Source Coding for Quasiarithmetic Penalties

Michael B. Baer, *Member, IEEE*

Abstract— Whereas Huffman coding finds a prefix code minimizing mean codeword length for a given finite-item probability distribution, *quasiarithmetic* or *quasilinear* coding problems have the goal of minimizing a generalized mean of the form $\varphi^{-1}(\sum_i p_i \varphi(l_i))$, where l_i denotes the length of the ith codeword, p_i denotes the corresponding probability, and φ is a monotonically increasing cost function. Such problems, proposed by Campbell, have a number of diverse applications. Several cost functions are shown here to yield quasiarithmetic problems with simple redundancy bounds in terms of a generalized entropy. A related property, also shown here, involves the existence of optimal codes: For "well-behaved" cost functions, optimal codes always exist for (possibly infinite-alphabet) sources having finite generalized entropy. An algorithm is introduced for finding binary codes optimal for convex cost functions. This algorithm, which can be extended to other minimization utilities, can be performed using quadratic time and linear space. This reduces the computational complexity of a problem involving minimum delay in a queue, allows combinations of previously considered problems to be optimized, and greatly expands the set of problems solvable in quadratic time and linear space.

Index Terms— Huffman codes, generalized entropies, generalized means, optimal prefix code, quasiarithmetic means, queueing.

I. INTRODUCTION

IT is well known that Huffman coding [1] yields a prefix code minimizing expected length for a known finite probability mass function. Less well known are the many variants of this algorithm that have been proposed for related problems [2]. For example, in his doctoral dissertation, Humblet discussed two problems in queueing that have nonlinear terms to minimize [3]. These problems, and many others, can be reduced to a certain family of generalizations of the Huffman problem introduced by Campbell in [4].

In all such source coding problems, a source emits symbols drawn from the alphabet $\mathcal{X} = \{1, 2, \ldots, n\}$, where n is an integer (or possibly infinity). Symbol i has probability p_i, thus defining probability mass function $\boldsymbol{p}$. We assume without loss of generality that $p_i > 0$ for every $i \in \mathcal{X}$, and that $p_i \leq p_j$ for every $i > j$ ($i, j \in \mathcal{X}$). The source symbols are coded into codewords composed of symbols of the D-ary alphabet $\{0, 1, \ldots, D-1\}$, most often the binary alphabet, $\{0, 1\}$. The codeword c_i corresponding to symbol i has length l_i, thus

defining length distribution $\boldsymbol{l}$. Finding values for $\boldsymbol{l}$ is sufficient to find a corresponding code.

Huffman coding minimizes $\sum_{i \in \mathcal{X}} p_i l_i$. Campbell's formulation adds a continuous (strictly) monotonic increasing *cost function* $\varphi(l) : \mathbb{R}_+ \to \mathbb{R}_+$. The value to minimize is then

$$L(\boldsymbol{p}, \boldsymbol{l}, \varphi) \triangleq \varphi^{-1}\left(\sum_{i \in \mathcal{X}} p_i \varphi(l_i)\right). \tag{1}$$

Campbell called (1) the "mean length for the cost function φ"; for brevity, we refer to it, or any value to minimize, as the *penalty*. Penalties of the form (1) are called *quasiarithmetic* or *quasilinear*; we use the former term in order to avoid confusion with the more common use of the latter term in convex optimization theory.

Note that such problems can be mathematically described if we make the natural coding constraints explicit: the integer constraint, $l_i \in \mathbb{Z}_+$, and the Kraft (McMillan) inequality [5],

$$\kappa(\boldsymbol{l}) \triangleq \sum_{i \in \mathcal{X}} D^{-l_i} \leq 1.$$

Given these constraints, examples of φ in (1) include a quadratic cost function useful in minimizing delay due to queueing and transmission,

$$\varphi(x) = \alpha x + \beta x^2 \tag{2}$$

for nonnegative α and β [6], and an exponential cost function useful in minimizing probability of buffer overflow, $\varphi(x) = D^{tx}$ for positive t [3], [7]. These and other examples are reviewed in the next section.

Campbell noted certain properties for convex φ, such as those examples above, and others for concave φ. Strictly concave φ penalize shorter codewords more harshly than the linear function and penalize longer codewords less harshly. Conversely, strictly convex φ penalize longer codewords more harshly than the linear function and penalize shorter codewords less harshly. Convex φ need not yield penalty functions convex in $\boldsymbol{l}$, although $\varphi(L(\boldsymbol{p}, \boldsymbol{l}, \varphi))$ is clearly convex in $\boldsymbol{l}$ if and only if $\varphi(\boldsymbol{l})$ is. Note that one can map decreasing φ to a corresponding increasing function $\tilde{\varphi}(l) = \varphi_{\max} - \varphi(l)$ without changing the value of $L(\boldsymbol{p}, \boldsymbol{l}, \varphi)$ (e.g., for $\varphi_{\max} = \varphi(0)$). Thus the restriction to increasing φ can be trivially relaxed.

We can generalize L by using a two-argument cost function $f(l, p)$ instead of $\varphi(l)$, as in (3), and adding $\{\infty\}$ to its range. We usually choose functions with the following property:

Definition 1: A cost function $f(l, p)$ and its associated penalty $\tilde{L}$ are *differentially monotonic* if, for every $l > 1$, whenever $f(l-1, p_i)$ is finite and $p_i > p_j$, $f(l, p_i) - f(l-1, p_i) > f(l, p_j) - f(l-1, p_j)$.

This property means that the contribution to the penalty of an lth bit in a codeword will be greater if the corresponding

event is more likely. Clearly any $f(l,p) = p\varphi(l)$ will be differentially monotonic. This restriction on the generalization will aid in finding algorithms for coding such cost functions, which we denote as *generalized quasiarithmetic* penalties:

Definition 2: Let $f(l,p) : \mathbb{R}_+ \times [0,1] \to \mathbb{R}_+ \cup \{\infty\}$ be a function nondecreasing in l. Then

$$\tilde{L}(\boldsymbol{p},\boldsymbol{l},f) \triangleq \sum_{i \in \mathcal{X}} f(l_i, p_i) \tag{3}$$

is called a *generalized quasiarithmetic* penalty. Further, if f is convex in l, it is called a *generalized quasiarithmetic convex* penalty.

As indicated, quasiarithmetic penalties — mapped with φ using $f(l_i, p_i) = p_i \varphi(l_i)$ to $\tilde{L}(\boldsymbol{p}, \boldsymbol{l}, f) = \varphi(L(\boldsymbol{p}, \boldsymbol{l}, \varphi))$ — are differentially monotonic, and thus can be considered a special case of differentially monotonic generalized quasiarithmetic penalties.

In this paper, we seek properties of and algorithms for solving problems of this form, occasionally with some restrictions (e.g., to convexity of φ). In the next section, we provide examples of the problem in question. In Section III, we investigate Campbell's quasiarithmetic penalties, expanding beyond Campbell's properties for a certain class of φ that we call *subtranslatory*. This will extend properties — entropy bounds, existence of optimal codes — previously known only for linear φ and, in the case of entropy bounds, for φ of the exponential form $\varphi(x) = D^{tx}$. These properties pertain both to finite and infinite input alphabets, and some are applicable beyond subtranslatory penalties. We then turn to algorithms for finding an optimal code for finite alphabets in Section IV; we start by presenting and extending an alternative to code tree notation, nodeset notation, originally introduced in [6]. Along with the Coin Collector's problem, this notation can aid in solving coding problems with generalized quasiarithmetic convex penalties. We explain, prove, and refine the resulting algorithm, which is $O(n^2)$ time and $O(n)$ space when minimizing for a differentially monotonic generalized quasiarithmetic penalty; the algorithm can be extended to other penalties with a like or slightly greater complexity. This is an improvement, for example, on a result of Larmore, who in [6] presented an $O(n^3)$-time $O(n^3)$-space algorithm for cost function (2) in order to optimize a more complicated penalty related to communications delay. Our result thus improves overall performance for the quadratic problem and offers an efficient solution for the more general convex quasiarithmetic problem. Conclusions are presented in Section V.

II. EXAMPLES

The additive convex coding problem considered here is quite broad. Examples include

$$f(l_i, p_i) = p_i l_i^a \qquad (\varphi(x) = x^a)$$

for $a \geq 1$, the moment penalty; see, e.g., [8, pp. 121–122]. Although efficient solutions have been given for $a = 1$ (the Huffman case) and $a = 2$ (the quadratic moment), no polynomial-time algorithms have been proposed for the general case.

The quadratic moment was considered by Larmore in [6] as a special case of the quadratic problem (2), which is perhaps the case of greatest relevance. Restating this problem in terms of f,

$$f(l_i, p_i) = p_i(\alpha x + \beta x^2) \qquad (\varphi(x) = \alpha x + \beta x^2).$$

This was solved with cubic space and time complexity as a step in solving a problem related to message delay. This larger problem, treated first by Humblet [3] then Flores [9], was solved with an $O(n^5)$-time $O(n^3)$-space algorithm that can be altered to become an $O(n^4)$-time $O(n^2)$-space algorithm using methods in this paper.

Another quasiarithmetic penalty is the exponential penalty, that brought about by the cost function

$$f(l_i, p_i) = p_i D^{tl_i} \qquad (\varphi(x) = D^{tx}) \tag{4}$$

for $t > 0$, D being the size of the output alphabet. This was previously proposed by Campbell [4] and algorithmically solved as an extension of Huffman's algorithm (and thus with linear time and space for sorted probability inputs) in [3], [7], [10], [11]. As previously indicated, in [3], [7] this is a step in minimizing the probability of buffer overflow in a queueing system. Thus the quasiarithmetic framework includes the two queueing-related source coding problems discussed in [3].

A related problem is that with the concave cost function

$$f(l_i, p_i) = p_i(1 - D^{tl_i}) \qquad (\varphi(x) = 1 - D^{tx})$$

for $t < 0$, which has a similar solution [7]. This problem relates to a problem in [12] which is based on a scenario presented by Rényi in [13].

Whereas all of the above, being continuous in l_i and linear in p_i, are within the class of cases considered by Campbell, the following convex problem is not, in that its range includes infinity. Suppose we want the best code possible with the constraint that all codes must fit into a structure with $l_{\max}$ symbols. If our measure of the "best code" is linear, then the appropriate penalty is

$$f(l_i, p_i) = \begin{cases} p_i l_i, & l_i \leq l_{\max} \\ \infty, & l_i > l_{\max} \end{cases} \tag{5}$$

for some fixed $l_{\max} \geq \lceil \log_D n \rceil$. This describes the length-limited linear penalty, an objective solved by several algorithms; see, e.g., [14]–[23] or [2] for a comprehensive list. Most notable among these are the efficient Package-Merge algorithm [16] for $D = 2$ and a less efficient dynamic programming approach for $D > 2$ [15]. In this paper the Package-Merge approach is extended to a generalized quasiarithmetic convex coding algorithm.

Note that if the measure of a "best code" is nonlinear, a combination of penalties should be used where length is limited. For example, if we wish to minimize the probability of buffer overflow in a queueing system with a limited length constraint, we should combine (4) and (5):

$$f(l_i, p_i) = \begin{cases} p_i D^{tl_i}, & l_i \leq l_{\max} \\ \infty, & l_i > l_{\max}. \end{cases} \tag{6}$$

This problem can be solved via dynamic programming in a manner similar to [14], but this approach takes $\Omega(n^2 l_{\max})$

time and $\Omega(n^2)$ space for $D = 2$ and greater complexity for $D > 2$ [15]. Our approach improves on this considerably.

In addition to the above problems with previously known applications — and penalties which result from combining these problems — one might want to solve for a different utility function in order to find a compromise among these applications or another trade-off of codeword lengths. These functions need not be like Campbell's in that they need not be linear in $\boldsymbol{p}$; for example, consider

$$f(l_i, p_i) = (1 - p_i)^{-l_i}.$$

Although the author knows of no use for this particular cost function, it is notable as corresponding to one of the simplest convex-cost penalties of the form (3).

III. PROPERTIES

A. Bounds and the Subtranslatory Property

Campbell's quasiarithmetic penalty formulation can be restated as follows:

$$
\begin{array}{ll}
\text{Given} & \boldsymbol{p} = (p_1, \dots, p_n), \ p_i > 0, \\
& \sum_i p_i = 1; \\
& \text{convex, monotonically increasing} \\
& \varphi : \mathbb{R}_+ \to \mathbb{R}_+ \\
\text{Minimize}_{\{l\}} & L(\boldsymbol{p}, \boldsymbol{l}, \varphi) = \sum_i p_i \varphi(l_i) \\
\text{subject to} & \sum_i 2^{-l_i} \leq 1; \\
& l_i \in \mathbb{Z}_+.
\end{array}
\tag{7}
$$

In the case of linear φ, the integer constraint is often removed to obtain bounds related to entropy, as we do in the nonlinear case:

$$
\begin{array}{ll}
\text{Given} & \boldsymbol{p} = (p_1, \dots, p_n), \ p_i > 0, \\
& \sum_i p_i = 1; \\
& \text{convex, monotonically increasing} \\
& \varphi : \mathbb{R}_+ \to \mathbb{R}_+ \\
\text{Minimize}_{\{l\}} & L(\boldsymbol{p}, \boldsymbol{l}, \varphi) = \sum_i p_i \varphi(l_i) \\
\text{subject to} & \sum_i 2^{-l_i} \leq 1; \\
& l_i \in \mathbb{R}_+.
\end{array}
\tag{8}
$$

Note that, given $\boldsymbol{p}$ and φ, $L^\dagger$, the minimum for the relaxed (real-valued) problem (8), will necessarily be less than or equal to L^*, the minimum for the original (integer-constrained) problem (7). Let $\boldsymbol{l}^\dagger$ and $\boldsymbol{l}^*$ be corresponding minimizing values for the relaxed and constrained problems, respectively. Restating, and adding a fifth definition:

$$
\begin{array}{rcl}
L^* & \triangleq & \min_{\substack{\sum_i D^{-l_i} \leq 1, \\ l_i \in \mathbb{Z}_+}} L(\boldsymbol{p}, \boldsymbol{l}, \varphi) \\[1em]
\boldsymbol{l}^* & \triangleq & \arg \min_{\substack{\sum_i D^{-l_i} \leq 1, \\ l_i \in \mathbb{Z}_+}} L(\boldsymbol{p}, \boldsymbol{l}, \varphi) \\[1em]
L^\dagger & \triangleq & \min_{\substack{\sum_i D^{-l_i} \leq 1, \\ l_i \in \mathbb{R}_+}} L(\boldsymbol{p}, \boldsymbol{l}, \varphi) \\[1em]
\boldsymbol{l}^\dagger & \triangleq & \arg \min_{\substack{\sum_i D^{-l_i} \leq 1, \\ l_i \in \mathbb{R}_+}} L(\boldsymbol{p}, \boldsymbol{l}, \varphi) \\[1em]
\boldsymbol{l}^\ddagger & \triangleq & (\lceil l_1^\dagger \rceil, \lceil l_2^\dagger \rceil, \dots, \lceil l_n^\dagger \rceil).
\end{array}
$$

This is a slight abuse of $\arg \min$ notation since L^* could have multiple corresponding optimal length distributions ($\boldsymbol{l}^*$). However, this is not a problem, as any such value will suffice. Note too that $\boldsymbol{l}^\ddagger$ satisfies the Kraft inequality and the integer constraint, and thus $L(\boldsymbol{p}, \boldsymbol{l}^\ddagger, \varphi) \geq L^*$.

We obtain bounds for the optimal solution by noting that, since φ is monotonically increasing,

$$
\begin{array}{rcl}
\varphi^{-1}\left(\sum_i p_i \varphi(l_i^\dagger) \right) & \leq & \varphi^{-1}\left(\sum_i p_i \varphi(l_i^*) \right) \\[0.6em]
& \leq & \varphi^{-1}\left(\sum_i p_i \varphi(l_i^\ddagger) \right) \\[0.6em]
& < & \varphi^{-1}\left(\sum_i p_i \varphi(l_i^\dagger + 1) \right).
\end{array}
\tag{9}
$$

These bounds are similar to Shannon redundancy bounds for Huffman coding. In the linear/Shannon case, $l_i^\dagger = -\log_2 p_i$, so the last expression is $\sum_i p_i(l_i^\dagger + 1) = 1 + \sum_i p_i l_i^\dagger = 1 + H(\boldsymbol{p})$, where $H(\boldsymbol{p})$ is the Shannon entropy, so $H(\boldsymbol{p}) \leq \sum_i p_i l_i^* < 1 + H(\boldsymbol{p})$. These Shannon bounds can be extended to quasiarithmetic problems by first defining φ-entropy as follows:

Definition 3: Generalized entropy or *φ-entropy* is

$$
H(\boldsymbol{p}, \varphi) \triangleq \inf_{\substack{\sum_i D^{-l_i} \leq 1, \\ l_i \in \mathbb{R}_+}} L(\boldsymbol{p}, \boldsymbol{l}, \varphi)
\tag{10}
$$

where here infimum is used because this definition applies to codes with infinite, as well as finite, input alphabets [4].

Campbell defined this as a generalized entropy [4]; we go further, by asking which cost functions, φ, have the following property:

$$
H(\boldsymbol{p}, \varphi) \leq L(\boldsymbol{p}, \boldsymbol{l}^*, \varphi) < 1 + H(\boldsymbol{p}, \varphi).
\tag{11}
$$

These bounds exist for the exponential case (4) with $H(\boldsymbol{p}, \varphi) = H_\alpha(\boldsymbol{p})$, where $\alpha \triangleq (1 + t)^{-1}$, and $H_\alpha(\boldsymbol{p})$ denotes Rényi α-entropy [24]. The bounds extend to exponential costs because they share with the linear costs (and only those costs) a property known as the *translatory* property, described by Aczél [25], among others:

Definition 4: A cost function φ (and its associated penalty) is *translatory* if, for any $\boldsymbol{l} \in \mathbb{R}_+^n$, probability mass function $\boldsymbol{p}$, and $c \in \mathbb{R}_+$,

$$L(\boldsymbol{p}, \boldsymbol{l} + c, \varphi) = L(\boldsymbol{p}, \boldsymbol{l}, \varphi) + c$$

where $\boldsymbol{l} + c$ denotes adding c to each l_i in $\boldsymbol{l}$ [25].

We broaden the collection of penalty functions satisfying such bounds by replacing the translatory equality with an inequality, introducing the concept of a *subtranslatory* penalty:

Definition 5: A cost function φ (and its associated penalty) is *subtranslatory* if, for any $\boldsymbol{l} \in \mathbb{R}_+^n$, probability mass function $\boldsymbol{p}$, and $c \in \mathbb{R}_+$,

$$L(\boldsymbol{p}, \boldsymbol{l} + c, \varphi) \leq L(\boldsymbol{p}, \boldsymbol{l}, \varphi) + c.$$

For such a penalty, (11) still holds.

If φ obeys certain regularity requirements, then we can introduce a necessary and sufficient condition for it to be subtranslatory. Suppose that the invertible function $\varphi : \mathbb{R}_+ \to \mathbb{R}_+$ is real analytic over a relevant compact interval. We might choose this interval to be, for example, $\mathcal{A} = [\delta, 1/\delta]$ for some $\delta \in (0, 1)$. (Let $\delta \to 0$ to show the following argument is

valid over all $\mathbb{R}_+$.) We assume φ^{-1} is also real analytic (with respect to interval $\varphi(\mathcal{A})$). Thus all derivatives of the function and its inverse are bounded.

Theorem 1: Given monotonically increasing real analytic cost function φ and its real analytic inverse φ^{-1}, φ is subtranslatory if and only if, for all positive l and all positive p summing to 1,

$$\sum_i p_i \varphi'(l_i) \le \varphi'\left(\varphi^{-1}\left(\sum_i p_i \varphi(l_i)\right)\right) \tag{12}$$

where φ' is the derivative of φ.

Proof: First note that φ, φ^{-1}, φ', and $(\varphi^{-1})'$ are all positive over their domain. Thus inequality (12) is equivalent to

$$\left(\sum_i p_i \varphi'(l_i)\right) \cdot (\varphi^{-1})'\left(\sum_i p_i \varphi(l_i)\right) \le 1. \tag{13}$$

We show that, when (13) is true everywhere, φ is subtranslatory, and then we show the converse. Let $\epsilon > 0$. Using power expansions of the form

$$g(x) + \epsilon g'(x) = g(x + \epsilon) \pm O(\epsilon^2)$$

on φ and φ^{-1},

$$\varphi^{-1}\left(\sum_i p_i \varphi(l_i)\right) + \epsilon$$
$$\overset{(a)}{\ge} \varphi^{-1}\left(\sum_i p_i \varphi(l_i)\right)$$
$$+ \epsilon \cdot \left(\sum_i p_i \varphi'(l_i)\right) \cdot (\varphi^{-1})'\left(\sum_i p_i \varphi(l_i)\right)$$
$$\overset{(b)}{=} \varphi^{-1}\left(\sum_i p_i \varphi(l_i) + \epsilon \cdot \sum_i p_i \varphi'(l_i)\right) \pm O(\epsilon^2)$$
$$\overset{(c)}{=} \varphi^{-1}\left(\sum_i p_i \varphi(l_i + \epsilon) \pm O(\epsilon^2)\right) \pm O(\epsilon^2)$$
$$\overset{(d)}{=} \varphi^{-1}\left(\sum_i p_i \varphi(l_i + \epsilon)\right) \pm O(\epsilon^2). \tag{14}$$

Step (a) is due to (13), step (b) due to the power expansion on φ^{-1}, step (c) due to the power expansion on φ, and step (d) due to the power expansion on φ^{-1} (where the bounded derivative of φ^{-1} allows for the asymptotic term to be brought outside the function).

Next, evoke the above inequality c/ϵ times:

$$\varphi^{-1}\left(\sum_i p_i \varphi(l_i + c)\right)$$
$$\le \epsilon + \varphi^{-1}\left(\sum_i p_i \varphi(l_i + c - \epsilon)\right) \pm O(\epsilon^2)$$
$$\le \cdots$$
$$\le \epsilon \left\lfloor \frac{c}{\epsilon} \right\rfloor + \varphi^{-1}\left(\sum_i p_i \varphi(l_i) + c - \epsilon \left\lfloor \frac{c}{\epsilon} \right\rfloor\right) \pm O(\epsilon)$$
$$\le c + \varphi^{-1}\left(\sum_i p_i \varphi(l_i)\right) \pm O(\epsilon). \tag{15}$$

Taking $\epsilon \to 0$,

$$\varphi^{-1}\left(\sum_i p_i \varphi(l_i + c)\right) \le c + \varphi^{-1}\left(\sum_i p_i \varphi(l_i)\right).$$

Thus, the fact of (12) is sufficient to know that the penalty is subtranslatory.

To prove the converse, suppose $\sum_i p_i \varphi'(l_i) > \varphi'\left(\varphi^{-1}\left(\sum_i p_i \varphi(l_i)\right)\right)$ for some valid l and p. Because φ is analytic, continuity implies that there exist $\delta_0 > 0$ and $\epsilon_0 > 0$ such that

$$\sum_i p_i \varphi'(l_i') \ge (1 + \delta_0) \cdot \varphi'\left(\varphi^{-1}\left(\sum_i p_i \varphi(l_i')\right)\right)$$

for all $l' \in [l, l + \epsilon_0)$. The chain of inequalities above reverse in this range with the additional multiplicative constant. Thus (14) becomes

$$\varphi^{-1}\left(\sum_i p_i \varphi(l_i')\right) + (1 + \delta_0)\epsilon$$
$$\le \varphi^{-1}\left(\sum_i p_i \varphi(l_i' + \epsilon)\right) \pm O(\epsilon^2)$$

for $l' \in [l, l + \epsilon_0)$, and (15) becomes, for any $c \in (0, \epsilon_0)$,

$$\varphi^{-1}\left(\sum_i p_i \varphi(l_i + c)\right)$$
$$\ge (1 + \delta_0)c + \varphi^{-1}\left(\sum_i p_i \varphi(l_i)\right) \pm O(\epsilon)$$

which, taking $\epsilon \to 0$, similarly leads to

$$\varphi^{-1}\left(\sum_i p_i \varphi(l_i + c)\right)$$
$$\ge (1 + \delta_0)c + \varphi^{-1}\left(\sum_i p_i \varphi(l_i)\right)$$
$$> c + \varphi^{-1}\left(\sum_i p_i \varphi(l_i)\right)$$

and thus the subtranslatory property fails and the converse is proved. ∎

Therefore, for φ satisfying (12), we have the bounds of (11) for the optimum solution. Note that the right-hand side of (12) may also be written $\varphi'(L(p, l, \varphi))$; thus (12) indicates that the average derivative of φ at the codeword length values is at most the derivative of φ at the value of the penalty for those length values.

The linear and exponential penalties satisfy these equivalent inequalities with equality. Another family of cost functions that satisfies the subtranslatory property is $\varphi(l_i) = l_i^a$ for fixed $a \ge 1$, which corresponds to

$$L(p, l, \varphi) = \left(\sum_i p_i l_i^a\right)^{1/a}.$$

Proving this involves noting that Lyapunov's inequality for moments of a random variable yields

$$\left(\sum_i p_i l_i^{a-1} \right)^{\frac{1}{a-1}} \leq \left(\sum_i p_i l_i^a \right)^{\frac{1}{a}}$$

which leads to

$$a \cdot \left(\sum_i p_i l_i^{a-1} \right) \leq a \cdot \left(\sum_i p_i l_i^{a-1} \right)^{\frac{a-1}{a}}$$

which, because $\varphi'(x) = a x^{a-1}$, is

$$\sum_i p_i \varphi'(l_i) \leq \varphi' \left(\varphi^{-1} \left(\sum_i p_i \varphi(l_i) \right) \right)$$

the inequality we desire.

Another subtranslatory penalty is the quadratic quasiarithmetic penalty of (2), in which

$$\varphi(x) = \alpha x + \beta x^2$$

for $\alpha, \beta \geq 0$. This has already been shown for $\beta = 0$; when $\beta > 0$,

$$\varphi'(x) = \alpha + 2\beta x$$
$$\varphi^{-1}(x) = \sqrt{\left(\frac{\alpha}{2\beta} \right)^2 - \frac{x}{\beta}} - \frac{\alpha}{2\beta}$$
$$L(\boldsymbol{p}, \boldsymbol{l}, \varphi) = \sqrt{\left(\frac{\alpha}{2\beta} \right)^2 + \sum_i p_i \left(\frac{\alpha}{\beta} l_i + l_i^2 \right)} - \frac{\alpha}{2\beta}.$$

We achieve the desired inequality through algebra:

$$\sum_i p_i l_i^2 \geq \left(\sum_i p_i l_i \right)^2$$
$$\alpha^2 + 4\beta \sum_i p_i (\alpha l_i + \beta l_i^2) \geq \left(\sum_i p_i (\alpha + 2\beta l_i) \right)^2$$
$$\sqrt{\alpha^2 + 4\beta \sum_i p_i (\alpha l_i + \beta l_i^2)} \geq \sum_i p_i (\alpha + 2\beta l_i)$$
$$\varphi'(L(\boldsymbol{p}, \boldsymbol{l}, \varphi)) \geq \sum_i p_i \varphi'(l_i).$$

We thus have an important property that holds for several cases of interest.

One might be tempted to conclude that every φ — or every convex and/or concave φ — is subtranslatory. However, this is easily disproved. Consider convex $\varphi(x) = x^3 + 11x$. Using Cardano's formula, it is easily seen that (12) does not hold for $\boldsymbol{p} = (\frac{1}{3}, \frac{2}{3})$ and $\boldsymbol{l} = (\frac{1}{2}, 1)$. The subtranslatory test also fails for $\varphi(x) = \sqrt{x}$. Thus we must test any given penalty for the subtranslatory property in order to use the redundancy bounds.

B. Existence of an Optimal Code

Because all costs are positive, the redundancy bounds that are a result of a subtranslatory penalty extend to infinite alphabet codes in a straightforward manner. These bounds thus show that a code with finite penalty exists if and only if the generalized entropy is finite, a property we extend to nonsubtranslatory penalties in the next subsection. However, 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. Must there exist an optimal code, or can there be an infinite sequence of codes of decreasing penalty without a code achieving the limit penalty value?

Fortunately, the answer is the former, as the existence results of Linder, Tarokh, and Zeger in [26] can be extended to quasiarithmetic penalties. Consider continuous strictly monotonic $\varphi : \mathbb{R}_+ \to \mathbb{R}_+$ (as proposed by Campbell) and $\boldsymbol{p} = (p_1, p_2, \ldots)$ such that

$$L^*(\boldsymbol{p}, \varphi) \triangleq \inf_{\substack{\sum_i D^{-l_i} \leq 1, \\ l_i \in \mathbb{Z}}} \varphi^{-1} \left(\sum_{i=1}^{\infty} p_i \varphi(l_i) \right) \qquad (16)$$

is finite. Consider, for an arbitrary $n \in \mathbb{Z}_+$, optimizing for φ with weights

$$\boldsymbol{p}^{(n)} \triangleq (p_1, p_2, \ldots, p_n, 0, 0, \ldots).$$

(We call the entries to this distribution "weights" because they do not necessarily add up to 1.) Denote the optimal code a *truncated code*, one with codeword lengths

$$\boldsymbol{l}^{(n)} \triangleq \{l_1^{(n)}, l_2^{(n)}, \ldots, l_n^{(n)}, \infty, \infty, \ldots\}.$$

Thus, for convenience, $l_i^{(j)} = \infty$ for $i > j$. These lengths are also optimal for $(\sum_{j=1}^n p_j)^{-1} \cdot \boldsymbol{p}^{(n)}$, the distribution of normalized weights.

Following [26], we say that a sequence of codeword length distributions $\boldsymbol{l}^{(1)}, \boldsymbol{l}^{(2)}, \boldsymbol{l}^{(3)}, \ldots$ *converges* to an infinite prefix code with codeword lengths $\boldsymbol{l} = \{l_1, l_2, \ldots\}$ if, for each i, the ith length in each distribution in the sequence is eventually l_i (i.e., if each sequence converges to l_i).

Theorem 2: Given quasiarithmetic increasing φ and $\boldsymbol{p}$ such that $L^*(\boldsymbol{p}, \varphi)$ is finite, the following hold:
1) There exists a sequence of truncated codeword lengths that converges to optimal codeword lengths for $\boldsymbol{p}$; thus the infimum is achievable.
2) Any optimal code for $\boldsymbol{p}$ must satisfy the Kraft inequality with equality.

Proof: Because here we are concerned only with cases in which the first length is at least 1, we may restrict ourselves to the domain $[\varphi^{-1}(p_1 \varphi(1)), \infty)$. Recall

$$L^*(\boldsymbol{p}, \varphi) = \inf_{\substack{\sum_i D^{-l_i} \leq 1, \\ l_i \in \mathbb{Z}}} \varphi^{-1} \left(\sum_{i=1}^{\infty} p_i \varphi(l_i) \right) < \infty.$$

Then there exists near-optimal $\boldsymbol{l}' = \{l_1', l_2', l_3', \ldots\} \in \mathbb{Z}_+^{\infty}$ such that

$$\varphi^{-1} \left(\sum_{i=1}^{\infty} p_i \varphi(l_i') \right) < L^*(\boldsymbol{p}, \varphi) + 1 \text{ and } \sum_{i=1}^{\infty} D^{-l_i'} \leq 1$$

and thus, for any integer n,

$$\varphi^{-1}\left(\sum_{i=1}^{n} p_i\varphi(l'_i)\right) < L^*(\boldsymbol{p},\varphi) + 1 \text{ and } \sum_{i=1}^{n} D^{-l'_i} < 1.$$

So, using this to approximate the behavior of a minimizing $\boldsymbol{l}^{(n)}$, we have

$$\begin{aligned}\varphi^{-1}\left(\sum_{i=1}^{n} p_i\varphi(l_i^{(n)})\right) &\leq \varphi^{-1}\left(\sum_{i=1}^{n} p_i\varphi(l'_i)\right) \\ &< L^*(\boldsymbol{p},\varphi) + 1\end{aligned}$$

yielding an upper bound on terms

$$\begin{aligned}p_j\varphi(l_j^{(n)}) &\leq \sum_{i=1}^{n} p_i\varphi(l_i^{(n)}) \\ &< \varphi(L^*(\boldsymbol{p},\varphi)+1)\end{aligned}$$

for all j. This implies

$$l_j^{(n)} < \varphi^{-1}\left(\frac{\varphi(L^*(\boldsymbol{p},\varphi)+1)}{p_j}\right).$$

Thus, for any $i \in \mathbb{Z}_+$, the sequence $l_i^{(1)}, l_i^{(2)}, l_i^{(3)}, \ldots$ is bounded for all $l_i^{(j)} \neq \infty$, and thus has a finite set of values (including ∞). It is shown in [26] that this suffices for the desired convergence, but for completeness a slightly altered proof follows.

Because each sequence $l_i^{(1)}, l_i^{(2)}, l_i^{(3)}, \ldots$ has a finite set of values, every infinite indexed subsequence for a given i has a convergent subsequence. An inductive argument implies that, for any k, there exists a subsequence indexed by n_j^k such that $l_i^{(n_1^k)}, l_i^{(n_2^k)}, l_i^{(n_3^k)}, \ldots$ converges for all $i \leq k$, where $l_i^{(n_1^k)}, l_i^{(n_2^k)}, l_i^{(n_3^k)}, \ldots$ is a subsequence of $l_i^{(n_1^{k'})}, l_i^{(n_2^{k'})}, l_i^{(n_3^{k'})}, \ldots$ for $k' \leq k$. Codeword length distributions $\boldsymbol{l}^{(n_1^1)}, \boldsymbol{l}^{(n_2^2)}, \boldsymbol{l}^{(n_3^3)}, \ldots$ (which we call $\boldsymbol{l}^{(n_1)}, \boldsymbol{l}^{(n_2)}, \boldsymbol{l}^{(n_3)}, \ldots$) thus converge to the codeword lengths of an infinite code $\widehat{C}$ with codeword lengths $\widehat{\boldsymbol{l}} = \{\widehat{l}_1, \widehat{l}_2, \widehat{l}_3, \ldots\}$. Clearly each codeword length distribution satisfies the Kraft inequality. The limit does as well then; were it exceeded, we could find i' such that

$$\sum_{i=1}^{i'} D^{-\widehat{l}_i} > 1$$

and thus n' such that

$$\sum_{i=1}^{i'} D^{-l_i^{(n')}} > 1$$

causing a contradiction.

We now show that $\widehat{C}$ is optimal. Let $\{\lambda_1, \lambda_2, \lambda_3 \ldots\}$ be the codeword lengths of an arbitrary prefix code. For every k, there is a $j \geq k$ such that $\widehat{l}_i = l_i^{(n_m)}$ for any $i \leq k$ if $m \geq j$.

Due to the optimality of each $\boldsymbol{l}^{(n)}$, for all $m \geq j$:

$$\begin{aligned}\sum_{i=1}^{k} p_i\varphi(\widehat{l}_i) &= \sum_{i=1}^{k} p_i\varphi(l_i^{(n_m)}) \\ &\leq \sum_{i=1}^{n_m} p_i\varphi(l_i^{(n_m)}) \\ &\leq \sum_{i=1}^{n_m} p_i\varphi(\lambda_i) \\ &\leq \sum_{i=1}^{\infty} p_i\varphi(\lambda_i)\end{aligned}$$

and, taking $k \to \infty$, $\sum_i p_i\varphi(\widehat{l}_i) \leq \sum_i p_i\varphi(\lambda_i)$, leading directly to $\varphi^{-1}\left(\sum_i p_i\varphi(\widehat{l}_i)\right) \leq \varphi^{-1}\left(\sum_i p_i\varphi(\lambda_i)\right)$ and the optimality of $\widehat{C}$.

Suppose the Kraft inequality is not satisfied with equality for optimal codeword lengths $\widehat{\boldsymbol{l}} = \{\widehat{l}_1, \widehat{l}_2, \ldots\}$. We can then produce a strictly superior code. There is a $k \in \mathbb{Z}_+$ such that $D^{-l_k+1} + \sum_i D^{-l_i} \leq 1$. Consider code $\{\widehat{l}_1, \widehat{l}_2, \ldots, \widehat{l_{k-1}}, \widehat{l_k} - 1, \widehat{l_{k+1}}, \widehat{l_{k+2}}, \ldots\}$. This code satisfies the Kraft inequality and has penalty $\varphi^{-1}\left(\sum_i p_i\varphi(\widehat{l}_i) + p_k(\varphi(\widehat{l}_k - 1) - \varphi(\widehat{l}_k))\right) < \varphi^{-1}\left(\sum_i p_i\varphi(\widehat{l}_i)\right)$. Thus $\widehat{\boldsymbol{l}}$ is not optimal. Therefore the Kraft inequality must be satisfied with equality for optimal infinite codes. ∎

Note that this theorem holds not just for subtranslatory penalties, but for any quasiarithmetic penalty.

C. Finiteness of Penalty for an Optimal Code

Recall the definition of (10),

$$H(\boldsymbol{p},\varphi) = \inf_{\substack{\sum_i D^{-l_i} \leq 1, \\ l_i \in \mathbb{R}_+}} \varphi^{-1}\left(\sum_i p_i\varphi(l_i)\right)$$

for $\varphi: \mathbb{R}_+ \to \mathbb{R}_+$.

Theorem 3: If $H(\boldsymbol{p},\varphi)$ is finite and either φ is subtranslatory or $\varphi(x+1) = O(\varphi(x))$ (which includes all concave and all polynomial φ), then the coding problem of (16),

$$L^*(\boldsymbol{p},\varphi) = \inf_{\substack{\sum_i D^{-l_i} \leq 1, \\ l_i \in \mathbb{Z}}} \varphi^{-1}\left(\sum_i p_i\varphi(l_i)\right)$$

has a minimizing $\boldsymbol{l}^*$ resulting in a finite value for $L^*(\boldsymbol{p},\varphi)$.

Proof: If φ is subtranslatory, then $L^*(\boldsymbol{p},\varphi) < 1 + H(\boldsymbol{p},\varphi) < \infty$. If $\varphi(x+1) = O(\varphi(x))$, then there are $\alpha, \beta > 0$ such that $\varphi(x+1) < \alpha + \beta\varphi(x)$ for all x. Then

$$\begin{aligned}&\varphi^{-1}\left(\sum_i p_i\varphi(l_i+1)\right) \\ &< \varphi^{-1}\left(\sum_i p_i(\alpha + \beta\varphi(l_i))\right) \\ &= \varphi^{-1}\left(\alpha + \beta\sum_i p_i\varphi(l_i)\right).\end{aligned}$$

So

$$L^*(\boldsymbol{p}, \varphi)$$
$$< L(\boldsymbol{p}, \boldsymbol{l}^\dagger + 1, \varphi)$$
$$< \varphi^{-1}\left(\alpha + \beta\varphi\left(H(\boldsymbol{p}, \varphi)\right)\right)$$
$$< \infty$$

and the infimum, which we know to also be a minimum, is finite. ∎

IV. ALGORITHMS

A. Nodeset Notation

We now examine algorithms for finding minimum penalty codes for convex cases with finite alphabets. We first present a notation for codes based on an approach of Larmore [6]. This notation is an alternative to the well known code tree notation, e.g., [27], and it will be the basis for an algorithm to solve the generalized quasiarithmetic (and thus Campbell's quasiarithmetic) convex coding problem.

In the literature nodeset notation is generally used for binary alphabets, not for general alphabet coding. Although we briefly sketch how to adapt this technique to general output alphabet coding at the end of Subsection IV-E, an approach fully explained in [28], until then we concentrate on the binary case ($D = 2$).

The key idea: Each node (i, l) represents both the share of the penalty $\tilde{L}(\boldsymbol{p}, \boldsymbol{l}, f)$ (weight) and the share of the Kraft sum $\kappa(\boldsymbol{l})$ (width) assumed for the lth bit of the ith codeword. If we show that total weight is an increasing function of the penalty and show a one-to-one correspondence between optimal nodesets and optimal codes, we can reduce the problem to an efficiently solvable problem, the Coin Collector's problem.

In order to do this, we first assume bounds on the maximum codeword length of possible solutions, e.g., the maximum unary codeword length of $n - 1$. Alternatively, bounds might be explicit in the definition of the problem. Consider for example the length-limited coding problems of (5) and (6), upper bounded by $l_{\max}$. A third possibility is that maximum length is implicit in some property of the set of optimal solutions [29]–[31]; we explore this in Subsection IV-E.

We therefore restrict ourselves to codes with n codewords, none of which has greater length than $l_{\max}$, where $l_{\max} \in [\lceil \log_2 n \rceil, n - 1]$. With this we now introduce the *nodeset* notation for binary coding:

Definition 6: A *node* is an ordered pair of integers (i, l) such that $i \in \{1, \ldots, n\}$ and $l \in \{1, \ldots, l_{\max}\}$. Call the set of all $nl_{\max}$ possible nodes I. Usually I is arranged in a grid; see example in Fig. 1. The set of nodes, or *nodeset*, corresponding to item i (assigned codeword c_i with length l_i) is the set of the first l_i nodes of column i, that is, $\eta_l(i) \triangleq \{(j, l) \mid j = i, \ l \in \{1, \ldots, l_i\}\}$. The nodeset corresponding to length distribution $\boldsymbol{l}$ is $\eta(\boldsymbol{l}) \triangleq \bigcup_i \eta_l(i)$; this corresponds to a set of n codewords, a code. We say a node (i, l) has *width* $\rho(i, l) \triangleq 2^{-l}$ and *weight* $\mu(i, l) \triangleq f(l, p_i) - f(l - 1, p_i)$, as in the example in Fig. 1.

If I has a subset N that is a valid nodeset, then it is straightforward to find the corresponding length distribution and thus a code. We can find an optimal valid nodeset using the Coin Collector's problem.

B. The Coin Collector's Problem

Let $2^{\mathbb{Z}}$ denote the set of all integer powers of two. The Coin Collector's problem of size m considers m "coins" with width $\rho_i \in 2^{\mathbb{Z}}$; one can think of width as coin face value, e.g., $\rho_i = \frac{1}{4}$ for a quarter dollar (25 cents). Each coin also has weight $\mu_i \in \mathbb{R}$. The final problem parameter is total width, denoted t. The problem is then:

$$\text{Minimize}_{\{B \subseteq \{1,\ldots,m\}\}} \quad \sum_{i \in B} \mu_i \qquad (17)$$
$$\text{subject to} \quad \sum_{i \in B} \rho_i = t.$$

We thus wish to choose coins with total width t such that their total weight is as small as possible. This problem is an input-restricted variant of the knapsack problem, which, in general, is NP-hard; no polynomial-time algorithms are known for such NP-hard problems [32], [33]. However, given sorted inputs, a linear-time solution to (17) was proposed in [16]. The algorithm in question is called the *Package-Merge algorithm*.

In the Appendix, we illustrate and prove a slightly simplified version of the Package-Merge algorithm. This algorithm allows us to solve the generalized quasiarithmetic convex coding problem (3). When we use this algorithm, we let $\mathcal{I}$ represent the m items along with their weights and widths. The optimal solution to the problem is a function of total width t and items $\mathcal{I}$. We denote this solution as $\text{CC}(\mathcal{I}, t)$ (read, "the [optimal] coin collection for $\mathcal{I}$ and t"). Note that, due to ties, this need not be unique, but we assume that one of the optimal solutions is chosen; at the end of Subsection IV-D, we discuss which of the optimal solutions is best to choose.

C. A General Algorithm

We now formalize the reduction from the generalized quasiarithmetic convex coding problem to the Coin Collector's problem.

We assert that any optimal solution N of the Coin Collector's problem for $t = n - 1$ on coins $\mathcal{I} = I$ is a nodeset for an optimal solution of the coding problem. This yields a suitable method for solving generalized quasiarithmetic convex penalties.

To show this reduction, first define $\rho(N)$ for any $N = \eta(\boldsymbol{l})$:

$$\rho(N) \triangleq \sum_{(i,l) \in N} \rho(i, l)$$
$$= \sum_{i=1}^{n} \sum_{l=1}^{l_i} 2^{-l}$$
$$= \sum_{i=1}^{n} \left(1 - 2^{-l_i}\right)$$
$$= n - \sum_{i=1}^{n} 2^{-l_i}$$
$$= n - \kappa(\boldsymbol{l}).$$

Because the Kraft inequality is $\kappa(\boldsymbol{l}) \leq 1$, $\rho(N)$ must lie in $[n-1, n)$ for prefix codes. The Kraft inequality is satisfied with equality at the left end of this interval. Optimal binary codes have this equality satisfied, since a strict inequality implies that the longest codeword length can be shortened by one, strictly
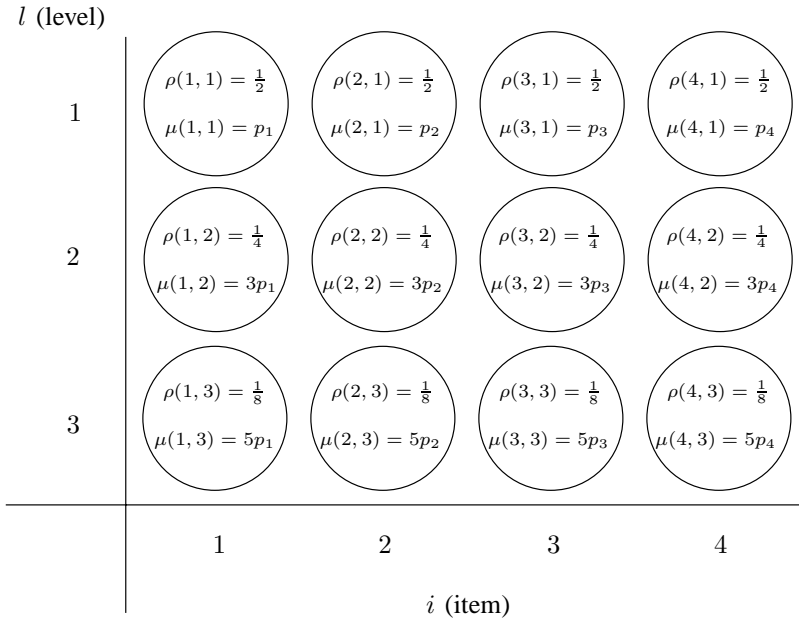
l (level)

Fig. 1.   The set of nodes I with widths $\{\rho(i,l)\}$ and weights $\{\mu(i,l)\}$ for $f(l_i,p_i)=p_i l_i^2$, $n=4$, $l_{\max}=3$

decreasing the penalty without violating the inequality. Thus the optimal solution has $\rho(N)=n-1$.

Also define:

$$M_f(l,p) \triangleq f(l,p)-f(l-1,p)$$
$$L_0(\boldsymbol{p},f) \triangleq \sum_{i=1}^{n} f(0,p_i)$$
$$\mu(N) \triangleq \sum_{(i,l)\in N} \mu(i,l).$$

Note that

$$\begin{aligned}
\mu(N) &= \sum_{(i,l)\in N} \mu(i,l) \\
&= \sum_{i=1}^{n}\sum_{l=1}^{l_i} M_f(l,p_i) \\
&= \sum_{i=1}^{n} f(l_i,p_i) - \sum_{i=1}^{n} f(0,p_i) \\
&= \tilde{L}(\boldsymbol{p},\boldsymbol{l},f) - L_0(\boldsymbol{p},f).
\end{aligned}$$

$L_0(\boldsymbol{p},f)$ is a constant given fixed penalty and probability distribution. Thus, if the optimal nodeset corresponds to a valid code, solving the Coin Collector's problem solves this coding problem. To prove the reduction, we need to prove that the optimal nodeset indeed corresponds to a valid code. We begin with the following lemma:

Lemma 1: Suppose that N is a nodeset of width $x2^{-k}+r$ where k and x are integers and $0<r<2^{-k}$. Then N has a subset R with width r. [16]

Proof: We use induction on the cardinality of the set. The base case $|N|=1$ is trivial since then $x=0$. Assume the lemma holds for all $|N|<n$, and suppose $|\tilde{N}|=n$. Let $\rho^*=\min_{j\in\tilde{N}}\rho_j$ and $j^*=\arg\min_{j\in\tilde{N}}\rho_j$. We can see ρ^* as

the smallest contribution to the width of $\tilde{N}$ and r as the portion of the binary expansion of the width of $\tilde{N}$ to the right of 2^{-k}. Then clearly r must be an integer multiple of ρ^*. If $r=\rho^*$, $R=\{j^*\}$ is a solution. Otherwise let $N'=\tilde{N}\backslash\{j^*\}$ (so $|N'|=n-1$) and let R' be the subset obtained from solving the lemma for set N' of width $r-\rho^*$. Then $R=R'\cup\{j^*\}$. ∎

We are now able to prove the main theorem:

Theorem 4: Suppose N is a solution of the Coin Collector's problem for $t=\rho(N)=n-1$ with $\rho(i,l)$ and $\mu(i,l)$ as defined above for $f(l,p)$ convex in l. Then there is a corresponding $\boldsymbol{l}^N$ such that $N=\eta(\boldsymbol{l}^N)$ and $\mu(N)=\min_{\boldsymbol{l}}\tilde{L}(\boldsymbol{p},\boldsymbol{l},f)-L_0(\boldsymbol{p},f)$.

Proof: By monotonicity of the penalty function, any optimal solution satisfies the Kraft inequality with equality. Thus all optimal length distribution nodesets have $\rho(\eta(\boldsymbol{l}))=n-1$. Suppose N is a solution to the Coin Collector's problem but is not a valid nodeset of a length distribution. Then there exists an (i,l) with $l>1$ such that $(i,l)\in N$ and $(i,l-1)\in I\backslash N$. Let $R'=N\cup\{(i,l-1)\}\backslash\{(i,l)\}$. Then $\rho(R')=n-1+2^{-l}$ and, due to convexity, $\mu(R')\leq\mu(N)$. Thus, using Lemma 1 with $k=0$, $x=n-1$, and $r=2^{-l}$, there exists an $R\subset R'$ such that $\rho(R)=2^{-l}$ and $\mu(R'\backslash R)<\mu(R')\leq\mu(N)$. Since we assumed N to be an optimal solution of the Coin Collector's problem, this is a contradiction, and thus any optimal solution of the Coin Collector's problem corresponds to an optimal length distribution. ∎

Note that the generality of this algorithm makes it trivially extensible to problems of the form $\sum_i f_i(l_i,p_i)$ for n different functions f_i. This might be applicable if we desire a nonlinear weighting for codewords — such as an additional utility weight — in addition to and possibly independent of codeword length and probability.

Because the Coin Collector's problem is linear in time and

problem type	time	space
flat, d.m.	$O(n \log n)$	$O(n \log n)$
space-optimized	$O(n \log n)$	$O(n)$
not flat, d.m.	$O(n^2)$	$O(n^2)$
space-optimized	$O(n^2)$	$O(n)$
flat, not d.m.	$O(n \log^2 n)$	$O(n \log n)$
not flat, not d.m.	$O(n^2 \log n)$	$O(n^2)$

TABLE I

COMPLEXITY FOR VARIOUS TYPES OF INPUTS (D.M. = DIFFERENTALLY
MONOTONIC)

space, the overall algorithm finds an optimal code in $O(nl_{\max})$ time and space for any "well-behaved" $f(l_i, p_i)$, that is, any f of the form specified for which same-width inputs would automatically be presorted by weight for the Coin Collector's problem.

The complexity of the algorithm in terms of n alone depends on the structure of both f and $\boldsymbol{p}$, because, if we can upper-bound the maximum length codeword, we can run the Package-Merge algorithm with fewer input nodes. In addition, if f is not "well-behaved," input to the Package-Merge algorithm might need to be sorted.

To quantify these behaviors, we introduce one definition and recall another:

Definition 7: A (coding) problem space is called a *flat class* if there exists a constant upper bound u such that $\frac{\max_i l_i}{\log n} < u$ for any solution $\boldsymbol{l}$.

For example, the space of linear Huffman coding problems with all $p_i \geq \frac{1}{2n}$ is a flat class. (This may be shown using [30].)

Recall Definition 1 given in Section I: A cost function $f(l, p)$ and its associated penalty $\tilde{L}$ are *differentially monotonic* or *d.m.* if, for every $l > 1$, whenever $f(l-1, p_i)$ is finite and $p_i > p_j$, $f(l, p_i) - f(l-1, p_i) > f(l, p_j) - f(l-1, p_j)$. This implies that f is continuous in $\boldsymbol{p}$ at all but a countable number of points. Without loss of generality, we consider only cases in which it is continuous everywhere.

If $f(l, p)$ is differentially monotonic, then there is no need to sort the input nodes for the algorithm. Otherwise, sorting occurs on $l_{\max}$ rows with $O(n \log n)$ on each row, $O(nl_{\max} \log n)$ total. Also, if the problem space is a flat class, $l_{\max}$ is $O(\log n)$; it is $O(n)$ in general. Thus time complexity for this solution ranges from $O(n \log n)$ to $O(n^2 \log n)$ with space requirement $O(n \log n)$ to $O(n^2)$; see Table I for details. As indicated in the table, space complexity can be reduced in differentially monotonic instances.

D. A Linear-Space Algorithm

Note that the length distribution returned by the algorithm need not have the property that $l_i \leq l_j$ whenever $i < j$. For example, if $p_i = p_j$, we are guaranteed no particular inequality relation between l_i and l_j since we did not specify a method for breaking ties. Also, even if all p_i were distinct, there are cost functions for which we would expect the inequality relation reversed from the linear case. An example of this is $f(l_i, p_i) = p_i^{-1} 2^{l_i}$, although this represents no practical problem that the author is aware of.

Practical cost functions will, given a probability distribution for nonincreasing p_i, generally have at least one optimal code of monotonically nondecreasing length. Differentially monotonicity is a sufficient condition for this, and we can improve upon the algorithm by insisting that the problem be differentially monotonic and all entries p_i in $\boldsymbol{p}$ be distinct; the latter condition we later relax. The resulting algorithm uses only linear space and quadratic time. First we need a definition:

Definition 8: A *monotonic* nodeset, N, is one with the following properties:

$$(i, l) \in N \Rightarrow (i+1, l) \in N \quad \text{for } i < n \qquad (18)$$
$$(i, l) \in N \Rightarrow (i, l-1) \in N \quad \text{for } l > 1. \qquad (19)$$

This definition is equivalent to that given in [16].

An example of a monotonic nodeset is the set of nodes enclosed by the dashed line in Fig. 2. Note that a nodeset is monotonic only if it corresponds to a length distribution $\boldsymbol{l}$ with lengths sorted in nondecreasing order.

Lemma 2: If a problem is differentially monotonic and monotonically increasing and convex in each l_i, and if $\boldsymbol{p}$ has no repeated values, then any optimal solution $N = \text{CC}(I, n-1)$ is monotonic.

Proof: The second monotonic property (19) was proved for optimal nodesets in Theorem 4, and the first is now proved with a simple exchange argument, as in [34, pp. 97–98]. Suppose we have optimal N that violates the first property (18). Then there exist unequal i and j such that $p_i < p_j$ and $l_i < l_j$ for optimal codeword lengths $\boldsymbol{l}$ ($N = \eta(\boldsymbol{l})$). Consider $\boldsymbol{l}'$ with lengths for symbols i and j interchanged. Then

$$\begin{aligned}
\tilde{L}(\boldsymbol{p}, \boldsymbol{l}', f) &- \tilde{L}(\boldsymbol{p}, \boldsymbol{l}, f) \\
&= \sum_k f(l'_k, p_k) - \sum_k f(l_k, p_k) \\
&= (f(l_j, p_j) - f(l_i, p_j)) - (f(l_j, p_i) - f(l_i, p_i)) \\
&= \sum_{l=l_i+1}^{l_j} (M_f(l, p_j) - M_f(l, p_i)) \\
&< 0
\end{aligned}$$

where we recall that $M_f(l, p) \triangleq f(l, p) - f(l-1, p)$ and the final inequality is due to differential monotonicity. However, this implies that $\boldsymbol{l}$ is not an optimal code, and thus we cannot have an optimal nodeset without monotonicity unless values in $\boldsymbol{p}$ are repeated. ∎

Taking advantage of monotonicity to trade off a constant factor of time for drastically reduced space complexity has been done in [6] for the case of the length-limited (linear) penalty (5). We now extend this to all convex differentially monotonic cases.

Note that the total width of items that are each less than or equal to width ρ is less than $2n\rho$. Thus, when we are processing items and packages of width ρ, fewer than $2n$ packages are kept in memory. The key idea in reducing space complexity is to keep only four attributes of each package in memory instead of the full contents. In this manner, we use linear space while retaining enough information to reconstruct the optimal nodeset in algorithmic postprocessing.

Define $l_{\text{mid}} \triangleq \lfloor \frac{1}{2}(l_{\max} + 1) \rfloor$. Package attributes allow us to divide the problem into two subproblems with total complexity

that is at most half that of the original problem. For each package S, we retain the following attributes:

1) $\mu(S) \triangleq \sum_{(i,l)\in S} \mu(i,l)$
2) $\rho(S) \triangleq \sum_{(i,l)\in S} \rho(i,l)$
3) $\nu(S) \triangleq |S \cap I_{\mathrm{mid}}|$
4) $\psi(S) \triangleq \sum_{(i,l)\in S \cap I_{\mathrm{hi}}} \rho(i,l)$

where $I_{\mathrm{hi}} \triangleq \{(i,l) \mid l > l_{\mathrm{mid}}\}$ and $I_{\mathrm{mid}} \triangleq \{(i,l) \mid l = l_{\mathrm{mid}}\}$. We also define $I_{\mathrm{lo}} \triangleq \{(i,l) \mid l < l_{\mathrm{mid}}\}$.

This retains enough information to complete the "first run" of the algorithm with $O(n)$ space. The result will be the package attributes for the optimal nodeset N. Thus, at the end of this first run, we know the value for $m = \nu(N)$, and we can consider N as the disjoint union of four sets, shown in Fig. 2:

1) A = nodes in $N \cap I_{\mathrm{lo}}$ with indices in $[1, n - m]$
2) B = nodes in $N \cap I_{\mathrm{lo}}$ with indices in $[n - m + 1, n]$
3) C = nodes in $N \cap I_{\mathrm{mid}}$
4) D = nodes in $N \cap I_{\mathrm{hi}}$.

Due to monotonicity of N, it is trivial that $C = [n - m + 1, n] \times \{l_{\mathrm{mid}}\}$ and $B = [n - m + 1, n] \times [1, l_{\mathrm{mid}} - 1]$. Note then that $\rho(C) = m 2^{-l_{\mathrm{mid}}}$ and $\rho(B) = m[1 - 2^{-(l_{\mathrm{mid}}-1)}]$. Thus we need merely to recompute which nodes are in A and in D.

Because D is a subset of I_{hi}, $\rho(D) = \psi(N)$ and $\rho(A) = \rho(N) - \rho(B) - \rho(C) - \rho(D)$. Given their respective widths, A is a minimal weight subset of $[1, n - m] \times [1, l_{\mathrm{mid}} - 1]$ and D is a minimal weight subset of $[n - m + 1, n] \times [l_{\mathrm{mid}} + 1, l_{\mathrm{max}}]$. The nodes at each level of A and D may be found by recursive calls to the algorithm. In doing so, we use only $O(n)$ space. Time complexity, however, remains the same; we replace one run of an algorithm on nl_{max} nodes with a series of runs, first one on nl_{max} nodes, then two on an average of at most $\frac{1}{4}nl_{\mathrm{max}}$ nodes each, then four on $\frac{1}{16}nl_{\mathrm{max}}$, and so forth. Formalizing this analysis:

Theorem 5: The above recursive algorithm for generalized quasiarithmetic convex coding has $O(nl_{\mathrm{max}})$ time complexity. [16]

Proof: As indicated, this recurrence relation is considered and proved in [16, pp. 472–473], but we analyze it here for completeness. To find time complexity, set up the following recurrence relation: Let $T(n, l)$ be the worst case time to find the minimal weight subset of $[1, n] \times [1, l]$ (of a given width), assuming the subset is monotonic. Then there exist constants c_1 and c_2 such that, if we define $\hat{l} \triangleq l_{\mathrm{mid}} - 1 \le \lfloor \frac{l}{2} \rfloor$ and $\check{l} \triangleq l - \hat{l} - 1 \le \lfloor \frac{l}{2} \rfloor$, and we let an adversary choose the corresponding $\hat{n} + \check{n} = n$,

$$
\begin{aligned}
T(n, l) &\le c_1 n & \text{for } l < 3 \\
T(n, l) &\le c_2 nl + T(\hat{n}, \hat{l}) + T(\check{n}, \check{l}) & \text{for } l \ge 3
\end{aligned}
$$

where $l < 3$ is the base case. Then $T(n, l) = O(\tau(n, l))$, where τ is any function satisfying the recurrence

$$
\begin{aligned}
\tau(n, l) &\ge c_1 n & \text{for } l < 3 \\
\tau(n, l) &\ge c_2 nl + \tau(\hat{n}, \tfrac{l}{2}) + \tau(n - \hat{n}, \tfrac{l}{2}) & \text{for } l \ge 3
\end{aligned}
$$

which $\tau(n, l) = (c_1 + 2c_2)nl$ does. Thus, the time complexity is $O(nl_{\mathrm{max}})$. ∎

The overall complexity is $O(n)$ space and $O(nl_{\mathrm{max}})$ time — $O(n \log n)$ considering only flat classes, $O(n^2)$ in general, as in Table I.

However, the assumption of distinct p_i's puts an undesirable restriction on our input. In their original (linear) algorithm, Larmore and Hirschberg suggest modifying the probabilities slightly to make them distinct [16], but this is unnecessarily inelegant, as the resulting algorithm has the drawbacks of possibly being slightly nonoptimal and being nondeterministic; that is, different implementations of the algorithm could result in the same input yielding different outputs. A deterministic variant of this approach could involve modifications by multiples of a suitably small variable $\epsilon > 0$ to make identical values distinct. In [35], another method of tie-breaking is presented for alphabetic length-limited codes. Here, we present a simpler alternative analogous to this approach, one which is both deterministic and applicable to all differentially monotonic instances.

Recall that $\boldsymbol{p}$ is a nonincreasing vector. Thus items of a given width are sorted for use in the Package-Merge algorithm; use this order for ties. For example, if we use the nodes in Fig. 1 — $n = 4$, $f(l, p) = pl^2$ — with probability $\boldsymbol{p} = (0.5, 0.2, 0.2, 0.1)$, then nodes $(4, 3)$ and $(3, 3)$ are the first to be paired, the tie between $(2, 3)$ and $(3, 3)$ broken by order. Thus, at any step, all identical-width items in one package have adjacent indices. Recall that packages of items will be either in the final nodeset or absent from it as a whole. This scheme then prevents any of the nonmonotonicity that identical p_i's might bring about.

In order to ensure that the algorithm is fully deterministic — whether or not the linear-space version is used — the manner in which packages and single items are merged must also be taken into account. We choose to merge nonmerged items before merged items in the case of ties, in a similar manner to the two-queue bottom-merge method of Huffman coding [27], [36]. Thus, in our example, the node $(1, 2)$ is chosen whereas the package of items $(4, 3)$ and $(3, 3)$ is not. This leads to the optimal length vector $\boldsymbol{l} = (2, 2, 2, 2)$, rather than $\boldsymbol{l} = (1, 2, 3, 3)$ or $\boldsymbol{l} = (1, 3, 2, 3)$, which are also optimal. As in bottom-merge Huffman coding, the code with the minimum reverse lexicographical order among optimal codes is the one produced. This is also the case if we use the position of the "last" node in a package (in terms of the value of $nl + i$) in order to choose those with lower values, as in [35]. However, the above approach, which is easily shown to be equivalent via induction, eliminates the need for keeping track of the maximum value of $nl + i$ for each package.

E. Further Refinements

In this case using a bottom-merge-like coding method has an additional benefit: We no longer need assume that all $p_i \ne 0$ to assure that the nodeset is a valid code. In finding optimal binary codes, of course, it is best to ignore an item with $p_i = 0$. However, consider nonbinary output alphabets, that is, $D > 2$. As in Huffman coding for such alphabets, we must add "dummy" values of $p_i = 0$ to assure that the optimal code has the Kraft inequality satisfied with equality, an
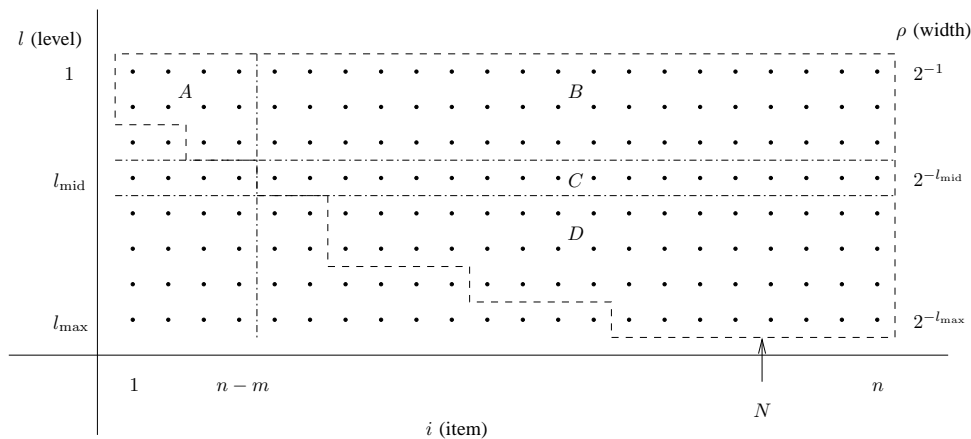
Fig. 2. The set of nodes I, an optimal nodeset N, and disjoint subsets A, B, C, D

assumption underlying both the Huffman algorithm and ours. The number of dummy values needed is $\mod(D - n, D - 1)$ where $\mod(x, y) \triangleq x - y \lfloor \frac{x}{y} \rfloor$ and where the dummy values each consist of $l_{\max}$ nodes, each node with the proper width and with weight 0. With this preprocessing step, finding an optimal code should proceed similarly to the binary case, with adjustments made for both the Package-Merge algorithm and the overall coding algorithm due to the formulation of the Kraft inequality and maximum length. A complete algorithm is available, with proof of correctness, in [28].

Note that we have assumed for all variations of this algorithm that we knew a maximum bound for length, although in the overall complexity analysis for binary coding we assumed this was $n - 1$ (except for flat classes). We now explore a method for finding better upper bounds and thus a more efficient algorithm. First we present a definition due to Larmore:

Definition 9: Consider penalty functions f and g. We say that g is *flatter* than f if, for probabilities p and p' and positive integers l and l' where $l' > l$, $M_g(l, p)M_f(l', p') \leq M_f(l, p)M_g(l', p')$ (where, again, $M_f(l, p) \triangleq f(l, p) - f(l - 1, p)$) [6].

A consequence of the Convex Hull Theorem of [6] is that, given g flatter than f, for any $\boldsymbol{p}$, there exist f-optimal $\boldsymbol{l}^{(f)}$ and g-optimal $\boldsymbol{l}^{(g)}$ such that $\boldsymbol{l}^{(f)}$ is greater lexicographically than $\boldsymbol{l}^{(g)}$ (again, with lengths sorted largest to smallest). This explains why the word "flatter" is used.

Thus, for penalties flatter than the linear penalty, we can obtain a useful upper bound, reducing complexity. All convex quasiarithmetic penalties are flatter than the linear penalty. (There are some generalized quasiarithmetic convex coding penalties that are not flatter than the linear penalty — e.g., $f(l_i, p_i) = l_i p_i^2$ — and some flatter penalties that are not Campbell/quasiarithmetic — e.g., $f(l_i, p_i) = 2^{l_i}(p_i + 0.1 \sin \pi p_i)$ — so no other similarly straightforward relation exists.) For most penalties we have considered, then, we can use the upper bounds in [30] or the results of a pre-algorithmic Huffman coding of the symbols to find an upper bound on codeword length.

A problem in which pre-algorithmic Huffman coding would be useful is delay coding, in which the quadratic penalty (2)

is solved for $O(n^2)$ values of α and β [6]. In this application, only one traditional Huffman coding would be necessary to find an upper bound for all quadratic cases.

With other problems, we might wish to instead use a mathematically derived upper bound. Using the maximum unary codeword length of $n - 1$ and techniques involving the Golden Mean, $\Phi \triangleq \frac{\sqrt{5}+1}{2}$, Buro in [30] gives the upper limit of length for a (standard) binary Huffman codeword as

$$\min \left\{ \left\lfloor \log_\Phi \left(\frac{\Phi + 1}{p_n \Phi + p_{n-1}} \right) \right\rfloor, n - 1 \right\}$$

which would thus be an upper limit on codeword length for the minimal optimal code obtained using any flatter penalty function, such as a convex quasiarithmetic function. This may be used to reduce complexity, especially in a case in which we encounter a flat class of problem inputs.

In addition to this, one can improve this algorithm by adapting the binary length-limited Huffman coding techniques of Moffat (with others) in [17]–[20], [22]. We do not explore these, however, as these cannot improve asymptotic results with the exception of a few special cases. Other approaches to length-limited Huffman coding with improved algorithmic complexity [21], [23] are not suited for extension to nonlinear penalties.

V. CONCLUSION

With a similar approach to that taken by Shannon for Shannon entropy and Campbell for Rényi entropy, we have shown redundancy bounds and related properties for optimal codes using Campbell's quasiarithmetic penalties and generalized entropies. For convex quasiarithmetic costs, our algorithms, based upon Larmore and Hirschberg's methods for the linear penalty, can efficiently find an optimal code. Such algorithms can be readily extended to the generalized quasiarithmetic convex class of penalties, as well as to the delay penalty, the latter of which results in more quickly finding an optimal code for delay channels.

One might ask whether the aforementioned properties can be extended; for example, can improved redundancy bounds similar to [12], [37]–[40] be found? It is an intriguing question,

albeit one that seems rather difficult to answer given that such general penalties lack a Huffman coding tree structure. In addition, although we know that optimal codes for infinite alphabets exist given the aforementioned conditions, we do not know how to find them. This, as with many infinite alphabet coding problems, remains open.

It would also be interesting if the algorithms could be extended to other penalties, especially since complex models of queueing can lead to other penalties aside from the delay penalty mentioned here. Also, note that the monotonicity property of the examples we consider implies that the resulting optimal code can be alphabetic, that is, lexicographically ordered by item number. If we desire items to be in a lexicographical order different from that of probability, however, the alphabetic and nonalphabetic cases can have different solutions. This was discussed for the length-limited penalty (5) in [35]; it might be of interest to generalize it to other penalties using similar techniques and to prove properties of alphabetic codes for such penalties.

APPENDIX
THE PACKAGE-MERGE ALGORITHM

Here we illustrate and prove the correctness of a recursive version of Package-Merge algorithm for solving the Coin Collector's problem. This algorithm was first presented in [16], which also has a linear-time iterative implementation.

Restating the Coin Collector's problem:

$$\text{Minimize}_{\{B\subseteq\{1,\ldots,m\}\}} \quad \sum_{i\in B}\mu_i$$
$$\text{subject to} \quad \sum_{i\in B}\rho_i = t$$
$$\text{where} \quad m\in\mathbb{Z}_+ \quad (20)$$
$$\mu_i\in\mathbb{R}$$
$$\rho_i\in 2^{\mathbb{Z}}$$
$$t\in\mathbb{R}_+.$$

In our notation, we use $i\in\{1,\ldots,m\}$ to denote both the index of a coin and the coin itself, and $\mathcal{I}$ to represent the m items along with their weights $\{\mu_i\}$ and widths $\{\rho_i\}$. The optimal solution, a function of total width t and items $\mathcal{I}$, is denoted $\mathrm{CC}(\mathcal{I},t)$.

Note that we assume the solution exists but might not be unique. In the case of distinct solutions, tie resolution for minimizing arguments may for now be arbitrary or rule-based; we clarify this in Subsection IV-D. A modified version of the algorithm considers the case where a solution might not exist, but this is not needed here. Because a solution exists, assuming $t > 0$, $t = \omega t_{\mathrm{pow}}$ for some unique odd $\omega\in\mathbb{Z}$ and $t_{\mathrm{pow}}\in 2^{\mathbb{Z}}$. (Note that t_{pow} need not be an integer. If $t = 0$, ω and t_{pow} are not defined.)

Algorithm variables
At any point in the algorithm, given nontrivial $\mathcal{I}$ and t, we use the following definitions:

Remainder
$$t_{\mathrm{pow}} \triangleq \text{the unique } x\in 2^{\mathbb{Z}} \text{ such that } \frac{t}{x} \text{ is an odd integer}$$

Minimum width
$$\rho^* \triangleq \min_{i\in\mathcal{I}}\rho_i \text{ (note } \rho^*\in 2^{\mathbb{Z}})$$

Small width set
$$\mathcal{I}^* \triangleq \{i \mid \rho_i = \rho^*\} \text{ (by definition, } |\mathcal{I}^*|\geq 1)$$

"First" item
$$i^* \triangleq \arg\min_{i\in\mathcal{I}^*}\mu_i$$

"Second" item
$$i^{**} \triangleq \arg\min_{i\in\mathcal{I}^*\setminus\{i^*\}}\mu_i$$
$$\text{(or null } \Lambda \text{ if } |\mathcal{I}^*|=1) .$$

Then the following is a recursive description of the algorithm:
Recursive Package-Merge Procedure [16]

Basis. $t = 0$: $\mathrm{CC}(\mathcal{I},t)$ is the empty set.

Case 1. $\rho^* = t_{\mathrm{pow}}$ *and* $\mathcal{I}\neq\emptyset$: $\mathrm{CC}(\mathcal{I},t) = \mathrm{CC}(\mathcal{I}\setminus\{i^*\},t-\rho^*)\cup\{i^*\}$.

Case 2a. $\rho^* < t_{\mathrm{pow}}$, $\mathcal{I}\neq\emptyset$, *and* $|\mathcal{I}^*| = 1$: $\mathrm{CC}(\mathcal{I},t) = \mathrm{CC}(\mathcal{I}\setminus\{i^*\},t)$.

Case 2b. $\rho^* < t_{\mathrm{pow}}$, $\mathcal{I}\neq\emptyset$, *and* $|\mathcal{I}^*| > 1$: Create i', a new item with weight $\mu_{i'} = \mu_{i^*}+\mu_{i^{**}}$ and width $\rho_{i'} = \rho_{i^*}+\rho_{i^{**}} = 2\rho^*$. This new item is thus a combined item, or *package*, formed by combining items i^* and i^{**}. Let $S' = \mathrm{CC}(\mathcal{I}\setminus\{i^*,i^{**}\}\cup\{i'\},t)$ (the optimization of the packaged version). If $i'\in S'$, then $\mathrm{CC}(\mathcal{I},t) = S'\setminus\{i'\}\cup\{i^*,i^{**}\}$; otherwise, $\mathrm{CC}(\mathcal{I},t) = S'$.

This algorithm always returns an optimal solution, as follows:

Theorem 6: If an optimal solution to the Coin Collector's problem exists, the above recursive (Package-Merge) algorithm will terminate with an optimal solution.

Proof: We show that the Package-Merge algorithm produces an optimal solution via induction on the depth of the recursion. The basis is trivially correct, and each inductive case reduces the number of items by one. The inductive hypothesis on $t \geq 0$ and $\mathcal{I}\neq\emptyset$ is that the algorithm is correct for any problem instance that requires fewer recursive calls than instance $(\mathcal{I},t)$.

If $\mathcal{I} = \emptyset$ and $t\neq 0$, or if $\rho^* > t_{\mathrm{pow}} > 0$, then there is no solution to the problem, contrary to our assumption. Thus all feasible cases are covered by those given in the procedure. Case 1 indicates that the solution must contain an odd number of elements of width ρ^*. These must include the minimum weight item in $\mathcal{I}^*$, since otherwise we could substitute one of the items with this "first" item and achieve improvement. Case 2 indicates that the solution must contain an even number of elements of width ρ^*. If this number is 0, neither i^* nor i^{**} is in the solution. If it is not, then they both are. If $i^{**} = \Lambda$, the number is 0, and we have Case 2a. If not, we may "package" the items, considering the replaced package as one item, as in Case 2b. Thus the inductive hypothesis holds and the algorithm is correct. ∎
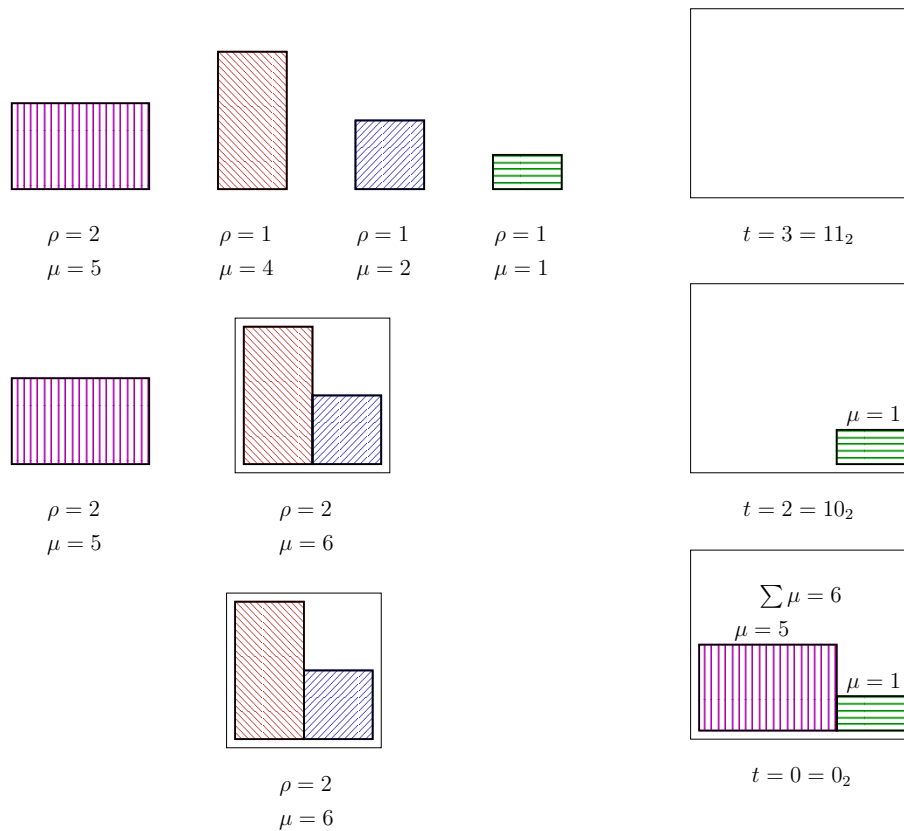
Fig. 3.   A simple example of the Package-Merge algorithm

Fig. 3 presents a simple example of this algorithm at work, finding minimum total weight items of total width $t = 3$ (or, in binary, 11_2). In the figure, item width represents numeric width and item area represents numeric weight. Initially, as shown in the top row, the minimum weight item with width $\rho_{i^*} = t_{\text{pow}} = 1$ is put into the solution set. Then, the remaining minimum width items are packaged into a merged item of width 2 (10_2). Finally, the minimum weight item/package with width $\rho_{i^*} = t_{\text{pow}} = 2$ is added to complete the solution set, which is now of weight 6. The remaining packaged item is left out in this case; when the algorithm is used for coding, several items are usually left out of the optimal set.

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]   P. A. Humblet, "Source coding for communication concentrators," Ph.D. dissertation, Massachusetts Institute of Technology, 1978.

[4]   L. L. Campbell, "Definition of entropy by means of a coding problem," *Z. Wahrscheinlichkeitstheorie und verwandte Gebiete*, vol. 6, pp. 113–118, 1966.

[5]   B. McMillan, "Two inequalities implied by unique decipherability," *IRE Trans. Inf. Theory*, vol. IT-2, no. 4, pp. 115–116, Dec. 1956.

[6]   L. L. Larmore, "Minimum delay codes," *SIAM J. Comput.*, vol. 18, no. 1, pp. 82–94, Feb. 1989.

[7]   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.

[8]   J. N. Kapur, "Noiseless coding theorems for different measures of entropy," *J. Comb., Inf. Syst. Sci.*, vol. 13, no. 3–4, pp. 114–126, 1989.

[9]   C. Flores, "Encoding of bursty sources under a delay criterion," Ph.D. dissertation, University of California, Berkeley, 1983.

[10]   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.

[11]   D. S. Parker, Jr., "Conditions for optimality of the Huffman algorithm," *SIAM J. Comput.*, vol. 9, no. 3, pp. 470–489, Aug. 1980.

[12]   M. Baer, "Rényi to Rényi — source coding under siege," in *Proceedings of the 2006 IEEE International Symposium on Information Theory*, 2006, to appear.

[13]   A. Rényi, *A Diary on Information Theory*.   New York, NY: John Wiley & Sons Inc., 1987, original publication: *Napló az információelméletről*, Gondolat, Budapest, Hungary, 1976.

[14]   M. R. Garey, "Optimal binary search trees with restricted maximal depth," *SIAM J. Comput.*, vol. 3, no. 2, pp. 101–110, June 1974.

[15]   A. Itai, "Optimal alphabetic trees," *SIAM J. Comput.*, vol. 5, no. 1, pp. 9–18, Mar. 1976.

[16]   L. L. Larmore and D. S. Hirschberg, "A fast algorithm for optimal length-limited Huffman codes," *J. ACM*, vol. 37, no. 2, pp. 464–473, Apr. 1990.

[17]   J. Katajainen, A. Moffat, and A. Turpin, "A fast and space-economical algorithm for length-limited coding," in *Proceedings of the International Symposium on Algorithms and Computation*, Dec. 1995, p. 1221.

[18]   M. Liddell and A. Moffat, "Incremental calculation of optimal length-restricted codes," in *Proceedings, IEEE Data Compression Conference*, Apr. 2002, pp. 182–191.

[19]   A. Moffat, A. Turpin, and J. Katajainen, "Space-efficient construction of optimal prefix codes," in *Proceedings, IEEE Data Compression Conference*, Mar. 28–30, 1995, pp. 192–202.

[20]   A. Turpin and A. Moffat, "Practical length-limited coding for large alphabets," *The Computer Journal*, vol. 38, no. 5, pp. 339–347, 1995.

[21]   A. Aggerwal, B. Schieber, and T. Tokuyama, "Finding a minimum-weight k-link path on graphs with the concave Monge property and applications," *Discrete and Computational Geometry*, vol. 12, pp. 263–280, 1994.

[22]   A. Turpin and A. Moffat, "Efficient implementation of the package-merge paradigm for generating length-limited codes," in *Proceedings of*

Computing: The Australasian Theory Symposium, Jan. 29–30, 1996, pp. 187–195.

[23] B. Schieber, "Computing a minimum-weight k-link path in graphs with the concave Monge property," *Journal of Algorithms*, vol. 29, no. 2, pp. 204–222, Nov. 1998.

[24] A. Rényi, "Some fundamental questions of information theory," *Magyar Tudományos Akadémia III. Osztalyanak Közlemenei*, vol. 10, no. 1, pp. 251–282, 1960.

[25] J. Aczél, "On Shannon's inequality, optimal coding, and characterizations of Shannon's and Rényi's entropies," in *Symposia Mathematica*, vol. 15.   New York, NY: Academic Press, 1973, pp. 153–179.

[26] 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.

[27] E. S. Schwartz, "An optimum encoding with minimum longest code and total number of digits," *Inf. Contr.*, vol. 7, no. 1, pp. 37–44, Mar. 1964.

[28] M. B. Baer, "Twenty (or so) questions: D-ary bounded-length Huffman coding," preprint available from http://arxiv.org/abs/cs.IT/0602085.

[29] G. O. H. Katona and T. Nemetz, "Huffman codes and self-information," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 3, pp. 337–340, May 1976.

[30] M. Buro, "On the maximum length of Huffman codes," *Inf. Processing Letters*, vol. 45, no. 5, pp. 219–223, Apr. 1993.

[31] Y. S. Abu-Mostafa and R. J. McEliece, "Maximal codeword lengths in Huffman codes," *Computers & Mathematics with Applications*, vol. 39, no. 11, pp. 129–134, Oct. 2000.

[32] M. R. Garey and D. S. Johnson, *Computers and Intractability*.   San Francisco, CA: W. H. Freeman and Company, 1979.

[33] U. Manber, *Introduction to Algorithms*.   Reading, MA: Addison-Wesley, 1989.

[34] T. Cover and J. Thomas, *Elements of Information Theory*.   New York, NY: Wiley-Interscience, 1991.

[35] L. L. Larmore and T. M. Przytycka, "A fast algorithm for optimal height-limited alphabetic binary-trees," *SIAM J. Comput.*, vol. 23, no. 6, pp. 1283–1312, Dec. 1994.

[36] J. van Leeuwen, "On the construction of Huffman trees," in *Proc. 3rd Int. Colloquium on Automata, Languages, and Programming*, July 1976, pp. 382–410.

[37] D. Baron and A. C. Singer, "On the cost of worst case coding length constraints," *IEEE Trans. Inf. Theory*, vol. IT-47, no. 7, pp. 3088–3090, Nov. 2001.

[38] 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.

[39] 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.

[40] I. J. Taneja, "A short note on the redundancy of degree α," *Inf. Sci.*, vol. 39, no. 2, pp. 211–216, Sept. 1986.