A method for Information hiding using Huffman coding

Divya P, Dr. Arun Kumar Marandi,
ARKA JAIN University, Jamshedpur-831014, India
Email- divya.p@arkajainuniversity.ac.in, dr.arun@arkajainuniversity.ac.in,

Abstract - *In current era, there are many techniques to embed data to increase efficiency. One of them is information hiding model using Microsoft world documents using track-changing feature. Track changing is a feature or way to keep track of the changes made in a document. Then it can be chosen to accept or reject those changes and MS Word store information about the changes that is made in a document .This process is done using Huffman code algorithm which is a lossless data compression algorithm based on the frequency of occurrence of a data item. So it helps in increasing the embedding capacity of change tracking data hiding technique and it is done by hiding the redundant synonyms bits compress it so that the word document becomes less in size using Huffman code process, in this methodology the embedding is efficient effectively.*

Keyword: - Watermarking, Information hiding, Huffman coding, Data compression, Track changing

I. INTRODUCTION

A stenographic technique was proposed for data hiding for the security purpose. The embedding process is done in the document so that the message sent to the recipient is secure. The embedding can be done on images as well as on the text. Huffman code is used for the process which is a lossless data compression algorithm so that the important data cannot be destroyed or removed from the document. Track changing keeps the record of data which are changed using the algorithm. Huffman code

algorithm is based on the frequency of occurrence of the word in the text document. It helps the embedding process as the redundant words can be embedded and also increases the efficiency of the embedding process. In this process the message is encrypted in binary form that is 0 and 1.All the information is encrypted in binary form and Huffman code is coded into the combination of 0 and 1.

II. METHODOLOGY

The Huffman coding algorithm finds the least uniquely variable length which is decodable, entropy code associated with a set of events given their probabilities of occurrence. The Huffman coding method is used for the watermarking process which increases the probability of security. The process is based on binary tree method. In this method the symbol which occurs most frequently will have shorter codeword than the less occurred symbol. Basic algorithm followed is

A. Create a leaf node for each unique character and build a min heap of all leaf nodes.
B. Extract two nodes with the minimum frequency from the min heap.
C. Create a new internal node with frequency equal to the sum of the two nodes frequencies. Make the first extracted node as its left child and the other extracted node as its right child. Add this node to the min heap.
D. Repeat steps 'B' and 'C' until the heap contains only one node. The remaining node is the root node and the tree is complete.

III. OCCURRENCE AND NON-OCCURRENCE PROBABILITY

Suppose an event 'A' can happen in 'r' way out of a total of n possible ways. Then the probability of occurrence of event is denoted by

$$P(A) = \frac{number\ of\ favourable\ choices\ for\ the\ event}{total\ number\ of\ possible\ choices\ in\ the\ experiment} \quad 1$$

$$P(A) = \frac{m_A}{n} \quad 2$$

Non- occurrence probability gives the opposites of occurrence probability.

Number of unfavorable choices for the event = Total number of possible choices in the experiment - Number of favourable choices for the event

$mA^c = n\text{-}mA$ 3

Probability of Non-Occurrence of event “A”

$$P(A^C) = \frac{number\ of\ unfavourable\ choices\ for\ the\ event}{total\ number\ of\ possible\ choices\ in\ the\ experiment} \quad 4$$

$$P(A) = \frac{m_{A^C}}{n} \quad 5$$

In the above equations have modifies the earlier change-tracking the technique to perform steganography to gain more embedding capacity and reduce suspicion is designed.

the proposed technique to embed a secret message into a Microsoft Word document by using change-tracking information with synonym substitution based on Huffman code generated by non-occurrence probability of words synonyms. The main idea of the proposed method is to redesign the Huffman code
generating method instead of using the original Huffman code generating method in order to gain more embedding capacity. It is found that if shorter Huffman codes are used then for large messages large number of degenerations will be required; which may raise suspicion. Therefore, non-occurrence probability can be used to increase the embedding capacity with common substitution. By generating Huffman code using non-occurrence probability instead of occurrence probability would help in embedding large message with less number of
degenerations. Huffman code is commonly used in data compression. Using proposed method Huffman code for common synonyms will have more number of bits than uncommon synonyms, which will decrease the number of degenerations in the cover
document file, which in turn will make the document less suspicious to attackers. Table - 1 depicts the Huffman code generation example with occurrence probabilities and non-occurrence probabilities.

The Huffman coding procedure finds the optimum (least rate) uniquely decodable, variable length entropy code associated with a set of events given their probabilities of occurrence. The procedure is simple enough that we can present it here.

The Huffman coding method is based on the construction of what is known as a binary tree. The path from the top or *root* of this tree to a particular event will determine the code group we associate with that event.

Suppose, for example, that we have six events with names and probabilities given in the table 1 below.

Table -1

Event Name	Probability
A	0.30
B	0.30
C	0.13
D	0.12
E	0.10
F	0.05

Our first step is to order these from highest (on the left) to lowest (on the right) probability as shown in the following figure, writing out next to each event its probability for since this value will drive the process of constructing the code.

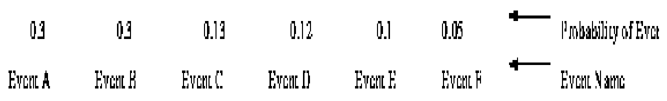


Figure 1: Preparing for Huffman code construction. List all events in descending order of probability.

Now we perform a construction process in which we will pair events to form a new single combined event which will replace the pair members. This step will be repeated many times, until there are no more pairs left.

First we find the two events with least combined probability. The first time we do this, the answer will always be the two right hand events. We connect them together, as shown in Figure -1 calling this a combined event (EF in this case) and noting its probability (which is the sum of those of E and F in this case.) We also place a 0 next to the left hand branch and a 1 next to the right hand branch of the connection of the pair as shown in the figure. The 0 and 1 make up part of the code we are constructing for these elements.

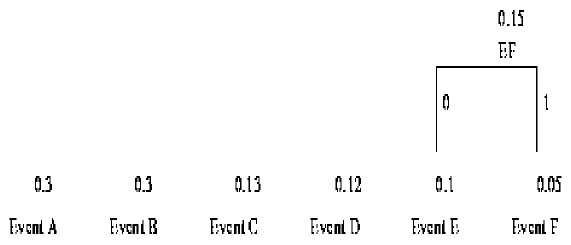
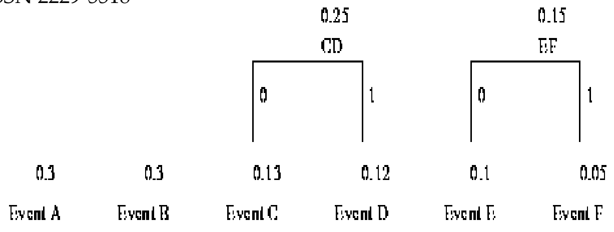
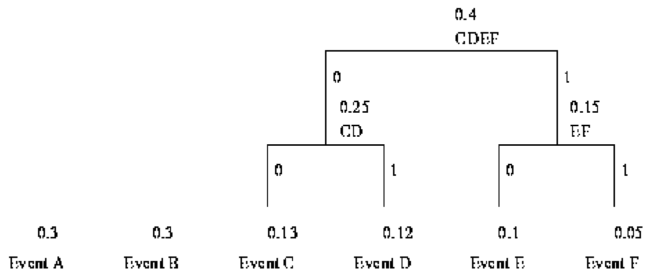


Figure 2: Preparing for Huffman code construction. List all events in descending order of probability.

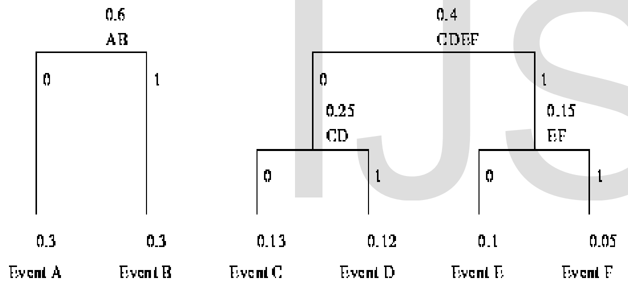
Now we repeat the last step, dealing only with the remaining events and the combined event. This time combining C and D creates a combined event with less probability than combining any others.



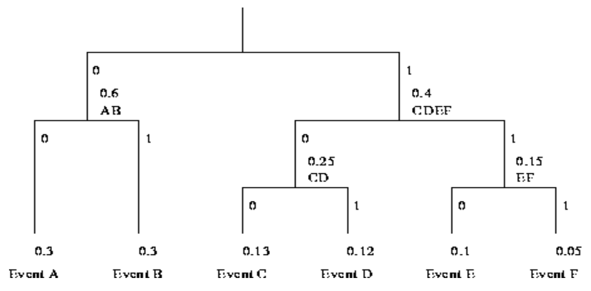
Again we repeat the process. This time, combining the combined events CD and EF create the new combined event with least probability.



The next time around the best combination is of A and B.



Finally there is only one pair left, which we simply combine.



Having finished our connection tree, we are ready to read off the diagram the codes that we will associated with each of the original events. To obtain the code, we start at the top level of the tree and make our way to the event we wish to code. The series of 0's and 1's we encounter along the way on the branches of the tree comprise our code. Doing so for each event in this case yields the following result.

Table - 2

Event Name	Probability	Code	Length
A	0.3	00	2
B	0.3	01	2
C	0.13	100	3
D	0.12	101	3
E	0.1	110	3
F	0.05	111	3

If we sum the products of the event probabilities and the code lengths for this case we obtain an average bit rate of 2.4 bits per event. If we compute the true minimum bit rate, that is the information rate, of these events as we did with the previous example, we obtain 2.34 bits.

Suppose that we had been originally planning to code our events originally as all 3 bit codes in a fixed length code scheme. Then, if we code a document which is long enough so that we obtain the average promised by this new scheme instead, we will find that we will obtain a compression ratio over the original scheme of 2.4/3 = 80% whereas the ultimate possible compression ratio is 2.34/3 = 78%.

It can be shown that the Huffman code provides the best compression for any communication problem for a given grouping of the events. In this problem we chose not to group events, but to code them individually. If we were to create the 36 events we would get by forming pairs of the above events, we would get substantially closer to the optimum rate suggested by the information rate calculation.

IV. PROPOSED MESSAGE EMBEDDING & EXTRACTION ALGORITHM

The message embedding process the words with its synonyms. The use of the synonym does not cange the meaning of the sentences. Synonym set ‘S_w’ is created to for word ‘d’. the embedding process consists of following steps:-

Algorithm :- Embedding a message by word’s synonym substitution and revision.

Input: A cover MSWord document, a message to be embedded
Output: A step go-document

Steps:

1. Select a message. Convert it into bit form using standard encoding scheme (ASCII).
2. Select a MS-Word document to be used as a cover. Select the available words in the cover document having synonyms.
3. Make synonym set table with respective occurrence probabilities.

4. Calculate the non-occurrence probability based on occurrence probability for all the synonym sets.
5. Generate Huffman code based on non-occurrence probability of the synonym sets.
6. Select words, which are synonymous according to the required bits, need to be embedded. From the longest bits string match is searched with the secret message, if match is found then substitution can be made else the next match with the 2nd longest bits string should be searched for and the process goes on.
7. Set the synonyms selected to the document using change-tracking technique and if the same synonym is selected then some spelling or grammatical mistake can be made in it to serve as a method of embedded.
8. If last left bits do not match any bit strings of the desired words synonym, then its next shortest bit strings synonym is considered.

After the completion of secret bits, embedding stops and the step go document is sent to the receiver.

The extraction process consists of following steps: -

Algorithm: Extracting a message by synonym tracing and revision.

Input: A step go-document
Output: The extracted message
Steps:

1. Select the step go document.
2. Trace the synonyms and actual words.
3. Based on the same database having non-occurrence probability, find the bits embedded in synonyms.
4. Merge the bits & divide the length of the bit string by 8, if any remainder is present discard that no. of bits from the end.
5. Map each 8 bits to respective characters to get the secret message.

V. Conclusion

The process we have applied has increased the embedding capacity of change tracking data technique. The technique which been proposed will use the most common data or synonyms to hide no of bits and thereby making the document original which cannot be easily detected or destroyed by any unknown person. A Huffman tree is generated from the synonyms of non occurrence probability. The result shows the increased efficiency of the technique.

VI. References

[1] S.Shanmugasundaram and R. Lourdusamy, " IIDBE: A Lossless Text Transform for Better Compression" International Journal of Wisdom Based Computing, Vol. 1 (2), August 2011

[2] A. B. sorting Lossless, M. Burrows, M. Burrows, D. Wheeler, and D.J. Wheeler, “A block-sorting lossless data compression algorithm,” Digital SRC Research Report, Tech.Rep., 1994

[3] P. Kumar and A.K Varshney, “ Double Huffman Coding ” International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) Volume 2, Issue 8, August 2012

[4] A. Singh and Y. Bhatnagar, “ Enhancement of data compression using Incremental Encoding” International Journal of Scientific & Engineering Research, Volume 3, Issue 5, May-2012

[5] Introduction to Data Compression, Khalid Sayood, Ed Fox (Editor), March 2000.

[6] Ziv and A. Lempel, “Compression of individual sequences via variable-rate coding,”Information Theory, IEEE Transactions on, vol.24, no. 5, pp. 530–536, sep. 1978

[7] M. Sharma, “Compression using Huffman Coding ” IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.5, May 2010

[8] Arya, I. G., & Dewangga, P. (2017). A New Approach of Data Hiding in BMP Image Using LSB Steganography and Caesar Vigenere Cipher Cryptography, 12(21), 10626–10636.

[9] Mstafa, R. J., Elleithy, K. M., & Abdelfattah, E. (2017). A Robust and Secure Video Steganography Method in DWT-DCT Domains Based on Multiple Object Tracking and ECC. IEEE Access, 5(c), 5354–5365. https://doi.org/10.1109/ACCESS.2017.2691581.

[10] Arya, I. G., & Dewangga, P. (2017). A New Approach of Data Hiding in BMP Image Using LSB Steganography and Caesar Vigenere Cipher Cryptography, 12(21), 10626–10636.