Asymptotic Optimality of Antidictionary Codes

Takahiro Ota
Dept. of Electronic Engineering
Nagano Prefectural Institute of Technology
Ueda Nagano 386-1211, Japan
Email: ota@pit-nagano.ac.jp

Hiroyoshi Morita
Graduate School of Information Systems
University of Electro-Communications
Chofu Tokyo 182-8585, Japan
Email: morita@is.uec.ac.jp

Abstract—An antidictionary code is a lossless compression algorithm using an antidictionary which is a set of minimal words that do not occur as substrings in an input string. The code was proposed by Crochemore et al. in 2000, and its asymptotic optimality has been proved with respect to only a specific information source, called balanced binary source that is a binary Markov source in which a state transition occurs with probability 1/2 or 1. In this paper, we prove the optimality of both static and dynamic antidictionary codes with respect to a stationary ergodic Markov source on finite alphabet such that a state transition occurs with probability p (0 < p ≤ 1).

I. INTRODUCTION

This paper proves two theorems with respect to asymptotic optimality of both static and dynamic antidictionary codes for stationary ergodic Markov information sources. An antidictionary for a given string is a set of words of minimal length that never appear in the string, and it is in particular useful for data compression. An antidictionary coding scheme, called Data Compression using Antidictionaries (DCA), was first proposed by Crochemore et al. [1] for binary strings. Some extensions of the DCA, which are able to handle a finite alphabet and applied to arithmetic codes, have been proposed [2]–[4] (cf. [5]). Those algorithms work in an off-line manner, while some on-line DCA algorithms using dynamic suffix trees work with linear time and space have been proposed [6]–[8]. Moreover, a memory-efficient DCA using suffix arrays was proposed [9]. It was shown that the algorithm [8] achieves compression ratios as well as an efficient off-line data compression algorithm using Burrows-Wheeler transformation [10] by simulation results.

On the other hand, for only balanced binary sources, asymptotic optimality of a static DCA algorithm has been proved [1]. It was shown that the algorithm is asymptotically optimal for the source generated by an antidictionary if and only if the antidictionary is given to the algorithm in advance [1]. The averaged code length per symbol converges to the entropy rate of the source with probability one. The balanced binary source is a Markov source of finite order and emits all the strings which do not contain any word of the antidictionary as the substrings. Moreover, for any state of the Markov source with only one outgoing edge, probability one is assigned to each edge, while for that with two outgoing edges, probability 1/2 is assigned to those edges.

In this paper, we prove asymptotic optimality of a static and a dynamic DCA for a Markov source constructed from an antidictionary on finite alphabet such that a state transition occurs with probability p (0 < p ≤ 1). This paper is organized as follows. Section II gives the basic definitions and notations. Section III shows review of the DCA algorithms. Section IV proves two theorems with respect to the asymptotic optimality of a static and a dynamic antidictionary code, respectively. Section V summarizes our results.

II. BASIC DEFINITIONS AND NOTATIONS

Let \( X = \{0, 1, \ldots, J - 1\} \) be a finite alphabet and \( X^* \) be the set of all finite strings over \( X \), including the null string of length zero, denoted by \( \lambda \). For \( X \) and \( x \in X^* \), \( |X| \) and \( |x| \) represent the size of \( X \) and the length of \( x \), respectively. For a string \( x = x_1x_2\ldots x_n \in X^* \) of length \( n \), let \( \Sigma(x) \) be the set of all suffixes of \( x \), that is, \( \Sigma(x) = \{x_i; x_{i+1}\ldots x_n|1 \leq i \leq n\} \cup \{\lambda\} \), and let \( D(x) \) be the dictionary of all substrings of \( x \), that is, \( D(x) = \{x_i; x_{i+1}\ldots x_j|1 \leq i \leq j \leq n\} \cup \{\lambda\}. \) Let \( x^i \) be the prefix of length \( i \) of \( x \), and we define that \( x^{\lambda} = \lambda \).

A. Markov Source

Let \( A \subset X^* \setminus \{\lambda\} \) be a non-empty finite set, and we assume that no word \( u \in A \) is a substring of any \( v \in A \) such as \( v \neq u \). Crochemore et al. showed a deterministic automaton \( F(A) \) which accepts all strings that contain no strings of \( A \) as their substrings [11]. In [1], \( F(A) \) is used as an encoder and a decoder of static DCA algorithm. The set \( A \) will be referred to as the antidictionary and a string in \( A \) will be referred to as the Minimal Forbidden Word (MFW). A deterministic automaton \( F(A) = (U, X, s_1, A) \) is defined as follows: Let \( s(w) \) be the state corresponding to string \( w \) in \( F(A) \). In other words, \( s(w) \) is the state reached by string \( w \) from the initial state \( s_1 \).

- The initial state \( s_1 \) is \( s(\lambda) \).
- A state \( s(v) \) for \( v \in A \) is called sink state. Any sink state has \( |X| \) outgoing edges, all having distinct labels, and all the edges of the state terminate the state.
- \( U = \{u|u \text{ is a proper prefix of } v \in A\} \). Note that a proper prefix of \( v = v_1v_2\ldots v_i \) is any of strings \( v_1v_2\ldots v_i \) for \( 1 \leq j < i \), or \( \lambda \). A state \( s(u) \) has \( |X| \) outgoing edges, all having distinct labels. These edges are defined in the following manner: for each \( a \in X \),
  - (i) if \( ua \in U \), then the edge labeled \( a \) from \( s(u) \) terminates at \( s(ua) \).
  - (ii) if \( ua \notin U \), then the edge labeled \( a \) from \( s(u) \) terminates at \( s(w) \), where \( w \) is the longest suffix of \( ua \) such as \( w \in U \cup A \).

Let \( G(A) \) be the automaton obtained by deleting from \( F(A) \) all sink states and all edges incoming sink states. Fig. 1 shows \( G(A) \) and \( F(A) \), where \( A = \{11, 000, 10101\} \) and \( X = \{0, 1\} \). In Fig. 1 the solid lines and circles represent \( G(A) \), while \( G(A) \) with the dotted lines and squares represents \( F(A) \), where squares represent sink states. To avoid trivial cases, we suppose that any state of \( G(A) \) has at least one outgoing edge. For a state \( s \) of \( G(A) \), let \( E(s) \) be the set of labeled symbols of all outgoing edges from \( s \).
Let $S$ be the set of all states of $G(A)$, and let $S_1$ and $S_2$ be the set of all states having only one outgoing edge and that of all states having at least two outgoing edges, respectively. For $G(A)$, let $T: S \times \mathcal{X} \to S$ be transition probabilities independent of time called transition probability matrix. A stationary Markov (or unifilar, cf. [12]) source $X_A$ is characterized by $T$ of $G(A)$, and let $(\mu_1, \mu_2, \ldots, \mu_{|S|})$ be the stationary distribution whose components are the stationary probabilities of their states. We call $X_A$ antidictionary source in this paper.

Moreover, $X_A$ is a source called shift of finite type [13] since $X_A$ is described by a finite set of forbidden strings. Hence, $X_A$ is a stationary ergodic source [13]. A sequence $X^n = X_1 X_2 \cdots X_n$ represents the sequence of random variables of length $n$ on $X_A = \{X_j : j = 1, 2, \ldots \}$. For a state $s_i$ of $G(A)$ in $X_A$, $\rho c_i$ represents the transition probability of the outgoing edge from $s_i$ with label $c$. The entropy $H(X_A)$ is given by

$$H(X_A) = - \sum_{i:s_i \in S_2} |X|-1 \mu_i \sum_{c=0} |X| \log \rho c_i, \quad (1)$$

where $0 \log_2 0 = 0$. Specially, if $X_A$ satisfies that $|X| = 2$, $p_{j0} = p_{j1} = 1/2$ for any $s_j \in S_2$ and $p_{k0} = 1$ or $p_{k1} = 1$ for any $s_k \in S_1$, then $X_A$ is called binary balanced source.

The automaton $G(A)$ has a useful property, called synchronization property [1]. For a state $s_i$, let $l(s_i)$ be the locus string $u$ such that $s_i = s(u)$ and $u \in \mathcal{U}$ are satisfied. Notice that $s(l(s_i)) = s_i$.

Let $u$ and $v$ be the string $l(s_i)$ and $l(s_j)$ for states $s_i$ and $s_j \neq j$, respectively, and let $m$ be the length of the longest MFW in $A$. Then, we have the following theorem.

**Theorem A (Theorem 3 [1]):** For any string $w \in \mathcal{X}^*$ of length $m - 1$, if both strings $uw$ and $vw$ do not contain any string of $A$ as the substrings, then $s(uw) = s(vw)$.

In other words, suppose that $s_d$ and $s_e$ are the states reached by $w$ from $s_1$ and $s_2$, respectively, so that $s_d = s_e$ if the conditions are satisfied shown in Theorem A. In Fig. 1, $m - 1$ is given by $4$ since length of the longest MFW, that is $10101$, is $5$. As an example, for $s_1$, $s_5$ and $w = 0100$, the states reached by $w$ from $s_1$ and $s_5$ are the same state $s_3$.

**B. Suffix Tree**

The suffix tree of $x$ is a tree structure [14] that stores all elements of $\Sigma(x)$. Let $T_x$ be the suffix tree of $x^*$. The string associated with the path from the root $\rho$ to a node $p$ in $T_x$ is denoted by $w(p)$, and we define that $w(\rho) = \lambda$. The string length $|w(p)|$ will be referred to the depth of $p$. For any node $p$ in $T_x$, let $L_i(p)$ be the set of labeled symbols of all edges sprouting from $p$, that is, $L_i(p) = \{a | w(p)a \in D(x^*), a \in \mathcal{X}\}$. For any node $p \neq \rho$, we can write $w(p) = aw$, where $a \in \mathcal{X}$ and $w \in \mathcal{X}^*$. Let $q$ be the node such that $w(q) = w$, and a pointer from $p$ to $q$, denoted by $\sigma(p)$, is called suffix link. For a given depth $d \geq 0$, if $|w(p)| \geq d$, then let $\sigma_j(p)$ be a node of depth $d$ pointed by one of a series of suffix links starting from $p$ and moving back to the root $\rho$.

**Definition 1 (active point):** An active point $c_0$, in $T_x$ is the node corresponding to the string $u$ such that the longest string in $\{\Sigma(x^*) \cap D(x^*-1)\}$ where $c_0$ is the root $\rho$.

The active point plays a key role in the on-line algorithm, called the Ukkonen algorithm, for constructing suffix trees with the linear complexity [15].

**III. REVIEW OF THE DCA ALGORITHMS**

First, we describe a static DCA algorithm [4]. We suppose that Assumption I is satisfied for the static DCA algorithm.

**Assumption I:** The static DCA algorithm knows $A$.

From Assumption I, notice that $G(A)$ plays as the encoder/decoder parts of the algorithm since $G(A)$ is constructed from $A$. Table I shows output for $x_{i+1}$ in the static DCA algorithm. In Case-(1), no symbol is output, that is, $x_{i+1}$ is predictable since there exists only one outgoing edge from $s(x^*)$. In Case-(2), $e(\cdot)$ represents an adaptive arithmetic coder of order-0 (cf. [16]). The probability $\Pr(x_{i+1}|s(x^*))$ is calculated by $N(x_{i+1}|s(x^*))/\sum_{x \in X} N(e(s(x^*)))$, where $N(e(s(x^*)))$ is a counter that has the number of traversed times from $s(x^*)$ with symbol $c$. Note that for $s_k$, if $c \in E(s_k)$, then the initial value of $N(e(s_k))$ is set to 1. Otherwise its initial value is 0.

For a given input string $x$ of length $n$, the codeword of the static DCA algorithm is given by the triplet, that is,

$$\langle A, e(x), n \rangle.$$ 

Next, we describe a dynamic DCA algorithm [8]. The algorithm uses a subtree of the dynamic suffix tree, which has a given fixed depth $d + 1$ (for $d \geq 0$). In [8], a node $\beta_i$ in $T_x$, called modified active point, is used to encode symbol $x_{i+1}$. The node $\beta_i$ is defined as follows:

**Definition 2 (modified active point):** For a given fixed integer $\delta \geq 0$,

$$\beta_i = \begin{cases} \alpha_i, & |w(\alpha_i)| < \delta, \\ \sigma_d(\alpha_i), & |w(\alpha_i)| \geq \delta. \end{cases} \quad (3)$$

Table II shows the output for $x_{i+1}$ in the dynamic DCA algorithm. In Case-(0), the pair $(I, R(x_{i+1}))$ is output, where $I$ represents an interval of insertion of new edge, and $R(x_{i+1})$ represents the rank of $x_{i+1}$ ($1 \leq R(x_{i+1}) \leq |X|$). Let $L_i(\beta_i)$ be a set $\{a | w(\beta_i)a \in D(x^*), a \in \mathcal{X}\}$. Let $R_i$ be a set of the longest string $w(p)c$ in $\{\Sigma(w(\beta_i)c) \cap D(x^*)\}$ for each $c \in (\mathcal{X} \setminus L_i(\beta_i))$. Suppose that $w(p)a, w(q)b \in R_i, a \neq b$. 

![Fig. 1. The automaton $G(A)$ and $F(A)$ for $A = \{11, 000, 10101\}$.](image-url)
If a following condition in (4), (5) and (6) is satisfied, then \( R(a) < R(b) \).

\[
|w(p)|a > |w(q)|b, \\
|w(p)|a = |w(q)|b \quad \text{and} \quad N(a|p) > N(b|q), \\
|w(p)|a = |w(q)|b, \quad N(a|p) = N(b|q) \quad \text{and} \quad a < b \quad (\text{in lexicographical}),
\]

where \( N(\cdot|\cdot) \) is a counter used in Case-(2). The rank \( R(x_{i+1}) \) is determined by traversing up suffix links starting from \( \beta_i \) to \( \rho \) and is the rank of the string which has \( x_{i+1} \) as the last symbol in \( \mathcal{R}_i \). The rank \( R(x_{i+1}) \) is used to convert \( x_{i+1} \) into a smaller integer to improve the compression ratio. The reason is that a symbol \( c \in (\mathcal{X}\setminus \mathcal{L}_i(\beta_i)) \) having high probability will be found at a node near \( \beta_i \) on the suffix links. The details are described in [7].

In Case-(1), no symbol is output since \( x_{i+1} \) is predictable from the fact that there exists only one edge from \( \beta_i \). In Case-(2), the probability \( Pr(x_{i+1}|\beta_i) \) is calculated by \( N(x_{i+1}|\beta_i)/\sum_{c \in \mathcal{X}} N(c|\beta_i) \), where \( N(c|\beta_i) \) is a counter that has the number of traversed times from the internal node \( \beta_j \) with symbol \( c \) \((0 \leq j \leq i - 1)\). Note that for an internal node \( n_k \) of \( \mathcal{T}_i \) such as \( \mathcal{L}_i(n_k) \geq 2 \), where \( c \in \mathcal{L}_i(n_k) \), then the initial value of \( N(c|n_k) \) is set to 1. Otherwise its initial value is 0.

Let \( l_n^a \) be the codeword length per symbol of the static DCA algorithm for a random string of length \( n \). That is, \( l_n^a \) is given by (the codeword length)/\( n \). Then, the following theorem holds.

**Theorem B**: [Theorem 7 [1]] Under Assumption 1, for a balanced binary source \( X_A \), \( l_n^a \) converges to \( H(X_A) \) with probability one as \( n \to \infty \).

IV. MAIN RESULTS

If \( X_A \) is stationary ergodic, then we obtain the following theorem for the static DCA algorithm.

**Theorem 1**: Under Assumption 1, for a stationary ergodic source \( X_A \), \( l_n^a \) converges to \( H(X_A) \) with probability one as \( n \to \infty \).

Now, let \( l_n^b \) be the codeword length per symbol of the dynamic DCA algorithm for a random string of length \( n \). And let \( m \) be the length of the longest MFW in \( A \). Moreover, we have the following assumption on the dynamic DCA algorithm.

**Assumption 2**: Both encoder and decoder of the dynamic DCA algorithm do not know \( A \) while they know \( m \).

**Theorem 2**: Under Assumption 2, for a stationary ergodic source \( X_A \), \( l_n^b \) converges to \( H(X_A) \) with probability one as \( n \to \infty \).

**A. Proof of Theorem 2**

We use three lemmas to prove Theorem 1. Let \( S_{2,0} \) and \( S_{2,\infty} \) be the set of states in \( S_2 \) for \( \mu_i = 0 \) and \( \mu_i > 0 \), respectively. For \( X^n \), let \( Y_{i,n} \) be a random variable taking values in the number of traversed times of \( s_i \), and let \( \epsilon(X^n) \) be a random variable taking values in the length of output of Case-(2), that is \( \epsilon(x) \) in (4). For a given symbol \( c \in \mathcal{X} \) and \( s_i \in S_{2,\infty} \), let \( Z_{ic} \) be a random variable, when \( s_i \) is traversed at the \( \text{ith} \) time, such as

\[
Z_{ic} = \begin{cases} 1 & (z = c), \\ 0 & (z \neq c), \end{cases}
\]

where \( z \) is the labeled symbol of traversed outgoing edge from \( s_i \) at the time. For a positive integer \( k \), \( |Z_{ic}|_k \) is given by \( |Z_{ic}|_k = (|Z_{ic}| + |Z_{ic}|_2 + \ldots + |Z_{ic}|_k)/k \).

**Lemma 1**: \( |Z_{ic}|_k \) is i.i.d. and \( Z_{ic} \) is i.i.d. and \( Z_{ic} = (h = 1,2, \ldots) \) has the same probability distribution. And, from the definition of \( Z_{ic} \), the expected value \( E(Z_{ic}) \) equals to \( p_{ic} \). Moreover, for \( s_i \in S_{2,\infty} \), from Lemma 1 \( Y_{i,n} \) converges to infinity as \( n \to \infty \) with prob. 1. Therefore, from the strong law of large numbers, the lemma holds.

**Proof of Lemma 2**: A sequence \( Z_{ic} = |Z_{ic}|_k \) is i.i.d. and \( Z_{ic} \) is i.i.d. and \( Z_{ic} = (h = 1,2, \ldots) \) has the same probability distribution. And, from the definition of \( Z_{ic} \), the expected value \( E(Z_{ic}) \) equals to \( p_{ic} \). Moreover, for \( s_i \in S_{2,\infty} \), from Lemma 1 \( Y_{i,n} \) converges to infinity as \( n \to \infty \) with prob. 1. Therefore, from the strong law of large numbers, the lemma holds.

**Proof of Lemma 3**: \( \lim_{n \to \infty} \frac{|e(X^n)|}{n} = \lim_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n-1} |X_i| \quad (14) \]

**Proof of Theorem 7**: From (12), \( l_n^a \) is given by

\[
\frac{\#A}{n} + \frac{|e(X^n)|}{n} + \frac{|\omega^*(n)|}{n},
\]

where \( \#A \) is a size of list of all the MFWs in \( A \), and \( \omega^*(n) \) is a representation of \( n \) using the Elias \( \omega^* \) code for positive integers [17] (cf. [12]). The length \( |\omega^*(n)| \) is given by

\[
|\omega^*(n)| \leq \log_2 n + 2 \log_2 (\log_2 n) + 7.
\]
From (14), the third term of the right-hand side of (13) converges to 0 as \( n \to \infty \). From Assumption \( \#A \) is a constant, so that the first term also converges to 0 as \( n \to \infty \). Therefore, from Lemma 12 the theorem holds with prob. 1.

B. Proof of Theorem \( \star \)

We use eight lemmas to prove Theorem \( \star \). For a given fixed integer \( m \geq 1 \) in Assumption 2 we use \( m - 1 \) as the depth \( d \) in Definition 2 that is

\[ d = m - 1. \]  

(15)

We define a random variable \( V_k \) for \( k \geq 1 \). For \( V_k \), a random variable \( Q_k \) is defined as

\[ Q_k = \left\{ \begin{array}{ll}
0 & (\exists \epsilon \in \epsilon: V_k = V_i, (1 \leq i \leq k - 1)),
1 & (V_k \neq V_i, (1 \leq \epsilon \leq k - 1)).
\end{array} \right. \]  

(16)

where \( \epsilon \) is a string satisfying that \( \Pr \{ V_1 = \epsilon \} > 0 \). Note that we define that \( Q_k \) takes value 1. For a string \( x^n \) on \( \mathcal{X} \), let \( \Delta_n \) be the set of all nodes whose depth is \( d \) in \( T_n \), and for any state \( s_j \) \( (1 \leq j \leq |S|) \) of \( G(A) \), we define that \( \Delta_{j,n} = \{ p | s_j = s(w(p)), p \in \Delta_n \}. \) Note that for a node \( p \in \Delta_n \), the unique state of \( G(A) \) is determined from Theorem 3 since \( |w(p)| = d \) and \( m = d - 1 \).

For a node \( p \), let \( N_n(p) \) be the random number of times \( \beta_h \) passed \( p \) \((0 \leq h \leq n - 1)\). For a given symbol \( c \in \mathcal{X} \) and \( p \in \Delta_{j,n} \), let \( Z_{j,c,k} \) be a random variable, when \( p \) is traversed at the \( k \)-th time, such as

\[ Z_{j,c,k} = \left\{ \begin{array}{ll}
1 & (z = c),
0 & (\neq c),
\end{array} \right. \]  

(17)

where \( z \) is the labeled symbol of traversed edge from \( p \) at the time. For a positive integer \( g \), \( Z_{j,c,g} \) is given by \( Z_{j,c,g} = (Z_{j,c,1} + Z_{j,c,2} + \cdots + Z_{j,c,g}) \). Let \( D_n \) be a random variable taking the depth of \( \beta_h \) that is \( |w(\beta_h)| \), and let \( E_n \) be a random variable taking the index of \( s(w(\beta_h)) \).

Lemma 4: If \( x_{n-d+1}x_{n-d+2} \cdots x_n \in D(x^{n-1}) \), then \( |w(\beta_n)| = d \).

Lemma 5: If \( \beta_n \in \Delta_n \), then \( s(w(\beta_n)) = s(x^n) \).

Lemma 6: \( \Pr \{ \lim_{n \to \infty} Q_n = 0 \} = 1 \).

Lemma 7: \( \lim_{n \to \infty} D_n = d \).

Lemma 8: \( \lim_{n \to \infty} E_n = s(x^n) \).

Lemma 9: \( \lim_{n \to \infty} \sum_{p \in \Delta_{j,n}} N_n(p)/n = \mu_j \).

Lemma 10: For \( p \in \Delta_{j,n} \), \( \lim_{n \to \infty} \sum_{p \in \Delta_{j,n}} N_n(p)/n = \mu_j \).

Lemma 11: For \( p \in \Delta_{j,n} \), \( \Pr \{ \lim_{n \to \infty} [Z_{j,c,n} + N_{n}(p)] = p_{j,c} \} = 1 \).

(Proof of Lemma 11) Since \( v = x_{n-d+1}x_{n-d+2} \cdots x_n \in \Sigma(x^n) \), we have \( v \in \Sigma(x^n) \cap D(x^{n-1}) \). From Definition 1 we obtain \( |w(\alpha_n)| \geq |v| = d \). Therefore, we have \( |w(\beta_n)| = d \).

(Proof of Lemma 12) Since \( \beta_n \in \Delta_n \), we have \( w(\beta_n) = w = x_{n-d+1}x_{n-d+2} \cdots x_n \) and \( |w| \geq |v| = d \). From Theorem 3 and \( s(w) = s(x^{n-d}) \), we have \( s(w(\beta_n)) = s(x^n) \).

(Proof of Lemma 13) Since \( X_A \) is a stationary ergodic source, the lemma holds (cf. [18]).

(Proof of Lemma 14) Since \( d \) is a constant, \( \Pr \{ \lim_{n \to \infty} Q_{n-d+1} = 0 \} = 1 \) from Lemma 6. Therefore, there exists \( j \) \( (1 \leq j \leq n - d + 1) \) such that \( x_{n-d+1}x_{n-d+2} \cdots x_n = x_jx_{j+1} \cdots x_{j+d-1} \) with probability 1. Hence from Lemma 4 the lemma holds.
Moreover, the maximum length of \( I \) is \( n \). Hence, by using Elias \( \omega^* \) code, we obtain
\[
I_0 \leq \omega^*(n). \tag{22}
\]
By using a fixed length code for a symbol with respect to \( R(x_{i+1}) \),
\[
R_0 = \log_2 |\mathcal{X}|. \tag{23}
\]
From (21), (22), and (23),
\[
C_0(x^n)/n \leq n_0 \cdot (I_0 + R_0)/n \leq \frac{(|\mathcal{X}|^2 - 1) \cdot (\omega^*(n) + \log_2 |\mathcal{X}|)}{(|\mathcal{X}| - 1) \cdot n}. \tag{25}
\]
Since \( |\mathcal{X}| \) and \( d \) are constants, from (14), \( C_0(x^n)/n \) converges to 0 as \( n \to \infty \). Therefore,
\[
\lim_{n \to \infty} C_0(x^n)/n = 0. \tag{26}
\]
Note that in case of \( |\mathcal{X}| = 1 \), since \( n_0 \leq d + 1 \) and \( I_0 = R_0 = 1 \), equation (26) holds.
Next, we evaluate \( C_2(X^n) \). For a given \( x^n \), let \( l(p) \) be the averaged code length of Case-(2) for a node \( p \) in \( \mathbb{T}_n \). Note that \( l(p) < \infty \) since \( |\mathcal{X}| \) is finite. We have
\[
\lim_{n \to \infty} \frac{C_2(x^n)}{n} \leq \limsup_{n \to \infty} \frac{1}{n} \sum_{|\mathcal{L}_n(p)| \geq 2} N_n(p)l(p) \tag{27}
\]
\[
= \limsup_{n \to \infty} \frac{1}{n} \sum_{|\mathcal{L}_n(p)| \geq 2, p \notin \Delta_n} N_n(p)l(p) + \limsup_{n \to \infty} \frac{1}{n} \sum_{|\mathcal{L}_n(p)| \geq 2, p \in \Delta_n} N_n(p)l(p). \tag{28}
\]
For \( p \notin \Delta_n \), the maximum value of \( N_n(p) \) is less than or equal to the total number \( M \) of strings whose lengths are not more than \( d \) in \( \mathcal{X}^n \), that is \( M \) described in the proof of Lemma 9. Therefore, the first term in the right-hand side of (28) converges to 0 as \( n \to \infty \) since \( M \) is a constant. Let \( \varepsilon_n \) be the first term. From (29), (30),
\[
\lim_{n \to \infty} \frac{C_2(x^n)}{n} \leq \varepsilon_n + \limsup_{n \to \infty} \frac{1}{n} \sum_{j,s \in S_{2,0}, p \in \Delta_n} N_n(p)l(p) + \sum_{j,s \in S_{2,0}} \limsup_{n \to \infty} \frac{1}{n} \sum_{p \in \Delta_n} N_n(p)l(p). \tag{29}
\]
where (a) follows from the fact that an index \( j \) of state of \( G(A) \) is independent of \( n \). From (29) the first term of right-hand side of (29) converges to 0 since \( \mu_j = 0 \) for \( s_j \in S_{2,0} \) as \( n \to \infty \). Let \( \varepsilon_n \) be the first term and let \( \varepsilon_n = \varepsilon_n + \varepsilon_n \). From (29),
\[
\lim_{n \to \infty} \frac{C_2(x^n)}{n} \leq \varepsilon_n + \sum_{j,s \in S_{2,0}} \limsup_{n \to \infty} \frac{1}{n} \sum_{p \in \Delta_n} N_n(p)l(p). \tag{30}
\]
For \( p \in \Delta_j,n \), \( l(p) \) is written by
\[
l(p) = -\sum_{c=0}^{|\mathcal{X}|-1} [\tilde{Z}_{jc}](N_n(p)) \log_2 [\tilde{Z}_{jc}](N_n(p)). \tag{31}
\]
Moreover, for \( p \in \Delta_j,n \), from Lemma 11 and (31),
\[
l(p) = -\sum_{c=0}^{|\mathcal{X}|-1} \mu_{jc} \log_2 \mu_{jc} \tag{32}
\]
with prob. 1 as \( n \to \infty \). From (30), (32), and Lemma 9,
\[
\lim_{n \to \infty} \frac{C_2(x^n)}{n} \leq \varepsilon_n + \sum_{j,s \in S_{2,0}} \limsup_{n \to \infty} \frac{1}{n} \sum_{p \in \Delta_n} N_n(p)l(p). \tag{33}
\]
with prob. 1. From (33) and (11),
\[
\lim_{n \to \infty} \frac{C_2(x^n)}{n} \leq \varepsilon_n + H(X_A). \tag{34}
\]
with prob. 1. From (19), (20), (26), and (34), and since \( \varepsilon_n \) converges to 0 with prob. 1 as \( n \to \infty \), we obtain
\[
l_n = H(X_A) \tag{35}
\]
with prob. 1 as \( n \to \infty \). Therefore, the theorem holds.

V. Conclusion

In this paper, we proved asymptotic optimality of both static and dynamic DCA algorithms with respect to antidictionary sources, that is a stationary ergodic Markov source driven by \( G(A) \). The averaged code length per symbol of the algorithms converge to the entropy rate of the source with probability one.

References

[1] M. Crochemore, F. Mignosi, A. Restivo, and S. Salemi, “Data compression using antidictionaries,” Proc. IEEE, vol. 88, no. 11, pp. 1756–1768, Nov. 2000.

[2] M. Crochemore, C. Epifanio, R. Grossi, and F. Mignosi, “A Trie-Based Approach for Compacting Automata,” Proc. of Combinatorial Pattern Matching 15th Annual Symp. (CPM’04), vol. 3109, pp.145–158, July 2004.

[3] N. Ohkawa, K. Harada, and H. Yamamoto, “Data compression by arithmetic coding and source modeling based on the anti-dictionary method,” IEICE Technical Report, IT2005-49, July 2005 (in Japanese).

[4] T. Ota, “Antidictionary data compression using dynamic suffix trees,” PhD thesis, University of Electro-Communications, 2007.

[5] D. Salomon and G. Motta, “Handbook of data compression,” Springer, 2009.

[6] T. Ota and H. Morita, “On the on-line arithmetic coding based on antidictionaries with linear complexity,” Proc. of 2007 IEEE Int’l Symp. on Information Theory (ISIT2007), pp.86–90, June 2007.

[7] T. Han and K. Kobayashi, “Mathematics of information coding and source modeling using dynamic suffix trees,” Proc. of 2008 Int’l Symp. on Information Theory and its Applications (ISITA2008), pp.1105–1110, Dec. 2008.

[8] T. Han and K. Kobayashi, “Adaptive antidictionary data compression using a set of restricted length of minimal forbidden words,” IEICE Technical Report, IT2008-99, Mar. 2009 (in Japanese).

[9] M. Fiala and J. Holub, “DCA using suffix arrays,” Proc. of Data Compression Conference (DCC2008), pp. 516, Mar. 2008.

[10] M. Burrows and D.J. Wheeler, “A block-sorting lossless data compression algorithm,” SRC Research Report, May 1994.

[11] M. Crochemore, F. Mignosi, and A. Restivo, “Antimata and forbidden words,” Inform. Proc. Lett., vol. 67, pp. 111-117, Aug. 1998.

[12] D. Gutfeld, “Algorithms on strings, trees, and sequences,” Cambridge University Press, 1997.

[13] D. Lind and B. Marcus, “Symbolic dynamics and coding,” Cambridge University Press, 1995.

[14] J. Ukkonen, “On-line construction of suffix-trees,” Algorithmica, vol. 14, pp. 249–260, 1995.

[15] P. Elias, “Universal codeword sets and representations of the integers,” IEEE Trans. Inform. Theory, vol.IT-21, no.2, pp.194–203, 1975.

[16] P. Billingsley, “Probability and Measure 3rd Ed., John Wiley & Sons Inc., 1995.

[17] S. Kullback, “Information Theory and Statistics,” Dover Publications, 1997.

[18] A. Moffat and T. Turpin, “Compression and coding algorithms,” Kluwer Academic Publishers, 2002.

[19] P. Elias, “Universal codeword sets and representations of the integers,” IEEE Trans. Inform. Theory, vol.IT-21, no.2, pp.194–203, 1975.

[20] P. Billingsley, “Probability and Measure 3rd Ed., John Wiley & Sons Inc., 1995.

[21] R. Sedgewick and P. Flajolet, “Analysis of algorithms,” Addiwen-Wesley, 1996.