The 42nd Symposium on Information Theory
and its Applications (SITA2019)
Kirishima, Kagoshima, Japan, Nov. 26–29, 2019

Asymptotic Optimality of Asymmetric Numeral Systems

Hidetoshi Yokoo* Danny Dubé †

Abstract—An entropy coder is defined to be *asymptotically optimal* if it attains asymptotically the source entropy in a certain limit of its parameter. In this paper, we first give a sufficient condition for a stream variant of ANS to attain the source entropy. Then, we show that the tANS variant with Duda's precise initialization algorithm for constructing a symbol distribution has asymptotic optimality in the sense that the average codeword length approaches the source entropy as the size of the state set grows.

Keywords—ANS, asymptotic optimality, data compression, entropy, memoryless source

1 Introduction

For a known (memoryless) information source, entropy coders may attain the entropy of the source in some cases, and may not in other cases. To attain the entropy in the latter cases, we have to introduce a technique of *alphabet extension* (known also as *source* or *symbol extension*). Some entropy coders, on the other hand, have their own parameters and can make the compression rate close to the entropy by tuning the parameters rather than by appealing to such general techniques. We define an entropy coder to be *asymptotically optimal* if the coder asymptotically attains the entropy in certain limits of parameter values[1]. In this paper, we show that a variation of *Asymmetric Numeral Systems* has such asymptotic optimality.

Asymmetric Numeral Systems (ANS), developed by Jarek Duda in a series of his papers [3]–[5], form a family of entropy coders, and have already been used in various applications [6], [7], [9]–[12]. While Shannon coding and arithmetic coding, which are the most typical examples of entropy coders, regard a fractional part of a number as a codeword, ANS generalizes an integer part. The term “asymmetric” means that ANS does not necessarily treat α distinct symbols equally in each digit of the radix-α representation of an integer.

ANS can be divided into several variations from multiple viewpoints. When the input alphabet is binary, they are called *Asymmetric Binary Systems* (ABS), and otherwise ANS. The most essential component of ANS may be the so-called *symbol distribution*, a sequence of inner states of the encoder and decoder. Basic ANS assumes an unbounded symbol distribution while *stream* ANS assumes a bounded one. The symbol distribution is given as a formula in a range variant (rANS), and as a table in a tabled variant (tANS). Thus, the variation is determined by giving a concrete definition of its symbol distribution. For example, Fujisaki [8] analyzed basic ABS with Sturmian sequences as symbol distributions. The symbol distribution also affects the compression performance of ANS, both theoretically and empirically. We have empirically compared various construction methods of the symbol distribution in terms of compression efficiency [1], and proposed a heuristic way for constructing almost optimal symbol distributions [2].

In spite of the importance of symbol distributions in ANS, few theoretical studies have been presented on their relation to the compression performance. In this paper, we first give a sufficient condition for a stream variant of ANS to attain the source entropy. Then, we shift our focus to the stream variant of tANS with Duda's procedure [5] for constructing a symbol distribution, and analyze the procedure. The symbol distribution in the variant is bounded with l states. We show that the compression rate attained by the variant approaches the source entropy as l goes to infinity.

2 Asymmetric Numeral Systems

2.1 Notation

Let $\mathcal{X}$ be the source alphabet. We assume that the size of $\mathcal{X}$ is α and its elements, called *symbols*, are integers from 0 to $\alpha - 1$. Let p_s denote the probability of symbol $s \in \mathcal{X}$. We consider lossless compression of a sequence of symbols drawn from a stationary, memoryless source with known probability distribution $\{p_s\}_{s=0}^{\alpha-1}$. We fix the code alphabet to be $\{0, 1\}$. When we use base-2 logarithms, the (binary) entropy of the source is defined as

$$H = -\sum_{s=0}^{\alpha-1} p_s \log p_s \quad \text{bit.}$$

Throughout this paper, we assume $p_s > 0$ for every symbol $s \in \mathcal{X}$ in the source. We represent the integer interval from i to j, both inclusive, by $[i, j]$, and regard it as a set of integers. The largest integer smaller than or equal to x is denoted by $\lfloor x \rfloor$, and the smallest integer larger than or equal to x by $\lceil x \rceil$. The remainder of the division of x by m is denoted by $\mathsf{mod}(x, m)$. The base of logarithms is 2 except when natural logarithms, denoted by $\ln(\cdot)$, will be used in the proof of Theorem 1.

2.2 Symbol Distribution and ANS

Before introducing ANS, we consider a “symmetric” numeral system, namely, an ordinary positional base-α representation for integers. Suppose that we have an array of integers $\overline{s} = \overline{s}[0, +\infty)$, in which we have $\overline{s}[i] = \mathsf{mod}(i, \alpha)$. Based on this array, we define the

* Division of Electronics and Informatics, Gunma University, Kiryu 376-8515, Japan. E-mail: yokoo@gunma-u.ac.jp
† Department of Computer Science and Software Engineering Université Laval, Canada. E-mail: Danny.Dube@ift.ulaval.ca

[1] This definition is different from one used in universal coding.

following two functions[2]:

$C(s,\, x)$: integer i such that $\overline{s}[i]$ is the $(x+1)$st occurrence of s,

$D(x)$: the number of occurrences of $\overline{s}[x]$ in $\overline{s}[0,\, x-1]$.

For example, assume that we have

$$\overline{s} = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|} \multicolumn{1}{c}{\scriptstyle 0} & \multicolumn{1}{c}{\scriptstyle 1} & \multicolumn{1}{c}{\scriptstyle 2} & \multicolumn{1}{c}{\scriptstyle 3} & \multicolumn{1}{c}{\scriptstyle 4} & \multicolumn{1}{c}{\scriptstyle 5} & \multicolumn{1}{c}{\scriptstyle 6} & \multicolumn{1}{c}{\scriptstyle 7} & \multicolumn{1}{c}{\scriptstyle 8} & \multicolumn{1}{c}{\scriptstyle 9} & \multicolumn{1}{c}{\scriptstyle 10} & \multicolumn{1}{c}{\scriptstyle 11} & \multicolumn{1}{c}{\scriptstyle 12} & \multicolumn{1}{c}{\scriptstyle 13} & \multicolumn{1}{c}{\scriptstyle 14} & \multicolumn{1}{c}{\scriptstyle 15} & \multicolumn{1}{c}{\scriptstyle 16} & \multicolumn{1}{c}{\scriptstyle 17} & \multicolumn{1}{c}{\scriptstyle 18} \\ \hline 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 \\ \hline \end{array}$$

for $\alpha = 3$. If we want to know the value of a ternary number $(s_1 s_2 \cdots s_n)_3$ of n digits, we begin with $i = x = 0$, and repeat $i := i+1$ and $x := C(s_i,\, x)$. For example, for $(122)_3$, we have $x := 0$; $x := C(1,0) = 1$; $x := C(2,1) = 5$; $x := C(2,5) = 17$, and we know $(122)_3 = (17)_{10}$. Conversely, starting with $x = 17$, we accumulate the symbols $\overline{s}[x]$ from right to left[3] while repeating $x := D(x)$ until $x = 0$ so that we obtain the ternary representation of $x = 17$. Specifically, we have $\overline{s}[17] = 2$, $x := D(17) = 5$; $\overline{s}[5] = 2$, $x := D(5) = 1$; $\overline{s}[1] = 1$, $x := D(1) = 0$, and we know $(17)_{10} = (122)_3$. The array $\overline{s}$ is called a *symbol distribution* [1], [2]. ANS generalizes a symbol distribution to an arbitrary sequence over $\mathcal{X}$. Even if we extend $\overline{s}$ to any sequence, the following holds for any non-negative integer x and any $s \in \mathcal{X}$:

$$\begin{aligned} C(\overline{s}[x], D(x)) &= x, \\ D(C(s,\, x)) &= x. \end{aligned} \tag{1}$$

For a sequence $s_1 s_2 \cdots s_n$ over $\mathcal{X}$ of size α, the binary representation of the final value of x obtained through a given symbol distribution results in a mapping from an α-ary input to a binary output. This mapping serves as the basic ANS encoder.

The main problem with basic ANS is that no codeword is output until the very last symbol of an input sequence is read. This causes to manipulate very large values of x, and makes ANS impractical. In order to solve this problem, a stream variant of ANS is introduced, in which the encoding of a single symbol is completed each time it is read. Stream ANS uses the interval

$$I = [l,\, 2l-1]$$

as subscripts of a fixed length $l > 0$ and its corresponding symbol distribution $\overline{s}[I]$. Repeating $x := C(s, x)$ may cause an overflow of x from I. Stream ANS solves this by emitting some of the less significant bits of x and keeping the more significant bits to form a new value for x so that $C(s, x)$ always lies in I. In this process, we assume that $\overline{s}[0,\, l-1]$ holds an identical copy of $\overline{s}[I]$. On this assumption, letting l_s and I_s denote the number of symbols s included in $\overline{s}[I]$ and $[l_s,\, 2l_s - 1]$, respectively, we can show the equivalence stated in Fig. 1.

```
while C(s, x) ∉ I do
  Emit mod(x, 2);  x := ⌊x/2⌋;
end while
```

|||

```
while x ∉ I_s do
  Emit mod(x, 2);  x := ⌊x/2⌋;
end while
```

Figure 1: Equivalent loops in stream ANS

Starting with having $x \in I$ but possibly without having $x \in I_s$ at the beginning of the while-loop, the loop is repeated until x belongs to I_s. Therefore, the loop is repeated $k-1$ times when the initial x is included in $[l, 2^k l_s - 1]$, and k times when it is in $[2^k l_s,\, 2l - 1]$. We can determine the value of k that satisfies $l \le 2^k l_s < 2l$ to be

$$k_s = \left\lceil \log \frac{l}{l_s} \right\rceil = \left\lceil -\log \frac{l_s}{l} \right\rceil,$$

since $l/l_s \le 2^k$. The number of bits emitted by the loop, which is the same as the number of times to repeat the loop, is given by

$$\begin{cases} k_s - 1 & \text{for} \quad l \le x < \xi_s, \\ k_s & \text{for} \quad \xi_s \le x < 2l \end{cases} \tag{2}$$

for

$$\xi_s = 2^{k_s} l_s. \tag{3}$$

From now on, we call $x \in I$ a *state*. The interval I is, therefore, the state set.

2.3 Sufficient Condition on Optimality

Let $\mathcal{P}(x)$ denote the probability of being in state $x \in I$ in encoding. It follows from (2) that the average number of emitted bits per input symbol is given by

$$\overline{L} = \sum_{s=0}^{\alpha-1} p_s \Big\{ (k_s - 1) \sum_{x=l}^{\xi_s - 1} \mathcal{P}(x) + k_s \sum_{x=\xi_s}^{2l-1} \mathcal{P}(x) \Big\}.$$

$\overline{L}$ depends generally on the ANS parameters including the symbol distribution $\overline{s}$ and on the source. However, we here consider it to be a function of the stationary distribution $\mathcal{P}$ of the set of states, and represent it by $\overline{L}(\mathcal{P})$. Then, we have

$$\begin{aligned} \overline{L}(\mathcal{P}) &= \sum_{s=0}^{\alpha-1} p_s k_s - \sum_{s=0}^{\alpha-1} p_s \sum_{x=l}^{\xi_s - 1} \mathcal{P}(x) & (4) \\ &\ge H. & (5) \end{aligned}$$

The last inequality (5) comes from the source coding theorem. Proving it directly, i.e. giving a converse coding theorem to ANS, is an open problem.

[2] The functions can be more concisely represented by the *rank* and *select* functions, which are popular in computer science [2].

[3] The ternary-to-decimal and decimal-to-ternary conversions using this idea have opposite directions in input and output orders of the digits of a ternary number. This corresponds to the fact that ANS encodes and decodes the source sequence in the opposite directions.

```
1.  let S_1 S_2 ··· S_n be the sequence to be encoded;
2.  let x := l;
3.  for i := 1, 2, ..., n do
        let s := S_i;
        while x ∉ I_s do
            Emit mod(x, 2); x := ⌊x/2⌋;
        end while
        let x := C[s, x];
    end for
4.  Output x.
```

```
1. let x := the final value output by the encoder;
2. for i := n, n − 1, ..., 1 do
       let (s, x) := D[x];  S_i = s;
       while x ∉ I do
           let x := 2x+'symbol emitted by the encoder';
       end while
   end for
3. Output S_1 S_2 ··· S_n.
```

Figure 2: Encoder (left) and decoder (right) of stream tANS

Let us introduce the following distribution [13] as a virtual stationary distribution of the state set:

$$\tilde{\mathcal{P}}(x) = \log \frac{x+1}{x}. \tag{6}$$

Then, we have

$$\sum_{x=l}^{\xi_s - 1} \tilde{\mathcal{P}}(x) = \log \frac{\xi_s}{l} = k_s + \log \frac{l_s}{l}$$

from (3), and

$$\overline{L}(\tilde{\mathcal{P}}) = -\sum_{s=0}^{\alpha-1} p_s \log \frac{l_s}{l}$$

from (4). Comparing $\overline{L}(\tilde{\mathcal{P}})$ with the definition of entropy, we know that we reach an equality in (5) when we use the distribution $\tilde{\mathcal{P}}$ in (6) and have

$$l_s = l p_s \quad \text{for every } s \in \mathcal{X}. \tag{7}$$

The combination of (6) and (7) forms a sufficient condition for $\overline{L}(\mathcal{P})$ to attain the source entropy.

2.4 Table Representation of $\overline{s}$ and tANS

A concrete form of stream ANS is thus specified by the size l of the state set I and $\overline{s}$, the symbol distribution. In a tabled variant of ANS (tANS), the symbol distribution is given explicitly as a table. As methods for defining a table, Duda [4], [5] proposed algorithms for creating directly the two arrays $C[s,\, x]$ and $D[x]$ instead of defining a symbol distribution. These arrays almost correspond to but are not equivalent to functions $C(s, x)$ and $D(x)$. They are specific to Duda's algorithm as well as to the subsequent discussions in this paper. The following algorithm is an example from his paper in 2014 (Page 19 in [5]), which he called the *precise initialization algorithm*. In this algorithm, "put$((v,\, s))$" and "getmin()" are operations on a list. The operation "put$((v,\, s))$" adds the pair $(v,\, s)$ to the list, and "getmin()" extracts the pair with the minimum first coordinate from the list.

```
1.  for s := 0, 1, ..., α − 1 do
        put((0.5/p_s, s)); x_s = l_s;
    end for
2.  for x := l, l + 1, ..., 2l − 1 do
        (v, s) := getmin(); put((v + 1/p_s, s));
        D[x] := (s, x_s); C[s, x_s] = x;
        x_s := x_s + 1;
    end for.
```

Corresponding to Eq. (1), we can show that

$$D[C[s,\, x_s]] \quad = \quad (s, x_s) \quad \text{for } s \in \mathcal{X},\ x_s \in I_s.$$

The two arrays are used in the encoder and decoder of tANS in the manner shown in Fig. 2. Figure 3 is an example of state transitions with emitted bits given by the above precise initialization algorithm. As the final step of the encoder, it has to output the final value of state x. The length to be required for this is $\log l$ bits, which is independent of the length n of the source sequence. We can ignore it for a sufficiently large n, and therefore we call $\overline{L}(\mathcal{P})$ in (4) the *average codeword length*.

s	x	8	9	10	11	12	13	14	15	$\in I$
0	Emitted			0	1	0	1	0	1	
	x_0	8	9	5	5	6	6	7	7	$\in I_0$
	$C[0, x_0]$	13	15	8	8	10	10	12	12	$\in I^{(0)}$
1	Emitted	00	10	01	11	00	10	01	11	
	x_1	2	2	2	2	3	3	3	3	$\in I_1$
	$C[1, x_1]$	9	9	9	9	14	14	14	14	$\in I^{(1)}$
2	Emitted	000	100	010	110	001	101	011	111	
	x_2	1	1	1	1	1	1	1	1	$\in I_2$
	$C[2, x_2]$	11	11	11	11	11	11	11	11	$\in I^{(2)}$

Figure 3: Example of state transitions in tANS with $\alpha = 3$, $l = 8$, $p_0 = 0.62, p_1 = 0.25, p_2 = 0.13$, and $\overline{s}[8,\, 15] = \{0, 1, 0, 2, 0, 0, 1, 0\}$. $I^{(s)}$ will be defined later.

3 Asymptotic Optimality of tANS

In this section, we will show that the average codeword length of tANS introduced above approaches the source entropy in the limit of $l \to \infty$.

To do so, we first analyze the precise initialization algorithm, and show that $C[s, x_s]$ is almost proportional to x_s with proportional constant p_s^{-1}. More pre-

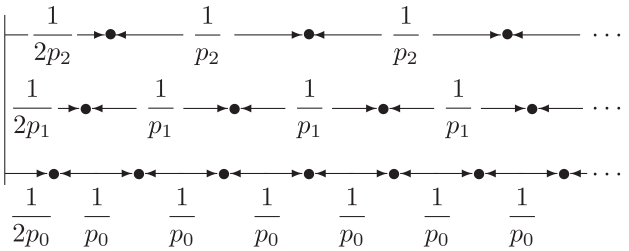

Figure 4: Example with $\alpha = 3$ of explaining Duda's precise initialization algorithm

cisely, we bound the reciprocal of $C[s, x_s]$ by the inequalities in (9) below. Then, we proceed to show our main result. We establish a simultaneous linear equation with the stationary distribution $\mathcal{P}$ as unknowns. We also give a similar equation that the virtual distribution $\tilde{\mathcal{P}}$ fulfills. We prove that the solutions to both equations, i.e. the true and virtual stationary distributions, converge as $l \to \infty$, and show for tANS to satisfy asymptotically the sufficient condition on asymptotic optimality of ANS.

3.1 Analysis of Precise Initialization Algorithm

As shown in Fig. 4, for each symbol, the algorithm produces a sequence of points with intervals of the reciprocal of the probability of the symbol. According to the order of occurring of points, the algorithm stores the corresponding symbols as the elements in $\overline{s}[I]$. In the case of Fig. 4, for example, we start with $\overline{s}[l] = 0$ and store 1, 2, 0, 0, 1, …, in order. When we focus on some symbol s in $\mathcal{X}$, we know that the sum of the intervals to the n_sth point is equal to $(n_s - 0.5)p_s^{-1}$ for $n_s = 1, 2, \ldots$. Let $n_{i,s}$ be the number of points corresponding to symbol $i \in \mathcal{X}$ in the interval of length $(n_s - 0.5)p_s^{-1}$. Then, we have

$$(n_{i,s} - 0.5)p_i^{-1} \le (n_s - 0.5)p_s^{-1} < (n_{i,s} + 0.5)p_i^{-1}.$$

Therefore, we have

$$\frac{(n_s - 0.5)p_i}{p_s} - 0.5 < n_{i,s} \le \frac{(n_s - 0.5)p_i}{p_s} + 0.5. \quad (8)$$

Note that $n_{i,s} = \lfloor (n_s - 0.5)p_i/p_s + 0.5 \rfloor \ge 0$ for any combination of positive p_s and p_i since $n_s \ge 1$. From (8), we have

$$\frac{n_s - 0.5}{p_s} - 0.5\alpha < \sum_{i=0}^{\alpha-1} n_{i,s} \le \frac{n_s - 0.5}{p_s} + 0.5\alpha.$$

Suppose that the n_sth s is stored as $\overline{s}[l + j_{n_s} - 1]$ ($j_{n_s} = 1, 2, \ldots$). Then, j_{n_s} must be in

$$\frac{n_s - 0.5}{p_s} - 1.5\alpha < j_{n_s} \le \frac{n_s - 0.5}{p_s} + 0.5\alpha.$$

The leftmost term includes -1.5α rather than -0.5α above because we have to take ties into consideration. Since $C[s, x_s] = l + j_{n_s} - 1$ for $x_s = l_s + n_s - 1$,

$$\begin{aligned} n_s &= x_s - l_s + 1, \\ j_{n_s} &= C[s, x_s] - l + 1. \end{aligned}$$

Substituting these into the above inequalities, we have

$$\begin{aligned} &\frac{x_s - l_s + 0.5}{p_s} + l - 1.5\alpha - 1 < C[s, x_s] \\ &\qquad \le \frac{x_s - l_s + 0.5}{p_s} + l + 0.5\alpha - 1. \end{aligned}$$

If we could set l_s so as to satisfy (7), which would be possible for a sufficiently large l when all p_s's are rational, the reciprocal of $C[s, x_s]$ would be bounded as

$$\begin{aligned} &\frac{p_s}{x_s + 0.5 + (0.5\alpha - 1)p_s} \le \frac{1}{C[s, x_s]} \\ &\qquad < \frac{p_s}{x_s + 0.5 - (1.5\alpha + 1)p_s}. \end{aligned} \quad (9)$$

For fixed α, since x_s grows as l grows, the above inequalities show that $C[s, x_s]$ can be approximated by x_s/p_s as l grows.

3.2 Asymptotic Optimality of tANS

In the encoder shown in Fig. 2, the encoding of a single symbol s causes a state transition from $x \in I$ to $y \in I$ in the following manner:

$$x \in I \longrightarrow \lfloor x/2 \rfloor \longrightarrow \cdots \longrightarrow x_s \in I_s \longrightarrow y = C[s, x_s].$$

We here write the state $y \in I$ after the transition by $F_s(x) = y$. Moreover, if we define

$$\begin{aligned} I^{(s)} &= \{F_s(x) \mid x \in I\} \ \text{ for } s \in \mathcal{X}, \\ \overline{D}(y) &= \{x \in I \mid \exists s \in \mathcal{X},\ F_s(x) = y\} \ \text{ for } y \in I, \end{aligned}$$

then we can show that $I^{(s)} \cap I^{(t)} = \emptyset$ ($s \ne t \in \mathcal{X}$), $|I_s| = |I^{(s)}|$ ($s \in \mathcal{X}$), and

$$I = \bigcup_{s \in \mathcal{X}} I^{(s)}.$$

Set $I^{(s)}$ contains all the states that we may reach after encoding symbol s. When we encode s and move from state $z \in I$ to $F_s(z) = x \in I^{(s)}$, the value of $x_s \in I_s$ immediately before reaching state x is unique with respect to x. That is, we have $F_s(z) = C[s, x_s]$, and $C[s, x_s]$ is bijective between I_s and $I^{(s)}$. $\overline{D}(x) \in I$ is the set of the previous states of x. From any state in $\overline{D}(x)$ for $x \in I^{(s)}$, we reach state x after encoding symbol s. Therefore, the stationary probability $\mathcal{P}(x)$ of being in state x is given as a solution to the simultaneous linear equation:

$$\begin{aligned} &\mathcal{P}(x) = p_s \mathcal{P}(\overline{D}(x)), \quad x \in I^{(s)},\ s \in \mathcal{X}, \\ &\mathcal{P}(I) = \sum_{x \in I} \mathcal{P}(x) = 1. \end{aligned} \quad (10)$$

Suppose that $p_s > 0$ for all $s \in \mathcal{X}$ and l_s satisfies (7) for a large l. Then, the simultaneous linear equation (10) has a unique set of solutions. In the following, instead of showing that the stationary distribution $\{\mathcal{P}(x)\}_{x\in I}$ given as the solution approaches $\tilde{\mathcal{P}}$ in (6), we will show that $\tilde{\mathcal{P}}$ fulfills the equations in the limit of $l \to \infty$. In order to show this, we introduce the following lemma:

Lemma 1 (Lemma 1 in [14]) *For set $\overline{D}(x)$ of states, from which we reach state $x \in I^{(s)}$ via $x_s \in I_s$ when encoding symbol $s \in \mathcal{X}$, the following holds.*

$$\tilde{\mathcal{P}}(\overline{D}(x)) = \log \frac{x_s+1}{x_s}.$$

Proof: Define $J_{\mathrm{L}} = [l, \xi_s - 1]$ and $J_{\mathrm{R}} = [\xi_s,\, 2l-1]$ for each symbol $s \in \mathcal{X}$. It follows from (3) that the number of bits emitted when encoding s is $k_s - 1$ bits if $x \in J_{\mathrm{L}}$, and k_s bits if $x \in J_{\mathrm{R}}$. Therefore, for x_s that is uniquely determined by x, the state set $\overline{D}(x)$ is given by either

$$J_{\mathrm{L}} \cap \{z \mid 2^{k_s-1}x_s \le z < 2^{k_s-1}(x_s+1)\} \text{ or}$$
$$J_{\mathrm{R}} \cap \{z \mid 2^{k_s}x_s \le z < 2^{k_s}(x_s+1)\}$$

if $l \le 2^{k_s-1}x_s$, and by

$$[l,\ 2^{k_s-1}(x_s+1)-1] \cup [2^{k_s}x_s,\ 2l-1]$$

if $2^{k_s-1}x_s < l$. In all of these cases,

$$\tilde{\mathcal{P}}(\overline{D}(x)) = \sum_{z=2^{k_s-1}x_s}^{2^{k_s-1}(x_s+1)-1} \log\frac{z+1}{z} = \log\frac{x_s+1}{x_s},$$

$$\tilde{\mathcal{P}}(\overline{D}(x)) = \sum_{z=2^{k_s}x_s}^{2^{k_s}(x_s+1)-1} \log\frac{z+1}{z} = \log\frac{x_s+1}{x_s},$$

$$\begin{aligned}\tilde{\mathcal{P}}(\overline{D}(x)) &= \sum_{z=l}^{2^{k_s-1}(x_s+1)-1} \log\frac{z+1}{z} + \sum_{z=2^{k_s}x_s}^{2l-1} \log\frac{z+1}{z} \\ &= \log\frac{x_s+1}{x_s}\end{aligned}$$

hold. Q.E.D.

Define

$$\begin{aligned}\tilde{p}_s(x) &= \log\frac{x+1}{x}\left(\log\frac{x_s+1}{x_s}\right)^{-1} \qquad (11)\\ &= \tilde{\mathcal{P}}(x)/\tilde{\mathcal{P}}(\overline{D}(x))\end{aligned}$$

for each $x \in I$ and such x_s that $x = C[s,\, x_s]$. Then, $\{\tilde{\mathcal{P}}(x)\}_{x\in I}$ is a solution to the simultaneous linear equation that is obtained by replacing p_s in (10) with $\tilde{p}_s(x)$. Namely, we have

$$\begin{aligned}&\tilde{\mathcal{P}}(x) = \tilde{p}_s(x)\tilde{\mathcal{P}}(\overline{D}(x)), \quad x \in I^{(s)},\, s \in \mathcal{X}, \\ &\tilde{\mathcal{P}}(I) = 1.\end{aligned} \qquad (12)$$

Note that $\{\tilde{p}_s(x)\}$ is a probability distribution over $\mathcal{X}$ and not over I, i.e. $\sum_{s\in\mathcal{X}} \tilde{p}_s(x) = 1$.

The following theorem states that we can approximate p_s by $\tilde{p}_s(x)$ in an arbitrary precision.

Theorem 1 *For an arbitrary $\varepsilon > 0$, there exists $L_0 = L_0(\varepsilon)$ such that, for an arbitrary $l \ge L_0$, we have*

$$|\tilde{p}_s(x) - p_s| < \varepsilon$$

for all states $x \in I$ and symbols s with $x = C[s, x_s]$.

Proof: For $x \in I$ that we reach after encoding $s \in \mathcal{X}$, since we have $x = C[s, x_s]$ and inequalities (9),

$$\begin{aligned}1 + \frac{p_s}{x_s + 0.5 + (0.5\alpha - 1)p_s} &\le 1 + \frac{1}{x} \\ &< 1 + \frac{p_s}{x_s + 0.5 - (1.5\alpha + 1)p_s}.\end{aligned} \qquad (13)$$

From the definition in (11),

$$\tilde{p}_s(x) = \ln\left(1+\frac{1}{x}\right)\left(\ln\left(1+\frac{1}{x_s}\right)\right)^{-1}.$$

Noting that $\eta(x)/x < \ln(1+1/x) \le 1/x$ for $\eta(x) = 1 - 1/(2x)$, we have

$$\frac{\eta(x_s)}{x_s} < \ln\left(1+\frac{1}{x_s}\right) \le \frac{1}{x_s},$$

and from (13)

$$\begin{aligned}\frac{\eta(x)p_s}{x_s + 0.5 + (0.5\alpha - 1)p_s} &< \ln\left(1+\frac{1}{x}\right) \\ &< \frac{p_s}{x_s + 0.5 - (1.5\alpha + 1)p_s}.\end{aligned}$$

Therefore, we have

$$\begin{aligned}\frac{\eta(x)p_s x_s}{x_s + 0.5 + (0.5\alpha - 1)p_s} &< \tilde{p}_s(x) \\ &< \frac{p_s x_s}{\eta(x_s)(x_s + 0.5 - (1.5\alpha + 1)p_s)}.\end{aligned}$$

Define $g(\alpha, p) = (0.5\alpha - 1)p + 0.5$ and $h(\alpha, p) = (1.5\alpha + 1)p - 0.5$. Then, the above inequalities become

$$p_s\left(1 - \frac{1}{2x}\right)\left(1 - \frac{g(\alpha,p_s)}{x_s + g(\alpha,p_s)}\right) < \tilde{p}_s(x),$$
$$\tilde{p}_s(x) < p_s\left(1 + \frac{1}{2x_s - 1}\right)\left(1 + \frac{h(\alpha,p_s)}{x_s + h(\alpha,p_s)}\right).$$

Therefore,

$$\begin{aligned}p_s - \tilde{p}_s(x) &< \frac{1}{2x} + \frac{g(\alpha,p_s)}{x_s + g(\alpha,p_s)} - \frac{1}{2x}\frac{g(\alpha,p_s)}{x_s + g(\alpha,p_s)} \\ &\le \frac{1}{2x} + \left|\frac{g(\alpha,p_s)}{x_s + g(\alpha,p_s)}\right| + \frac{1}{2x}\left|\frac{g(\alpha,p_s)}{x_s + g(\alpha,p_s)}\right|,\end{aligned}$$

$$\begin{aligned}\tilde{p}_s(x) - p_s &< \frac{1}{2x_s - 1} + \frac{h(\alpha,p_s)}{x_s + h(\alpha,p_s)} \\ &\qquad + \frac{1}{2x_s - 1}\frac{h(\alpha,p_s)}{x_s + h(\alpha,p_s)} \\ &\le \frac{1}{2x_s - 1} + \left|\frac{h(\alpha,p_s)}{x_s + h(\alpha,p_s)}\right| \\ &\qquad + \frac{1}{2x_s - 1}\left|\frac{h(\alpha,p_s)}{x_s + h(\alpha,p_s)}\right|.\end{aligned}$$

Let us define $L_0(\varepsilon) = c\varepsilon^{-1}$ for some constant $c > 0$. Since $l \le x$ and $lp_s \le x_s$ for any $l \ge L_0 = L_0(\varepsilon)$, we have

$$\begin{aligned} \frac{1}{2x} &\le \frac{1}{2c}\,\varepsilon, \\ \Bigl|\frac{g(\alpha,p_s)}{x_s + g(\alpha,p_s)}\Bigr| &\le \frac{|g(\alpha,p_s)|}{cp_s + \varepsilon g(\alpha,p_s)}\,\varepsilon, \\ \frac{1}{2x_s - 1} &\le \frac{1}{2cp_s - \varepsilon}\,\varepsilon, \\ \Bigl|\frac{h(\alpha,p_s)}{x_s + h(\alpha,p_s)}\Bigr| &\le \frac{|h(\alpha,p_s)|}{cp_s + \varepsilon h(\alpha,p_s)}\,\varepsilon. \end{aligned}$$

From the assumption that $p_s > 0$ for all s, we can choose c such that the following holds:

$$\max_{s\in\mathcal{X}} \Biggl\{ \frac{1}{2c} + \frac{|g(\alpha,p_s)|}{cp_s + \varepsilon g(\alpha,p_s)} + \frac{1}{2c}\frac{|g(\alpha,p_s)|}{cp_s + \varepsilon g(\alpha,p_s)}, \frac{1}{2cp_s - \varepsilon} + \frac{|h(\alpha,p_s)|}{cp_s + \varepsilon h(\alpha,p_s)} + \frac{1}{2cp_s - \varepsilon}\frac{|h(\alpha,p_s)|}{cp_s + \varepsilon h(\alpha,p_s)} \Biggr\} < 1.$$

By using such a value of c, we can have

$$|\tilde{p}_s(x) - p_s| < \varepsilon$$

for any $x \in I$ and $s \in \mathcal{X}$. Q.E.D.

Equation (10) is a simultaneous equation with unknowns $\{\mathcal{P}(x)\}$ whose coefficient matrix is determined by $\{p_s\}$. By the definition of $\{\tilde{\mathcal{P}}(x)\}$ and $\{\tilde{p}_s(x)\}$, we know that $\{\tilde{\mathcal{P}}(x)\}$ is a solution to the similar simultaneous equation whose coefficient matrix is given by replacing $\{p_s\}$ by $\{\tilde{p}_s(x)\}$. Theorem 1 shows that both coefficient matrices asymptotically coincide with each other as $l \to \infty$. Therefore, their solutions, namely, the true stationary distribution $\{\mathcal{P}(x)\}$ and the virtual distribution $\{\tilde{\mathcal{P}}(x)\}$ supposed in (6) converge. As $l \to \infty$, we can also approximate l_s in (7) in any precision. Thus, we asymptotically attain the sufficient condition that the equality in (5) holds.

4 Conclusion

We have given a sufficient condition for stream ANS to achieve the source entropy. We have also shown that tANS with Duda's precise initialization algorithm asymptotically attains the condition as the size of the state set grows. The next problem naturally raised may be to analyze the redundancy, namely, to show the rate of convergence. This problem is hard to tackle at this moment because the last discussion after Theorem 1 is more intuitive than quantitative. Before proceeding to the next step, we have to make our discussion more rigorous. Another problem to be solved is to make clear whether or not our sufficient condition on the optimality of stream ANS is also a necessary condition.

Acknowledgment

The first author is supported by the JSPS Kakenhi grant number JP17K00004.

References

[1] D. Dubé and H. Yokoo, Empirical evaluation of the effect of the symbol distribution on the performance of ANS, *The 41st Symposium on Information Theory and its Applications (SITA2018)*, Poster, Iwaki, 2018.

[2] D. Dubé and H. Yokoo, Fast construction of almost optimal symbol distributions for asymmetric numeral systems, *2019 IEEE International Symposium on Information Theory (ISIT2019)*, pp. 1682–1686, Paris, France, July 2019.

[3] J. Duda, Optimal encoding on discrete lattice with translational invariant constrains using statistical algorithms, `arXiv:0710.3861`, 2007.

[4] J. Duda, Asymmetric numeral systems, `arXiv:0902.0271v5 [cs.IT]`, 2009.

[5] J. Duda, Asymmetric numeral systems: Entropy coding combining speed of Huffman coding with compression rate of arithmetic coding, `arXiv:1311.2540v2 [cs.IT]`, 2014.

[6] J. Duda, K. Tahboub, N. J. Gadgil, and E. J. Delp, The use of asymmetric numeral systems as an accurate replacement for Huffman coding, *Picture Coding Symposium (PCS)*, pp. 65–69, Cairns, Australia, 2015.

[7] J. Duda and M. Niemiec, Lightweight compression with encryption based on asymmetric numeral systems, `arXiv:1612.04662`, 2016.

[8] H. Fujisaki, Invariant measures for the subshifts associated with the asymmetric binary systems, *International Symposium on Information Theory and its Applications (ISITA2018)*, pp. 675–679, Singapore, October, 2018.

[9] F. Giesen, Interleaved entropy coders, `arXiv:1402.3392v1 [cs.IT]`, 2014.

[10] A. Moffat and M. Petri, ANS-based index compression, *2017 ACM Conference on Information and Knowledge Management (CIKM '17)*, pp. 677–686, Singapore, 2017.

[11] A. Moffat and M. Petri, Index compression using byte-aligned ANS coding and two-dimensional contexts, *International Conference on Web Search and Data Mining*, pp. 405–413, Marina del Rey, California, USA, February 2018,

[12] S. M. Najmabadi, Z. Wang, Y. Baroud, and S. Simon, High throughput hardware architectures for asymmetric numeral systems entropy coding, *2015 9th International Symposium on Processing and Analysis (ISPA)*, pp. 256–259, Zagreb, Croatia, 2015.

[13] H. Yokoo, On the stationary distribution of asymmetric binary systems, *2016 IEEE International Symposium on Information Theory (ISIT2016)*, pp. 11–15, Barcelona, Spain, July 2016.

[14] H. Yokoo, On the stationary distribution of asymmetric numeral systems, *International Symposium on Information Theory and its Applications (ISITA2016)*, pp. 662–666, Monterey, CA, USA, 2016.