5-2005

Inversion Ranks for Lossless Compression of Color Palette Images

Ziya Arnavut
SUNY Fredonia

Ferat Sahin
Rochester Institute of Technology

Recommended Citation

Z. Arnavut and F. Sahin, "Inversion ranks for lossless compression of color palette images," 2005 IEEE International Conference on Electro Information Technology, Lincoln, NE, 2005, pp. 5 pp.-5. doi: 10.1109/EIT.2005.1626986

Inversion Ranks for Lossless Compression of Color Palette Images

Ziya Arnavut

Department of Computer Science
SUNY Fredonia
Fredonia, NY 14063
email:ziya.arnavut@fredonia.edu

Ferat Sahin

Department of Electrical Engineering
Rochester Institute of Technology
Rochester, NY 14623
email:feseee@rit.edu

Abstract

Palette images are widely used in World-Wide-Web (WWW) and game cartridges applications. Many images used in the WWW are stored and transmitted after they are compressed losslessly with the standard Graphics Interchange Format (GIF), or Portable Network Graphics (PNG). Well known two dimensional compression schemes, such as JPEG-LS and CALIC, fails to yield better compression than GIF or PNG, due to the fact that the pixel values represent indices that point to color values in a look-up table.

The GIF standard uses Lempel-Ziv compression, which treats the image as a one-dimensional sequence of index values, ignoring two-dimensional nature. Bzip, another universal compressor, yields even better compression gain than the GIF, PNG, JPEG-LS, and CALIC.

Variants of block sorting coders, such as Bzip2, utilizes Burrows-Wheeler Transformation (BWT) [7], followed by Move-to-Front (MTF) transformation [5], [11] before using a statistical coder at the final stage. In this paper, we show that the compression performance of Block Sorting Coders can be improved almost 14% on average by utilizing inversion ranks instead of the Move-to-Front coding.

KEY WORDS
Burrows Wheeler Transformation, Palette Images, Lossless Compression

1. Introduction

Image compression usually operates on one or multiple intensity planes of digitized images. For computer graphics images, most often each color is mapped to an index from a look-up table and the indexes of the pixels are then compressed, usually losslessly. The size of the look-up table is much smaller than that of the color intensity space. Hence, this color palette approach usually can achieve better compression ratio for images with limited color.

Highly compressed palletized images are used in many applications. For example, palletized images are widely used in WWW. Many images used in WWW are stored and transmitted as GIF files, which uses Lempel-Ziv compression. Lempel-Ziv compression treats the data as one dimensional sequence of index values. The Portable Network Graphics (PNG) format was designed to replace GIF format. PNG was developed as a patent-free answer to the GIF format. But it is also an improvement on the GIF technique. An image in a lossless PNG file can be 5%-25% better than GIF file of the same image. PNG builds on the idea of transparency in GIF images and allows the control of the degree of transparency, known as opacity. Saving, restoring and re-saving a PNG image will not degrade its quality.

Several researchers have investigated different techniques to improve compression performance of Palette images. Memon and Venkateswaran [12] treated the problem as a general smoothness maximization problem. Zeng et al. [20] suggested index difference based re-indexing method. They show that these reordering and re-indexing help two dimensional compression schemes, such as JPEG-LS and JPEG-2000, and better compression rates than GIF were possible. Ausbeck [4] introduces Piecewise-Constant (PWC) image model, which is two-pass object-based model. In the first pass boundaries between constant color pieces are determined, and with the second pass domain colors. PWC image model, which takes into account two dimensionality of the pseudo-color images, yields the best compression performance [4]. Most recently, Forchammer and Salinas [10] developed a 2D version of Prediction by Partial Matching (PPM). This technique has been shown to perform better than PWIC on palette images that have less number of color (less than 10 color).

In this work, we investigate the utilization of block-sorting transformations for compression of palette images.

Introduced by Burrows and Wheeler [7], the *Block Sorting Coder* (also known as BW94, Block Sorting Compression Algorithm, and Burrows-Wheeler Transform) is one of the best universal coder. The Block Sorting Coder (BSC) received considerable attention. It achieves compression rates as good as context-based methods, such as PPM, but at execution speeds closer to Ziv-Lempel techniques [7], [8], [9].

The BSC is mainly composed of a block-sorting transformation which is known as Burrows-Wheeler Transformation (BWT), followed by Move-To-Front (MTF) coding. The Move-To-Front (MTF) coding originally was introduced by Bentley et al. [5] and also independently discovered by Elias [11] (called Recency Ranking by Elias). The MTF coder, when implemented with a symbol or character base, starts with an identity permutation of the size of the underlying data source's alphabet. For example, when an MTF coder is implemented for an 8-bit-symbol data string the identity permutation is constructed from the set of $\{0, \ldots, 255\}$. Whenever a new symbol or character is received from an underlying data string the coder outputs the index of the symbol, and if the symbol is not at the front of the list, the coder adjusts the permutation (list) by simply moving the symbol to the front of the existing permutation. In essence, a new permutation is generated for each symbol in the data string.

In [3] we have shown that while the Move-to-Front (MTF) coder may be needed to obtain better compression gains when the underlying data stream is text MTF coder is not needed for compression of image data. In particular, we have shown that after transforming image data with the BWT by utilizing the structured arithmetic coder [9] about 8% more compression can be attained over the well-known Block Sorting coder Bzip2. Our recent study has shown that utilizing inversion ranks [2] after the BWT transformation followed by structured arithmetic coding improves the compression gain about 14% with respect to Bzip2.

This paper is organized as follows: in Section 2, we briefly expose the reader to the Burrows-Wheeler Transformation. In Section 3, we explain inversion ranks. In Section 4, we present the experimental results of the proposed technique and compare them to CALIC and Bzip2. In section 5, we conclude our discussion.

2. Burrows-Wheeler Transformation

In this work we assume a basic knowledge of discrete mathematics. Interested readers may consult [13] for the definition of *multiset permutations*. Further description and implementation of the BWT is given in [7].

Given a multiset permutation ω of size n, construct a matrix M, by forming successive rows of M which are consecutive cyclic right-shifts of the sequence ω. By sorting the rows of M lexically, we may transform M to a different matrix, M'. For example, if the multiset permutation is $\omega = [3, 1, 3, 1, 2]$, we construct the matrix

$$M = \begin{pmatrix} 3 & 1 & 3 & 1 & 2 \\ 1 & 3 & 1 & 2 & 3 \\ 3 & 1 & 2 & 3 & 1 \\ 1 & 2 & 3 & 1 & 3 \\ 2 & 3 & 1 & 3 & 1 \end{pmatrix},$$

by forming the successive rows of M, which are consecutive cyclic right-shifts of the sequence ω. By sorting the rows of M lexically we transform it to

$$M' = \begin{pmatrix} 1 & 2 & 3 & 1 & 3 \\ 1 & 3 & 1 & 2 & 3 \\ 2 & 3 & 1 & 3 & 1 \\ 3 & 1 & 2 & 3 & 1 \\ 3 & 1 & 3 & 1 & 2 \end{pmatrix},$$

Let F' be the first and L' be the last column vector of M'. It is clear that the first column F' of matrix M' is sorted values of ω in ascending order, while the other columns are not sorted. The above process is the forward BWT, where the transmitter transmits the index of the ω in M' and L' to the receiver after applying MTF coder on L' and coding the resulting data with a coder, such as arithmetic coder.

The reverse BWT transformation is faster than the forward transformation. Indeed, there is a bijection between the last column L' and the first column F' of M'. The rows which have the same v value in their last positions (in L'), are the right cyclic shifts of the rows that have v in their first position. Starting from $v = 1$, searching L' from top to bottom for each occurrence of v and numbering each occurrence consecutively for each v ($1 \leq v \leq m$), yields a permutation. This permutation defines a bijective map between the first column (F') and the last column (L') of M'. The permutation obtained from the process described above is called Reverse BWT.

3. The Inversion Ranks

All the block sorting compressor variants (Bzip2, Szip, Bks98) utilize the MTF coder on the data stream transformed with the Burrows-Wheeler transformation, before actually sending the data stream to an encoder such as Arithmetic or Huffman encoder. The MTF coder may be needed to optimize the compression gains for text data. In [3] we showed that after the palette images are BWT transformed by utilizing the hierarchical arithmetic coder [14] %8 more compression can be attained over Bzip2 on average, on the test image data set.

The notion of an *inversion table* for a given permutation was introduced quite early [13] in an effort to provide con-

cise representations of ordinary permutations. Several variants and types of inversions were defined at different times by different authors. In [16], Sedgewick gives some other inversion generation methods for permutations.

In this paper, we first extend the inversion ranks which are defined for permutations [16], [2] to general data strings.

Definition 4.1
Let M be a multiset permutation of elements from an underlying set $S = \{1, 2, \ldots, k\}$. Let "$\odot$" denote *catenation* of data strings. We define the *inversion rank* vector $D = D_k$ for M as follows:

1. $D_0 =<>$.
2. $D_i = D_{i-1} \odot T_i$ *with* $T_i =< x_1, x_2, \ldots, x_{f_i} >$ *where*
   - **i)** $x_1 =$ position of the first occurrence of i in M.
   - **ii)** and for $j > 1$, $x_j =$ number of elements y in M, $y > i$ occurring between the $(j-1)^{st}$ and j^{th} occurrence of i in M.

For example, for the multiset permutation $M = [1, 1, 2, 3, 1, 2, 4, 3, 4, 2, 4]$, we have $S = (1, 2, 3, 4)$. Initially $D_0 =<>$. For $i = 1 \in S$, we have $D_1 = D_0 \odot T_1$, where $T_1 =< 1, 0, 2 >$, so $D_1 =< 1, 0, 2 >$. For $i = 2 \in S$, $D_2 = D_1 \odot T_2$, where $T_2 =< 3, 1, 3 >$. Therefore, $D_2 =< 1, 0, 2, 3, 1, 3 >$. For $i = 3 \in S$, $D_3 = D_2 \odot T_3$, where $T_3 =< 4, 1 >$, so $D_3 =< 1, 0, 2, 3, 1, 3, 4, 1 >$. For $i = 4 \in S$, $D_4 = D_3 \odot T_4$, where $T_4 =< 7, 0, 0 >$. Hence, $D = D_4 =< 1, 0, 2, 3, 1, 3, 4, 1, 7, 0, 0 >$.

To recover the original multiset permutation M from D, we need the knowledge of the multiset described by $F = (f_1, f_2, \ldots, f_k)$ and $S = (1, 2, \ldots, k)$. We initially, let $M = [-, -, \ldots, -]$ where $|M| = |D| = \sum_{i=1}^{k} f_i$. From the definition of inversion ranks, D is built recursively as $D_i = D_{i-1} \odot T_i$, where $T_i =< x_1, x_2, \ldots, x_{f_i} >$ and x_1 represents the position of the first occurrence of i in M and each x_j, $j > 1$, represents the number of elements greater than i, which occur between $(j-1)^{st}$ and j^{th} occurrence of i in M. Hence, we can recover the elements of M by first inserting i in location $M[x_1]$ and for $j = 2, \ldots, f_i$ inserting i in the $(x_j + 1)^{st}$ dash position in M from the last inserted i.

For example, for the above multiset permutation M, $F =< 3, 3, 2, 3 >$. The receiver, upon receiving vectors S, F and D, can reconstruct the multiset permutation M as follows: It first creates a vector M of size $\sum_{i=1}^{4} f_i = 11$. From the ordered set S and F, it determines that the first element of M is 1 and there are three $1's$ in the multiset M. The receiver then knows that, the first three entries in D are the locations related to 1 and in the first pass, the receiver inserts 1's in their location in M correctly. Since the first entry in D is 1, it follows that the location of the first element in D is at position 1, hence $M = [1, -, -, -, -, -, -, -, -, -, -]$. The second entry in D is 0. This means that, there is no element which is greater than 1, between the first and second occurrence of 1. Hence, the receiver inserts the second 1 in the first blank position next to the first 1, so $M = [1, 1, -, -, -, -, -, -, -, -, -]$. The third entry in D is a 2. This means that, there are two elements greater than 1, between the second and third 1's. Hence, the third 1 should be placed in the third empty position after the second 1. Therefore, $M = [1, 1, -, -, 1, -, -, -, -, -, -]$. Again, from S and F, the receiver knows that there are three 2's in M. Accessing the fourth position in D it learns the location of the first 2 in M, that is, the first 2 should occur in position 3 in M. Hence, $M = [1, 1, 2, -, 1, -, -, -, -, -, -]$.

Figure 1. Sunset color-mapped image in gray.

The receiver then proceeds to insert the second 2 into M. From D, the receiver determines that between the first 2 and second 2, there is one element which is greater than 2. So, starting from the location of the first 2 in M, the receiver skips one blank and inserts the second 2 into the second blank position. Similarly, for the third 2 the receiver determines from D that between the second and third 2, there are three elements which are greater than 2. Therefore, the receiver inserts the third 2 into the fourth blank position, after the second 2. Hence, $M = [1, 1, 2, -, 1, 2, -, -, -, 2, -]$. Repeating the above procedure, the receiver can fully reconstruct M.

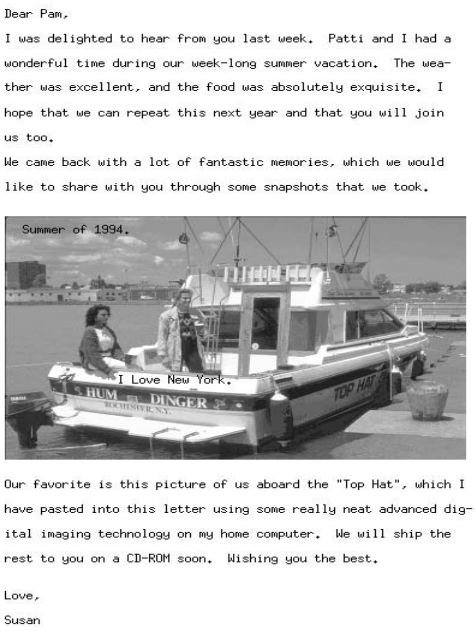

Figure 2. Compound color-mapped image in gray.

4. Experimental Results

The test images utilized in our work are obtained from Paul Ausbeck [4]. Two of the test images are shown in Figures 1–2.

Table 1 presents the experimental results of our work on palette images. Our technique is represented under the column *Binv* in Table 1. Upon transforming the data with the BWT transformation, we utilized the inversion ranks and later the structured arithmetic coder. The structured arithmetic coder was originally introduced by Moffat et al. [14] and improved by Fenwick [9].

Clearly, CALIC performs poorly on the palette images where Bzip2 yields better compression than CALIC. However, our technique outperforms the results of CALIC and Bzip2. On average, we have a gain of 14% over Bzip2, while the gain over CALIC is about 42%.

5. Conclusion

All variants of the block sorting coder (Bzip2, Szip, and Bks98) first transform a given data stream with the Burrows-Wheeler Transformation (BWT) and then utilize the Move-to-Front (MTF) transformation, before coding the resulting data stream with an Arithmetic or Huffman coder.

File	Size	Bzip2	GIF	Binv
benjerry	28326	3840	4401	**3156**
books	23300	10290	11177	**9218**
ccitt01	505286	24809	38862	**22304**
cmpndd	394294	59324	62682	**54481**
cmpndu	394294	49166	76759	**43559**
gate	61302	18340	23313	**16227**
netscape	61382	13842	17442	**11835**
stone	11822	4088	4753	**3910**
sunset	308278	76103	100186	**63618**
winaw	148918	16066	18559	**13930**
Total	1937202	275810	358134	**242238**
Avg. Bpp.		1.139	1.479	**1.000**

Table 1. Compression results of different techniques for Palette images.

In this paper we show that after a palette image is BWT transformed, by using the inversion ranks and a structured arithmetic coder, approximately 14% better compression can be obtained over the Bzip2, while our technique yields 48% better compression than GIF.

References

[1] Arnavut, Z., and Magliveras, S. S., "Block Sorting and Compression," *Data Compression Conference*, DCC-97, Snowbird Utah, pp. 181-190, March 1997.

[2] Arnavut, Z., "Move-to-Front and Inversion Coding," *Data Compression Conference*, DCC-2000, Snowbird Utah, pp. 191-200, March 2000.

[3] Arnavut, Z., and Otu, H., "Compressing Color-mapped Images with Burrows-Wheeler Transformation," *Proceedings of Signal Processing Conference*, IASTED, Rhodes, Greece, Acta-Press, pp. 185-189, July 2001.

[4] Ausbeck P. J. Jr. "A Streaming-Constant Model", *Proc. of Data Compression Conference*, DCC-99, Snowbird Utah, pp. 208-217, March 1999.

[5] Bentley J. L., Sleator D. D., Tarjan, R.E., and Wei V. K. "A Locally Adaptive Data Compression Scheme," *Communications of the ACM*, Vol. 29, No. 29, pp. 320-330, April 1986.

[6] Balkenhol B., Kurtz S., and Shartov Y. M. "Modifications of Burrows and Wheeler Data Compression Algorithm" *Proc. of Data Compression Conference*, DCC-99, Snowbird Utah, pp. 188-197, 1999.

[7] Burrows M. and Wheeler D. J. "A Block-sorting Lossless Data Compression Algorithm," *SRC Research Report 124*, Digital Systems Research Center, Palo Alto, CA., May 1994. (*ftp site: gatekeeper.dec.com, /pub/DEC/SRC/research-reports/SRC-124.ps.Z*)

[8] Cleary J. G., Teahan W. J., and Witten, I. H. "Unbounded Length Contexts for PPM," *Proc. of Data Compression Conference*, DCC-95, Snowbird Utah, pp. 52-61, 1995.

[9] Fenwick P. "The Burrows-Wheeler Transform for Block Sorting Text Compression: Principles and Improvements," *The Computer Journal*, Vol. 39, No. 9, 1996.

[10] Forchhammer, S., and Salinas, M., J. "Progressive Coding of Palette Images and Digital Maps" *Proceedings of Data Compression Conference*, DCC-2002, Snowbird, Utah, pp.362-371, 2002.

[11] Elias P. "Interval and recency Rank Source Coding: Two On-Line Adaptive Variable-Length Schemes", *IEEE Transactions on Information Theory*, Vol. IT-33, No. 1, pp. 3-10, 1987.

[12] Memon, N. D. and Venkateswaran, A., "On Ordering Color Maps for Lossless Predictive Coding", *IEEE Transactions on Image Processing*, Vol 5, No. 11, 1996, pp.1522-152.

[13] Knuth, D., *The Art of Computer Programming*, Vol. 3, Addison-Wesley Publishing Company, Reading, Mass., 1973.

[14] Moffat A., Neal R., and Witten, I. H., "Arithmetic Coding Revisited" *Proc. of Data Compression Conference*, DCC-95, Snowbird Utah, pp. 202-211, 1995.

[15] Schindler M. "A Fast Block Sorting Algorithm for Lossless Data Compression", *Proc. of Data Compression Conference*, DCC-97, Snowbird Utah, pp. 469, 1997.

[16] Sedgewick, R. "Permutation Generation Methods: A review", *ACM Computing Surveys*, Vol 9, No. 2:137-164, 1977.

[17] Seward J. "The Bzip2 program", vers. 0.1pl2, 1997. http://www.muraroa.demon.co.uk.

[18] Shannon, C. E., "A Mathematical Theory of Communication". *Bell System Technical Journal*, 27:398-403, 1948.

[19] Wu X. "An Algorithmic Study on Lossless Image Compression" *Proc. of Data Compression Conference*, DCC-96, Snowbird Utah, pp. 150-159, 1996.

[20] Zeng, W., Li, J., and Lei, S. "An Efficient Color Re-indexing Scheme for Palette-Based Compression" *Proc. of IEEE International Conference on Image Processing*, ICIP-2000, Vancouver, BC., September 2000.