AIFV Codes
(Almost Instantaneous Fixed-to-Variable Length Codes) and Their Extended Codes

Hirosuke Yamamoto
The University of Tokyo
i.i.d. cases

Noiseless Source Coding

Uniquely Decodable Codes

new, fundamental, interesting results
Outline

1. Instantaneous codes and almost instantaneous codes in the class of uniquely decodable FV (fixed-to-variable length) codes.
2. Binary AIFV code (code trees, encoding, decoding, performance)
3. Total number of binary AIFV code trees.
5. Dynamic (adaptive) AIFV code
6. Binary AIFV-\(m\) code
7. \(K\)-ary AIFV code
FV (Fixed-to-Variable length) coding

Source sequence

---

Codeword sequence

---

An i.i.d. source on finite alphabet $\mathcal{X}$

<table>
<thead>
<tr>
<th>$l_S$</th>
<th>FV code</th>
<th>$l_C$</th>
<th>VF code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed</td>
<td>Variable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Fixed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Ex. 1

Source alphabet \( \mathcal{X} = \{a, b, c, d, e\} \)

Code alphabet \( \mathcal{Y} = \{0, 1\} \)

Source probability distribution

\[
P_X(a) = 0.35, \quad P_X(b) = 0.1, \quad P_X(c) = 0.3 \quad P_X(d) = 0.15, \quad P_X(e) = 0.1
\]

<table>
<thead>
<tr>
<th>( x )</th>
<th>codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>00</td>
</tr>
<tr>
<td>(b)</td>
<td>110</td>
</tr>
<tr>
<td>(c)</td>
<td>01</td>
</tr>
<tr>
<td>(d)</td>
<td>10</td>
</tr>
<tr>
<td>(e)</td>
<td>111</td>
</tr>
</tbody>
</table>

Source sequence: \( a \ c \ e \ d \ c \ a \ b \ a \)

Codeword sequence: \( 00 \ 10 \ 111 \ 01 \ 10 \ 00 \ 110 \ 00 \)
**Code tree**

**Ex. 1**

Source alphabet \( \mathcal{X} = \{a, b, c, d, e\} \)

Code alphabet \( \mathcal{Y} = \{0, 1\} \)

Source probability distribution

\[
P_X(a) = 0.35, \quad P_X(b) = 0.1, \quad P_X(c) = 0.3, \quad P_X(d) = 0.15, \quad P_X(e) = 0.1
\]

<table>
<thead>
<tr>
<th>(x)</th>
<th>codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>00</td>
</tr>
<tr>
<td>(b)</td>
<td>110</td>
</tr>
<tr>
<td>(c)</td>
<td>01</td>
</tr>
<tr>
<td>(d)</td>
<td>10</td>
</tr>
<tr>
<td>(e)</td>
<td>111</td>
</tr>
</tbody>
</table>

![Code tree diagram](image)

Instantaneous code

Prefix code
Huffman code, Kraft inequality, and McMillan theorem

**Huffman code** is optimal. (Huffman, 1952)

Kraft Inequality

\[ \sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1 \]

Kraft Theorem

(McMillan, 1956)

Mcmillan Theorem

(McMillan, 1949)

Uniquely decodable codes

Instantaneous codes

(Prefix codes)

\( l(x) \): codeword length of source symbol \( x \)
Huffman code, Kraft inequality, and McMillan theorem

**Huffman code** is optimal. (Huffman, 1952)

**Kraft Inequality**

\[
\sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1
\]

**Kraft Theorem** (Kraft, 1949)

**McMillan Theorem** (McMillan, 1956)

**Uniquely decodable codes**

**Instantaneous codes** (Prefix codes)

\(l(x)\): codeword length of source symbol \(x\)
Huffman code, Kraft inequality, and McMillan theorem

**Huffman code** is optimal. (Huffman, 1952)

**Kraft Inequality**

\[ \sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1 \]

**Kraft Theorem** (Kraft, 1949)

**McMillan Theorem** (McMillan, 1956)

**Implicit assumption for i.i.d sources:**
A single fixed code tree is optimal.

**Multiple code trees:** we can construct a uniquely decodable FV code that attains better compression than the Human code.
Hierarchy of FV codes

Decoding delay may become very large.

Decoding delay:
- Binary AIFV codes at most 2 bits
- Binary AIFV-$m$ codes at most $m$ bits
- K-ary AIFV codes at most 1 code symbol

FV codes
- Uniquely decodable codes
- AIFV codes (almost instantaneous)
- Instantaneous codes (Prefix codes)
Binary AIFV codes

(Yamamoto, Wei, 2013)
(Yamamoto, Tsuchihasi, Honda, 2015)

Ex.2 Source alphabet: \(\mathcal{X} = \{a, b, c, d\}\)
Code alphabet: \(\mathcal{Y} = \{0, 1\}\)

- leaf
- Complete internal node
- Incomplete internal node
  - Master node
  - Slave node

![Diagram showing binary AIFV codes with examples of source and code alphabets.](Image)
Encoding of binary AIFV codes

Source sequence: c b d c a

Transition rule:
- Leaf: Moves to $T_0$
- Master node: Moves to $T_1$

$T_0$: Initial code tree

$T_1$: New code tree after transition
Encoding of binary AIFV codes

Source sequence: \text{c b d c a}

Codeword sequence: \text{11 10 1100 11 01}

Transition rule:
- Leaf \rightarrow T_0
- Master node \rightarrow T_1
Decoding of binary AIFV codes

Codeword sequence 111011001101
Source sequence C

\[ T_0 \]

\[ T_1 \]

Transition rule

- Leaf
- Master node

\[ T_0 \]
Decoding of binary AIFV codes

Codeword sequence: 111011001101
Source sequence: c b d c a

Transition rule:
- Leaf: \( T_0 \)
- Master node: \( T_1 \)

Initial code tree:
- Initial code tree for \( T_0 \) and \( T_1 \) with nodes labeled a, b, c, d.
Decoding of binary AIFV codes

Codeword sequence: 111011001101
Source sequence: c b d c a

Uniquely decodable with at most 2-bit decoding delay
### Ex. 3

**Source alphabet:** \( \mathcal{X} = \{a, b, c, d\} \), **Code alphabet:** \( \mathcal{Y} = \{0, 1\} \)

- \( P(a) = 0.45 \)
- \( P(b) = 0.3 \)
- \( P(c) = 0.2 \)
- \( P(d) = 0.05 \)

**Entropy:** \( H(X) \approx 1.7200 \)

**Huffman code**

```
<table>
<thead>
<tr>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>b</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>c</td>
</tr>
<tr>
<td>d</td>
</tr>
</tbody>
</table>
```

**Average codeword length**

\[
L_H = 1.8.
\]
Average codeword length

**Ex. 3**

**Source alphabet:** \( \mathcal{X} = \{a, b, c, d\} \), **Code alphabet:** \( \mathcal{Y} = \{0, 1\} \)

\[
P(a) = 0.45, \ P(b) = 0.3, \ P(c) = 0.2, \ P(d) = 0.05
\]

**Entropy:** \( H(X) \approx 1.7200 \)

**Huffman code:** \( L_H = 1.8 \)

---

**Tree 0**

- \( L_0 = 1.65 \)

---

**Tree 1**

- \( L_1 = 2.1 \)

---

\[
Q(T_1|T_0) = P(c) = 0.2 \\
Q(T_0|T_1) = P(a) + P(b) + P(d) = 0.8 \\
Q(T_0) = 0.8 \quad Q(T_1) = 0.2
\]

\[
L_{AIFV} = 1.65 \times 0.8 + 2.1 \times 0.2 = 1.74
\]
Average codeword length

**Ex. 4**

Source alphabet: \( \mathcal{X} = \{a, b, c\} \), Code alphabet: \( \mathcal{Y} = \{0, 1\} \)

\( P(a) = 0.9, P(b) = P(c) = 0.05 \)

Entropy: \( H(X) \approx 0.5690 \)

Huffman code: \( L_H = 1.1 \)

\[ Q(T_1|T_0) = 0.9, Q(T_0|T_1) = 1 \]

\[ Q(T_0) = \frac{10}{19}, Q(T_1) = \frac{9}{19} \]

\[ L_{AIFV} = Q(T_0)L_0 + Q(T_1)L_1 \]

\( \approx 0.7263 \)
Encoding and decoding

Source sequence:  a  a  a  a  b
Codeword sequence:  λ  1  λ  010  (λ : null)
1010

\[ T_0 \]
- \( 0 \to a \)
- \( 0 \to 0 \)
- \( 0 \to 0 \)
- \( 0 \to b \)
- \( 0 \to c \)

\[ T_1 \]
- \( 0 \to 0 \)
- \( 1 \to 1 \)
- \( 0 \to b \)
- \( 1 \to c \)

\[ L_0 = 0.3 \]
\[ L_1 = 1.2 \]

\[ Q(T_1|T_0) = 0.9, Q(T_0|T_1) = 1 \]
\[ Q(T_0) = \frac{10}{19}, Q(T_1) = \frac{9}{19} \]

\[ L_{AIFV} = Q(T_0)L_0 + Q(T_1)L_1 \approx 0.7263 \]
Redundancy of average codeword length

\[ \text{Redundancy} = L - H(X) \]

Upper bound for Huffman codes (Ye, Young, 2002)

\[ p_{\max} = \max_{x \in \mathcal{X}} P(x) \]

This bound is tight for \( p_{\max} \geq \frac{1}{6} \approx 0.1667 \)

\[ H(X) \leq L_H < H(X) + 1 \]

\( h(p) \): binary entropy function
Redundancy of average codeword length

Upper bound for AIFV codes (Hu, Yamamoto, Honda, 2017)

This bound is tight for $p_{\text{max}} \geq 0.5$

$$f(x) = \begin{cases} 
  x^2 - 2x + 2 - h(x) & \text{if } \frac{1}{2} \leq x \leq \frac{-1+\sqrt{5}}{2}, \\
  -\frac{2x^2 + x + 2}{1+x} - h(x) & \text{if } \frac{-1+\sqrt{5}}{2} \leq x < 1.
\end{cases}$$

For $p_{\text{max}} \leq 0.5$, redundancy $\leq 0.25$.

$$H(X) \leq L_{AIFV} < H(X) + 0.5$$
### Comparison for $n = |\mathcal{X}|$

<table>
<thead>
<tr>
<th></th>
<th>Huffman code</th>
<th>Huffman code for $\mathcal{X}^2$</th>
<th>AIFV code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst-case Redundancy:</td>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Tree size:</td>
<td>$O(n)$</td>
<td>$O(n^2)$</td>
<td>$O(2n)$</td>
</tr>
<tr>
<td>Decoding delay:</td>
<td>0</td>
<td>large for $a_1$</td>
<td>at most 2 bits</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$a_1a_2 \in \mathcal{X}^2$</td>
<td></td>
</tr>
</tbody>
</table>
Comparison for $\mathcal{X} = \{a_1, a_2, \ldots, a_n\}$

\[ P_1(a_t) = \frac{t}{A_1} \]

\[ P_2(a_t) = \frac{t^2}{A_2} \]
Total number of code trees

Huffman code trees for $n = |X|$.

$N_{H,n} = C_{n-1}$

$C_n \equiv \frac{1}{n+1} \binom{2n}{n}$ : Catalan number

Binary full trees

Dyck paths

bijection

$\text{D: } (0, -1)$

$\text{L: } (-1, 0)$
Total number of code trees

AIFV code trees for $n = |\mathcal{X}|$ (Sumigawa, Yamamoto, 2017)

$$N_{T_0,n} = S_{n-1}, \quad N_{T_1,n} = S_{n-1} - S_{n-2}$$

$$S_n \equiv \sum_{k=0}^{n} \frac{1}{n-k+1} \binom{2n-2k}{n-k} \binom{2n-k}{k} : \text{Large Schröder number}$$

Schröder paths

$(3, 3) \ (n-1, n-1)$

D: $(0, -1)$
L: $(-1, 0)$
S: $(-1, -1)$
Total number of code trees \[ n = |\mathcal{X}| \]

**Huffman code trees**

\[ N_{H,n} = C_{n-1} \]

\( C_n \): Catalan number

\[ \log_2 C_n = 2n + o(n) \]

**AIFV code trees**

\[ N_{T_0,n} = S_{n-1} \]

\( S_n \): Large Schröder number

\[ \log_2 S_n = n \log_2 \left( 3 + 2\sqrt{2} \right) + o(n) \]

\[ \approx 2.5431n + o(n) \]
Coding of AIFV code trees (Sumigawa, Yamamoto, 2017)

\[ n = |\mathcal{X}| \]

**bomination map**

AIFV code tree \( T_0 \)

\[ \text{pre-calculation with space complexity } O(n^2) \]

\[ \text{time-complexity: } O(n^2) \]

\[ \left\lceil \log_2 S_{n-1} \right\rceil \text{ bits} \]

Schrödor path

\( (3, 3) \) \( (n-1, n-1) \)

DLDLS

\( (0, 0) \)

D: \( (0, -1) \)

L: \( (-1, 0) \)

S: \( (-1, -1) \)

Encode the order of the path. We can assign a total order to all Schrödor paths.

Total order: \( L \prec S \prec D \)
Construction of the optimal AIFV code trees

Huffman code trees
Huffman algorithm: simple

AIFV code trees
Iterative optimization + IP (integer programming)
(Yamamoto, Tsuchihashi, Honda, 2015)

Iterative optimization + DP (dynamic programming)
(Iwata, Yamamoto, 2016)

Time complexity: $O(n^5)$, Space complexity: $O(n^3)$

$n = |\mathcal{X}|$
Iterative optimization of AIFV code trees
(Yamamoto, Tsuchihashi, Honda, 2015)

Average codeword length

\[
L_{AIFV} = Q(T_0) L_0 + Q(T_1) L_1 \\
= \frac{Q_{0|1} L_0 + Q_{1|0} L_1}{Q_{0|1} + Q_{1|0}} \\
= L_0 + C Q_{1|0} \\
= L_1 - C Q_{0|1}
\]

\[
Q(T_0) = \frac{Q_{0|1}}{Q_{0|1} + Q_{1|0}} \\
Q(T_1) = \frac{Q_{1|0}}{Q_{0|1} + Q_{1|0}}
\]

\[
Q_{0|1} \equiv Q(T_0|T_1) \\
Q_{1|0} \equiv Q(T_1|T_0) \\
C \equiv \frac{L_1 - L_0}{Q_{0|1} + Q_{1|0}}
\]

which are determined from \( T_0 \) and \( T_1 \).
Iterative optimization of AIFV code trees
(Yamamoto, Tsuchihashi, Honda, 2015)

Average codeword length

\[ L_{AIFV} = Q(T_0) L_0 + Q(T_1) L_1 \]

\[ = Q_{0|1} L_0 + Q_{1|0} L_1 \]

\[ = \frac{Q_{0|1} L_0 + Q_{1|0} L_1}{Q_{0|1} + Q_{1|0}} \]

\[ = L_0 + C Q_{1|0} \]

\[ = L_1 - C Q_{0|1} \]

\[ C \equiv \frac{L_1 - L_0}{Q_{0|1} + Q_{1|0}} \]

\[ T_0 \text{ and } T_1 \]

\[ T_0 \]

\[ T_1 \]
Iterative optimization of AIFV code trees
(Yamamoto, Tsuchihashi, Honda, 2015)

Step 1: Initialization
\[ C = C_{\text{init}}(= 2 - \log_2 3) \]

Step 2: For a given \( C \),
- obtain \( T_0 \) that minimizes
  \[ L_0 + C Q_{1|0} \]
- obtain \( T_1 \) that minimizes
  \[ L_1 - C Q_{0|1} \]

Step 3: Calculate \( Q_{0|1}, Q_{1|0}, L_0, L_1, C \) for obtained \( T_0 \) and \( T_1 \).

Step 4: If \( C \) changes, go back to Step 2 with new \( C \).

If \( C \) changes, \( L_{AIFV} \) strictly decreases.
If \( C \) does not change, then \( T_0 \) and \( T_1 \) are optimal.
Iterative optimization of AIFV code trees
(Yamamoto, Tsuchihashi, Honda, 2015)

Step 1 : Initialization
\[ C = C_{\text{init}} = 2 - \log_2 3 \]

Step 2 : For a given \( C \),
obtain \( T_0 \) that minimizes
\[ L_0 + C Q_{1|0} \]
obtain \( T_1 \) that minimizes
\[ L_1 - C Q_{0|1} \]

Step 3 : Calculate \( Q_{0|1}, Q_{1|0}, L_0, L_1, C \) for obtained \( T_0 \) and \( T_1 \).

Step 4 : If \( C \) changes, go back to Step 2 with new \( C \).

This optimization converges with a finite number of repetition times.

IP (Yamamoto, Tsuchihashi, Honda, 2015)
DP (Iwata, Yamamoto, 2016)

If \( C \) changes, \( L_{\text{AIFV}} \) strictly decreases.
If \( C \) does not change, then \( T_0 \) and \( T_1 \) are optimal.
Easy construction of AIFV codes

Construction of the optimal AIFV codes for a given source.

Iterative Optimization + IP or DP \rightarrow \text{Not easy}

The optimal AIFV code is not necessary, but we want to attain a better compression than the Huffman code, especially $H(X) \leq L_{AIFV} < H(X) + 0.5$.

\[ \text{Easy construction :} \]
AIFV code trees $T_0$ and $T_1$ can be constructed from the Huffman tree.
Easy construction of an AIFV code from Huffman code
(Hu, Yamamoto, Honda, 2017)

For depth $D \geq 2$, if a leaf $a$ satisfies $2p_b < p_a$, 

Huffman tree

\[
\begin{align*}
\text{If } & \ p_{\text{max}} < \frac{-1 + \sqrt{5}}{2}, \\
T_0 &= T_{\text{base}} \\
T_1 &= \\
\text{then } & \\
H(X) \leq L_{\text{AIFV}} < H(X) + 0.5
\end{align*}
\]
Easy construction of an AIFV Code from Huffman code
(Hu, Yamamoto, Honda, 2017)

If \( \frac{-1 + \sqrt{5}}{2} \leq p_{\text{max}} < 1 \),

\[
T_{\text{base}}
\]

\[\begin{array}{c}
0 \\
1
\end{array}
\]

\[b \quad a\]

\[
\begin{array}{c}
T_0 \\
0 \\
1
\end{array}
\]

\[
\begin{array}{c}
T_1 \\
0 \\
1
\end{array}
\]

\[b \quad a\]

\[
H(X) \leq L_{\text{AIFV}} < H(X) + 0.5
\]
Dynamic (Adaptive) Huffman code and AIFV code

Statistic Huffman code:
A fixed code tree is used.

Dynamic (Adaptive) Huffman code:
After each one symbol in a source sequence is encoded, the code tree is updated based on the frequency distribution.

Vitter algorithm (Vitter, 1989)

Statistic AIFV code: Optimal code. Easy construction.

Dynamic (Adaptive) AIFV code:
Dynamic (Adaptive) AIFV code based on dynamic Huffman code

(Hiraoka, Yamamoto, 2017)

It is possible to encode $x_i$ directly from $T_H$ by considering only sibling nodes on the codeword path of $x_i$. 

Source sequence: $x_1 x_2 \cdots x_i \cdots$
### Canterbury Corpus

<table>
<thead>
<tr>
<th>File Name</th>
<th>Size (bytes)</th>
<th>Dynamic Huffman (Vitter)</th>
<th>Dynamic AIFV</th>
</tr>
</thead>
<tbody>
<tr>
<td>alice29.txt</td>
<td>152,089</td>
<td>4.618</td>
<td>4.618</td>
</tr>
<tr>
<td>asyoulik.txt</td>
<td>125,179</td>
<td>4.851</td>
<td>4.851</td>
</tr>
<tr>
<td>cp.html</td>
<td>24,603</td>
<td>5.303</td>
<td>5.303</td>
</tr>
<tr>
<td>fields.c</td>
<td>11,150</td>
<td>5.121</td>
<td>5.121</td>
</tr>
<tr>
<td>grammar.lsp</td>
<td>3,721</td>
<td>4.848</td>
<td>4.848</td>
</tr>
<tr>
<td>kennedy.xls</td>
<td>1,029,744</td>
<td>3.596</td>
<td>3.598</td>
</tr>
<tr>
<td>lcet10.txt</td>
<td>426,754</td>
<td>4.699</td>
<td>4.699</td>
</tr>
<tr>
<td>plrabn12.txt</td>
<td>481,861</td>
<td>4.578</td>
<td>4.578</td>
</tr>
<tr>
<td>ptt5</td>
<td>513,216</td>
<td>1.664</td>
<td>1.347</td>
</tr>
<tr>
<td>sum</td>
<td>38,240</td>
<td>5.430</td>
<td>5.432</td>
</tr>
<tr>
<td>xargs.1</td>
<td>4,227</td>
<td>5.087</td>
<td>5.087</td>
</tr>
<tr>
<td>file1</td>
<td>1,372,962</td>
<td>2.112</td>
<td>1.903</td>
</tr>
<tr>
<td>file2</td>
<td>1,268,406</td>
<td>2.736</td>
<td>2.570</td>
</tr>
</tbody>
</table>

(Hiraoka, Yamamoto, 2017)
Extension of binary AIFV codes

Huffman code: one code tree
no decoding delay
redundancy: \( H(X) \leq L_H < H(X) + 1 \)

AIFV code: two code trees
at most 2-bit decoding delay
redundancy: \( H(X) \leq L_{AIFV} < H(X) + 0.5 \)

Can we reduce the redundancy if we use more code trees and allow more decoding delay?
Extension of binary AIFV codes

Huffman code: one code tree
no decoding delay
redundancy: \( H(X) \leq L_H < H(X) + 1 \)

AIFV code: two code trees
at most 2-bit decoding delay
redundancy: \( H(X) \leq L_{AIFV} < H(X) + 0.5 \)

AIFV-\( m \) code: \( m \) code trees
at most \( m \)-bit decoding delay
redundancy: \( H(X) \leq L_{AIFV-m} < H(X) + \frac{1}{m} \)
(proved for \( 2 \leq m \leq 4 \) and conjectured for \( m \geq 5 \))
Example of AIFV-\( m \) codes

Ex. 5

\begin{align*}
S1: & \quad P_X = (0.900, 0.050, 0.049, 0.001) \\
S2: & \quad P_X = (0.33, 0.3, 0.119, 0.1, 0.05, 0.04, 0.03, 0.02, 0.01, 0.001)
\end{align*}

Average code length (Kawai, Iwata, Yamamoto, 2017)

<table>
<thead>
<tr>
<th>( m )</th>
<th>(Huffman)</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>( H(X) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>1.200</td>
<td>0.6252</td>
<td>0.4925</td>
<td>0.4395</td>
<td>0.4186</td>
<td>0.3373</td>
</tr>
<tr>
<td>S2</td>
<td>2.336</td>
<td>2.2963</td>
<td>2.2865</td>
<td>2.2841</td>
<td>2.2839</td>
<td>2.2719</td>
</tr>
</tbody>
</table>

AIFV-\( m \) code: No. of code trees: \( m \)

Decoding delay: at most \( m \) bits

Redundancy: \( H(X) \leq L_{AIFV-m} < H(X) + \frac{1}{m} \)

(proved for \( 2 \leq m \leq 4 \) and conjectured for \( m \geq 5 \))
Binary AIFV-\( m \) codes  (Hu, Yamamoto, Honda, 2017)

Binary AIFV-2 code trees

\[
\begin{align*}
&T_0  \\
&\quad \quad \quad 0 \quad 1  \\
&\quad \quad \quad \quad \quad 0 \quad 1  \\
&\quad \quad \quad \quad \quad \quad 0 \quad c  \\
&a \quad b \quad c \quad d
\end{align*}
\]

\[
\begin{align*}
&T_1  \\
&\quad \quad \quad 0 \quad 1  \\
&\quad \quad \quad \quad \quad 0 \quad 1  \\
&\quad \quad \quad \quad \quad \quad 0 \quad c  \\
&a \quad b \quad 0 \quad c \quad d
\end{align*}
\]
Binary AIFV-\(m\) codes (Hu, Yamamoto, Honda, 2017)

\[ T_k \ (1 \leq k \leq m-1) \]

Code trees: \( T_0, T_1, \cdots, T_{m-1} \)

\( k \) internal and/or master nodes

slave-1 node must exist.

Source symbols are assigned to master nodes with degrees 0 to \( m-1 \).
Binary AIFV-$m$ codes (Hu, Yamamoto, Honda, 2017)

Example 6: $\mathcal{X} = \{a, b, c, d\}$, $m = 3$

Transition rule of code trees:
- $x_i$ is encoded at a master node of degree $k$.
- We use $T_k$ to encode $x_{i+1}$.

```
0 1 1 1 0 1 1 0
a c c b d
```

a c c b d
0 11 11 01 1100
Binary AIFV-\(m\) codes  
(Hu, Yamamoto, Honda, 2017)

Ex. 7  \(\mathcal{X} = \{a, b, c\}, m = 3\)

Transition rule of code trees:
\(x_i\) is encoded at a master node of degree \(k\).

We use \(T_k\) to encode \(x_{i+1}\).

\[
\begin{align*}
\lambda & \quad \lambda & \quad 1 & \quad 0000 & \lambda & \quad 0011 \\
(a & \quad a & \quad a & \quad b & \quad a & \quad c) \\
100000011
\end{align*}
\]
Redundancy of Binary AIFV-$m$ codes

\[ p_{\text{max}} = \max_{x \in \mathcal{X}} P(x) \]

(AIFV-3 code)

Redundancy is less than 0.25 for \( p_{\text{max}} \leq 0.5 \).

(AIFV-4 code)
**K-ary AIFV codes** \((K \geq 3)\)

\[\mathcal{Y} = \{0, 1, 2, \ldots, K - 1\}\]

Root of \(T_k\) \((0 \leq k \leq K - 2)\)

- Transition rule of code trees:
  - \(x_i\) is encoded at a master node of degree \(k\).

- We use \(T_k\) to encode \(x_{i+1}\).

(Yamamoto, Wei, 2013)

(Yamamoto, Tsuchihasi, Honda, 2015)

Master node of degree \(k\) \((1 \leq k \leq K - 2)\)

(Leaf : m.n. of degree 0)
K-ary AIFV codes ($K \geq 3$)

Ex. 8 $K=4$ $\mathcal{Y} = \{0, 1, 2, 3\}$

Fig. 8. An example of 4-ary AIFV code trees.

TABLE II

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$T_0$</th>
<th>$T_1$</th>
<th>$T_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>3</td>
<td>0123</td>
<td>0123</td>
<td>0123</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
<td>e</td>
<td>f</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>f</td>
<td>g</td>
</tr>
<tr>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
</tr>
<tr>
<td>f</td>
<td>g</td>
<td>h</td>
<td>i</td>
</tr>
<tr>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
</tr>
</tbody>
</table>

$T_k :$ 0 0 1 2 1 0 2 2 0 1

Source sequence: a b a c g c e b b d

Codeword sequence: 0 1 1 31 30 2 33 30 1 1
**K-ary AIFV codes** \( (K \geq 3) \)

**Ex.9** \( K=4 \) \( \mathcal{Y} = \{0, 1, 2, 3\} \)

![Diagram of AIFV codes with incomplete roots](image)

\[ T_k : \quad 0 \quad 1 \quad 2 \quad 1 \quad 0 \quad 2 \quad 0 \quad 1 \quad 2 \quad 2 \]

source sequence: \( b \ a \ d \ b \ a \ c \ g \ a \ e \ c \)

codeword sequence: \( 0 \ \lambda \ 32 \ 10 \ \lambda \ 31 \ 13 \ \lambda \ 33 \ 31 \)
Comparison for $\mathcal{X} = \{a_1, a_2, \cdots, a_n\}$

\[ P_1(a_t) = \frac{t}{A_1} \]

\[ P_2(a_t) = \frac{t^2}{A_2} \]
AIVF code

FV (fixed-to-variable length) code
- Huffman code
- AIFV code

VF (variable-to-fixed length) code
- Tunstall code
- AIVF code
  (Yamamoto, Yokoo, 2001)
Conclusion

Binary AIFV-\(m\) code with \(m\) code trees and at most \(m\) bits decoding delay.

\[
H(X) \leq L_{AIFV-m} < H(X) + \frac{1}{m} \quad \text{for} \ 2 \leq m \leq 4
\]

Optimal AIFV-\(m\) code can be constructed by the iterative algorithm with dynamic programming.

Dynamic (adaptive) AIFV coding can be realized by using the dynamic Huffman code and easy construction of AIFV code.

The total number of binary AIFV code trees is given by “large Schröder number”.

K-ary AIFV code  AIVF code
Open problems

Binary AIFV-$m$ code with $m$ code trees and at most $m$-bit decoding delay.

Improve the redundancy bound of binary AIFV-$m$ codes for other cases of $2 \leq m \leq 4$ and $0.5 \leq p_{\text{max}} < 1$.

In the case of $m = 2$
Can we improve the binary AIFV code?
Prove the optimality or show a counter example.

In the case of $m \geq 3$
Our binary AIFV-$m$ code is not always optimal in the class of FV codes with $m$ code trees and at most $m$-bit decoding delay.
Give another formulation of binary AIFV-$m$ codes.
Open problems

Binary AIFV-$m$ code with $m$ code trees and at most $m$-bit decoding delay.

Give non-iterative construction algorithm for the optimal AIFV-$m$ codes.

Give the optimal and simple update algorithm of dynamic (adaptive) AIFV codes.

Drive the tight redundancy bound for $K$-ary AIFV codes
Thank you for your attention.
Coding of AIFV code trees (Sumigawa, Yamamoto, 2017)

\[ n = |\mathcal{X}| \]

**Huffman coding**

\[
\lim_{{n \to \infty}} \frac{L_H(T_0)}{\lceil \log_2 S_{n-1} \rceil} \approx 1.0644
\]

**AIFV coding**

\[
\lim_{{n \to \infty}} \frac{L_{AIFV}(T_0)}{\lceil \log_2 S_{n-1} \rceil} \lesssim 1.0409
\]

**Variable length coding**

- **time-complexity:** \(O(n)\)
- **space-complexity:** \(O(n)\)

in the case that every code tree occurs with equal probability.