Review of[1] of
The Burrows-Wheeler Transform:
Data Compression, Suffix Arrays, and Pattern Matching
by Donald Adjeroh, Timothy Bell and Amar Mukherjee
Springer, 2008
352 pages, Hardcover

Review by
Shoshana Neuburger shoshana@sci.brooklyn.cuny.edu
Graduate Center, CUNY, New York, NY, 10016

1 Introduction

David Wheeler and Mike Burrows introduced the Burrows-Wheeler Transform (BWT) as a practical method of data compression in 1994. The elegant simplicity of the BWT has captivated researchers for the past decade. The BWT is simple yet powerful. Other lossless compression techniques are a lot more cumbersome, making it difficult to assess their efficiency. Lossless compression is the preferred method of conserving space in text files. Audio and video files are typically compressed with lossy compression mechanisms since some content can be eliminated without noticeably compromising the quality of compressed data.

The Burrows-Wheeler Transform presents an innovative approach to data compression. The transform is simply a permutation of the original data. This unique permutation makes it easier to process the data. Since the Burrows-Wheeler Transform is easily reversed, it lies at the basis of lossless compression techniques. The nature of the transform allows the transformed data to readily be compacted by a variety of compression techniques. Among them are run-length and move-to-front encoding.

The Burrows-Wheeler transform works with one block of text at a time. The transform sorts the cyclic shifts of the text lexicographically. Its output consists of a string, L, composed of the last character in each of the sorted rotations, along with a pointer to the position of the last character in this permutation. The reverse transform reconstructs the original text by a series of sort and count operations. The transform exploits the property that several occurrences of a character will appear in the same order in both the original and permuted data.

This book exposes the reader to many applications of the Burrows-Wheeler Transform beyond data compression. The book begins by explaining how the transform works. Then the reader is introduced to the various compression techniques that can be applied to the transformed text to yield a concise representation of the original data. Related techniques in pattern matching algorithms are presented as well.

Only recently has the BWT spurred so many results. This book invites the reader to explore a new topic. The material is easily internalized in the textbook's clearly defined style and concise explanations.

2 Summary

Chapter 1

The text begins with an example of the Burrows-Wheeler Transform that is easy to follow. The inverse operation that reconstructs the text is also presented. With the use of pointers, a block of text is sorted without incurring additional storage space. Retrieving the original data from the transform consists of a series of sort and count operations, which can also be done in constant space using pointers. A transform does not alter data, but it does present a new perspective on existing information. The transform does not change the size of the file. However, it simplifies compression of a file since like characters are clustered together. The sorted permutation of text characters brings together characters that precede identical substrings. Repetition in the original text becomes noticeable in the permuted data. The chapter concludes with a brief history of data compression.

Chapter 2

After an exposure to the basic function of the Burrows-Wheeler Transform, the reader is presented with a practical implementation of the transform. With a few tricks, encoding and decoding is performed in time and space proportional to the input size. The text includes step-by-step examples with pseudo code that is easy to follow.

Chapter 3

Once we have a basic understanding of how the BWT works, we can consider its usage in data compression. Among the techniques used to concisely encode the output of a BWT are run-length encoding, move-to-front encoding, inversion frequencies, distance coding, frequency counting methods, and wavelet trees. Run-length encoding, which is one of the simplest approaches, yields the best compression of Burrows-Wheeler transformed data. In addition to this benefit, run-length encoding is also a very fast technique.

When space is a consideration, we seek to represent symbols in the fewest bits possible, using *entropy coding*. *Entropy* is a lower bound on the space required to represent a string. In entropy encoding, the representation of a symbol is based on the probability of that symbol occurring. The probabilities can either be estimated adaptively, "learning" during encoding, or non-adaptively, based on historical data.

Since the BWT works on entire blocks of text, the block size affects compression performance. There is a balance between larger block size and limited cache size, to allow faster access to Random Access Memory. The BZIP2 program is a compression system based on the BWT.

Chapter 4

The suffix tree is a compact, linear representation of the suffixes of a string. The suffix array is a permutation of the string that represents the lexicographical order of its suffixes. The suffix array and the suffix tree represent the same information in different forms. There exists a close relationship between the BWT and the suffix array of a string.

This chapter covers linear-time suffix tree and suffix array construction algorithms, as well as implementation issues to consider. In its obvious form, the suffix tree represents quadratic

information, and thus occupies quadratic space relative to the size of the input string. Yet, its linear representation can be obtained in linear time. The most remarkable results are Ukkonen's online construction algorithm and Farach's alphabet-independent, recursive construction algorithm. The suffix array requires less space than the suffix tree. The first direct suffix tree construction algorithm is due to Manber and Myers, which relies on the *doubling* technique of Karp, Miller, and Rosenberg to construct the data structure in $O(n \log n)$ time, where n is the size of the input. More recently, several linear time algorithms have been developed using the divide and conquer approach.

Chapter 5

This chapter examines the theoretical performance of the BWT, in terms of time and space complexity. The output is also analyzed through empirical evaluation and with assumptions about the input. The compression performance is described in terms of entropy. The relationship with other compression schemes is also explored. Data compression by Prediction by Partial Matching (PPM) and Ziv-Lempel coding are compared to the compressed BWT.

Chapter 6

This chapter presents many variations, extensions, and generalizations of the BWT that have been considered. For each modification considered, empirical performance data is included whenever possible.

Chapter 7

The pattern matching problem is that of locating all occurrences of a *pattern* string in a larger string called the *text*. Pattern matching and data compression are intimately related, and at the same time can work against each other. Compression removes redundancies in a text, which also destroys the text's natural structure, thereby complicating information retrieval. The *decompress-then-search approach* involves restoring the original data so that pattern matching proceeds as usual. *Fully-compressed pattern matching*, searching among compressed data, is preferred since decompression can be time and space consuming. Many compression mechanisms use different representations for a substring depending on its context, making fully-compressed pattern matching an impossibility.

Efficient exact pattern matching algorithms are reviewed. Among them are the Knuth-Morris-Pratt automaton, the Boyer-Moore algorithm, and the Karp-Rabin randomized algorithm. Algorithms that solve variants of pattern matching, such as multiple pattern matching and pattern matching with don't cares, are also presented. The BWT provides a framework for efficient pattern matching directly on compressed data. Among the methods are binary search, adapted Boyer-Moore, and the suffix array. The *FM-index* combines the BWT and the suffix array to obtain a compressed suffix array. It is a *f*ull text index in *m*inute space. A comparison of the performance of the BWT pattern matching methods is presented.

Chapter 8

The text concludes with detailed applications of the Burrows-Wheeler Transform to demonstrate that the BWT can be used to effectively address problems that at first glance seem unrelated to

the transform. The compressed suffix tree occupies $O(\log n)$ bits rather than the usual $O(n \log n)$ bits, for an input string of size n. Compressed suffix trees and compressed suffix arrays lead to compressed full-text indexing.

3 Opinion

Although familiarity with algorithms and their analysis is assumed, the reader is not expected to have any prior exposure to pattern matching algorithms. I can recommend this text to a wide variety of readers. This book is well suited for a researcher familiar with the field of pattern matching algorithms who seeks to understand the Burrows-Wheeler Transform and its many applications to the existing framework of algorithms. Yet, it is also suitable as an introduction to pattern matching algorithms since the text describes the classical pattern matching algorithms and data structures as they are encountered.

This would be a wonderful course textbook since its coverage is comprehensive and its explanations do not assume that the reader has a great deal of background. Because this was not the author's original intent, the instructor would have to design his own exercises.

The clear exposition of the text and its well-defined structure facilitate a smooth traversal of its contents. Many computer scientists can gain from reading this book. Since the textbook touches a variety of topics, the curious reader might seek greater detail than that provided by this text. Each chapter ends with a *Further Reading* section which refers the reader to sources that discuss each specific topic at greater length. This section is a wonderful resource to researchers who would like to delve further into the topics and extend the work described by the text. There is plenty of room for innovation since there remain many unexplored avenues around the Burrows-Wheeler Transform.