ISSN 2959-6157

Analyzing the Trade-offs in Lossless Image Compression Techniques: Insights for Computer Science Research

Eric Ziming Lu

Wellington College International Shanghai, China
*Corresponding author: 24lue@wellington-shanghai.cn

Abstract:
This paper aims to provide a comprehensive analysis of the pros and cons of various lossless image compression algorithms for computer scientists, including RLE, Huffman coding, and LZ77. The pros and cons of different compression methods will be examined by various metrics such as space efficiency, space complexity, and time complexity. Each method will be tested upon various image file types, including BMP, TIFF, PPM, JPG, and PNG. The results indicated that Huffman encoding was particularly effective for PPM images, outperforming RLE and LZ77 with notably higher compression ratios. RLE had slightly higher compression ratios in compressing BMP files. TIFF images exhibit lower compressibility compared to BMP and PPM, but with Huffman encoding still demonstrating superior results. However, when lossless compression algorithms are applied to JPG and PNG images, they yield negative outcomes, indicating that JPG and PNG files have limited compressibility due to prior compression.

Keywords: lossless, image, compression, RLE, Huffman.

1. Introduction

As technology has evolved, social media have seamlessly become a part of our life. Digital image, an indispensable part of social media serving the purpose of communication, has experienced exponential proliferation. This drew attention to various properties of images, such as speed of transmission, storage space required, and overall quality. Image compression techniques have been developed to reduce the size of digital images while preserving their visual quality [1].

There are two categories of image compression: lossless and lossy. Lossless image compression allows reconstructed image to be identical to the original image while shrinking the file size. Lossy compression lowers the file size significantly, but in exchange, loses data permanently, and cannot be reconstructed identically to the original image [2].

This paper will examine lossless compression algorithms including RLE, Huffman coding, and LZ77 through the lens of several factors: space efficiency, space complexity, and time complexity. The research is aimed for assisting computer scientists in selecting the most suitable algorithm for their specific applications.

2. Lossless image compression methods

This section provides an overview of three lossless compression methods: RLE, Huffman coding, and LZ77. Each method will be introduced, explaining their operation and applications.

2.1 RLE

Run length-encoding (RLE) is a simple type of data compression in which sequences of redundant data values are kept as a single value and count rather than each value separately. It works by identifying consecutive redundant symbols and their count, then organizing them into tuples in the format: (symbol, count).

In terms of the three types of lossless image compression algorithms discussed this paper, images are separated into R, G and B layers and processed independently. The R, G, or B value of each pixel represents a symbol in this case. For example, consider the R channel of an image with the pixel values [125,125,125,34,34,255,255,255,255,0,0,0]. Using RLE compression, the list can be represented as [125,3,34,2,255,4,0,3], where every 2 consecutive elements form a tuple defined as: (symbol, count). This method requires repetition upon consecutive symbols, and therefore would be suitable for images with large areas of uniform colors [2].

2.2 Huffman coding

Huffman coding involves assigning unique binary variable-length codes to symbols with the most frequently occurring symbol holding the shortest code, vice versa. It then builds a binary tree based on the symbol's variable-length codes, with frequent symbols having shorter paths from the root [3].

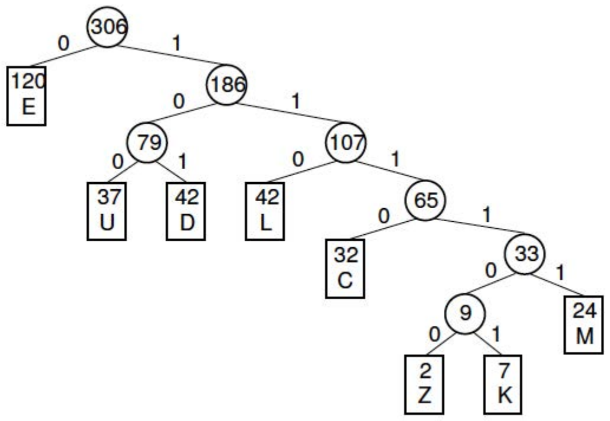

Fig. 1 Huffman tree

Fig 1 shows that Huffman coding is particularly effective when compressing images with unbalanced distribution of pixel values, as shorter codes can be assigned to frequently occurring symbols.

2.3 LZ77

Lempel-Ziv 1977 is one of the milestones of dictionary compression algorithms, invented by Abraham Lempel and Jacob Ziv in 1977 [4]. The algorithm consists of two parts: the sliding window and the dynamic dictionary. The sliding window represents a fixed-length buffer that moves along the data stream as it is being processed. The dynamic dictionary stores previously encountered phrases in a tuple in the format: (offset, run length, deviating character), where:

-The offset is the number of characters away from the start of the window.

-The run length is how many characters that should be read forward, starting from the offset.

-The deviating character indicates that a new character with unrecognized pattern has been found [5].

This algorithm is particularly advantageous identifiable patterns occur frequently in the processed image, allowing it to represent long streams of patterns with only one tuple.

3. Evaluation metrics

This section is a review of the metrics that will be used to evaluate the effectiveness of various compression algorithms, including: space efficiency, space complexity, and time complexity.

Space efficiency refers to the amount of storage space reduced in the context of image compression. We typically quantify space efficiency with compression ratio, calculated by , where denotes compression ratio [6]. It is a crucial factor in optimizing storage capacity and transmission bandwidth. Space complexity measures the amount of memory required for completing an algorithm, calculated by the formula: [7], where: is the space required by the algorithm itself that remains constant, excluding input data. is the space used by the algorithm to store data. Time complexity on the other hand, measures the time required for an algorithm to finish running, excluding hardware-related restrictions. It can be calculated by summing up the number of operations performed in an algorithm. The sum of both space and time complexity are expressed using the big-O notation to showcase the theoretical maximum memory that the algorithm can consume [7]. Common types of space and time complexity include: O(1) indicating constant complexity; O(n) indicating linear complexity; O(log(n)) indicating logarithmic complexity; O(n log(n)) indicating linearithmic complexity; O(n^2) indicating polynomial complexity; O(2^n) indicating exponential complexity; O(n!) indicating factorial complexity [8].

4. Performance analysis

To ensure fairness of experiment, a variety of image types that don't inherently include compression by default will be evaluated, such as TIFF, BMP, and PPM. Images encoded by PNG and JPG are as well included, but it is anticipated that the efficacy of compression will be diminished, given their pre-existing substantial compression.

4.1 Results

Table 1. Compression ratio of TIFF images

	Compression ratio			
Image compression methods	A	B	C	D
RLE	1.00	1.00	1.00	1.00
Huffman encoding	1.12	1.03	1.04	1.03
LZ77	1.04	0.78	0.82	0.83

Table 2. Compression ratio of BMP images

	Compression ratio			
Image compression methods	E	F	G	H
RLE	1.37	1.33	1.33	0.84
Huffman encoding	1.29	1.27	1.29	1.23
LZ77	1.25	1.22	1.25	1.67

Table 3. Compression ratio of PPM images

	Compression ratio			
Image compression methods	I	J	K	L
RLE	2.00	2.00	1.90	2.00
Huffman encoding	7.55	11.89	6.43	6.36
LZ77	7.21	5.02	1.81	2.33

Table 4. Compression ratio of JPG images

	Compression ratio			
Image compression methods	M	N	O	P
RLE	0.17	0.33	0.27	0.12
Huffman encoding	0.60	1.08	0.84	0.27
LZ77	0.27	0.30	0.29	0.16

Table 5. Compression ratio of PNG images

	Compression ratio			
Image compression methods	Q	R	S	T
RLE	0.20	0.26	0.34	0.25
Huffman encoding	0.88	1.27	1.54	1.14
LZ77	0.31	0.67	0.52	0.35

Table 6. Space and time complexity of various image compression algorithm

	Image compression methods		
Complexity	RLE	Huffman encoding	LZ77
Space complexity			
Time complexity			

4.2 Compression ratio analysis

This section analyzes the performance of lossless compression methods when applied to various image formats based on the results.

Tabel 1 shows that for TIFF images, Huffman encoding was relatively the most effective algorithm. RLE has a compression ratio of 1.00 for all the images being tested, meaning that RLE isn't effective upon TIFF. LZ77 performed poorly, with its compression ratio ranging from

0.83 to 1.04. Overall, the three compression methods being tested performed below expectations, with poor, ineffective and insignificant compression ratios.

We can see from Table 2, the three algorithms performe decently for BMP images, and RLE performed the best. Excluding the anomaly at component H, RLE had an average compression ratio of 1.34. Following up, Huffman encoding scored an average of 1.27. LZ77 scored the worst, but with no significant disadvantage, with an average of 1.24, excluding the anomaly image H. It is worth noting that the marginal variance in compression ratios among the three algorithms suggests that the outcomes may not be entirely conclusive.

Table 3 illustrates that the PPM file format exhibits highest compression ratio among the five file types. All three compression algorithms performed well. Huffman encoding was the most effective, with compression ratio around 6 to 7. LZ77 comes next, but with varying values, indicating that its performance is sensitive to the characteristics of the PPM images being compressed. RLE demonstrated the lowest but most stable compression ratio upon PPM, with an average of 1.98.

In Table 4, the three compression algorithms were ineffective when applied to JPG images. RLE performed poorly, with compression ratios ranging from 0.12 to 0.33. LZ77 fared no better, with compression ratio ranging from 0.16 to 0.30. The compression ratio of Huffman encoding varied inconsistently, from 0.27 to 1.08. Although Huffman encoding had relatively higher compression ratios, they were still inadequate.

There was no significant difference between RLE and LZ77 when applied to PNG or JPG, from Table 5. However, Huffman encoding performed slightly better when applied to PNG, with compression ratios ranging from 0.88 to 1.54. Despite Huffman encoding performed much better when compared to JPG, the compression ratios are unevenly distributed, again suggesting that the compression ratio heavily depends on the actual characteristics of given PNG image. The three methods were ineffective amongst JPG and PNG files, validating the expectations mentioned at the start of section 4.

4.3 Evaluations

The methods showed significant variation in compression ratio across a variety of tested images. Certain algorithms proved to be successful on particular file types.

BMP and PPM are the most compressible amongst the selected file types. Huffman encoding outperformed RLE and LZ77 in the case of PPM, while RLE demonstrated slight advantages in compressing BMP files, yielding higher compression ratios. TIFF is observed to be less compressible compared to BMP and PPM formats, with Huffman encoding once more exhibiting the best results. Compression on JPG and PNG images had negative effects when RLE and LZ77 were applied, suggesting that they had been already compressed to an extent where further compression with lossless image compression techniques is unachievable. This observation also applies to Huffman encoding with JPG. The only exception was when PNG was encoded using Huffman, resulting in reasonably good compression ratios, although the results differed depending on the tested image.

From Table 6, we find that despite the advantage on compression ratio, Huffman encoding incorporates a time complexity of , indicating that it may not be the most efficient option for real-time applications where speed is paramount. This does have a great effect, as when running the code, Huffman encoding did take much longer to run when compared to RLE and LZ77. Therefore, it would be reasonable to consider applying RLE to PPM as an alternative. LZ77, however had no conspicuous advantage over RLE or Huffman encoding according to the results of experiments, with poor compression ratio, and higher time complexity compared to RLE. This is perhaps why LZ77 has been replaced by improved modern compression algorithms such as LZW.

5. Conclusion

This paper presents a comprehensive analysis of the three lossless image compression algorithms: RLE, Huffman encoding, and LZ77. The algorithms were evaluated based on their space efficiency, space complexity, and time complexity. The results show that the performance of the algorithms varied significantly depending on the type of image being compressed. For PPM images, Huffman encoding is the most effective algorithm. For BMP images or images with varying characteristics, RLE is the most stable and effective algorithm, with also the lowest time complexity . LZ77 is not as effective as RLE or Huffman coding for any of the image types tested.

In terms of the actual compression speed, RLE is the fastest algorithm, followed by LZ77 and then Huffman encoding. This corresponds to the time complexity being analyzed, where RLE has linear complexity , LZ77 has complexity , and Huffman encoding with complexity . In terms of space complexity, the three algorithms are equivalent, with linear complexity .

Based on the study, it is reasonable to conclude that Huffman encoding is preferably the best algorithm for compressing images in the format of PPM. RLE would be the next considerable choice, as it exhibits a lower time complexity with stable and acceptable compression ratio, along with slightly better compression ratio for BMP images. LZ77 is not in advantage in any of the file types being tested, and therefore is not recommended.

References

[1]E. A. B. da Silva and G. V. Mendonca. Digital Image Processing in The Electrical Engineering Handbook, 2005.

[2]K. Kaur, J. Saxena, and S. Singh. Image Compression Using Run Length Encoding (RLE). International Conference on Computing Communication and Automation (ICCCA), 2017, 1280-1285.

[3]W. Z. Wahba and A. Y. A. Maghari. Lossless Image Compression Techniques Comparative Study. International Conference on Engineering Technology and Applied Sciences (ICETAS), 2016, 1-5.

[4]J. Ziv and A. Lempel. A universal algorithm for sequential data compression, in IEEE Transactions on Information Theory, 23(3), 337-343, 1977.

[5]P. Shi, B. Li, P. H. Thike, and L. Ding. A knowledge-embedded lossless image compressing method for high throughput corrosion experiment. International Journal of Distributed Sensor Networks, 14(1), 2018.

[6]Pu, Ida Mengyi. Fundamental Data Compression, Butterworth-Heinemann, 2006.

[7]Algorithm Examples. Best Methods to Determine Time and Space Complexity. Algorithm Examples, [Online].

[8]GeeksforGeeks. Time Complexity and Space Complexity. GeeksforGeeks, [Online].