Develop and Design Hybrid Genetic Algorithms with Multiple Objectives in Data Compression

Khalil Ibrahim Mohammad Abuzanouneh

Qassim University, College of Computer, IT Department, Saudi Arabia

Summary
In this paper, Data compression plays a significant role and is necessary to minimize the storage size and accelerate the data transmission by the communication channel object, The quality of dictionary-based text compression is a new approach, in this concept, the hybrid dictionary compression (HDC) is used in one compression system, instead of several compression systems of the characters, syllables or words, HDC is a new technique and has proven itself especially on the text files. The compression effectiveness is affected by the quality of hybrid dictionary of characters, syllables, and words. The dictionary is created with a forward analysis of text file a. In this research, the genetic algorithm (GAs) will be used as the search engine for obtaining this dictionary and data compression, therefore, GA is stochastic search engine algorithms related to the natural selection and mechanisms of genetics, the s research aims to combine Huffman's multiple trees and GAs in one compression system. The Huffman algorithm coding is a tactical compression method of creating the code lengths of variable-length prefix code, and the genetic algorithm is algorithms are used for searching and finding the best Huffman tree [1], that gives the best compression ratio. GAs can be used as search algorithms for both Huffman method of different codes in the text characters leads to a difference in the trees, The GAs work in the same way to create the population, and can be used to guide the research to better solutions, after applying the test survive and their genetic information is used.
The proposal is a new multi-trees approach to compute and present the best Huffman code trees of characters, syllables, and words to get the best possible of minimization compression [2]. The frequency increases to the text have an impact on compression rate. This approach generates more distinctive code words and efficient of space consumption and computation for compression rate without losing any data on the original files, the experiments and results show that a significant improvement and accuracy can be achieved by using the code words derived [3].

Key words:
Huffman algorithm, Data compression, Genetic information, Huffman code trees.

1. Introduction

Data compression is one of the main important topics in the last years. Big data must be stored in data warehouses or archives, and a big data must be transmitted through communication channels. There are several of Data compression algorithms were designed for data processing byte by byte [4]. For example, input any symbols are taken from ASCII code table, therefore, the size of all symbols are is 256 represented by 8-bits. and it's easy to store all symbols into compressed data, because it's available on every system. Data compression is the creating binary process of representations of the file which requires minimum storage space than the original file. The main data compression techniques are a lossy and lossless compression. The lossless data compression is used when the data has to be uncompressed exactly as it was before compression. The files is stored by using lossless technique, losing any single character can in the worst case make the data misleading. There are limits to the amount of space saving that can be gotten with lossless compression. Lossless compression ratios are generally in the range of 0.5 to 0.2. The compression of lossy works on the assumption that the file doesn't have to be stored perfectly. Much information can be thrown away from some images, data of video and audio, and when uncompressed some of the data will still be acceptable quality. The popular method for data compression is Huffman Code runs under various platforms. It is used as a multistep compression process to produce better codes; the best code produces if the symbol probabilities are negative and power of 2.
The compression methods are developed and have the best research for the data compression By D. Huffman in 1952, the algorithm was started to build a list of all the symbol probabilities in descending order. It can construct a Huffman trees at every leaf, with a symbol node, from the down up, it's done in several steps, and for each step, the two symbols are selected with minimum probabilities, where it will be added to the partial tree top, and from the list is deleted, in the next step, the two original symbols can be replaced with an auxiliary symbol, where the list is reduced to one auxiliary symbol representing all alphabet, the characters after completing , the tree will be traversed to determine all codes .

2. Huffman algorithm

All Huffman code defines a frequency based on codes of scheme that can generate a set of variables with size code words of the minimum average length as follows:

Manuscript received October 5, 2017
Manuscript revised October 20, 2017

1- The frequency table will be constructed and sorted in descending order from text file to Build binary trees, derive Human tree and generate Huffman code to be compressed and find the best Huffman trees, as in the figure 1 below.

The data will be readied from file text of the proposed system and to create the initial population, and from the initial population, the chromosomes will be produced.

Huffman encoding example: When a message has n symbols, block encoding requires log n bits to encode each symbol. Message Size = frequency character * fixed length 43*8= 44 bits.

Huffman encode massage:

10 01 10 001 111 01 111 10 01 001 10 000 000 10 01 110 10 01 001 10 110 111 110 10 001 111 01 10 01 10 01 10 10 01 111 110 10 01 111 000 111 10 01 10

All character in this message a=111, b=01, c=001, d=000, e=10, f=110.

Message Size after encoding =

a(3*7)+b(2*10)+c(3*4)+d(3*3)+ e(10*15) + f(3*4) = 224 bits.

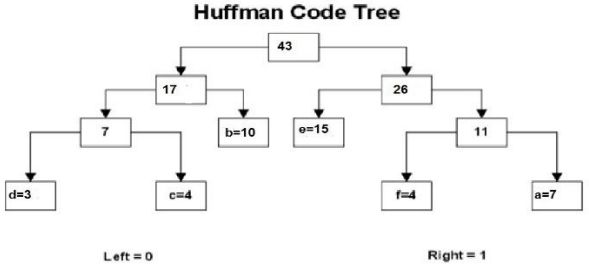


Fig 1: Huffman Code Trees

The Huffman encoding = 224 bits long. The Huffman encode tree saves: 344 - 224= 220 bits. Data compression ratio is the ratio between the uncompressed data size and compressed data size: CompRatio=UncompSize / CompSize =1.535 to 1. DataRate savings is defined as the reduction of DataRate relative to the uncompressed of DataRate saving =1- Compressed DataRate/ Uncompressed DataRate = 34.5%.

3. Design of Huffman Multiple Trees

This section describes the GAs applying for Huffman's Multi-Trees. The GAs design involves several operators: Genetic_ representation, Population initialization, Ftness_function, Selection scheme, Crossover, and Mutation. Huffman multi-trees consist adjacent nodes sequence in the graph. Hence, this is a natural choice to apply the path oriented of the encoding method. The path based crossover, the behavior and effectiveness of a genetic algorithm depend on the several parameters settings, all these parameters include the population size, crossover probability, mutation probability , generations number, where mutation range are also very popular [6]. Siding degree and elitism are used for h better individuals in selection scheme. In this paper, the multiple point crossover is applied to get its positive effect, when used them with populations to avoid creating unproductive clones for the algorithm.

In this research ,will be applied the elitism algorithm to get and prevent losing of the best solution that found, the elitism t means that instead of replacing some old individuals with the new individual, and keep set of the old individuals as long as and it will be better than the worst individuals of the next new population. Too much elitism may cause premature convergence, which is a really unpleasant consequence. To avoid this, we control applying of the elitism algorithm to select individuals as small number no more one percent of the population selection.

The chromosomes will be produced randomly from the initial population length by depending on the variety in the text file of the inserted symbols. For each character the frequency is computed and for each character the probability is found in our research and for each chromosome will build the Huffman tree in the population by depending on its probability. The code word for each gene is determined from its tree. The fitness function is computed by evaluating the chromosome using the variance of characters that depend on calculating the average for each chromosome.

Chromosome 1 :

1	1	0	0	0		1	0	1

Chromosome 2 :

0	1	0	1	0	1	0	0

Chromosomes 3 :

1	1		1	0	1		1	0	0

The fitness function is calculated depend on the individual evaluation by computing the average size, is shown in the Eq.1, 2.

$$A = \sum_{i=1}^{n} (Pi * ai) \quad , \qquad (1)$$

$$Variance = \sum_{i=1}^{n} n\ pi\ (ai - A)^2 \quad , \qquad (2)$$

Where pi: probability, ai: number of bits; A: average size.

Crossover operations generates a variety of individuals and can avoids the conflict between the genes constructing and the chromosome, and the most important property must be available in our approach, and to complete building of the Hoffman trees for each new individual, the fitness function will be evaluated after each computing of tree. In next step the new offspring will be applied to compare it with the worst individuals in the exiting population and will be exchanged with the worst individual's selection to produce the best variety in the next population.

3.1 Fitness Function

In this research there are different methods to evaluate fitness function, the byte series method uses to evaluate the bytes series data, the second is the byte frequency method to evaluate the frequency bytes, the words frequency method is to evaluate the words frequency, and the decision method is to select the best decision for each method as multi-objectives [7]. The fitness functions are applying for measuring the problems quality of each method. The genetic algorithms are evaluated using different methods to find the best results of combinations methods, and performance, also to determine the best corresponding between environmental characteristics files and their algorithms , genetic algorithms will be selected a fitness measurement each process and produced a new individuals of population,. all objectives based on its fitness function will be forced optimize using the evolution process ,the fitness function is the absolute variation value average between the compression ratio and achieved compression in the environmental characteristics files, and also the fitness of the byte frequency method can be expressed as the output of the byte frequency of file n, file n is the file in the experiment files, and the compression saving file n can be denoted as Comp(algorithm, file n) , the file compression saving n when using the compression algorithm .

3.2 Genetic Operators

In this research will be applied a multi-objective algorithm of the individuals, there are several options to apply multi-objective genetic operators to generate new individual based on multi-objective to guide evolution through the search space as follows: 1- Using a particular operator that has been selected to all models within an individual or select a different operator for each model. 2-The crossover technique will be applied between homogenous structures models, used crossing over trees at different positions in the swapping of useless material of genetic algorithm for restricting the crossover positions during evolution.3- The system will be selected an operator and a predefined probabilities for Huffman coding on characters, Huffman coding on syllables, and Huffman coding on words. 4- In the crossover will be applied only homologous data are allowed for crossing process. 5- The system will take the structural constrains as multi-objective of Huffman code trees into consideration to be ensure its syntax is implemented.

4. Proposed Compression Method

In this research, the author proposed a multi-trees representation, first tree is selected to analyze the byte series, second tree is selected to analyze the byte frequency, and a third tree is selected to analyze the words frequency, and decision method to select the best prediction for the compression ratio.

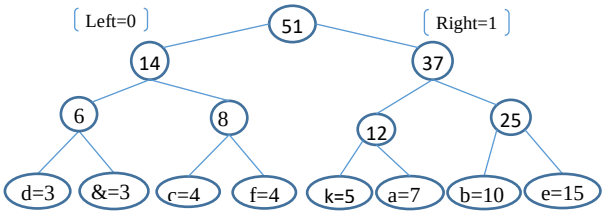
The test data compression coding and its strategy is based on compression techniques such as run length coding, statistical coding, and dictionary based coding. In this paper, the researcher evolves Huffman multi-trees and syllable-based compression of LZW using GAs to predict the compression ratio of a specific compression models to apply with different text[12], so the data representation will be depended on the file type, for example each unit for the file text will be represented a character as ASCII text, and if a file type is an executable or instructions file can be represented as numeric or textual data, the file type could be determined by file name or extensions, and after this stage , the compression model can be decided to build multi-trees representation based on decision method [9].

The proposed compression method demonstrated the optimal Huffman code trees via a genetic algorithm, the experiment determined the all possible strings were known during of processing time, the genetic algorithm applied to select the most efficient set of strings to use it in the Huffman multi-trees encoding, the bits string can be represented as a collection of variable-length for matching series of bits 0, 1, the matching vectors are assigned a prefix free code, a string can be represented as a set of matching-series coding with the values of the unspecified positions, GAs will be generated a set of multi-trees for matching strings and used it to compress the files data , the better fitness is producing optimal compression ratio .

The binary tree is the best way to visualize any particular encoding diagram, and any character can be stored at a leaf node. The character encoding is obtained by following the path from the root to its node tree, left going edge represents a 0, and right going edge a 1, as Shawn in the figures 3,4 , this diagram tree using the fixed length encoding are given in Eq. 3, 4.

$$Am = \sum_{i=1.l=0}^{n.k} (Pi * ai + Tl) \quad (3)$$

$$Variance = \sum_{i=1,l=0}^{n,k} Pi((ai + T\,l) - Am) \quad (4)$$



d=000, &=001, c=010, f=101, k=100, a=101, b=110, e=111

Fig 2: Illustrates the binary tree diagrams of the fixed-length encoding.

MassageSize=d(3*3)+&(3*3)+c(3*4)+f(3*4)+k(3*5)+a(3*7)+b(3*10)+e(3*15)=153 bits.

Frequency	d	&	C	f	K	a	B	E
Symbols	3*3	3*3	3*4	3*4	3*5	3*7	3*10	3*15
Bits	000	010	010	011	100	101	110	111

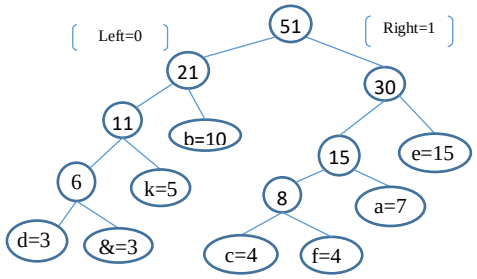


Fig 3: Illustrates the binary tree diagrams of the variable-length encoding.

Massage Size=a(4*3)+d(4*3)+ & (4*3)+c(4*4)+ f(3*4) + k(3*5)+b(2*10)+e(2*15) = 129 bits.

Frequency	d	&	c	F	k	a	b	e
Symbols	4*3	4*3	4*4	4*4	3*5	3*7	2*10	2*15
Bits	1111	1110	1000	1001	001	101	01	11

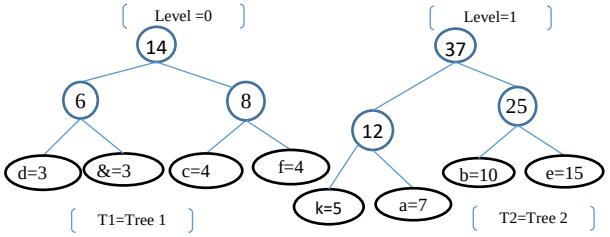


Fig 4: illustrates the Huffman multi-trees diagrams of the variable-length encoding.

T=T1+T2 + …….

MassageSize=d(3*2)+&(3*2)+c(3*4)+f(3*4)+k(2*5)+a(2*7)+b(2*10)+e(2*15)= 110 bits.

	Tree 1=36 bits				Tree 2=74 bits				
Frequency	d	&	c	f	K	a	b	e	
Symbols	3*2	3*2	3*4	3*4	2*5	2*7	2*10	2*15	110
Bits	00	01	10	11	00	01	10	11	

The different types mutation will be applied, which have named bytes' mutation, is identical to that used in genetic system, there are a set of binary nodes in Huffman multiple trees, each of its node can be a leaf or an internal node and replaced with another node, in this case can be swapped between an internal node and a leaf node as in Figures 5 and 6.

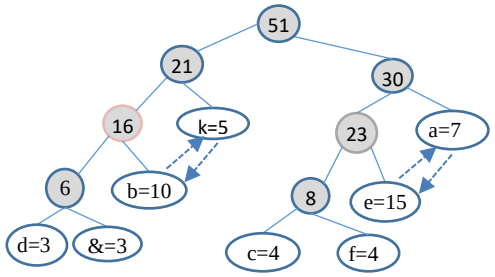


Fig 5: The swap mutation probability in genetic algorithm has been applied as a prefix-free tree.

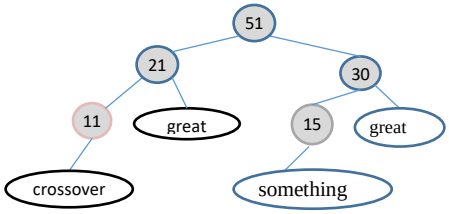


Fig 6: Illustrates a prefix free tree before the combine mutations have been implemented.

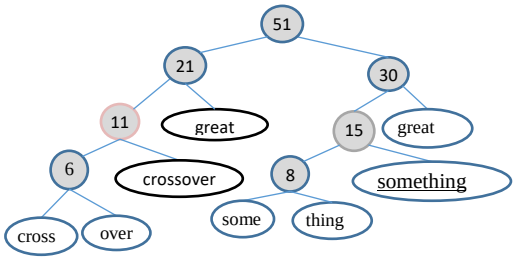


Fig 7: Illustrates the trees from figure 6 after the combine mutations have been applied, with the nodes 'cross' with 'over' and 'some' with 'thing' selected for combination.

There is another new mutation, known as join mutation to provide the functionality of creating new long string is constructed from a set of words, there are the set of symbols occur more frequently than others symbols in English text. For example the most common letters are E and T, and therefore to create an optimal Huffman code tree for the English alphabet, and encode some texts with it to apply the compression ratio and the quality of results [9].

In this part of research, they are three priorities, the system will take words in the first priority, in the second priority will select frequent syllables, and in the third priority will take individual letters, and in a typical text file, there are some words occur more than others. And to represent these words in Huffman binary code, the common words will be replaced with short letter sequences and uncommon words with longer sequences, for increasing the information entropy of file text, the compression ratio

will be more efficient, when the compressed of file text includes on frequented words with longer sequence of [11].

The sequence of frequented words are computing the probability for a new string, and randomly selected the leaf nodes, two strings are joined together into a new one string, a new node is created for containing that string. The new node is selected to be new leaf node in Huffman binary tree, the new node is created its sibling.

There is another mutation, known as words mutation to provide the functionality of swapping some words between an internal node and a leaf node in the Huffman binary tree described above.

There are randomly selected two leaf nodes for swapping, the words are created into a Huffman binary tree, and the new node is created to include that word as one string. One of the two string, randomly selected of leaf nodes, brought down a level, and the new node is made its children [14].Figures 3 and 4 are an illustration of one particular case of the word mutation.

Unlike in genetic programming; by the nature of the trees used in this algorithm, any crossover would have a significant potential to cause some strings in the tree to occur twice while others disappeared entirely. The fitness of a chromosome is determined by the length of a file, in bits, as compressed using the encoding it represents; in this experiment, therefore, fitness is to be minimized.

In this research the tree binary structure will be represented by 2n−1 bits of n internal nodes, a prefix free encoding tree will be represented in its entirety by selecting a preorder traversal of the tree as it is a full binary tree to identify each of the nodes visited is a leaf node or an internal node. The uncompressed string in the encoding has 8L bits for each, where L is the string length in bytes, characters in the code word will be calculated for the total lengths of symbols, the compressed data length is determined by looking up each encoding trees.

Consider a multicast tree is defined by G(VT,ET), where VT $\subseteq$V, ET$\subseteq$E and G$\subseteq$T, and all trees are associated with link cost C(Ti), is shown in Eq. 5.

T=T1+T2+T3 ……,Level={0, 1,2,3…k} (5)

A unicast probabilities request from the root node r to node d as to the destination node with the encoding probability bound (Δ), see Eq. 6.

$$\Delta\,(\mathrm{Pi}\,) = \sum_{e \in P_{Ti}(\mathrm{v,di})}^{n} P_{Ti} \le \Delta \qquad (6)$$

The shortest path of encoding probability is obtained by selecting the path from the root to its node to find a series of paths Pi; i$\in$ {0, 1, 2.3….}, is given Eq.7.

$$P_{Ti} = \sum_{i=1 \cdot l=0}^{n} (Pi * ai\ + l\,) \qquad (7)$$

Consider a symbol a occurs at a node n at depth k in the tree, and D array encoding of length k is given by C(a) = D = {d1,d2,d2….dk},where di $\in$ [D],all prefixes Codes of C (di) are matching to nodes on the path from root(r) to destination (di). While length max Lmax can be the longest encoded symbol, the possible value of leaves at depth lmax is D(Lmax), The code word leaf of depth i will be D(Lmax − Li) , and so Symbol_Coding C(a) = Encoding lengths E(La), see Eq. 8.

$$C(a) = \sum_{i=1}^{k} D(L_{\max\,-}\ L_i) \qquad (8)$$

In a series of graphs Gi, where i $\in$ { 0,1,.2.3….}, the value of fitness function, used to evaluate the solution quality is the tree cost among a set of candidate solutions, is given in Eq.9.

$$f(C_T) = \sum_{i=1}^{n} P_{Ti} \qquad Ti \in \mathrm{T} \qquad (9)$$

|The cost of the minimal tree will satisfy the encoding probability of constraint, the minimum cost of encoding probability, is given in Eq. 10.

$$\mathrm{C(T)}\ =\ Min\ P_T \sum_{e \epsilon P_{Ti}(\mathrm{v},d_i)}^{n} C(Ti) \qquad (10)$$

The Stability of the best solution's fitness function can be measure after calculating the standard deviation between the different solution of fitness for each process, the fitness solutions and average increasing indicates that the system produce good visible solutions in each new process [5]. The fitness value should be accurately evaluated as is quality and determined by the fitness function [10]. In the fitness algorithm for finding the minimum cost of encoding probability between the root of tree and the leaf node. the path cost is the primary criterion of solution quality between a set of candidate solutions as minimum cost of the fitness value , for example a chromosome Chi representing the path P, is denoted as F(Chj), see Eq. 11 , below.

$$\mathrm{F(Ch_i)} = \left[\sum_{\mathrm{chi} \in \mathrm{Pi(r,l)}} \mathrm{C(chi)} \right]^{-1} \qquad (11)$$

In this research the system will be started randomly to initialize the individuals population using standard genetic operators for guiding evolution during the search space related to the compression field , and construct Huffman multiple trees encoding that work with different data types and data frequencies sizes, and after several studies to get advantages of all available knowledge concerning the data compression parameters, for implementing various compression algorithms to the heterogeneous data file, will generate different compression ratios with extremely time periods consuming, the best Testing compression algorithms to determine data compression and avoid time consuming ,when the data size is less than

1GB [11], and to avoid applying a random compression algorithm that could be loosed efficiency by increasing storage space or it might even get unexpected results, the researcher applied several compression algorithm side by side with estimating and analyzing the compression ratio to be helpful for saving the computational resource and the time required for executing the compression process [12].

In this paper, the proposed algorithms presented a new compression algorithm of lossless data and used Genetic algorithm to apply the different data compressibility and compare it with different multi-trees for minimizing the file total size. When GAs was successfully implemented for constructed different Huffman's multiple trees to give an evaluation of a compression ratio, the proposed approach based on GAs and Huffman's multi-trees to implement program that use to predict data compression and high efficiency for different compression model trees, and to get a fast analysis for determining and which compression model trees is to be used minimum resources and save the time processed to run Huffman multi-trees [13].

3. Compression methods and Technical

In this research, the author supposed the word-based compression will be more suitable for large files, the syllable-based compression will be useful for middle-sized files, and the character based compression will be more suitable for short files, used syllable based compression of LZW and Human coding with genetic algorithm to compare each compression methods with their counterpart variants for words, syllables, and characters for making the best decisions to have the minimum cost of Huffman encoding probability [14].

We have created for each language a database of frequent words to improve a compression, the words from database are used for compressing algorithms initialization. As we know the English languages have a set of syllables characteristically [15]. So set of testing documents was created for English language, and we created for it decomposition algorithm into syllables one database of frequent syllables from letters, and also created databases of frequent other syllables.

For encoding length of syllables are used three Huffman trees, the first for Huffman coding trees on characters, the second for Huffman coding trees on syllables. The third for Huffman coding trees on words, all Trees are initialized from received from text documents.

Optimizing Compression Algorithms: The proposed system applied genetic algorithms to determine the most repetitive strings inside the text file to be a compressed file, the population search as contained a set of nodes that represent different strings, the Huffman trees are generated to encode higher frequency with lower references. In this work, the researchers utilized specialized search operators. The final results demonstrated that GAs was used with Huffman trees to achieve higher compression than the standard coding algorithm [15].

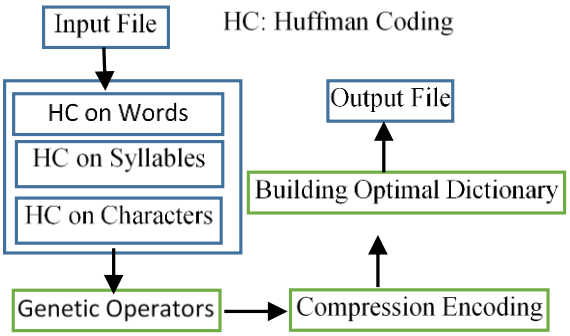


Fig 8: Illustrates the block diagram of the proposed system and its algorithms.

4. Experiments and Results

There are two main criteria in the compression methods should be taken into consideration: the compression ratio and the quality of results, the differences between the input file and the output file. GAs use a genome including a binary trees structures, while a genotype represents a particular prefix free encoding, and the initializing of population consists of a set of trees with 256 possible bytes, the experiments were implemented using the Java Genetic Algorithms Package , All experiments were performed using the following parameters settings: a population size 100-200 individuals, maximum generations 1000, a crossover probability 80%, a mutation probability 5%, and a maximum tree depth 15. The population size and number of generations dependent on type of the experimental analysis, in each mutation was applied full probability to get optimal solution. There were used for 'swap' and 'combine' mutation for each new individual.

In the first experiment, the space saving will be calculated and defined as the decreasing in data size relative to the uncompressed data size: Space Savings = 1− Compressed Size / Uncompressed Size. The best compression performs is The higher value for the space saving. In our first tests, we used the set of text files to study the compression behavior with trees including 256 possible 8bits ASCII characters, the achieved space savings were compared to ratios that can be achieved using Huffman coding tree SpaceSaving, Table 1, presents the results of applying the GAs Huffman coding based tree to all files.We performed 10 runs using four text file and let each run for 1000 total of generations, the best SpaceSaving achieved was 49%, resulting in a file size of 5055 bits using GAs Huffman coding tree comparing 49% resulting in the same file using Huffman coding tree.

Table 1. Comparison Huffman coding Tree and GAs Huffman coding Tree SpaceSaving taking into consideration effect of different files sizes, while the population size and generation number are fixed.

File	HuffTree Comp_Size	GAHuffTree Comp_Size	HuffTree SpaceSaving	GAs HuffTree SpaceSaving
File1	4211	3762	0.33	0.41
File2	4502	4122	0.36	0.42
File3	4952	4491	0.40	0.46
File4	5055	4901	0.47	0.49

In the second experiment, we applied the GAs Huffman coding algorithm in a set of the files, using different generation numbers, the file must be more than 800 generation numbers, in order to get maximum compression ratio and SpaceSaving. In this experiment, the results satisfied the probability of compression and gave good results are illustrated in Table (2).

Table 2. Illustrates a set of experiments results taking into consideration different generation number, while the population size and file sizes are fixed.

		Huff word		GAs+Huff word	
File	FileSize	CompSize	SpaseSaving	CompSize	SpaseSaving
File5	547632	301112	45.02	312621	42.91
File6	717302	395679	44.84	416992	41.87
File7	884235	524361	40.70	545112	38.35
File8	1023351	611224	40.27	639881	37.47
File9	1272142	825545	35.11	841235	33.87

In the third experiment, we developed a GAs to be able extending the standard Huffman coding to multi-trees of character encodings for getting better SpaceSaving, we used the effect of the probability of genes on the compression ratio, using GAs Huffman code multi-trees in set of the files, the increment in the probability of genes gives the gene the shortest code word and as a result increases the Space Saving of compression. In the experiment, results will be satisfied the probability of compression and gave good results are illustrated in Table (3).

Table 3. Illustrates a set of experiments results taking into consideration different generation number, while the population size and file sizes are fixed.

No	Generation Number	Population Size	Compression Size	GAsHuffTree SpaceSaving
1	200	42	83021	0.54
2	400	42	80136	0.56
3	600	42	72126	0.60
4	800	42	69141	0.62

In the Fourth experiment, we used GAs with LZWL and HuffSyllble compression algorithm side by side with estimating and analyzing the compression ratio ,the results have been increased compression ratio and getting better SpaceSaving , we used 'swap' and combine mutation to improve the probability of genes in the compression process, using GAs Huffman code multi-trees in a set of the files, the increment in the probability of genes gives the gene the good code word and as a result increases the Space Saving of compression. The experiment result was satisfied the probability of compression and gave good results are illustrated in Figure 9.

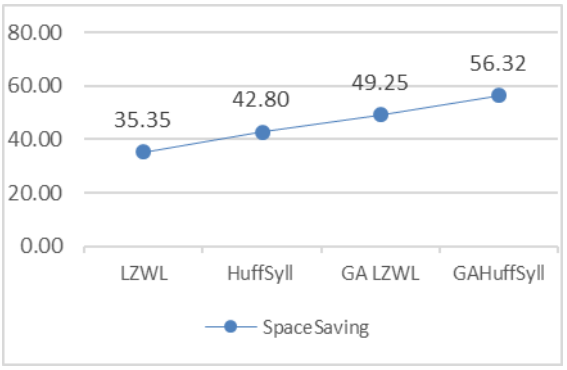


Fig 9. Illustrates a set of examples that comparing different algorithm with fixed files.

In the fifth experiment, we have to compare GA Huffman coding with WLZW word-compression algorithm, in a set of the different files sizes, while the other parameters are fixed, GAs Huffman encoding that designed as HuffWord, we compared in this test with compressor LZ77 designed as the WLZ77 Word-based compression method, this experiments were effectiveness of GAs Huffman encoding, Results has shown in figure 10. This test shown, that GAs Huffman encoding algorithm is much better than WLZ77 Word-based compression algorithm.

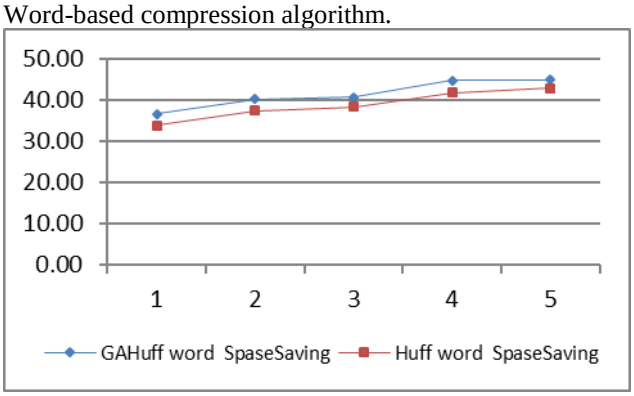


Fig 10. Illustrates a set of examples that comparing GAs Huffman coding with WLZW algorithm in a set of the different files sizes, while the other parameters are fixed.

Acknowledgment

The author would like to acknowledge the financial support of this work from the Deanship of Scientific Research, Qassim University, according to the agreement of the funded project No.1597-COC-2016-1-12-S, Qassim, Kingdom of Saudi Arabia.

References

[1] D. A. Huffman, 'A Method for the Construction of Minimum Redundancy Codes','Proceedings of the IRE,Vol.40, pp.1098-1101,1952 .

[2] Jeffrey N.Ladino,"Data Compression Algorithms", http://www.faqs.org/faqs/compression-faq/part2/section1.html.

[3] Article-Compressing and Decompressing Data using Java by Qusay H.Mahmoud with contributions from KonstantinKladko. February 2002.
[4] The Data Compression Book, 2nd edition by Mark Nelson and Jean-loup Gailly, M&T Books, New York, NY 1995 ,ISBN 1-55851-434-1.
[5] T. A. Welch, “A Technique for High -Performance Data Compression," Computer, pp. 8--18, 1984.
[6] Khalil IbrahimMohammad Abuzanouneh. “Hybrid MultiObjectives Genetic Algorithms and Immigrants Scheme for DynamicRouting Problems in Mobile Networks”. International Journal of Computer Applications 164(5): 49 -57, April 2017.
[7] Khalil IbrahimMohammad Abuzanouneh “Parallel and Distributed Genetic Algorithm with MultipleObjectives to Improve and Develop of Evolutionary Algorithm , International Journal of Advanced Computer Science and Applications, Volume7 Issue 5, 2016.
[8] L.ansky J., Zemlicka M. Compression of Small Text Files Using Syllables.Technical report no. 2006/1. KSI MFF , Praha, January 2006.
[9] U¸colu¨k G., Toroslu H.: A Genetic Algorithm Approach for Verification of the Syllable Based Text Compression Technique. Journal of Information Science, Vol. 23, No. 5, (1997) 365–372
[10] Mark.Nelson, Interactive Data Compression Tutor & The data compression book2nd Ed. by M&T books, http:// www.bham. ac.uk.
[11] LZW Data Compression by Mark Nelson, Dr. Dobb's Journal October, 1989.
[12] D. Hankerson, P. D. Johnson, and G. A. Harris, "Introduction to Information Theory and Data Compression“.
[13] Soumit Chowdhury, Amit Chowdhury, S. R. Bhadra Chaudhuri, C.T. Bhunia “Data Transmission using Online Dynami Dictionary Based Compression Technique of Fixed and Variable Length Coding” published at International Conference on Computer Science and Information Technology, 2008.
[14] P.G.Howard , J.C.Vitter, “Arithmetic Coding for Data Compression,” Proceedings of the IEEE, vol. 82, no.6, 1994, pp.857-865.
[15] Ahmed Kattan, Riccardo Poli, "Evolutionary lossless compression with GP-ZIP*," in Proceedings of the 10th annual conference on Genetic and evolutionary computation, Atlanta, Georgia, USA, 2008, 2008, pp. 1211-1218.