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

# 平均性能最適化の繰り返しアルゴリズムによる AIVF 符号の改良
# An Improvement of AIVF Codes Based on the Iterative Algorithm to Optimize the Average Performance

牧野　楓也 * 藤田 龍星 * 岩田 賢一 * Danny Dubé † 山本 博資 ‡
Fuya Makino Ryusei Fujita Ken-ichi Iwata Hirosuke Yamamoto

**Abstract**— We propose an improvement of the almost instantaneous variable-to-fixed length (AIVF) codes based on two algorithms. The first algorithm is dynamic programming proposed by Dubé and Haddad to construct AIVF codes, and the second algorithm is the iterative algorithm proposed by Fujita, Iwata, Yamamoto to optimize the average performance. As a result, simulation results demonstrate that the presented AIVF codes achieve better redundancy than the known codes in the class of AIVF codes for the stationary discrete memoryless sources.

**Keywords**— lossless data compression, AIVF codes, higher average parseword length

## 1　まえがき

無歪み情報源符号の Variable-to-Fixed(VF) 符号のクラス [1, 第 6 章] について検討する．情報源は情報源記号 $a \in \mathcal{A}$ の生起確率 $p_A(a)$ が既知である定常無記憶情報源を仮定し，以下では単に情報源 $A$ と表記する．

VF 符号は，符号器と復号器が共有する辞書に登録された可変長の記号系列ごとに情報源系列の分割を行い，分割された記号系列ごとに固定長の符号語を割り当てる情報源符号である．VF 符号のクラスの代表的な符号として Tunstall 符号 [2] がある．Tunstall 符号は，符号器と復号器が用いる辞書を固定した場合であり，辞書に登録された可変長の記号系列と符号語の対応を 1 個の分節木で表現できる．本稿では，分節木で切り出される情報源系列の部分系列を分節系列と呼ぶ．また，分節系列長の平均値を簡単のため平均分節長と呼ぶ．Tunstall 符号の符号化は情報源系列の先読みなしに可変長の分節系列に分割可能であり，瞬時分割可能符号のクラスに属する．符号語アルファベットの要素数と符号語数を $D$ で表すと，Tunstall 符号の符号化率 $R_{\mathtt{Tunstall}}(A, D)$ は，

$$R_{\mathtt{Tunstall}}(A, D) := \frac{\log D}{\mathtt{E}[L_{\mathtt{Tunstall}}(A)]} \tag{1}$$

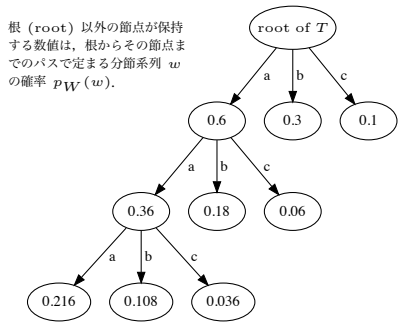
で定義され，$R_{\mathtt{Tunstall}}(A, D)$ は，瞬時分割可能符号のクラスにおいて最適な符号化率を達成する [3, 4, Problem 13.11]．本文を通して log の底は 2 とする．$\mathtt{E}[L_{\mathtt{Tunstall}}(A)]$ は，情報源 $A$ からの出力系列に対する Tunstall 符号による平均分節長 ((4) を参照) を表す．すなわち，Tunstall 符号は，瞬時分割可能な VF 符号のクラスにおいて，$\mathtt{E}[L_{\mathtt{Tunstall}}(A)]$ を最大にする符号である．Tunstall 符号の理論解析としては，文献 [5, 第 3.5 節][1, 第 6 章][6][7, 第 4 章] などがある．

情報源系列の分節系列への分割において情報源系列の先読みを許容すると，分節系列への分割に遅延が発生する．任意に長い分割遅延を符号器に許容することは伝送するシステムにおいて好ましくない．本稿では，分割遅延を高々1 記号の情報源記号に限定した VF 符号のクラスを準瞬時分割可能符号 (almost instantaneous codes) のクラスと呼ぶ．Yamamoto と Yokoo は，準瞬時分割可能符号のクラスに属する AIVF 符号を提案した [8]．Yamamoto と Yokoo の AIVF 符号は，複数の分節木を用い，符号化する情報源系列と復号する符号語系列に応じて符号器と復号器が使用する分節木を同期して取り換える．さらに，Dubé と Haddad は，AIVF 符号の平均分節長の更なる改善とともに，動的計画法を用いた分節木の構成法 (以下では，Dubé–Haddad 法と呼ぶ．) を提案した [9, 10]．

複数の分節木を用いる既存の AIVF 符号では，情報源 $A$ に対して各分節木毎に平均分節長の最大化を試みている．しなしながら，各分節木間における遷移確率から定まる定常確率を考慮した平均分節長の最大化はなされていない．本研究では，Fujita, Iwata, Yamamoto の平均性能最適化の手法 [11] で，Dubé–Haddad 法の AIVF 符号を改良し，より長い平均分節長を達成する AIVF 符号を提案する．本稿で提案する AIVF 符号は，著者らが知る範囲において，準瞬時分割可能符号のクラスの中で最長の平均分節長を達成する．

第 2 章で Tunstall 符号 [2] を概観し，第 3 章で既存の AIVF 符号 [8, 10] について簡単に説明する．第 4 章では，Dubé–Haddad 法 [10] と Fujita, Iwata, Yamamoto の平均性能最適化法 [11] に基づく AIVF 符号の改良を提案する．第 5 章はまとめである．

* 福井大学　〒 910-8507 福井県福井市文京 3 丁目 9-1, University of Fukui, 3–9–1, Bunkyo, Fukui-shi, Fukui, 910-8507, Japan
† Université Laval, Quebec City, G1V 0A6 Canada
‡ 東京大学 〒 277–0871 千葉県柏市若柴 178–4–4. The University of Tokyo, 178–4–4 Wakashiba, Kashiwa-shi, Chiba, 277–0871 Japan




| $w \in \mathcal{D}$ | $p_C(w)$ |
|---|---|
| aaa | 0.216 |
| aab | 0.108 |
| aac | 0.036 |
| ab | 0.18 |
| ac | 0.06 |
| b | 0.3 |
| c | 0.1 |

図 1. 例 1 に対する Tunstall 符号の分節木 (左) と辞書 (右).

## 2 Tunstall 符号

まず，情報源アルファベットが $\mathcal{A} = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$ であり，$p_A(\mathrm{a}) = 0.6, p_A(\mathrm{b}) = 0.3, p_A(\mathrm{c}) = 0.1$ を満たし，$D = 7$ である場合を例 1 と呼び，例 1 の場合における Tunstall 符号の分節木 $T$ と辞書 $\mathcal{D}$ を図 1 に示す．辞書 $\mathcal{D}$ は，Tunstall 符号の符号化と復号で用いられる分節系列の集合であり，辞書 $\mathcal{D}$ に登録される分節系列の個数は符号語アルファベットの要素数と等しく，$|\mathcal{D}| = D$ を仮定する[1]．

図 1 の分節木 $T$ において，節点を楕円で表し，枝の右側には枝に割り当てた情報源記号を示す．根以外の節点が保持する値は，根からその節点までのパスにおける分節系列 $w$ が生起する確率 $p_W(w)$ である．分節系列 $w$ の確率 $p_W(w)$ は，$\lambda$ を長さ 0 の空列とし，$a \in \mathcal{A}$ とし，

$$p_W(w) := \begin{cases} 1 & \text{if } w = \lambda, \\ p_W(w')\, p_A(a) & \text{else if } w = w'a \end{cases} \tag{2}$$

と再帰的に定義される．$w'$ は分節系列 $w$ の最後 1 記号を削除した系列であり，$w'a$ は系列 $w'$ の最後に $a$ を連接した系列である．

Tunstall 符号において，辞書 $\mathcal{D}$ に登録する分節系列は，根から葉までのパスで定まる情報源記号列であり，図 1 の分節木では 7 個ある．Tunstall 符号では，各分節系列 $w \in \mathcal{D}$ に対して等長の相異なる符号語を割り当てる．

Tunstall 符号の符号化で，情報源系列の先頭から分節系列 $w \in \mathcal{D}$ が切り出されて符号化される確率 $p_C(w)$ は，分節系列 $w$ の確率 $p_W(w)$ と等しく，

$$p_C(w) = \begin{cases} 0 & \text{if } w \notin \mathcal{D}, \\ p_W(w) & \text{if } w \in \mathcal{D} \end{cases} \tag{3}$$

である．それゆえ，情報源 $A$ に対する Tunstall 符号の平均分節長 $\mathtt{E}[L_{\mathtt{Tunstall}}(A)]$ は，

$$\mathtt{E}[L_{\mathtt{Tunstall}}(A)] = \sum_{w \in \mathcal{D}} p_C(w)\, l(w) \tag{4}$$

[1] 本稿を通して，集合 $\mathcal{S}$ の要素数を $|\mathcal{S}|$ と表記する．

**Algorithm 1** Tunstall($M$): $M - |\mathcal{A}| < |\mathcal{D}(T)| \le M$ を満たす Tunstall 符号の分節木 $T$ を返す．

**Require:** $M \ge |\mathcal{A}|$ と $\{p_A(a), a \in \mathcal{A}\}$

1: $T_{\mathtt{new}} \leftarrow$ ROOT+$\mathcal{A}$
2: **repeat**
3: $\quad T_{\mathtt{old}} \leftarrow T_{\mathtt{new}}$
4: $\quad T_{\mathtt{new}} \leftarrow T_{\mathtt{old}} + \{w_{\max} a, a \in \mathcal{A}\}$
5: **until** $|\mathcal{D}(T_{\mathtt{new}})| > M$
6: **return** $T_{\mathtt{old}}$

を満たす．$l(w)$ は分節系列 $w$ を構成する情報源記号の個数である．例 1 の場合，$\mathtt{E}[L_{\mathtt{Tunstall}}(A)] = 1.96$ である．

Tunstall 符号の分節木を求めるアルゴリズムを，文献 [10] の Algorithm 2 に倣い，Algorithm 1 に示す．本稿を通して ROOT は根のみからなる分節木を表す．分節木 $T$ と分節系列の集合 $\mathcal{W}$ に対して，$T + \mathcal{W}$ は，分節木 $T$ に分節系列 $w \in \mathcal{W}$ に対応するパスと節点を追加した分節木を表す．$\mathcal{D}(T)$ は分節木 $T$ における根から葉までのパスで定まる分節系列 $w$ の集合であり，分節木 $T$ で定まる辞書である．Algorithm 1 における $w_{\max}$ は，$w_{\max} \in \mathrm{argmax}_{w \in \mathcal{D}(T_{\mathtt{old}})}\ p_C(w)$ と定める．Tunstall 符号の符号語数 $D$ は $D = 1 + k(|\mathcal{A}| - 1)$ に限定され，$k$ は正整数である．

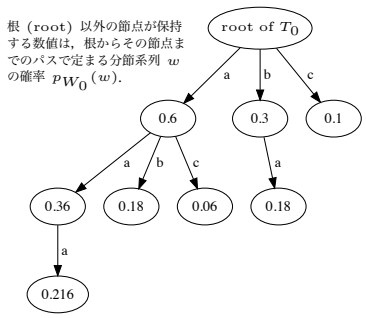
## 3 既存の AIVF 符号

Tunstall 符号を表現する分節木 $T$ の節点は，$|\mathcal{A}|$ 個の子節点を有する完全節点か 0 個の子節点を有する葉である．これに対して，Yamamoto と Yokoo は，0 個から $|\mathcal{A}| - 2$ 個の子節点を有する不完全節点を分節木に許容し，根から不完全節点までのパスで定まる分節系列を辞書に登録する改良を提案し，AIVF 符号を定義した [8]．Yamamoto と Yokoo が定義した AIVF 符号には，単一の分節木のみで構成される AIVF 符号 [8, Sect. IV] と $|\mathcal{A}| - 1$ 個の分節木で構成される AIVF 符号 [8, Sect. V] の 2 種類がある．

本稿では，$|\mathcal{A}| - 1$ 個の分節木 $T_i, i \in \mathcal{I}_{|\mathcal{A}|-1}$[2]で定義される AIVF 符号 $(T_i, i \in \mathcal{I}_{|\mathcal{A}|-1})$ を用いて，情報源 $A$ からの出力系列である情報源系列 $\boldsymbol{x} := (x_j \in \mathcal{A}, j = 1, 2, \ldots)$ の先頭から切り出しを行い，逐次的に符号化する場合を考える．第 2 章の例 1 の場合，$|\mathcal{A}| = 3$ であり，Dubé と Haddad による Algorithm 2 と Algorithm 3 を用いると，図 2 の分節木 $T_0, T_1$ が得られ，AIVF 符号 $(T_0, T_1)$ が構成される[3]．

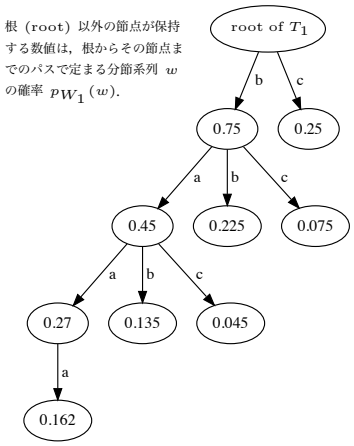
[2] 0 から $m-1$ までの $m$ 個の整数の集合を $\mathcal{I}_m := \{0, 1, \ldots, m-1\}$.

[3] 例 1 の場合，Yamamoto と Yokoo[8] の AIVF 符号 $(T_0, T_1)$ も Dubé と Haddad[10] の AIVF 符号 $(T_0, T_1)$ も同一の平均分節長である．本稿では，符号化率のより良い Dubé–Haddad 法による AIVF 符号に基づき，次章で AIVF 符号の改良を提案する．AIVF




| $w \in \mathcal{D}_0$ | $p_{C_0}(w)$ |
|---|---|
| aa | 0.144 |
| aaa | 0.216 |
| ab | 0.18 |
| ac | 0.06 |
| b | 0.12 |
| ba | 0.18 |
| c | 0.1 |

(a) $T_0$ の分節木 (左) と辞書 (右).




| $w \in \mathcal{D}_1$ | $p_{C_1}(w)$ |
|---|---|
| baa | 0.108 |
| baaa | 0.162 |
| bab | 0.135 |
| bac | 0.045 |
| bb | 0.225 |
| bc | 0.075 |
| c | 0.25 |

(b) $T_1$ の分節木 (左) と辞書 (右).

図 2. 例 1 に Dubé–Haddad 法の AIVF 符号 $(T_0, T_1)$ を用いた場合.

情報源アルファベットを $\mathcal{A} = \{a_j, j \in \mathcal{I}_{|\mathcal{A}|}\}$ と表記し，$i < j, i, j \in \mathcal{I}_{|\mathcal{A}|}$ に対して $p_A(a_i) \geq p_A(a_j)$ を仮定する．Algorithm 2 の $\mathrm{Tie}(L, a_j, R)$ は図 3 に示すように，分節木 $R$ の根と分節木 $L$ の根を新たに枝でつなぎ，枝には $a_j$ を割り当てて構成される分節木であり，新たな分節木の根は分節木 $R$ の根とする．$\mathrm{Tie}(T_0^{(d)}, a_{|\mathcal{A}|-1}, \mathrm{ROOT})$ の模式図を図 3 の右図に示す．分節木 $T_i^{(d)}$ の $d$ は，$T_i^{(d)}$ で構成される辞書 $\mathcal{D}(T_i^{(d)})$ に登録される分節系列数 $|\mathcal{D}(T_i^{(d)})|$ を表す．分節木 $T_i^1, i \in \mathcal{I}_{|\mathcal{A}|-1}$ は，Algorithm 2 の 2: で ROOT と初期化され，辞書 $\mathcal{D}(T_i^{(1)})$ に空列 $\lambda$ の分節系列のみが登録される．分節木 $T_{|\mathcal{A}|-1}^{(d)}$ は AIVF 符号 $(T_i, i \in \mathcal{I}_{|\mathcal{A}|-1})$ で用いないが，Algorithm 2 の動的計画法で利用する．分節木 $T_{|\mathcal{A}|-1}^{(d)}$ は Algorithm 2 の 1: で構成される．$T_{|\mathcal{A}|-1}^{(1)}$ は，根と 1 つの子節点と $a_{|\mathcal{A}|-1}$ を割り当てた枝で初期化され，辞書 $\mathcal{D}(T_{|\mathcal{A}|-1}^{(1)})$ に分節系列 $a_{|\mathcal{A}|-1}$ のみが登録される[4]．Algorithm 2 の 3: では，最適な分節

符号の詳細は，文献 [8, 10] を参照されたい．

[4] AIVF 符号 $(T_i, i \in \mathcal{I}_{|\mathcal{A}|-1})$ で $T_{|\mathcal{A}|-1}^{(1)}$ は必要ないが，もし，$T_{|\mathcal{A}|-1}^{(1)}$ を符号化に用いるならば，次に符号化する情報源系列の先頭の記号として $a_{|\mathcal{A}|-1}$ を仮定する．

**Algorithm 2** [10] Build($i$,$d$): $T_i^{(d)}$ を構成.

**Require:** $d \geq 1$ と $i \in \mathcal{I}_{|\mathcal{A}|}$ と $\{p_A(a_j), a_j \in \mathcal{A}\}$.

1: **if** $i = |\mathcal{A}|-1$ **then return** $\mathrm{Tie}(T_0^{(d)}, a_{|\mathcal{A}|-1}, \mathrm{ROOT})$
2: **else if** $d = 1$ **then return** ROOT
3: **else** $\mathcal{T}_i^{(d)} \leftarrow \{\mathrm{Tie}(T_0^{(l)}, a_{i+1}, T_{i+1}^{(r)}) \mid l,r \geq 1, l+r = d\}$
4: **return** $\operatorname*{argmax}_{T_i^{(d)} \in \mathcal{T}_i^{(d)}} \mathtt{E}[L_{T_i^{(d)}}(A)]$

**Algorithm 3** [10] Fill($D$): $(T_i, i \in \mathcal{I}_{|\mathcal{A}|-1})$ を構成.

1: **for** $d = 1$ **to** $D$ **do**
2: **for** $i \in \mathcal{I}_{|\mathcal{A}|-1}$ **do**
3: $T_i^d \leftarrow \mathrm{Build}(i,d)$
4: **return** $(T_i^{(D)}, i \in \mathcal{I}_{|\mathcal{A}|-1})$

木 $T_i^{(d)}$ の探索を，$\mathrm{Tie}(T_0^l, a_{i+1}, T_{i+1}^r)$, $d = l + r$ により，2 個の小規模な $T_i^{(r)}$ と $T_i^{(d)}$ で構成される候補 $\mathcal{T}_i^{(d)} := \{\mathrm{Tie}(T_0^l, a_{i+1}, T_{i+1}^r) \mid l, r \geq 1, l+r = d\}$ に絞り，$\mathcal{T}_i^{(d)}$ の要素に対して，(7) で定義される平均分節長 $\mathtt{E}[L_{T_i^{(d)}}(A)]$ が最長となる分節木 $T_i^{(d)}$ を Algorithm 2 の 4: で選択する．Algorithm 3 では，Algorithm 2 の Bulid($i$,$d$) を利用した動的計画法により $(T_i^{(d)}, i \in \mathcal{I}_{|\mathcal{A}|}, 1 \leq d \leq D)$ を求め，与えられた情報源 $A$ と符号語数 $D$ に対する AIVF 符号 $(T_i, i \in \mathcal{I}_{|\mathcal{A}|-1})$ の分節木 $T_i := T_i^{(D)}, i \in \mathcal{I}_{|\mathcal{A}|-1}$ を作成する．

Dubé–Haddad 法で求まる AIVF 符号 $(T_i, i \in \mathcal{I}_{|\mathcal{A}|-1})$ の分節木 $T_i$ において，$j \in \mathcal{I}_{|\mathcal{A}|}$ である $j$ 個の子節点を有する節点 $v_j$ は，次の性質を満たす．

1. $v_j$ が根であるとき，根 $v_j$ とその子節点を結ぶ枝のラベルは相異なる $a_{j'}, j' \in \{i, \ldots, i+j-1\}$ であり，$i + j \leq |\mathcal{A}|$ を満たす．
2. $v_j$ が根でない内節点のとき，節点 $v_j$ とその子節点を結ぶ枝のラベルは相異なる $a_{j'}, j' \in \mathcal{I}_j$ であり，$1 \leq j \leq |\mathcal{A}|$ と $j \neq |\mathcal{A}| - 1$ を満たす．
3. $v_j$ が葉のとき，$j = 0$ を満たす．

分節木 $T_i$ における根の子節点の個数は，高々 $|\mathcal{A}| - i$ 個であり，分節木 $T_0$ の根のみが $|\mathcal{A}|$ 個の子節点を有する

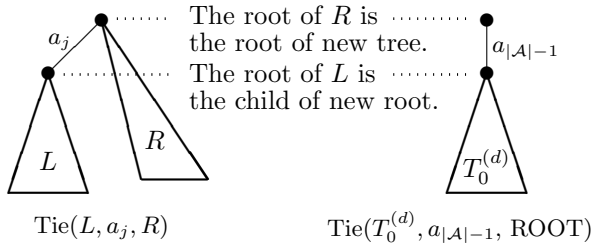



図 3. Tie の定義の模式図.

完全節点となりえる．分節木 $T_i$ の根に関しては，子節点の個数が $|\mathcal{A}|-i$ 個である根を特別に完全節点と呼び，子節点の個数が $|\mathcal{A}|-i$ 個未満である分節木 $T_i$ の根を不完全節点と呼ぶ．根以外の節点に関しては，$|\mathcal{A}|$ 個の子節点を有する節点を完全節点と呼び，$|\mathcal{A}|$ 個未満の子節点を有する節点を不完全節点と呼ぶ．

符号化と復号で用いる分節木 $T_i$ に基づく辞書 $\mathcal{D}(T_i)$ を $\mathcal{D}_i$ と略記し，

$$\mathcal{D}_i = \bigcup_{j\in\mathcal{I}_{|\mathcal{A}|-1}} \mathcal{D}_{i,j}, \quad \mathcal{D}_{i,j}\cap\mathcal{D}_{i,j'}=\emptyset \quad \text{for } j\neq j'$$

を満たす $\mathcal{D}_i$ の分割 $\mathcal{D}_{i,j}$ を次段落で定義する．$\emptyset$ は空集合であり，上式の和集合は直和集合である．

分節木 $T_i$ の $j$ 個の子節点を有する根でない不完全節点 $v_j$ に対して，根から不完全節点 $v_j$ までのパスで定まる分節系列 $w_j$ を $\mathcal{D}_{i,j}$ の要素とする．分節木 $T_i$ の根が $j$ 個の子節点を有する不完全節点 $v_j$ であれば，空列 $\lambda$ を $\mathcal{D}_{i,i+j}$ の要素とする．

AIVF 符号 $(T_i, i\in\mathcal{I}_{|\mathcal{A}|-1})$ による情報源系列 $\boldsymbol{x}$ の符号化について述べる．符号化の最初に用いる分節木は $T_0$ であり，辞書 $\mathcal{D}_0$ を用いる．情報源系列 $\boldsymbol{x}$ の語頭系列と分節木 $T_0$ の根からの分節系列との最長一致列 $w\in\mathcal{D}_0$ を求め，分節木 $T_0$ の分節系列 $w$ に対応する符号語に符号化するとともに，符号化した $w$ を $\boldsymbol{x}$ の先頭から削除する．情報源系列 $\boldsymbol{x}$ でまだ符号化されていない部分列を $\boldsymbol{x}_{\text{next}}$ と表記する．分節木 $T_i$ を用いて分節系列 $w\in\mathcal{D}_{i,j}\subset\mathcal{D}_i$ を符号化したとき，次の符号化には分節木 $T_j$ を用い，$\boldsymbol{x}_{\text{next}}$ における語頭系列と分節木 $T_j$ の根からの分節系列との最長一致列 $w\in\mathcal{D}_j$ を符号化し，削除することで $\boldsymbol{x}_{\text{next}}$ を更新する．この操作を情報源系列の最後まで繰り返し，最後に終端の処理を行い符号化を終了する．なお，現在の符号化に分節木 $T_i$ を用い，符号化される分節系列 $w\in\mathcal{D}_i$ が $w=\lambda$ であるのは，分節木 $T_i$ の根が $j$ 個の子節点を有する不完全節点であり，$\boldsymbol{x}_{\text{next}}$ の先頭が $a_k, k\in\{i+j,\ldots,|\mathcal{A}|-1\}$ のときである．符号化に対して，復号は符号化の逆の操作を行えばよい．

上記の符号化において分節木 $T_i$ の根から分節系列 $\hat{w}$ を辿る確率 $p_{W_i}(\hat{w})$ を考える．ただし，分節系列 $\hat{w}$ は，$w\in\mathcal{D}_i$ の語頭系列とする．$\hat{w}$ が $w$ の語頭であることを $\hat{w}\preceq w$ と表記し，$\hat{w}=w$ の場合も $\hat{w}\preceq w$ とする．

$i>1$ である分節木 $T_i$ では，分節系列 $w$ の先頭の情報源記号として $a_k, k\in\mathcal{I}_i$ はありえない．それゆえ，次の符号化で分節木 $T_i$ を用い，分節木 $T_i$ の根から分節系列 $\hat{w}$ を辿る確率 $p_{W_i}(\hat{w})$ は，

$$p_{W_i}(\hat{w}) = \begin{cases} 1 & \text{if } \hat{w}=\lambda, \\ \frac{p_W(\hat{w})}{\sum_{k=i}^{|\mathcal{A}|-1} p_A(a_k)} & \text{else if } \hat{w}\preceq w\in\mathcal{D}_i, \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

表 1. 例 1 に AIVF 符号 $(T_0, T_1)$ を構成した場合.

| | Dubé–Haddad 法<br>図 2 の $(T_0, T_1)$ | 提案法<br>図 4 の $(T_0, T_1)$ |
|---|---|---|
| $(\mathtt{E}[L_{T_i}(A)])_{i=0,1}$ | (1.996, 2.362) | (1.8856, 2.332) |
| 遷移行列 $Q$ | $\begin{pmatrix} 0.736 & 0.264 \\ 0.892 & 0.108 \end{pmatrix}$ | $\begin{pmatrix} 0.4096 & 0.5904 \\ 0.6120 & 0.3880 \end{pmatrix}$ |
| $(p_S(T_i))_{i=0,1}$ | $\left(\frac{223}{289}, \frac{66}{289}\right)$ | $\left(\frac{85}{167}, \frac{82}{167}\right)$ |
| $\mathtt{E}[L_{\text{AIVF}}(A)]$ | $\frac{601}{289}\approx 2.07958$ | $\frac{703}{334}\approx 2.10479$ |
| 冗長度，式 (10) | $\approx 0.054497511$ | $\approx 0.038331248$ |

である．さらに，上記の符号化において分節木 $T_i$ を用いて分節系列 $w\in\mathcal{D}_i$ を符号化する確率 $p_{C_i}(w)$ は，

$$p_{C_i}(w) = \begin{cases} 0 & \text{if } w\notin\mathcal{D}_i, \\ p_{W_i}(w) - \sum_{\substack{a\in\mathcal{A} \text{ s.t.} \\ wa\preceq w'\in\mathcal{D}_i}} p_{W_i}(wa) & \text{if } w\in\mathcal{D}_i \end{cases} \tag{6}$$

を満たす．瞬時分割可能である VF 符号を考えた場合，(6) の右辺 2 行目の第 2 項は 0 であり，(6) は (3) を準瞬時分割可能な VF 符号へ拡張した式である．例 1 の場合の計算例を図 2 に示す．

情報源 $A$ に対して，AIVF 符号 $(T_i, i\in\mathcal{I}_{|\mathcal{A}|-1})$ の分節木 $T_i$ を用いて符号化する平均分節長を $\mathtt{E}[L_{T_i}(A)]$ と表記すると次式を満たす．

$$\mathtt{E}[L_{T_i}(A)] = \sum_{w\in\mathcal{D}_i} p_{C_i}(w)\, l(w). \tag{7}$$

上記の符号化において，分節木 $T_i, i\in\mathcal{I}_{|\mathcal{A}|-1}$ を用いて符号化を行った次の符号化に分節木 $T_j, j\in\mathcal{I}_{|\mathcal{A}|-1}$ を用いる確率を $q(T_j|T_i)$ と表記すると，

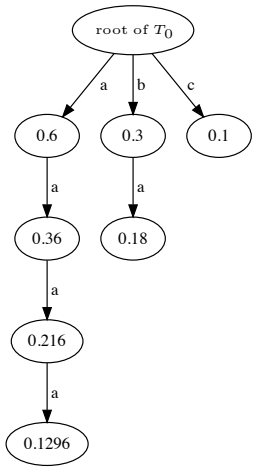
$$q(T_j|T_i) = \sum_{w\in\mathcal{D}_{i,j}} p_{C_i}(w) \tag{8}$$

で求まる．ここで，$q(T_j|T_i)$ は情報源 $A$ と $T_i$ に依存して定まる確率であり，今後，$q(T_j|T_i)$ を $q_{i,j}(T_i)$ と略記する．また，$i$ 行 $j$ 列の要素が $q_{i,j}(T_i)$ である $|\mathcal{A}|-1$ 行 $|\mathcal{A}|-1$ 列の遷移行列 $Q$ と表記する．遷移行列 $Q$ で定まるマルコフ過程の定常分布を $\boldsymbol{p}_S := (p_S(T_0), p_S(T_1), \ldots, p_S(T_{|\mathcal{A}|-1}))$ と表記すると，$\boldsymbol{p}_S$ の各要素 $p_S(T_i)$ は，$\boldsymbol{p}_S = \boldsymbol{p}_S\, Q$ と $\sum_{i\in\mathcal{I}_{|\mathcal{A}|-1}} p_S(T_i) = 1$ から求まる [12]．情報源 $A$ に対する AIVF 符号 $(T_i, i\in\mathcal{I}_{|\mathcal{A}|-1})$ の符号化で，分節木 $T_i$ を用いる確率は $p_S(T_i)$ であり，AIVF 符号 $(T_i, i\in\mathcal{I}_{|\mathcal{A}|-1})$ の平均分節長を $\mathtt{E}[L_{\text{AIVF}}(A)]$ と表記すると次式を満たす．

$$\mathtt{E}[L_{\text{AIVF}}(A)] = \sum_{i\in\mathcal{I}_{|\mathcal{A}|-1}} p_S(T_i)\, \mathtt{E}[L_{T_i}(A)]. \tag{9}$$

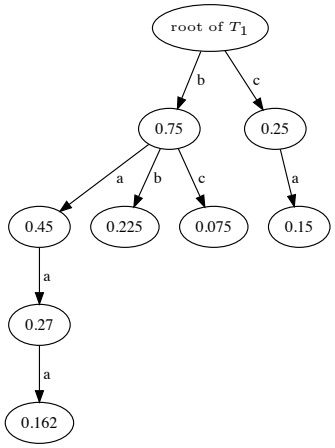
例 1 に図 2 の分節木を用いると表 1 が得られる．表 1 の冗長度は，

$$\frac{\log D}{\mathtt{E}[L_{\text{AIVF}}(A)]} - H(A) \tag{10}$$




| $w \in \mathcal{D}_0$ | $p_{C_0}(w)$ |
|---|---|
| a | 0.24 |
| aa | 0.144 |
| aaa | 0.0864 |
| aaaa | 0.1296 |
| b | 0.12 |
| ba | 0.18 |
| c | 0.1 |

(a) $T_0$ の分節木 (左) と辞書 (右).




| $w \in \mathcal{D}_1$ | $p_{C_1}(w)$ |
|---|---|
| ba | 0.18 |
| baa | 0.108 |
| baaa | 0.162 |
| bb | 0.225 |
| bc | 0.075 |
| c | 0.1 |
| ca | 0.15 |

(b) $T_1$ の分節木 (左) と辞書 (右).

図 4. 例 1 に提案する AIVF 符号 $(T_0, T_1)$ を用いた場合.

と定義され，$H(A) := \sum_{a \in \mathcal{A}} -p_A(a) \log p_A(a)$ である．

Dubé–Haddad 法の AIVF 符号 $(T_i, i \in \mathcal{I}_{|\mathcal{A}|-1})$ では，$T_i$ として，$\mathcal{T}_i^{(D)} = \{\mathrm{Tie}(T_0^{(l)}, a_{i+1}, T_{i+1}^{(r)}) \mid l, r \geq 1, l + r = D\}$ の中で $\mathtt{E}[L_{T_i^{(D)}}(A)]$ を最も長くする $T_i^{(D)}$ を定める．しかしながら，(10) で定義される冗長度を小さくするためには，AIVF 符号 $(T_i, i \in \mathcal{I}_{|\mathcal{A}|-1})$ の平均分節長 $\mathtt{E}[L_{\mathrm{AIVF}}(A)]$ を長くすることが望まれる．そのために，次章では，Fujita, Iwata, Yamamoto[11] が提案した手法を用いて，Dubé–Haddad 法を改良する．

## 4 提案する AIVF 符号

Fujita, Iwata, Yamamoto[11] の Algorithm 1 を用いて，Dubé–Haddad 法を改良するために，同アルゴリズムを本稿の表記を用いて Algorithm 4 に書き直す．Algorithm 4 の 4: における $\mathrm{NewFill}\big(D, c_{j,j+1}, j \in \mathcal{I}_{|\mathcal{A}|-2}\big)$ は，Algorithm 3 の Fill($D$) の引数に，$(c_{j,j+1}, j \in \mathcal{I}_{|\mathcal{A}|-2})$ を追加し，Algorithm 3 の 3: の Build($i$,$d$) を $\mathrm{NewBuild}\big(i, d, c_{j,j+1}, j \in \mathcal{I}_{|\mathcal{A}|-2}\big)$ に変更したものである．$\mathrm{NewBuild}\big(i, d, c_{j,j+1}, j \in \mathcal{I}_{|\mathcal{A}|-2}\big)$ は，Algorithm 2 の 4: における $\mathrm{argmax}_{T_i^{(d)} \in \mathcal{T}_i^{(d)}} \mathtt{E}[L_{T_i^{(d)}}(A)]$ を

$$\operatorname*{argmax}_{T_i^{(d)} \in \mathcal{T}_i^{(d)}} \left\{ \mathtt{E}[L_{T_i^{(d)}}(A)] - \sum_{j=0}^{i-1} c_{j,j+1} \sum_{k=0}^{j} q_{i,k}(T_i^{(d)}) + \sum_{j=i}^{m-2} c_{j,j+1} \sum_{k=j+1}^{m-1} q_{i,k}(T_i^{(d)}) \right\} \quad (11)$$

に変更したものである．

**Algorithm 4** Iterative($D$): $(T_i, i \in \mathcal{I}_{|\mathcal{A}|-1})$ を構成.

1: **for** $j \in \mathcal{I}_{|\mathcal{A}|-2}$ **do** $c_{j,j+1} \leftarrow 0$
2: **repeat**
3: $\quad \big(\hat{c}_{j,j+1}, j \in \mathcal{I}_{|\mathcal{A}|-2}\big) \leftarrow \big(c_{j,j+1}, j \in \mathcal{I}_{|\mathcal{A}|-2}\big)$
4: $\quad \big(T_i, i \in \mathcal{I}_{|\mathcal{A}|-1}\big) \leftarrow \mathrm{NewFill}\big(D, c_{j,j+1}, j \in \mathcal{I}_{|\mathcal{A}|-2}\big)$
5: $\quad$ **for** $j \in \mathcal{I}_{|\mathcal{A}|-2}$ **do** renew $c_{j,j+1}$ by (12)
6: **until** $\big(\hat{c}_{j,j+1}, j \in \mathcal{I}_{|\mathcal{A}|-2}\big) = \big(c_{j,j+1}, j \in \mathcal{I}_{|\mathcal{A}|-2}\big)$
7: **return** $\big(T_i, i \in \mathcal{I}_{|\mathcal{A}|-1}\big)$

Algorithm 4 の 1: で $(c_{j,j+1}, j \in \mathcal{I}_{|\mathcal{A}|-2})$ は $c_{j,j+1} = 0$ と初期化され，初回に実行される NewFill は，Dubé–Haddad 法の Algorithm 3 と同一である．2 回目以降の NewFill は，一つ前の NewFill で求めた分節木を $(\hat{T}_i, i \in \mathcal{I}_{|\mathcal{A}|-1})$ と表記すると，新たに追加した $(c_{j,j+1}, j \in \mathcal{I}_{|\mathcal{A}|-2})$ を (12) で $(\hat{T}_i, i \in \mathcal{I}_{|\mathcal{A}|-1})$ に応じて定めることで，$d = D$ の (11) では，$(\hat{T}_i, i \in \mathcal{I}_{|\mathcal{A}|-1})$ で計算される $\mathtt{E}[L_{\mathrm{AIVF}}(A)]$ に対して，$T_i^{(D)}$ のみを更新することで改善を試みている．

Algorithm 4 の 5: では，NewFill で新たに得られた分節木 $(T_i, i \in \mathcal{I}_{|\mathcal{A}|-1})$ を用いて，$c_{j,j+1}, j \in \mathcal{I}_{|\mathcal{A}|-2}$ を

$$c_{j,j+1} = \frac{1}{\mathrm{Tr}\ (\mathrm{adj}\ R)} \sum_{x=1}^{m-1} \sum_{y=0}^{x-1} (-1)^{(x+y+1)} \cdot \det R_{\{x,y\},\{j,j+1\}} \cdot \big(\mathtt{E}[L_{T_x}(A)] - \mathtt{E}[L_{T_y}(A)]\big) \quad (12)$$

と更新する．上式の $R$ は，単位行列 $I$ と前章の遷移行列 $Q$ を用いて，$R := I - Q$ と定める．$R_{\{x,y\},\{j,j+1\}}$ は，インデックスを 0 から数えて，$R$ の $x$ 行目と $y$ 行目を削除し，$j$ 列目と $j+1$ 列目を削除した $|\mathcal{A}| - 3$ 行 $|\mathcal{A}| - 3$ 列の行列である．また，$\det M$ と $\mathrm{Tr}\ M$ は，それぞれ，正方行列 $M$ の行列式と対角和を意味する．$\mathrm{adj}\ M$ は，$M$ の余因子行列であり，$M \cdot \mathrm{adj}\ M = \det M \cdot I$ を満たす．

例 1 の情報源 $A$ に対して，分節木 $T_0$ と $T_1$ を Algorithm 4 で求めた結果が図 4 であり，表 1 の最終列が

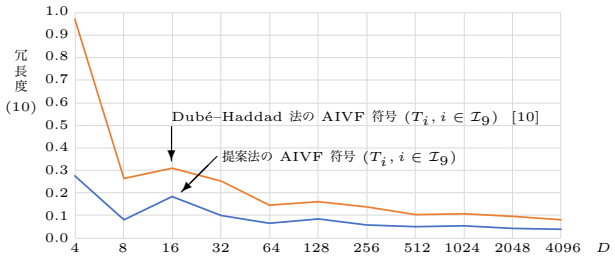



図 5. 例 2 の情報源に対する Dubé–Haddad 法の AIVF 符号と提案する AIVF 符号の冗長度の比較.

得られる．提案した改良により，各分節木の平均分節長 $(\mathrm{E}[L_{T_i}(A)])_{i=0,1}$ は短くなるが，分節木 $T_1$ を用いる確率 $P_S(T_1)$ が上がり，平均分節長 $\mathrm{E}[L_{AIVF}(A)]$ と冗長度の改善が数値的に確認できる．

次に，情報源アルファベットが $|\mathcal{A}| = \{a_i, i \in \mathcal{I}_{10}\}$，生起確率が $p_A(a_i) = (10-i)/55$ である情報源 $A$ の場合を例 2 として，Dubé–Haddad 法と提案法の AIVF 符号 $(T_i, i \in \mathcal{I}_9)$ の冗長度の比較を行う．符号語数 $D = 2^i, i = 2, 3, \ldots, 12$ の数値結果を図 5 のグラフに示す．AIVF 符号では，Tunstall 符号と異なり，2 以上の任意の整数 $D$ に対して符号の構成が可能である．例 2 の情報源に対しても，提案法より冗長度の改善が確認できる．

## 5 まとめ

Dubé, Haddad が提案した AIVF 符号の構成法に対して，Fujita, Iwata, Yamamoto の平均性能最適化の繰り返し法に基づく改良を提案した．提案した AIVF 符号の改良により冗長度が改善できることを計算機実験により確認した． 提案した AIVF 符号は，著者らが知る範囲において，準瞬時分割可能符号のクラスの中で最長の平均分節長を達成している．なお，AIVF 符号で用いる複数の分節木をそれぞれ記述すると，Tunstall 符号に比べて多くのメモリ量を必要とするが，複数の分節木を多重化して記述することでメモリ使用量を削減する Yoshida らの手法 [13, 14] がある．

## 謝辞

本研究は，JSPS 科研費 JP18H01436 の助成を受けた．

## 参考文献

[1] 小林欣吾，森田啓義，**情報理論講義**，培風館，2008 年.

[2] B. Tunstall, *Synthesis of noiseless compression codes*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, 1967.

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

[4] 山本博資, 古賀弘樹, 有村光晴, 岩本貢, **情報理論 –基礎と広がり–**， 共立出版，2012 年.

[5] J. Rissanen, *Information and Complexity in Statistical Modeling*. NY, USA: Springer-Verlag, 2007.

[6] M. Arimura, “On the Average Coding Rate of the Tunstall Code for Stationary and Memoryless Sources,” *IEICE Trans. on Fundam.* vol. E93-A, no. 11, pp. 1904-1911, Nov. 2010.

[7] M. Drmota, W. Szpankowski, ”Redundancy of Lossless Data Compression for Known Sources by Analytic Methods,” *Foundations and Trends in Communications and Information Theory*. NOW Publishers, 2017.

[8] H. Yamamoto and H. Yokoo, “Average-Sense Optimality and Competitive Optimality for Almost Instantaneous VF Codes,” *IEEE Trans. Inf. Theory*, vol. 47, no. 6, pp. 2174–2184, Sep. 2001.

[9] D. Dubé and F. Haddad, “Optimal Single- and Multiple-Tree Almost Instantaneous Variable-to-Fixed Codes,” in *Proc. Data Compress. Conf.* (DCC), Snowbird, UT, USA Mar. 2018, p.405.

[10] D. Dubé and F. Haddad, “Individually Optimal Single- and Multiple-Tree Almost Instantaneous Variable-to-Fixed Codes,” in *Proc. IEEE Int. Symp. Inf. Theory* (ISIT), Vail, CO, USA, Jun. 2018, pp.2192–2196.

[11] R. Fujita, K. Iwata, and H. Yamamoto, “An Iterative Algorithm to Optimize the Average Performance of Markov Chains with Finite States,” in *Proc. IEEE Int. Symp. Inf. Theory* (ISIT), Paris, France, Jul. 2019, pp. 1902–1906.

[12] 藤田 龍星, 岩田 賢一, 山本 博資, “マルコフ連鎖に対する状態遷移確率を用いた定常確率分布の陽な表現公式,” 2017 年情報理論とその応用シンポジウム (SITA2017) 予稿集, pp. 464–469, 2017 年 11–12 月.

[13] S. Yoshida, T. Uemura, T. Kida, T. Asai, and S. Okamoto, “Improving parse trees for efficient variable-to-fixed length codes,” *Journal of Information Processing*, vol. 20, no. 1, pp. 238–249, Jan. 2012.

[14] S. Yoshida and T. Kida, “An efficient Variable-to-Fixed length encoding using multiplexed parse trees,” *Journal of Discrete Algorithms*, vol. 32, pp. 75–86, May 2015.