Implications of Huffman Algorithm for Alphabet Partitioning Entropy Coding

***Abstract*—Alphabet partitioning group entropy coding techniques i.e., modified Huffman codes, are being used effectively in practice and also play an essential role in the standardization of various compression techniques like JPEG and MPEG. However, as compared to the original Huffman entropy coding, implementation of this technique observes additional grouping redundancy i.e., increment of average codeword length under the assumption of fixed length coding, that can be minimized or resolved using different adaptive algorithms or uniform distribution approximations. This correspondence revisits the alphabet partitioning technique and redesign Huffman algorithm in such a way that the average codeword length of symbols can be achieved same as conventional Huffman algorithm. Accordingly, with the help of examples, it is revealed that the overall compression ratio of the proposed method is the same as the original Huffman source coding algorithm.**

***Index Terms*—Entropy coding, prefix code, alphabet partitioning, modified Huffman codes.**

I. Introduction

II. Proposed Algorithm

As discussed in [1], the original source entropy i.e., $H(p)$, can be written as the addition of coding rate of group selection (i.e., $g(s)$)) and the symbol index (i.e., $x(s)$). In similar fashion, we can also write the expression of code word length (i.e.,$L(p)$) for conventional Huffman algorithm as given below

$$L(p) = \sum_{n=1}^{N_g} \Big[\rho_n l_g + \sum_{s \in G_n}^{N_s} p_s l_{sn} \Big], \tag{1}$$

where, ρ_n signifies the total probability ($\sum_{s=1}^{N_s} p_s$) of particular group and p_s is probability of symbols. Note that, as per the literature [1], (1) will be satisfied only if probability distribution (i.e.,p_s) is uniform and all the groups N_g should have number of symbols that is dyadic. Now, our objective is to choose the symbols in groups (i.e., N_s) and number of groups (i.e., N_g) such that the code word length of partitioned alphabet i.e., alignment of bits required in symbol l_{sn} and group index l_g, must be same as $L(p)$. For example, a i.i.d. message source has seven symbols a, b, c, d, e, f, and g with probability 0.3, 0.2, 0.15, 0.15, 0.10, 0.05, and 0.05, respectively. As per the conventional Huffman algorithm, these symbols can be coded in binary as given in Table I. Thereafter, the average codeword length of the given symbols would be calculated as $L(p)$= 2.6. Herewith, we are applying the grouping strategy and, furthermore, use of conventional Huffman algorithm for coding the symbol index. This observation is shown in Table II where $N_g = 2$ with symbols in first group i.e., G1, are 4 and 3 in second group i.e., G2. Here, Huffman algorithm is applied in two parts like group and symbol index coding. First, we calculate the group symbol probability and then apply Huffman with the corresponding symbols lie in same group. Moreover, we can apply Huffman with group probability (i.e., 0.8 and 0.2) considering as symbols. This process is illustrated in Table II whereas, for the clarity, binary coded symbols that belongs to group index (ρ_n=0.8 and 0.2) are being red colored. Similar to the conventional approach, with the help of (1), the average codeword length of this group arrangement can be evaluated as $L'(p)$= 2.9. Certainly, the average codeword length has been increased to 2.9 which makes this approach inefficient.

To avoid this redundancy and capture the same efficiency as the conventional algorithm, we redefine the grouping of symbols i.e., proposed algorithm, as follows:

1) Generate a list for the given set of symbols as per the known probabilities or frequency counts.
2) Arrange these symbols in decreasing order of probability i.e., the most probable ones to the top and least probable to the bottom of the list.
3) Split this list into two groups, with the total probability

TABLE I
Conventional Huffman Algorithm.

Parameter	Huffman Code (HC)						
Symbols	a	b	c	d	e	f	g
Probability	0.3	0.2	0.15	0.15	0.10	0.05	0.05
Codeword	00	10	010	011	110	1110	1111

TABLE II
Conventional Huffman Algorithm with Symbol Grouping.

Parameter	HC(G1) (ρ_n=0.8)				HC(G1) (ρ_n=0.2)		
Symbols	a	b	c	d	e	f	g
Probability (p_s)	0.3	0.2	0.15	0.15	0.10	0.05	0.05
Group Symbol Prob.(p_s/ρ_n)	0.375	0.25	0.1875	0.1875	0.5	0.25	0.25
Codeword	01	001	0000	0001	10	110	111

TABLE III
CONVENTIONAL HUFFMAN ALGORITHM WITH OPTIMIZED SYMBOL GROUPING.

Parameter	HC(G1) (ρ_n=0.5)		HC(G2) (ρ_n=0.5)				
Symbols	a	b	c	d	e	f	g
Probability	0.3	0.2	0.15	0.15	0.10	0.05	0.05
Group Symbol Prob.(p_s/ρ_n)	0.6	0.4	0.3	0.3	0.2	0.1	0.1
Codeword	00	01	100	101	110	1110	1111

TABLE IV
CONVENTIONAL HUFFMAN ALGORITHM WITH OPTIMIZED MORE SYMBOL GROUPING.

Parameter	HC(G1) ρ_n=0.5)		HC(G2) ρ_n=0.3)		HC(G3) ρ_n=0.2)		
Symbols	a	b	c	d	e	f	g
Probability	0.3	0.2	0.15	0.15	0.10	0.05	0.05
Group Symbol Prob.(p_s/ρ_n)	0.6	0.4	0.5	0.5	0.5	0.25	0.25
Codeword	00	01	100	101	110	1110	1111

of both the groups being as close to each other as possible.

4) Recursively, we can create more possible number groups within the existed groups as defined above.
5) Calculate group symbol probability i.e., p_s/ρ_n, to make the sum of symbol probabilities one.
6) Assign binary code to each group index and related symbol index using Huffman algorithm until all the symbols and groups index get their binary code.

After using proposed algorithm, we can form the groups of symbols as shown in Table III.

Similarly, we can form more groups recursively as shown in Table IV. Here, G2 is further divided in G2 and G3. Furthermore, we can calculate the average codeword length of symbols as arranged in Table III and IV. By using the (1), $L(p)$ will be 2.6 for both Table III and IV which is same as conventional Huffman coding. Therefore, we can create the symbol grouping of the message source without increment in average codeword length or grouping redundancy.

A. *Result and discussion*

III. CONCLUSION

ACKNOWLEDGMENT

REFERENCES

[1] A. Said, "Efficient alphabet partitioning algorithms for low-complexity entropy coding," in *Data Compression Conference.,* pp. 183-192, 2005.
[2] A. Painsky, S. Rosset and M. Feder, "Large Alphabet Source Coding Using Independent Component Analysis,"*IEEE Transactions on Information Theory.,* vol. 63, pp. 6514-6529, Oct. 2017.
[3] W. A. Pearlman and Amir Said, *Digital Signal Compression: Principles and Practice.* Cambridge University Press, Cambridge, UK, 2011.