+2)B \quad \text{(A.10)}$$

where A is the joint probability of all zero runs of sizes $r = 0$ to $r = 2^{(k-1)} - 1$ and B is the joint probability of all remaining runs, of sizes $r = 2^{(k-1)}$ to $r = 3 \cdot 2^k - 1$, that is

$$A = \sum_{n=0}^{2^{k-1}-1}(1-\theta)\theta^n = 1-\theta^{2^{k-1}}, \ B = \sum_{n=0}^{3 \cdot 2^{k-1}-1}(1-\theta)\theta^n - A = \theta^{2^{k-1}} - \theta^{3 \cdot 2^{k-1}} \quad \text{(A.11)}$$

from which we obtain

$$R_G\left(\theta, M = 3 \cdot 2^{k-1}\right) = (1-\theta)\left((k+1) + \frac{\theta^{2^{k-1}}}{1-\theta^M}\right) \quad \text{(A.12)}$$

We are now in a position to compare the average code length of a Golomb/Rice code for $M = 2^k$ (integer Rice) with a Golomb code for $M = 3\cdot 2^{(k-1)}$ (our code for *h* = 1). For the transition from Rice mode to half mode, we want

$$R_G\left(\theta, M = 2^k\right) = R_G\left(\theta, M = 3\cdot 2^{k-1}\right) \tag{A.13}$$

After straightforward algebraic manipulation, the solution for the crossover probabilities satisfying the above equation above is given by

$$\theta'_k = x_1^{2^{-(k-1)}},\ k = 1, 2, 3, \ldots \tag{A.14}$$

where $0 < x_1 < 1$ is the only solution to $x^3 + x^2 = 1$, that is x_1 = 0.754877666247. Simplifying, we calculate

$$\theta'_k = x_1^{2\cdot 2^{-k}} = (0.569840290998)^{2^{-k}},\ k = 1, 2, 3, \ldots \tag{A.15}$$

Similarly, the other set of crossover probabilities of interest is obtained by matching the rates of a Golomb code for $M = 3\cdot 2^{(k-1)}$ and a Golomb/Rice code for $M = 2^{k+1}$:

$$R_G\left(\theta, M = 3\cdot 2^{k-1}\right) = R_G\left(\theta, M = 2^{k+1}\right) \tag{A.16}$$

Using the same methodology as above, with the substitution $x = \theta^{2^{k-1}}$, we find that the transition points are

$$\theta''_k = x_2^{2^{-(k-1)}},\ k = 1, 2, 3, \ldots \tag{A.17}$$

where $0 < x_2 < 1$ is the only solution to $x^4 + x^3 = 1$, that is x_2 = 0.819172513396. As before, we can simplify the calculation as

$$\theta''_k = x_2^{2\cdot 2^{-k}} = (0.671043606704)^{2^{-k}},\ k = 1, 2, 3, \ldots \tag{A.18}$$

To determine the transitions between {*k* = 0, *h* = 0} and the special mode {*k* = 0, *h* = 1} and between the special mode and {*k* = 1, *h* = 0}, we need to compute the rate of the special mode {*k* = 0, *h* = 0} as a function of θ. It is easy to show that it is given by

$$R_G\left(\theta, k = 0, h = 1\right) = \frac{3 - \theta\left(\theta^2 - \theta + 1\right)}{2 + \theta} \tag{A.19}$$

Thus, the transition from {*k* = 0, *h* = 0} to {*k* = 0, *h* = 1} is given by the solution to

$$R_G\left(\theta, k = 0, h = 0\right) = R_G\left(\theta, k = 0, h = 1\right) \Rightarrow 1 = \frac{3 - \theta\left(\theta^2 - \theta + 1\right)}{2 + \theta} \tag{A.20}$$

which is given by $\theta'_0 = x_1^2$ = 0.569840290998. Similarly, the transition from {*k* = 0, *h* = 1} to {*k* = 1, *h* = 0} is given by the solution to

$$R_G\left(\theta,k=0,h=1\right)=R_G\left(\theta,k=1,h=0\right)\Rightarrow\frac{3-\theta\left(\theta^2-\theta+1\right)}{2+\theta}=\frac{2-\theta^2}{1+\theta}\qquad\text{(A.21)}$$

which is given by $\theta_0'' = x_2^2$ = 0.671043606704.

Finally, we note that, since the average zero run length $\bar{x}\equiv E\left[x\right]$ is related to the probability of zero θ by (1), the crossover probabilities can be mapped to corresponding crossover points for the estimated average run lengths. In practice, such crossover points can be stored in a small table, as exemplified in Table 2.

1.3247175
2.0399165
3.0795946
4.5301326
6.6240971
9.5353532
13.730626
19.558243
27.952466
39.611025
56.400539
79.717389
113.29888
159.93322

Table 2. A small list of crossover points for the estimated average run lengths.

References

[1] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *IEEE Trans. Image Processing*, vol. 9, pp. 1309–1324, Aug. 2000.

[2] D. S. Taubman and M. W. Marcellin. *JPEG2000 Image Compression, Fundamentals, Standards, and Practice.* Boston, MA: Kluwer, 2002.

[3] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits Systems for Video Technology*, vol. 13, pp. 620–636, July 2003.

[4] R. G. Gallager and D. C. van Voorhis, “Optimal source codes for geometrically distributed integer alphabets,” *IEEE Trans. Information Theory,* vol. IT-21, pp. 228–230, Mar. 1975.

[5] S. W. Golomb, “Run-length encodings,” *IEEE Trans. Information Theory,* vol. IT-12, pp. 399–401, July 1966.

[6] H. Tanaka and A. Leon-Garcia, “Efficient run-length encodings,” *IEEE Trans. Information Theory,* vol. IT-28, pp. 880–890, Nov. 1982.

[7] F. Ono, S. Kino, M. Yoshida, and T. Kimura, “Bi-level image coding with MELCODE – comparison of block type code and arithmetic type code,” *Proc. IEEE Global Telecommunications Conf. (GLOBECOM),* Dallas, TX, pp.255–260, Nov. 1989.

[8] E. Ordentlich, M. Weinberger, and G. Seroussi, “A low-complexity modeling approach for embedded coding of wavelet coefficients,” *Proc. Data Compression Conf.,* Snowbird, Utah, pp. 408–417, Mar. 1998.

[9] J. Teuhola, “A compression method for clustered bit-vectors,” *Information Processing Letters,* vol. 7, pp. 308–311, Oct. 1978.

[10] R. F. Rice, “Some practical universal noiseless coding techniques,” Tech. Report JPL-79-22, Jet Propulsion Laboratory, Pasadena, CA, 1979.

[11] H. S. Malvar, “Fast adaptive encoder for bi-level images,” *Proc. Data Compression Conf.,* Snowbird, UT, pp. 253–262, Mar. 2001.

[12] H. L. Van Trees, *Detection, Estimation, and Modulation Theory,* vol. I. New York: John Wiley and Sons, 1968.

[13] F. C. A. Oliveira and M. H. M. Costa, “Embedded DCT image coding,” *Proc. IEEE/SBrT International Telecommunications Symposium,* Natal, RN, Brazil, Sept. 2002.