Prefix Codes for Power Laws

Michael B. Baer
23 N. Ellsworth Ave., Apt. 8
San Mateo, California USA
Email: calbear@ieee.org

***Abstract*— In prefix coding over an infinite alphabet, methods that consider specific distributions generally consider those that decline more quickly than a power law (e.g., a geometric distribution for Golomb coding). Particular power-law distributions, however, model many random variables encountered in practice. Estimates of expected number of bits per input symbol approximate compression performance of such random variables and can thus be used in comparing such methods. This paper introduces a family of prefix codes with an eye towards near-optimal coding of known distributions, precisely estimating compression performance for well-known probability distributions using these new codes and using previously known prefix codes. One application of these near-optimal codes is an improved representation of rational numbers.**

I. Introduction

Consider discrete power-law distributions, those of the form

$$p(i) \sim ci^{-\alpha} \tag{1}$$

for constants $c > 0$ and $\alpha > 1$, where $p(i)$ is the probability of symbol i, and $f(i) \sim g(i)$ implies that the ratio of the two functions goes to 1 with increasing i. Such distributions could be either inherently discrete or discretized versions of continuous power-law distributions.

Several researchers in varied fields have, in classic papers ranging from decades to centuries old, observed power-law behavior for various discrete phenomena, from continued fractions [1], [2] to Internet connections [3], [4]. Several recent expositions survey this [3], [5], [6].

Exponential-Golomb codes [7] (generalizations of Elias' γ code [8]) are a good fit for certain power laws [9], leading to their widespread use in compressing video and numerical data [9], [10]. However, there are few specific infinite-cardinality power-law distributions that have been used to judge compression performance of prefix codes. Only the Gauss-Kuzmin distribution is considered in [11], [12], while the more recent [4] analyzes zeta distributions for which $\alpha \in (1, 2]$ in (1).

Here we propose simple codes which not only improve upon existing codes for encoding symbols distributed according to the Gauss-Kuzmin distribution — which applies to representing rational numbers using continued fractions — but also efficiently code other common distributions, such as the zeta distribution with parameter 2 [13], [14]. We precisely estimate compression performance for dozens of code/distribution combinations, concentrating on power laws of the form (1) with $\alpha \in [2, 3]$, which, [5] notes, most power-law distributions occurring in nature satisfy. However, [5] itself includes phenomena with α as low as 1.8 and as high as 3.51, and other papers, such as [15], find Internet phenomena conforming to α as low as 1.2. We therefore analyze some codes lying outside $\alpha \in [2, 3]$ as well.

II. Background, formalization, and motivation

The most common infinite-alphabet codes are codes that are optimal for geometric [16], [17] and geometrically based [18]–[22] distributions. For geometric distributions, these are known as Golomb codes, and are based on the *unary code* — ones terminated by a zero, i.e., a code consisting of codewords the form $\{1^j 0\}$ for $j \geq 0$. In a Golomb code (Gk), a unary code prefix precedes a binary code suffix. This binary suffix is a *complete binary code*, in that it has (k) codewords of the same length or length differing by at most one (the first $2^{\lceil \lg k \rceil} - k$ items having length $\lfloor \lg k \rfloor$ and the last $2k - 2^{\lceil \lg k \rceil}$ items having length $\lceil \lg k \rceil$). For example, the order-preserving complete binary code of size three which is monotonically nonincreasing in length is $\{0, 10, 11\}$, so the Golomb code G3 is $\{0\mathbf{0}, 0\mathbf{10}, 0\mathbf{11}, 10\mathbf{0}, 10\mathbf{10}, \ldots\}$ (complete suffix in boldface). Codes that exhibit an efficient coding rate for infinite-support power laws, by contrast, are not known to be optimal (excepting trivial examples for dyadic probability mass functions).

We restrict ourselves to binary codes and assume without loss of generality that the infinite-alphabet source emits symbols drawn from the alphabet $\mathcal{X} = \{1, 2, 3, \ldots\}$. Symbol i has probability $p(i) > 0$, forming probability mass function $P = \{p(i)\}$. The source symbols are coded into binary codewords. The codeword $c(i) \in \{0, 1\}^*$, corresponding to symbol i, has length $n(i) \in \mathbb{Z}_+$, thus defining length distribution $N = \{n(i)\}$. An optimal code minimizes $\sum_{i \in \mathcal{X}} p(i)n(i)$ with the constraint of being uniquely decodable; thus we consider two codes with identical lengths equivalent. N corresponds to at least one such code if and only if the Kraft inequality, $\sum_{i \in \mathcal{X}} 2^{-n(i)} \leq 1$, is satisfied. We assume without loss of generality that these codes are prefix codes, that is, codes where there are no two codewords of the form $c(i)$ and $c(j) = c(i)x$, where $c(i)x$ denotes the concatenation of strings $c(i)$ and (nontrivial) x. (In a similar use of notation, 0^k and 1^k denote k 0's and k 1's, respectively. Also, $\lg$ denotes $\log_2$ and $\ln$ denotes $\log_e$, the natural logarithm.)

One cannot use the Huffman source coding algorithm [23] to find an optimal code, as one can for a finite source alphabet. However, it is sensible that a code over the integers should be *monotonic*, that is, that $n(i) \leq n(i+1)$ for all $i \geq 0$. An exchange argument easily shows that this is necessary for the code to be optimal given a distribution for which $p(i) > p(i+1)$ for all i.

Another desirable property is one we call "smoothness":

Definition: We call $N = \{n(i)\}$ *j-smooth* if, for every $i > j$, if $n(i+1) = n(i+2)$, then $n(i+1) - n(i) \leq 1$, that is, the sequence of codeword lengths has no "gaps" (where $n(i)$ and $n(i+1)$ differ by more than 1) followed by "plateaus" (with multiple codewords — $n(i+1)$ and $n(i+2)$ — at the same length); *weakly smooth* means that it is j-smooth for some j. Thus, for any j, a j-smooth code includes all weakly smooth codes. Similarly, 0-smooth (or *strongly smooth*) codes include all j-smooth (and thus weakly smooth) codes. Also, we call a $P = \{p(i)\}$ *j-antiunary* if, for every $i > j$, $p(i) < p(i+1) + p(i+2)$; *antiunary* means that it is j-antiunary for some j.

Observation: No j-antiunary distribution has an optimal code which is not j-smooth. Thus no antiunary distribution has an optimal code which is not weakly smooth.

Proof: Suppose a j-antiunary distribution P has an optimal code with lengths N which is not j-smooth. Then there exists an $i > j$ such that $n(i+1) = n(i+2)$ and $n(i+1) - n(i) > 1$. Consider $N' = \{n'(i)\}$ for which $n'(k) = n(k)$ except at values $n'(i) = n(i) + 1$, $n'(i+1) = n(i+1) - 1$, and $n'(i+2) = n(i+2) - 1$. Clearly N' satisfies the Kraft inequality and $\sum_i p(i)n'(i) < \sum_i p(i)n(i)$, so N is not optimal. ■

Every power law is antiunary, but most previously proposed codes suitable for power-law distributions are not weakly smooth, so they could not be optimal solutions, and it is always a simple matter to improve such codes for use with such distributions. This gives us reason to believe that, among suboptimal codes, (weakly and/or strongly) smooth codes might be better suited to these distributions than those that are not smooth, something that is empirically confirmed in the remainder of this paper.

We discuss suboptimal codes because, although optimal codes always exist [24], there is no guarantee that optimal codes would be computationally tractable, let alone computationally practical for compression applications. We thus judge performance of candidate codes by expected number of bits per coded symbol rather than by strict optimality. One of the contributions of this paper is a comparison of various codes for well-known power-law distributions.

III. A NEW FAMILY OF CODES FOR INTEGERS

We propose a family of monotonic, computational efficient, 0-smooth codes, starting with the code shown in the center set of columns ($n_0(\cdot)$ and $c_0(\cdot)$) of Table I, which is defined as

$$c_0(i) = \begin{cases} 0b(i-1,3), & i < 4 \\ 1c_0\left(\frac{i-2}{2}\right)0, & i = \{4,6,8,\ldots\} \\ 1c_0\left(\frac{i-3}{2}\right)1, & i = \{5,7,9,\ldots\}. \end{cases}$$

The term $b(j,k)$ denotes the $(j+1)$th codeword of a complete binary code with k items, e.g., $b(\cdot,3) = \{0,10,11\}$. Thus, for example, $c_0(12) = 1c_0(5)0 = 11c_0(1)10 = 110010$. This is thus straightforward to encode, decode, and write in the form of an implicit infinite coding tree.

This code, like exponential-Golomb codes, is a modification of the γ code. Whereas the γ code has an m-bit unary code followed by a complete binary code for 2^{m-1} items, Code 0 follows the unary prefix by a complete binary suffix for $3 \cdot 2^{m-1}$ items. This assures not only its monotonicity, but also its 0-smoothness, due to the complete binary suffix being of variable length. This and its similarity to the γ code, which is not smooth, means that the code is especially suitable for power laws.

Straightforward extensions of this can be obtained by modifying the coding tree. We can add a k-bit binary number to each possible codeword — as in the fourth and fifth set of columns in Table I — extending Code 0 by adding a fixed-length suffix in the same manner as exponential-Golomb codes, that is,

$$c_k(i) = c_0\left(1 + \left\lfloor \frac{i-1}{2^k} \right\rfloor\right) b((i-1) \bmod 2^k, 2^k)$$

where $k > 0$ and $b\left((i-1) \bmod 2^k, 2^k\right)$ is the k-bit representation of $(i-1) \bmod 2^k$. Call any of the new extensions Code k.

Another extension, similar to [9] and [25], involves first coding with a finite code tree, then, if this initial codeword is all 1's, adding Code 0. If we start as in a unary code and switch to Code 0 after κ ones, then let Code $-\kappa$ denote the implied code, e.g., Code -1, the second set of columns ($n_{-1}(\cdot)$ and $c_{-1}(\cdot)$) in Table I. Formally, for $k = -\kappa < 0$,

$$c_k(i) = \begin{cases} 1^{i-1}0, & i \leq -k \\ 1^{(-k)}c_0(i+k), & i > -k. \end{cases}$$

All codes presented here are 0-smooth, and can be coded and decoded using only additions, subtractions, and shifts such that the total number of operations is proportional to the number of encoded output bits.

Before comparing the performance of these codes to extant codes, we should first be sure we are making a fair comparison. For example, the "negative" codes ($k < 0$) previously introduced have an analogue in exponential-Golomb codes, and, indeed, at least one of these codes is used in the H.264 video compression standard [9]. Therefore, for the sake of comparison, we define exponential-Golomb codes with parameter $k < 0$ to be defined as above; that is,

$$c_{\mathrm{EG}k}(i) = \begin{cases} 1^{i-1}0, & i \leq -k \\ 1^{(-k)}c_{\mathrm{EG}0}(i+k), & i > -k. \end{cases}$$

IV. APPLICATION

Table II lists various distributions for which no optimal code is known and estimates, in expected number of bits per input symbol, of coding performance using several different codes. Values in the table are shown to the calculated precision, and values that are exactly calculated from infinite sums, rather than estimated, are indicated by the reduced number of figures (for integers) or through ellipses for $5/3$, $\zeta(1.5)/\zeta(2.5)$, and $\zeta(2)/\zeta(3)$. The entropy and the expected number of bits per

	Code -2		Code -1		Code 0		Code 1		Code 2	
i	$n_{-2}(i)$	$c_{-2}(i)$	$n_{-1}(i)$	$c_{-1}(i)$	$n_0(i)$	$c_0(i)$	$n_1(i)$	$c_1(i)$	$n_2(i)$	$c_2(i)$
1	1	0	1	0	2	0 **0**	3	0 **0** 0	4	0 **0** 00
2	2	10	3	1 0 **0**	3	0 **1** 0	3	0 **0** 1	4	0 **0** 01
3	4	11 0 **0**	4	1 0 **1** 0	3	0 **1** 1	4	0 **1** 0 0	4	0 **0** 10
4	5	11 0 **1** 0	4	1 0 **1** 1	4	10 **0** 0	4	0 **1** 0 1	4	0 **0** 11
5	5	11 0 **1** 1	5	1 10 **0** 0	4	10 **0** 1	4	0 **1** 1 0	5	0 **1** 0 00
6	6	11 10 **0** 0	5	1 10 **0** 1	5	10 **1** 00	4	0 **1** 1 1	5	0 **1** 0 01
7	6	11 10 **0** 1	6	1 10 **1** 00	5	10 **1** 01	5	10 **0** 0 0	5	0 **1** 0 10
8	7	11 10 **1** 00	6	1 10 **1** 01	5	10 **1** 10	5	10 **0** 0 1	5	0 **1** 0 11
9	7	11 10 **1** 01	6	1 10 **1** 10	5	10 **1** 11	5	10 **0** 1 0	5	0 **1** 1 00

TABLE I

FIVE OF THE CODES INTRODUCED HERE

		H	N^* (estimate)	Golin	Code k	л	EGk/ζ/δ/ω	Y	Gk
Gauss-Kuzmin, $\alpha = 2$		3.43253	3.47207	$3.50705^{(1,2)}$	$\mathbf{3.472346}^{(-1)}$	3.77915	$3.50705^{(0)}$	3.48765	$\infty^{(\forall k)}$
Yule-Simon $\alpha = \rho + 1$	$\rho = 1$	2.95215	2.98136	$3.^{(1,2)}$	$2.983338^{(-1)}$	3.17826	$3.^{(0)}$	$\mathbf{2.98138}$	$\infty^{(\forall k)}$
	$\rho = 1.5$	2.17073	2.21571	$\mathit{2.22507}^{(1)}$	$\mathbf{2.230792}^{(-2)}$	2.32233	$2.23222^{(-1)}$	2.26031	$2.85003^{(3)}$
	$\rho = 2$	1.74685	1.83787	$\mathit{1.84024}^{(1)}$	$\mathbf{1.848484}^{(-4)}$	1.91747	$1.84788^{(-1)}$	1.92361	$2.^{(1)}$
	$\rho = 2.5$	1.47629	1.62102	$\mathit{1.62191}^{(1)}$	$\mathbf{1.626668}^{(-5)}$	1.68947	$1.63115^{(-2)}$	1.73044	$2.66666\ldots^{(1)}$
zeta $\alpha = s$	$s = 1.6$	3.93017	3.95	$4.0427^{(1)}$	$\mathbf{3.995727}^{(-1)}$	4.32479	$4.06504^{(0)}$	4.05307	$\infty^{(\forall k)}$
	$s = 1.75$	3.17604	3.1938	$3.2331^{(1)}$	$\mathbf{3.199677}^{(-1)}$	3.42948	$3.23385^{(0)}$	3.21909	$\infty^{(\forall k)}$
	$s = 2$	2.36259	2.41766	$2.43310^{(1)}$	$\mathbf{2.417772}^{(-2)}$	2.53468	$2.43310^{(-1)}$	2.43042	$\infty^{(\forall k)}$
	$s = 2.5$	1.46525	1.65431	$\mathit{1.65767}^{(1)}$	$\mathbf{1.658015}^{(-4)}$	1.70907	$1.65943^{(-2)}$	1.71963	$1.94737\ldots^{(1)}$
	$s = 3$	0.97887	1.33453	$\mathit{1.33504}^{(1)}$	$1.336680^{(-4)}$	1.36956	$\mathbf{1.33656}^{(-3)}$	1.41389	$1.36843\ldots^{(1)}$
		entropy	estimated / ad hoc codes		new codes	previously known codes			

TABLE II

COMPRESSION (IN BITS PER SYMBOL) AND CODE PARAMETER (WHERE APPLICABLE)

symbol of an (unknown) optimal code are also estimated, the latter based on suboptimal codes. The Appendix explains the methods by which the estimates are calculated. H denotes the entropy of the distribution ($H(P) = -\sum_i p(i) \lg p(i)$) and N^* (the expected codeword length of) the optimal code. Golin denotes the best Golin code [26]; Code k denotes the best of the codes introduced here; л denotes the Levenshtein (Левенштейн) code [27]; EGk/ζ/δ/ω denotes the best of the exponential-Golomb codes [7], the extensions to the exponential-Golomb codes in the previous section, the Elias codes [8], and the ζ codes [4], where codes in the exponential-Golomb family are indicated by the code number (e.g., EG0, Elias' γ code, by 0); Y denotes Yokoo's code for the Gauss-Kuzmin distribution [12]; and Gk denotes the best Golomb code (with parameter k) [16]. These codes are defined in the cited papers. In cases for which there are multiple codes and/or parameters, the best one is chosen and indicated in superscript.

In Table II, the best code among previously known codes and the new codes is in bold, and, if a Golin code is better, this is in italics. Note that Golin codes do well for inputs with rapidly declining probabilities, whereas Yokoo's code and the codes introduced here have the best results for heavier tails. (Like the codes here, Yokoo's code can be viewed as a "smoothed" version of the exponential-Golomb codes.) However, Golin codes, in being calculated on the fly, are often impractical, both due to the potential for rounding errors to lead to coding errors and due to the computational complexity of the required floating point divisions. Note also that ζ codes are never superior for these distributions, as they were designed for α (in (1)) in the $[1.06, 1.57]$ range, where fewer practical distributions arise. On a related note, while the ζ code ζ_2 is superior to γ (EG0) for zeta codes with $s = \alpha \in [1.27, 1.57]$, Code -1 is best for $s = 1.53$ (not shown in the above table). For this distribution, Code -1 averages 4.537 bits per symbol input while ζ_2 averages 4.539 bits. Similar results occur for slightly higher s; the table includes examples of higher s values.

Code -1 is of particular interest as it happens to be an excellent code for the Gauss-Kuzmin distribution, defined (and well-approximated) as follows:

$$p^{\text{GK}}(i) \triangleq -\lg\left[1 - \frac{1}{(i+1)^2}\right] \approx \frac{\lg e}{(i+1)^2}$$

The Gauss-Kuzmin distribution is the one for which to code when expressing coefficients of continued fractions, as in [11], [28], in which EG0 is proposed for use, and [12], in which Yokoo's code is proposed. Code -1 is only about 0.008% worse than the (approximated) optimal code, whereas Yokoo's code is 0.449% worse and the Elias γ code (EG0) is 1.007% worse. Each of these codes provides a method for expressing rational numbers without round-off as their continued fraction representation; see [11], [12] for details, which are omitted here for space. This representation is *alphabetic* or *order preserving* if the code is; a code is order preserving if $c(i, j)$ is the jth bit of the ith codeword, then $c(i + 1, j) < c(i, j)$ only if there is a $k < j$ such that $c(i + 1, k) \neq c(i, k)$. The codes presented here are order preserving due to the unary prefix and complete binary suffix being monotonic and order preserving in Code 0; the order preservation of other codes follows. Code -1 thus provides an improved representation over prior codes.

Note also that Code -2 is a good code for the zeta distribution with parameter $s = \alpha = 2$, where the zeta distribution is defined as

$$p_s^\zeta(i) \triangleq \frac{1}{i^s \zeta(s)}$$

and ζ is the Riemann zeta function $\zeta(s) \triangleq \sum_{i=1}^{\infty} i^{-s}$ for $s > 1$. The zeta distribution is used to model several phenomena including language [29] and Internet phenomena [15]. Optimal codes for the zeta 2 distribution ($s = 2$) were considered in Kato's unpublished manuscript [14], in which the optimal codeword lengths for the first ten symbols are shown to lie in ranges of two possible values for each codeword (or one for the first, which has $n(1) = 1$). The codeword lengths of Code -2 all lie within the allowed ranges. However, we can empirically find better codes, showing that Code -2, although the best simply described code we know of, is about 0.005% worse than an optimal code.

A third distribution family is that of Yule [30] and Simon [31],

$$p_\rho^{\mathrm{YS}}(i) \triangleq \rho B(i, \rho+1) \qquad \left(p_\rho^{\mathrm{YS}}(i) = \rho \frac{(i-1)!\rho!}{(\rho+i)!}\right)$$

where $B(i,j)$ is the beta function, $\rho = \alpha - 1 > 0$, and the right equation applies for integer ρ. Thus, for example, if $\rho = 1$, then $p(i) = 1/i(i+1)$. Several statistics, from species population to word frequencies, have been observed to obey a Yule-Simon distribution, most often with parameter $\rho = 1$ [31]. This particular distribution is also related to continued fractions, being the distribution of the first coefficient of a continued fraction of a number chosen uniformly over the unit interval $(0,1)$. For P_1^{YS}, Yokoo's code is 0.066% better than Code -1, and a mere 0.0007% worse than an optimal code.

As in many previous papers on these and similar codes [7], [32], the best code is chosen by its empirical performance; as with exponential-Golomb codes, there appears to be no simple, accurate, analytically derived rule for deciding which code to use.

We find that the codes introduced here do quite well, only failing to improve upon previously known codes in one case with $\alpha \in [2,3)$ — α as in (1) — the Yule-Simon distribution with parameter $\rho = 1$ ($p(i) = 1/i(i+1)$). Because Yokoo's code, the best code for this instance, requires computing codewords for complete binary codes with unequal codeword lengths, Code -1 introduced here might still be preferable in this instance, requiring less computation to encode and decode. For all tested distributions, Yokoo's code and the codes introduced here are both strict improvements on exponential-Golomb and Elias codes, confirming that, in practice, 0-smooth codes are very often preferable to those lacking this property.

Note that not all known codes for integers were tested here; certain codes can be ruled out due to the length of the first few codewords (e.g., Even-Rodeh [33], Zeta codes with $k > 2$ [4]), whereas others have significantly higher computational complexity (e.g., Fibonacci [34], [35]). In comparison to other feasible codes, the codes introduced here are a notable improvement. While not optimal, they can be quite useful in practical applications.

Appendix

Codeword lengths for infinite codes are estimated in relatively simple fashion, as more complex methods are unnecessary. Golin codes are each estimated based on the partial code and conditional entropy of the remaining items; we omit details for space. For other codes, some expected lengths are exactly known. The unary (G1) code is a mean value, which is known in the case of zeta and Yule-Simon distributions, while the average length of the Elias γ code (EG0) code for Yule-Simon with $\rho = 1$ is easily calculated as

$$\sum_{i=1}^{\infty} p(i)n(i) = 1 + 2\sum_{i=1}^{\infty} \frac{\lfloor \lg i \rfloor}{i(i+1)} = 1 + 2\sum_{j=0}^{\infty} j2^{-j-1} = 3.$$

Golin's algorithms both result in the same code for this distribution, since the algorithms' conditions result in groupings of probabilities summing to powers of two.

Optimal expected codeword lengths are estimated using an optimal code for a truncated distribution and the entropy of the remaining items; although not having the same guaranteed accuracy, the results seem to provide accurate estimates based upon the behavior of coding truncated probability distributions of increasing size. In [24], it is shown that sequences of such truncated distributions always have a subsequence converging to the optimal code, providing theoretical justification for the use of this technique, which also causes distributions for especially heavy tails, e.g., zeta distributions with $s < 2$, to be estimated with lower precision.

For most code/probability combinations considered here, we have that they are monotonic and we can find $\alpha, \beta, \kappa > 0, \mu, \xi > 0, \tau > 0, \upsilon > 0, \phi > 0$ such that

$$n(i) \in [\tau \ln(i+\mu+1) + \alpha, \upsilon \ln(i+\mu) + \beta]$$

and monotonic

$$p(i) \in \left[\frac{\phi}{(i+\kappa)^{\xi+1}}, \frac{\phi}{i^{\xi+1}}\right]$$

for large enough $i \geq i_{\min}$. Then, for $x > i_{\min}$, we have

$$\begin{aligned}
\sum_{i=x}^{\infty} p(i)n(i) &\geq \int_x^\infty p(i)n(i-1)di \\
&\geq \int_x^\infty \frac{\tau\phi \ln(i+\mu) + \alpha\phi}{(i+\kappa)^{\xi+1}} di \\
&\geq \phi \int_x^\infty \frac{\tau \ln(i+\kappa) + \tau f_{\min}(x) + \alpha}{(i+\kappa)^{\xi+1}} di \\
&= \frac{\tau\phi \ln(x + \min(\kappa,\mu)) + \tau\phi\xi^{-1} + \alpha\phi}{\xi(x+\kappa)^\xi}
\end{aligned}$$

where $f_{\min}(x) = \min(\ln(x+\mu) - \ln(x+\kappa), 0)$, and, similary,

$$\sum_{i=x}^{\infty} p(i)n(i) \leq \frac{\upsilon\phi \ln(x + \max(-1,\mu)) + \upsilon\phi\xi^{-1} + \beta\phi}{\xi(x-1)^\xi}$$

providing upper and lower bounds to average codeword length using code $N = \{n(i)\}$ for probability distribution $P = \{p(i)\}$. Other distributions (such as Golomb codes) and Shannon entropy can be bounded similarly. Such an approach enables us to find estimates with accuracies limited only by the precision of the partial summations (i.e., round-off error). For the probability distributions currently under consideration, we have:

	ξ	ϕ	κ
P_{GK}	1	1	$\lg e$
P_ρ^{YS}	ρ	ρ	$\rho\Gamma(\rho+1)$
P_s^ζ	0	$s-1$	$\zeta^{-1}(s)$

For γ [8], Yokoo [12], Levenshtein (Левенштейн) [27], and ζ_2 [4] codes, and for the Codes and extensions of the exponential-Golomb introduced here, $\alpha, \beta, \mu, \tau > 0, \upsilon > 0$ can be

	α	β	μ	τ	υ
γ, Yokoo	-1	1	0	$2\lg e$	$2\lg e$
л $(i > 1)$	2	2	-1	$\lg e$	$2.5\lg e$
ζ_2	0	2	0	$1.5\lg e$	$1.5\lg e$
EG k $(k \le 0)$ $(i > -k)$	$-1-k$	$1-k$	k	$2\lg e$	$2\lg e$
Code k $(k \le 0)$ $(i > -k)$	$\alpha_0 - k$	$-1-k$	$2+k$	$2\lg e$	$2\lg e$

where $\alpha_0 = 1 - 2\lg 3$. (Parameters for other codes can be similarly formulated, but these are unused here due to their inferiority at the distributions in question.)

For finding the best code within code families with multiple codes — such as Code k, EGk, and Gk (Golomb code k, defined in the main text) — partial sums can be used to limit the number of codes tested to a finite number. For example, these codes have $n(1) \to \infty$ as $k \to +\infty$, so at some point $p(1)n(1)$ will be too large to consider Code k with parameters $k > k_{\max}$ for some $k_{\max}$. Similarly, as $k \to -\infty$, the unary portion of the code can be used for the partial sum.

References

[1] C. F. Gauss, "Eine Aufgabe der Wahrscheinlichkeitsrechnung," 1800, in Werke Sammlung, Band 10 Abt 1, pp. 552–556, available from http://www-gdz.sub.uni-goettingen.de/cgi-bin/digbib.cgi?PPN235957348.

[2] R. O. Kuzmin, "Sur un problème de Gauss," in *Atti del Congresso Internazionale dei Matematici*, vol. 6, Sept. 1928, pp. 83–89.

[3] M. Mitzenmacher, "A brief history of generative models for power law and lognormal distributions," *Internet Math.*, vol. 1, no. 2, pp. 226–251, 2004.

[4] P. Boldi and S. Vigna, "Codes for the World-Wide Web," *Internet Math.*, vol. 2, no. 4, pp. 407–429, 2005.

[5] M. E. J. Newman, "Power laws, Pareto distributions and Zipf's law," *Contemporary Physics*, vol. 46, no. 5, pp. 323–351, Sept. 2005.

[6] N. N. Taleb, *The Black Swan: The Impact of the Highly Improbable*. New York, NY: Random House, 2007.

[7] J. Teuhola, "A compression method for clustered bit-vectors," *Inf. Processing Letters*, vol. 7, no. 6, pp. 308–311, Oct. 1978.

[8] P. Elias, "Universal codeword sets and representations of the integers," *IEEE Trans. Inf. Theory*, vol. IT-21, no. 2, pp. 194–203, Mar. 1975.

[9] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620–636, July 2003.

[10] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003.

[11] D. W. Matula and P. Kornerup, "An order preserving finite binary encoding of the rationals," in *Proc., 6th Symposium on Computer Arithmetic*, 1983, pp. 201–209.

[12] H. Yokoo, "An efficient representation of the integers for the distribution of partial quotients over the continued fractions," *J. Inform. Processing*, vol. 11, no. 4, pp. 288–293, 1988.

[13] S. W. Golomb, "A class of probability distributions on the integers," *Journal of Number Theory*, vol. 2, no. 2, pp. 189–192, May 1970.

[14] A. Kato, "Huffman-like optimal codes and search codes for infinite alphabets," 1997, unpublished manuscript.

[15] P. Boldi and S. Vigna, "The WebGraph framework I: Compression techniques," in *Proc. Thirteenth International World Wide Web Conference*. ACM Press, 2004, pp. 595–601.

[16] S. W. Golomb, "Run-length encodings," *IEEE Trans. Inf. Theory*, vol. IT-12, no. 3, pp. 399–401, July 1966.

[17] R. G. Gallager and D. C. van Voorhis, "Optimal source codes for geometrically distributed integer alphabets," *IEEE Trans. Inf. Theory*, vol. IT-21, no. 2, pp. 228–230, Mar. 1975.

[18] J. Abrahams, "Huffman-type codes for infinite source distributions," *Journal of the Franklin Institute*, vol. 331B, no. 3, pp. 265–271, May 1994.

[19] T. Chow and M. Golin, "Convergence and construction of minimal-cost infinite trees," in *Proc., 1998 IEEE Int. Symp. on Information Theory*, Aug. 1998, p. 227.

[20] N. Merhav, G. Seroussi, and M. Weinberger, "Optimal prefix codes for sources with two-sided geometric distributions," *IEEE Trans. Inf. Theory*, vol. IT-46, no. 2, pp. 121–135, Mar. 2000.

[21] M. J. Golin and K. K. Ma, "Algorithms for constructing infinite Huffman codes," Hong Kong University of Science & Technology Theoretical Computer Science Center, Tech. Rep. HKUST-TCSC-2004-07, Aug. 2004, available from http://www.cs.ust.hk/tcsc/RR/index_7.html.

[22] F. Bassino, J. Clément, G. Seroussi, and A. Viola, "Optimal prefix codes for two-dimensional geometric distributions," in *Proc., IEEE Data Compression Conf.*, Mar. 28–30, 2006, pp. 113–122.

[23] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, Sept. 1952.

[24] T. Linder, V. Tarokh, and K. Zeger, "Existence of optimal prefix codes for infinite source alphabets," *IEEE Trans. Inf. Theory*, vol. IT-43, no. 6, pp. 2026–2028, Nov. 1997.

[25] P. A. Humblet, "Optimal source coding for a class of integer alphabets," *IEEE Trans. Inf. Theory*, vol. IT-24, no. 1, pp. 110–112, Jan. 1978.

[26] S. J. Golin, "A simple variable-length code," *Signal Processing*, vol. 45, no. 1, pp. 23–35, Mar. 1995.

[27] V. I. Levenshtein, "On the redundancy and delay of separable codes for the natural numbers (об избыточности и замедлении разделимого кодирования натуральных чисел)," *Problems of Cybernetics*, vol. 20, pp. 173–179, 1968.

[28] P. Kornerup and D. W. Matula, "LCF: A lexicographic binary representation of the rationals," *J. Universal Comput. Sci.*, vol. 1, no. 7, pp. 484–503, July 1995.

[29] G. K. Zipf, "Relative frequency as a determinant of phonetic change," *Harvard Studies in Classical Philology*, vol. 40, pp. 1–95, 1929.

[30] G. U. Yule, "A mathematical theory of evolution, based on the conclusions of Dr. J. C. Willis, F.R.S." *Philos. Trans. Roy. Soc. London Ser. B*, vol. 213, pp. 21–87, 1925.

[31] H. A. Simon, "On a class of skew distribution functions," *Biometrika*, vol. 42, no. 3/4, pp. 425–440, 1955.

[32] J. Wen and J. D. Villasenor, "Structured prefix codes for quantized low-shape-parameter generalized Gaussian sources," *IEEE Trans. Inf. Theory*, vol. IT-45, no. 4, pp. 1307–1314, May 1999.

[33] S. Even and M. Rodeh, "Economical encoding of commas between strings," *Commun. ACM*, vol. 21, no. 4, pp. 315–317, Apr. 1978.

[34] R. M. Capocelli and A. De Santis, "On the redundancy of optimal codes with limited word length," *IEEE Trans. Inf. Theory*, vol. IT-38, no. 2, pp. 439–445, Mar. 1992.

[35] A. Apostolico and A. S. Fraenkel, "Fibonacci representation of strings of varying length using binary separators," *IEEE Trans. Inf. Theory*, vol. IT-33, no. 2, pp. 238–240, Mar. 1987.