Dual Set Arithmetic Coding And Its Applications To Image Coding

Bin Zhu, Enhui Yang, and Ahmed H. Tewfik
Department of Electrical Engineering, University of Minnesota
Minneapolis, MN 55455, USA
email: binzhu, ehyang, tewfik@ee.umn.edu

ABSTRACT

Arithmetic coding is usually implemented in fixed precision. Such an implementation cannot efficiently code sources, such as image coding algorithms, that locally produce a small fraction of a large alphabet of symbols. In this paper, we propose a novel approach to overcome this inefficiency. The proposed algorithm uses dual symbol sets: a primary symbol set that contains the symbols that have occurred in the recent past and a secondary symbol set that contains all other symbols. Both sets are dynamically adapted to the local statistics. We summarize an analysis of the proposed approach and describe the results that we have obtained by applying it to images.

1 Introduction

Adaptive arithmetic coding is an efficient entropy coding technique widely used to code quantized symbols in image compression. It is characterized by a high compression efficiency and an ability to adapt itself to the local statistics of a source. However, it suffers from a fundamental inefficiency when it is applied to image compression. Image coding usually produces a large source alphabet. Moreover, the distribution of the symbol sequence is highly inhomogeneous in general. When adaptive arithmetic coding is used in such a case, its coding efficiency is greatly impaired because it cannot find a good balance between two conflicting requirements: accurate estimation of the probability of each symbol and good tracking of changes in the source statistics. Reliable estimation of the probabilities of symbols requires that scaling of the symbol occurrence counts be performed infrequently [4]. One the other hand, good tracking of changing statistics requires frequent scaling of these counts.

Furthermore, in image coding the symbols that appear locally are generally a small fraction of the total alphabet. The large number of symbols that do not occur locally distorts the weighted probabilities of local symbols. This further reduces compression efficiency.

In this paper, we propose a novel modified arithmetic coding procedure which does not suffer from the problems mentioned above. The proposed method uses dual symbol sets. A primary symbol set contains all the symbols which have occurred recently. A secondary symbol set contains all other symbols. The primary set is used as the default coding alphabet set. A special symbol is used to escape to the secondary symbol set when the procedure encounters a symbol that is not in the primary set. Both symbol sets are dynamically adapted to the local statistics.

Comparison between the proposed method and the conventional method is also reported in this paper. Simulations show that the proposed method can achieve 6-9% more compression than conventional arithmetic coding [1] when used with a modified JPEG image coding technique, and up to 33% more compression when applied with the wavelet based image coding approach of [2].

2 Limitations of Conventional Arithmetic Coding

Arithmetic coding procedures are implemented in fixed precision, generally in integer arithmetic. The probability of a symbol is estimated by the ratio of its frequency count (i.e., the number of times it occurs) to the total frequency count of all symbols (i.e., the sum of the frequency counts of the individual symbols). Arithmetic coding allows no zero probability. Thus the smallest frequency count associated with a given a symbol is 1, since 1 is the smallest non-zero number that we can represent with integers.

To avoid overflow and underflow, the number of bits f that we use to represent frequency counts and the number of bits p that we use to represent integer words in fixed arithmetic must satisfy certain conditions. Specifically, suppose that we use c bits to represent code values, then we must have [1]:

$$f \leq c-2 \quad (1)$$

$$p \geq f+c. \quad (2)$$

Scaling occurs when the total frequency count reaches its maximum value of 2^f [1]. Scaling introduces a locality of reference. However, it also reduces the reliabil-

ity of the estimated probabilities of individual symbols. For reliable estimation of these probabilities, 2^f must be much larger than the alphabet size.

If the alphabet is large, as is frequently the case in image coding, 2^f must be large too. A larger 2^f means that scaling is performed less often. This in turn implies a reduced ability to track changing source statistics. The slow learning of changing statistics reduces compression efficiency. It is therefore desirable to increase the frequency of scaling while accommodating all symbols.

Furthermore, in image coding, the symbols that appear locally in an image are usually a small fraction of the symbol alphabet. Since the probability of a symbol is estimated by its weighted frequency count and the frequency count for a symbol is at least 1, the large number of symbols that do not occur locally distorts the values of the estimated probabilities of the symbols that appear locally in the image.

3 Dual Set Arithmetic Coding

The proposed arithmetic coding uses dual symbol sets: a primary symbol set and a secondary symbol set. The primary symbol set contains the symbols that have occurred in the recent past plus a special symbol ESC. The secondary symbol set contains all other symbols. The primary symbol set is used as the default symbol set. When the procedure encounters a symbol that is not in the primary symbol set, it encodes ESC first. This indicates that the secondary symbol set will be used to code the current symbol. Symbols can move dynamically from one symbol set to the other to adapt to the local statistics. Specifically, the proposed algorithm operates as follows:

1. Initial State: All symbols except EOF (End Of File) are in the primary symbol set. The symbol EOF is in the secondary symbol set since it is used only once. The symbol ESC always stays in the primary symbol set with frequency count 1. All frequency counts are initialized to 1.

2. (a) If the current symbol is in the primary symbol set, encode the symbol and increase its frequency count by 1. Then go to Step 3.

   (b) If the current symbol is not in the primary symbol set, encode the special symbol ESC. Then use the secondary symbol set to encode the current symbol. The symbol is then moved from the secondary symbol set to the primary set since it is likely to occur in the future.

3. If the sum of all frequency counts in the primary symbol set reaches maximum, the frequency counts associated with all symbols in the primary symbol set are halved and rounded up to integers (the *scaling* process). All symbols with a frequency count of 1 except ESC are moved to the secondary symbol set. We note here that the frequency counts of those symbols which occur recently are larger than 1, and thus stay in the primary symbol set.

4. Go back to Step 2 to code the next symbol.

The decoding procedure proceeds similarly. Several variations of the basic coding and decoding procedures described here are reported in [3].

Observe that the algorithm described above assumes that the source is memoryless, i.e., that symbols are independent. We can easily extend it to deal with Markov sources. The performance margin of the proposed algorithm increases when higher-order Markov models are used. The frequency count of a symbol produced by a Markov source depends on its context. The probability distortions caused by symbols that have not occurred are therefore more pronounced than with memoryless sources.

4 Performance Analysis

Notice that, as compared to the conventional arithmetic coding procedure, our algorithm encodes symbols that are in the primary set more efficiently than those in the secondary set. It is therefore important to quantify the difference in compression that results from using the conventional and proposed arithmetic coding procedures. To this end, let us assume with no loss of generality that the input file is divided into blocks with the size of mth block B_m. Further, let F_m and S_m be the respective primary and secondary symbol sets at the beginning of block m, and let T_m be the total frequency count of symbols in F_m. Denote by V_m the set of symbols in the secondary symbol set which occur in block m, and $c_{a,m}$ the number of occurrences of symbol a in block m. Let the cardinality of a finite set A be $|A|$. Finally, let $A^{\bar{B}}$ represent $A(A+1)\cdots(A+B-1)$, and $A^{-\bar{B}}$ for $A(A-1)\cdots(A-B+1)$. We provide in [3] a proof of the following theorem:

Theorem 1 *The total saving in bits that results from using the proposed procedure, rather than conventional arithmetic coding, to encode a file is given by:*

$$\sum_m \log_2 [\,(T_m + |S_m| - 1)^{\bar{B}_m}\,] - \log_2 [\,T_m^{\bar{B}_m}\,] - \log_2 \prod_{a \in V_m} c_{a,m} - \log_2 [\,|S_m|^{-\overline{|V_m|}}\,]. \qquad (3)$$

If (3) is positive, the proposed algorithm is more efficient than the conventional arithmetic coding. Otherwise, it is less efficient. The difference between the first and second terms in (3) represents the saving that is due to the more accurate probability model used by the proposed algorithm. The last two terms in (3) correspond to the overhead associated with using dual symbol sets. When the cardinality of the secondary set is large and few symbols are moved to the primary set, it is always

positive. This is often the case in image coding, which has been confirmed by the simulations that we shall describe later in this paper.

Theorem 1 also implies that the performance margin that the proposed algorithm enjoys over conventional arithmetic coding increases when f decreases or when the ratio of the size of the total alphabet to that of a specific source increases. Note that (3) does not depend on the statistical properties of the symbols in F_m for block m. It implies that, among all sources that differ from one another in the probabilities of the symbols in F_m for some block m, the relative saving is higher for sources that can be encoded with less bits.

Finally, note that Theorem 1 allows us to predict easily when the dual set arithmetic coding is more efficient in compression than conventional arithmetic coding. For example, suppose that in an imaging coding application we need to encode 30,000 symbols for a 256 by 256 image. Further, assume that the alphabet size is 500, and the actual symbol set of the image is of size 50. We may take $f = 10$ to balance our needs for accurate estimates of probabilities and fast adaptation to the changing underlying statistics. Suppose that 5 symbols are in the secondary symbol set for each block. Then Theorem 1 tells us the savings that result from using dual set arithmetic coding is about 0.64 bits/symbol, or 0.30 bits per pixel for the image. This translates into 20% saving if the bit rate with conventional arithmetic coding is about 1.5 bits per pixel.

We have also derived lower and upper bounds of the proposed method. For details, interested readers are referred to [3].

5 Experimental Results

To gain some insight into the performance of the proposed method, we have applied it to several image coding schemes, along with the conventional arithmetic coding described in [1]. The following dimensionless percentage performance ratio is used to measure the compression enhancement

$$PR = \frac{L_c - L_p}{L_c} \times 100\%, \tag{4}$$

where L_p is the coded file length with the proposed method, and L_c is that with the conventional method. Both methods use exactly the same parameters in the simulations. The original images used in simulations are shown Fig. 1. They are either 256 by 256 or 512 by 512 gray images with 8 bits per pixel.

We first applied both arithmetic coding approaches to the wavelet based coding method described in [2]. The method uses a bottom up approach to track and predict edges in the wavelet domain. The method offers an ideal case for the application of the proposed method, due to its large symbol set and fast changing local statistics. Several tracking approaches as well as different wavelet filters were studied in [2]. Table 1 shows some experimental results on the 256 by 256 image *Zelda* for 5 different coding approaches. The second row is the coding results using the proposed arithmetic coding. The next row in the table is the corresponding results using the conventional one. The performance ratio PR is listed in the last row. As we can see from the table, the proposed method is consistently better than the conventional method, in the range from 12% to about 33%.

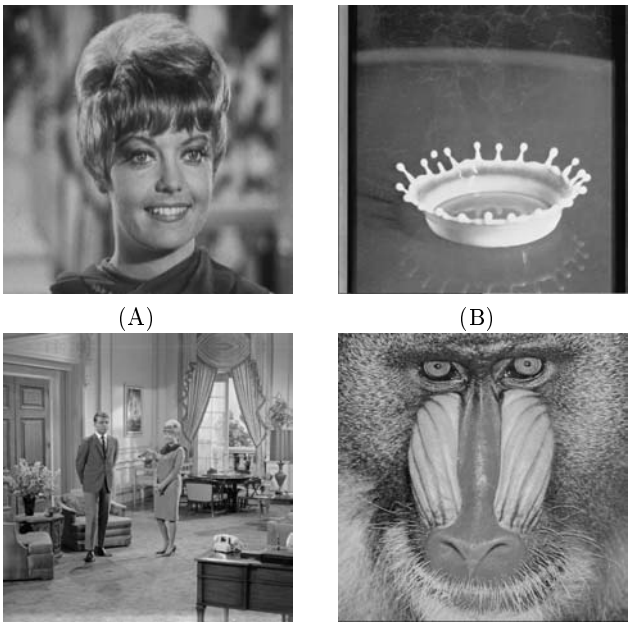

Figure 1: Original images used in the test. (A) Zelda, (B) Milkdrop, (C) Couple, and (D) Mandrill.

Table 1: Compression of 256 by 256 Zelda with the coders in [2]

Approaches	1	2	3	4	5
Dual (bpp)	0.985	1.52	1.48	1.17	0.757
Conv. (bpp)	1.35	1.73	1.77	1.66	1.13
PR (%)	27.1	12.2	19.2	29.2	32.9

Next we applied both methods to the baseline JPEG method [5] whose Huffman coding part was replaced by the proposed method or the conventional arithmetic coding. Recall that JPEG includes two coding parts: the entropy coding part and the "variable length integer " (VRI) part. Table 2 shows the results on the 512 by 512 images shown in Fig. 1. The fourth column of the table (labeled as "Entropy") provides a comparison of the performances of the conventional and proposed arithmetic coding approaches when applied to the entropy coding part of the JPEG procedure. The fifth column compares the two algorithms based on overall bit rates. Note that since arithmetic coding is used only to encode the entropy coding part, a performance com-

parison based on the ratio of the overall bit rates is not a good indicator of performance.

Table 2: **Comparison for JPEG and JPEG-like coders**

Images	JPEG				JPEG-like		
	Conv. (bpp)	Dual (bpp)	PR (%)		Conv. (bpp)	Dual (bpp)	Overall PR (%)
			Entropy	Overall			
Zelda	1.60	1.54	5.70	3.77	1.47	1.36	7.58
	1.05	1.00	6.01	4.07	0.976	0.900	7.77
	0.650	0.621	6.56	4.55	0.617	0.561	9.00
Milkdrop	1.47	1.41	5.80	3.85	1.37	1.27	7.07
	0.922	0.886	5.60	3.84	0.858	0.791	7.83
	0.559	0.535	6.07	4.33	0.521	0.476	8.67
Couple	2.32	2.23	6.07	3.86	2.17	2.03	6.33
	1.55	1.49	5.88	3.86	1.45	1.34	7.10
	1.02	0.979	6.12	4.18	0.945	0.868	8.21
Mandrill	3.54	3.39	6.90	4.25	3.34	3.10	7.10
	2.36	2.47	6.87	4.44	2.31	2.13	7.91
	1.64	1.56	6.96	4.66	1.52	1.38	9.21

To better compare the two arithmetic coding methods, we replaced the VRI part in the baseline JPEG by direct entropy coding of all quantized dc and ac coefficients as well as the run-lengths of zeros. Note that in this case, the symbol alphabet depends on the quantization table used. We always selected a symbol alphabet of minimum size that included all possible symbols. For example, if the scaling factor is chosen to be 0.8, and assume the smallest ac quantization step is $0.8 * 11 = 8.8$, the alphabet used is then [-117, 117]. This is obtained from the division by 8.8 of the ac DCT coefficients, which are in the range from -1024 to 1023 [5]. This is possible since the scaling factor (or more precisely, the quality number) is transmitted to the decoder as side information. The simulations results are also shown in Table 2, which are referred as "JPEG-like". In Table 2, each row gives exactly the same coding images.

From the Table 2, we can see that the proposed method offers better compression for both JPEG and "JPEG-like" image coding methods. For the "JPEG-like" method, the improvement ranges from 6.33% to 9.21%, depending on images and bit rate. When the bit rate is reduced, the performance margin increases with the "JPEG-like" method. We note here that the "JPEG-like" approach is always better than its JPEG counterparts.

We notice that the performance margin for the JPEG or JPEG-like methods is not as large as that for the wavelet based method in [2]. This is because the wavelet based coding method was coded from small scales to large scales. The symbol changes from one scale to another scale is more predictable than the block based JPEG or JPEG-like method. This better prediction yields less hitting of the secondary symbol set, and thus improves performance.

6 Conclusion

We have proposed a novel arithmetic coding method in this paper. The proposed method works considerably better than the conventional arithmetic coding method for typical image compression methods in which the alphabet is large but symbols generated by an image are just small fraction of the alphabet, especially when the source statistics changes significantly. Simulations on several image coding schemes have confirmed this assertion. They all show favorable results for the proposed method.

References

[1] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic Coding for Data Compression," *Communications of the ACM*, Vol. 30, No. 6, PP. 520-540, 1987.

[2] E. Sofie Olsson, "Video Coding for Interactive Multimedia Applications," *Master's thesis*, University of Minnesota, June 1995.

[3] Bin Zhu, Enhui Yang, and Ahmed H. Tewfik, "Arithmetic Coding With Dual Symbol Sets And Its Performance Analysis," Submitted to *IEEE Trans. Image Processing,* Nov. 1995.

[4] Paul G. Howard and Jeffrey S. Vitter, "Analysis of Arithmetic Coding for Data Compression," *Information Processing & Management*, Vol. 28, No. 6, PP. 749-763, 1992.

[5] G. K. Wallace, "The JPEG Still Picture Compression Standard," *Communications ACM*, Vol. 34, no. 4, pp. 31-44, 1991.