A Method for the Construction of Optimal Task Encoders

Amos Lapidoth and Christoph Pfister
ETH Zurich
Email: {lapidoth,pfistchr}@ethz.ch

***Abstract*—An algorithm of polynomial complexity is proposed that produces an optimal task encoder for tasks that are generated according to some given law and that need to be described using a given number of labels. It thus minimizes the expectation (or ρ-th moment) of the number of tasks that share the label of a randomly-generated task.**

***Index Terms*—Task Encoding, Rényi Entropy, Dynamic Programming, Guessing.**

I. Introduction

Task encoders were introduced in [1], where asymptotically-tight upper and lower bounds on their performance in terms of Rényi entropy were derived. But [1] did not solve for the optimal firm (nonasymptotic) task encoder. The present paper fills this gap by describing an algorithm that produces optimal task encoders. It can be viewed as the task-encoding analog of Huffman's procedure from source coding [2].

The task-encoding problem can be described as follows. A task X is drawn from a finite set[1] $\mathcal{X}$ according to a probability mass function P and is mapped by a task encoder f to a label $f(X)$. You must perform the task X, but you only see its label $f(X)$. To be sure to perform X, you thus perform all the tasks that have the same label as X.[2] The goal is to choose a task encoder f that minimizes the expectation (or, more generally, the ρ-th moment) of the number of performed tasks. This paper presents an algorithm to produce such an optimal task encoder.

As shown in [4], the asymptotics of the task-encoding problem are related to those of the Massey-Arikan guessing problem [5]. But the algorithmic aspects are quite different. In the former, an algorithm like the one proposed here is required, whereas in the latter the optimal rule is obvious: one should simply guess in decreasing order of probabilities.

Let us denote the number of different tasks by K, so $\mathsf{K} = |\mathcal{X}|$, where throughout this paper $|\cdot|$ denotes the cardinality of a set. Let M denote the number of available labels. A *task encoder* is a function from $\mathcal{X}$ to $\{1, \ldots, \mathsf{M}\}$ that maps tasks to labels. The number of tasks that share the label of a task $x \in \mathcal{X}$ under a task encoder f is denoted by

$$L_f(x) = \big|f^{-1}(f(x))\big| = \big|\{x' \in \mathcal{X} \colon f(x') = f(x)\}\big|. \tag{1}$$

[1]The task-encoding problem for continuous alphabets is described in [3].

[2]Depending on our interpretation of probabilities, this may or may not be true in the presence of tasks that are assigned with probability zero. This philosophical issue affects only the interpretation of (1) and none of the mathematical statements.

The *cost* is the expected value of $L_f(X)$ or, more generally, its ρ-th moment (for any $\rho > 0$):

$$\mathrm{E}[L_f(X)^\rho] = \sum_{x \in \mathcal{X}} P(x) \big|f^{-1}(f(x))\big|^\rho. \tag{2}$$

A task encoder f is called *optimal* if its cost is minimal among all possible task encoders, i.e., if

$$\mathrm{E}[L_f(X)^\rho] = \min_{g \colon \mathcal{X} \to \{1,\ldots,\mathsf{M}\}} \mathrm{E}[L_g(X)^\rho]. \tag{3}$$

Optimal task encoders exist because the set of all possible task encoders is finite. The algorithm presented in Section III finds an optimal task encoder in $O(\mathsf{K}^2\mathsf{M})$ arithmetic operations.

II. Optimal Task Encoders

From now on, let us denote the different tasks by $x_1, \ldots, x_\mathsf{K}$, and let us assume that they are ordered according to their probabilities, so

$$P(x_1) \leq P(x_2) \leq P(x_3) \leq \ldots \leq P(x_\mathsf{K}). \tag{4}$$

Given such an order, we say that a task encoder f is *monotonic* if tasks with higher probabilities have larger labels, or, more precisely, if

$$f(x_1) \leq f(x_2) \leq f(x_3) \leq \ldots \leq f(x_\mathsf{K}). \tag{5}$$

The main result of this section is Lemma II.3, which states that there exists an optimal task encoder that is also monotonic. To that end, we show in Lemma II.1 and Lemma II.2 how every task encoder can be turned into a monotonic task encoder without increasing its cost.

Lemma II.1. *For every task encoder g, there exists a task encoder f that has the same cost as g and that satisfies*

$$|f^{-1}(1)| \geq |f^{-1}(2)| \geq |f^{-1}(3)| \geq \ldots \geq |f^{-1}(\mathsf{M})|. \tag{6}$$

Proof: We construct f by relabeling the tasks. Let ϕ be a permutation on $\{1, \ldots, \mathsf{M}\}$ that maps one of the labels with the largest preimage to 1 and so on. Then the task encoder $f(x) = \phi(g(x))$ fulfills (6) and has the same cost as g because permuting the labels does not change the cost. ■

Lemma II.2. *If the tasks are ordered according to (4), then any task encoder satisfying (6) can be turned into a monotonic task encoder satisfying (6) without increasing its cost.*

Proof: Let f be a task encoder that satisfies (6). If f is not monotonic, i.e., if we have $f(x_i) > f(x_{i+1})$ for some $i \in \{1, \ldots, \mathsf{K}-1\}$, define a task encoder g that has the labels of the tasks x_i and x_{i+1} swapped:

$$g(x) = \begin{cases} f(x_{i+1}) & \text{if } x = x_i, \\ f(x_i) & \text{if } x = x_{i+1}, \\ f(x) & \text{if } x \in \mathcal{X} \setminus \{x_i, x_{i+1}\}. \end{cases} \tag{7}$$

Because $|g^{-1}(m)| = |f^{-1}(m)|$ holds for all $m \in \{1, \ldots, \mathsf{M}\}$, (6) is also satisfied for g. Replace f by g and repeat these steps until f is monotonic.

Next, we argue that this procedure terminates. Note that the quantity $\sum_{k=1}^{\mathsf{K}} k \cdot f(x_k)$ increases by at least one after every repetition, yet it is bounded from above. Therefore, the procedure must terminate after finitely many steps.

Finally, we show that the cost never increases:

$$\begin{aligned} &\mathrm{E}[L_g(X)^\rho] - \mathrm{E}[L_f(X)^\rho] \\ &\quad = P(x_i) \cdot (L_g(x_i)^\rho - L_f(x_i)^\rho) \\ &\qquad + P(x_{i+1}) \cdot (L_g(x_{i+1})^\rho - L_f(x_{i+1})^\rho) && (8) \\ &\quad = \underbrace{(P(x_i) - P(x_{i+1}))}_{\leq 0} \cdot \underbrace{(L_f(x_{i+1})^\rho - L_f(x_i)^\rho)}_{\geq 0} && (9) \\ &\quad \leq 0. && (10) \end{aligned}$$

Here (8) follows because $L_g(x)$ is equal to $L_f(x)$ for every $x \in \mathcal{X} \setminus \{x_i, x_{i+1}\}$, and (9) is true because $L_g(x_i) = L_f(x_{i+1})$ and $L_g(x_{i+1}) = L_f(x_i)$ hold. The first factor on the RHS of (9) is nonpositive because of (4). Combining $f(x_i) > f(x_{i+1})$ and (6) leads to $L_f(x_i) \leq L_f(x_{i+1})$, and as y^ρ is increasing in y (for $y \geq 0$ and $\rho > 0$), the second factor on the RHS of (9) is nonnegative. ■

Lemma II.3. *If the tasks are ordered according to (4), then there exists an optimal task encoder that is monotonic and that satisfies (6).*

Proof: Let g be an optimal task encoder. By Lemma II.1, there exists an optimal task encoder f satisfying (6). Applying Lemma II.2 to f leads to an optimal task encoder that is monotonic and that satisfies (6). ■

Although the following observations are not needed in the next sections, they still deserve to be stated here.

Lemma II.4. *Let the tasks be ordered according to (4). If a monotonic task encoder f satisfies (6), then tasks with higher probabilities end up in smaller sets, or, more precisely,*

$$L_f(x_1) \geq L_f(x_2) \geq L_f(x_3) \geq \ldots \geq L_f(x_{\mathsf{K}}). \tag{11}$$

In particular, there exists an optimal task encoder that satisfies (11).

Proof: By Lemma II.3, there exists an optimal task encoder that is monotonic and that satisfies (6). This task encoder satisfies (11) because (5) and (6) imply (11). ■

Lemma II.5. *Let the tasks be ordered according to (4). Let $\alpha_1, \ldots, \alpha_{\mathsf{M}} \in \{0, \ldots, \mathsf{K}\}$ satisfy $\sum_{m=1}^{\mathsf{M}} \alpha_m = \mathsf{K}$ and*

$$\alpha_1 \geq \alpha_2 \geq \alpha_3 \geq \ldots \geq \alpha_{\mathsf{M}}. \tag{12}$$

If the set sizes of the task encoders are fixed, i.e., if only task encoders f are considered that satisfy

$$|f^{-1}(m)| = \alpha_m \ \forall m \in \{1, \ldots, \mathsf{M}\}, \tag{13}$$

then the following task encoder g minimizes the cost among these task encoders:

$$g(x_i) = \begin{cases} 1 & \text{if } i \in \{1, \ldots, \alpha_1\}, \\ 2 & \text{if } i \in \{\alpha_1 + 1, \ldots, \alpha_1 + \alpha_2\}, \\ \vdots & \\ \mathsf{M} & \text{if } i \in \left\{ \sum\limits_{m=1}^{\mathsf{M}-1} \alpha_m + 1, \ldots, \sum\limits_{m=1}^{\mathsf{M}} \alpha_m \right\}. \end{cases} \tag{14}$$

Proof: Let $\mathcal{S}$ denote the set of all task encoders satisfying (13). $\mathcal{S}$ is not empty because it contains g. Let $f \in \mathcal{S}$ be a task encoder with minimal cost. Because of (12), f satisfies (6), and Lemma II.2 can be applied to obtain a monotonic task encoder f'. f' satisfies (13) because the procedure employed in the proof of Lemma II.2 does not change the set sizes. As only one task encoder exists that satisfies (5) and (13) at the same time, f' must be equal to g. Because $g = f'$ does not have a higher cost than f, its cost is minimal among $\mathcal{S}$. ■

III. Algorithm

We continue to assume that the tasks $x_1, \ldots, x_{\mathsf{K}}$ are ordered by their probability, i.e., that (4) holds. The algorithm that we present in this section will produce a monotonic task encoder with minimal cost. By Lemma II.3, we know that there is no loss in optimality in restricting ourselves to monotonic task encoders, therefore the task encoder produced by the algorithm will be optimal.

This section is organized as follows. First, we discuss the basic idea behind the algorithm. Next, we provide a formal treatment in Lemma III.1. Finally, we show a simple implementation of the algorithm in Figure 1.

The algorithm is based on *dynamic programming*. (For a general introduction to this method, see for example [6].) Assume $\mathsf{M} \geq 2$ and let f be an optimal monotonic task encoder. Let $l = |f^{-1}(\mathsf{M})|$ denote the number of tasks that are mapped to M, and let $\mathcal{X}' \subseteq \mathcal{X}$ be the set of the tasks that are not mapped to M. As f is monotonic, $\mathcal{X}'$ is equal to $\{x_1, \ldots, x_{\mathsf{K}-l}\}$. The restriction of f to $\mathcal{X}'$ is a monotonic task encoder $f' \colon \mathcal{X}' \to \{1, \ldots, \mathsf{M}-1\}$ with $f'(x) = f(x)$ for all $x \in \mathcal{X}'$. The cost of f can then be expressed as

$$\mathrm{E}[L_f(X)^\rho] = \sum_{x \in \mathcal{X}'} P(x) L_{f'}(x)^\rho + \sum_{x \in \mathcal{X} \setminus \mathcal{X}'} P(x) l^\rho. \tag{15}$$

As we next argue, the optimality of f implies that f' must be optimal with respect to the partial cost $\sum_{x \in \mathcal{X}'} P(x) L_{f'}(x)^\rho$. Indeed if f' were not optimal, i.e., if there existed a task encoder $g' \colon \mathcal{X}' \to \{1, \ldots, \mathsf{M}-1\}$ with a lower partial cost, then (15) demonstrates that the extension of g' to $\mathcal{X}$, namely

$$x \mapsto \begin{cases} g'(x) & \text{if } x \in \mathcal{X}', \\ \mathsf{M} & \text{if } x \in \mathcal{X} \setminus \mathcal{X}', \end{cases} \tag{16}$$

would have a lower cost than f. But this would contradict the assumption that f is optimal.

As we do not know the optimal value of l in advance, we could solve the problem recursively for all $l \in \{0, \ldots, \mathsf{K}\}$ and select the value with the lowest cost. But this would lead to an algorithm of exponential complexity. Instead, we build an $\mathsf{M} \times \mathsf{K}$ table whose cell in the m-th row and the k-th column contains an optimal monotonic task encoder for the first k tasks and the first m labels. We will show that we can construct this table easily row by row and that we need $O(\mathsf{K})$ arithmetic operations per cell. Taking into account that we have an $\mathsf{M} \times \mathsf{K}$ table, we can thus find an optimal task encoder in $O(\mathsf{K}^2\mathsf{M})$ arithmetic operations.

Lemma III.1. *Let the tasks be ordered according to (4). Define an optimal partial cost for every $m \in \{1, \ldots, \mathsf{M}\}$ and for every $k \in \{1, \ldots, \mathsf{K}\}$ by*

$$J(m,k) = \min_{g\colon \{x_1,\ldots,x_k\}\to\{1,\ldots,m\}} \sum_{i=1}^{k} P(x_i) L_g(x_i)^\rho. \tag{17}$$

We seek $J(\mathsf{M},\mathsf{K})$. This can be computed inductively on m as follows. If there is only one label, i.e., for $m = 1$, we have

$$J(1,k) = \sum_{i=1}^{k} P(x_i) k^\rho. \tag{18}$$

If there is more than one label, we can build on the case with one fewer label, i.e., for $m > 1$, we can express $J(m,k)$ in terms of $J(m-1,\cdot)$ as

$$\begin{aligned} J(m,k) &= \min_{g\colon \{x_1,\ldots,x_k\}\to\{1,\ldots,m\}} \sum_{i=1}^{k} P(x_i) L_g(x_i)^\rho \qquad (19)\\ &= \min_{k'\in\{1,\ldots,k\}} \Bigg[\min_{g'\colon \{x_1,\ldots,x_{k'}\}\to\{1,\ldots,m-1\}} \sum_{i=1}^{k'} P(x_i) L_{g'}(x_i)^\rho \\ &\qquad + \sum_{i=k'+1}^{k} P(x_i)(k-k')^\rho \Bigg] \qquad (20)\\ &= \min_{k'\in\{1,\ldots,k\}} \Bigg[J(m-1,k') + \sum_{i=k'+1}^{k} P(x_i)(k-k')^\rho \Bigg]. \qquad (21) \end{aligned}$$

Proof: The case $m = 1$ is easy: there is only one label, so all tasks have to be mapped to the same label, and (18) follows from (17).

Next, we prove the case $m > 1$. To improve the readability, (17) has been restated as (19). The RHS of (20) is equal to the RHS of (21) because of (17), so (21) is valid. We are left to prove (20). If $P(x_1) = \ldots = P(x_k) = 0$ holds, then (20) is trivially satisfied because both sides are zero. Otherwise, we have $\sum_{i=1}^{k} P(x_i) > 0$ because P is a nonnegative function. In that case, we show (20) by proving that neither side of the equation is larger than the other.

First, we show that the RHS of (19) is not larger than the RHS of (20). To that end, let $k' \in \{1, \ldots, k\}$ and $g'\colon \{x_1, \ldots, x_{k'}\} \to \{1, \ldots, m-1\}$ attain the minima in the RHS of (20). The RHS of (19) cannot be larger than the RHS of (20) because equality is achieved for the following choice of $g\colon \{x_1, \ldots, x_k\} \to \{1, \ldots, m\}$ in the RHS of (19):

$$g(x) = \begin{cases} g'(x) & \text{if } x \in \{x_1, \ldots, x_{k'}\}, \\ m & \text{otherwise.} \end{cases} \tag{22}$$

Second, we show that the RHS of (20) is not larger than the RHS of (19). Note that for any $\alpha > 0$, the RHS of (19) is equal to

$$\alpha^{-1} \min_{g\colon \{x_1,\ldots,x_k\}\to\{1,\ldots,m\}} \sum_{i=1}^{k} (\alpha P(x_i)) L_g(x_i)^\rho. \tag{23}$$

We choose $\alpha = 1/\sum_{i=1}^{k} P(x_i) > 0$, so that $x \mapsto \alpha P(x)$ is a probability mass function for $\{x_1, \ldots, x_k\}$. By Lemma II.3, there exists a monotonic task encoder g that attains the minimum in (23) and that satisfies (6). This g also attains the minimum in the RHS of (19). Set $k' = k - |g^{-1}(m)|$. As we have $m > 1$ and as g satisfies (6), $|g^{-1}(m)| < k$ holds and k' is in $\{1, \ldots, k\}$. Set $g'(x) = g(x)$ for all $x \in \{x_1, \ldots, x_{k'}\}$. Because g is monotonic and because of our choice of k', the codomain of g' is $\{1, \ldots, m-1\}$. The RHS of (20) cannot be larger than the RHS of (19) because equality is achieved for our choice of k' and g' in the RHS of (20). ■

Figure 1 illustrates a simple implementation of these ideas. We assume that the tasks $x_1, \ldots, x_\mathsf{K}$ are ordered by their probability, i.e., that (4) holds, and use following shorthand notation: $p_i = P(x_i)$ for all $i \in \{1, \ldots, \mathsf{K}\}$. The optimal partial cost function (17) is represented as the $\mathsf{M} \times \mathsf{K}$ table J. As we want to be able to construct an optimal task encoder at the end of the algorithm, we keep track of the necessary information in the $\mathsf{M} \times \mathsf{K}$ table *setSizes*. For every cell, it contains the number of tasks that an optimal task encoder in that cell maps to the last label. In other words, its value is equal to $k - k'$ for an optimal value of k' in the RHS of (21). In lines 3–6, the first row of the table is computed according to (18). In lines 7–21, the other rows of the table are computed according to (21). Finally, an optimal task encoder is constructed for $J(\mathsf{M},\mathsf{K})$ by assigning an optimal number of tasks to the label M, to the label $\mathsf{M}-1$, etc.

Note that it is not efficient to compute the summations by adding up the values. Instead, compute $\Psi(k) = \sum_{i=1}^{k} p_i$ for all $k \in \{1, \ldots, \mathsf{K}\}$ once at the beginning and replace the summations in line 4 and 12 by the following expressions:

$$\sum_{i=1}^{k} p_i k^\rho = \Psi(k) k^\rho, \tag{24}$$

$$\sum_{i=k'+1}^{k} p_i (k-k')^\rho = (\Psi(k) - \Psi(k')) \cdot (k-k')^\rho. \tag{25}$$

Lines 12–16 are executed $O(\mathsf{K}^2\mathsf{M})$ times and determine the asymptotic complexity. Using (25), these lines can be executed with a constant number of arithmetic operations. Therefore, the algorithm can find an optimal task encoder in $O(\mathsf{K}^2\mathsf{M})$ arithmetic operations.

```
**Input:** $\mathsf{M}, \mathsf{K}, \rho$ and $p_1, \ldots, p_\mathsf{K}$ (satisfying $p_1 \leq \ldots \leq p_\mathsf{K}$)
**Output:** an optimal task encoder
$J \leftarrow \mathsf{M} \times \mathsf{K}$ table of real numbers
$setSizes \leftarrow \mathsf{M} \times \mathsf{K}$ table of integers
for $k \leftarrow 1, 2, \ldots, \mathsf{K}$ do                ▷ compute first row
    $J[1][k] \leftarrow \sum_{i=1}^{k} p_i k^\rho$
    $setSizes[1][k] \leftarrow k$
end for
for $m \leftarrow 2, 3, \ldots, \mathsf{M}$ do                ▷ compute other rows
    for $k \leftarrow 1, 2, \ldots, \mathsf{K}$ do
        $minCost \leftarrow J[m-1][k]$
        $optSetSize \leftarrow 0$
        for $k' \leftarrow k-1, k-2, \ldots, 1$ do
            $c \leftarrow J[m-1][k'] + \sum_{i=k'+1}^{k} p_i (k-k')^\rho$
            if $c < minCost$ then
                $minCost \leftarrow c$
                $optSetSize \leftarrow k - k'$
            end if
        end for
        $J[m][k] \leftarrow minCost$
        $setSizes[m][k] \leftarrow optSetSize$
    end for
end for
$encoder \leftarrow$ task encoder                ▷ construct task encoder
$task \leftarrow \mathsf{K}$
for $m \leftarrow \mathsf{M}, \mathsf{M}-1, \ldots, 1$ do
    $count \leftarrow setSizes[m][task]$
    for $i \leftarrow 1, 2, \ldots, count$ do
        $encoder[task] = m$
        $task \leftarrow task - 1$
    end for
end for
return $encoder$
```

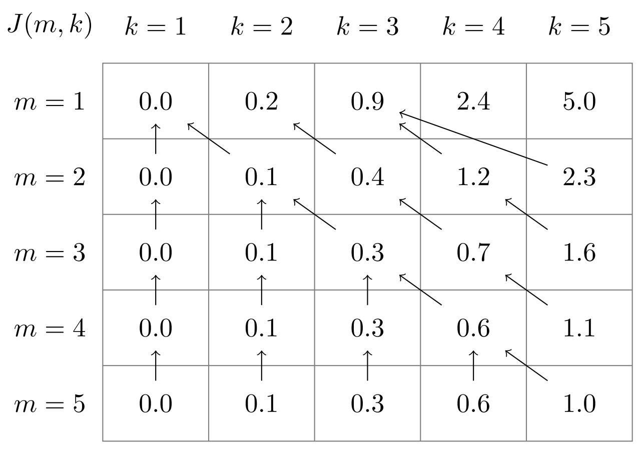
Fig. 1. Algorithm to produce an optimal task encoder.

IV. Example

In Figure 2, a numerical example of the algorithm is depicted. The set $\mathcal{X}$ consists of five tasks and the probability mass function P is given by

$$P(x) = \begin{cases} 0.0 & \text{if } x = x_1, \\ 0.1 & \text{if } x = x_2, \\ 0.2 & \text{if } x = x_3, \\ 0.3 & \text{if } x = x_4, \\ 0.4 & \text{if } x = x_5. \end{cases} \tag{26}$$

We set $\rho = 1$, and the resulting partial cost function $J(m,k)$ defined in Lemma III.1 is shown in Figure 2. The arrows are used to keep track of recurrence relation (21). The arrow from $J(3,5)$ to $J(2,4)$ for example means that an optimal task encoder g for five tasks and three labels is obtained by extending an optimal task encoder g' that achieves $J(2,4)$ in



$J(m,k)$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
$m=1$	0.0	0.2	0.9	2.4	5.0
$m=2$	0.0	0.1	0.4	1.2	2.3
$m=3$	0.0	0.1	0.3	0.7	1.6
$m=4$	0.0	0.1	0.3	0.6	1.1
$m=5$	0.0	0.1	0.3	0.6	1.0

Fig. 2. Numerical example with five tasks.

the following way:

$$g(x_i) = \begin{cases} g'(x_i) & \text{if } i \in \{1, \ldots, 4\}, \\ 3 & \text{if } i = 5. \end{cases} \tag{27}$$

Therefore, it is easy to read off the optimal costs and the task encoders achieving these costs for $\mathsf{M} \in \{1, \ldots, 5\}$:

1) For $\mathsf{M} = 1$, the optimal cost is 5.0, and all tasks are mapped to 1.
2) For $\mathsf{M} = 2$, the optimal cost is 2.3, and $x_1, \ldots, x_3$ are mapped to 1, while x_4 and x_5 are mapped to 2.
3) For $\mathsf{M} = 3$, the optimal cost is 1.6, and $x_1, \ldots, x_3$ are mapped to 1, x_4 is mapped to 2, and x_5 is mapped to 3.
4) For $\mathsf{M} = 4$, the optimal cost is 1.1, and x_1 and x_2 are mapped to 1, x_3 is mapped to 2, x_4 is mapped to 3, and x_5 is mapped to 4.
5) For $\mathsf{M} = 5$, the optimal cost is 1.0, and task x_i is mapped to i for $i \in \{1, \ldots, 5\}$.

V. Discussion

We conclude the discussion of the algorithm with a few remarks.

1) If $m \geq k$ holds, then $J(m,k)$ is equal to $\sum_{i=1}^{k} P(x_i)$. Every injective task encoder achieves that partial cost, and one cannot do better because $L_g(x_i)^\rho$ in the RHS of (17) cannot be smaller than 1.
2) The idea behind the algorithm also works if an $\mathsf{M} \times \mathsf{K}$ table is built whose cells contain an optimal monotonic task encoder for the last tasks and the last labels. In other words, an analog of Lemma III.1 can be derived for the following partial cost function:

$$J'(m,k) = \min_{\substack{g\colon \{x_{\mathsf{K}-k+1}, \ldots, x_\mathsf{K}\} \\ \to \{\mathsf{M}-m+1, \ldots, \mathsf{M}\}}} \sum_{i=\mathsf{K}-k+1}^{\mathsf{K}} P(x_i) L_g(x_i)^\rho. \tag{28}$$

3) For $\mathsf{M} > \mathsf{K}/2$, it follows from (11) that there is no loss in optimality if the last $(2\mathsf{M} - \mathsf{K})$ tasks have their own label.

4) If $h\colon \{1,2,\ldots\} \to \mathbb{R}$ is an increasing function, then the algorithm in Figure 1 can be easily adapted to produce a task encoder that is optimal with respect to the following generalized cost:

$$\mathrm{E}[h(L_f(X))] = \sum_{x\in\mathcal{X}} P(x)h(|f^{-1}(f(x))|). \qquad (29)$$

The cost (2) is then a special case of (29) for $h(l) = l^\rho$. In the proof of Lemma II.2 it is required that h is an increasing function, but apart from that, our derivation of the algorithm does not need any assumption about h.

References

[1] C. Bunte and A. Lapidoth, "Encoding Tasks and Rényi Entropy," *IEEE Trans. Inf. Theory*, vol. 60, no. 9, pp. 5065–5076, Sept. 2014.

[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Hoboken, NJ: Wiley, 2006.

[3] C. Bunte and A. Lapidoth, "Rényi entropy and quantization for densities," in *Proc. 2014 IEEE Information Theory Workshop (ITW)*, Hobart, Tasmania, Australia, 2014, pp. 257–261.

[4] A. Bracher, E. Hof, and A. Lapidoth, "Distributed storage for data security," in *Proc. 2014 IEEE Information Theory Workshop (ITW)*, Hobart, Tasmania, Australia, 2014, pp. 506–510.

[5] E. Arikan, "An inequality on guessing and its application to sequential decoding," *IEEE Trans. Inf. Theory*, vol. 42, no. 1, pp. 99–105, Jan. 1996.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA: The MIT Press, 2009.