# Universal Lossless Source Coding With the Burrows Wheeler Transform

Michelle Effros, *Member, IEEE*, Karthik Visweswariah, *Member, IEEE*, Sanjeev R. Kulkarni, *Senior Member, IEEE*, and Sergio Verdú, *Fellow, IEEE*

***Abstract*—The Burrows Wheeler Transform (BWT) is a reversible sequence transformation used in a variety of practical lossless source-coding algorithms. In each, the BWT is followed by a lossless source code that attempts to exploit the natural ordering of the BWT coefficients. BWT-based compression schemes are widely touted as low-complexity algorithms giving lossless coding rates better than those of the Ziv–Lempel codes (commonly known as LZ'77 and LZ'78) and almost as good as those achieved by prediction by partial matching (PPM) algorithms. To date, the coding performance claims have been made primarily on the basis of experimental results. This work gives a theoretical evaluation of BWT-based coding. The main results of this theoretical evaluation include: 1) statistical characterizations of the BWT output on both finite strings and sequences of length $n \to \infty$, 2) a variety of very simple new techniques for BWT-based lossless source coding, and 3) proofs of the universality and bounds on the rates of convergence of both new and existing BWT-based codes for finite-memory and stationary ergodic sources. The end result is a theoretical justification and validation of the experimentally derived conclusions: BWT-based lossless source codes achieve universal lossless coding performance that converges to the optimal coding performance more quickly than the rate of convergence observed in Ziv–Lempel style codes and, for some BWT-based codes, within a constant factor of the optimal rate of convergence for finite-memory sources.**

***Index Terms*—Burrows Wheeler Transform (BWT), rate of convergence, redundancy, text compression, universal noiseless source coding.**

## I. Introduction

THE Burrows Wheeler Transform (BWT) [1] is a slightly expansive reversible sequence transformation currently receiving considerable attention from researchers interested in practical lossless data compression algorithms (e.g., [2]–[7]). To date, the majority of research devoted to BWT-based compression algorithms has focused on experimental comparisons of BWT-based algorithms with competing codes. Experimental results on algorithms using this transformation (e.g., [2], [3], [5]) indicate lossless coding rates better than those achieved by Ziv–Lempel-style codes (LZ'77 [8], LZ'78 [9], and their descendants) but typically not quite as good as those achieved by the prediction by partial mapping (PPM) schemes described in works like [10], [11], [2]. BWT code implementation yields complexity comparable to that of the Ziv–Lempel codes, which are significantly faster than algorithms like PPM [1], [2].

Early theoretical investigations of BWT-based algorithms include the work of Sadakane, Ariumura and Yamamoto, and Effros. In [12], [13], Sadakane considers the performance of source codes based on a variant of the BWT described in [14] and states that codes based on block sorting are asymptotically optimal for finite-order Markov sources if the permutation of all symbols sharing a common context is random. Sadakane notes, however, that "the permutation in the BWT is not completely random" but conjectures that the proposed algorithms work for BWT-transformed data sequences. In [15]–[17], Arimura and Yamamoto present a sequence of information-theoretic results on BWT-based source coding, demonstrating the universality of BWT-based codes for finite memory and stationary totally ergodic sources. In [18], Effros gives an information-theoretic analysis of both the traditional BWT-based codes considered by previous authors and a collection of new BWT-based codes introduced in that work. The analysis demonstrates the universality of each of the BWT-based codes considered and gives the first rate of convergence bounds for BWT-based universal codes.

This paper combines the aforementioned results by Effros with the asymptotic analyses of convergence rate and output statistics derived by Visweswariah, Kulkarni, and Verdú [19], [20] and a nonasymptotic analysis of the BWT output statistics by Effros. The key results are: statistical characterizations of the BWT output for both finite strings and sequences of length $n \to \infty$, a proof of the universality and bound on the rate of convergence of a minor variation on existing BWT-based codes for finite-memory sources, proofs of the universality of the family of algorithms introduced in [18] on both stationary finite-memory sources and more general stationary ergodic sources, rate of convergence bounds for the same codes and sources, and a comparison of BWT-based codes to each other and to other universal coding algorithms. The comparison confirms and quantifies the experimentally observed results.

Manuscript received July 16, 1999; revised December 27, 2001. This paper is based on work at the California Institute of Technology supported in part by the National Science Foundation under CAREER Grant MIP-9501977, a grant from the Powell Foundation, and donations through the Intel 2000 Technology for Education Program. Work at Princeton University was supported in part by ODDR&E MURI through the Army Research Office under Grant DAAD19-00-1-0466, by the National Science Foundation under Grant NCR-9523805, and by the National Science Foundation KDI under Contract ECS-9873451.

M. Effros is with the Department of Electrical Engineering (MC 136-93), California Institute of Technology, Pasadena, CA 91125 USA (e-mail: effros@caltech.edu).

K. Visweswariah was with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA. He is now with IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: kv1@us.ibm.com).

S. R. Kulkarni and S. Verdú are with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: kulkarni@ee.princeton, edu; verdu@ee.princeton.edu).

Communicated by M. Weinberger, Associate Editor for Source Coding.

Publisher Item Identifier S 0018-9448(02)02800-6.

On sequences of length $n$ drawn from a finite-memory source, the performance of the best BWT-based codes converges to the optimal performance at a rate of $O(\log n/n)$, surpassing the $O(\log\log n/\log n)$ convergence of LZ'77 [21] and the $O(1/\log n)$ convergence of LZ'78 [22], [23] and the variation of LZ'77 given in [21]. This $O(\log n/n)$ convergence comes within a constant factor of the optimal rate of convergence for finite-memory sources. Note that many of the codes considered here use sequential codes on the BWT output but that the overall data compression algorithms are nonsequential since the transform itself requires simultaneous access to all $n$ symbols of a data string.

The paper is organized as follows. Section II contains a variety of background material, including an introduction to universal source coding, a description of the class of stationary finite-memory sources, and a brief summary of previous universal coding results for these sources. Section III contains a description of the BWT and a discussion of its algorithmic complexity and memory use. Section IV considers BWT-reordered data sequences for stationary finite-memory sources, focusing on those properties needed for efficient coding of the transform output. The description of the BWT output highlights a key characteristic of this transform: the BWT of a reversed data string groups together all symbols that follow the same context. This property leads both to the lossless coding strategies used in the BWT-based codes discussed in Section V and to the asymptotic analysis of the statistical properties of the BWT output given in Section VI. Section V describes the family of BWT-based codes and proves the universality and rate of convergence of each for both finite-memory sources and stationary ergodic sources. The rate of convergence results on finite-memory sources range from $O(\sqrt{\log n/n})$ for the codes requiring the least memory and computation to $O(\log n/n)$ for a slightly more complex BWT-based algorithm or a BWT-based code in which the encoder uses *a priori* information about the source memory. Thus, even the simplest new BWT-based code gives a rate of convergence faster than that of either of the Ziv–Lempel algorithms, while the BWT code with the fastest rate of convergence achieves—to within a constant factor—the optimal rate of convergence. Section VI treats the question of statistical characterization of the BWT output considered in [19], demonstrating the convergence to zero of the normalized Kullback–Leibler distance between the BWT output distribution and a piecewise independent and identically distributed (p.i.i.d.) source distribution. A summary of results and conclusions—including a comparison of the performance, complexity, and memory use of BWT-based algorithms and Ziv–Lempel codes—follows in Section VII.

## II. Background and Definitions

A universal lossless source code is a sequence of source codes that asymptotically achieves the optimal performance for every source in some broad class of possible sources. Making this notion more precise requires some definitions.

Consider any class $\{P_\theta\colon \theta\in\Lambda\}$ of stationary ergodic sources on finite source alphabet $\mathcal{X}$. For each $\theta\in\Lambda$, let $H_\theta(X^n)$ and $H_\theta(\mathcal{X})$ be the $n$th-order entropy and entropy rate, respectively, of $P_\theta$. Thus,

$$H_\theta(X^n)=\sum_{x^n\in\mathcal{X}^n}[-P_\theta(x^n)\log P_\theta(x^n)]$$

and

$$H_\theta(\mathcal{X})=\lim_{n\to\infty}\frac{1}{n}H_\theta(X^n)$$

for each $\theta\in\Lambda$. Given any variable-rate lossless source coding strategy for coding $n$-sequences from $\mathcal{X}$, for each $x^n=(x_1,\ldots,x_n)\in\mathcal{X}^n$, let $\ell_n(x^n)$ be the description length used in the lossless description of $x^n$ with the chosen coding strategy. For each $\theta\in\Lambda$, $\delta_n(\theta)$ describes the resulting expected redundancy in coding samples from distribution $P_\theta$. That is, $\delta_n(\theta)$ is the difference between the expected rate per symbol $E_\theta\ell_n(X^n)/n$ using the given blocklength-$n$ code and the optimal rate per symbol $H_\theta(X^n)/n$ for coding $n$-vectors from $P_\theta$; thus,

$$\delta_n(\theta)=\frac{1}{n}E_\theta\ell_n(X^n)-\frac{1}{n}H_\theta(X^n).$$

A sequence of coding strategies, here referred to by their redundancy functions $\{\delta_n(\cdot)\}_{n=1}^{\infty}$, is a *weakly minimax universal lossless source code* on $\Lambda$ if $\delta_n(\theta)\to 0$ for each $\theta\in\Lambda$ and a *strongly minimax universal lossless source code* on $\Lambda$ if that convergence is uniform in $\theta$ [24]. This work focuses primarily on minimax universal lossless source coding. The redundancy results derived in this work are, however, all achieved by first finding deterministic bounds on the source coding rate. These deterministic bounds characterize the code performance on sequence $X^n$ in terms of the "empirical entropy" of $X^n$ relative to a distribution model approximating the true underlying source statistics. The result is a stronger characterization of the code performance than that given by the expected redundancy alone.

In [24], Davisson describes a minimax universal lossless code on the class of stationary, ergodic sources using a construction due to Fitingof. Davisson's argument demonstrates the existence of minimax universal lossless source codes and establishes the rate of convergence of $\delta_n(\theta)$ to zero as a second-order measure of performance for minimax universal lossless source codes. Rissanen and others extend Davisson's results for finitely parameterized sources and quantify the condition of second-order optimality in universal lossless source coding [25]–[29]. For any class $\Lambda$ of sources smoothly parameterized by $K$ real numbers, the optimal rate of convergence of $(K/2)\log n/n$ is proven achievable to within $O(1/n)$ for almost all $\theta\in\Lambda$ [27], [28].

This work focuses first on the problem of minimax universal lossless source coding for stationary finite-memory sources. A review of the class of unifilar, ergodic, finite-state-machine (FSM) sources is useful to the discussion that follows. An FSM source is defined by a finite alphabet $\mathcal{X}$, a finite set of states $\mathcal{S}$, $|\mathcal{S}|$ conditional probability measures $\{p(\cdot|s)\}_{s\in\mathcal{S}}$, and a next-state function $f\colon\mathcal{S}\times\mathcal{X}\to\mathcal{S}$. Given an FSM data source and an initial state $s_0$, the conditional probability of string $x^n=x_1,\ldots,x_n\in\mathcal{X}^n$ given $s_0$ is defined as

$$\Pr(x^n|s_0)=\prod_{i=1}^{n}p(x_i|s_{i-1})$$

where $s_i=f(s_{i-1},x_i)$ for all $1\le i\le n$.

The class of FSMX sources [30], also called finite-order FSM sources, is the subset of the class of FSM sources for which there exists an integer $M$ such that for every $i \geq M$, the $M$ most recent symbols $x_{i-M+1}^{i}$ uniquely determine the state $s_i$ at time $i$. For FSMX sources, the set $\mathcal{S}$ is defined by a minimum suffix set of strings from $\mathcal{X}^*$ with the property that for every $s \in \mathcal{S}$ and every $x \in \mathcal{X}$ such that $p(x|s) \neq 0$, the string $sx$ has exactly one suffix in $\mathcal{S}$. Thus, for any FSMX source,

$$f(s_{i-1}, x_i) = \text{suf}\,(s_{i-1}x_i), \qquad \text{for all } i$$

where $\text{suf}\,(sx)$ denotes the suffix of the string achieved by concatenating symbol $x$ to the end of string $s$.

FSMX sources inherit from FSM sources the condition that the current state is a function only of the current source symbol and the previous state ($s_i = f(s_{i-1}, x_i)$ for all $i$). This condition is both restrictive [31] and unnecessary for this work. As a result, the restriction is dropped, yielding a class of generalized FSMX sources, here called finite-memory sources after [32]. For any finite-memory source, there exists a minimum suffix set $\mathcal{S}$ of strings from $\mathcal{X}^*$ and an integer $M$ such that

$$\Pr\left(x^n|x_{-(M-1)}^0\right) = \prod_{i=1}^{n} p(x_i|s_{i-1})$$

and

$$s_{i-1} = \text{suf}\,(x_{i-M}, x_{i-(M-1)}, \ldots, x_{i-1}), \qquad \text{for all } i.$$

The state variables $\{s_i\}$ are variable-length strings describing the finite "context" of previous symbols on which the current symbol's distribution depends. For stationarity, the symbols $X_{-M+1}, X_{-M+2}, \ldots, X_0$ should be drawn from the stationary distribution on $\mathcal{X}^M$ induced by the finite-memory source model, giving

$$\Pr(x^n) = p(x^M) \prod_{i=M+1}^{n} p(x_i|s_{i-1})$$

where $p(x^M)$ is the stationary distribution on $\mathcal{X}^M$ induced by the given finite-memory source.

The class of finite-memory sources discussed here is more restrictive than the class introduced in [32]. Like the finite-memory sources described here, the finite-memory sources of [32] describe the probability of the next symbol using a conditional distribution that depends on no more than some maximal number of previously coded symbols. Unlike the finite-memory sources described here, the finite-memory sources of [32] do not require all contexts of length $k$ to comprise exactly the previous $k$ symbols in the data string. The variable and noncontiguous contexts of [32] create considerable difficulties for BWT-based algorithms, and are therefore excluded. Thus, the class of finite-memory sources described here is a subset of the earlier defined class. Notice, though, that any source meeting the broader definition for finite-memory sources but not requiring context variation across symbols may also be modeled within the definition of finite-memory sources considered here, with the caveat that the resulting contiguous model might require more states than its predecessor. This increase in $|\mathcal{S}|$ results from the fact that prior symbols cannot be rearranged and $|\mathcal{S}|$ is affected by the length of the history used in the conditional distributions. This increase in $|\mathcal{S}|$ may cause significant performance degradation, since the rate of convergence results described in Section V grow with $|\mathcal{S}|$.

In [28], Rissanen considers universal source coding for binary FSM sources when the number $|\mathcal{S}|$ of states is unknown. In that work, he demonstrates the existence of universal source codes for which $\delta_n(\theta)$ approaches zero as $(|\mathcal{S}|/2)\log n/n + O(1/n)$ for almost all $\theta$ and demonstrates the optimality (to within $O(1/n)$) of the achieved rate of convergence when the given model is the most efficient model for the chosen source. In this case, $\theta = (p(1|s)\colon\ s \in \mathcal{S})$ describes the distribution $P_\theta$, and thus, $K = |\mathcal{S}|$ and $\Lambda \subseteq \mathbb{R}^K$, giving the familiar $(K/2)\log n/n + O(1/n)$. For more general finite alphabets, $K = |S|(|\mathcal{X}| - 1)$ gives the number of parameters needed to describe the conditional probabilities $p(x|s)$ for all but one value of $x \in \mathcal{X}$ and all values of $s \in \mathcal{S}$. The optimal algorithm traverses the entire data sequence to determine the optimal estimate of $\mathcal{S}$ and then describes the data sequence using the chosen estimate. In the same work, Rissanen conjectures the optimality of a related sequential algorithm for estimating $\mathcal{S}$ during the encoding procedure rather than in a separate pass through the entire data sequence prior to coding. A flaw in that algorithm is pointed out in [31] by Weinberger, Lempel, and Ziv, who also present an alternative to Rissanen's algorithm for universal source coding of FSMX sources with *known* memory constraint $M$. The algorithm computes and sequentially updates an on-line estimate of $\mathcal{S}$ during the coding process. The resulting code asymptotically achieves Rissanen's optimal $(K/2)\log n/n$ rate of convergence for FSM sources using a sequential coding strategy. When $n$ is known, this strategy reduces the maximal coding delay from $n$ in Rissanen's code to $O(\log n)$. The number of arithmetic operations used grows linearly with both $M$ and $n$. The same results apply to finite-memory sources.

## III. The BWT

The BWT [1] is a reversible block-sorting transform that operates on a sequence of $n$ data symbols to produce a permuted data sequence of the same symbols and a single integer in $\{1, \ldots, n\}$. Let

$$\text{BWT}_n\colon \mathcal{X}^n \to \mathcal{X}^n \times \{1, \ldots, n\}$$

denote the $n$-dimensional BWT function and

$$\text{BWT}_n^{(-1)}\colon \mathcal{X}^n \times \{1, \ldots, n\} \to \mathcal{X}^n$$

denote the inverse of $\text{BWT}_n$. Since the sequence length $n$ is evident from the source argument, the functional transcript is typically dropped, giving

$$(y^n, u) = \text{BWT}\,(x^n) \quad \text{and} \quad \text{BWT}^{(-1)}(y^n, u) = x^n.$$

The notations $\text{BWT}_{\mathcal{X}}$ and $\text{BWT}_{\mathbb{N}}$ denote the character and integer portions of the BWT, respectively.

The forward BWT proceeds by forming all $n$ cyclic shifts of the original data string and sorting those cyclic shifts lexicographically. The BWT output has two parts. The first part is a length-$n$ string giving the last character of each of the (lexicographically ordered) cyclic shifts. The second part is an integer

Step 1: Find each cyclic shift of the given data sequence.
Step 2: Order the cyclic shifts (rows) lexicographically.
Step 3: Output the last column of the given array and the row index of the original data sequence.

| | Step 1 | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | **b** | **a** | **n** | **a** | **n** | **a** | **s** |
| 2 | s | b | a | n | a | n | a |
| 3 | a | s | b | a | n | a | n |
| 4 | n | a | s | b | a | n | a |
| 5 | a | n | a | s | b | a | n |
| 6 | n | a | n | a | s | b | a |
| 7 | a | n | a | n | a | s | b |

| | Step 2 | | | | | | | Step 3 |
|---|---|---|---|---|---|---|---|---|
| 1 | a | n | a | n | a | s | b | b |
| 2 | a | n | a | s | b | a | n | n |
| 3 | a | s | b | a | n | a | n | n |
| 4 | **b** | **a** | **n** | **a** | **n** | **a** | **s** | s |
| 5 | n | a | n | a | s | b | a | a |
| 6 | n | a | s | b | a | n | a | a |
| 7 | s | b | a | n | a | n | a | a |

Fig. 1. The BWT of the sequence "bananas." The original data sequence (in bold) appears in row 4 of the ordered table (Step 2); the final column of that table contains the sequence "bnnsaaa." Hence $BWT(\text{bananas}) = (\text{bnnsaaa}, 4)$.

describing the location of the original data sequence in the ordered list. An example giving the BWT of the word "bananas" appears in Fig. 1. Here $BWT(\text{bananas}) = (\text{bnnsaaa}, 4)$.[1]

For the BWT to be a *reversible* sequence transformation, it must be possible to reconstruct the full table of lexicographically ordered cyclic shifts using only the last column of the table (the BWT output). Intuitively, this reconstruction proceeds column by column as follows. By the table construction, the first column of the table is an ordered copy of the last column of the table. Thus, the *first* column reconstruction requires only an alphabetization of the list found in the last column. To reconstruct the second column, notice that each row is a cyclic shift of every other row, and hence that the last and first columns together provide a list of all consecutive pairs of symbols. Ordering this list of pairs yields the (first and) *second* column(s) of the table. Repeating this process on triples, quadruples, etc., sequentially reproduces all columns of the original table. The transform index indicates the desired row of the completed table. An example of the inverse BWT of the pair (bnnsaaa, 4) from the example in Fig. 1 appears in Fig. 2. Here $BWT^{-1}(\text{bnnsaaa}, 4) = \text{bananas}$.

While the above description of the BWT elucidates the algorithm, implementation of the forward and inverse transformation in the above manner would be impractical for long sequence lengths $n$. Practical implementations of the BWT require algorithms that are efficient in both time and space. As a result, a number of variations on the BWT appear in the literature. For example, the data may be passed through a run-length preprocessor to replace long strings of the same character (which, in addition to their obvious redundancy, cause longer sort times)

Table construction: For $i = 1, \ldots, n-1$,
Step $(3i-2)$: Place column $n$ in front of columns $1, \ldots, i-1$.
Step $(3i-1)$: Order the resulting length-$i$ strings lexicographically.
Step $3i$: Place the ordered list in the first $i$ columns of the table.

| | $(i=1)$ | | | $(i=2)$ | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | b | a | a ··· b | ba | an | an ··· b |
| 2 | n | a | a ··· n | na | an | an ··· n |
| 3 | n | a | a ··· n | na | as | as ··· n |
| 4 | s | b | b ··· s | sb | ba | ba ··· s |
| 5 | a | n | n ··· a | an | na | na ··· a |
| 6 | a | n | n ··· a | an | na | na ··· a |
| 7 | a | s | s ··· a | as | sb | sb ··· a |

| | $(i=3)$ | | | $(i=4)$ | | |
|---|---|---|---|---|---|---|
| | 7 | 8 | 9 | 10 | 11 | 12 |
| 1 | ban | ana | ana···b | bana | anan | anan···b |
| 2 | nan | ana | ana···n | nana | anas | anas···n |
| 3 | nas | asb | asb···n | nasb | asba | asba···n |
| 4 | sba | ban | ban···s | sban | bana | bana···s |
| 5 | ana | nan | nan···a | anan | nana | nana···a |
| 6 | ana | nas | nas···a | anas | nasb | nasb···a |
| 7 | asb | sba | sba···a | asba | sban | sban···a |

| | $(i=5)$ | | | $(i=6)$ | |
|---|---|---|---|---|---|
| | 13 | 14 | 15 | 16 | 17 |
| 1 | banan | anana | anana··b | banana | ananas |
| 2 | nanas | anasb | anasb··n | nanasb | anasba |
| 3 | nasba | asban | asban··n | nasban | asbana |
| 4 | sbana | banan | banan··s | sbanan | banana |
| 5 | anana | nanas | nanas··a | ananas | nanasb |
| 6 | anasb | nasba | nasba··a | anasba | nasban |
| 7 | asban | sbana | sbana··a | asbana | sbanan |

| | $(i = 6, \mathit{cont.})$ |
|---|---|
| | 18 |
| 1 | ananasb |
| 2 | anasban |
| 3 | asbanan |
| 4 | **bananas** |
| 5 | nanasba |
| 6 | nasbana |
| 7 | sbanana |

Fig. 2. The inverse BWT for (bnnsaaa, 4). The table is initialized with bnnsaaa in column $n$. Row 4 of the final table is the inverse BWT: $BWT^{-1}(\text{bnnsaaa}, 4) = \text{bananas}$.

[1] A variation of the BWT appends a unique "end-of-file" symbol to the end of the data sequence $x^n$. The algorithms used for coding employ the end-of-file symbol, as discussed in Section IV. The computational complexity results for the BWT assume a suffix-tree implementation, which uses an end-of-file symbol.

with run-length descriptions. Further, maximal sort lengths are sometimes imposed, with ties broken based on position in the original string. Descriptions of some of these variations and their performances appear in works like [1], [33]–[35], [7], [4]. While the choice of sorting technique used in any practical implementation should depend on the system priorities for that application, for the sake of simplicity, complexity and memory requirements given here refer to the first (of several) implementations of the BWT described by Burrows and Wheeler in [1]. The chosen implementation uses the suffix tree algorithm described in [36], which achieves $O(n)$ worst case complexity and memory results.

The BWT achieves data *expansion* rather than data compression. How then do algorithms working in the BWT domain yield such good performance–complexity tradeoffs? Roughly speaking, the BWT shifts the source redundancy caused by memory to a redundancy caused by a nonequiprobable and nonstationary first-order distribution.

Early BWT-based codes (e.g., [1], [37], [33], [34]) capitalize on the observation that the BWT tends to group together long strings of like characters (see, for example, Fig. 1), thereby producing a string that is more easily compressed than the original data sequence. Since the table's last column had the least impact on the ordering of the table's rows and is thus—in some sense—the *least* ordered of all columns, it is tempting to consider using some other column of the code table as the BWT output. Unfortunately, for general strings and sequence lengths, the last column is the only column that yields a reversible transformation. These observations together motivate a variety of alternatives to the BWT, such as the algorithms described in [3], [6], where modifications in the table generation techniques allow for use of earlier table columns.

While the argument that the last column of the BWT table has the least impact on the ordering of the table rows is indisputable, the supposition that the last column should therefore be the "least ordered" of all columns in the BWT table seems to fail when the data sequence derives from a finite-memory source. For example, according to this perspective, the columns of the BWT encoding table—taken from left to right—should appear progressively less ordered. Yet text files and other data types well-modeled as finite-memory sources fail to demonstrate this property. In particular, the last column almost always appears more ordered—with long sequences of like characters—than the columns that closely precede it (see Fig. 1). Understanding this paradox requires a better understanding of the BWT output when $X^n$ is drawn according to a finite-memory distribution.

## IV. The BWT on Finite-Memory Sources

Lexicographical ordering of the rows of the BWT table groups together all cyclic shifts of $X^n$ that begin with the same string. As a result, the BWT output, which describes the character that precedes the given string in each row, groups together symbols that precede like strings in $X^n$. Performing the BWT on a reversed data string groups together characters that follow like strings—i.e., characters with a common context. In finite-memory sources, this process groups together symbols from the same conditional distribution, creating a transformed data stream on which codes designed for p.i.i.d. source statistics yield excellent performance.[2] (A string is called $C$-p.i.i.d. if it is formed by concatenating together $C$ independent and identically distributed (i.i.d.) data streams.) The BWT's sorting properties are described precisely below. Coding results inspired by these properties are introduced in Section V. A more complete statistical characterization of the BWT output and its relationship to p.i.i.d. data streams is considered in Section VI.

| | Step 1 | Step 2 | Step 3 |
|---|---|---|---|
| 1 | sananab§ | ab§sanan | n |
| 2 | §sananab | anab§san | n |
| 3 | b§sanana | ananab§s | s |
| 4 | ab§sanan | b§sanana | a |
| 5 | nab§sana | nab§sana | a |
| 6 | anab§san | nanab§sa | a |
| 7 | nanab§sa | sananab§ | § |
| 8 | ananab§s | §sananab | b |

Fig. 3. The BWT of (sananab§) $=$ ($\mathcal{R}$(bananas)§). The end-of-file symbol § $\notin \mathcal{X}$ is ordered last lexicographically. Here $Z^{n+1}$ $=$ nnsaaa§b, $U = 7$, $Z_U = \S$, and $W^n$ $=$ nnsaaab.

Consider a stationary finite-memory source with alphabet $\mathcal{X}$, state space $\mathcal{S}$, and next-state function $\mathrm{suf}(\cdot)$. Given $X^n = X_1, X_2, \ldots, X_n$ drawn according to this distribution, let $Y^n = \mathcal{R}(X^n)$ and

$$(Z^{n+1}, U) = \mathrm{BWT}\,(Y^n\S) = \mathrm{BWT}\,(\mathcal{R}(X^n)\S)$$

where $\S \notin \mathcal{X}$ denotes an "end-of-file" symbol not found in the original source alphabet and $\mathcal{R}_n\colon \mathcal{X}^n \to \mathcal{X}^n$ is the time-reversal operator. Thus, $Y^n = (Y_1, \ldots, Y_n) = (X_n, \ldots, X_1)$, and $Z^{n+1}$ and $U$ are the BWT-reordered data sequence and row index, respectively, of the reversed data string modified by an end-of-file symbol.

The end-of-file symbol alleviates "edge effects," separating the beginning and end of the data stream in each cyclic shift and thereby avoiding problems where the contexts of the first $M$ symbols appear to contain characters from the end of the data stream. The use of the end-of-file symbol results in no expansion in either the sequence length or the alphabet size of $BWT_{\mathcal{X}}$ and makes the sequence $Z^{n+1}$ unique. More specifically, since all data strings must now end with the end-of-file symbol, if $U = BWT_{\mathbb{N}}(Y^n\S)$, then the $Z_U$ must equal §. Further, § can appear nowhere else in $Z^{n+1}$. Thus, the data string $X^n$ is uniquely characterized by either $Z^{n+1}$ or $(W^n, U)$, where

$$W^n = (Z_1, \ldots, Z_{U-1}, Z_{U+1}, \ldots, Z_{n+1}) \in \mathcal{X}^n.$$

An example for $X^n =$ bananas appears in Fig. 3, giving $W^n =$ nnsaaab. In this example, symbol $b$ appears to come from a context ending with the end-of-file symbol § rather than the character $s$ found at the end of the data stream.

[2] The same property is shared by the output of the BWT without time reversal, as discussed later in this section.

Recall that $\mathcal{S}$ is the state space for source $X^n$, and $\mathrm{suf}\,(\cdot)$ is the corresponding next-state function. Define $\mathcal{S}'$ to be the modified suffix set given by

$$\begin{aligned}\mathcal{S}' = \mathcal{S} \cup \{(\S, X_1, \ldots, X_i)&: \\ i < M \wedge [(X_j, \ldots, X_i) &\notin \mathcal{S}\, \forall 1 \le j \le i]\}\end{aligned}$$

where $M$ is the memory bound, and let $\mathrm{suf}'\,(\cdot)$ be the suffix operator for $\mathcal{S}'$. Let

$$\mathcal{Q} = \{\mathcal{R}(s)\colon s \in \mathcal{S}\} \quad \text{and} \quad \mathcal{Q}' = \{\mathcal{R}(s)\colon s \in \mathcal{S}'\}$$

and use $\mathrm{pre}\,(\cdot)$ and $\mathrm{pre}'\,(\cdot)$ to indicate the prefix operators for $\mathcal{Q}$ and $\mathcal{Q}'$, respectively. Assume that the prefixes of $\mathcal{Q}'$ are given by $\mathcal{Q}' = \{q_1', q_2', \ldots, q_C'\}$, where $q_1', \ldots, q_C'$ are ordered lexicographically and $C = |\mathcal{Q}'| = |\mathcal{S}'| \le |\mathcal{S}| + M$.

The symbols of $X^n$ arrange into $C$ contiguous substrings in $W^n$ so that the $j$th substring contains all characters with context $\mathcal{R}(q_j')$. Since $\S$ appears only once in the data string, each prefix $q' \in \mathcal{Q}' \cap \mathcal{Q}^c$ begins in the leftmost column of the BWT table exactly once. As a result, the substring associated with any such $q'$ contains exactly one element. For each $j \in \{1, \ldots, C+1\}$, define $T_j$ as

$$\begin{aligned}T_j &= 1 + \sum_{i=1}^{n} \sum_{k=1}^{j-1} 1(\mathrm{suf}'(\S X_1^i) = \mathcal{R}(q_k')) \\ &= 1 + \sum_{i=1}^{n} \sum_{k=1}^{j-1} 1(\mathrm{pre}'(Y_i^n \S) = q_k').\end{aligned}$$

The substring $W_{T_j}, \ldots, W_{T_{j+1}-1}$ contains all characters $Y_i$ that precede $q_j'$ in $Y^n\S$, or, equivalently, all characters $X_i$ that occur in context $\mathcal{R}(q_j')$ in the original data stream $X^n$. As noted earlier, $T_{j+1} - T_j = 1$ for all $j$ such that $q_j' \in \mathcal{Q}' \cap \mathcal{Q}^c$, and, thus, $|\{j\colon T_{j+1} - T_j > 1\}| \le |\mathcal{S}|$.

Since the mappings between $X^n$ and $Y^n$ and $Z^{n+1}$ and $(W^n, U)$ are all one-to-one

$$\begin{aligned}\Pr(W^n, U) &= \Pr(Z^{n+1}) = \Pr(Y^n) = \Pr(X^n) \\ &= p(X_1^M) \prod_{i=M+1}^{n} p(X_i | \mathrm{suf}(X_1^{i-1})) \\ &= \prod_{i=1}^{n} p(X_i | \mathrm{suf}'(\S X_1^{i-1})) \\ &= \prod_{i=1}^{n} p(Y_i | \mathcal{R}(\mathrm{pre}'(Y_{i+1}^n \S))) \\ &= \prod_{j=1}^{C} \prod_{i=T_j}^{T_{j+1}-1} p(W_i | \mathcal{R}(q_j')) \\ &= \prod_{j=1}^{C} \prod_{i=T_j}^{T_{j+1}-1} p_j(W_i) \qquad (1)\end{aligned}$$

where $p_j(x) = p(x|\mathcal{R}(q_j'))$ and $p(x|s') = p(x|s_2, \ldots, s_k)$ for any $x \in \mathcal{X}$ and any $s' = (\S, s_2, \ldots, s_k) \in \mathcal{S}' \cap \mathcal{S}^c$.

While (1) resembles the distribution of a p.i.i.d. data stream, the statement that $W^n$ is $C$-p.i.i.d. [18, Lemma 1] is not accurate. First, (1) implies that $\Pr((W^n, U) = (w^n, u)) = 0$ for all $(w^n, u)$ for which

$$((w_1, \ldots, w_{u-1}, \S, w_u, \ldots, w_n), u) \notin \{BWT(y^n \S)\colon y^n \in \mathcal{X}^n\}.$$

Second, (1) describes the probability of $(W^n, U)$; the probability of $W^n$, given by

$$\Pr(W^n) = \sum_{u=1}^{n+1} \Pr(W^n, U = u)$$

is greater than or equal to $\Pr(W^n, U)$. Nonetheless, the decision in [18] to code $W^n$ using codes designed for $C$-p.i.i.d. data strings is well motivated, and none of the results of [18] require that the source is actually $C$-p.i.i.d. Section V gives the derivations for all of these results. The analysis from [19] of the distribution on the BWT output demonstrates, for a variety of input distributions, that the normalized divergence between the output distribution of the BWT and a p.i.i.d. distribution is asymptotically vanishing; we consider this issue in Section VI. A few remarks are useful before proceeding with those results.

*Remark 1:* While the idea of reversing the data string $X^n$ prior to transformation is conceptually useful, string reversal is not necessary to obtain an equation of the form given in (1). This assertion follows from [38], which proves that the time reversal of any finite-memory source yields another finite-memory source. As a result, for any data sequence $X_1, X_2, \ldots$ drawn from a stationary finite-memory distribution for which the reversed data string has minimum suffix set $\hat{\mathcal{S}}$ and memory constraint $\hat{M}$, if $(\hat{Z}^{n+1}, \hat{U}) = \text{BWT}\,(X^n\S)$ and

$$\hat{W}^n = (\hat{Z}_1, \ldots, \hat{Z}_{\hat{U}-1}, \hat{Z}_{\hat{U}+1}, \ldots, \hat{Z}_{n+1})$$

then there exists $\hat{C} \le |\hat{\mathcal{S}}| + \hat{M}$, $\{\hat{p}_j\}_{j=1}^{\hat{C}}$, and

$$1 = \hat{T}_1 \le \cdots \le \hat{T}_{\hat{C}+1} = n + 1$$

such that $|\{j\colon \hat{T}_{j+1} - \hat{T}_j > 1\}| \le |\hat{\mathcal{S}}|$ and

$$\Pr(\hat{W}^n, \hat{U}) = \prod_{j=1}^{\hat{C}} \prod_{i=\hat{T}_j}^{\hat{T}_{j+1}-1} \hat{p}_j(\hat{W}_i).$$

Note, however, that $|\hat{\mathcal{S}}|$ is not necessarily equal to $|\mathcal{S}|$ [38]. Since rate of convergence results—including the optimal rate of convergence results described in Section II—typically depend on the number of states in the model, the optimal rate of convergence for the forward finite-memory source model may differ from the optimal rate of convergence for the reverse finite-memory source model. This observation reminds us that while the $(|\mathcal{S}|(|\mathcal{X}| - 1)/2) \log n/n$ bounds from below the rate of convergence achievable using a finite-memory source model with $|\mathcal{S}|$ states, proving this rate of convergence optimal for the underlying random process requires proof that there does not exist an equivalent model with fewer than $|\mathcal{S}|$ states. This work follows the approach found throughout the universal coding literature and bounds the performance achieved subject to a particular source model. Thus, the data string $X^n$ may be thought of as either the original data sequence or its reversal. Since the "finite-memory source model" refers to the model for $X^n$, the time-reversal step is left in the algorithmic description. Equivalent results for the time-reversed source apply immediately when running the algorithms without time reversal.

*Remark 2:* As shown in (1), the BWT of the data string (or its time-reversed equivalent) is similar in distribution to a sorted list of i.i.d. samples with a number of parameter changes comparable to the number of states in the finite-memory source. The BWT achieves this property on *any* finite-memory source independent of the suffix set $\mathcal{S}$ and without any required *a priori* knowledge of the state space in operation. In particular, the results described in the section that follows hold for the *best* finite-memory source model for the source in operation, and the bounds of this best model dominate.

*Remark 3:* In addition to its direct source coding ramifications, (1) also lends insight into the characteristics of good source models for common data types such as text. Applying the BWT to text data sets tends to yield long strings of like characters. Combining the statistical property described by (1) with this experimental observation suggests that the conditional distributions found in the finite-memory source model for text tend to have very narrow supports. While some short contexts achieve narrow supports (e.g., the letter "q" is almost always followed by the letter "u" in English text), most short contexts may be followed by many different characters. Thus, the prevalence of narrow supports suggests that long context lengths are in effect in data types such as text. As a result, algorithms that achieve good performance on sources with long contexts and conditional distributions with narrow supports should take precedence over algorithms lacking these properties.

## V. Universal Lossless Source Codes

The BWT, as a reversible transformation, cannot affect the shortest description length achievable in lossless compression of samples from a particular source model. It can, however, make achieving that performance less computationally taxing. This goal motivates the following discussion, containing introductions to and analyses of a variety of BWT-based source coding strategies for achieving universal source coding performance on stationary finite-memory sources. All but one of the strategies and all of the rate of convergence results considered here were originally described in [18]. The remaining strategy, treated first, is a variation on the BWT-based lossless source code in common use for practical coding. Information-theoretic analyses, including proofs of universality and bounds on the associated rates of convergence, play central roles here. Discussions of the complexity and memory requirements are also included.

Recall from (1) that $(Z^{n+1}, U) = \text{BWT}_{\mathcal{X}}(\mathcal{R}(X^n)\S)$ for some $X^n$ drawn according to an $|\mathcal{S}|$-state finite-memory source and $W^n = (Z_1, \dots, Z_{U-1}, Z_{U+1}, \dots, Z_{n+1})$ implies that $U \in \{1, \dots, n+1\}$ and

$$\Pr(W^n, U) = \prod_{j=1}^{C} \prod_{i=T_j}^{T_{j+1}-1} p_j(W_i)$$

with $C \leq |\mathcal{S}| + M$ and no more than $|\mathcal{S}|$ subsequences of more than one character each. The algorithms considered here use independent codes for describing $U$ and $W^n$. Assuming that the decoder knows the sequence length $n$, the natural $(\lfloor \log(n+1) \rfloor + 1)$-bit binary expansion suffices for describing $U$. (When the decoder does not know the sequence length $n$, a description of length $2\log n + O(\log\log n)$ bits suffices for describing both $U$ and $n$.) While the BWT is extremely computationally efficient and most of the algorithms considered use very simple sequential codes to describe the BWT output, none is a sequential code.

### A. Finite-Memory Sources

First, consider the question of universality on the class of finite-memory sources. Most of the algorithms considered here achieve universality on this class of sources without *a priori* knowledge of the memory constraint $M$ (as in algorithms such as [31]) or state space $\mathcal{S}$.

The rate of convergence results for finite-memory sources use the redundancy expression

$$\Delta_n(\theta) = \frac{1}{n} E_\theta \ell(X^n) - H_\theta(\mathcal{X})$$

rather than the expression

$$\delta_n(\theta) = E_\theta \ell(X^n)/n - H_\theta(X^n)/n$$

used in Section II. (Here $H_\theta(\mathcal{X})$ denotes the entropy rate of source $P_\theta$.) Note that

$$\Delta_n(\theta) = \delta_n(\theta) + (H_\theta(X^n)/n - H_\theta(\mathcal{X}))$$

and recall that the optimal rate of convergence of $\delta_n(\theta)$ on the class $\Lambda$ of stationary finite-memory sources is $O(\log n/n)$. Thus, since $H_\theta(X^n)/n - H_\theta(\mathcal{X})$ is $O(1/n)$ for stationary finite-memory sources (see Lemma 4 in the Appendix), the optimal rates of convergence for $\delta_n(\theta)$ and $\Delta_n(\theta)$ are identical to first order.

Given a stationary finite-memory source with state space $\mathcal{S}$ and conditional distributions $\{p(x|s)\}$, the entropy rate of the given source is

$$H_\theta(\mathcal{X}) = \sum_{s \in \mathcal{S}} \pi_s H(p_s)$$

where, for each $s \in \mathcal{S}$, $\pi_s$ is the probability of state $s$ and

$$H(p_s) = -\sum_{x \in \mathcal{X}} p(x|s) \log p(x|s)$$

is the conditional entropy of $X$ given $s$. For any data string $x^n \in \mathcal{X}^n$, let $\{\hat{\pi}_s(x^n)\}$ denote the empirical distribution over the states $s \in \mathcal{S}$ for sequence $x^n$. Similarly, for each $s \in \mathcal{S}$, let $H(\hat{p}_s(x^n))$ denote the conditional entropy associated with the empirical distribution on $\mathcal{X}$ given $s$. Then the "empirical entropy" of $x^n$ relative to state space $\mathcal{S}$ is defined as

$$H_{\hat{\theta}(x^n, \mathcal{S})}(\mathcal{X}) = \sum_{s \in \mathcal{S}} \hat{\pi}_s(x^n) H(\hat{p}_s(x^n)).$$

### B. A Move-to-Front Code: The Baseline BWT Algorithm

The BWT-based codes described in works like [1]–[3] present a logical starting point in the analysis of BWT-based codes. Since all of these algorithms use variations on move-to-front coding in describing the BWT output, a description of move-to-front coding follows.

The idea behind move-to-front coding appears in a variety of works under a variety of names, including the "book stack" codes of [39], the "move-to-front" codes of [40], [41], and the "interval" and "recency ranking" codes of [42]. In each case, the description length of a particular symbol or word depends on the recency of its last appearance. Symbols used more recently get shorter descriptions than symbols used less recently. The algorithms differ somewhat in their definitions of recency, describing either the interval since that symbol's last appearance [42] or that symbol's rank in a list of symbols ordered by their recency [39]–[42]. More precisely, in describing $x_1, \ldots, x_n$, at time $i$, the interval coding encoder describes

$$f_{\text{int}}(i) = \min\{k \geq 1: x_{i-k} = x_i\}$$

while the recency ranking encoder describes

$$f_{\text{rr}}(i) = |\{x_k: i - f_{\text{int}}(i) < k \leq i\}|$$

where for any set $\mathcal{A} \subseteq \mathcal{X}$, $|\mathcal{A}|$ denotes the number of *distinct* elements in $\mathcal{A}$, and thus $|\mathcal{A}| \leq |\mathcal{X}|$. Assuming that the system memory is initialized with an ordered list of all elements from alphabet $\mathcal{X}$, then the data sequence $x_1, x_2, \ldots, x_n$ may be uniquely derived from either $f_{\text{int}}(1), \ldots, f_{\text{int}}(n)$ or $f_{\text{rr}}(1), \ldots, f_{\text{rr}}(n)$. Thus, any lossless code on either the intervals or the recency ranks uniquely describes any $x^n \in \mathcal{X}^n$.

Given a collection of symbols drawn i.i.d. from some fixed distribution $p(x)$ on source alphabet $\mathcal{X}$, the known performance bounds for codes based on interval and recency ranking strategies are the same [39]–[42]. Nonetheless, for any data sequence $X_1, X_2, \ldots$, $f_{\text{int}}(i) \geq f_{\text{rr}}(i)$ for all $i$, and, thus, for any code in which the description length for integer $j$ is nondecreasing in $j$, the description length using a code based on interval coding cannot be better than the description length using a code based on recency ranking. Further, the *maximal* value of $f_{\text{int}}(i)$ equals $i$, which may grow arbitrarily large, while the maximal value of $f_{\text{rr}}(i)$ equals the alphabet size $|\mathcal{X}|$, which is finite and fixed, a fact that simplifies later arguments. Thus, the discussion that follows uses recency ranking rather than interval coding.

The move-to-front algorithm considered here uses an integer code to describe the recency rank of each symbol $W_i$, $i \in \{1, \ldots, n\}$. The chosen integer code is a logical extension of Elias' codes [42]. In place of Elias' codes of lengths

$$L_1(j) = 1 + 2\lfloor\log(j)\rfloor$$
$$\text{and } L_2(j) = 1 + \lfloor\log(j)\rfloor + 1 + 2\lfloor\log(\lfloor\log(j)\rfloor + 1)\rfloor$$

this code describes any $j \geq 1$ with $L_3(j) \leq G_3(\log(j))$ bits, where

$$G_3(x) = x + \log(x+1) + 2\log(\log(x+1)+1) + 3.$$

The function $L_3(j)$ approximates $L_\infty(j) = \log^*(j) + c$ to sufficient accuracy for this work. (Here $\log^*(j) = \log(j) + \log(\log(j)) + \cdots$, ending the sum with its last positive term, and $c$ is chosen to satisfy Kraft's inequality on the alphabet of interest.)

The use of an integer code after the BWT and the move-to-front algorithm differs significantly from the approaches used in algorithms like [1]–[3], which follow the BWT and move-to-front algorithm with a first-order entropy code. Since this work contains no direct analysis of this (extremely popular) entropy coding approach, a brief digression to compare these alternatives follows. In any move-to-front code—based either on interval coding or on recency ranking—at time $i$, after coding subsequence $X^{i-1}$, there are exactly $|\mathcal{X}|$ possible integers that the encoder might need to describe. These $|\mathcal{X}|$ integers (which, in the case of interval coding, vary as a function of $X^{i-1}$) describe the intervals or recency ranks at time $i$ of all characters $x \in \mathcal{X}$ and are known to both the encoder and the decoder. If the data sequence to be compressed happens to be i.i.d., then at time $i$ the true conditional probability—conditioned on the full history of the data sequence—of the interval for character $x$ equals $p(x)$. Thus, for a memoryless source, the best entropy code on the move-to-front symbols requires memory and achieves performance no better than that of the best first-order entropy code on the original data sequence. In fact, given an i.i.d. source and a first-order entropy code, move-to-front coding may actually *hurt* performance, since the first-order statistics of the move-to-front symbols may not match the source's first-order statistics.

The analysis is more complicated for data sequences with piecewise-constant distributions. Intuitively, by typically mapping more probable characters to low indexes and less probable characters to high indexes, move-to-front coding may effectively make the distributions of neighboring subsequences look more similar to each other. The result, then, would be to decrease the penalty associated with treating neighboring subsequences as if they come from the same distribution. Notice, however, that this argument only applies when symbols from different distributions are treated as if they came from the same distribution. Since any code that takes such an approach on more than an asymptotically insignificant portion of the data sequence cannot help but fail the test for universality, this argument suggests that the move-to-front algorithm should, at best, have an asymptotically negligible benefit for the performance of *universal* codes that employ both the BWT and entropy coding. Since the combination of the move-to-front algorithm and entropy coding complicates the analysis considerably, this work contains an analysis of the move-to-front algorithm with integer coding and several analyses of entropy coding without the move-to-front algorithm but does not treat the move-to-front algorithm with entropy coding.

Combining the BWT with the move-to-front algorithm and integer coding results in a very simple source coding algorithm. The BWT gives

$$(Z^{n+1}, U) = \text{BWT}\,(\mathcal{R}(X^n)\S).$$

Replacing $W^n = (Z_1, \ldots, Z_{U-1}, Z_{U+1}, \ldots, Z_n)$ with the associated recency ranks yields sequence $f_{\text{rr}}(1), \ldots, f_{\text{rr}}(n)$ from alphabet $\{1, \ldots, |\mathcal{X}|\}$. Finally, $\lfloor\log(n+1)\rfloor + 1$ bits and $\sum_{i=1}^{n} L_3(f_{rr}(i))$ bits, respectively, suffice for describing first $U$ and then $f_{\text{rr}}(1), \ldots, f_{\text{rr}}(n)$. The decoder reverses the above procedure. While this algorithm is not universal when performed on alphabet $\mathcal{X}$, it can be made universal by applying the algorithm on extensions $\mathcal{X}^m$ of $\mathcal{X}$ with $m \to \infty$, as discussed in [15]–[17]. The proof used here takes a different approach from the typicality arguments of the earlier works. The new approach results in a rate of convergence result in addition to a proof of universality. The following analysis of the properties of a data sequence created by blocking together symbols from a finite-memory source plays an important role in that analysis.

Consider a data sequence $X_1, X_2, X_3, \ldots$ drawn from a finite-memory source with alphabet $\mathcal{X}$, state space $\mathcal{S}$, and memory constraint $M$. Blocking the data sequence into $m$-vectors yields a new data sequence $X_1^m, X_{m+1}^{2m}, X_{2m+1}^{3m}, \ldots$ on alphabet $\mathcal{X}^m$. The resulting $m$-vector source is also a finite-memory source, since the distribution on the next $m$-vector relies on a maximum of $\lceil M/m \rceil$ previous $m$-vectors. Next, consider the number $|\hat{\mathcal{S}}_m|$ of distinct conditional distributions for the $m$-vector source. This calculation is less straightforward since that size relies on the states in the state space $\mathcal{S}$ rather than merely the size of that state space. The number $|\hat{\mathcal{S}}_m|$ of distinct conditional distributions in the $m$-vector source is as small as $|\mathcal{S}|$ for some sources but exceeds $|\mathcal{S}|$ for others. In particular, $|\hat{\mathcal{S}}_m| \le |\hat{\mathcal{S}}_M|$ for all $m$, and $|\hat{\mathcal{S}}_m| = |\hat{\mathcal{S}}_M|$ for all $m \ge M$. Further, $|\hat{\mathcal{S}}_M| \le |\mathcal{X}^M| = |\mathcal{X}|^M$.

*Theorem 1:* The BWT-based source code that combines recency-ranking with an integer code describing integer $j > 1$ with description length $L_3(j) \le G_3(\log j)$ and

$$G_3(x) = x + \log(x+1) + 2\log(\log(x+1)+1) + 3$$

achieves per-symbol description length

$$\frac{\ell_n(x^n)}{n} \le G_3\left(H_{\hat{\theta}(x^n,\mathcal{S})}(\mathcal{X}) + \frac{(|\mathcal{S}|+M)|\mathcal{X}|}{n}\log e\right) + \frac{\log(n+1)}{n} + \frac{1}{n}$$

bits per symbol for each $x^n \in \mathcal{X}^n$. Given a finite-memory source with unknown state space $\mathcal{S}$ and memory constraint $M < \infty$, the resulting redundancy is

$$\Delta_n(\theta) \le \log(H_\theta(\mathcal{X})+1) + 2\log(\log(H_\theta(\mathcal{X})+1)+1) + 3 + O\left(\frac{\log n}{n}\right)$$

bits per symbol. Given any $\epsilon > 0$, applying the above code to alphabet $\mathcal{X}^m$, where $m$ grows with $n$ as

$$m = \frac{\log(n \log\log n/(|\hat{\mathcal{S}}_M|(\log n)^2))}{\log|\mathcal{X}|}$$

yields a weakly minimax universal code with redundancy bounded as

$$\Delta_n(\theta) \le (1 + (1+\epsilon)\log|\mathcal{X}|)\frac{\log\log n}{\log n} + O\left(\frac{\log\log\log n}{\log n}\right)$$

bits per symbol for all $\theta$ in the class of finite-memory sources. When $|\hat{\mathcal{S}}_M|$ is unknown and the growth rate of $m$ cannot depend on $|\hat{\mathcal{S}}_M|$, setting

$$m = \frac{\log(n\log\log n/(\log n)^2)}{\log|\mathcal{X}|}$$

yields $\Delta_n(\theta) \le O(\log\log n/\log n)$.

*Proof:* For any fixed sequence $v^n \in \mathcal{X}^n$ and any $x \in \mathcal{X}$, use $\mathcal{N}(x) \subseteq \{1, \ldots, n\}$ to describe all positions in which symbol $x \in \mathcal{X}$ appears in sequence $v^n$. Following the argument of [42], the description length of the given recency rank code on sequence $v^n$ is bounded as

$$\begin{aligned}
\ell_n(v^n) &= \sum_{x\in\mathcal{X}}\sum_{i\in\mathcal{N}(x)} L_3(f_{\mathrm{rr}}(i)) \\
&\le nG_3\left(\sum_{x\in\mathcal{X}}\frac{|\mathcal{N}(x)|}{n}\log\left(\frac{\sum_{i\in\mathcal{N}(x)} f_{\mathrm{rr}}(i)}{|\mathcal{N}(x)|}\right)\right) \\
&= nG_3\left(H(\hat{\theta}(v^n)) + \sum_{x\in\mathcal{X}}\frac{|\mathcal{N}(x)|}{n}\log\left(\frac{1}{n}\sum_{i\in\mathcal{N}(x)} f_{\mathrm{rr}}(i)\right)\right) \\
&\le nG_3\left(H(\hat{\theta}(v^n)) + \log\left(1 + \frac{|\mathcal{X}|}{n}\right)\right) \\
&\le nG_3\left(H(\hat{\theta}(v^n)) + \frac{|\mathcal{X}|}{n}\log e\right)
\end{aligned}$$

where $H(\hat{\theta}(v^n))$ is the first-order entropy of the empirical distribution of $v^n$. The first inequality results from two applications of Jensen's inequality; the second inequality follows from $\sum_{i\in\mathcal{N}(x)} f_{\mathrm{rr}}(i) \le n + |\mathcal{X}|$, since $G_3(\cdot)$ and $\log(\cdot)$ are increasing; $\log(1+a) \le a\log e$ gives the final inequality.

Now recall from (1) that the distribution of $(W^n, U)$ contains $C$ subsequences with $C \le |\mathcal{S}| + M$. Fix an arbitrary $x^n \in \mathcal{X}^n$, and let $(w^n, u)$ be the corresponding BWT description. Since the above analysis uses an arbitrary memory initialization, that analysis applies to each of the $C$ subsequences. In particular, if $\{T_j\}$ describes the transitions between the $C$ subsequences, then summing up the description lengths for the $C$ subsequences of $w^n$ and the description length for $u$ gives

$$\begin{aligned}
\ell_n(x^n) &\le \sum_{j=1}^{C}(T_{j+1}-T_j)G_3\left(H\left(\hat{\theta}\left(w_{T_j}^{T_{j+1}-1}\right)\right) + \frac{|\mathcal{X}|}{T_{j+1}-T_j}\log e\right) + \log(n+1) + 1 \\
&\le nG_3\left(\sum_{j=1}^{C}\frac{(T_{j+1}-T_j)}{n}H\left(\hat{\theta}\left(w_{T_j}^{T_{j+1}-1}\right)\right) + \frac{C|\mathcal{X}|}{n}\log e\right) + \log(n+1) + 1 \\
&\le nG_3\left(H_{\hat{\theta}(x^n,\mathcal{S})}(\mathcal{X}) + \frac{(|\mathcal{S}|+M)|\mathcal{X}|}{n}\log e\right) + \log(n+1) + 1.
\end{aligned}$$

Taking an expectation with respect to the distribution on $X^n$ gives

$$E_\theta\ell_n(X^n) \le nG_3\left(H_\theta(\mathcal{X}) + \frac{(|\mathcal{S}|+M)|\mathcal{X}|}{n}\log e\right) + \log(n+1) + 1$$

by Jensen's inequality. The redundancy of the resulting code is

$$\Delta_n(\theta) \le \log(H_\theta(\mathcal{X})+1) + 2\log(\log(H_\theta(\mathcal{X})+1)+1) + 3 + O\left(\frac{\log n}{n}\right)$$

which approaches a constant greater than zero as $n$ grows without bound.

Now consider applying the above algorithm to the $m$-vector data source. To be more exact, first reverse the data sequence $X^n$, and then break the *reversed* data sequence into $m$-vectors. (For simplicity, assume that $m$ divides $n$ evenly.) Notice that since the distribution of any symbol from the original data sequence depends on at most $M$ previous symbols, the distribution on any $m$-vector in the blocked data sequence depends on at most $\lceil M/m \rceil$ previous $m$-vectors. Now append an $m$-vector of end of file symbols § and run the BWT on the $m$-vector alphabet. In this case, the integer portion of the BWT falls between 1 and $n/m+1$ and the transformed data sequence has $C$ subsequences with $C \le |\hat{\mathcal{S}}_m| + \lceil M/m \rceil$. Using the move-to-front algorithm (for alphabet $\mathcal{X}^m$) and an integer code on the data sequence of $m$-vectors and applying the natural fixed-length binary description to the BWT row index gives a description length satisfying

$$\frac{E_\theta \ell_n(X^n)}{n} \le \frac{1}{n}\left[\frac{n}{m} G_3\left(H_\theta(\mathcal{X}^m) + \frac{(|\hat{\mathcal{S}}_m| + M/m + 1)|\mathcal{X}|^m \log e}{n/m}\right) + \log\left(\frac{n}{m}+1\right) + 1\right]$$

where $H_\theta(\mathcal{X}^m)$ is the entropy rate of the vector source created by breaking the data sequence $X_1, X_2, \ldots$ into $m$-vectors. When $|\mathcal{X}|^m/n \to 0$ as $m$ and $n$ grow without bound, the redundancy (relative to the original source alphabet) of the resulting code satisfies

$$\begin{aligned}
\Delta_n(\theta) &\le \frac{(|\hat{\mathcal{S}}_M| + M/m + 1)|\mathcal{X}|^m \log e}{n} \\
&\quad + \frac{1}{m}\log\left(H_\theta(\mathcal{X}^m) + 1 + \frac{(|\hat{\mathcal{S}}_M| + M/m + 1)|\mathcal{X}|^m \log e}{n/m}\right) \\
&\quad + \frac{2}{m}\log\left(\log\left(H_\theta(\mathcal{X}^m) + 1 + \frac{(|\hat{\mathcal{S}}_M| + M/m + 1)|\mathcal{X}|^m \log e}{n/m}\right) + 1\right) + \frac{c}{m} \\
&\le \frac{(|\hat{\mathcal{S}}_M| + M/m + 1)|\mathcal{X}|^m \log e}{n} \\
&\quad + \frac{\log m}{m} + \frac{2\log\log m}{m} + \frac{\log n}{n} + \frac{c'}{m}
\end{aligned}$$

for large enough $m$ and $n$, where $c$ and $c'$ are nonnegative constants.

If

$$m = \frac{1}{\log|\mathcal{X}|}\log\left(\frac{n\log\log n}{(|\hat{\mathcal{S}}_M| + 2)\log e \log n}\right)$$

then $|\mathcal{X}|^m/n = O(\log\log n/\log n)$ satisfies the constraint that $|\mathcal{X}|^m/n \to 0$ as $m$ and $n$ grow without bound, and for any $\epsilon > 0$ and $n$ large enough

$$\begin{aligned}
&\frac{(|\hat{\mathcal{S}}_M| + M/m + 1)|\mathcal{X}|^m \log e}{n} \\
&\quad = \left(\frac{n\log\log n}{(|\hat{\mathcal{S}}_M| + 2)\log e \log n}\right)\frac{(|\hat{\mathcal{S}}_M| + 1 + M/m)\log e}{n} \\
&\quad \le \frac{\log\log n}{\log n}
\end{aligned}$$

while

$$\begin{aligned}
\frac{\log m}{m} &= \frac{\log\log\left(\frac{n\log\log n}{(|\hat{\mathcal{S}}_M|+2)\log e \log n}\right) - \log\log|\mathcal{X}|}{\frac{1}{\log|\mathcal{X}|}\log\left(\frac{n\log\log n}{(|\hat{\mathcal{S}}_M|+2)\log e \log n}\right)} \\
&\le \log|\mathcal{X}|\,\frac{\log\log n}{\log n/(1+\epsilon)}
\end{aligned}$$

and, thus, the dominant terms balance. When $|\hat{\mathcal{S}}_M|$ is unknown, similar results may be achieved by simply removing the dependence of $m$ on $|\hat{\mathcal{S}}_M|$. In particular, setting

$$m = \log(n\log\log n/\log n)/\log|\mathcal{X}|$$

yields $O(\log\log n/\log n)$ convergence. □

While the baseline code is not universal, the code is very simple, and for practical $n$-values the constant to which the redundancy converges may be benign. The algorithm uses a fixed integer code with only $|\mathcal{X}|$ symbols, and implementation of the move-to-front transformation, like the BWT, requires only linear complexity, making the algorithm $O(n)$ in space and time complexity. (Throughout this work, space and time complexity appear as a single result since most algorithms allow easy tradeoffs between the two.)

In contrast with the baseline algorithm, the extension code is universal, but the resulting code appears to be more expensive in space and time complexity. In particular, allowing $m$ to grow with $n$ as in Theorem 1 gives alphabet size $|\mathcal{X}|^m = O(n\log\log n/\log n)$, a value similar in size to the sequence length itself.

Applying McCreight's suffix tree algorithm [36] on the new larger alphabet results in worst case

$$O(|\mathcal{X}|^m n) = O(n^2 \log\log n/\log n)$$

space and time complexity. The expected space and time complexity may be considerably lower than these worst case results for distributions encountered in data sources such as text. In particular, the expected space and time complexity are proportional to the number of distinct characters from $|\mathcal{X}|^m$ *that appear in that data string* rather than the number of characters in the alphabet itself. Since many combinations of characters never appear in English text, this number of distinct characters used may be significantly smaller than $|\mathcal{X}|^m$.

A second approach for implementing the BWT on alphabet $\mathcal{X}^m$, proposed by McCreight [36] for use on large alphabets, involves an alternative hash table implementation of the same

**Alphabet:** $\mathcal{X} = \{\text{a,b,n}\}$
**Output:** $BWT(\text{banana}) = (\text{nnbaaa}, 4)$

| | Step 1 | Step 2 | Step 3 |
|---|---|---|---|
| 1 | b a n a n a | a b a n a n (2) | n |
| 2 | a b a n a n | a n a b a n (4) | n |
| 3 | n a b a n a | a n a n a b (6) | b |
| 4 | a n a b a n | b a n a n a (1) | a |
| 5 | n a n a b a | n a b a n a (3) | a |
| 6 | a n a n a b | n a n a b a (5) | a |

**Alphabet:** $\mathcal{X}^2 = \{\text{aa,ab,an,ba,bb,bn,na,nb,nn}\}$
**Output:** $BWT(\text{banana}) = ((\text{na,na,ba}), 1)$

| | Step 1 | Step 2 | Step 3 |
|---|---|---|---|
| 1 | ba na na | ba na na | na |
| 2 | na ba na | na ba na | na |
| 3 | na na ba | na na ba | ba |

Fig. 4. The BWT encoding table on alphabets $\mathcal{X}$ and $\mathcal{X}^2$. The BWT encoding table on alphabet $\mathcal{X}^2$ may be derived from the BWT encoding table on alphabet $\mathcal{X}$ by deleting those elements that sat in rows 2, 4, and 6 in Step 1.

suffix tree algorithm. The resulting implementation reduces the memory to $O(n \log n)$, but yields $O(n \log n)$ memory even on small alphabets.

The last approach considered here for implementing the BWT on alphabet $\mathcal{X}^m$ derives from the relationship between the BWT on alphabet $\mathcal{X}$ and the BWT on alphabet $\mathcal{X}^m$. Assume that $m$ divides $n$ evenly. For any fixed data string $x^n$, the BWT output achieved by treating $x^n$ as $n/m$ symbols from $\mathcal{X}^m$ may be strikingly different from the BWT output achieved by treating $x^n$ as $n$ symbols from $\mathcal{X}$. Yet as Fig. 4 demonstrates, the BWT encoding tables are closely related. While the BWT table for $\mathcal{X}^m$ has fewer rows, each row in the BWT table for $\mathcal{X}^m$ has a corresponding row in the BWT table for $\mathcal{X}$. Further, the ordering of those rows is the same in both tables. As a result, the BWT for alphabet $\mathcal{X}^m$ may be achieved by building a BWT encoding table on alphabet $\mathcal{X}$ and then removing all values corresponding to rows not used for alphabet $\mathcal{X}^m$. This approach yields $O(n)$ space and time complexity. Thus, alphabet extension does not increase the order of the memory or complexity required.

Nonetheless, several drawbacks of the alphabet extension procedure persist. In particular, the BWT implementation must vary as a function of $n$ and application of entropy coding (as in [1] and most of its followers) rather than integer coding (used here) after the move-to-front algorithm becomes computationally prohibitive for large alphabet sizes. Further, [43, Theorem 1] shows that applying the move-to-front algorithm on the $m$th-order extensions yields universal coding performance; thus, the BWT is unnecessary for universality given the $m$th-order extensions. The universal algorithms described in the remainder of this work use no alphabet extensions and extremely simple and memory-efficient source codes.

### *C. Known State Space $\mathcal{S}$ or Memory Constraint $M \leq L$*

As discussed in Section IV, the BWT sorts the data sequence of a finite memory source so that all symbols drawn according to the same conditional distribution are grouped in a single contiguous subsequence of the transform output. When the boundaries between those subsequences are known, universal coding performance on the finite-memory source may be achieved using a separate universal source code on each subsequence.

In order to employ a strategy based on the above observation, it is necessary for the state space to be known *a priori*. For this work, reference to a "known state space" implies that the *encoder* knows the state space in operation; the decoder's knowledge (or lack thereof) does not affect the algorithm. Note that the known state-space condition is not as restrictive as it seems initially. The algorithm considered here achieves performance approaching the best possible performance achievable using the model assumed at the encoder. If the encoder estimates $\mathcal{S}$ as $\mathcal{X}^m$, then the resulting algorithm guarantees performance approaching the best possible performance for a Markov-$m$ model of the given source. (Allowing $m$ to grow eventually yields a code with $m$ greater than or equal to the true source memory constraint $M$.) Further, the encoder has access to the full data sequence $X^n$, and thus the encoder can *always* know the state space $\mathcal{S}$ to arbitrary accuracy given sufficient computational and memory resources. Thus, the assumption of a known state space $\mathcal{S}$ may be matched by practical algorithms that use either guesses or estimates of the state space in their encoders. In this subsection, the space and time complexity of the estimation procedure are not included in the analysis, and statements of universality apply only when the estimate of $\mathcal{S}$ or $M$ is accurate. The known state-space assumption applies only to this algorithm.

Let $X_1, X_2, \ldots$ be drawn from a finite-memory source with *known* state space $\mathcal{S}$. If $(Z^{n+1}, U) = \text{BWT}(\mathcal{R}(X^n)\S)$ and $W^n = (Z_1, \ldots, Z_{U-1}, Z_{U+1}, \ldots, Z_{n+1})$, then $W^n$ comprises the $C$ subsequences of (1). Given $\mathcal{S}$ and the BWT encoding table illustrated in Fig. 3, for each $j \in \{1, \ldots, C\}$ the encoder can immediately determine the boundary $T_{j+1}$ between distribution $p_j$ and distribution $p_{j+1}$ in this model. The algorithm achieves universal coding performance by explicitly describing the boundaries to the decoder and then independently encoding the subsequences. A variety of codes may be used in coding the individual subsequence of $W^n$. The algorithm used here is an arithmetic code [44] with a Krichevsky–Trofimov (KT) [25] probability model. The elegance, simplicity, and convergence properties of this sequential code motivate the choice.

Given a probability model $P_c(x^n)$ for symbols $x_1, \ldots, x_n$, the arithmetic code [44] guarantees a description length $\ell_n(x^n)$ such that

$$\ell_n(x^n) < \log(1/P_c(x^n)) + 2$$

for all possible $x^n$. The KT estimate $P_c(x^n)$ uses $|\mathcal{X}|$ counters $\{r(x): x \in \mathcal{X}\}$. Let $r_i(x)$ denote the value of counter $r(x)$ after seeing the $i$th symbol in $x^n$. Set $r_0(x) = 1/2$ for each $x \in \mathcal{X}$. Then at each time $i \geq 1$, increment the counter corresponding to symbol $x_i$, leaving the remaining counters unchanged. Thus for any $i \geq 0$

$$r_i(x) = \frac{1}{2} + \sum_{k=1}^{i} 1(x_k = x)$$

where $1(\cdot)$ is the indicator function. The KT probability estimate equals

$$P_c(x^n) = \prod_{k=1}^{n} \frac{r_{k-1}(x_k)}{\sum\limits_{x \in \mathcal{X}} r_{k-1}(x)}.$$

This probability is calculated sequentially and used in a sequential arithmetic code. The sequential probability updates are calculated as

$$P_c(x^i) = P_c(x^{i-1}) \frac{r_{i-1}(x_i)}{\sum\limits_{x \in \mathcal{X}} r_{i-1}(x)}$$

where $P_c(x^0)$ (the probability of the length zero data sequence) equals 1 by definition.

By [25], the resulting description length is bounded as

$$\begin{aligned} nH(\hat{\theta}(x^n)) &+ \frac{|\mathcal{X}|-1}{2}\log n - c \\ &\le -\log P_c(x^n) \le nH(\hat{\theta}(x^n)) + \frac{|\mathcal{X}|-1}{2}\log n + c \qquad (2) \end{aligned}$$

where $H(\hat{\theta}(x^n))$ is the first-order entropy of the empirical distribution of $x^n \in \mathcal{X}^n$. For any $X^n$ drawn from i.i.d. distribution $P_\theta$, taking an expectation gives

$$E_\theta H(\hat{\theta}(X^n)) - H_\theta(X) \le 0$$

by Jensen's inequality, and thus the redundancy of the KT code on i.i.d. symbols from distribution $\theta$ is bounded as

$$\Delta_n(\theta) \le \frac{|\mathcal{X}|-1}{2}\frac{\log n}{n} + O\left(\frac{1}{n}\right).$$

*Theorem 2:* The arithmetic code that uses an independent KT distribution on each subsequence of the BWT of the reversed data sequence yields description length

$$\frac{\ell_n(x^n)}{n} \le H_{\hat{\theta}(x^n,\mathcal{S})}(\mathcal{X}) + \frac{|\mathcal{S}|(|\mathcal{X}|+1)}{2}\frac{\log n}{n} + O\left(\frac{1}{n}\right).$$

The resulting code is weakly minimax universal over the class of finite-memory sources and strongly minimax universal over the class of finite-memory sources with state space size $|\mathcal{S}| \le K$ for some constant $K \ge 1$. Given a finite-memory source with alphabet $\mathcal{X}$ and known state space $\mathcal{S}$, the redundancy associated with this code is bounded as

$$\Delta_n(\theta) \le \frac{|\mathcal{S}|(|\mathcal{X}|+1)}{2}\frac{\log n}{n} + O\left(\frac{1}{n}\right)$$

bits per symbol.

When $\mathcal{S}$ is unknown but a bound $M \le L$ on the source's memory constraint is known, then the application of the same algorithm with state-space estimate $\hat{\mathcal{S}} = \mathcal{X}^L$ gives

$$\Delta_n(\theta) \le \frac{|\mathcal{X}|^L(|\mathcal{X}|+1)}{2}\frac{\log n}{n} + O\left(\frac{1}{n}\right)$$

bits per symbol for all $\theta$ in the class of finite-memory sources.

*Proof:* Given $x^n$, let

$$(z^{n+1}, u) = \text{BWT}\,(\mathcal{R}(x^n)\S)$$

and $w^n = (z_1, \ldots, z_{u-1}, z_{u+1}, \ldots, z_{n+1})$. Denote the boundary points from (1) by $T(w^n) = (T_1, \ldots, T_{C+1})$. Since these boundary points correspond to regions of fixed state or context in the BWT encoding table and the encoder knows the contexts, the encoder also knows $T(w^n)$.

The encoder begins by describing the index value $u$ and the lengths of the $C$ subsequences.[3] Rather than describing index $u$ by its natural binary description, the encoder includes the description of $u$ with the description of the subsequence lengths using the following two-stage code. In the first stage, the encoder passes through $z^{n+1}$ in order, sending a 00 for each subsequence of $w^n$ that has length one, a 11 for each subsequence of $w^n$ that has length greater than one, and a 10 for the (length-one) subsequence $z_u = \S$ of $z^{n+1}$. These descriptions are followed by 01, indicating that no more subsequences need be described. (This inclusion is necessary since the decoder does not generally know the value of $C$.) Next, the encoder describes the lengths of all but the last subsequence receiving a 11 description; for simplicity, each of these descriptions uses the natural $\lfloor \log n \rfloor + 1$ bit expansion of the desired subsequence length. (The last subsequence length need not be described since the sum of all subsequence lengths must equal the sequence length $n$.) By (1), the resulting description requires no more than $2(|\mathcal{S}|+M+2)+(|\mathcal{S}|-1)(\log n+1)$ bits to describe both the transition points between the $C$ subsequences and the row index $u = \text{BWT}_{\mathbb{N}}(\mathcal{R}(x^n)\S)$.[4]

The encoder follows its description of the subsequence division points with an independent description of each subsequence. Let $\boldsymbol{w}_j = w_{T_j}, w_{T_j+1}, \ldots, w_{T_{j+1}-1}$ denote the $j$th subsequence. Then by (2), the per-symbol code length may be bounded as

$$\begin{aligned} &\frac{\ell_n(x^n)}{n} \\ &\quad\le \frac{1}{n}\sum_{j=1}^{C}\left[|\boldsymbol{w}_j| H(\hat{\theta}(\boldsymbol{w}_j)) + \frac{|\mathcal{X}|-1}{2}\log|\boldsymbol{w}_j| + c\right] \\ &\qquad + (|\mathcal{S}|-1)\frac{\log n}{n} \\ &\quad\le H_{\hat{\theta}(x^n,\mathcal{S})}(\mathcal{X}) + \left(\frac{|\mathcal{S}|(|\mathcal{X}|+1)}{2} - 1\right)\frac{\log n}{n} + O\left(\frac{1}{n}\right) \end{aligned}$$

since $|\boldsymbol{w}_j| > 1$ for at most $|\mathcal{S}|$ values of $j$. The resulting redundancy is

$$\Delta_n(\theta) \le \left(\frac{|\mathcal{S}|(|\mathcal{X}|+1)}{2} - 1\right)\frac{\log n}{n} + O\left(\frac{1}{n}\right)$$

by Jensen's inequality and the concavity of $-x\log x$. □

The rate of convergence described in Theorem 2 differs from Rissanen's optimal rate of convergence by a constant factor of $(|\mathcal{X}|+1)/(|\mathcal{X}|-1)$. (The bound given inside the proof is slightly tighter.) For very small $|\mathcal{X}|$ (e.g., a binary source), this factor grows as large as 3. For text compression using the ASCII alphabet, $|\mathcal{X}|=128$, giving a factor bounded by $129/128=1.008$

[3] In practice, the encoder would likely intersperse the descriptions of the lengths of the subsequence with the descriptions of the subsequences themselves. This modification affects the ordering of the bit stream but not its content or length.

[4] More sophisticated (and more complex) boundary point encoders would exploit the relationships between these boundary points, which are not independent. The discussion used here sticks to the simplest approach.

and nearly optimal performance. The factor shrinks to 1 for large alphabets.

The suboptimal constant in the rate of convergence results from the algorithm's inefficiencies. Using a code matched to the statistics of $W^n$ rather than a code matched to the statistics of $Z^{n+1}$ or taking advantage of that fact that there exist data sequences $w^n \in \mathcal{X}^n$ for which the inverse BWT is undefined, should give better performance.

To its credit, this algorithm achieves very good performance while remaining both conceptually and computationally simple. Further, the algorithmic complexity does not grow with $|\mathcal{S}|$ or $M$. In particular, while the above code tracks as many as $|\mathcal{S}|+M$ distributions, only one distribution is tracked at a time, and thus the memory and computation requirements for the codes are independent of $|\mathcal{S}|$ and $M$. Since the space and time complexity of arithmetic coding and the sequential calculation of the KT estimate are linear in the sequence length $n$, the resulting code is $O(n)$ in memory and computation.

### *D. A Finite Memory Code*

Explicit knowledge or calculation of $\mathcal{S}$ need not be part of BWT-based universal codes. The algorithms that follow code $Z^{n+1}$ by employing strategies that can deal with the piecewise-constant nature of its statistics. While many such algorithms exist, this work treats only three examples, chosen for their simplicity and their relationship with earlier codes.

The first algorithm results from a very simple observation about $W^n$. The bound $C \leq |\mathcal{S}| + M$ on the number of distinct distributions in (1) does not grow with the sequence length $n$. Further, for large $n$, the length of the subsequence for prefix $s$ should approximate $\pi(s)n$ by the law of large numbers. Given "window" length $\omega(n)$, suppose that the encoder breaks the data sequence into consecutive subsequences of length $\omega(n)$ and uses an independent KT code on each.[5] The window length $\omega(n)$ must grow with $n$ so that the per-symbol redundancy on each length-$\omega(n)$ sequence goes to zero. The growth should be slow, however, so that the fraction of windows containing two or more distributions is small. The following theorem bounds the redundancy achieved using the optimal $\omega(n)$. This $\omega(n)$ is instructive for designing "forgetting" mechanisms in practical codes.

*Theorem 3:* The arithmetic code that codes the BWT of the reversed data string using the KT distribution with a fixed-length finite memory yields per-symbol description length

$$\frac{\ell_n(x^n)}{n} \leq H_{\hat{\theta}(x^n,\mathcal{S})}(\mathcal{X}) + \sqrt{(|\mathcal{S}|-1)(|\mathcal{X}|-1)\log|\mathcal{X}|} \cdot \sqrt{\frac{\log n}{n}} + O\left(\frac{\log\log n}{\sqrt{n\log n}}\right)$$

bits per symbol for each $x^n \in \mathcal{X}^n$. The resulting code is strongly minimax universal on the class of finite-memory sources with $|\mathcal{S}| \leq K$ and weakly minimax universal on the class of finite-memory sources. Given a finite-memory source with unknown state space $\mathcal{S}$ and unknown memory constraint $M$, the redundancy is bounded as

$$\Delta_n(\theta) \leq \sqrt{(|\mathcal{S}|-1)(|\mathcal{X}|-1)\log|\mathcal{X}|}\sqrt{\frac{\log n}{n}} + O\left(\frac{\log\log n}{\sqrt{n\log n}}\right)$$

bits per symbol for all $\theta$ in the given class if the choice of the memory length is allowed to depend on $|\mathcal{S}|$. When the memory length cannot depend on $|\mathcal{S}|$, the redundancy equation varies by a constant factor, again giving $\Delta_n(\theta) \leq O(\sqrt{\log n/n})$.

*Proof:* Given a finite-memory source model, fix $x^n \in \mathcal{X}^n$ and again let

$$(z^{n+1}, u) = \mathrm{BWT}(\mathcal{R}(x^n)\S)$$

and $w^n = (z_1, \ldots, z_{u-1}, z_{u+1}, \ldots, z_{n+1})$. The encoder breaks the data sequence $w^n$ into subsequences $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{\lceil n/\omega(n)\rceil}$, where

$$\boldsymbol{v}_i = \begin{cases} w_{(i-1)\omega(n)+1}^{i\omega(n)}, & 1 \leq i < \lceil n/\omega(n)\rceil \\ w_{(i-1)\omega(n)+1}^{n}, & i = \lceil n/\omega(n)\rceil. \end{cases}$$

The encoder uses an independent KT probability model for each subsequence. Thus, after each $\omega(n)$ samples, the counts $r(x)$ for all $x \in \mathcal{X}$ reset to $1/2$ and the coding algorithm begins again. Recall from (1) that the data sequence $w^n$ breaks into $C$ component subsequences with $C \leq |\mathcal{S}| + M$ and at most $|\mathcal{S}|$ subsequences of length greater than one. Thus, for any window length $\omega(n) > 1$, at most $|\mathcal{S}| - 1$ code sequences $\boldsymbol{v}_i$ contain samples from more than one distribution. For any such $\boldsymbol{v}_i$, the description length for the entire window of symbols is

$$\begin{aligned} \ell_{|\boldsymbol{v}_i|}(\boldsymbol{v}_i) &\leq |\boldsymbol{v}_i| H(\hat{\theta}(\boldsymbol{v}_i)) + \frac{|\mathcal{X}|-1}{2}\log|\boldsymbol{v}_i| + c \\ &\leq \omega(n)\log|\mathcal{X}| + \frac{|\mathcal{X}|-1}{2}\log\omega(n) + c. \end{aligned}$$

The total per-symbol description length is bounded as

$$\begin{aligned} \frac{\ell_n(x^n)}{n} \leq& \sum_{i=1}^{\lceil n/\omega(n)\rceil} \frac{1}{n}\left[|\boldsymbol{v}_i| H(\hat{\theta}(\boldsymbol{v}_i)) + \frac{|\mathcal{X}|-1}{2}\log|\boldsymbol{v}_i| + c\right] \\ &+ \frac{\log(n+1)+1}{n} \\ \leq& \sum_{j=1}^{C} \frac{|\boldsymbol{w}_j|}{n} H(\hat{\theta}(\boldsymbol{w}_j)) + \frac{(|\mathcal{S}|-1)\omega(n)\log|\mathcal{X}|}{n} \\ &+ \frac{|\mathcal{X}|-1}{2}\frac{\log\omega(n)}{\omega(n)} + O\left(\frac{1}{\omega(n)}\right) \\ =& H_{\hat{\theta}(x^n,\mathcal{S})}(\mathcal{X}) + (|\mathcal{S}|-1)\log|\mathcal{X}|\frac{\omega(n)}{n} \\ &+ \frac{|\mathcal{X}|-1}{2}\frac{\log\omega(n)}{\omega(n)} + O\left(\frac{1}{\omega(n)}\right) \end{aligned}$$

by (2) and Jensen's inequality if $1/\omega(n)$ decays more slowly than $\log n/n$. Choosing

$$\omega(n) = \frac{1}{2}\sqrt{\frac{|\mathcal{X}|-1}{(|\mathcal{S}|-1)\log|\mathcal{X}|}}\sqrt{n\log n} \tag{3}$$

[5] An alternative to the above finite-memory approach would be a sliding-window approach.

gives

$$\begin{aligned}\frac{\ell_n(x^n)}{n} &\le H_{\hat{\theta}(x^n,\mathcal{S})}(\mathcal{X}) + \frac{\sqrt{(|\mathcal{S}|-1)(|\mathcal{X}|-1)\log|\mathcal{X}|}}{2}\\ &\quad\cdot\left(\sqrt{\frac{\log n}{n}} + \frac{\log(n\log n)+c}{\sqrt{n\log n}}\right)\\ &= H_{\hat{\theta}(x^n,\mathcal{S})}(\mathcal{X})\\ &\quad + \sqrt{(|\mathcal{S}|-1)(|\mathcal{X}|-1)\log|\mathcal{X}|}\sqrt{\frac{\log n}{n}}\\ &\quad + O\left(\frac{\log\log n}{\sqrt{n\log n}}\right).\end{aligned}$$

Thus,

$$\Delta_n(\theta) = \sqrt{(|\mathcal{S}|-1)(|\mathcal{X}|-1)\log|\mathcal{X}|}\sqrt{\frac{\log n}{n}} + O\left(\frac{\log\log n}{\sqrt{n\log n}}\right)$$

for all $\theta$ in the given class. While the optimal window length (3) depends on $|\mathcal{S}|$, setting $\omega(n) = \sqrt{n\log n}$ maintains $\Delta_n(\theta) = O(\sqrt{\log n/n})$. □

The above analysis gives worst case results. For many sources such as text, neighboring distributions in the BWT output often have similar contexts and thus similar statistics, yielding good performance even in coding regions overlapping more than one distribution.

Since this algorithm, like its predecessor, relies only on the BWT and arithmetic coding, the complexity and memory requirements of the code are again $O(n)$.

### *E. Coding for Piecewise-Constant Parameters*

Next, consider coding the BWT's output using a code designed for data sequences with piecewise-constant parameters.

In [45], Merhav considers the problem of universal lossless coding for sources with piecewise-constant parameters, considering both upper and lower bounds on coding performance. The achievability argument given in [45] gives a sequential code yielding

$$\begin{aligned}\frac{\ell_n(x^n)}{n} \le \min_{1\le T\le n+1}&\left[\frac{T-1}{n}H(\hat{\theta}(x_1^{T-1})) + \frac{n-T+1}{n}\right.\\ &\left.\cdot H(\hat{\theta}(x_T^n))\right] + |\mathcal{X}|\frac{\log n}{n} + O\left(\frac{1}{n}\right)\end{aligned}$$

bits per symbol for any $x^n \in \mathcal{X}^n$ and suggests that the result generalizes from two subsequences to $C$ subsequences to give

$$\begin{aligned}\frac{\ell_n(x^n)}{n} \le \min_{T^C}&\sum_{j=1}^{C}\frac{T_{j+1}-T_j}{n}H\left(\hat{\theta}\left(x_{T_j}^{T_{j+1}-1}\right)\right)\\ &+\left(\frac{C(|\mathcal{X}|+1)}{2}-1\right)\frac{\log n}{n}+O\left(\frac{1}{n}\right).\end{aligned}$$

Unfortunately, the algorithmic complexity grows exponentially with $n$ for unknown $C$ [46].

In [46], Willems suggests two alternative sequential algorithms. The algorithms differ in their performances and their complexities, giving

$$\begin{aligned}\frac{\ell_n^{(1)}(x^n)}{n} \le \min&\left[\sum_{j=1}^{C}\frac{T_{j+1}-T_j}{n}H\left(\hat{\theta}\left(x_{T_j}^{T_{j+1}-1}\right)\right)\right.\\ &\left.+\left(\frac{C(|\mathcal{X}|+1)}{2}-\frac{1}{2}\right)\frac{\log n}{n}+O\left(\frac{1}{n}\right)\right]\\ \frac{\ell_n^{(2)}(x^n)}{n} \le \min&\left[\sum_{j=1}^{C}\frac{T_{j+1}-T_j}{n}H\left(\hat{\theta}\left(x_{T_j}^{T_{j+1}-1}\right)\right)\right.\\ &\left.+\left(\frac{C(|\mathcal{X}|+2)}{2}-1\right)\frac{\log n}{n}+O\left(\frac{1}{n}\right)\right]\end{aligned}$$

where the minima are both taken with respect to the choice of $C$ and $T^{C+1}$ with $1 = T_1 \le \cdots \le T_{C+1} = n+1$. The space complexities of the two algorithms grow more slowly than the time complexities, which are $O(n^3)$ and $O(n^2)$, respectively.

In [47], Shamir and Merhav describe an algorithm giving

$$\begin{aligned}&\frac{\ell_n^{(1)}(x^n)}{n}\\ &\le \min\left[\sum_{j=1}^{C}\frac{T_{j+1}-T_j}{n}H\left(\hat{\theta}\left(x_{T_j}^{T_{j+1}-1}\right)\right)\right.\\ &\qquad\left.+\left(\frac{C(|\mathcal{X}|+1)}{2}-1\right)\frac{\log n}{n}\right]+O\left(\frac{\log\log n}{n}\right).\end{aligned}$$

The space and time complexity of their algorithm is $O(n^2)$.

Even though the results in Merhav [45], Willems [46], and Shamir and Merhav [47] are for p.i.i.d. sources, it is easy to check that if (1) holds, then all of their results go through. This yields Theorem 4. In each case, the redundancy of the earlier algorithms is increased by $\log n/n$ due to the need to describe BWT$_{\mathbb{N}}(\mathcal{R}(X^n)\,\S)$.

*Theorem 4:* Coding the BWT of the reversed data string using

- an algorithm achieving Merhav's bound yields the rate

$$\frac{\ell_n(x^n)}{n} \le H_{\hat{\theta}(x^n,\mathcal{S})}(\mathcal{X}) + \frac{|\mathcal{S}|(|\mathcal{X}|+1)}{2}\frac{\log n}{n} + O\left(\frac{1}{n}\right)$$

  and, on a finite-memory source with unknown state space $\mathcal{S}$ and unknown memory constraint $M$, redundancy

$$\Delta_n(\theta) \le \frac{|\mathcal{S}|(|\mathcal{X}|+1)}{2}\frac{\log n}{n} + O\left(\frac{1}{n}\right);$$

- Willems' $O(n^3)$ algorithm yields the rate

$$\frac{\ell_n(x^n)}{n} \le H_{\hat{\theta}(x^n,\mathcal{S})}(\mathcal{X}) + \left(\frac{|\mathcal{S}|(|\mathcal{X}|+1)}{2}+\frac{1}{2}\right)\frac{\log n}{n} + O\left(\frac{1}{n}\right)$$

  and, on a finite-memory source with unknown state space $\mathcal{S}$ and unknown memory constraint $M$, redundancy

$$\Delta_n(\theta) \le \left(\frac{|\mathcal{S}|(|\mathcal{X}|+1)}{2}+\frac{1}{2}\right)\frac{\log n}{n} + O\left(\frac{1}{n}\right);$$

- Willems' $O(n^2)$ algorithm yields the rate

$$\frac{\ell_n(x^n)}{n} \le H_{\hat{\theta}(x^n,\mathcal{S})}(\mathcal{X}) + \left(\frac{|\mathcal{S}|(|\mathcal{X}|+2)}{2}\right)\frac{\log n}{n} + O\left(\frac{1}{n}\right)$$

and, on a finite-memory source with unknown state space $\mathcal{S}$ and unknown memory constraint $M$, redundancy

$$\Delta_n(\theta) \le \left(\frac{|\mathcal{S}|(|\mathcal{X}|+2)}{2}\right)\frac{\log n}{n} + O\left(\frac{1}{n}\right);$$

- Shamir and Merhav's $O(n^2)$ algorithm yields the rate

$$\frac{\ell_n(x^n)}{n} \le H_{\hat{\theta}(x^n,\mathcal{S})}(\mathcal{X}) + \left(\frac{|\mathcal{S}|(|\mathcal{X}|+1)}{2}\right)\frac{\log n}{n} + O\left(\frac{\log\log n}{n}\right)$$

and, on a finite-memory source with unknown state space $\mathcal{S}$ and unknown memory constraint $M$, redundancy

$$\Delta_n(\theta) \le \left(\frac{|\mathcal{S}|(|\mathcal{X}|+1)}{2}\right)\frac{\log n}{n} + O\left(\frac{\log\log n}{n}\right).$$

These algorithms are strongly minimax universal on the class of finite-memory sources with $|\mathcal{S}| \le K$ and weakly minimax universal on the class of finite-memory sources.

### *F. Stationary Ergodic Sources*

The approach taken in the following discussion is to model an arbitrary stationary ergodic source using a Markov model with memory $m-1$. As $m$ grows without bound, the accuracy of the model in approximating the true source statistics becomes arbitrarily tight. As a result, the performance of the BWT-based source code that uses a finite-memory model designed for state space $|\mathcal{X}|^{m-1}$ converges to the optimal coding performance for the source in operation. The following discussion treats expected performance results only since the individual sequence results given previously require no source model assumption.

Recall that while the definition for universality relies on the redundancy $\delta_n(\theta)$, discussions in previous sections bound the rate of convergence of $\Delta_n(\theta)$. This choice made the analysis simpler and caused no harm since $\Delta_n(\theta) - \delta_n(\theta) = O(1/n)$ for finite memory sources by Lemma 4. Unfortunately, the same does not hold for the more general class of stationary ergodic sources. As a result, the focus in this subsection turns to a third measure of redundancy, here denoted by $\overline{\delta}_n(\theta)$ and defined as

$$\overline{\delta}_n(\theta) = \frac{1}{n}\,E_\theta \ell_n(X^n) - H_\theta(X_n|X^{n-1}).$$

The stationarity of the source and the fact that conditioning reduces entropy give

$$\begin{aligned}\delta_n(\theta) &= \frac{1}{n}\,E_\theta\ell_n(X^n) - \frac{1}{n}\sum_{i=1}^{n} H_\theta(X_i|X^{i-1})\\ &\le \frac{1}{n}\,E_\theta\ell_n(X^n) - \frac{1}{n}\sum_{i=1}^{n} H_\theta(X_i|X_{i-n+1}^{i-1})\\ &= \frac{1}{n}\,E_\theta\ell_n(X^n) - H_\theta(X_n|X^{n-1})\\ &= \overline{\delta}_n(\theta)\\ &= \frac{1}{n}\,E_\theta\ell_n(X^n)\\ &\quad - \lim_{n'\to\infty}\left[\frac{1}{n'}\,H_\theta(X^n) + \frac{n'-n}{n'}\,H_\theta(X_n|X^{n-1})\right]\\ &\le \frac{1}{n}\,E_\theta\ell_n(X^n) - \lim_{n'\to\infty}\frac{1}{n'}\,H_\theta(X^{n'})\\ &= \Delta_n(\theta).\end{aligned}$$

Thus,

$$\delta_n(\theta) \le \overline{\delta}_n(\theta) \le \Delta_n(\theta)$$

for all $\theta$ in the given class of stationary ergodic sources. Now for any integer $m$, let

$$\overline{\delta}_{(n,\,m)}(\theta) = \frac{1}{n}\,E_\theta\ell_n(X^n) - H_\theta(X_m|X^{m-1}).$$

This $(n,\,m)$th-order redundancy bounds the difference between the per-symbol expected description length for sequence length $n$ and a lower bound on the optimal per-symbol description length for sequence length $m < n$ on the same distribution. The difference between $\overline{\delta}_n(\theta)$ and $\overline{\delta}_{n,\,m}(\theta)$ equals

$$\begin{aligned}\overline{\delta}_n(\theta) - \overline{\delta}_{(n,\,m)}(\theta) &= H_\theta(X_m|X^{m-1}) - H_\theta(X_n|X^{n-1})\\ &\le H_\theta(X_m|X^{m-1}) - H_\theta(\mathcal{X})\end{aligned}$$

which does not vary with the algorithm in operation.

In [48], Shields proves that for any function $\rho(n)$ such that $\lim_{n\to\infty}\rho(n) = 0$, there exists a source $\theta$ in the class of stationary ergodic sources such that

$$\limsup_{n\to\infty}\frac{\delta_n(\theta)}{\rho(n)} = \infty.$$

Thus, there do not exist general bounds on $\delta_n(\theta)$ (or, consequently, $\overline{\delta}_n(\theta)$) for the class of stationary ergodic sources. Nonetheless, there *do* exist bounds on $\overline{\delta}_{(n,\,m)}(\theta)$, and the derivation of such a bound for the BWT-based source codes discussed in this work appears in Theorem 5. Several corollaries following the theorem discuss the consequences of this rate of convergence result for different subsets of the class of stationary ergodic sources. These subsets are characterized by bounds on the rate of convergence of $H_\theta(X_m|X^{m-1}) - H_\theta(X_n|X^{n-1})$ when $m \ll n$ or the rate of convergence of $H_\theta(X_m|X^{m-1}) - H_\theta(\mathcal{X})$ for all $\theta$ in the given class. Any of the algorithms described previously may be used to effectively code sources from stationary ergodic sources. The simplest choice is the known state-space algorithm from Theorem 2.

*Theorem 5:* Given a stationary ergodic source, applying the known state-space algorithm of Theorem 2 with state-space model $\mathcal{X}^{m-1}$ achieves an $(n,\,m)$th-order redundancy $\overline{\delta}_{(n,\,m)}(\theta)$ bounded as

$$\overline{\delta}_{(n,\,m)}(\theta) \le \frac{|\mathcal{X}|^{m-1}(|\mathcal{X}|+1)}{2}\,\frac{\log n}{n} + O\left(\frac{|\mathcal{X}|^{m-1}}{n}\right)$$

bits per symbol for all $\theta$ in the class of stationary ergodic sources. Letting $m$ and $n$ grow without bound yields performance approaching the source's entropy rate provided that $m$ grows more slowly than $\log(n/\log n)/\log|\mathcal{X}|$. Under these conditions, the BWT-based source code is weakly minimax universal on the class of stationary ergodic sources.

*Proof:* Consider a data sequence $X^n$ drawn from an arbitrary stationary ergodic source $P_\theta$. Again let

$$(Z^{n+1}, U) = \text{BWT}\,(\mathcal{R}(X^n)\S)$$

and $W^n = (Z_1, \ldots, Z_{U-1}, Z_{U+1}, \ldots, Z_{n+1})$. While (1) does not apply when the source violates the finite-memory condition, a similar property applies. In particular, for the $j$th context of length $m-1$, the BWT aligns all symbols following that context into a contiguous subsequence $\boldsymbol{W}_j$ of $W^n$. Since the encoder has access to all of the information in the BWT encoding table, the encoder can determine the start and stop positions for each of these contexts and can describe them to the decoder. Applying the argument of Theorem 2 with $\hat{\mathcal{S}} = \mathcal{X}^{m-1}$ gives

$$\begin{aligned}\frac{E_\theta \ell_n(x^n)}{n} &\le \frac{1}{n}\, E_\theta \sum_{j=1}^{C} \left[|\boldsymbol{W}_j| H(\hat{\theta}(\boldsymbol{W}_j)) + \frac{|\mathcal{X}|-1}{2} \log|\boldsymbol{W}_j| + c\right] \\ &\quad + (|\mathcal{X}|^{m-1} - 1)\,\frac{\log n}{n} \\ &\le H(X_m|X^{m-1}) + \frac{|\mathcal{X}|^{m-1}(|\mathcal{X}|+1) - 2}{2}\,\frac{\log n}{n} \\ &\quad + O\left(\frac{|\mathcal{X}|^{m-1}}{n}\right)\end{aligned}$$

yielding $(n, m)$th-order redundancy

$$\overline{\delta}_{(n,m)}(\theta) \le \frac{|\mathcal{X}|^{m-1}(|\mathcal{X}|+1)-2}{2}\,\frac{\log n}{n} + O\left(\frac{|\mathcal{X}|^{m-1}}{n}\right). \quad \square$$

Obtaining a bound on the rate of convergence of the universal lossless code requires knowledge of either the rate of convergence of $H(X_m|X^{m-1})$ to $H(X_n|X^{n-1})$ or the rate of convergence of $H(X_m|X^{m-1})$ to $H(\mathcal{X})$ as a function of $m$. The optimal growth rate for $m$ as a function of $n$ depends on these rates of convergence, as the following examples illustrate.

*Corollary 1:* Consider the class of stationary ergodic sources for which there exists an $M \ge 0$ such that

$$H(X_m|X^{m-1}) = H(\mathcal{X}), \qquad \text{for all } m \ge M+1.$$

Using $m = M+1$ in the algorithm described in Theorem 5, then for all $n$ sufficiently large, $H(X_n|X^{n-1}) = H(\mathcal{X})$, and thus

$$\Delta_n(\theta) = \overline{\delta}_n(\theta) \le \frac{|\mathcal{X}|^M(|\mathcal{X}|+1)}{2}\,\frac{\log n}{n} + O\left(\frac{1}{n}\right).$$

This is the case for finite-memory sources.

*Corollary 2:* Consider the class of stationary ergodic sources for which $H(X_m|X^{m-1}) - H(\mathcal{X}) \le c/m$ for constant $c > 0$ and $m$ sufficiently large. Given any $\epsilon > 0$, allowing $m$ to grow as

$$m = \frac{1}{\log|\mathcal{X}|} \log\left(\frac{2|\mathcal{X}|cn}{(|\mathcal{X}|+1)\log^2 n}\right)$$

gives a rate of convergence

$$\delta_n(\theta) \le \overline{\delta}_n(\theta) \le c(1 + (1+\epsilon)\log|\mathcal{X}|)\,\frac{1}{\log n}.$$

*Corollary 3:* Consider the class of stationary ergodic sources with $H(X_m|X^{m-1}) - H(\mathcal{X}) \le c\log m/m$ for constant $c > 0$ and $m$ sufficiently large. Given any $\epsilon > 0$, allowing $m$ to grow as

$$m = \frac{1}{\log|\mathcal{X}|} \log\left(\frac{2|\mathcal{X}|cn\log\log n}{(|\mathcal{X}|+1)\log^2 n}\right)$$

gives a rate of convergence

$$\delta_n(\theta) \le \overline{\delta}_n(\theta) \le c(1 + (1+\epsilon)\log|\mathcal{X}|)\,\frac{\log\log n}{\log n}.$$

## VI. BWT Output Statistics: Asymptotic Properties

Section V discussed the effect of the BWT on finite-memory sources, drawing a connection between the distribution of the BWT output and the family of p.i.i.d. distributions. While this connection is sufficient for all of the coding results described in Section V, it does not fully characterize the statistics of the BWT output. Such a characterization is the topic of this section. The approach taken here deviates from that of previous sections by discontinuing the use of the end-of-file symbol.

Consider an arbitrary data sequence $x^n \in \mathcal{X}^n$. Let $y^n = \mathcal{R}(x^n)$ and

$$(z^n, u) = \text{BWT}\,(y^n) = \text{BWT}\,(\mathcal{R}(x^n)).$$

Notice that while BWT: $\mathcal{X}^n \to \mathcal{X}^n \times \{1, \ldots, n\}$ is a one-to-one mapping, $\text{BWT}_{\mathcal{X}}$: $\mathcal{X}^n \to \mathcal{X}^n$ is a many-to-one mapping. More precisely,

$$\text{BWT}_{\mathcal{X}}(\mathcal{R}(\hat{x}^n)) = \text{BWT}_{\mathcal{X}}(\mathcal{R}(x^n))$$

if and only if $\hat{x}^n$ is a cyclic shift of $x^n$. To avoid complications in the notation, fix the value of $n$ and—for notational purposes only—treat the data sequence as if it were periodic with period $n$, using $x_{i+n}$ and $x_{i-n}$ as alternative names for $x_i$ for each $i \in \{1, \ldots, n\}$. Using this notation, the $n$ cyclic shifts of $x^n$ are $x_{t+1}^{t+n}$, $t \in \{0, \ldots, n-1\}$.

For any distribution $P_{X^n}$ on alphabet $\mathcal{X}^n$, let $\mathcal{I}(P_{X^n}, x^n)$ be defined as

$$\begin{aligned}\mathcal{I}(P_{X^n}, x^n) = \Big\{0 \le t \le n-1 \colon P_{X^n}(x_{t+1}^{t+n}) > 0 \\ \wedge x_{t+1}^{t+n} \ne x_{j+1}^{j+n}\ \forall\, 0 \le j < t\Big\}.\end{aligned}$$

Thus, $\{x_{t+1}^{t+n} \colon t \in \mathcal{I}(P_{X^n}, x^n)\}$ gives all distinct cyclic shifts of $x^n$ with $P_{X^n}(x_{t+1}^{t+n}) > 0$. For example, if $P_{X^n}$ is the uniform distribution on $\{0, 1\}^n$, then

$$\mathcal{I}(P_{X^n}, (0, 0, \ldots, 0)) = \{0\}$$

$$\mathcal{I}(P_{X^n}, (0, 1, 0, 1, \ldots, 0, 1)) = \{0, 1\}$$

and

$$\mathcal{I}(P_{X^n}, x^n) = \{0, \ldots, n-1\}$$

for any nonperiodic $x^n \in \{0, 1\}^n$.

Let

$$\mathcal{Z}(n) = \{\mathrm{BWT}_{\mathcal{X}}(x^n) \colon x^n \in \mathcal{X}^n\}.$$

For any random sequence $X^n$ drawn according to distribution $P_{X^n}$, let

$$Z^n = \mathrm{BWT}_{\mathcal{X}}(Y^n) = \mathrm{BWT}_{\mathcal{X}}(\mathcal{R}(X^n)).$$

Then, for any $z^n \in \mathcal{Z}(n)$, $x^n = \mathcal{R}(\mathrm{BWT}^{-1}(z^n, 1))$ implies that the probability $P_{Z^n}(z^n)$ that $Z^n$ equals $z^n$ is

$$P_{Z^n}(z^n) = \sum_{t \in \mathcal{I}(P_{X^n}, x^n)} P_{X^n}(x_{t+1}^{t+n}). \tag{4}$$

For any $z^n \notin \mathcal{Z}(n)$, $P_{Z^n}(z^n) = 0$.

If $P_{X^n}$ describes an i.i.d. source, then

$$P_{X^n}(x_{t+1}^{t+n}) = P_{X^n}(x_1^n) = \prod_{i=1}^{n} p(x_i)$$

for all $t \in \{0, \ldots, n-1\}$. In this case $z^n \in \mathcal{Z}(n)$ and $x^n = \mathcal{R}(\mathrm{BWT}^{-1}(z^n, 1))$ together imply

$$P_{Z^n}(z^n) = |\mathcal{I}(P_{X^n}, x^n)| P_{X^n}(x^n)$$

by (4). Since

$$P_{X^n}(x^n) = \prod_{i=1}^{n} p(x_i) = \prod_{i=1}^{n} p(z_i)$$

and $|\mathcal{I}(P_{X^n}, x^n)| \le n$

$$\begin{aligned} \frac{1}{n} D(P_{Z^n} \| P_{X^n}) &= \frac{1}{n} \sum_{z^n \in \mathcal{Z}(n)} P_{Z^n}(z^n) \log \frac{P_{Z^n}(z^n)}{P_{X^n}(z^n)} \\ &= \frac{1}{n} E_{X^n} \log |\mathcal{I}(P_{X^n}, X^n)| \\ &\le \frac{\log n}{n}. \end{aligned}$$

Thus, the distribution of the BWT output is asymptotically close to an i.i.d. distribution when the input is i.i.d. The bound on $D(P_{Z^n} \| P_{X^n})$ is tight to within an additive constant. For example, if $X^n$ is uniformly distributed on $\{0, 1\}^n$, then

$$\begin{aligned} \log n &- E_{X^n} \log |\mathcal{I}(P_{X^n}, X^n)| \\ &= E_{X^n} \log \frac{n}{|\mathcal{I}(P_{X^n}, X^n)|} \\ &= \sum_{d=2}^{n} \Pr(|\mathcal{I}(P_{X^n}, X^n)| = n/d) \log d \\ &\le \sum_{d=2}^{n} 2^{-n} 2^{n/2} \log d \le (n \log n) 2^{-n/2} \le c \end{aligned}$$

giving $D(P_{Z^n} \| P_{X^n}) \ge \log n - c$ for some constant $c$.

While the above analysis does not apply directly to sources with memory, a similar argument applies. If $P_{X^n}$ describes a stationary finite-memory source with memory $M$, state space $\mathcal{S}$, and next-state function $\mathrm{suf}(\cdot)$, then

$$P_{X^n}(x^n) = p(x^M) \prod_{i=M+1}^{n} p(x_i | \mathrm{suf}(x_{i-M}^{i-1}))$$

where $p(x^M)$ is the stationary distribution on $\mathcal{X}^M$ induced by the given finite-memory source. In this case, (4) implies that for any $z^n \in \mathcal{Z}(n)$ and $x^n = \mathcal{R}(\mathrm{BWT}^{-1}(z^n, 1))$

$$\begin{aligned} P_{Z^n}(z^n) = &\left[ \sum_{t \in \mathcal{I}(P_{X^n}, x^n)} \frac{p(x_{t+1}^{t+M})}{\left[ \prod_{k=t+1}^{t+M} p(x_k | \mathrm{suf}(x_{k-M}^{k-1})) \right]} \right] \\ &\cdot \left[ \prod_{i=1}^{n} p(x_i | \mathrm{suf}(x_{i-M}^{i-1})) \right]. \end{aligned}$$

If the state space $\mathcal{S} = \{s_1, s_2, \ldots, s_{|\mathcal{S}|}\}$ is ordered so that for all $j \in \{1, \ldots, |\mathcal{S}| - 1\}$, $\mathcal{R}(s_j)$ comes before $\mathcal{R}(s_{j+1})$ lexicographically, then BWT $(\mathcal{R}(X^n))$ places all rows beginning with $\mathcal{R}(s_j)$ before all rows beginning with $\mathcal{R}(s_{j+1})$ in the BWT encoding table. Thus,

$$\begin{aligned} P_{Z^n}(z^n) = &\left[ \sum_{t \in \mathcal{I}(P_{X^n}, x^n)} \frac{p(x_{t+1}^{t+M})}{p(x_{t+1}^{t+M} | x_{t-M+1}^{t})} \right] \\ &\cdot \left[ \prod_{j=1}^{|\mathcal{S}|} \prod_{i=T_j}^{T_{j+1}-1} p_j(z_i) \right] \end{aligned}$$

where, for each $j \in \{1, \ldots, |\mathcal{S}| + 1\}$

$$T_j = 1 + \sum_{i=1}^{n} \sum_{k=1}^{j-1} 1(\mathrm{suf}(x_{i-M}^{i-1}) = s_k)$$

and $p_j(\cdot) = p(\cdot | s_j)$.

While it seems appropriate to compare the distribution on the BWT output to a $|\mathcal{S}|$-p.i.i.d. distribution with the same random boundary points $\{T_j\}$, defining such a distribution is difficult. As a result, the analysis that follows treats the distance between the distribution of the BWT output and an $|\mathcal{S}|$-p.i.i.d. distribution with deterministic boundary points. The bounds on the divergence of the output distribution of the BWT and an $|\mathcal{S}|$-p.i.i.d. distribution with deterministic boundary points $\{T'_j\}$ are dominated by a term stemming from the difference between $\{T_j\}$ and $\{T'_j\}$.

Consider an $|\mathcal{S}|$-p.i.i.d. data sequence with distribution

$$Q^n(z^n) = \prod_{j=1}^{|\mathcal{S}|} \prod_{i=T'_j}^{T'_{j+1}-1} p_j(z_i)$$

where $p_j(\cdot)$ is defined as above, $T'_j = 1 + \lfloor C(j) n \rfloor$, and

$$C(j) = \sum_{k=1}^{j-1} \pi(s_k), \qquad \text{for all } j \in \{1, \ldots, |\mathcal{S}| + 1\}.$$

Lemma 1 bounds the normalized Kullback–Leibler distance between $P_{Z^n}$ and $Q^n$ under the assumption that $p_j(z) > 0$ for all $j \in \{1, \ldots, |\mathcal{S}|\}$ and all $z \in \mathcal{X}$. This assumption is removed in Lemma 2.

*Lemma 1:* Given a stationary finite-memory source for which $p(x|s) > 0$ for all $s \in \mathcal{S}$ and $x \in \mathcal{X}$, $Z^n = \text{BWT}\,(\mathcal{R}(X^n))$ implies that there exists some constant $c' > 0$ for which

$$\frac{1}{n} D(P_{Z^n} \| Q^n) \le \frac{c}{\sqrt{n}}.$$

*Proof:* Using $x^n = \mathcal{R}(\text{BWT}^{-1}(z^n, 1))$ for $z^n \in \mathcal{Z}(n)$

$$\begin{aligned}
&D(P_{Z^n} \| Q^n) \\
&\quad = \sum_{z^n \in \mathcal{Z}(n)} P_{Z^n}(z^n) \left[ \log P_{Z^n}(z^n) + \log \frac{1}{Q^n(z^n)} \right] \\
&\quad \le \sum_{z^n \in \mathcal{Z}(n)} \sum_{i \in \mathcal{I}(P_{X^n},\, x^n)} P_{X^n}(x_{i+1}^{i+n}) \\
&\qquad \cdot \left[ \log \frac{P_{X^n}(x_{i+1}^{i+n})}{Q^n(z^n)} + \log |\mathcal{I}(P_{X^n},\, x^n)| \right] \\
&\quad \le \sum_{z^n \in \mathcal{Z}(n)} \sum_{i \in \mathcal{I}(P_{X^n},\, x^n)} P_{X^n}(x_{i+1}^{i+n}) \\
&\qquad \cdot \left[ \log \left[ \frac{p(x_{i+1}^{i+M})}{p(x_{i+1}^{i+M} | x_{i-M+1}^{i})} \frac{\prod_{j=1}^{|\mathcal{S}|} \prod_{i=T_j}^{T_{j+1}-1} p_j(z_i)}{\prod_{j=1}^{|\mathcal{S}|} \prod_{i=T_j'}^{T_{j+1}'-1} p_j(z_i)} \right] + \log n \right] \\
&\quad \le -I(X_1^M; X_{M+1}^{2M}) + \sum_{j=1}^{|\mathcal{S}|} c' E_{X^n} |T_j - T_j'| + \log n \\
&\quad \le \sum_{j=1}^{|\mathcal{S}|} c' \sqrt{E_{X^n}(T_j - T_j')^2} + \log n
\end{aligned}$$

where the first inequality follows from (4) and the log-sum inequality and the last inequality follows from Jensen's inequality. Note that

$$c' = \max_{j,\, z} \log(1/p_j(z)) < \infty$$

since $p_j(z)$ is positive for all $j$ and $z$ by assumption. For each $j \in \{1, \ldots, |\mathcal{S}|\}$, let

$$\hat{\pi}_j = \frac{1}{n} \sum_{i=1}^{n} 1(\text{suf}(x_{i-M}^{i-1}) = s_k).$$

Then

$$\begin{aligned}
&E_{X^n}(T_j - T_j')^2 \\
&\quad = E_{X^n} \left( n \sum_{k=1}^{j-1} \hat{\pi}_k - \left\lfloor n \sum_{k=1}^{j-1} \pi_k \right\rfloor \right)^2 \\
&\quad \le E_{X^n} \left( n \sum_{k=1}^{j-1} \hat{\pi}_k - n \sum_{k=1}^{j-1} \pi_k \right)^2 + 2n \sum_{k=1}^{j-1} \pi_k + 1 \\
&\quad \le c'' n + c'''
\end{aligned}$$

by Lemma 5 in the Appendix. □

The above argument fails when there exists some $x \in \mathcal{X}$ and $j \in \{1, \ldots, \mathcal{S}\}$ such that $p_j(x) = 0$, since in this case there exists a $z^n \in \mathcal{Z}(n)$ for which $P_{Z^n}(z^n) > 0$ while $Q^n(z^n) = 0$, giving $D(P_{Z^n} \| Q^n) = \infty$. This problem may be avoided by comparing $P_{Z^n}$ to some distribution $Q_\epsilon^n$ approximating $Q^n$ for which $Q_\epsilon^n(z^n) > 0$ for all $z^n$. Specifically, define $Q_\epsilon^n$ as

$$Q_\epsilon^n(z^n) = \prod_{j=1}^{|\mathcal{S}|} \prod_{i=T_j'}^{T_{j+1}'-1} p_{j,\,\epsilon}(z_i)$$

where, for each $j \in \{1, \ldots, |\mathcal{S}|\}$ and $x \in \mathcal{X}$

$$p_{j,\,\epsilon}(x) = p(x|s_j)(1 - \epsilon) + \frac{\epsilon}{|\mathcal{X}|}$$

and $\{T_j'\}$ is defined as described previously.

*Lemma 2:* Let $X_1, X_2, \ldots$ be drawn from a stationary finite-memory source. Then $Z^n = \text{BWT}\,(\mathcal{R}(X^n))$ implies that there exists some constant $c' > 0$ for which

$$\frac{1}{n} D\left(P_{Z^n} \| Q_{(1/n)}^n\right) \le c \frac{\log n}{\sqrt{n}}.$$

*Proof:* Using an argument similar to the one previously given

$$\begin{aligned}
D\left(P_{Z^n} \| Q_{\epsilon(n)}^n\right) \le &\sum_{z^n \in \mathcal{Z}(n)} \sum_{i \in \mathcal{I}(P_{X^n},\, x^n)} P_{X^n}(x_{i+1}^{i+n}) \\
&\cdot \left[ \log \frac{P_{X^n}(x_{i+1}^{i+n})}{Q_{\epsilon(n)}^n(z^n)} + \log |\mathcal{I}(P_{X^n},\, x^n)| \right]
\end{aligned}$$

where $\log |\mathcal{I}(P_{X^n},\, x^n)| \le \log n$. The first logarithm breaks into a sum of $n$ terms, corresponding to the $n$ symbols $z^n$. Most of these terms take the form

$$\log \frac{p_j(z_i)}{(1 - \epsilon(n)) p_j(z_i) + \epsilon(n)/|\mathcal{X}|}$$

for some $p_j(z_i) > 0$, which is bounded as

$$\log \frac{p_j(z_i)}{(1 - \epsilon(n)) p_j(z_i) + \epsilon(n)/|\mathcal{X}|} \le \log \frac{1}{1 - \epsilon(n)}.$$

The remainder of the terms take the form

$$\log \frac{p}{(1 - \epsilon(n)) q + \epsilon(n)/|\mathcal{X}|}$$

where $p$ and $q$ are distinct probability values with $p > 0$ and $q \ge 0$. This term is bounded as

$$\log \frac{p}{(1 - \epsilon(n)) q + \epsilon(n)/|\mathcal{X}|} \le \log \frac{|\mathcal{X}|}{\epsilon(n)}.$$

Terms of this type can occur in the first $M$ symbols of $x_{i+1}^{i+n}$ or in the $\sum_{j=1}^{|\mathcal{S}|} |T_j - T_j'|$ remaining symbols, where the probability models rely on different histories. There can be at most $n$ terms of the first type and at most $M + \sum_{j=1}^{|\mathcal{S}|} |T_j - T_j'|$ terms of the second type. Thus, for all $n \ge 2$

$$\begin{aligned}
&D\left(P_{Z^n} \| Q_{\epsilon(n)}^n\right) \\
&\quad \le E_{P_{X^n}} \left[ n \log \frac{1}{1 - \epsilon(n)} \right. \\
&\qquad \left. + \left( M + \sum_{j=1}^{|\mathcal{S}|} |T_j - T_j'| \right) \log \frac{|\mathcal{X}|}{\epsilon(n)} + \log n \right] \\
&\quad \le 2n\epsilon(n) \log e + \left(M + \sqrt{cn}\right) \log \frac{|\mathcal{X}|}{\epsilon(n)} + \log n
\end{aligned}$$

since $\log x \leq (x-1)\log e$ and $\epsilon(n) \leq 1/2$ for all $n \geq 2$. Finally, $\epsilon(n) = 1/n$ gives

$$\frac{1}{n} D(P_{Z^n} \| Q^n_{1/n}) \leq c' \frac{\log n}{\sqrt{n}}. \qquad \square$$

For general stationary ergodic distributions, the distribution of the BWT output can be compared to the p.i.i.d. distribution corresponding to the BWT output of an $m$th-order Markov distribution. Toward this end, let $\mathcal{S} = \mathcal{X}^m = \{s_1, \ldots, s_{|\mathcal{S}|}\}$, where $|\mathcal{S}| = |\mathcal{X}|^m$ and $s_1, \ldots, s_{|\mathcal{S}|}$ are ordered lexicographically. Define

$$T_j = 1 + \sum_{i=1}^{n} \sum_{k=1}^{j-1} 1(x_{i-M}^{i-1} = s_k)$$

and

$$T'_j = 1 + \lfloor C(j)n \rfloor, \qquad j \in \{1, \ldots, |\mathcal{S}|+1\}$$

where

$$C(j) = \sum_{k=1}^{j-1} \pi(s_k), \qquad \text{for all } j \in \{1, \ldots, |\mathcal{S}|+1\}.$$

Define $Q^n$ and $Q^n_\epsilon$ as

$$Q^n(z^n) = \prod_{j=1}^{|\mathcal{S}|} \prod_{i=T'_j}^{T'_{j+1}-1} p_j(z_i)$$

$$Q^n_\epsilon(z^n) = \prod_{j=1}^{|\mathcal{S}|} \prod_{i=T'_j}^{T'_{j+1}-1} p_{j,\epsilon}(z_i),$$

where

$$p_j(x) = p(x|s_j)$$

gives the conditional distribution of $x$ given the preceding $m$-tuple $s_j$ and

$$p_{j,\epsilon}(x) = (1-\epsilon)p_j(x) + \frac{\epsilon}{|\mathcal{X}|}.$$

The proof of the following lemma closely follows the earlier arguments.

*Lemma 3:* Let $X_1, X_2, \ldots$ be drawn from a stationary ergodic source with distribution $p$ and entropy rate $H(\mathcal{X})$. Let $Z^n = BWT(\mathcal{R}(X^n))$. For any $m$ such that $p(x|s) > 0$ for all $s \in \mathcal{X}^m$ and all $x \in \mathcal{X}$

$$\lim_{n\to\infty} \frac{1}{n} D(P_{Z^n} \| Q^n) \leq H(X_{m+1}|X^m) - H(\mathcal{X}).$$

Further, there exists some sequence $\{m(n)\}$ such that

$$\lim_{n\to\infty} \frac{1}{n} D\left(P_{Z^n} \| Q^n_{(1/m(n))}\right) = 0.$$

*Proof:* First, consider the case where $p(x|s) > 0$ for all $x \in \mathcal{X}$ and $s \in \mathcal{X}^m$ for some fixed $m \geq 0$. Let $p_\star = \min_{x\in\mathcal{X},\, s\in\mathcal{S}} p(x|s) > 0$. Then

$$\begin{aligned}
&D(P_{Z^n} \| Q^n) \\
&\quad\leq \sum_{z^n \in \mathcal{Z}(n)} \sum_{i \in \mathcal{I}(P_{X^n}, x^n)} P_{X^n}(x_{i+1}^{i+n}) \\
&\qquad \cdot \left[\log \frac{P_{X^n}(x_{i+1}^{i+n})}{Q^n(z^n)} + \log |\mathcal{I}(P_{X^n}, x^n)|\right] \\
&\quad\leq \sum_{x^n \in \mathcal{X}^n} P_{X^n}(x^n) \\
&\qquad \cdot \log \left[\frac{P_{X^n}(x^n)}{\prod_{i=1}^{n} p(x_i|x_{i-m}^{i-1})} \frac{\prod_{i=1}^{n} p(x_i|x_{i-m}^{i-1})}{\prod_{j=1}^{|\mathcal{S}|} \prod_{i=T'_j}^{T'_{j+1}-1} p_j(z_i)}\right] + \log n \\
&\quad= \sum_{x^n \in \mathcal{X}^n} P_{X^n}(x^n) \\
&\qquad \cdot \log \left[\frac{P_{X^n}(x^n)}{\prod_{i=1}^{n} p(x_i|x_{i-m}^{i-1})} \frac{\prod_{j=1}^{|\mathcal{S}|} \prod_{i=T_j}^{T_{j+1}-1} p_j(z_i)}{\prod_{j=1}^{|\mathcal{S}|} \prod_{i=T'_j}^{T'_{j+1}-1} p_j(z_i)}\right] + \log n \\
&\quad\leq -H(X^n) - m\log p_\star + (n-m)H(X_{m+1}|X^m) \\
&\qquad + \sum_{j=1}^{|\mathcal{S}|} c' E_{X^n}|T_j - T'_j| + \log n \\
&\quad\leq nH(X_{m+1}|X^m) - H(X^n) + c|\mathcal{X}|^m \sqrt{n}
\end{aligned}$$

for some constant $c$. Thus,

$$\lim_{n\to\infty} \frac{1}{n} D(P_{Z^n} \| Q^n) \leq H(X_{m+1}|X^m) - H(\mathcal{X}).$$

Dropping the $p_\star > 0$ assumption and replacing $Q^n$ with $Q^n_\epsilon$ gives

$$\begin{aligned}
&D(P_{Z^n} \| Q^n_\epsilon) \\
&\quad\leq \sum_{z^n \in \mathcal{Z}(n)} \sum_{i \in \mathcal{I}(P_{X^n}, x^n)} P_{X^n}(x_{i+1}^{i+n}) \\
&\qquad \cdot \left[\log \frac{P_{X^n}(x_{i+1}^{i+n})}{Q^n_\epsilon(z^n)} + \log |\mathcal{I}(P_{X^n}, x^n)|\right] \\
&\quad\leq \sum_{x^n \in \mathcal{X}^n} P_{X^n}(x^n) \log \left[\frac{P_{X^n}(x^n)}{\prod_{i=1}^{n} p\left(x_i|x_{\max\{1,\, i-m\}}^{i-1}\right)}\right. \\
&\qquad \left. \cdot \frac{\prod_{i=1}^{n} p\left(x_i|x_{\max\{1,i-m\}}^{i-1}\right)}{\prod_{j=1}^{|\mathcal{S}|} \prod_{i=T'_j}^{T'_{j+1}-1} (1-\epsilon)p_j(z_i) + \epsilon/|\mathcal{X}|}\right] + \log n \\
&\quad\leq mH(X_1) + nH(X_{m+1}|X^m) - H(X^n) \\
&\qquad + \log \frac{|\mathcal{X}|}{\epsilon} \left(m + \sum_{j=1}^{|\mathcal{S}|} E_{X^n}|T_j - T'_j|\right) + c'n\epsilon + \log n \\
&\quad\leq nH(X_{m+1}|X^m) - H(X^n) + (c|\mathcal{X}|^m \sqrt{n}) \log \frac{1}{\epsilon} + c'n\epsilon.
\end{aligned}$$

Using $\epsilon(m) = 1/m$ gives

$$\frac{1}{n} D(P_{Z^n} \| Q^n_\epsilon) \leq H(X_{m+1}|X^m) - \frac{1}{n} H(X^n) + \frac{c|\mathcal{X}|^m \log m}{\sqrt{n}}.$$

Careful choice of $m$ gives the desired result. $\square$

TABLE I
RATE OF CONVERGENCE (DOMINANT TERM ONLY) AND COMPLEXITY RESULTS FOR BWT-BASED CODES ON FINITE-MEMORY SOURCES. THE THEORETICAL LIMITS AND CORRESPONDING RESULTS FOR ZIV–LEMPEL CODES ARE INCLUDED FOR COMPARISON

| Source Coding Results on Finite-Memory Sources | | | | |
|---|---|---|---|---|
| | Thm | Algorithm | Redundancy | Compl. |
| BWT-Based Codes | 1 | Move-to-Front | $(1+(1+\epsilon)\lvert\mathcal{X}\rvert)\log\log n/\log n$ | $O(n)$ |
| | 2 | Known State Space $\mathcal{S}$ | $\lvert\mathcal{S}\rvert(\lvert\mathcal{X}\rvert+1)\log n/(2n)$ | $O(n)$ |
| | 2 | Known Memory $M$ | $\lvert\mathcal{X}\rvert^M(\lvert\mathcal{X}\rvert+1)\log n/(2n)$ | $O(n)$ |
| | 3 | Finite Memory | $\sqrt{(\lvert\mathcal{S}\rvert-1)(\lvert\mathcal{X}\rvert-1)\log\lvert\mathcal{X}\rvert\log n/n}$ | $O(n)$ |
| | 4 | I.P.I.D. | $\lvert\mathcal{S}\rvert(\lvert\mathcal{X}\rvert+1)\log n/(2n)$ | $O(n^2)$ |
| Other Univ. Codes | | LZ'77 [21] | $O(\log\log n/\log n)$ | $O(n)$ |
| | | LZ'78 [23] LZ'77 var [22] | $O(1/\log n)$ | $O(n)$ |
| | | CTW [49] | $\lvert\mathcal{S}\rvert(\lvert\mathcal{X}\rvert-1)\log n/(2n)$ | $O(n)$ |
| Limit | | Optimal Redundancy [28] | $\lvert\mathcal{S}\rvert(\lvert\mathcal{X}\rvert-1)\log n/(2n)$ | |

## VII. SUMMARY AND CONCLUSION

The preceding sections describe a variety of universal lossless source codes employing the BWT. One of these codes is a minor variation on an existing BWT-based code, while the other strategies are new. Analyses of the expected description lengths achieved by these algorithms on both finite-memory sources and more general stationary ergodic sources yield both proofs of minimax universality and bounds on the resulting rates of convergence. Table I summarizes the rates of convergence and complexities of the BWT-based source codes on finite-memory sources, comparing those results both to the corresponding bounds for LZ'77, LZ'78, and CTW [49] and to the optimal rate of convergence. While CTW, like the algorithms described in Theorems 1–3, requires complexity that grows only linearly with $n$, that complexity has a hidden dependence on the memory constraint $M$ that makes the algorithm computationally expensive when $M$ is large or unknown. (In the interest of space, the rate of convergence results give only the *dominant terms* in those convergences.) For stationary ergodic sources, discussed at the end of Section V, BWT-based codes achieve $(m,\, n)$th-order redundancy

$$\overline{\delta}_{(n,\,m)}(\theta) \le \frac{|\mathcal{X}|^{m-1}(|\mathcal{X}|+1)}{2}\,\frac{\log n}{n} + O\left(\frac{|\mathcal{X}|^m}{n}\right).$$

As indicated by these results, the BWT is an extremely useful tool for data compression, leading to algorithms that yield near-optimal rates of convergence on finite-memory sources with very low complexity.

While many of the algorithms considered here use sequential codes on the BWT output, the overall data compression algorithms are nonsequential since the transform itself requires simultaneous access to all $n$ symbols of a data string. (Note that $n$ need not be the length of the entire data sequence, as the algorithm may be applied independently on $n$-blocks from the original file.)

## APPENDIX

*Lemma 4:* For any stationary finite-memory source $P_\theta, \theta \in \Lambda$

$$\frac{1}{n}\,H_\theta(X^n) - H_\theta(\mathcal{X}) = O\left(\frac{1}{n}\right).$$

*Proof:* Given a stationary finite-memory source, there exists some integer $M \ge 0$ such that

$$\begin{aligned}\frac{1}{n}\,H_\theta(X^n) &\le \frac{1}{n}\,[MH_\theta(X_{M+1}) + (n-M)H_\theta(X_{M+1}|X_1^M)]\\ &= H_\theta(\mathcal{X}) + \frac{MI_\theta(X_1^M; X_{M+1})}{n}.\end{aligned}$$

□

*Lemma 5:* Let $Z_1, Z_2, \ldots$ be a first-order Markov chain (in steady state). Let $p(1|0) = p > 0$ and $p(0|1) = q > 0$. Let the steady-state probabilities be $\{\pi_0, \pi_1\}$ and let $N(Z^n)$ be the number of 1's in $Z^n$. Then there exist constants $c$ and $d$ such that

$$E((N(Z^n) - n\pi_1)^2) \le cn + d.$$

*Proof:* The steady-state probabilities are given by $\pi_0 = q/(p+q)$ and $\pi_1 = p/(p+q)$. Let $p_{11}(i)$ denote the probability $Z_i = 1$ given $Z_0 = 1$. It can be verified that

$$p_{11}(i) = \frac{p + q(1-p-q)^i}{p+q}.$$

Since $N(Z^n) = \sum_{i=1}^n Z_i$

$$\begin{aligned}&E((N(Z^n) - n\pi_1)^2)\\ &\quad= E\left(\sum_{i=1}^n\sum_{j=1}^n Z_iZ_j\right) - n^2\pi_1^2\\ &\quad= \sum_{i=1}^n\sum_{j=1}^n \pi_1 p_{11}(|j-i|) - n^2\pi_1^2\\ &\quad= \sum_{i=1}^n\sum_{j=1}^n \left(\frac{p}{p+q}\right)\left(\frac{p+q(1-p-q)^{|j-i|}}{p+q}\right)\\ &\qquad - n^2\left(\frac{p}{p+q}\right)^2\\ &\quad= \sum_{i=1}^n\sum_{j=1}^n \frac{pq(1-p-q)^{|j-i|}}{(p+q)^2}\\ &\quad= 2\pi_1\pi_0\sum_{i=0}^{n-1}(n-i)(1-p-q)^i\end{aligned}$$

$$= 2\pi_1\pi_0 \left[ n\,\frac{1-r^n}{1-r} - \frac{-r(1-r)nr^{n-1} + r(1-r^n)}{(1-r)^2} \right]$$

$$= 2\pi_1\pi_0 \left[ n\left(\frac{1}{1-r}\right) - \frac{r(1-r^n)}{(1-r)^2} \right]$$

where $r = 1-p-q$. Finally, $0 < p+q \le 2$ implies $-1 \le r < 1$, giving the desired result. □

## Acknowledgment

The authors wish to thank D. Linde and J. Kahn for helpful discussions related to this work.

## References

[1] M. Burrows and D. J. Wheeler, "A block-sorting lossless data compression algorithm," Digital Syst. Res. Ctr., Palo Alto, CA, Tech. Rep. SRC 124, May 1994.

[2] J. G. Cleary, W. J. Teahan, and I. H. Witten, "Unbounded length contexts for PPM," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 1995, pp. 52–61.

[3] Z. Arnavut and S. S. Magliveras, "Lexical permutation sorting algorithm," *Comput. J.*, vol. 40, no. 5, pp. 292–295, 1997.

[4] K. Sadakane, "A fast algorithm for making suffix arrays and for Burrows–Wheeler transformation," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 1998, pp. 129–138.

[5] N. J. Larsson, "The context trees of block sorting compression," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 1998, pp. 189–198.

[6] Z. Arnavut, D. Leavitt, and M. Abdulazizoglu, "Block sorting transformations," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 1998, p. 524.

[7] B. Chapin and S. R. Tate, "Higher compression from the Burrows–Wheeler Transform by modified sorting," in *Proc. Data Compression Conf.*. Snowbird, UT, Mar. 1998, p. 532.

[8] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 337–343, May 1977.

[9] ——, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 530–536, Sept. 1978.

[10] J. G. Cleary and I. H. Witten, "Data compression using adaptive coding and partial string matching," *IEEE Trans. Commun.*, vol. 32, pp. 396–402, Apr. 1984.

[11] A. Moffat, "Implementating the PPM data compression scheme," *IEEE Trans. Commun.*, vol. 38, pp. 1917–1921, Nov. 1990.

[12] K. Sadakane, "Text compression using recency rank with context and relation to context sorting, block sorting, and PPM*," in *Proc. Conf. Compression and Complexity of Sequences*, June 1997.

[13] ——, "On optimality of variants of the block sorting compression," in *Proc. Data Compression Conf.*. Snowbird, UT, Mar. 1998, p. 570.

[14] ——, "On optimality of variants of the block sorting compression" (in Japanese), in *Proc. Symp. Information Theory and Its Applications*, Dec. 1997, pp. 357–360.

[15] M. Arimura and H. Yamamoto, "Asymptotic optimality of the block sorting data compression algorithm," *IEICE Trans. Fundamentals*, vol. E81-A, no. 10, pp. 2117–2122, Oct. 1998.

[16] ——, "Almost sure convergence coding theorem for block sorting data compression," in *Proc. Int. Symp. Information Theory and Its Applications*, Mexico City, Mexico, Oct. 1998, pp. 286–289.

[17] M. Arimura, "Information theoretic analyzes of block sorting data compression method," Ph.D. dissertation (in Japanese), Univ. Tokyo, Tokyo, Japan, Mar. 1999.

[18] M. Effros, "Universal lossless source coding with the Burrows Wheeler transform," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 1999, pp. 178–187.

[19] K. Visweswariah, S. R. Kulkarni, and S. Verdú, "Output distribution of the Burrows–Wheeler transform," in *Proc. IEEE Int. Symp. Information Theory*. Sorrento, Italy, June 2000, p. 53.

[20] K. Visweswariah, "Topics in the analysis of universal compression algorithms," Ph.D. dissertation, Princeton Univ., Princeton, NJ, 2000.

[21] A. D. Wyner and A. J. Wyner, "Improved redundancy of a version of the Lempel–Ziv algorithm," *IEEE Trans. Inform. Theory*, vol. 41, pp. 723–732, May 1995.

[22] G. Louchard and W. Szpankowski, "On the average redundancy rate of the Lempel–Ziv code," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1–8, Jan. 1997.

[23] S. A. Savari, "Redundancy of the Lempel–Ziv incremental parsing rule," *IEEE Trans. Inform. Theory*, vol. 43, pp. 9–21, Jan. 1997.

[24] L. D. Davisson, "Universal noiseless coding," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 783–795, Nov. 1973.

[25] R. E. Krichevsky and V. K. Trofimov, "The performance of universal encoding," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 199–207, Mar. 1981.

[26] L. D. Davisson, "Minimax noiseless universal coding for Markov sources," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 211–215, Mar. 1983.

[27] J. Rissanen, "Universal coding, information, prediction, and estimation," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 629–636, July 1984.

[28] ——, "Complexity of strings in the class of Markov processes," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 526–532, July 1986.

[29] P. A. Chou, M. Effros, and R. M. Gray, "A vector quantization approach to universal noiseless coding and quantization," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1109–1138, July 1996.

[30] J. Rissanen, "A universal data compression system," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 656–664, Sept. 1983.

[31] M. Weinberger, A. Lempel, and J. Ziv, "A sequential algorithm for the universal coding of finite memory sources," *IEEE Trans. Inform. Theory*, vol. 38, pp. 1002–1014, May 1992.

[32] M. Weinberger, J. Rissanen, and M. Feder, "A universal finite memory source," *IEEE Trans. Inform. Theory*, vol. 41, pp. 643–652, May 1995.

[33] P. Fenwick, "Block sorting text compression," in *Proc. Australasian Computer Science Conf.*, Melbourne, Australia, Feb. 1996.

[34] M. Nelson, "Data compression with the Burrows–Wheeler transform," *Dr. Dobbs' J.*, Sept. 1996.

[35] M. Schindler, "A fast block-sorting algorithm for lossless data compression," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 1997, p. 469.

[36] E. M. McCreight, "A space-economical suffix tree construction algorithm," *J. ACM*, vol. 23, no. 2, pp. 262–272, Apr. 1976.

[37] P. Fenwick, "Improvements to the block sorting text compression algorithm," Dept. Comput. Sci., Univ. Auckland, Tech. Rep. TR-120, Aug. 1995.

[38] G. Seroussi and M. Weinberger, "On tree sources, finite state machines, and time reversal," in *Proc. IEEE Int. Symp. Information Theory*, Whistler, Canada, Sept. 1995, p. 390.

[39] B. Ya. Ryabko, "Book stack data compression," *Probl. Pered. Inform.*, vol. 16, no. 4, pp. 16–21, 1980.

[40] J. L. Bentley, D. D. Sleator, R. E. Tarjan, and V. K. Wei, "A locally adaptive data compression scheme," in *Proc. 22nd Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Oct. 1984, pp. 233–242.

[41] ——, "A locally adaptive data compression scheme," *Commun. Assoc. Comput. Mach.*, vol. 29, pp. 320–330, Apr. 1986.

[42] P. Elias, "Interval and recency rank source coding: Two on line adaptive variable-length schemes," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 3–10, Jan. 1987.

[43] J. Muramatsu, "On the performance of recency-rank and block-sorting universal lossless data compression algorithms," in *Proc. IEEE Int. Symp. Information Theory*, Sorrento, Italy, June 2000, p. 327.

[44] J. Rissanen and G. G. Langdon, Jr., "Arithmetic coding," *IBM J. Res. Devel.*, vol. 23, no. 2, pp. 149–162, Mar. 1979.

[45] N. Merhav, "On the minimum description length principle for sources with piecewise constant parameters," *IEEE Trans. Inform. Theory*, vol. 39, pp. 1962–1967, Nov. 1993.

[46] F. M. J. Willems, "Coding for a binary independent piecewise-identically-distributed source," *IEEE Trans. Inform. Theory*, vol. 42, pp. 2210–2217, Nov. 1996.

[47] G. Shamir and N. Merhav, "Low complexity sequential lossless coding for piecewise stationary memoryless sources," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1498–1519, 1999.

[48] P. C. Shields, "Universal redundancy rates do not exist," *IEEE Trans. Inform. Theory*, vol. 39, pp. 520–524, Mar. 1993.

[49] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens, "The context-tree weighting method: Basic properties," *IEEE Trans. Inform. Theory*, vol. 41, pp. 653–664, May 1995.