Low-Complexity Lossless Codes for Image and Video Coding

Yuriy A. Reznik

Qualcomm Inc., 5775 Morehouse Dr., San Diego, CA 92121; USA

ABSTRACT

We describe design of lossless block codes for geometric, Laplacian, and similar distributions frequently arising in image and video coding. Proposed codes can be understood as a generalization of Golomb codes, allowing more precise adaptation to values of parameters of distributions, and resulting in lower redundancy. Design of universal block codes for a class of geometric distributions is also studied.

1. INTRODUCTION

The task of coding of integers distributed according to geometric

$$P(i) = (1-\theta)\,\theta^i, \quad i \in \mathbb{Z}^+, \quad \theta \in (0,1), \tag{1}$$

or double-geometric

$$P(i) = \frac{1-\theta}{1+\theta}\,\theta^{|i|}, \quad i \in \mathbb{Z}, \quad \theta \in (0,1), \tag{2}$$

distributions appears in a number of practical situations. Such integers can represent, for example, runs of appearances of the same events, residual signal values after prediction, values of transform coefficients in image or video coding, and so on.

As it was shown by S. Golomb,[1] construction of prefix codes for geometric sources (1) is remarkably simple:

- split input value i: $i = \ell j + r$, $j = \lfloor i/\ell \rfloor$, where ℓ is some given integer (code parameter);
- encode j by a unary code (i.e. transmit j zeros followed by a single one);
- encode the remainder r by a simple prefix code with codewords of length $\lfloor \log_2 \ell \rfloor$ if $r < 2^{\lfloor \log \ell \rfloor + 1} - \ell$, or $\lfloor \log_2 \ell \rfloor + 1$ otherwise.

It is known that minimal expected length of such codes is achieved by selecting ℓ as follows:[2]

$$\ell = \lceil -\log(1+\theta) / \log\theta \rceil .$$

R.F. Rice[3] has independently discovered a special case of Golomb codes with $\ell = 2^m$. Picking a dyadic number for ℓ makes the encoding/decoding process even simpler: division by ℓ is substituted by a right shift, and the remainder r can be transmitted directly by using its m least significant bits. We will refer to such codes as Golomb-Rice codes.

With simple adaptations Golomb or Golomb-Rice codes can also be used for coding of signed integers distributed according to double-geometric (2) or similarly shaped two-sided distributions.[6] Such codes have found applications in a number of data compression algorithms and standards, including JPEG-LS,[7] MPEG-4 ALS,[8] and others. However, despite their simplicity, Golomb and Golomb-Rice codes suffer from one important disadvantage: their redundancy can be relatively high.

Recall, that the *redundancy* of a code ϕ is usually defined as follows

$$R(\theta) = \sum_i P(i)|\phi(i)| - H(\theta), \tag{3}$$

Contact information: yreznik@ieee.org, phone: +1 (858) 658-1866.

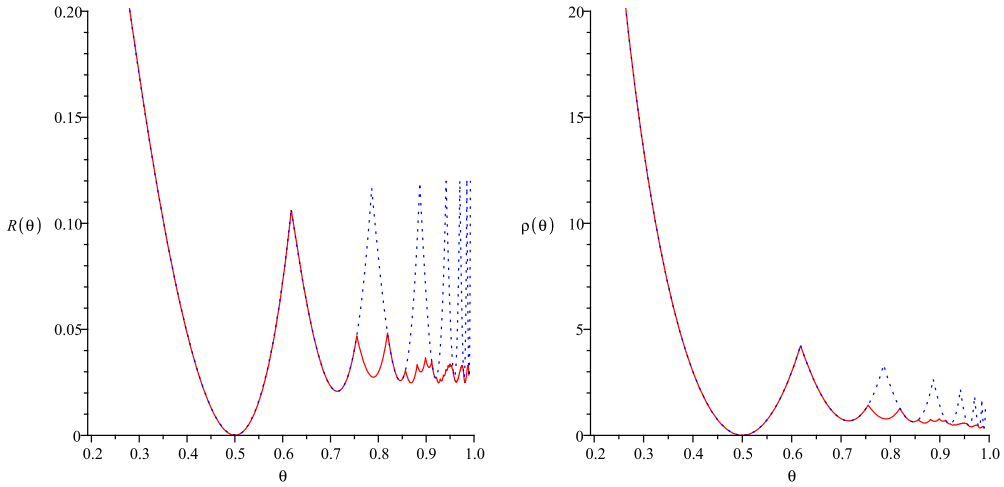

Figure 1. Redundancy $R(\theta)$ and relative redundancy $\rho(\theta) = \frac{R(\theta)}{H(\theta)} \times 100[\%]$ of Golomb codes (red, solid line) and Golomb-Rice codes (blue, dotted line), constructed for a geometric source (1).

where $|\phi(i)|$ denotes the length of an i-th code, and where

$$H(\theta) = -\sum_i P(i) \log P(i) \; = -\frac{1}{1-\theta} \left[\theta \log \theta + (1-\theta) \log(1-\theta)\right], \tag{4}$$

is the entropy of the geometric source (1). In practice, it is also convenient to assess *relative redundancy*

$$\rho(\theta) = \frac{R(\theta)}{H(\theta)} \times 100\,[\%]\,. \tag{5}$$

This quantity represents percentage of code bits that are overspent relative to the entropy limit. Indeed, the better is the code, the smaller is its redundancy.

It is an easy exercise to show that the redundancy of Golomb and Golomb-Rice codes under geometric source (1) behaves as shown in Figure 1. The discontinuities in these curves are caused by changes in code parameter ℓ, selected to achieve minimum code length under different values of θ. It can be observed, that when $\theta \to 1$, the redundancy of Golomb codes oscillates around some constant, close to 0.03 bits.[2] Golomb-Rice codes skip non-diadic values of ℓ, causing redundancy oscillations to increase. Typical redundancy range for Golomb-Rice codes is $0.03 \ldots 0.12$-bits, which, on relative scale, corresponds to about $1 \ldots 4\%$ portion of the total bitrate (see right image on Figure 1).

Much more significant is the degradation of performance of Golomb and Golomb-Rice codes in situations when parameter θ becomes small: $\theta \ll 1/2$. Thus, when $\theta \to 0$, the entropy of the source tends to 0, but the redundancy of Golomb codes tends to 1. Consequently, the relative redundancy tends to infinity. This is why, in practice, algorithms employing Golomb codes usually use additional logic to detect such cases, and change coding scheme accordingly. For example, they can flip the value of the "most probable symbol", use run-length or block grouping of symbols, etc. This complicates things.

In this paper we study design of *block codes* for geometric sources, i.e. constructing codes for n-tuples of input numbers, as opposed to encoding them one by one. A special case, when $n = 2$ has been recently considered by F. Bassino, *et. al.*,[4] but extensions to larger n has not, to the best of our knowledge, been offered yet. Even in case of $n = 2$ the construction of optimal prefix codes was shown to be rather complicated.[4] We use somewhat different approach. Instead of constructing prefix codes that are precisely optimal, we are trying to make them optimal in the asymptotic sense, assuming that block sizes can be large. This leads to a much simpler design, extendable to any n. We produce such codes for known values of model parameter θ, as well as *universal codes* for the class of geometric sources.

2. DESIGN OF BLOCK CODES FOR GEOMETRIC SOURCES

Our task is to construct codes for n-tuples of input values

$$i_1, \ldots, i_n,$$

where $n \geqslant 2$ is some fixed parameter. As customary, we will call an input n-tuple a *block*, and the resulting code a *block code*.

Assuming memoryless geometric model (1), we can express probability of a block as follows:

$$P(i_1, \ldots, i_n) = \prod_{j=1}^{n} P(i_j) = (1-\theta)^n \, \theta^S \,, \tag{6}$$

where S denotes the sum of all block values:

$$S = S(i_1, \ldots, i_n) = \sum_{j=1}^{n} i_j \,. \tag{7}$$

Hence, probability of a block is a function of the sum S of block values, and the could be many possible input blocks appearing with the same probability.

The exact number of all possible blocks with total S is given by a so-called *multiset coefficient*:[5]

$$\left(\!\binom{n}{S}\!\right) = \binom{S+n-1}{n-1} \,. \tag{8}$$

Moreover, as shown in Appendix A, we can easily compute a mapping:

$$\xi: \quad \left\{ [i_1, \ldots, i_n] : \ \forall j : i_j \geqslant 0, \ \textstyle\sum_{j=1}^{n} i_j = S \right\} \quad \longleftrightarrow \quad \left\{ 0, \ldots, \left(\!\binom{n}{S}\!\right) - 1 \right\} . \tag{9}$$

associating each block $i_1, \ldots, i_n$ in a set of all possible blocks with total S with a unique index $\xi(i_1, \ldots, i_n)$. Inverse mapping is also very easy to compute.

This means, that a code for a block $i_1, \ldots, i_n$ can be designed using two parts:

- *a variable-length code for the sum S, and*
- *a simple prefix code for an index of an input block $\xi(i_1, \ldots, i_n)$ in a set of blocks with total S.*

The prefix code to be used in the second part is a special case of a Huffman code for uniform distribution. It uses codewords of length $\left\lfloor \log_2 \left(\!\binom{n}{S}\!\right) \right\rfloor$ if $\xi < 2^{\left\lfloor \log\left(\!\binom{n}{S}\!\right) \right\rfloor + 1} - \left(\!\binom{n}{S}\!\right)$, or $\left\lfloor \log_2 \left(\!\binom{n}{S}\!\right) \right\rfloor + 1$ otherwise. Appendix B explains details of this algorithm.

The first part in our code design is much more interesting. We need to construct a variable-length code for the sum (7). We next take a look at distribution of this quantity.

2.1 Distribution of the sum S

Since our input quantities $i_1, \ldots, i_n$ are distributed according a geometric model (1), and there is a $\left(\!\binom{n}{S}\!\right)$ combinations producing total S, we can conclude that S is distributed as follows:

$$P(S) = \left(\!\binom{n}{S}\!\right) (1-\theta)^n \, \theta^S \tag{10}$$

In statistics, this distribution is usually referred to as *negative Binomial distribution*.[9] We show the shape of this distribution for several values of parameter θ in Figure 2.

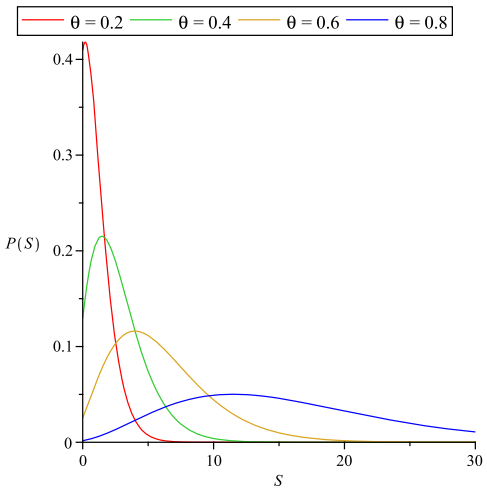

Figure 2. Negative Binomial distribution: $P(S) = \binom{n}{S} (1-\theta)^n \theta^S$. Plots are rendered for $n = 4$.

It is known, that for $n \geqslant 2$ the mode of negative Binomial distribution is

$$S^* = \left\lfloor (n-1) \frac{\theta}{1-\theta} \right\rfloor , \tag{11}$$

and that its mean is

$$\bar{S} = n \frac{\theta}{1-\theta} . \tag{12}$$

The differences between mean and mode, as well as asymmetric tails introduce some challenging for code construction. But as we will show below, much of this can be resolved by proper normalization of values S.

2.2 Encoding of the sum S in case of known source parameter θ

When parameter θ is known, we encode S as follows:

- split S into two parts:

$$S = s\ell + r , \quad r \in [0, \ell - 1] , \quad \ell = \left\lceil \frac{\theta}{1-\theta} \right\rceil ; \tag{13}$$

- encode s by using Huffman codes for the first $s < 2n$ values; or escape code followed by unary code for $s - 2n$, if $s \geqslant 2n$;
- encode the remainder r by a simple prefix code with codewords of length $\lfloor \log_2 \ell \rfloor$ if $r < 2^{\lfloor \log \ell \rfloor + 1} - \ell$, or $\lfloor \log_2 \ell \rfloor + 1$ otherwise. If $s < S^*/\ell$ values r are be replaced by $r = \ell - 1 - r$ prior to such encoding.

The reason why in the second step we allocate only $2n$ entries to Huffman codes, is that the mean of distribution is now

$$\bar{s} = \bar{S}/\ell \leqslant n , \tag{14}$$

implying, that the remaining values are monotonically decaying.

The use of unary codes for large values s is justified by an observation that (for large S)

$$-\log P(S) \sim -S \log \theta \tag{15}$$

and that (cf. [9, Equation 4.1.27]):

$$-\log \theta \sim 2\frac{1-\theta}{1+\theta} = \frac{1}{\frac{\theta}{1-\theta} + \frac{1}{2}} \sim \frac{1}{\ell} . \tag{16}$$

As easily observed, such codes can be constructed in $O(n)$ space.

2.3 Design of a universal code

When parameter θ is not known, one way to design a universal code for S is to consider so-called maximum likelihood density:[13]

$$Q(S) = \frac{\sup_{\theta\in[0,1)} P(S)}{\sum_{s\in[0,S_{\max}]} \sup_{\theta\in[0,1)} P(s)}\,, \tag{17}$$

where $S_{\max}$ defines the maximum value that we allow S to reach.

We immediately notice, that

$$\sup_{\theta\in[0,1)} P(S) = P(S)|_{\theta=\frac{S}{S+n}} = \left(\!\binom{n}{S}\!\right) \frac{n^n S^S}{(S+n)^{S+n}} \tag{18}$$

and therefore:

$$Q(S) = \frac{\left(\!\binom{n}{S}\!\right) \frac{n^n S^S}{(S+n)^{S+n}}}{\sum_{s\in[0,S_{\max}]} \left(\!\binom{n}{s}\!\right) \frac{n^n s^s}{(s+n)^{s+n}}}\,. \tag{19}$$

Most importantly, we also notice that for large S:

$$\left(\!\binom{n}{S}\!\right) \frac{n^n S^S}{(S+n)^{S+n}} \sim \frac{\left(\frac{n}{e}\right)^n}{(n-1)!}\, S^{-1} \sim \sqrt{\frac{n}{2\pi}}\, S^{-1}\,, \tag{20}$$

which means, that the length of an ideal minimax code for S shall grow as

$$-\log Q(S) \sim \log S. \tag{21}$$

This implies that universal codes should have asymptotically logarithmic lengths with S. Well known monotonic prefix codes that exhibit similar behavior include Levenstein code,[15] Elias-ω code,[16] Bentley-Yao code,[17] etc. For simplicity, we will use Levenshtein code for S in our design of a universal code. Such code can be easily constructed in-place.[15]

We must note, that in the unconstrained case ($S_{\max} = \infty$) we can't rely on the above logic to suggest that the code is asymptotically optimal. In this case, the sum in the denominator in (17) does not converge. However, the proposed code can still be constructed, and in the next section we will show that its redundancy comes pretty close to $\frac{1}{2}\log n$ term that one can expect for a universal block code with single unknown parameter.[11]

3. REDUNDANCY ANALYSIS

Recall, that given a code $\phi(i_1,\ldots,i_n)$ for a block of input quantities $i_1,\ldots,i_n$, its redundancy is usually computed as follows

$$R(\theta,n) = \frac{1}{n}\sum_{i_1,\ldots,i_n} P(i_1,\ldots,i_n)\,|\phi(i_1,\ldots,i_n)| - H(\theta)\,, \tag{22}$$

where $|.|$ denotes the length of a code, and $H(\theta)$ is the entropy of the input process (4). The factor $\frac{1}{n}$ is used to convert average code length for a block of n symbols to a per-symbol quantity.

Since probability of a block (6) is actually only a function of the sum S, we can replace enumeration of block values $i_1,\ldots,i_n$ in (22) with sum over S. The sum probabilities next to sum become:

$$P(i_1,\ldots,i_n)\left(\!\binom{n}{S}\!\right) = P(S)\,. \tag{23}$$

Recall now that our code consists of two parts:

$$\phi(i_1,\ldots,i_n) = \phi_1(S)\;\;\phi_2(\xi(i_1,\ldots,i_n))$$

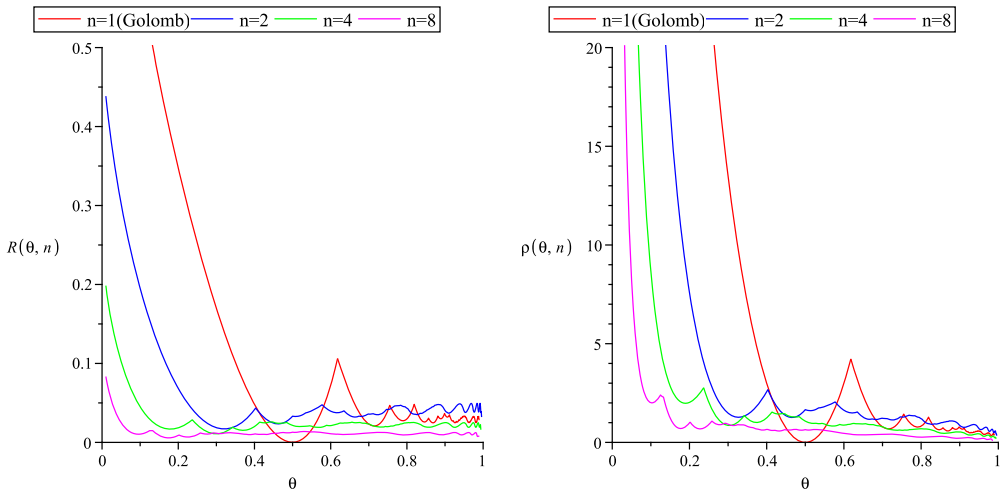

Figure 3. Redundancy $R(\theta, n)$ and relative redundancy $\rho(\theta, n) = \frac{R(\theta,n)}{H(\theta)} \times 100[\%]$ of Golomb codes ($n = 1$), and our proposed block codes ($n = 2, 4, 8$) constructed for different values of distribution parameter θ. It is shown that redundancy of our codes becomes smaller as block size n increases.

where variable length code $\phi_1(.)$ is used to transmit total S, and code $\phi_2(.)$ is used to transmit index of a corresponding combination of values $i_1, \ldots, i_n$. The length of code ϕ_2 satisfies: $|\phi_2(\xi)| \leqslant \log_2 \left(\binom{n}{S}\right) + 1$.

By putting everything together we can now show that

$$\begin{aligned} R(\theta, n) &\leqslant \frac{1}{n} \sum_S P(S) \left[|\phi_1(S)| + \log_2 \left(\binom{n}{S}\right) + 1 \right] - H(\theta) \\ &= \frac{1}{n} \left[\mathbf{E} |\phi_1(S)| + \sum_S P(S) \log_2 \left(\binom{n}{S}\right) + 1 \right] - H(\theta), \end{aligned} \tag{24}$$

where $\mathbf{E}|\phi_1(S)|$ denotes expected length of a code for the sum parameter S.

Now, if the source (i.e. parameter θ) is known, and we use Huffman code to encode S, then we can say that

$$\mathbf{E} |\phi_1(S)| \leqslant H(S) + O(1), \tag{25}$$

where

$$\begin{aligned} H(S) &= -\sum_S P(S) \log P(S) \\ &= -\sum_S P(S) \left[\log_2 \left(\binom{n}{S}\right) + n \log_2(1-\theta) + S \log_2 \theta \right] \\ &= -n \log_2(1-\theta) - \bar{S} \log_2 \theta - \sum_S P(S) \log_2 \left(\binom{n}{S}\right) \\ &= nH(\theta) - \sum_S P(S) \log_2 \left(\binom{n}{S}\right), \end{aligned} \tag{26}$$

is the entropy of the negative binomial distribution (10).

By combining (26), (25) and (24) we obtain:

$$R(\theta, n) = O\left(\frac{1}{n}\right), \tag{27}$$

which is the standard rate of convergence of a block code for a known source.[12]

We show exact redundancy plots for $n = 2, 4, 8$ constructed by using our codes in Figure 3. Same figure also includes plots obtained for classic Golomb codes. It can be observed, that our proposed block codes can perform significantly better than classic Golomb codes, particularly in the region $\theta \ll 1/2$.

3.1 Redundancy of universal code

When we design a code for unknown parameter, we simply replace ϕ_1 with the Levenshtein code.[15] This is a simple monotonic prefix code for integers, with lengths that can be approximately characterized as:

$$|\mathrm{Lev}(S)| = \log_2(1+S) + \log_2\log_2(2+S)(1+o(1))\,. \tag{28}$$

In our subsequent analysis we are dealing with several sums, that can be generally described as $\sum_S P(S)\, f(S)$, where $f(.)$ is some continuous function of S. Rigorous analysis of such sums is actually a rather complicated matter, and the reader is referred to references[18–20] describing several available techniques. Instead, here we will only guess values of 1-st terms in asymptotic expressions, by relying on concentration property:

$$\sum_S P(S)\, f(S) \sim f(S^*)$$

where S^* is the mode of the distribution.

This produces the following:

$$\begin{aligned}
R_{\mathrm{universal}}(\theta, n) &\leqslant \frac{1}{n}\sum_S P(S)\left[|\mathrm{Lev}(S)| + \log\left(\binom{n}{S}\right) + 1\right] - H(\theta) \\
&= \frac{1}{n}\sum_S P(S)\left[\log_2(1+S) + \log_2\log_2(2+S)(1+o(1)) + \log_2\left(\binom{n}{S}\right) + 1\right] - H(\theta) \\
&\sim \frac{1}{n}\left[\log_2(1+S^*) + \log_2\log_2(2+S^*)(1+o(1)) + \log_2\left(\binom{n}{S^*}\right) + 1\right] - H(\theta)
\end{aligned}$$

Next, by assuming that n is large, with the help of MAPLE, we can show that:

$$\begin{aligned}
\log_2(1+S^*) &= \log_2 n + \log_2\frac{\theta}{1-\theta} + O\left(\frac{1}{n}\right), \\
\log_2\log_2(2+S^*) &= \log_2\log n + O\left(\frac{1}{\log n}\right), \\
\log_2\left(\binom{n}{S^*}\right) &= H(\theta)\, n - \frac{1}{2}\log_2 n - H(\theta) - \frac{1}{2}\log_2(2\pi\theta) + O\left(\frac{1}{n}\right).
\end{aligned}$$

By combining all these terms together we finally obtain:

$$R_{\mathrm{universal}}(\theta, n) \lesssim \frac{1}{n}\left[\frac{1}{2}\log_2 n + \log_2\log n(1+o(1))\right]. \tag{29}$$

This expression shows the redundancy of such code decays as block length n increases. That is, this code is universal. It is also important to note that the leading factor in our redundancy estimate $\frac{1}{2n}\log n$ matches best one achievable for sources with single unknown parameter.[11,14]

4. CONCLUSIONS

A design of block codes for a class of geometric distributions is proposed. This includes codes for known source parameter, and universal codes. Proposed codes are very simple to construct, and achieve optimal performance in the asymptotic sense (with large blocks).

REFERENCES

[1] S. W. Golomb, "Run-Length Encodings," *IEEE Trans. Inform. Theory*, **12** (7) 399–401 (1966).

[2] R. G. Gallager, and D.C. Van Voorhis, "Optimal Source Codes for Geometrically Distributed Integer Alphabets," *IEEE Trans. Inform. Theory*, **21** (3) 228–230 (1975).

[3] R. F. Rice, "Some practical universal noiseless coding techniques," *Tech. Rep. JPL-79-22*, Jet Propulsion Laboratory, Pasadena, CA (1979).

[4] F. Bassino, J. Clément, G. Seroussi, A. Viola, "Optimal Prefix Codes for Some Families of Two-Dimensional Geometric Distributions," *Proc Data Compression Conference*, 113–122 (2006).

[5] D.E. Knuth, [*The Art of Computer Programming Vol. 2: Seminumerical Algorithms (3rd edition)*], Addison Wesley, (1998).

[6] R.F. Rice, and J.R. Plaunt, "Adaptive variable length coding for efficient compression of spacecraft television data," *IEEE Trans. Comm. Tech.*, **19** (1) 889-897 (1971).

[7] M. J. Weinberger, G. Seroussi, and G. Sapiro, "LOCO-I: A Low Complexity, Context-Based, Lossless Image Compression Algorithm," *Proc. Data Compression Conference*, 140–149 (1996).

[8] T. Liebchen, and Y. A. Reznik, "MPEG-4 ALS: an Emerging Standard for Lossless Audio Coding," *Proc. Data Compression Conference*, 439–448 (2004).

[9] M. Abramowitz and I. Stegun, [*Handbook of Mathematical Functions*], Dover, New York (1972).

[10] T. M. Cover and J. M. Thomas, [*Elements of Information Theory*], John Wiley & Sons, New York (1991).

[11] T. S. Han, and K. Kobayashi, [*Mathematics of Information and Coding*], American Mathematical Society, Boston (2001).

[12] R. E. Krichevsky and V. K. Trofimov, "The Performance of Universal Encoding," *IEEE Trans. Information Theory*, **27** (1981) 199–207.

[13] Y. Shtarkov, "Universal Sequential Coding of Single Messages," *Problems of Information Transmission*, **23** 175–186 (1987). (in Russian)

[14] J. Rissanen, "Universal coding, information, prediction and estimation," *IEEE Trans. Inform. Theory*, **30** 629-636 (1984).

[15] V. I. Levenshtein, "Redundancy and delay of recoverable coding of natural numbers," *Probl. Cybern.*, **20**, 173-179 (1968). (in Russian)

[16] P. Elias, "Universal Codeword Sets and Representations of the Integers," *IEEE Trans. Inform. Theory*, **21** (2) 194-203 (1975).

[17] J. L. Bentley and A. C. Yao, "An almost optimal algorithm for unbounded searching," *Inform. Processing Lett.*, **5** (3) 82-87 (1976).

[18] P. Flajolet, "Singularity analysis and asymptotics of Bernoulli sums," *Theoretical Computer Science*, 215, 371–381 (1999).

[19] P. Flajolet and A. Odlyzko, "Singularity analysis of generating functions," *SIAM J. Discrete Math.*, 3 (2) 216–240 (1990).

[20] P. Jacquet, and W. Szpankowski, "Entropy computations via analytic depoissonization," *IEEE Trans. Information Theory*, **45** (4) 1072–1081 (1999).

APPENDIX A. ENUMERATION OF N-TUPLES WITH TOTAL S

Consider blocks of n non-negative integers $i_1, \ldots, i_n$ such that

$$\sum_{j=1}^{n} i_j = S,$$

where S is some given constant. It is known, that the total number of such possible blocks is given by a *multiset coefficient*:[5]

$$\left(\!\binom{n}{S}\!\right) = \binom{S+n-1}{n-1},$$

Our task is to compute a mapping

$$\xi: \quad \left\{ i_1, \ldots, i_n : \ \forall j : i_j \geqslant 0, \ \sum_{j=1}^{n} i_j = S \right\} \quad \longleftrightarrow \quad \left\{ 0, \ldots, \left(\!\!\binom{n}{S}\!\!\right) - 1 \right\} .$$

associating each block $i_1, \ldots, i_n$ in a set of all possible blocks with total S with a unique index $\xi(i_1, \ldots, i_n)$.

By induction (starting with $n = 1, 2, 3, \ldots$), we can show that such index can be computed as follows:

$$\xi(i_1, i_2, \ldots, i_n) = \sum_{j=1}^{n-2} \sum_{k=0}^{i_j - 1} \left(\!\!\binom{n-j}{S-k-\sum_{\ell=1}^{j-1} i_\ell}\!\!\right) + i_{n-1} . \tag{30}$$

With precomputed array of multiset coefficients, formula (30) can be easily implemented by using only addition operations. Specifically, $S - 1$ additions are needed.

Alternatively, with small block sizes, this formula can also be unrolled and implemented as follows:

$$\begin{aligned}
\xi(i_1, i_2) &= i_1 , \\
\xi(i_1, i_2, i_3) &= \frac{(i_1 + 1)\,(2S - i_1 + 2)}{2} + i_2 , \\
\xi(i_1, i_2, i_3, i_4) &= \frac{(i_1 + 1)\,(3S\,(S + 3 - i_1) + i_1\,(i_1 - 4) + 6)}{6} + \frac{(i_2 + 1)(2(S - i_1) - i_2 + 2)}{2} + i_3 ,
\end{aligned}$$

and so on. Few multiplications are needed in such cases.

APPENDIX B. SIMPLE PREFIX CODES FOR UNIFORM DISTRIBUTIONS

In our design we often use codes for quantities x that are uniformly distributed in some known range, for example $x \in [0, m - 1]$, where m is not necessarily a dyadic number. Let, specifically $m = 2^\ell + r$, where $0 < r < 2^\ell$.

Such codes are constructed as follows:

$$\mathrm{U}(x, m) = \left[\begin{array}{ll} \mathrm{Bin}_\ell(x) , & \text{if} \quad x < 2^{\ell+1} - m , \\ \mathrm{Bin}_{\ell+1}(2^{\ell+1} - m + x) , & \text{if} \quad x \geqslant 2^{\ell+1} - m . \end{array} \right.$$

where $\mathrm{Bin}_\ell(x)$ denotes binary transmission of ℓ least significant bits of x.

Only a couple of operations are needed to perform encoding/decoding.