THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

COMPRESSION UP TO SINGULARITY: A MODEL FOR LOSSLESS COMPRESSION AND DECOMPRESSION OF INFORMATION

Paul GAGNIUC[1] and Constantin IONESCU-TÎRGOVIȘTE[2]

[1] Faculty of Engineering in Foreign Languages, Politehnica University of Bucharest, Bucharest, Romania
[2] "N.C. Paulescu" Institute, Bucharest, Romania
Corresponding author: Paul GAGNIUC, E-mail: paul_gagniuc@acad.ro

Accepted August 24, 2016

Data compression is a niche area of great interest for the information technology field. Here we propose a novel theoretical concept for a total lossless compression and decompression of information regardless of size. Two fundamental methods have been considered in the same context, namely the use of hash functions and Markov chains. Our high-level strategy uses hashing to reduce message size, combined with an encoding of Markov chain probabilities used for generating candidate messages. The compression procedure is then applied recursively to the compressed representation to further reduce its size. To decompress a message, a message generator is used based on the Markov chain probabilities. For each candidate message, the hash function is applied and compared to the known hashed message.

Key words: information; singularity; lossless; compression; hash; markov.

INTRODUCTION

Lossless compression term summarizes those methods that allow the integral reconstruction of the original message from a compressed one[1]. Repeated patterns in the initial message form the basis of all lossless data compression algorithms used today. Most often complex implementations are used push against the redundancy limits of compression[2-4]. However, a redundancy reduction strategy imposes certain limitations related to the law of information entropy[5]. In this regard, high information content messages and randomly generated messages contain less redundant information and make further reductions unlikely[6-9]. In the past, by considering the transition probabilities between letters/words from a large number of natural English texts, Claude Shannon has shown that an n-state Markov Chain can be used for creating meaningful artificial messages that accurately resemble the human text construct[5,10,11]. Thus, the transition probabilities were derived from a large number of different messages. In a similar manner, we have wondered what would be the behavior of a Markov Chain if the transition probabilities between letters are extracted from a single message? Can the Markov Chain reach the initial message after a large number of runs? Moreover, if the initial message is unknown, then how can we detect if the Markov Chain has reached the initial message? If the transition probabilities between letters of the initial message are known, the rationale for the first two questions dictates that there would be no reason why it could not reach the initial message. In this case the only consideration of the Markov Chain would be the number of runs per unit of time. However, to know when the Markov Chain has reached the initial message a different reasoning has been used. We further wondered whether a small unique identifier may be generated from the initial message. Thus, we resorted to the use of hash functions. In theory, ideal hash functions are used to create unique identifiers (keys) for specific finite messages[12-14]. Thus, a hashing function receives a variable length message and generates a small key of a constant length. Therefore, these functions have a wide range of security applications from databases to communications. Nevertheless, a prominent property of the most advanced hash functions is the power of reduction regardless of the redundancies inside the initial message. This property has not yet been explored in regard to data compression since one way hash functions are not reversible. This implies that the initial message can not be directly recreated relying only on the hash key. One of the many reasons is that for a large number of messages, a constant length of the hash keys limits the number of possible keys that can be assigned by the function. Thus, for a very large set of messages, any hashing function will inevitably associate a key to a subset of messages (an event called collision)[15-18]. In a hypothetical case one can not know which of those messages is the initial one. However, in practice other verification mechanisms related to message integrity can be implemented to avoid collisions between messages. In this thought experiment, one last question would be whether a method can be found by which the original message can be obtained using the hash key. As a result, here we propose a model for a total compression and decompression of information in which three main components are used: 1) a one way hash function (h), 2) a transition probability function (Tr), and 3) an n-state Markov Chain Generator (MCG).

COMPRESSION TO SEED

In the case of compression the transition probabilities of the initial message and a hashing function are used for a gradual reduction of information (Figure 1). Instead of text, we consider that the initial message (M_i) consists of a string of 1s and 0s. The initial message (M_i) is divided into smaller segments ($X_1 \dots X_n$) of constant length (L_x). For each segment ($X_1 \dots X_n$) a hash value is generated $h(X_1) \dots h(X_n)$).

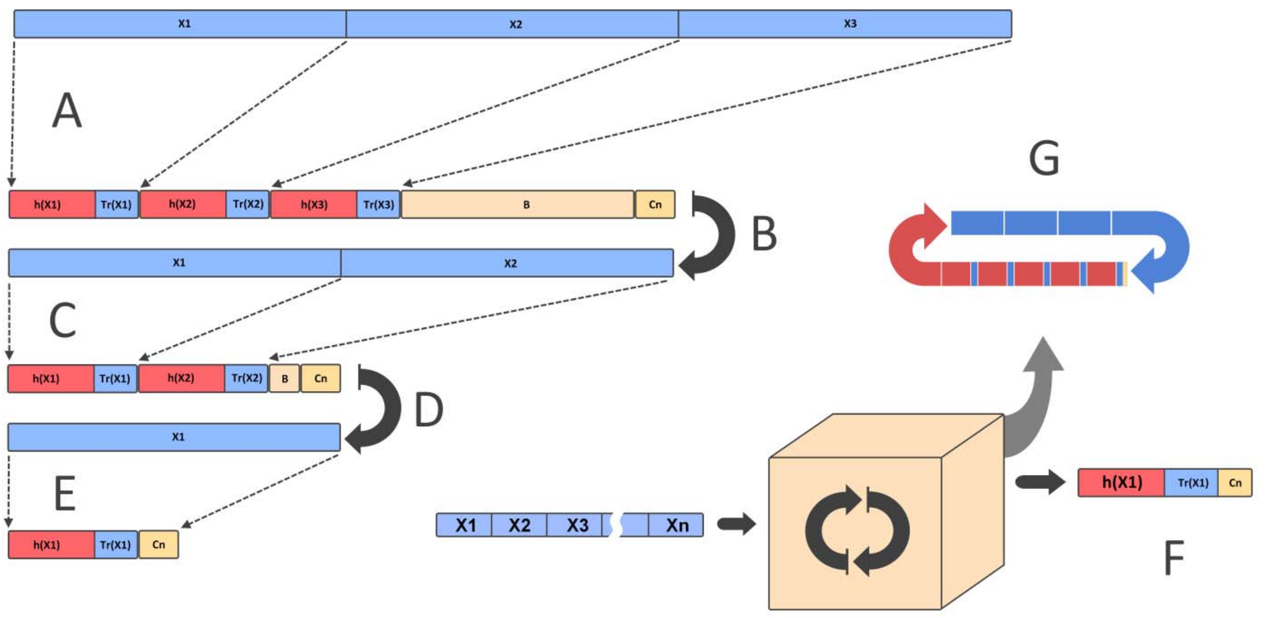

Fig. 1. Compression. (A) The initial message (M_i) is divided into smaller segments $X_1 \dots X_n$ (in blue) from which a chain of seeds is generated ($h(X_1),T_r(X_1) \dots h(X_n),T_r(X_n)$), (B) the chain of seeds becomes the new message (M_{i+1}), (C) the new message (M_{i+1}) is divided into smaller segments ($X_1 \dots X_n$) from which a chain of seeds is again generated, (D) the new chain of seeds becomes the new message once again (M_{i+2}), (E) the M_{i+2} message generates the last seed, (F) an rationalization of the model in which any finite message is reduced to one seed, (G) a graphical representation of the reduction process in which the new chain of seeds becomes the new message until singularity is achieved.

Next, the transition probabilities between 1s and 0s can be calculated using a transition probability function ($T_r(X_1) \dots T_r(X_n)$). The transition probability function (Tr) should provide at least two values, namely P_0 and P_1 (see decompression). Thus, the $X_1 \dots X_n$ segments are reduced to "seeds" of constant length (L_s), that consist of a series of hash values each followed by transition probability values derived from the corresponding segment ($S_{(1\dots m)} = h(X_1),T_r(X_1) \dots h(X_n),T_r(X_n)$). In the next step the chain of seeds ($S_{(1\ldots m)}$) is considered, the new message (M_{i+1}) and the process of compression is repeated (Figure 1B). At each repetition the algorithm should add a counter variable (C_n) to the seed chain, which is intended to show the compression depth. Thus, this process is repeated until all that remains from the initial message is one seed S_1= ($h(X_1),T_r(X_1),C_n$) consisting of one hash value, transition probabilities (P_0,P_1), and the counter variable (Figure 1E,G). Nevertheless, before these seeds ($S_{(1\ldots m)}$) can be considered as a new message (M_{i+1}), the algorithm must ensure that the seeds will occupy the entire space allocated for the next $X_1 \ldots X_n$, namely a multiple of L_x. At each repetition the length of the resulting chain of seeds ($h(X_1),T_r(X_1) \ldots h(X_n),T_r(X_n)$) will not always match the entire length of the $X_1 \ldots X_n$ segments from the next step in the compression process (Figure 1A). Therefore, a "ballast" function is introduced, namely $B(L_x \times n - L_s \times m)$. The B function aims to insert random information in the remaining space ($L_x \times n - L_s \times m$). The addition of random information is not relevant (since the algorithm is time-dependent) and is removed at decompression time (see decompression). Of course, the ballast function is optional since the length of a seed (L_s) is constant and L_x can be calculated prior to compression.

DECOMPRESSION OF SEED

In the case of decompression the hash value ($h(X_1)$) and the transition probabilities ($Tr(X_1)$) stored in the initial seed ($S_1 = (h(X_1),T_r(X_1),C_n)$) are used for a gradual expansion of information (Figure 2). As an example, a two-state Markov Chain Generator (MCG) has been considered (Figure 2C).

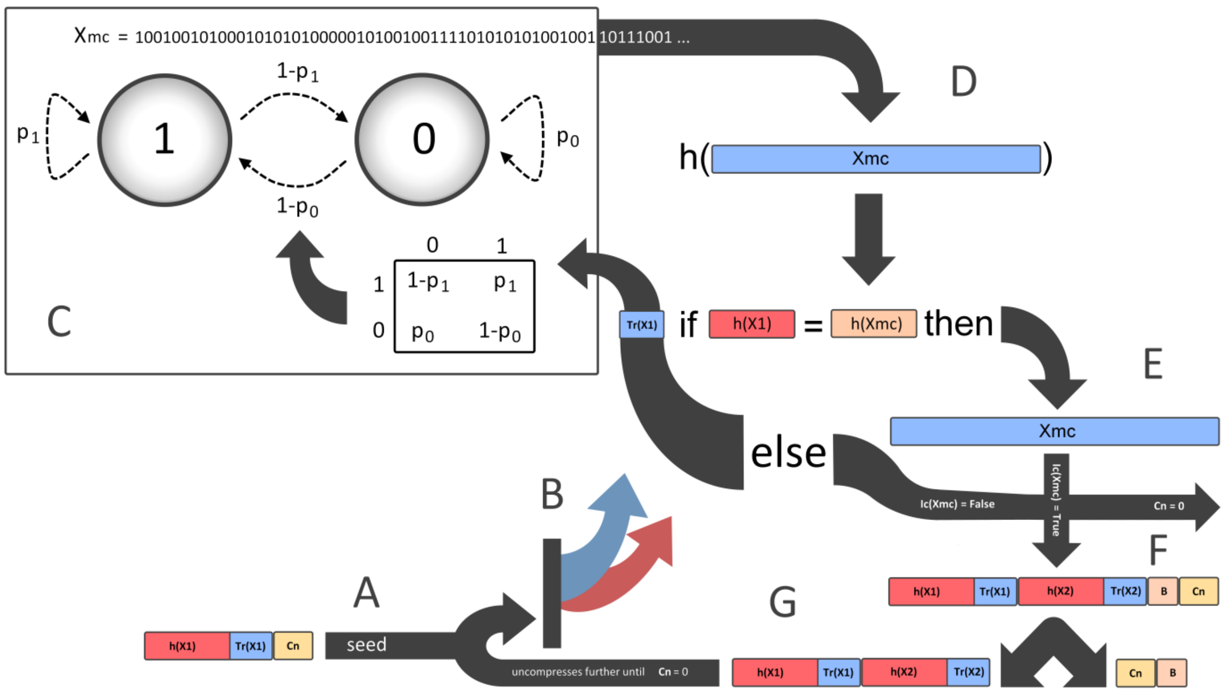
Xmc = 10010010100010101010000010100100111101010101001001 10111001 ...
1-p1
p1
1
0
p0
1-p0
0 1
1 1-p1 p1
0 p0 1-p0
C
D
h(Xmc)
Tr(X1)
if h(X1) = h(Xmc) then
E
Xmc
else
Ic(Xmc) = False
Ic(Xmc) = True
Cn = 0
F
h(X1) Tr(X1) h(X2) Tr(X2) B Cn
B
A
G
h(X1) Tr(X1) Cn
seed
uncompresses further until Cn = 0
h(X1) Tr(X1) h(X2) Tr(X2)
Cn B

Fig. 2. Decompression. (A) initial seed, (B) decomposition of the seed in the two main components, namely the hash value (red) and the transition probability values (blue), (C) the two-state Markov Chain Generator (MCG). The MCG message (X_{MC}) is generated within the limits imposed by the two transition probability values ($Tr(X_1)$) of the seed. (D) A hash value is made from the MCG message ($h(X_{MC})$), (E) If the new hash value ($h(X_{MC})$) is identical to the hash value ($h(X_1)$) of the seed then the message under $h(X_1)$ is temporarily found, (F) the integrity check function ($I_c(X_{MC})$) examines the presence of delimiters in order to avoid a random message that may generate the same hash value. If delimiters are not found then a new message (X_{MC}) is generated by the MCG within the limits imposed by the two transition probability values ($Tr(X_1)$) of the seed. If $C_n = 0$ then the initial message (M_i) has been entirely reconstructed, (G) If $C_n > 0$ the balast and the C_n variable are removed and the new seeds are independently decompressed in a new cycle.

The MCG receives the task of generating a message (X_{MC}) of length L_x within the limits imposed by the two transition probability values (P_0 and P_1) of the seed (Figure 2A,B,C). A new hash value ($h(X_{MC})$) is calculated for each segment generated by MCG (Figure 2D). The new hash value ($h(X_{MC})$) is then compared with the hash value ($h(X_1)$) of the original segment (Figure 2E). If the two hash values do not coincide then the MCG will proceed forward and generate a new segment (Figure 2E,C,D). However, if the hash value of the new segment ($h(X_{MC})$) and the hash value of the original segment ($h(X_1)$) have the same value then it is considered that the original segment has been temporarily found ($X_{MC} = X_1$). If X_{MC} is identical to the original (X_1) then it must contain seeds and a counter variable (Figure 2E,F). Because hash functions could sometimes associate a hash value to multiple messages, an integrity check ($I_c(X_{MC})$) of the X_{MC} is made in order to avoid collisions. Such a verification is based on the presence of delimiters at equal length intervals between seeds in the X_{MC} sequence. If the integrity check function ($I_c(X_{MC})$) does not detect the presence of delimiters, the MCG will proceed forward to a new X_{MC} segment. Otherwise, if the integrity check function ($I_c(X_{MC})$) detects the presence of delimiters, then it may be considered that the original segment (X_1) has been found and no collisions have occurred (Figure 2F). Therefore, the first step in the decompression process can be completed and the C_n variable verified. Since X_{MC} contains a series of new seeds the next phase begins. The seeds are extracted from the X_{MC} whereas the balast and the C_n variable are removed (Figure 2F,G). Next, each seed from the X_{MC} segment will go through the same process of decompression described above until $C_n = 0$ (Figure 2G). When C_n equals zero the initial message (M_i) has been found (Figure 2F).

DISCUSSIONS

Here we propose a theoretical model for data compression. In the case of compression the transition probabilities of the initial message and a hashing function are used for a gradual reduction of information, whereas the gradual expansion of information begins through the use of the hash value and the transition probabilities stored in the initial seed (Figure 1 and Figure 2).

This paper proposes a data compression scheme that uses one-way hash functions for compression. In the proposed scheme, the data source is segmented and a hash value is calculated for each segment and stored together with transition probabilities in order to form the seed (*i.e.*, compressed representation of the source). The method is applied recursively until only a single hash value and a set of transition probabilities remain. In order to reconstruct the source, a Markov chain generator is used for the synthesis of the source segments that lead to similar hash values as the ones obtained in the compression phase. Additional overhead is required to avoid hash collisions in the reconstruction.

Variations of the method may also be used. For instance, for refining the decompression process the two states of the MCG can be increased. Although an implementation of this method may reache full compression as fast as the algorithms used today, the situation would be different in the case of decompression. Since the MCG acts as an uncertainty reducer the decompression time would be difficult to accurately assess. Overall the method can be applicable using the computing power of today, however, the decompression time would perhaps be too high for immediate applications. With the evolution of quantum computers, this method may have practical applications in the future. The exponential growth of the seeds makes it an ideal candidate for quantum processing[19].

CONCLUSION

A unique identifier (a hash key) followed by two transition probability values is defined as a seed. Here we show a method by which large sized messages are reduced to a seed and vice versa. Initially, large sized messages are reduced to a chain of seeds. Each chain of seeds is then gradually reduced in the same manner until only one seed remains. In contrast, for expansion of information a Markov Chain Generator (MCG) uses the two transition probability values of a seed to recreate the message behind the hash value of the same seed. If found, underlying the message are other seeds that follow the same process until the original message is recreated without loss.

REFERENCES

1. Smith S.W., "The Scientist and Engineer's Guide to Digital Signal Processing", California Technical Publishing, 1997.
2. Sayood K., "Introduction to Data Compression", Morgan Kaufmann Publishers, 2000.
3. Held G., Marshell T.R., "Data and Image Compression", John Wiley and Sons, 1996.

4. Gryder R., Hake K., "Survey of Data Compression Techniques", ORNL/TM-11797, 1991.
5. Shannon, C.E., *A Mathematical Theory of Communication. Bell System Technical Journal*, **1948**, 27: 379–423.
6. Drost G.W., Bourbakis N.G., *A hybrid system for real-time lossless image compression, Microprocessors and Microsystems*, **2001**, 25 (1): 19–31.
7. Yu-Chen H., Chang C.C., *A new lossless compression scheme based on Huffman coding scheme for image compression, Signal Processing: Image Communication*, **2000**, 16(4): 367–372.
8. Asli A.Z., Alipour S., *An effective method for still image compression–decompression for transmission on PSTN lines based on modification of Huffman coding, Computer and Electrical Engineering*, **2004**, 30 (2): 129–145.
9. Lelewer D.A., Hirschberg D.S., *Data compression, ACM Computing Surveys*, **1987**, 19(3): 261–296.
10. Марков А.А., Распространение закона больших чисел на величины, зависящие друг от друга. Известия Физико-математического общества при Казанском университете, 2-я серия, том 15, ст. 135–156, 1906.
11. Markov A.A., "Extension of the limit theorems of probability theory to a sum of variables connected in a chain". Reprinted in Appendix B of: R. Howard. Dynamic Probabilistic Systems, volume 1: Markov Chains. John Wiley and Sons, 1971.
12. Preneel B., "Analysis and design of cryptographic hash functions", PhD thesis, Katholieke Universiteit Leuven, 1993.
13. Preneel B., Govaerts R., and Vandewalle J., *Hash Functions Based on Block Ciphers: A Synthetic Approach. Lecture Notes in Computer Science*, **1994**, 773:368–378.
14. Preneel B., *Cryptographic primitives for information authentication – state of the art. Lecture Notes in Computer Science*, **1998**, 1528:49–104.
15. den Boer B. and Bosselaers A., *Collisions for the compression function of MD-5. Lecture Notes in Computer Science*, **1994**, 765:293–304.
16. Hohl W., Lai X., Meier T., Waldvogel C., *Security of iterated hash functions based on block ciphers. Lecture Notes in Computer Science*, **1994**, 773:379–390.
17. van Oorschot P.C. and Wiener M.J., *Parallel collision search with cryptanalytic applications. Journal of Cryptology*, **1999**, 12(1):1–28.
18. van Rompay B., "Analysis and design of cryptographic hash functions, MAC Algorithms and Block Ciphers", PhD thesis, Katholieke Universiteit Leuven, 2004.
19. DiVincenzo, David P., *Quantum Computation. Science*, **1995**, 270 (5234): 255–261.