

# Infinite-Alphabet Prefix Codes Optimal for $\beta$ -Exponential Penalties

Michael B. Baer  
Electronics for Imaging  
303 Velocity Way  
Foster City, California 94404  
Email: Michael.Baer@efi.com

**Abstract**—Let  $P = \{p(i)\}$  be a measure of strictly positive probabilities on the set of nonnegative integers. Although the countable number of inputs prevents usage of the Huffman algorithm, there are nontrivial  $P$  for which known methods find a source code that is optimal in the sense of minimizing expected codeword length. For some applications, however, a source code should instead minimize one of a family of nonlinear objective functions,  $\beta$ -exponential means, those of the form  $\log_a \sum_i p(i)a^{n(i)}$ , where  $n(i)$  is the length of the  $i$ th codeword and  $a$  is a positive constant. Applications of such minimizations include a problem of maximizing the chance of message receipt in single-shot communications ( $a < 1$ ) and a problem of minimizing the chance of buffer overflow in a queueing system ( $a > 1$ ). This paper introduces methods for finding codes optimal for such exponential means. One method applies to geometric distributions, while another applies to distributions with lighter tails. The latter algorithm is applied to Poisson distributions. Both are extended to minimizing maximum pointwise redundancy.

## I. INTRODUCTION, MOTIVATION, AND MAIN RESULTS

If probabilities are known, optimal lossless source coding of individual symbols (and blocks of symbols) is usually done using David Huffman's famous algorithm [1]. There are, however, cases that this algorithm does not solve [2]. Problems with an infinite number of possible inputs — e.g., geometrically-distributed variables — are not covered. Also, in some instances, the optimality criterion — or *penalty* — is not the linear penalty of expected length. Both variants of the problem have been considered in the literature, but not simultaneously. This paper discusses cases which are both infinite and nonlinear.

An infinite-alphabet source emits symbols drawn from the alphabet  $\mathcal{X}_\infty = \{0, 1, 2, \dots\}$ . (More generally, we use  $\mathcal{X}$  to denote an input alphabet whether infinite or finite.) Let  $P = \{p(i)\}$  be the sequence of probabilities for each symbol, so that the probability of symbol  $i$  is  $p(i) > 0$ . The source symbols are coded into binary codewords. The codeword  $c(i) \in \{0, 1\}^*$  in code  $C$ , corresponding to input symbol  $i$ , has length  $n(i)$ , thus defining length distribution  $N$ .

Perhaps the most well-known such codes are the optimal codes derived by Golomb for geometric distributions [3], [4]. There are many reasons for using infinite-alphabet codes rather than codes for finite alphabets, such as Huffman codes. The most obvious use is for cases with no upper bound — or at least no known upper bound — on the number of possible

items. In addition, for many cases it is far easier to come up with a general code for integers rather than a Huffman code for a large but finite number of inputs. Similarly, it is often faster to encode and decode such well-structured codes. For these reasons, infinite-alphabet codes and variants of them are widely used in image and video compression standards [5], [6], as well as for compressing text, audio, and numerical data.

To date, the literature on infinite-alphabet codes has considered only finding efficient uniquely decipherable codes with respect to minimizing expected codeword length  $\sum_i p(i)n(i)$ . Other utility functions, however, have been considered for finite-alphabet codes. Campbell [7] introduced a problem in which the penalty to minimize, given some continuous (strictly) monotonic increasing *cost function*  $\varphi(x) : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ , is

$$L(P, N, \varphi) = \varphi^{-1} \left( \sum_i p(i)\varphi(n(i)) \right)$$

and specifically considered the exponential subcases with exponent  $a > 1$ :

$$L_a(P, N) \triangleq \log_a \sum_i p(i)a^{n(i)}, \quad (1)$$

that is,  $\varphi(x) = a^x$ . Note that minimizing penalty  $L$  is also an interesting problem for  $0 < a < 1$  and approaches the standard penalty  $\sum_i p(i)n(i)$  for  $a \rightarrow 1$  [7]. While  $\varphi(x)$  decreases for  $a < 1$ , one can map decreasing  $\varphi$  to a corresponding increasing function  $\tilde{\varphi}(l) = \varphi_{\max} - \varphi(l)$  (e.g., for  $\varphi_{\max} = 1$ ) without changing the penalty value. Thus this problem, equivalent to maximizing  $\sum_i p(i)a^{n(i)}$ , is a subset of those considered by Campbell. All penalties of the form (1) are called  $\beta$ -exponential means, where  $\beta = \log_2 a$ .

Campbell noted certain properties for  $\beta$ -exponential means, but did not consider applications for these means. Applications were later found for the problem with  $a > 1$  [8]–[11] and with  $a < 1$  [12], [13]. The former set of applications all relate to a buffer overflow problem discussed in the full version of this paper [13].

One can solve any instance of the exponential penalty with a finite number of inputs using a linear-time algorithm found independently by Hu *et al.* [14, p. 254], Parker [15, p. 485], and Humblet [16, p. 25], [10, p. 231], although only the last

of these considered  $a < 1$ . (The simultaneity of these lines of research was likely due to the appearance of the first paper on adapting the Huffman algorithm to a nonlinear penalty,  $\max_i(p(i) + n(i))$  for given  $p(i) \in \mathbb{R}_+$ , in 1976 [17].) We present the exponential-penalty algorithm here; even though it cannot be used for an infinite alphabet, it can be used to derive and show the optimality of infinite-alphabet codes:

#### Procedure for Exponential Huffman Coding

This procedure finds the optimal code whether  $a > 1$  (a minimization of the average of a growing exponential) or  $a < 1$  (a maximization of the average of a decaying exponential). Note that it minimizes (1), even if the “probabilities” do not add to 1. We refer to such arbitrary positive inputs as *weights*, denoted by  $w(i)$  instead of  $p(i)$ :

- 1) Each item  $i$  has weight  $w(i) \in W_{\mathcal{X}}$ , where  $\mathcal{X}$  is the (finite) alphabet and  $W_{\mathcal{X}} = \{w(i)\}$  is the set of all such weights. Assume each item  $i$  has codeword  $c(i)$ , to be determined later.
- 2) Combine the items with the two smallest weights  $w(j)$  and  $w(k)$  into one item with the combined weight  $\tilde{w}(j) = a \cdot (w(j) + w(k))$ . This item has codeword  $\tilde{c}(j)$ , to be determined later, while item  $j$  is assigned codeword  $c(j) = \tilde{c}(j)0$  and  $k$  codeword  $c(k) = \tilde{c}(j)1$ . Since these have been assigned in terms of  $\tilde{c}(j)$ , replace  $w(j)$  and  $w(k)$  with  $\tilde{w}(j)$  in  $W_{\mathcal{X}}$  to form  $W_{\tilde{\mathcal{X}}}$ .
- 3) Repeat procedure, now with the remaining codewords (reduced in number by 1) and corresponding weights, until only one item is left. The weight of this item is  $\sum_i w(i)a^{n(i)}$ . All codewords are now defined by assigning the null string to this trivial item.

Optimality of the algorithm is justified as in Huffman coding, in that an exchange argument can be used to show that an optimal code exists for which the least likely two codewords differ in only their final bit, allowing a reduction to the equivalent smaller problem that linearly combines their weights. This algorithm can be modified to run in linear time (to input size) given sorted weights in the same manner as Huffman coding [18].

Note that this algorithm assigns an explicit weight to each node of the resulting code tree implied by having each item represented by a node with its parent representing the combined items: If a node is a leaf, its weight is given by the associated probability; otherwise its weight is defined recursively as  $a$  times the sum of its children. This concept is useful in visualizing both the coding procedure and its output.

It is also worthwhile to note that  $a \leq 0.5$  is degenerate, always resulting in the *unary code* (for infinite inputs) or a unary-like code (for finite inputs) being optimal for any probability distribution. The unary code has ones terminated by a zero, i.e., the code with codewords of the form  $\{1^i 0 : i \geq 0\}$ . The unary-like code is a truncated unary code, that is, a code with identical codewords to the unary code except for the longest codeword, with is of the form  $1^{|\mathcal{X}|-1}$ . For the unary-like code, optimality can be shown using the coding procedure; the smallest two items,  $j$  and  $k$ , are combined, and the resulting item has weight  $a \cdot (w(j) + w(k))$ . This is no

larger than the larger of the constituent weights, meaning that the resulting item will be combined with third-smallest item, and so forth, resulting in a unary-like code. Taking limits, informally speaking, results in a unary limit code; formally, this is a straightforward corollary of Theorem 2 in Section III.

If  $a > 0.5$ , a code with finite penalty exists if and only if Rényi entropy of order  $\alpha(a) = (1 + \log_2 a)^{-1}$  is finite, e.g., [19]. It was Campbell who first noted the connection between the optimal value of  $L_a(P, N)$  and Rényi entropy

$$\begin{aligned} H_\alpha(P) &\triangleq \frac{1}{1-\alpha} \log_2 \sum_{i \in \mathcal{X}} p(i)^\alpha \\ \Rightarrow H_{\alpha(a)}(P) &= \frac{1+\log_2 a}{\log_2 a} \log_2 \sum_{i \in \mathcal{X}} p(i)^{(1+\log_2 a)^{-1}}. \end{aligned}$$

This relationship is

$$H_{\alpha(a)}(P) \leq L_a(P, N) < H_{\alpha(a)}(P) + 1$$

which should not be surprising given the similar relationship between Huffman coding and Shannon entropy [20], which corresponds to  $a \rightarrow 1$ ,  $H_1(P)$  [21].

One must be careful regarding the meaning of an “optimal code” when there are an infinite number of possible codes satisfying the Kraft inequality with equality. One might ask whether there must exist an optimal code or if there can be an infinite sequence of codes of decreasing penalty without any code achieving the limit penalty value. Fortunately the answer is the former, the proof being a special case of Theorem 2 in [19]. The question is then how to find one of these optimal source codes given parameter  $a$  and probability measure  $P$ .

As in the linear case, this is not known for general  $P$ , but can be found for certain common distributions. In the next section, we consider geometric distributions and find that Golomb codes are optimal, although the optimal Golomb code for a given probability mass function varies according to  $a$ . The main result of Section II is that, for  $p_\theta(i) = (1 - \theta)\theta^i$  and  $a \in \mathbb{R}_+$ ,  $Gk$ , the Golomb code with parameter  $k$ , is optimal for

$$k = \max(1, \lceil -\log_\theta a - \log_\theta(1 + \theta) \rceil).$$

In Section III, we consider distributions that are relatively light-tailed, that is, that decline faster than certain geometric distributions. If there is a nonnegative integer  $r$  such that for all  $j > r$  and  $i < j$ ,

$$p(i) \geq p(j)$$

and

$$p(i) \geq \sum_{k=j+1}^{\infty} p(k)a^{k-j}$$

then an optimal binary prefix code can be found which is a generalization of the unary code. A specific case of this is the Poisson distribution, considered in Section IV, where an aforementioned  $r$  is given by  $r = \max(\lceil 2a\lambda \rceil - 2, \lceil e\lambda \rceil - 1)$  for  $p_\lambda(i) = \lambda^i e^{-\lambda} / i!$ . Section V discusses the maximum point-wise redundancy penalty, which has a similar solution with light-tailed distributions and for which the Golomb code  $Gk$  is optimal for  $k = \lceil -1/\log_2 \theta \rceil$  with geometric distributions. Complete proofs and illustrations, as well as additional results, are given in the full version [13].

## II. GEOMETRIC DISTRIBUTION WITH EXPONENTIAL PENALTY

Consider the geometric distribution

$$p_\theta(i) = (1 - \theta)\theta^i$$

for parameter  $\theta \in (0, 1)$ . This distribution arises in run-length coding as well as in other circumstances [3], [4].

For the traditional linear penalty, a Golomb code with parameter  $k$  — or  $Gk$  — is optimal for  $\theta^k + \theta^{k+1} \leq 1 < \theta^{k-1} + \theta^k$ . Such a code consists of a unary prefix followed by a binary suffix, the latter taking one of  $k$  possible values. If  $k$  is a power of two, all binary suffix possibilities have the same length; otherwise, their lengths differ by at most 1 and  $\sum_i 2^{-n(i)} = 1$ . Such binary codes are called *complete* codes. This defines the Golomb code; for example, the Golomb code for  $k = 3$  is:

$i$	$p(i)$	$c(i)$
0	$1 - \theta$	0 0
1	$(1 - \theta)\theta$	0 10
2	$(1 - \theta)\theta^2$	0 11
3	$(1 - \theta)\theta^3$	10 0
4	$(1 - \theta)\theta^4$	10 10
5	$(1 - \theta)\theta^5$	10 11
6	$(1 - \theta)\theta^6$	110 0
7	$(1 - \theta)\theta^7$	110 10
8	$(1 - \theta)\theta^8$	110 11
9	$(1 - \theta)\theta^9$	1110 0
$\vdots$	$\vdots$	$\vdots$

where the space in the code separates the unary prefix from the complete suffix. In general, codeword  $j$  for  $Gk$  is of the form  $\{1^{[j/k]}0b(j \bmod k, k) : j \geq 0\}$ , where  $b(j \bmod k, k)$  is a complete binary code for the  $j - k[j/k] + 1$ th of  $k$  items.

It turns out that such codes are optimal for the exponential penalty:

*Theorem 1:* For  $a \in \mathbb{R}_+$ , if

$$\theta^k + \theta^{k+1} \leq \frac{1}{a} < \theta^{k-1} + \theta^k \quad (2)$$

for  $k \geq 1$ , then the  $k$  Golomb code is the optimal code for  $P_\theta$ . If no such  $k$  exists, the unary code is optimal.

This rule for finding an optimal Golomb  $Gk$  code is equivalent to

$$k = \max(1, \lceil -\log_\theta a - \log_\theta(1 + \theta) \rceil).$$

This is a generalization of the traditional linear result since this corresponds to  $a \rightarrow 1$ . Cases in which the left inequality of (2) is an equality have multiple solutions, as with linear coding; see, e.g., [22, p. 289].

It is equivalent for the bits of the unary prefix to be reversed, that is, to use  $\{0^{[j/k]}1b(j \bmod k, k) : j \geq 0\}$  (as in [4]) instead of  $\{1^{[j/k]}0b(j \bmod k, k) : j \geq 0\}$  (as in [3]). The latter has the advantage of being alphabetic, that is,  $i > j$  if and only if  $c(i)$  is lexicographically after  $c(j)$ .

A little algebra reveals that, for a distribution  $P_\theta$  and a Golomb code with parameter  $k$  (lengths  $N_k$ ),

$$\begin{aligned} L_a(P_\theta, N_k) &= \log_a \sum_{i=0}^{\infty} (1 - \theta)\theta^i a^{\lceil \frac{i+1-z}{k} \rceil + g} \\ &= g + \log_a \left(1 + \frac{(a-1)\theta^z}{1 - a\theta^k}\right) \end{aligned} \quad (3)$$

where  $g = \lfloor \log_2 k \rfloor + 1$  and  $z = 2^g - k$ . Therefore, Theorem 1 provides the  $k$  that minimizes (3). If  $a > 0.5$ , the corresponding Rényi entropy is

$$H_{\alpha(a)}(P_\theta) = \log_a \frac{1 - \theta}{(1 - \theta^{\alpha(a)})^{1/\alpha(a)}} \quad (4)$$

where we recall that  $\alpha(a) = (1 + \log_2 a)^{-1}$ . (Again,  $a \leq 0.5$  is degenerate, an optimal code being unary with no corresponding Rényi entropy.)

In evaluating the effectiveness of the optimal code, one might use the following definition of *average pointwise redundancy* (or just *redundancy*):

$$\bar{R}_a(N, P) \triangleq L_a(P, N) - H_{\alpha(a)}(P).$$

For nondegenerate values, we can plot the  $\bar{R}_a(N_{\theta,a}^*, P_\theta)$  obtained from the minimization. This is done for  $a > 1$  and  $a < 1$  in Fig. 2. As  $a \rightarrow 1$ , the plot approaches the redundancy plot for the linear case, e.g., [4], reproduced as Fig. 1.

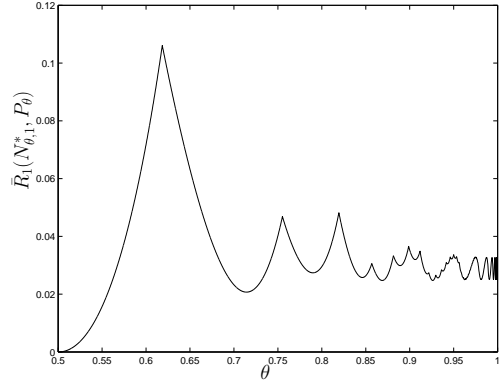


Fig. 1. Redundancy of the optimal code for the geometric distribution with the traditional linear penalty.

## III. OTHER INFINITE SOURCES

In this section we consider another type of probability distribution for binary coding, a type with a light tail. Humblet's approach [23], later extended in [24], uses the fact that there is always an optimal code consisting of a finite number of nonunary codewords for any probability distribution with a relatively light tail, one for which there is an  $r$  such that, for all  $j > r$  and  $i < j$ ,  $p(i) \geq p(j)$  and  $p(i) \geq \sum_{k=j+1}^{\infty} p(k)$ . Due to the additive nature of Huffman coding, the unary part can be considered separately, and the remaining codewords can be found via the Huffman algorithm. Once again, this has to be modified for the exponential case.

We wish to show that the optimal code can be obtained when there is a nonnegative integer  $r$  such that, for all  $j > r$

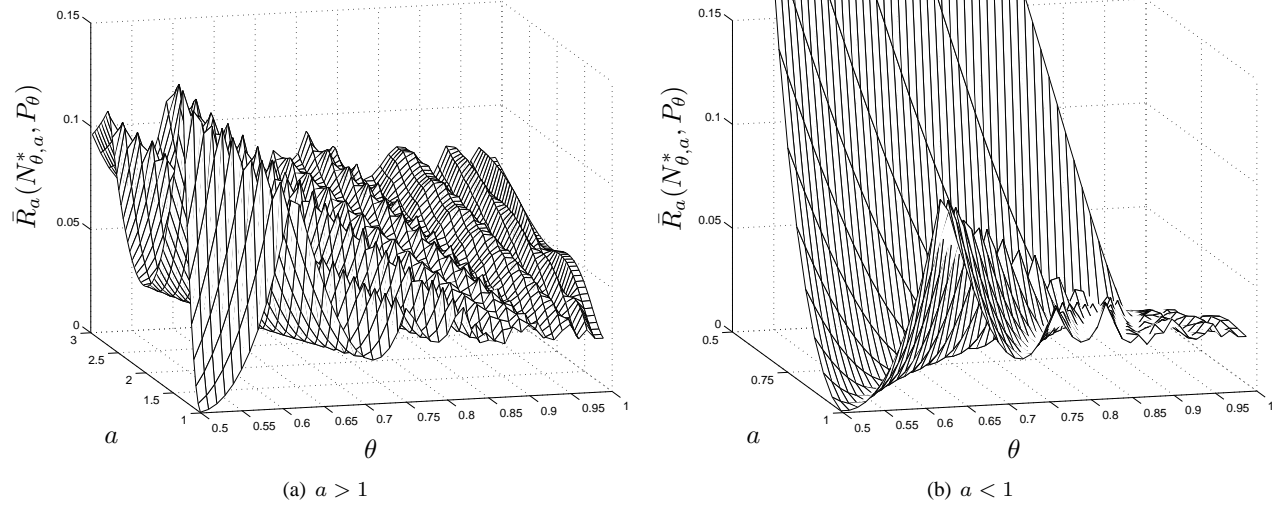


Fig. 2. Redundancy of the optimal code for the geometric distribution with the exponential penalty (parameter  $a$ ).  $\bar{R}_a(N_{\theta,a}^*, P_\theta) = L_a(P_\theta, N_{\theta,a}^*) - H_{\alpha(a)}(P_\theta)$ , where  $\alpha(a) = (1 + \log_2 a)^{-1}$ ,  $P_\theta$  is the geometric probability sequence implied by  $\theta$ , and  $N_{\theta,a}^*$  is the optimal length sequence for distribution  $P_\theta$  and parameter  $a$ .

and  $i < j$ ,

$$p(i) \geq \max \left( p(j), \sum_{k=j+1}^{\infty} p(k) a^{k-j} \right).$$

The optimal code is obtained by considering the reduced alphabet consisting of symbols  $0, 1, \dots, r+1$  with weights

$$w(i) = \begin{cases} p(i), & i \leq r \\ \sum_{k=r+1}^{\infty} p(k) a^{k-r}, & i = r+1. \end{cases} \quad (5)$$

Apply exponential Huffman coding to this reduced set of weights. For items  $0$  through  $r$ , the Huffman codewords for the reduced and the infinite alphabets are identical. Each other item  $i > r$  has a codeword consisting of the reduced codeword for item  $r+1$  (which, without loss of generality, consists of all 1's) followed by the unary code for  $i-r-1$ . We call such codes *unary-ended*.

**Theorem 2:** Let  $p(\cdot)$  be a probability measure on the set of nonnegative integers, and let  $a$  be the parameter of the penalty to be optimized. If there is a nonnegative integer  $r$  such that for all  $j > r$  and  $i < j$ ,

$$p(i) \geq p(j)$$

and

$$p(i) \geq \sum_{k=j+1}^{\infty} p(k) a^{k-j} \quad (6)$$

then there exists a minimum-penalty binary prefix code with every codeword  $j > r$  consisting of  $j-x$  1's followed by one 0 for some fixed nonnegative integer  $x$ .

The rate at which  $p(\cdot)$  must decrease in order to satisfy condition (6) clearly depends on  $a$ . One simple sufficient condition — provable via induction — is that it satisfies  $p(i) \geq ap(i+1) + ap(i+2)$  for large  $i$ . A less general condition is that  $p(i)$  eventually decreases at least as fast as

$g^i$  where  $g = (\sqrt{1+4/a} - 1)/2$ , the same ratio needed for a unary geometric code for  $\theta = g$ , as in (2). For  $a \rightarrow 1$ , these conditions approach those derived in [23]. The stronger results of [24] do not easily extend here due to the nonadditivity of the exponential penalty.

Before moving on to an example application of Theorem 2, it is worthwhile to note that these techniques are easily extensible to finding an optimal alphabetic code — that is, one with  $c(i)$ 's arranged in lexicographical order — for  $a > 1$ . One needs only to find the optimal alphabetic code for the reduced code with weights given in equation (5), as in [14], with codewords for  $i > r$  consisting of the reduced code's codeword for  $r+1$  followed by  $i-r-1$  ones and one zero. As previously mentioned, Golomb codes are also alphabetic and thus are optimal alphabetic codes for the geometric distribution.

#### IV. EXAMPLE: POISSON RANDOM VARIABLES

Examples for the geometric case are trivial given Theorem 1. Consider though the optimal codes for the Poisson distribution,

$$p_\lambda(i) = \frac{\lambda^i e^{-\lambda}}{i!}.$$

How does one find a suitable value for  $r$  (as in Section III) in such a case? It has been shown that  $r \geq \lceil e\lambda \rceil - 1$  yields  $p(i) \geq p(j)$  for all  $j > r$  and  $i < j$ , satisfying the first condition of Theorem 2 [23]. Moreover, if, in addition,  $j \geq \lceil 2a\lambda \rceil - 1$  (and thus  $j > a\lambda - 1$ ), then

$$\begin{aligned} & \sum_{k=1}^{\infty} p(j+k) a^k \\ &= \frac{e^{-\lambda} \lambda^j}{j!} \left[ \frac{a\lambda}{j+1} + \frac{a^2 \lambda^2}{(j+1)(j+2)} + \dots \right] \\ &< p(j) \left[ \frac{a\lambda}{j+1} + \frac{a^2 \lambda^2}{(j+1)^2} + \dots \right] \\ &= p(j) \frac{\frac{a\lambda}{j+1}}{1 - \frac{a\lambda}{j+1}} \\ &\leq p(j) \\ &\leq p(i). \end{aligned}$$

Thus, since we consider  $j > r$ ,  $r = \max(\lceil 2a\lambda \rceil - 2, \lceil e\lambda \rceil - 1)$  is sufficient to establish an  $r$  such that the above method yields the optimal infinite-alphabet code.

In order to find the optimal reduced code, use

$$\begin{aligned} w(r+1) &= \sum_{k=r+1}^{\infty} p(k)a^{k-r} \\ &= a^{-r}e^{\lambda(a-1)} - \sum_{k=0}^r p(k)a^{k-r}. \end{aligned}$$

For example, consider the Poisson distribution with  $\lambda = 1$ . We code this for both  $a = 1$  and  $a = 2$ . For both values,  $r = 2$ , so both are easy to code. For  $a = 1$ ,  $w(3) = 1 - 2.5e^{-1} \approx 0.0803\dots$ , while, for  $a = 2$ ,  $w(3) = 0.25e - 1.25e^{-1} \approx 0.2197\dots$ . After using the appropriate Huffman procedure on each reduced source of 4 weights, we find that the optimal code for  $a = 1$  has lengths  $N = \{1, 2, 3, 4, 5, 6, \dots\}$  — those of the unary code — while the optimal code for  $a = 2$  has lengths  $N = \{2, 2, 2, 3, 4, 5, \dots\}$ .

## V. REDUNDANCY PENALTIES

It is natural to ask whether the above results can be extended to other penalties. One penalty discussed in the literature is that of maximal pointwise redundancy [25], in which one seeks to find a code to minimize

$$R^*(N, P) \triangleq \max_{i \in \mathcal{X}} [n(i) + \log_2 p(i)].$$

This can be shown to be a limit of the exponential case, as in [26], allowing us to analyze it using the same techniques as exponential Huffman coding. This limit can be shown by defining  $d$ th exponential redundancy as follows:

$$\begin{aligned} R_d(N, P) &\triangleq \frac{1}{d} \log_2 \sum_{i \in \mathcal{X}} p(i) 2^{d(n(i) + \log_2 p(i))} \\ &= \frac{1}{d} \log_2 \sum_{i \in \mathcal{X}} p(i)^{1+d} 2^{dn(i)}. \end{aligned}$$

Thus  $R^*(N, P) = \lim_{d \rightarrow \infty} R_d(N, P)$ , and the above methods should apply in the limit. In particular, the

$$k = \lceil -1/\log_2 \theta \rceil$$

Golomb code is optimal for minimizing maximum pointwise redundancy for  $P_\theta$ . For light tails, if there is a nonnegative integer  $r$  such that, for all  $j \geq r$  and  $i < r$ ,

$$p(j) \leq 2^{r-j} p(i)$$

then a binary prefix code with minimum maximum redundancy exists with every codeword  $j > r$  consisting of  $j - x$  1's followed by one 0 for some fixed nonnegative integer  $x$ . The first  $r + 1$  lengths are identical to those for the optimal code for weight distribution  $\{p(0), p(1), \dots, p(r), p(r+1)\}$ . The proofs of these results are in the full version [13].

## REFERENCES

- [1] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, Sept. 1952.
- [2] J. Abrahams, "Code and parse trees for lossless source encoding," *Communications in Information and Systems*, vol. 1, no. 2, pp. 113–146, Apr. 2001.
- [3] S. W. Golomb, "Run-length encodings," *IEEE Trans. Inf. Theory*, vol. IT-12, no. 3, pp. 399–401, July 1966.
- [4] 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.
- [5] 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.
- [6] M. 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, no. 8, pp. 1309–1324, Aug. 2000, originally as Hewlett-Packard Laboratories Technical Report No. HPL-98-193R1, November 1998, revised October 1999. Available from <http://www.hpl.hp.com/loco/>.
- [7] L. L. Campbell, "Definition of entropy by means of a coding problem," *Z. Wahrscheinlichkeitstheorie und verwandte Gebiete*, vol. 6, pp. 113–118, 1966.
- [8] F. Jelinek, "Buffer overflow in variable length coding of fixed rate sources," *IEEE Trans. Inf. Theory*, vol. IT-14, no. 3, pp. 490–501, May 1968.
- [9] F. Jelinek and K. S. Schneider, "On variable-length-to-block coding," *IEEE Trans. Inf. Theory*, vol. IT-18, no. 6, pp. 765–774, Nov. 1972.
- [10] P. A. Humblet, "Generalization of Huffman coding to minimize the probability of buffer overflow," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 2, pp. 230–232, Mar. 1981.
- [11] A. C. Blumer and R. J. McEliece, "The Rényi redundancy of generalized Huffman codes," *IEEE Trans. Inf. Theory*, vol. IT-34, no. 5, pp. 1242–1249, Sept. 1988.
- [12] M. Baer, "Rényi to Rényi — source coding under siege," in *Proc., 2006 IEEE Int. Symp. on Information Theory*, July 9–14, 2006, pp. 1258–1262.
- [13] M. B. Baer, "Integer coding with nonlinear costs," *IEEE Trans. Inf. Theory*, submitted for publication, preprint available from <http://arxiv.org/abs/cs.IT/0511003>.
- [14] T. C. Hu, D. J. Kleitman, and J. K. Tamaki, "Binary trees optimum under various criteria," *SIAM J. Appl. Math.*, vol. 37, no. 2, pp. 246–256, Apr. 1979.
- [15] D. S. Parker, Jr., "Combinatorial merging and Huffman's algorithm," *IEEE Trans. Computers*, vol. 28, no. 5, pp. 365–367, May 1979.
- [16] P. A. Humblet, "Source coding for communication concentrators," Ph.D. dissertation, Massachusetts Institute of Technology, 1978.
- [17] M. C. Golumbic, "Combinatorial merging," *IEEE Trans. Comput.*, vol. C-25, no. 11, pp. 1164–1167, Nov. 1976.
- [18] J. van Leeuwen, "On the construction of Huffman trees," in *Proc. 3rd Int. Colloquium on Automata, Languages, and Programming*, July 1976, pp. 382–410.
- [19] M. B. Baer, "Source coding for quasiarithmetic penalties," *IEEE Trans. Inf. Theory*, vol. IT-52, no. 10, pp. 4380–4393, Oct. 2006.
- [20] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, July 1948.
- [21] A. Rényi, "On measures of entropy and information," in *Proc. 4th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, 1961, pp. 547–561.
- [22] M. J. Golin, "A combinatorial approach to Golomb forests," *Theoretical Computer Science*, vol. 263, no. 1–2, pp. 283–304, July 2001.
- [23] 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.
- [24] A. Kato, T. S. Han, and H. Nagaoka, "Huffman coding with an infinite alphabet," *IEEE Trans. Inf. Theory*, vol. IT-42, no. 3, pp. 977–984, May 1996.
- [25] M. Drmota and W. Szpankowski, "Precise minimax redundancy and regret," *IEEE Trans. Inf. Theory*, vol. IT-50, no. 11, pp. 2686–2707, Nov. 2004.
- [26] M. B. Baer, "A general framework for codes involving redundancy minimization," *IEEE Trans. Inf. Theory*, vol. IT-52, no. 1, pp. 344–349, Jan. 2006.