Techniques for Specialized Data Compression

Jakub Swacha
University of Szczecin, Poland

INTRODUCTION

In today's world, the continuing improvement in data storage technologies makes enormous storage capacities available to users. For instance, during the 1996-2010 period, the average storage capacity of desktop personal computer drive increased by 375 times (Adams, 2012). Even so, the growth of available storage capacity is still outpaced by the growth of the information produced worldwide, especially that a gigabyte of stored content can generate a much higher volume of transient data that is not typically stored, but is often transmitted (Gantz & Reinsel, 2011). Hence the need to conserve not only data storage space but also data transmission bandwidth. This need is answered by data compression.

Data compression is "the process of converting an input data stream into another data stream that has a smaller size" (Solomon, 2007). Data compression is possible thanks to redundancy of data; it makes use of the fact that some portions of the input stream need not to be stored, as they may be recreated given the remaining parts of the stream, and/or the fact that some portions of the data are either not relevant to the user at all, or their relevance is negligible. Obviously, pursuing the latter option usually leads to achieving much higher compression ratios, but results in a loss of information qualified as irrelevant during compression, which may be considered as relevant in the future, for another use, or by another user. As there is extensive literature devoted to lossy data compression (see e.g., Sayood, 2012, and references therein), this article describes only lossless methods.

The lossless data compression methods can be classified into two types. The general-purpose methods use a general model that adapts to its input, and thus manage to compress various types of data. The specialized methods are designed to process only one type of data (defined more or less narrowly). Thus, they can not only start compression with a model prepared for data of that specific type, but also exploit redundancy specific only to that type, which would be invisible to a general-purpose method.

Contemporary data compression methods typically combine a set of techniques to achieve superior compression ratios. The aim of this article is to review such component techniques, useful for specialized compression of various types of data. First, however, the Background section gives some information on how most popular general-purpose data compression methods work.

BACKGROUND

The base technique of data compression, that is included in most contemporary compression methods, is statistical coding. It uses statistics of occurrence of respective symbols in the input stream to minimize the size of the output stream. The most widely-used technique of this kind is Huffman coding, which assigns short codewords to frequent input symbols, and long codewords to rare symbols (Huffman, 1952). The Huffman coding is optimal in the sense that no other codeword assignment could produce shorter output stream. Further improvement is still possible by assigning value ranges, instead of individual codewords, to input symbols. Such approach is taken by the arithmetic coding, where the entire input stream is encoded with a binary fraction representing its cumulative probability (Rissanen, 1976).

Most real-world data types exhibit some form of correlation between symbols within them, which cannot be exploited by mere counting occurrences of individual symbols in the input stream. There are

DOI: 10.4018/978-1-4666-5888-2.ch351

three popular types of solutions to this problem. The first (also historically) exploits the idea of grouping input symbols into phrases, which are treated in a way similar to the other input symbols. One approach of this kind (represented, e.g., by Deflate) uses the processed input stream buffer as a dictionary (Ziv & Lempel, 1977) whereas another (represented, e.g., by LZW) maintains an explicit phrase dictionary (Ziv & Lempel, 1978). Although solutions of this type do not achieve superior compression ratios, due to their modest usage of resources, they became the standard of general-purpose compression, used commonly in compression utilities, such as PKZip, gzip, 7-Zip, or WinRar, proprietary container formats, such as CDR, CHM, JAR or SWF, and in various data transmission protocols.

The second type of solutions is, presumably, the easiest to conceive, though the most resource intensive in practice. It aims to develop a more sophisticated statistical model of the source, one which takes into consideration the context of symbol occurrence. One such solution is Prediction by Partial Matching in which, in its classic form, counters are updated in a single context for each input symbol (Cleary & Witten, 1984); another is Context Mixing in which counters in multiple contexts are updated for each input symbol, i.e., the actual symbol count used for encoding is an average of its counts in different contexts, weighted by how efficient respective context has been at symbol probability estimation so far (Mahoney, 2005). Whereas PPM is available as a user-selectable compression mode of several popular general-purpose compression utilities, such as WinRar or 7-Zip, most implementations of CM are still too slow for typical practical usage scenarios, and can only be found in not popular, but compression-record-breaking programs, such as PAQ.

These two types of solutions can be combined to obtain in-between compression efficiency and resource usage (Szyjewski & Swacha, 2002). Similar compromise can be achieved using the solutions of the third type. They apply sort-based transforms, such as the Burrows-Wheeler Transform (BWT), to the input stream, so that it can be efficiently encoded using simple statistical coding (Burrows & Wheeler, 1994). Best-known example of compression utility using this approach is bzip2.

TECHNIQUES FOR SPECIALIZED DATA COMPRESSION

H

Reasons for Using Specialized Data Compression

General-purpose compression methods are based on data models designed for 'typical' data. One problem is that their parameters cannot be fine-tuned to any specific type of data, as it would have inevitably led to hurting their compression efficiency on other types of data. Yet, due to the adaptive nature of the general-purpose compression methods, if the compressed data are large enough for their model to actually adapt, it is not the biggest obstacle to achieve higher compression ratios.

Much bigger problem is caused by the fact that their adaptivity has its limitations. They are based on several assumptions that are usually, but only to some extent, held for most, but not all, types of data encountered in everyday use. Typical assumptions are that the input data stream is, e.g., homogeneous, byte-oriented, one-dimensional, and unstructured. In practice, the data files in common use are often:

- Not homogeneous, containing sections with vastly different statistical properties, e.g., a word processor document with page settings, text, and embedded images;
- Not byte-oriented, being composed of elements of smaller or larger size, e.g., an audio file with 16-bit sampling resolution;
- Having internal structure and/or more than one dimension, and resulting strong correlation between elements which are related but not located next to each other in the file, e.g., data in different table rows but the same column.

Review of Specialized Lossless Data Compression Techniques

Modeling Content at a Higher Level

The general-purpose compression methods usually treat their input as a stream of bytes. Most real-world data types are composed of elements that span over