

An Alternate Way to Develop Lossless Graphical Data Compression Package using Non-Linear Single Cycle Multiple Attractor Cellular Automata

Anirban Kundu^{1,6}, Alok Ranjan Pal², Tanay Sarkar³, Arijit Samanta¹, Nirupam Chakraborty¹, Subhendu Mandal¹, Sutirtha Guha⁴, Debajyoti Mukhopadhyay^{5,6}

¹Netaji Subhash Engineering College (West Bengal University of Technology),
Kolkata, West Bengal 700152, India

²College of Engineering & Management, Kolaghat, West Bengal 721171, India

³Mu Sigma Business Solutions Pvt. Ltd., Bangalore, Karnataka 560042, India

⁴Seacom Engineering College (West Bengal University of Technology),
Howrah, West Bengal 711302, India

⁵Calcutta Business School, Diamond Harbour Road, Bishnupur, West Bengal 743503, India

⁶Web Intelligence & Distributed Computing Research Lab, Kolkata, West Bengal 700095, India
{anik76in, chhaandasik, tanay.sarkar, arijitsamanta8, nirupam2u, subhendu.mndl, sutirthaguha,
debajyoti.mukhopadhyay@gmail.com

doi: 10.4156/jcit.vol4.issue2.kundu

Abstract

Compression is one of the major topics in the field of research as it is used for storing data in a lesser storage space of the repository. Compression is furthermore required in case of sending data from source to destination through network (internet or intranet). The time required is less in case of sending fewer amounts of data. So, compression is a necessity for transmission of original huge data as a modified small amount of data with a low overhead. Image compression is the application of data compression on digital images. This research work takes into consideration an image as the source file (.bmp) and converts it to the standard PNM format and further in plain PNM format (raw file) to utilize the binary format specification used in PNM. Single Cycle Multiple Attractor Cellular Automata (SMACA) is used as a compression tool. The results achieved through experimentation are based on lossless image compression. The first pass compresses the PNM image using SMACA based techniques followed by the second pass which utilizes Huffman variable length encoding scheme. Initially, 31% (approx.) & 95% (approx.) compression ratio are achieved using SMACA structure on the PNM & raw files respectively. Final compressed data of the image obtained achieves a further compression by 6% over the previous compressed data using the Huffman algorithm.

Keywords

Image Compression, PNM Image, Lossless Compression, Rule Min Term (RMT), Cellular Automata (CA), Multiple Attractor Cellular Automata (MACA), Single Cycle Multiple Attractor Cellular Automata (SMACA)

1. Introduction

The modern day Internet is a rich kaleidoscope of text, pictures, animation and the like. What this translates to is the fact that any user, while viewing a modern interactive Web page, has to download voluminous quantities of information on to his computer. Even though Internet connections now-a-days are substantially faster than their counter parts half a decade ago, caution must still be exercised when designing Web pages in order to prevent file sizes from becoming too large and bogging down the user experience. Hence, the concept of compression comes into the picture [1-6].

It is generally true that even for a rich Web experience, moderate quality in terms of pictures and video is more than adequate at most times. Hence compression proves to be an easy and implement able solution to this major stumbling block. Image compression is definitely of prime importance among the different types of compressions since almost all Web pages contain at least one image. As it is, image

compression can be lossless or lossy, the latter achieving smaller file size at the expense of quality. However, modern approaches to lossless compression have the possibility of achieving an easier compromise between file size and quality [7-9].

The paper herein tries to achieve lossless compression to a new degree. Lossless compression works with compressing data which, when decompressed, will be an exact replica of the original data. Multiple Attractor Cellular Automata (MACA) and/or Single Cycle Multiple Attractor Cellular Automata (SMACA) form the backbone on which the proposed compression technique is based and we also use the Huffman encoding algorithm which rather than using a fixed number of bits to represent component values, uses variable length codes. The more frequently used values are assigned shorter codes. The final output is obtained after the two passes, first using MACA / SMACA as required and then Huffman algorithm.

In Section 2, we present Cellular Automata preliminaries. Then, we present image compression related work in Section 3. In Section 4, proposed methodology is described along with algorithms. In Section 5, experimental results are shown. We conclude our work in Section 6 and finally Section 7 is for future strategy.

2. Cellular Automata Preliminaries

A cellular automaton or CA is a mathematical ‘machine’ or ‘organism’ that lends itself to some very remarkable and beautiful ideas [16]. One of the most exciting aspects of this area of mathematics is that cellular automata arise from very basic mathematical principles. Though they are remarkably simple at the start, CAs had a variety of applications and can show up in unexpected places. From simple games of chance to computer simulations of biology to art, we will see that these simple ideas are evidence that mathematics is very real and is all around us [11-12].

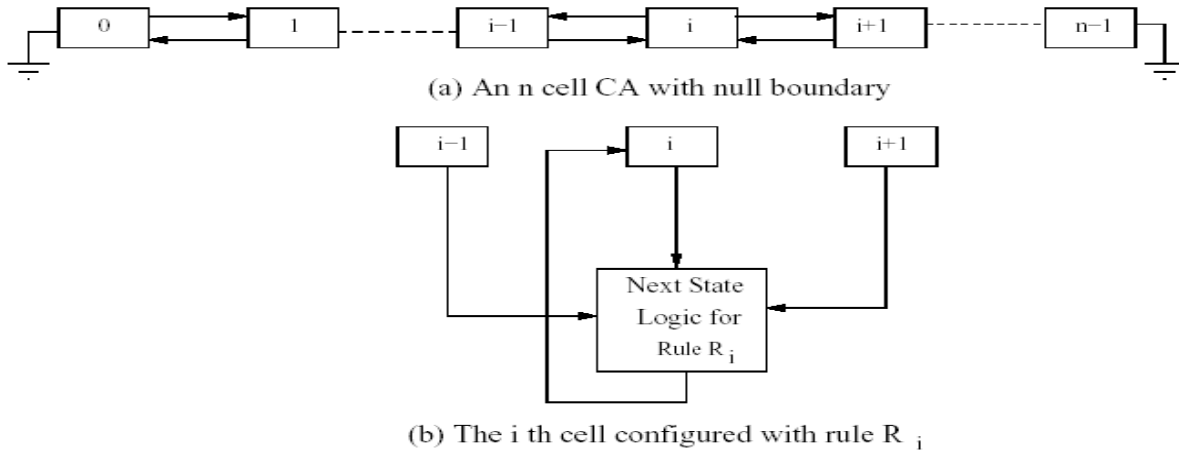
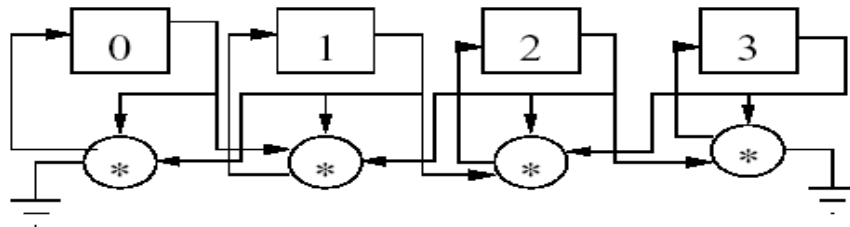


Figure 1. Local Interaction between Cellular Automata Cells

An ‘n’ cell CA consists of ‘n’ cells (Figure 1(a)) with local interactions. It evolves in discrete time and space. The next state function of three neighborhood CA cell (Figure 1(b)) can be represented as a rule as defined in Table 1 [13]. First row of Table 1 represents $2^3=8$ possible present states of 3 neighbors of i^{th} cell - ($i-1$), i , ($i+1$) cells. Each of the 8 entries (3 bit binary string) represents a minterm of a 3 variable boolean function for a 3 neighborhood CA cell. Figure 2 shows the structure (Figure 2(a)) and state transition behavior (Figure 2(b)) of a CA having the Rule Vector as <6 240 60 65>.

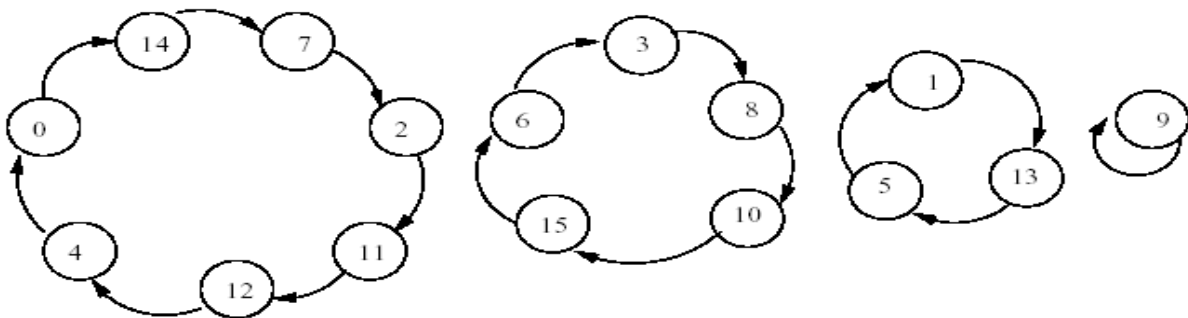
In subsequent discussions, each of the 8 entries in Table 1 is referred to as a Rule Min Term (RMT). The decimal equivalent of 8 minterms are 0, 1, 2, 3, 4, 5, 6,

7 noted within () below the three bit string. Each of the next five rows of Table 1 shows the next state (0 or 1) of i^{th} cell. Hence there can be $2^8=256$ possible bit strings. The decimal counterpart of such an 8 bit combination is referred to as a CA rule [13]. The rule of a CA cell represents its next state logic as illustrated in Table 2 for a few example rules. It can be derived from the truth table (Table 1) of the i^{th} cell, where q_i^{t+1} is the next state of i^{th} cell, while q_{i-1}^t , q_i^t , and q_{i+1}^t are the current states of ($i-1$)th, i^{th} , and ($i+1$)th cells respectively; the \oplus represents XOR logic and \bullet denotes AND function. If a CA cell is configured with a specific rule, its next state function implements the truth table as illustrated for sample rules in Table 2.



Note : * denotes the next state logic corresponding to rule R_i ($i = 0, 1, 2, 3$) in the Rule Vector $\langle R_0 R_1 R_2 R_3 \rangle$ for i^{th} cell

(a) CA structure



(b) State transition behavior displaying four cycles of length 7, 5, 3, and 1

Figure 2. State Transition Behavior of Cellular Automata Rule Vector $\langle 6\ 240\ 60\ 65 \rangle$

Table 1. Truth Table of sample rules of a CA cell showing the next state logic for the Minterms of a 3 variable boolean function - The 8 minterms having decimal values 0, 1, 2, 3, 4, 5, 6, 7 are referred to as Rule Minterms (RMTs)

Note: Set of Minterms $T = \{7, 6, 5, 4, 3, 2, 1, 0\}$ represented as $\{T(7), T(6), T(5), T(4), T(3), T(2), T(1), T(0)\}$ ($T(m) = m, m = 0$ to 7) in the text, are noted simply as q.

Present states of 3-neighbours ($i - 1$), i , and ($i + 1$) of i^{th} cells (Minterms of a 3 variable boolean function)	111 (7) T(7)	110 (6) T(6)	101 (5) T(5)	100 (4) T(4)	011 (3) T(3)	010 (2) T(2)	001 (1) T(1)	000 (0) T(0)	Rule Number
Next state of i^{th} cell	0	1	0	1	1	0	1	0	90
	1	0	0	1	0	1	1	0	150
	0	1	1	1	1	0	0	0	120
	0	0	0	0	1	1	0	0	12
	1	1	0	1	0	0	1	0	210

The first two rules 90 and 150 of Table 2 are linear rules employing XOR logic while remaining non-linear rules employ AND logic in addition to XOR. Out of 256 possible rules, as shown in Table 3, there are 7 rules with XOR logic and another 7 rules employ

XNOR logic. The Rule 0 sets the cell to state '0' for each of the 8 minterms. The remaining rules are non-linear rules [14-15] employing AND/OR/NOT logic. Linear and Additive CA employing XOR/XNOR logic have been characterized with matrix algebraic

formulation [16]. In general, CAs can be utilized in any type of pattern generation and/or pattern recognition real-time systems. Some basic definitions which are

the mandatory requirement for this paper are depicted hereafter in Sub-Section 2.1.

Table 2. Next state logic of a few rules

Note: \oplus represents XOR while \odot denotes AND logic functions.

$$\begin{aligned} \text{Rule 90 : } q_i^{t+1} &= q_{i-1}^t \oplus q_{i+1}^t \\ \text{Rule 150 : } q_i^{t+1} &= q_{i-1}^t \oplus q_i^t \oplus q_{i+1}^t \\ \text{Rule 120 : } q_i^{t+1} &= q_{i-1}^t \oplus (q_i^t \odot q_{i+1}^t) \\ \text{Rule 12 : } q_i^{t+1} &= q_i^t \oplus (q_{i-1}^t \odot q_i^t) \\ \text{Rule 210 : } q_i^{t+1} &= q_{i-1}^t \oplus q_{i+1}^t \oplus (q_i^t \odot q_{i+1}^t) \end{aligned}$$

Table 3. Linear/additive CA rules employing Next State Function with XOR/XNOR logic

Note: Rule 0 sets the cell to state '0' for each of the 8 minterms

Rule No.	Next state function with XOR logic	Rule No.	Next state function with XNOR logic
Rule 60	$q_i(t+1) = q_{i-1}(t) \oplus q_i(t)$	Rule 195	$q_i(t+1) = \overline{q_{i-1}(t) \oplus q_i(t)}$
Rule 90	$q_i(t+1) = q_{i-1}(t) \oplus q_{i+1}(t)$	Rule 165	$q_i(t+1) = \overline{q_{i-1}(t) \oplus q_{i+1}(t)}$
Rule 102	$q_i(t+1) = q_i(t) \oplus q_{i+1}(t)$	Rule 153	$q_i(t+1) = \overline{q_i(t) \oplus q_{i+1}(t)}$
Rule 150	$q_i(t+1) = q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)$	Rule 105	$q_i(t+1) = \overline{q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)}$
Rule 170	$q_i(t+1) = q_{i+1}(t)$	Rule 85	$q_i(t+1) = \overline{q_{i+1}(t)}$
Rule 204	$q_i(t+1) = q_i(t)$	Rule 51	$q_i(t+1) = \overline{q_i(t)}$
Rule 240	$q_i(t+1) = q_{i-1}(t)$	Rule 15	$q_i(t+1) = \overline{q_{i-1}(t)}$

2.1 Definitions

Definition 2.1: Group CA - Each state in the state transition behavior of a group CA has only one predecessor and consequently each state is reachable from only one state. A group CA traverses all the states in a cycle. A group CA is a reversible CA in the sense that the CA will always return to its initial state.

Definition 2.2: Non-group CA - A non-group CA has states that have \$r\$ number of predecessors, where $r = 0, 1, 2, 3, \dots$

Definition 2.3: Reachable state - A state having 1 or more predecessors is a reachable state.

Definition 2.4: Non-reachable state - A state having no predecessor (i.e., $r = 0$) is termed as non-reachable.

Definition 2.5: Cyclic state - A state in a cycle of the state transition behavior of a CA. Cyclic states are also referred to as Stable (semi stable) states.

Definition 2.6: Transient state - A non-cyclic state of a non-group CA is referred to as a transient state.

Definition 2.7: Attractor Cycle - The set of states in a cycle is referred to as an attractor cycle.

Definition 2.8: Self-Loop Attractor (SLA) - A single cycle attractor state with self-loops is referred to as SLA.

Definition 2.9: Multiple Attractor Cellular Automata (MACA) - A non-group CA containing one or more number of cycles, each having two or more length, and also have some single length cycles is known as MACA. Thus, a MACA means a non-group CA structure which has some single length cycles as well as some non-single length cycles.

Definition 2.10: Single Cycle Multiple Attractor Cellular Automata (SMACA) - A non-group CA having one or more number of single length cycles and not having two or more length cycles, is defined as SMACA. Thus, SMACA consists of only self-loop attractors (SLAs) along with some non-reachable and transient states.

Definition 2.11: Rule Vector (RV) - The sequence of rules $\langle R_0 R_1 \dots R_i \dots R_{n-1} \rangle$, where i^{th} cell is configured with rule R_i .

3. Related Work

Image compression justifies its importance based on the fact that it is imperative if one wishes to keep the size of modern day Web applications within reasonable limits.

Image compression is minimizing the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. The reduction in file size allows more images to be stored in a given amount of disk or memory space. It also reduces the time required for images to be sent over the Internet or downloaded from Web pages.

There are several different ways in which image files can be compressed. For Internet use, the two most common compressed graphic image formats are the 'JPEG' format and the 'GIF' format. The JPEG method is more often used for photographs, while the GIF method is commonly used for line art and other images in which geometric shapes are relatively simple.

Other techniques for image compression include the use of fractals and wavelets. These methods have not gained widespread acceptance for use on the Internet as of this writing. However, both methods offer promise because they offer higher compression ratios than the JPEG or GIF methods for some types of images. Another new method that may in time replace the GIF format is the PNG format.

A text file or program can be compressed without the introduction of errors, but only up to a certain extent. This is called lossless compression. Beyond this point, errors are introduced. In text and program files, it is crucial that compression be lossless because a single error can seriously damage the meaning of a text file, or cause a program not to run. In image compression, a small loss in quality is usually not noticeable. There is no "critical point" up to which compression works perfectly, but beyond which it becomes impossible. When there is some tolerance for loss, the compression factor can be greater than it can when there is no loss tolerance. For this reason, graphic images can be compressed more than text files or programs [17-18].

Lossless and lossy compression are terms that describe whether or not, in the compression of a file, all original data can be recovered when the file is uncompressed. With lossless compression, every single bit of data that was originally in the file remains after the file is uncompressed. All of the information is completely restored. This is generally the technique of choice for text or spreadsheet files, where losing words or financial data could pose a problem. The Graphics

Interchange File (GIF) is an image format used on the Web that provides lossless compression [19-20].

On the other hand, lossy compression reduces a file by permanently eliminating certain information, especially redundant information. When the file is uncompressed, only a part of the original information is still there (although the user may not notice it). Lossy compression is generally used for video and sound, where a certain amount of information loss will not be detected by most users. The JPEG image file, commonly used for photographs and other complex still images on the Web, is an image that has lossy compression. Using JPEG compression, the creator can decide how much loss to introduce and make a trade-off between file size and image quality [21-22].

While work on image compression methods is ongoing at various research labs, the scope for related work in this field is considerable nonetheless. Improvement is a process that never stops and making the compression algorithm even more efficient is entirely plausible. Furthermore, a reliable lossless compression method can be applied to various other fields as well, albeit with the necessary modification. Hence different related content such as text, animations and even video can possibly use modified cousins of the work highlighted herein to achieve compressions to an impressive scale. While Internet connection speeds are increasing manifold every day, the Web pages are getting just as complex and most new Web content is considerably more resource hungry. Hence the need for efficient compression cannot be undermined, especially in the case of lossless compression which preserves quality while reducing size.

4. Proposed Methodology

This current work aims to achieve standard image compression through the use of Cellular Automata concept which is something different from the typical compression schemes. The Huffman Entropy encoding algorithm forms the back-end model of the proposed compression system. Consequently, output files generated after the use of Cellular Automata serves as the input files on which the Huffman algorithm is used. The resultant output gives us the compressed image. The functional details, organized into different modules are elaborated below.

MODULE 1: Module 1 is mainly concerned with conversion of the BMP image used as the source image into the plain PNM format and converting the entire file to binary code for future processing.

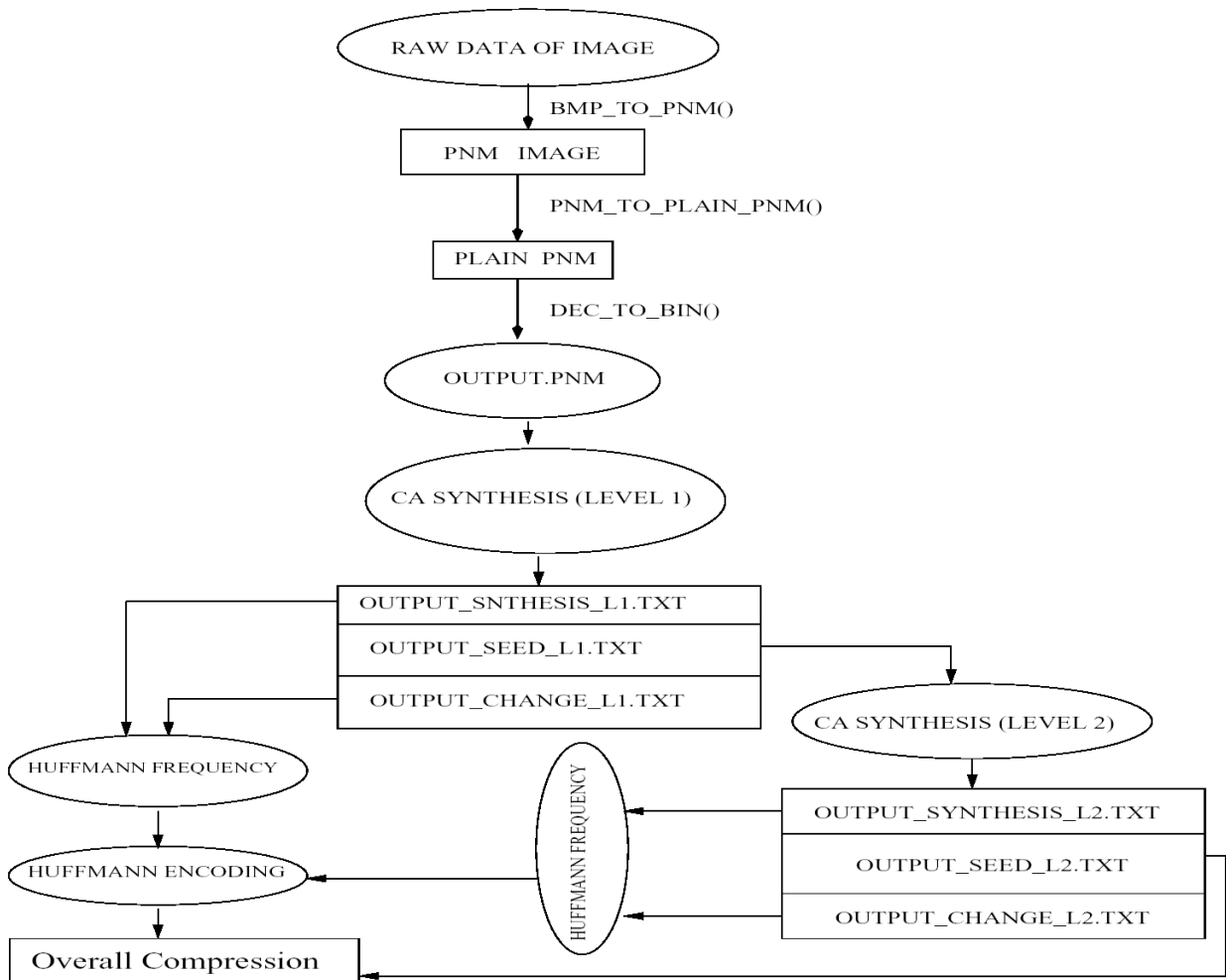


Figure 3. Pictorial Representation of our approach

The BMP image has been converted to a PNM image using user defined BMP_to_PNM function. Then, the PNM image has further been converted to 'Plain PNM' image using a PNM_to_PLAINPNM function. The DEC_to_BIN function code has been applied on the 'Plain PNM' file to convert it to an output file containing binary values.

MODULE 2: Module 2 is the Synthesis functional code which is applied over the output file of the Module 1 to obtain three output files as follows.

- (a) output_synthesis_L1.txt
- (b) output_seed_L1.txt
- (c) output_change_L1.txt

Using Cellular Automata (CA) state transition behavior, a set of n-cell rule vectors (RVs) are generated by following the binary patterns of the above mentioned file. At any time instance, each pattern is considered as the current state and the immediate next pattern is treated as its next state. Thus, using this

“current state - next state” relationship, the RMT of the concerned n-cell CA has been created step-by-step. “output_synthsis_L1.txt” file stores all these n-cell RVs as the compressed form of several states (patterns). “output_seed_L1.txt” file consists of respective seed values (starting pattern(s)) of the CAs. Here, seed means the starting state value of the specific state transitions (chain of states) of the CA. These seed values are essential for getting back all the patterns at the time of decompression. This is all about best case scenario. But, in general, at the time of state transition calculation, there may be some conflicts with respect to bit calculation between current state and next state while generating RMT of the concerned CA. So, these conflicted bit values (0/1) of the specific states are written within “output_change_L1.txt” as a precaution to get the exact original patterns at the time of decompression.

An Alternate Way to Develop Lossless Graphical Data Compression Package using Non-Linear Single Cycle Multiple Attractor Cellular Automata

Anirban Kundu, Alok Ranjan Pal, Tanay Sarkar, Arijit Samanta, Nirupam Chakraborty, Subhendu Mandal, Sutirtha Guha, Debajyoti Mukhopadhyay

MODULE 3: Module 3 is actually the functional code of level 2 of the Synthesis part and it has been applied over output_seed.txt to obtain further three output files as follows.

- (a) output_synthesis_L2.txt
- (b) output_seed_L2.txt
- (c) output_change_L2.txt

In this level of experimentation (Module 3), further compression of the seed patterns is achieved using the same synthesis procedure depicted in 'Module 2'. Hence, this level is called as the 2nd level of synthesis procedure. Basically, to reduce the size of "output_seed_L1.txt", the synthesis scheme is highly required at this level since the "output_seed_L1.txt" contains the binary (0/1) values as the starting state values of the CAs. These seed values are actually the non-reachable states of the CAs in 'Module 2'. But, in 'Module 3', for the sake of compression, these state (seed) values are considered as consecutive state values of any CA. Thus, RMT creation of n-cell CA has been achieved. Hence, "output_synthesis_L2.txt" contains the RVs in a similar fashion like "output_synthesis_L1.txt". All the seed values are stored in "output_seed_L2.txt" and all the bitwise conflicts between the consecutive states are marked in "output_change_L2.txt" within the 'Module 3' also.

MODULE 4: Initially, Huffman frequency has been calculated in 'Module 4' using the following files.

- (a) output_synthesis_L1.txt
- (b) output_change_L1.txt
- (c) output_synthesis_L2.txt
- (d) output_change_L2.txt

Huffman encoding is then applied to these files to get a variable length patterns to represent the previously mentioned fixed length patterns resulting in a compression. So, the output_synthesis_L1.txt and output_change_L1.txt both are operated by the functional code of Huffman frequency to form the files output_synthesis_L1_frequency.txt and output_change_L1_frequency.txt respectively. Similarly, output_synthesis_L2.txt and output_change_L2.txt are also operated by the same functional code of Huffman frequency to produce output_synthesis_L2_frequency.txt and output_change_L2_frequency.txt respectively for further processing using Huffman coding scheme which takes place where data input is taken from the previously mentioned frequency related files and the corresponding output file result.txt is formed (refer Figure 3).

In Computer Science and Information theory, Huffman Coding is an entropy encoding algorithm used for lossless data compression at the majority. Huffman was able to design the most efficient

compression method that would produce a smaller average output.

Huffman Coding today is often used as a 'back-end' to some Compression method. Multimedia codes like JPEG and MP3 have a front-end model followed by Huffman Coding.

Algorithm 1: Image to Data Conversion

Input: .bmp file

Output: .pnm file

- Step 1: Invoke BMP_to_PNM function to convert .bmp file into .pnm file
- Step 2: Invoke PNM_to_PLAINPNM function to convert .pnm file into plain PNM file
- Step 3: Invoke DEC_to_BIN function to convert the decimal values of plain PNM format into binary (0/1) values
- Step 4: Stop

The BMP image is taken as input initially. The function BMP_to_PNM is invoked which converts the image into a PNM image. Once this has been achieved, the function PNM_to_PLAINPNM is invoked on the PNM image in order to convert it to a plain PNM image. The plain PNM image is now subject to the DEC_to_BIN function to change all decimal values to their binary counterparts. The final output comes as a plain PNM image with binary values (refer Algorithm 1).

Algorithm 2: Compression Scheme using CA

Input: Output of Algorithm 1

Output: Three files containing synthesis data, seed (non-reachable states) and change of bit patterns respectively

- Step 1: Take a pattern as a non-reachable state (NRS) as 'state 1' or seed
- Step 2: Loop start: while (! feof (fp))
- Step 3: Take next pattern as transient state as 'state 2'
- Step 4: Call state_transition_function () as discussed in section 2
- Step 5: If ((conflict does not occur) && (count<threshold))
- Step 6: Then 'state 1' = 'state 2' && go to Step 2
- Step 7: if ((conflict occurs) && (count<threshold))
- Step 8: Then change of specified bit within current state ('state 2') && 'state 1' = 'state 2' && go to Step 2
- Step 9: If (count == threshold)
- Step 10: Then call RMT_creation_function ()

Step 11: RMT to Rule Vector (RV) conversion to generate corresponding MACA / SMACA
 Step 12: Stop

The binary file output from the first algorithm is taken as input. The first pattern is chosen as the initial 'Non reachable state'. Now the entire output file from the first algorithm is traversed to the end and all the bit patterns analyzed. Each pattern is first considered as a 'transient state'. The state_transition_function () is called on the given pattern. If a conflict does not occur and the count is below the threshold, then it is assigned to the "Non reachable state". If a conflict occurs, the specified bit within the transient state is first changed and then it is assigned to the "Non Reachable State" set. After all the bit patterns have been matched, if the counted value reaches the threshold, the RMT_creation_function () is invoked. Subsequently, RMT to Rule Vector conversion takes place in order to generate the corresponding MACA / SMACA. The output involves three files, Synthesis Data (containing MACA / SMACA generated), seed (containing non reachable states) and Change of bit patterns undertaken during the program (refer Algorithm 2). Algorithm 2 is used twice in the overall procedure as discussed previously within this section (refer 'Module 2' & 'Module 3').

5. Experimental Results

In this research work, the lossless compression is done on the image files. Minimum 35% lossless compression has been achieved on the original image file and 96% lossless compression has been achieved on the raw data files which are discussed in this section. A case study is shown within the paper for better understanding within this experimental section. Figure 4 shows an arbitrary image file which is being compressed further to show the analysis. Initially, the input file (refer Figure 4) is modified into raw data format. Then, the file is compressed using CA structure as discussed in previous section using two different levels as mentioned in Table 4 & Table 5. Here, Table 4 shows the 1st level of compression in which different threshold values are used to get better compression ratio. The best case (maximum compression) has been chosen for the next level of compression using the same procedure on the specific file as mentioned in Table 4 and the compression achieved in the 2nd level is depicted in Table 5. Further, Huffman encoding scheme is introduced to compress the already compressed files up to some extent using well known variable length concept.

Table 4. Level1: Compression based on different threshold values

File Name	Threshold = 100	Threshold = 150	Threshold = 200	Threshold = 250
output_synthesis_L1.txt	45 KB	30 KB	22 KB	18 KB
output_seed_L1.txt	82 KB	81 KB	79 KB	80 KB
output_change_L1.txt	452 KB	460 KB	467 KB	469 KB
Total	579 KB	571 KB	568 KB	567 KB

Table 5. Level2: Compression on selected threshold of Level1

File Name	Compressed Value
output_synthesis_L2.txt	18 KB
output_seed_L2.txt	14 KB
output_change_L2.txt	36.84 KB
Total	68.84 KB

An Alternate Way to Develop Lossless Graphical Data Compression Package using Non-Linear Single Cycle Multiple Attractor Cellular Automata

Anirban Kundu, Alok Ranjan Pal, Tanay Sarkar, Arijit Samanta, Nirupam Chakraborty, Subhendu Mandal, Sutirtha Guha, Debajyoti Mukhopadhyay

As per Figure 4, input file size = 565.6 KB
Converted file size = 9.4 MB (raw data file)

From Table 4, the best case scenario has been chosen. Here 'threshold = 250' is the best case. Using bit level calculation on the best case (threshold = 250), 'output_change_L1.txt' is further reduced to 298.42 KB.

In level2, input file size = 80 KB
(output_seed_L1.txt)

Therefore, after completion of level2,
total compressed size = $\{18 + (18 + 14 + 36.84) + 298.42\}$ KB
= 385.26 KB



Figure 4. Picture considered for lossless compression

Ultimately, Huffman encoding scheme is used to compress the files further. In this case, a minimum of 6% compression has been achieved on the already compressed specific files as shown in Figure 3.

Therefore, after completion of Huffman encoding,
total compressed size = $\{(18*0.94) + ((18*0.94) + 14 + (36.84*0.94)) + (298.42*0.94)\}$ KB
= 362.97 KB

Thus, the final lossless compression achieved with respect to the original PNM file = 35.82%; and, the final lossless compression achieved with respect to the raw data file = 96.22%

6. Conclusion

In this paper, CA based lossless compression scheme on BMP image has been achieved successfully. A set of MACAs and/or SMACAs have been generated using the synthesis procedure. Since a single MACA / SMACA can inherit a lot of patterns as states, thus synthesizing a MACA / SMACA means actually the compression has been done. Further, Huffman encoding scheme is applied to achieve more compression. Firstly, the image is converted into data; and then, the data is compressed by formation of MACA and/or SMACA as required based on the consecutive patterns. This paper discussed the mechanism of replacing all the patterns (state values) with MACA and/or SMACA. This approach solves the problem of huge memory requirement for storing patterns. Since this is a replacement of original image file containing different patterns, here memory requirement is less with 100% prediction accuracy. As lossless compression is used in the paper, always 100% prediction accuracy is achieved using CA structure.

7. Future Work

In future, our target is to develop a compression scheme which will be comparable to JPEG lossless compression scheme using the Cellular Automata (CA) as a tool. Since, CA is hardware related tool; so, it can be used for faster processing with cost effectiveness. By survey, our research group has come to know that existing lossless compression on JPEG image is less than 25%. So, with our continuous research work on image compression using CA, we are trying to compress JPEG image at least 25% in a lossless manner.

References

- [1] S. Brin, L. Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine, Proceedings of the Seventh International World Wide Web Conference, Brisbane, Australia, April 1998.
- [2] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, S. Raghavan, Searching the Web, ACM Transactions on Internet Technology, vol.1, no.1, August 2001.
- [3] G. W. Flake, S. Lawrence, C. L. Giles, F. M. Coetzee, Self Organization and Identification of Web Communities, IEEE Computer, 35(3), 2000.
- [4] E. J. Glover, K. Tsioutsoulouklis, S. Lawrence, D. M. Pennock, G. W. Flake, Using Web Structure for Classifying and Describing Web Pages, WWW2002, Honolulu, Hawaii, USA, May 2002.

- [5] S. Chakrabarti, B. E. Dom, R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, J. Kleinberg, Mining the Web's Link Structure, *IEEE Computer*, (32)8: August 1999.
- [6] D. Mukhopadhyay, S. R. Singh, An Algorithm for Automatic Web-Page Clustering using Link Structures, *Proceedings of the IEEE INDICON 2004 Conference, India, December 2004*.
- [7] W. O. Cochran, J. C. Hart, P. J. Flynn, Fractal Volume Compression, *IEEE Transactions on Visualization and Computer Graphics*, 2(4), December 1996.
- [8] Z. Bar-Joseph, D. Cohen-Or, Hierarchical Context-based Pixel Ordering, *Computer Graphics Forum*, 22(3), September 2003.
- [9] E. Ageenko, P. Fränti, Lossless compression of large binary images in digital spatial libraries, *Computers & Graphics*, 24(1), February 2000.
- [10] J. V. Neumann, *The Theory of Self-Reproducing Automata*, A. W. Burks, Ed. University of Illinois Press, Urbana and London, 1966.
- [11] M. Sipper, Co-evolving Non-Uniform Cellular Automata to Perform Computations, *Physica D*, vol.92, 1996.
- [12] P. Maji, C. Shaw, N. Ganguly, B. K. Sikdar, P. P. Chaudhuri, Theory and Application of Cellular Automata For Pattern Classification, *Fundamenta Informaticae*, vol.58, December 2003.
- [13] S. Wolfram, *Theory and Application of Cellular Automata*, World Scientific, 1986.
- [14] S. Das, A. Kundu, B. K. Sikdar, Nonlinear CA Based Design of Test Set Generator Targeting Pseudo-Random Pattern Resistant Faults, *Asian Test Symposium*, 2004.
- [15] S. Das, A. Kundu, S. Sen, B. K. Sikdar, P. P. Chaudhuri, Non-Linear Cellular Automata Based PRPG Design (Without Prohibited Pattern Set) In Linear Time Complexity, *Asian Test Symposium*, 2003.
- [16] P. P. Chaudhuri, D. R. Chowdhury, S. Nandi, S. Chatterjee, *Additive Cellular Automata, Theory and Applications*, vol.1, IEEE Computer Society Press, Los Alamitos, California, 1997.
- [17] C. Yang, T. Mitra, T. Chiueh, On-the-fly rendering of losslessly compressed irregular volume data, *IEEE Visualization 2000*, October 2000.
- [18] S. Chattopadhyay, S. M. Bhandarkar, K. Li, Human Motion Capture Data Compression by Model-Based Indexing: A Power Aware Approach, *IEEE Transactions on Visualization and Computer Graphics*, vol.13 no.1, January 2007.
- [19] H. C. Yun, B. K. Guenter, R. M. Mersereau, Lossless Compression of Computer-Generated Animation Frames, *ACM Transactions on Graphics*, 16(4), October 1997.
- [20] T. Inada, M. D. McCool, Compressed Lossless Texture Representation and Caching, *Graphics Hardware 2006*, September 2006.
- [21] P. Diblík, P. Slavík, Control of Lossy Compression of Three-dimensional Data, 13th Spring Conference on Computer Graphics, June 1997.
- [22] C. Eisenacher, S. Lefebvre, M. Stamminger, Texture Synthesis From Photographs, *Computer Graphics Forum*, 27(2), April 2008.